Single Cell RNA-Seq on Mouse Olfactory System

Alyx Gray, Chris Chua, Matt Chang

Mon Jan 25 15:17:40 2021

Here, we are analyzing P10 mouse olfactory sensory neurons (OSNs) using Seurat v4 in order to illuminate the neural map formation for olfaction. Mouse olfactory epithelial cells were dissociated at 10 days post-birth. Single cell RNA-seq was performed using the 10x Chromium v2. The raw data is available upon request from the Yu Lab at the Stower Institute for Medical Research.

This analysis was adapted from the Satija lab's Seurat - Guided Clustering Tutorial.

CellRanger Pipeline

We used the beta version of the CellRanger pipeline (v4) because that version includes code to trim adaptor and poly-A contamination. CellRanger `mkfastq` demultiplexes raw base call (BCL) files into FASTQ files. CellRanger `count` performs alignment, filtering, barcode counting, and UMI counting. It generates a raw and filter count matrix.

scRNA-seq Analysis on the Filtered Counts Matrix

Setup the Seurat Object

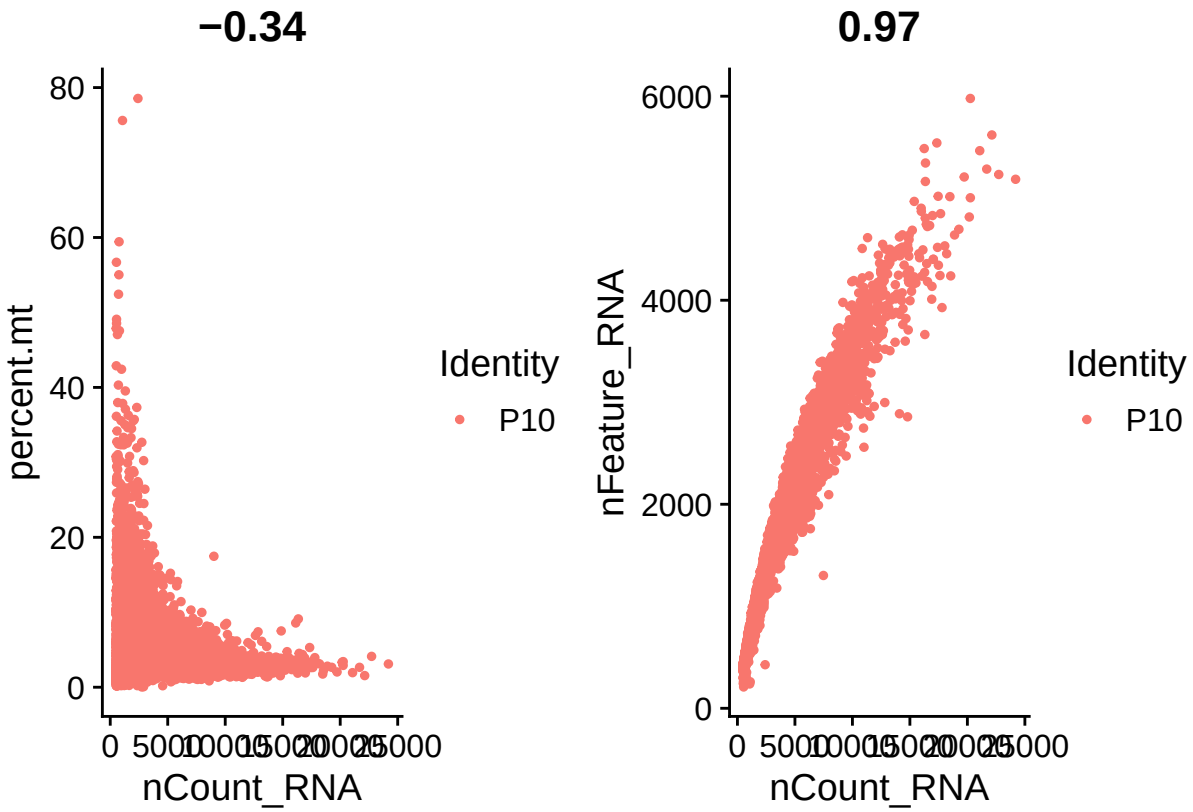
We read in the data with Seurat's `Read10x` function specifying the path to the filtered feature_bc_matrix. Then, we created a Seurat Object with the `CreateSeuratObject` function.

```
#install.packages("png")
library(dplyr)
library(Seurat)
library(patchwork)
library(ggplot2)

# Load the P10 dataset
P10_Filtered.data <- Read10X(data.dir = "/projects/bgmp/shared/groups/2020/neuron_nerds/chris/full_coun

# Initialize the Seurat object with the row (non-normalized data)
P10_Filtered <- CreateSeuratObject(counts = P10_Filtered.data, project = "P10", min.cells = 3, min.featu
```

Quality Control

The following plots describe ncount_RNA, nFeature_RNA, and percent.mt. No mitochondrial contamination cutoff was set due to minimal differences noticed between different cuttoffs. Furthermore, mitochondrial contamination is also normalized in our SCTransform step.

```
P10_Filtered[["percent.mt"]] <- PercentageFeatureSet(P10_Filtered, pattern = "^mt-")

plot1 <- FeatureScatter(P10_Filtered, feature1 = "nCount_RNA", feature2 = "percent.mt")

plot2 <- FeatureScatter(P10_Filtered, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot1 + plot2
```



Data Normalization with SCTransform

Our experiment was run on two flow cells which introduced batch effects. SCTransform is the ideal normalization function for our data due to its ability to account for this because it removes confouding sources of variation. SCTransform replaces the typical Data Normalization and Data Scaling steps in the regular tutorial.

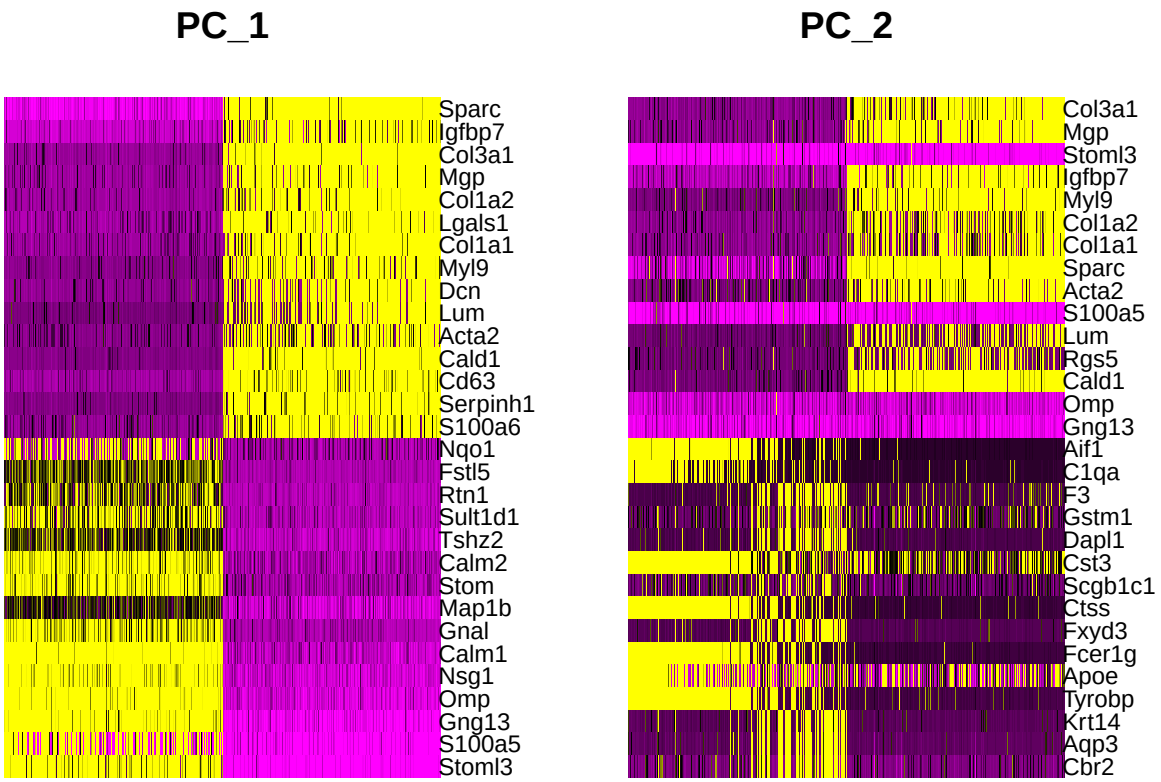
```
P10_Filtered <- SCTransform(P10_Filtered, vars.to.regress = "percent.mt", verbose = FALSE)
```

Linear Dimensional Reduction

We performed linear dimensional reduction using principal component analysis.

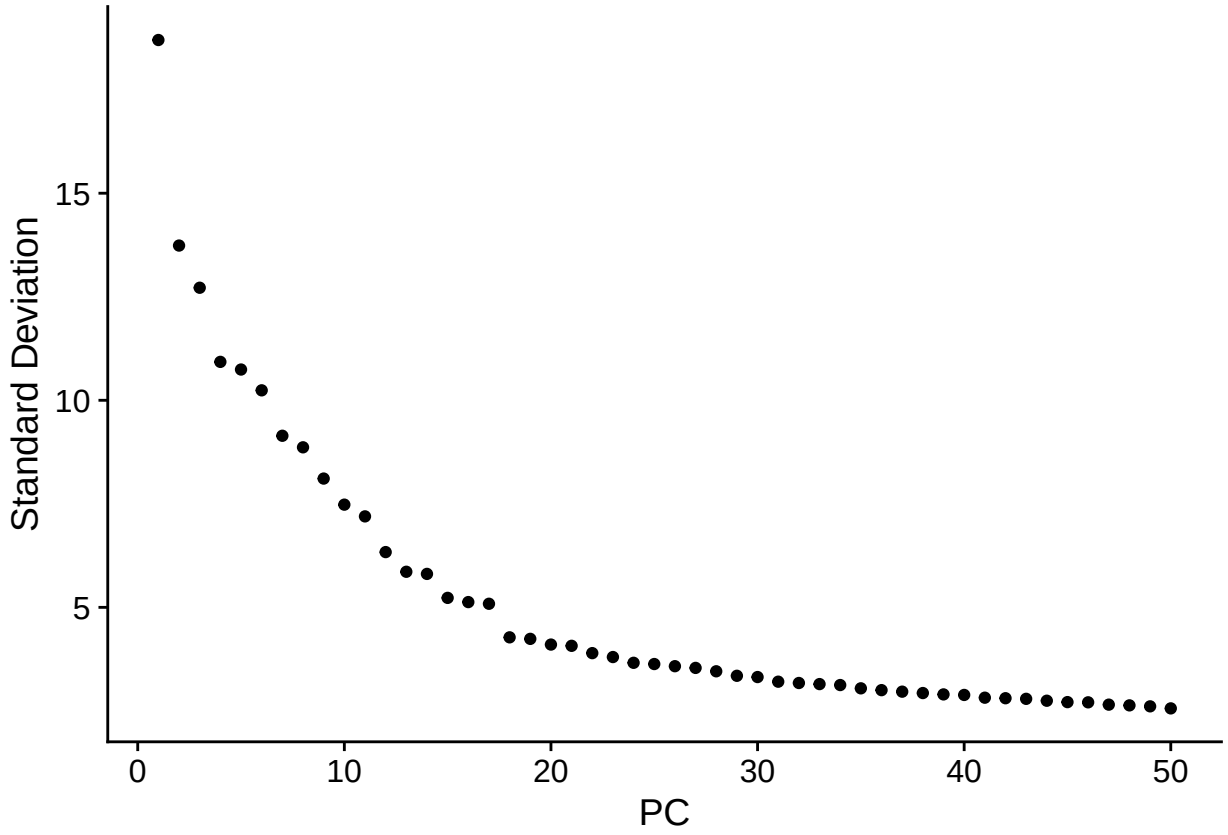
```
P10_Filtered <- RunPCA(P10_Filtered, features = VariableFeatures(object = P10_Filtered))

# Examine and visualize PCA results a few different ways
```

```
DimHeatmap(P10_Filtered, dims = 1:2, cells = 500, balanced = TRUE)
```



Determination of the Dataset's 'Dimensionality'

```
ElbowPlot(P10_Filtered, ndims = 50)
```



Cell Clustering

Resolution of 0.5 was chosen because it separated our cells into clusters that were biologically relevant.

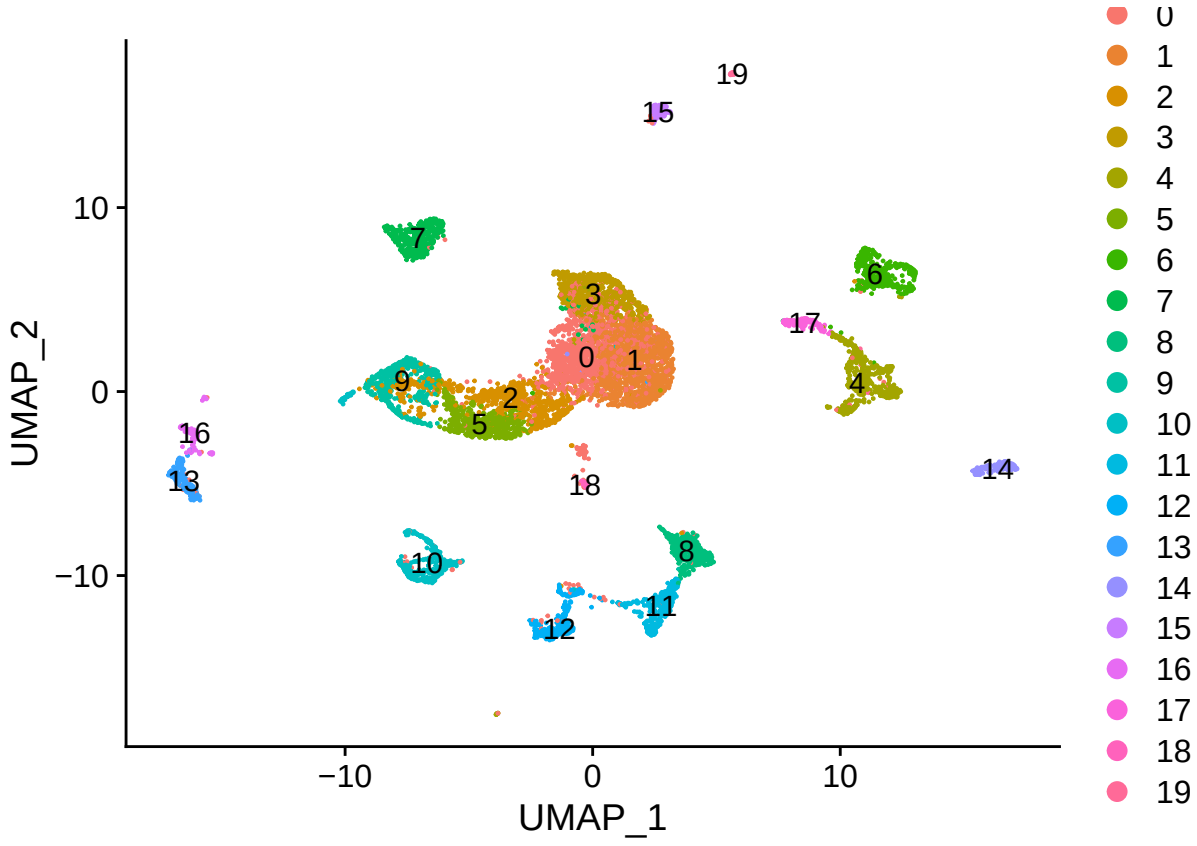
```
P10_Filtered <- FindNeighbors(P10_Filtered, dims = 1:50)
P10_Filtered <- FindClusters(P10_Filtered, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 9149
## Number of edges: 321830
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9214
## Number of communities: 20
## Elapsed time: 0 seconds
```

Non-Linear Dimensional Reduction

We ran umap to visualize the clustering of similar cells in low-dimensional space using the same number of PCs as the analysis above.

```
# reticulate::py_install(packages = 'umap-learn')
P10_Filtered <- RunUMAP(P10_Filtered, dims = 1:50)

DimPlot(P10_Filtered, reduction = "umap", label = TRUE, repel = FALSE)
```

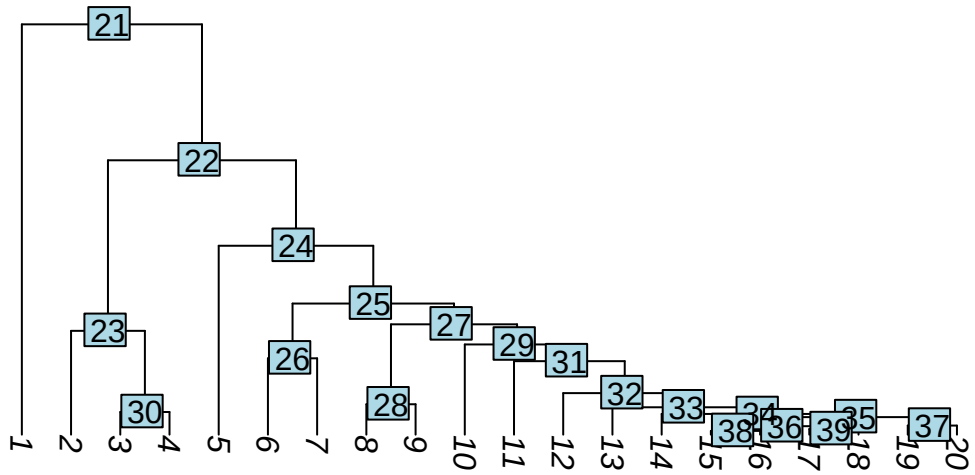



```
#saveRDS(P10_Filtered, file = "p10_filtered.rds")
```

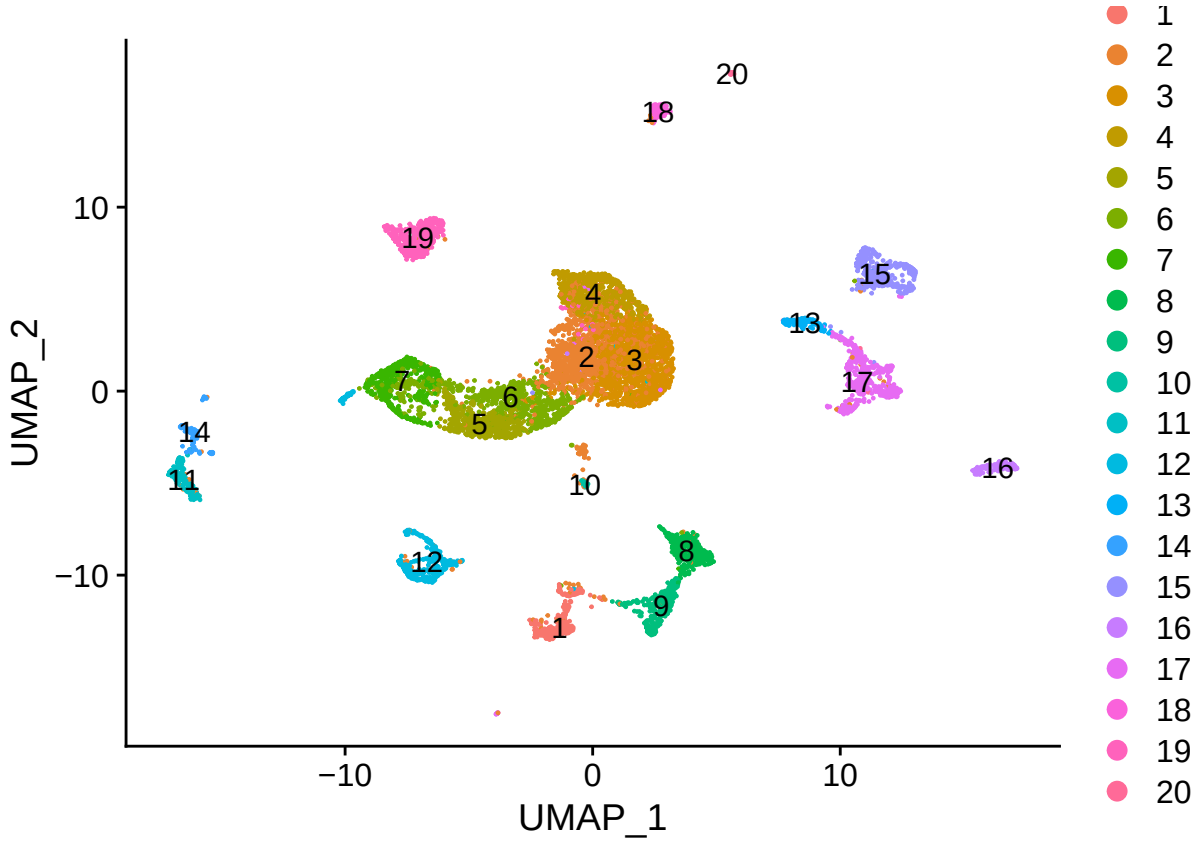
Phylogenetic Relationship Between Clusters Based on Olfactory Epithelial Markers

We reordered clusters based on their transcriptional proximity using the `BuildClusterTree` function.

```
P10_Filtered <- BuildClusterTree(P10_Filtered,
                 features = c("Omp", "Gnal", "Cnga2", "Gap43", "Gng8", "Lhx2",
                              "Neurog1", "Neurod1", "Ascl1", "Krt5", "Trp63",
                              "Krt14", "Cyp2g1", "Cyp1a2", "Coch", "Ascl3",
                              "Cftr", "Ms4a4c", "Ms4a6c", "Ms4a7", "Sox9",
                              "Trpm5"
                              ),
                 reorder = TRUE,
                 reorder.numeric = TRUE)

PlotClusterTree(P10_Filtered)
```

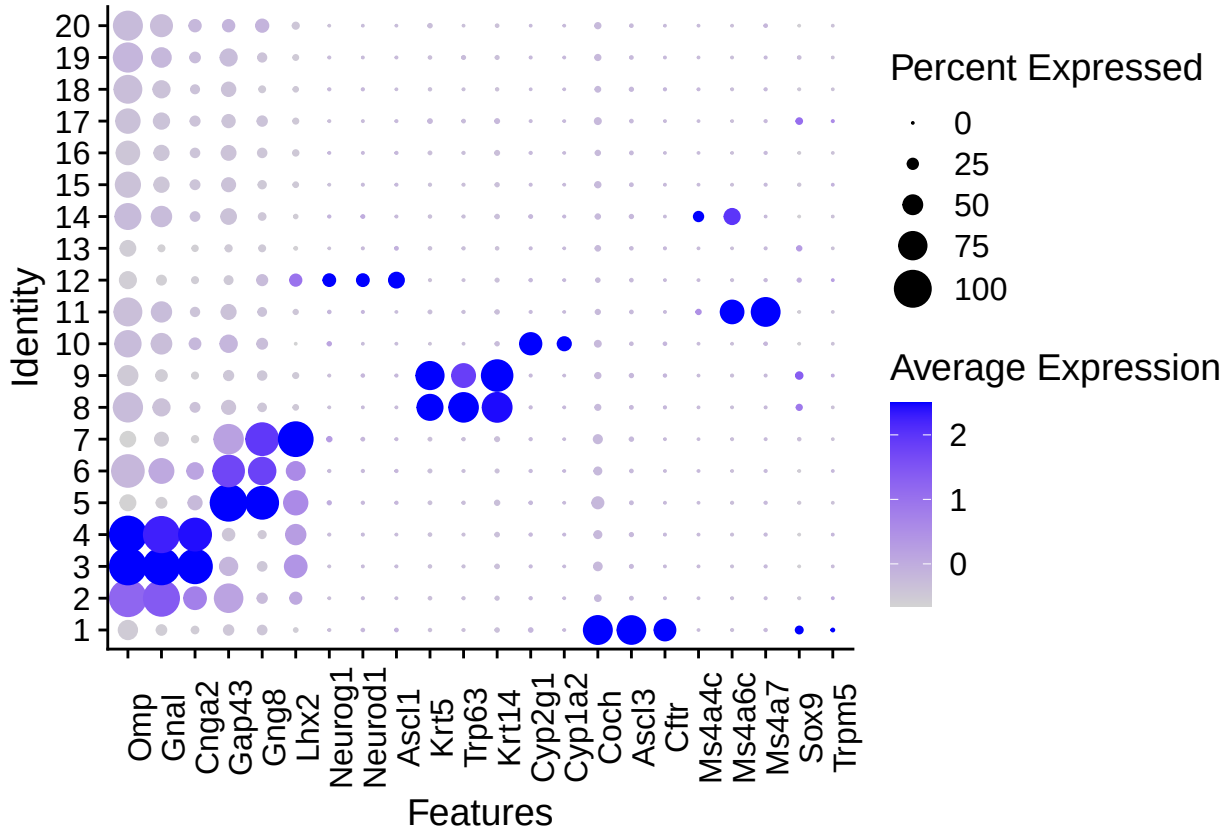



```
DimPlot(P10_Filtered, reduction = "umap", label = TRUE, repel = FALSE)
```

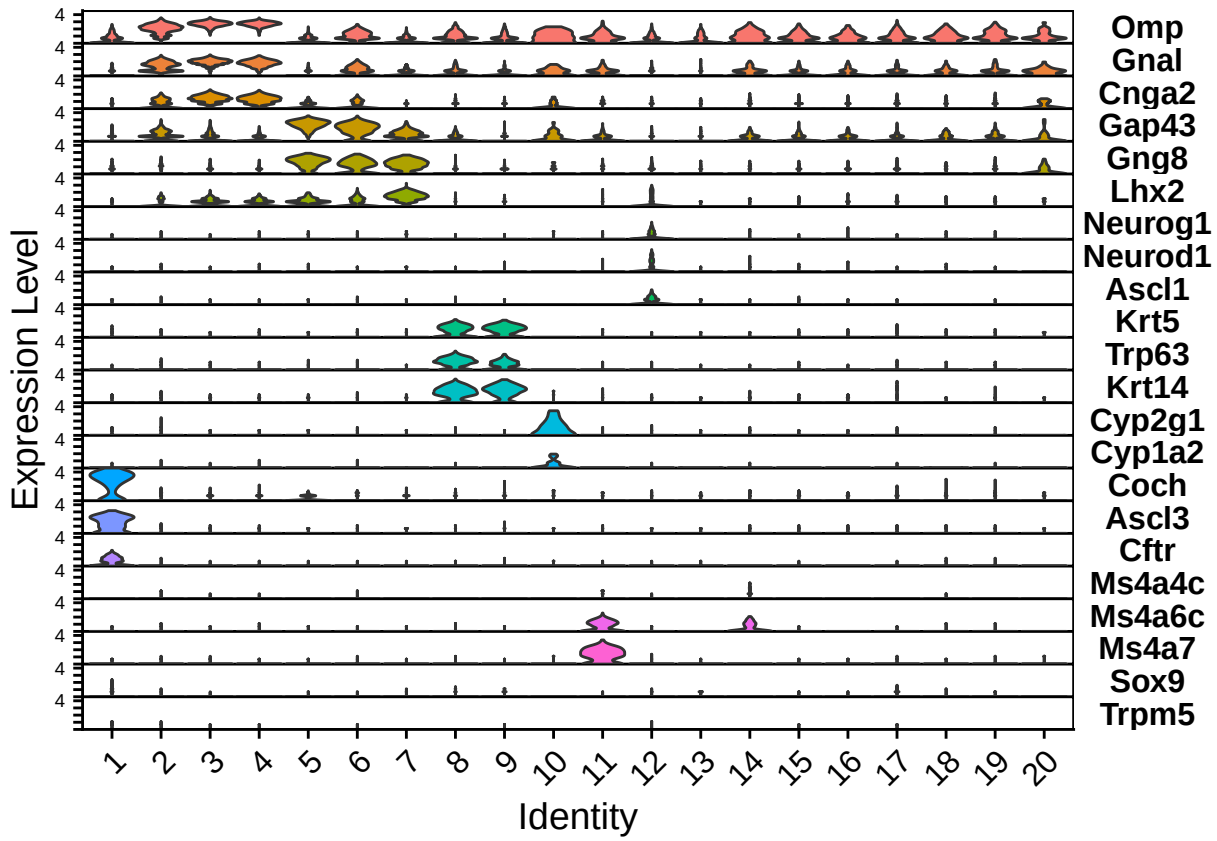


Cluster Identification Based on Expression of Marker Genes

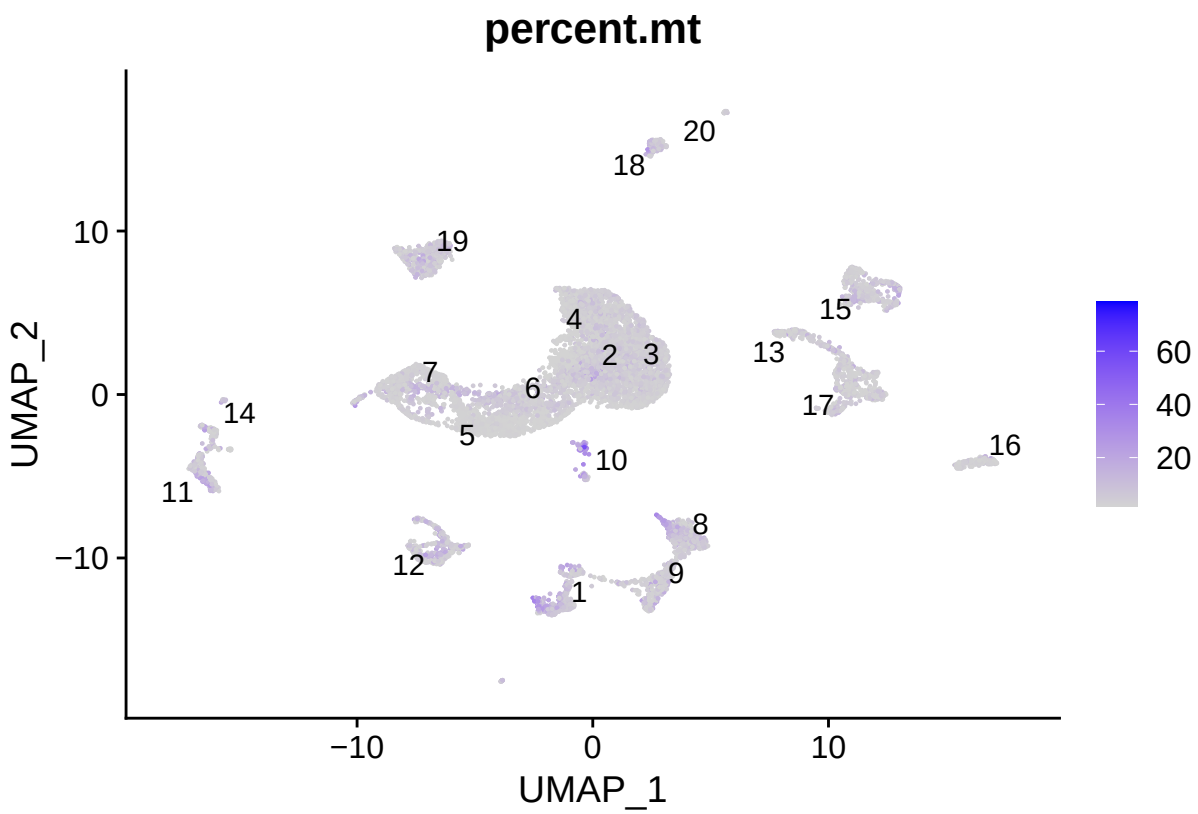
```
marker_genes <- c("Omp", "Gnal", "Cnga2", "Gap43", "Gng8", "Lhx2",
                  "Neurog1", "Neurod1", "Ascl1", "Krt5", "Trp63",
                  "Krt14", "Cyp2g1", "Cyp1a2", "Coch", "Ascl3", "Cftr",
                  "Ms4a4c", "Ms4a6c", "Ms4a7", "Sox9", "Trpm5")

DotPlot(P10_Filtered, features = marker_genes) + theme(axis.text.x = element_text(angle=90))
```




```
VlnPlot(P10_Filtered, features = marker_genes,
        stack = TRUE, same.y.lims = TRUE, pt.size = 0, flip = TRUE) + theme(legend.position = "none")
```




```
FeaturePlot(P10_Filtered, features = c("percent.mt"), label = TRUE, min.cutoff = 'q10', repel = TRUE)
```



Unknown Cluster Identification Using a Weighted Differential Gene Analysis

The top differentially expressed genes were exported to a CSV table along with the average expression. These tables were then analyzed to find the most revelent differentially expressed genes describing each cluster using the "PETE" score, which is the log fold change weighted by the transcript abundance and expression relative to other clusters ((pct.1/pct.2)*LogFold*average_expression).

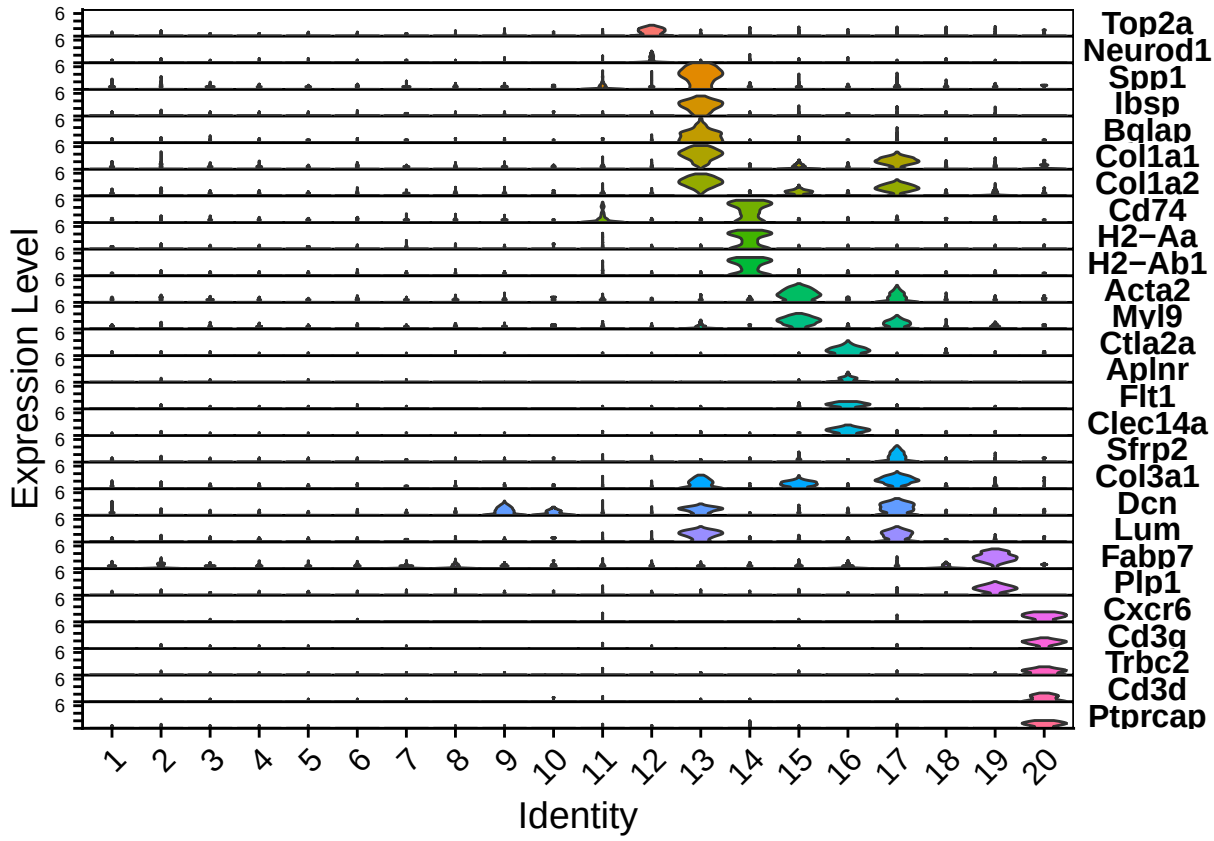
```
p10.markers <- FindAllMarkers(P10_Filtered, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)

p10.markers %>% group_by(cluster) %>% top_n(n = 2, wt = avg_logFC)
```

```
## # A tibble: 40 x 7
## # Groups:   cluster [20]
##        p_val avg_logFC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>     <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 0.            1.87  0.804 0.022 0.        1       Stap1
##  2 1.57e-261     3.03  0.766 0.136 3.07e-257 1       Coch
##  3 7.06e-256     0.768 0.992 0.877 1.38e-251 2       Gm42418
##  4 1.08e- 84     0.542 0.617 0.412 2.12e- 80 2       Ahi1
##  5 0.            1.43  0.999 0.71  0.        3       Stoml3
##  6 1.66e-157     1.15  0.643 0.363 3.25e-153 3       Nqo1
##  7 0.            2.45  1     0.937 0.        4       S100a5
##  8 0.            2.04  0.993 0.285 0.        4       Pcp4l1
##  9 0.            1.60  0.995 0.471 0.        5       Gap43
```

```
## 10 0.            1.43  0.993 0.764 0.        5       Tubb5
## # ... with 30 more rows
```

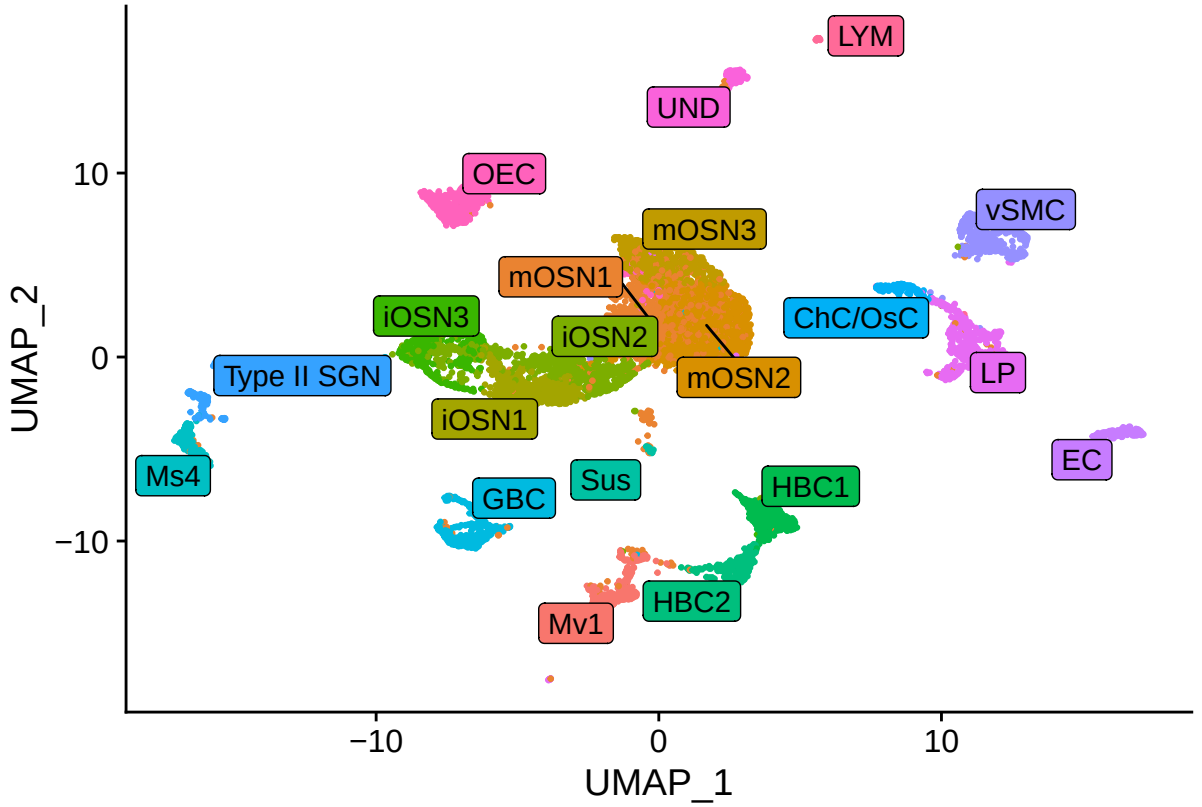
```
alltop <- p10.markers %>% group_by(cluster)

write.table(alltop, sep=",", "./topmarkers_P10_Filtered_markers.csv")
write.table(AverageExpression(P10_Filtered, assays="SCT"), sep = ",", "./topmarkers_P10_Filtered_averag
```

```
marker_genes_2 <- c("Top2a", "Neurod1",
                    "Spp1", "Ibsp", "Bglap", "Col1a1", "Col1a2",
                    "Cd74", "H2-Aa", "H2-Ab1",
                    "Acta2", "Myl9",
                    "Ctla2a", "Aplnr", "Flt1", "Clec14a",
                    "Sfrp2", "Col3a1", "Dcn", "Lum",
                    "Fabp7", "Plp1",
                    "Cxcr6", "Cd3g", "Trbc2", "Cd3d", "Ptprcap"
)

VlnPlot(P10_Filtered, features = marker_genes_2,
        stack = TRUE, same.y.lims = TRUE, pt.size = 0, flip = TRUE) + theme(legend.position = "none")
```



Cluster Annotation

```
new.cluster.ids <- c("Mv1", "mOSN1", "mOSN2", "mOSN3", "iOSN1", "iOSN2", "iOSN3", "HBC1", "HBC2", "Sus"
names(new.cluster.ids) <- levels(P10_Filtered)
P10_Filtered <- RenameIdents(P10_Filtered, new.cluster.ids)
DimPlot(P10_Filtered, reduction = "umap", label = TRUE, pt.size = 0.5, repel = TRUE, label.size = 4, lab
```



Subclustering Olfactory Epithelial Cells

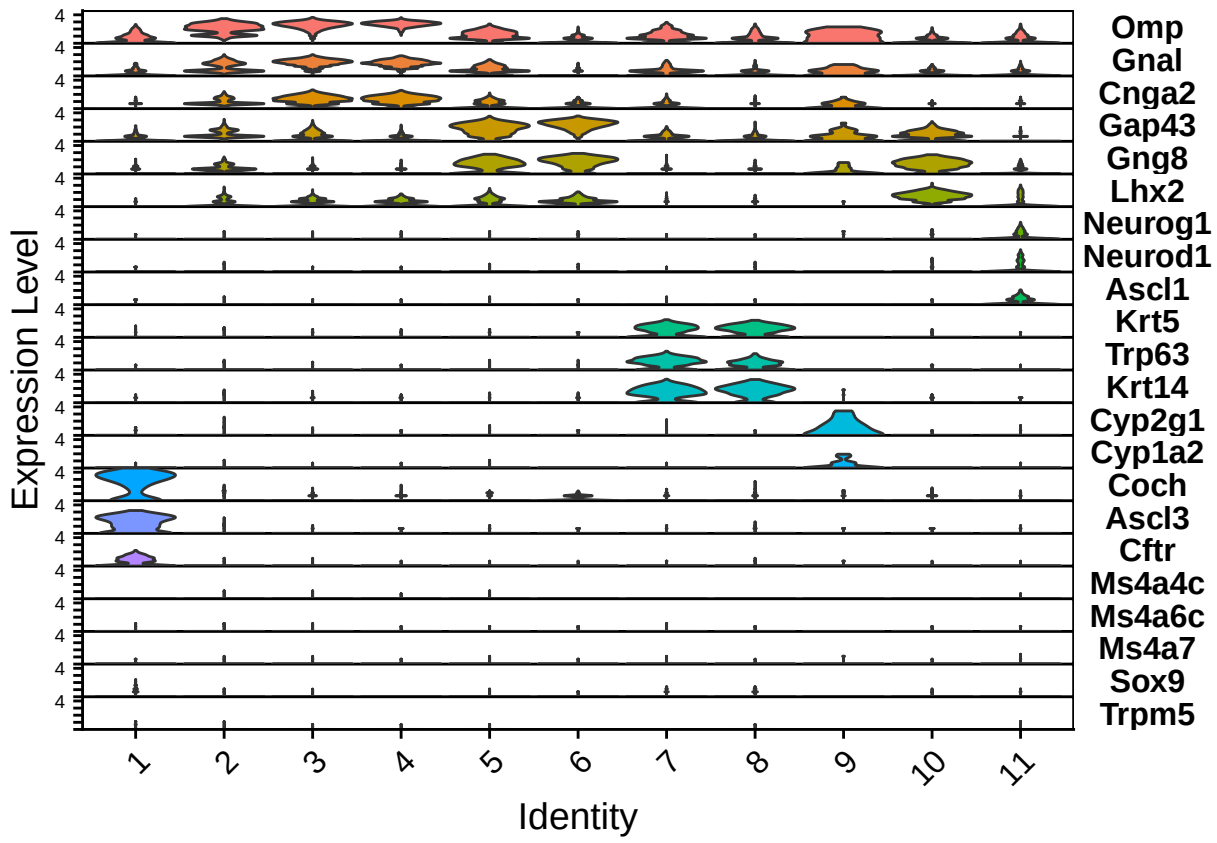
The following code chunk is run in a similar manner as before, subsetting for only clusters containing cells from the major cell types within the olfactory epithelium.

```
Focused_cells <- colnames(subset(P10_Filtered, subset = seurat_clusters == 0 | seurat_clusters == 1 | s
p10_subset_focused <- CreateSeuratObject(counts = P10_Filtered.data, project = "P10", min.cells = 3, mi
p10_subset_focused[["percent.mt"]] <- PercentageFeatureSet(p10_subset_focused, pattern = "^mt-")
p10_subset_focused <- p10_subset_focused[,Focused_cells]

# Run sctransform (Focused_cells subclustering)
p10_subset_focused <- SCTransform(p10_subset_focused, vars.to.regress = "percent.mt", verbose = FALSE)

# Perform linear dimensional reduction (focused cells subclustering)
p10_subset_focused <- RunPCA(p10_subset_focused, features = VariableFeatures(object = p10_subset_focuse

# Subclustering the focused cells
p10_subset_focused <- FindNeighbors(p10_subset_focused, dims = 1:50)
```

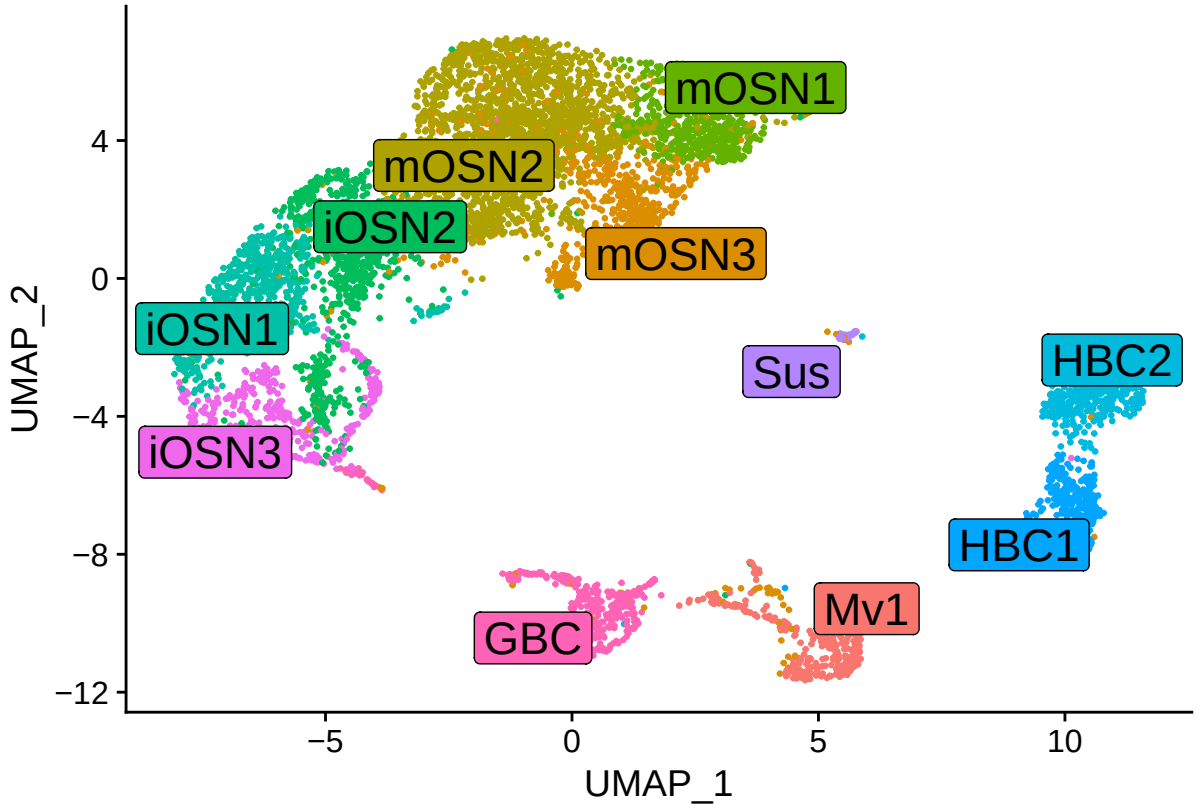
```r
# Changed resolution to 0.25 due to having fewer cells with which to analyze
p10_subset_focused <- FindClusters(p10_subset_focused, resolution = 0.25)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 6763
## Number of edges: 258324
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9225
## Number of communities: 11
## Elapsed time: 0 seconds
```

```r
# Run non-linear dimensional reduction (UMAP) (focused cells subclustering)
p10_subset_focused <- RunUMAP(p10_subset_focused, dims = 1:50)

# Reorder clusters based on their transcriptional proximity
p10_subset_focused <- BuildClusterTree(p10_subset_focused,
                features = c("Omp", "Gnal", "Cnga2", "Gap43", "Gng8", "Lhx2",
                             "Neurog1", "Neurod1", "Ascl1", "Krt5", "Trp63",
                            "Krt14",
                             "Cyp2g1", "Cyp1a2", "Coch", "Ascl3", "Cftr",
                             "Ms4a4c", "Ms4a6c", "Ms4a7", "Sox9", "Trpm5"),
                reorder = TRUE,
                reorder.numeric = TRUE)
marker_genes_subset <- c("Omp", "Gnal", "Cnga2", "Gap43", "Gng8", "Lhx2",
                             "Neurog1", "Neurod1", "Ascl1", "Krt5", "Trp63",
                         "Krt14",
                             "Cyp2g1", "Cyp1a2", "Coch", "Ascl3", "Cftr",
                             "Ms4a4c", "Ms4a6c", "Ms4a7", "Sox9", "Trpm5")
VlnPlot(p10_subset_focused, features = marker_genes_subset,
        stack = TRUE, same.y.lims = TRUE, pt.size = 0, flip = TRUE) + theme(legend.position = "none")
```

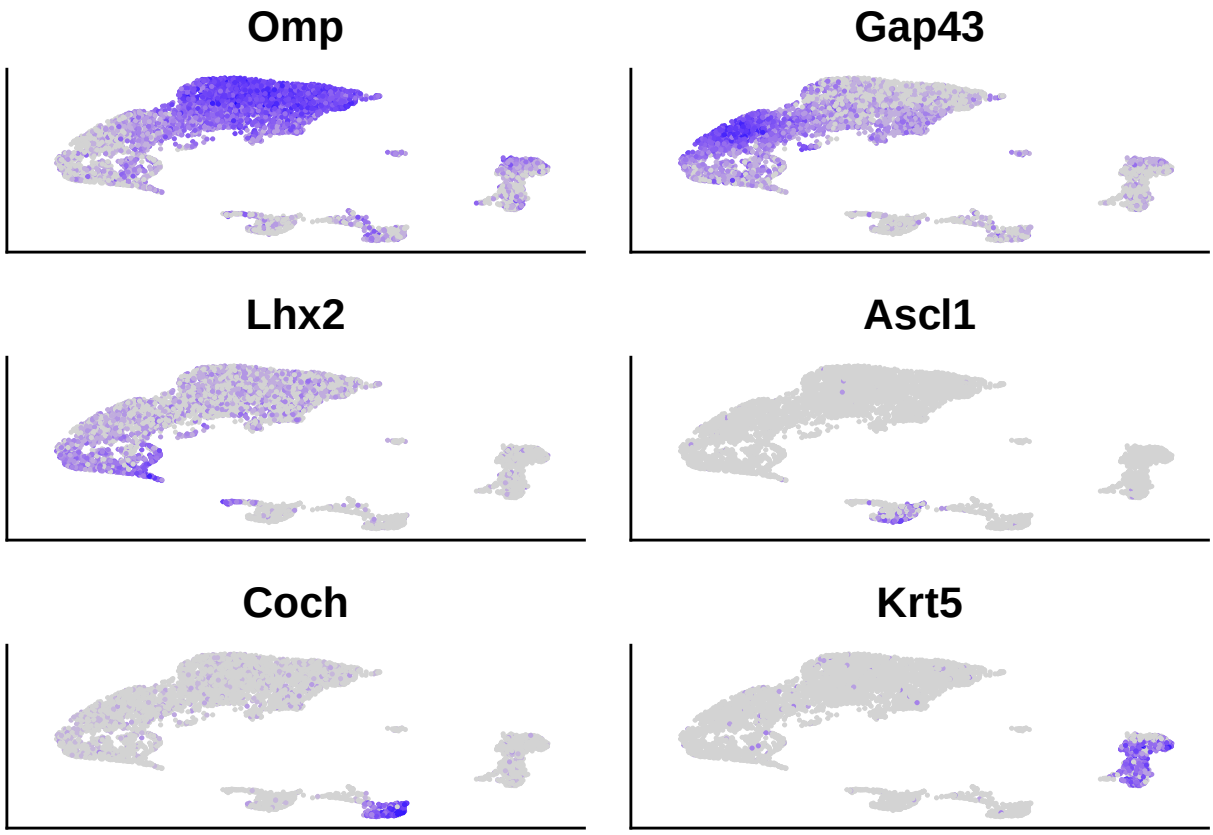



```
new.cluster.ids <- c("Mv1", "mOSN3", "mOSN2", "mOSN1", "iOSN2", "iOSN1", "HBC2", "HBC1", "Sus", "iOSN3"
names(new.cluster.ids) <- levels(p10_subset_focused)
p10_subset_focused <- RenameIdents(p10_subset_focused, new.cluster.ids)

# Figure 5A
DimPlot(p10_subset_focused, reduction = "umap", label = TRUE, pt.size = 0.5, repel = TRUE, label.size =
```



Composite FeaturePlot for Poster (Figure 5B) to Demonstrate Annotation Method

```
#install.packages("ggpubr")
library(ggpubr)
a <- FeaturePlot(p10_subset_focused, features = "Omp") + NoLegend() + theme(axis.text=element_blank(), a

b <- FeaturePlot(p10_subset_focused, features = "Gap43") + NoLegend() + theme(axis.text=element_blank()

c <- FeaturePlot(p10_subset_focused, features = "Lhx2") + NoLegend() + theme(axis.text=element_blank(),

d <- FeaturePlot(p10_subset_focused, features = "Ascl1") + NoLegend() + theme(axis.text=element_blank()

e <- FeaturePlot(p10_subset_focused, features = "Coch") + NoLegend() + theme(axis.text=element_blank(),

f <- FeaturePlot(p10_subset_focused, features = "Krt5") + NoLegend() + theme(axis.text=element_blank(),

figure <- ggarrange(a, b, c, d, e, f, ncol=2, nrow=3)  +ggsave("FeaturePlotsCombined.png", height=5.5, w

figure
```



Subclustering mOSNs

Repeat same steps as subclustering for olfactory epithelium cell types. This time, use only clusters from the original dataset containing mOSNs.

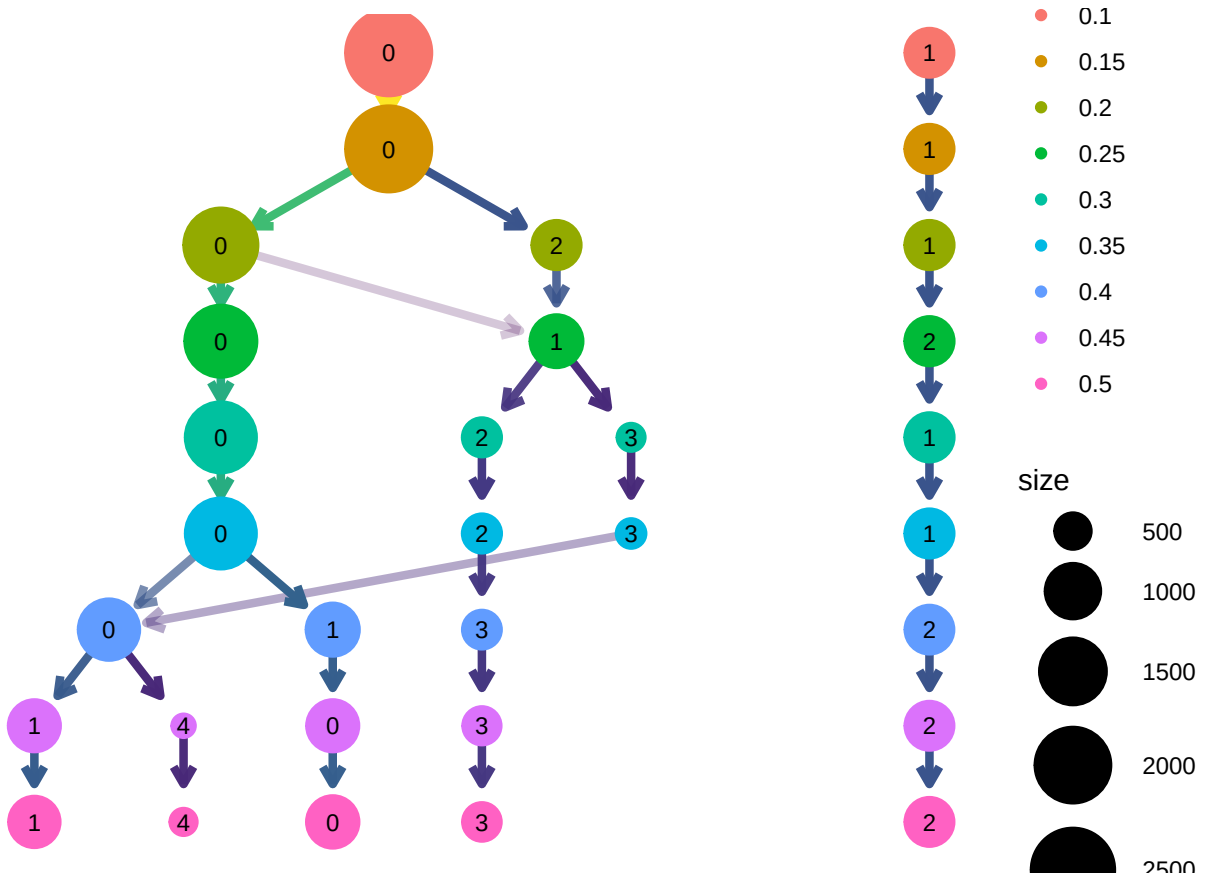
```
mOSN_cells <- colnames(subset(P10_Filtered, subset = seurat_clusters == 0 | seurat_clusters == 1 | seura

# Initialize the Seurat object with the row (non-normalized data)
p10_subset_mOSN <- CreateSeuratObject(counts = P10_Filtered.data, project = "P10", min.cells = 3, min.fe
p10_subset_mOSN <- p10_subset_mOSN[, mOSN_cells]

# QC Analysis (mOSN subclustering)
p10_subset_mOSN[["percent.mt"]] <- PercentageFeatureSet(p10_subset_mOSN, pattern = "^mt-")

# Run sctransform (mOSN subclustering)
p10_subset_mOSN <- SCTransform(p10_subset_mOSN, vars.to.regress = "percent.mt", verbose=FALSE)

# Perform linear dimensional reduction (mOSN subclustering)
p10_subset_mOSN <- RunPCA(p10_subset_mOSN, features = VariableFeatures(object = p10_subset_mOSN), npcs =

# Subclustering the mOSNs
p10_subset_ct <- FindNeighbors(p10_subset_mOSN, dims = 1:50)
p10_subset_ct <- FindClusters(
  p10_subset_ct, resolution=seq(from=0.1, to=.5, by=0.05, verbose=FALSE)
```

```
)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9327
## Number of communities: 2
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9057
## Number of communities: 2
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8804
## Number of communities: 3
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8618
## Number of communities: 3
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8438
## Number of communities: 4
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
```

```
## Maximum modularity in 10 random starts: 0.8269
## Number of communities: 4
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8110
## Number of communities: 4
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7971
## Number of communities: 5
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7847
## Number of communities: 5
## Elapsed time: 0 seconds
```

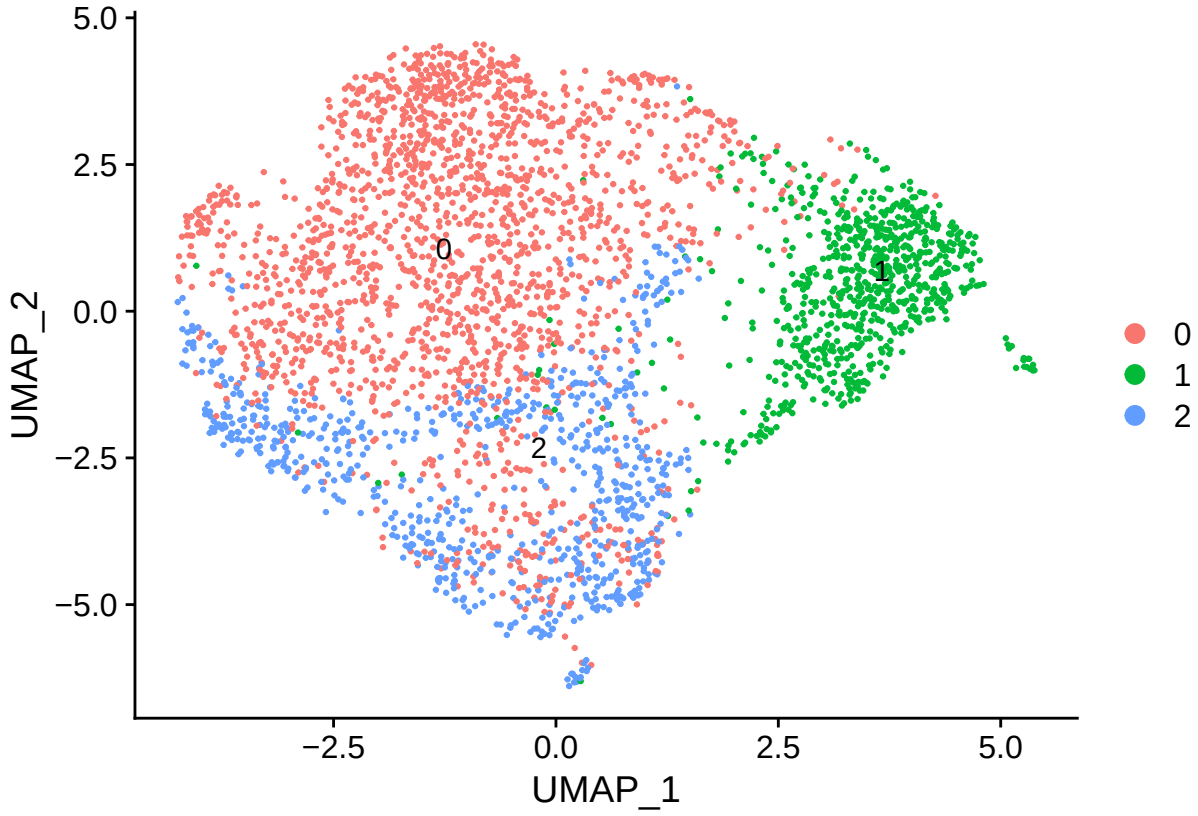
```
# Use library clustree to determine optimal resolution to avoid over or under clustering our data
library(clustree)
cluster_seurat <- p10_subset_ct@meta.data %>%
  dplyr::select(dplyr::contains("RNA_snn_res."))
clustree::clustree(p10_subset_ct, prefix="SCT_snn_res.")
```



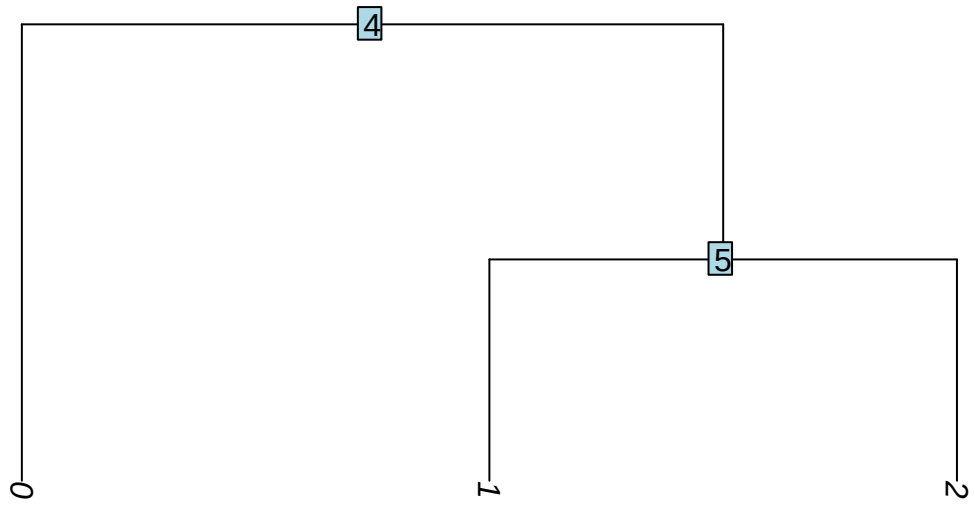

```
p10_subset_mOSN <- FindNeighbors(p10_subset_mOSN, dims = 1:50)
p10_subset_mOSN <- FindClusters(p10_subset_mOSN, resolution = 0.2)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3474
## Number of edges: 192438
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8804
## Number of communities: 3
## Elapsed time: 0 seconds
```

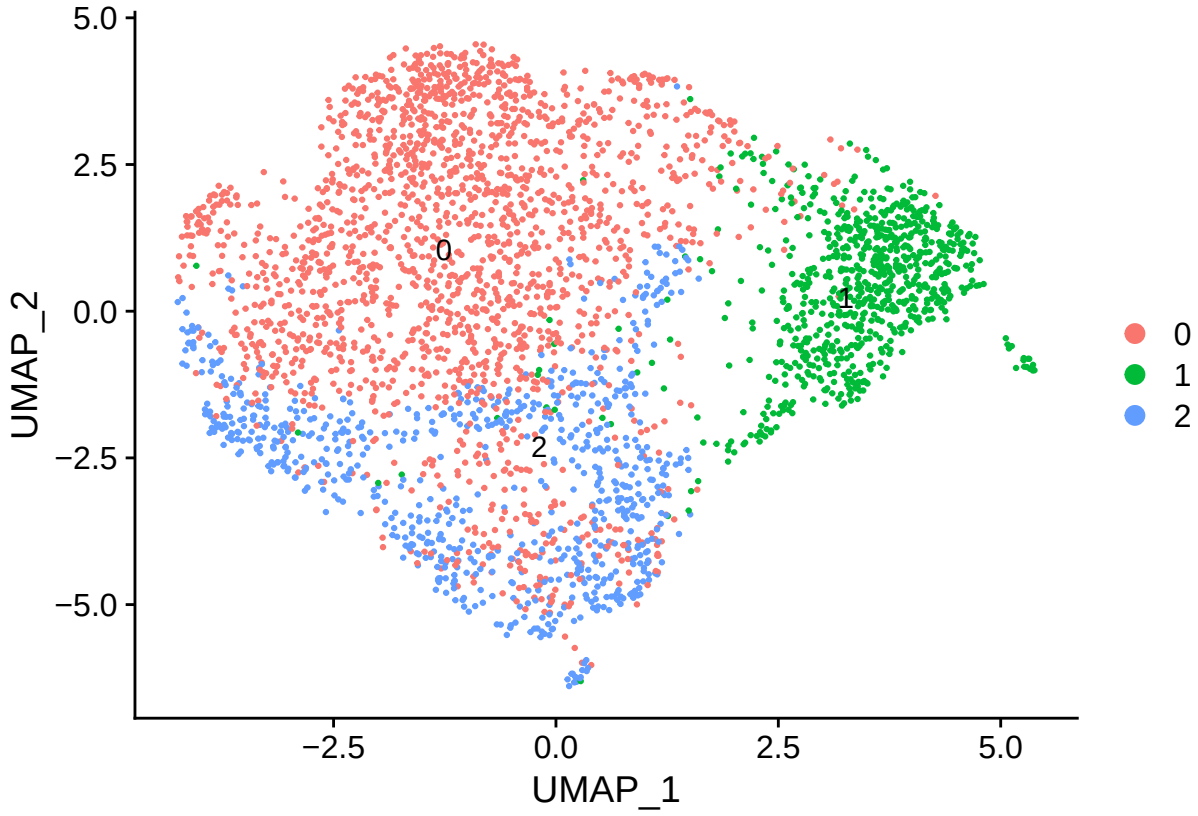
```
# Run non-linear dimensional reduction (UMAP/tSNE) (mOSN subclustering)
p10_subset_mOSN <- RunUMAP(p10_subset_mOSN, dims = 1:50)
DimPlot(p10_subset_mOSN, reduction = "umap", label = TRUE, repel = TRUE)
```



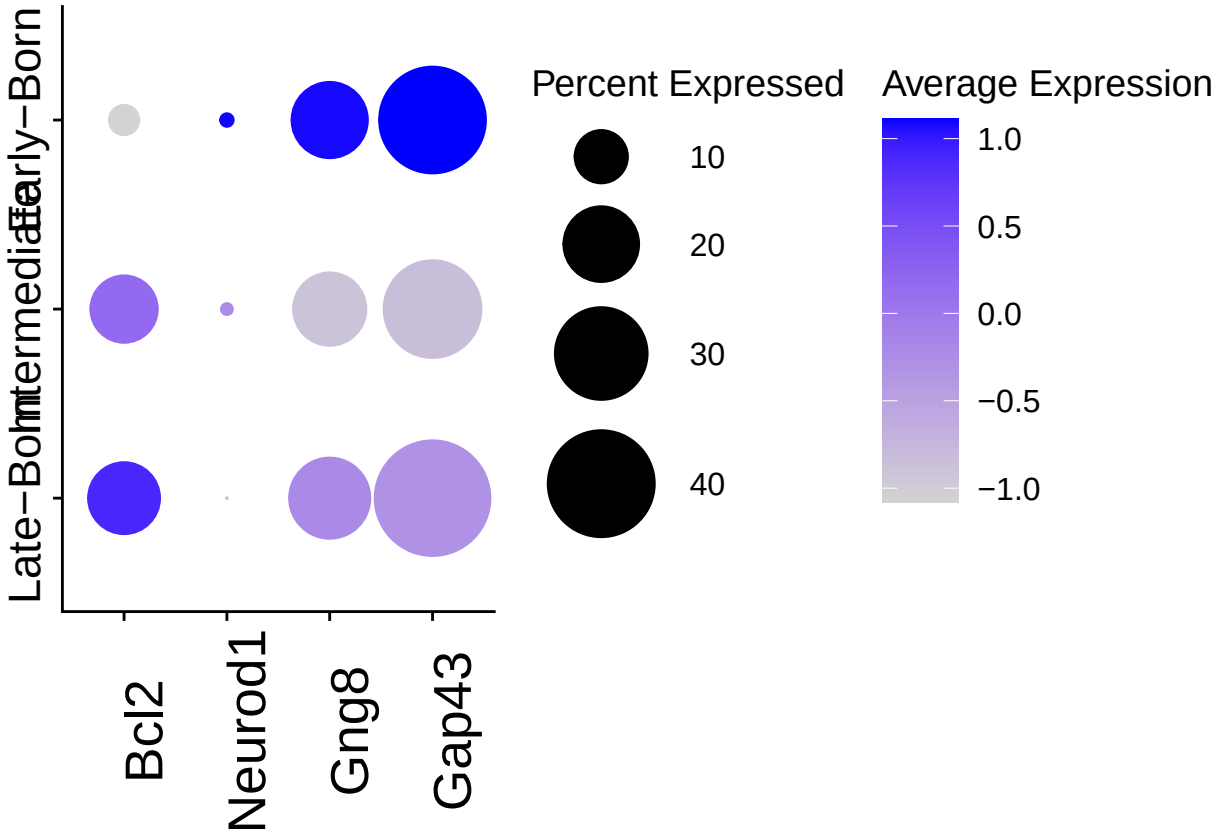

```
# Reorder clusters based on their transcriptional proximity
p10_subset_mOSN <- BuildClusterTree(p10_subset_mOSN,
                  features = c("Bcl2", "Malat1", "Gap43", "Gng8", "Lhx2",
                               "Neurod1"),
                  reorder = FALSE,
                  reorder.numeric = FALSE)
PlotClusterTree(p10_subset_mOSN)
```



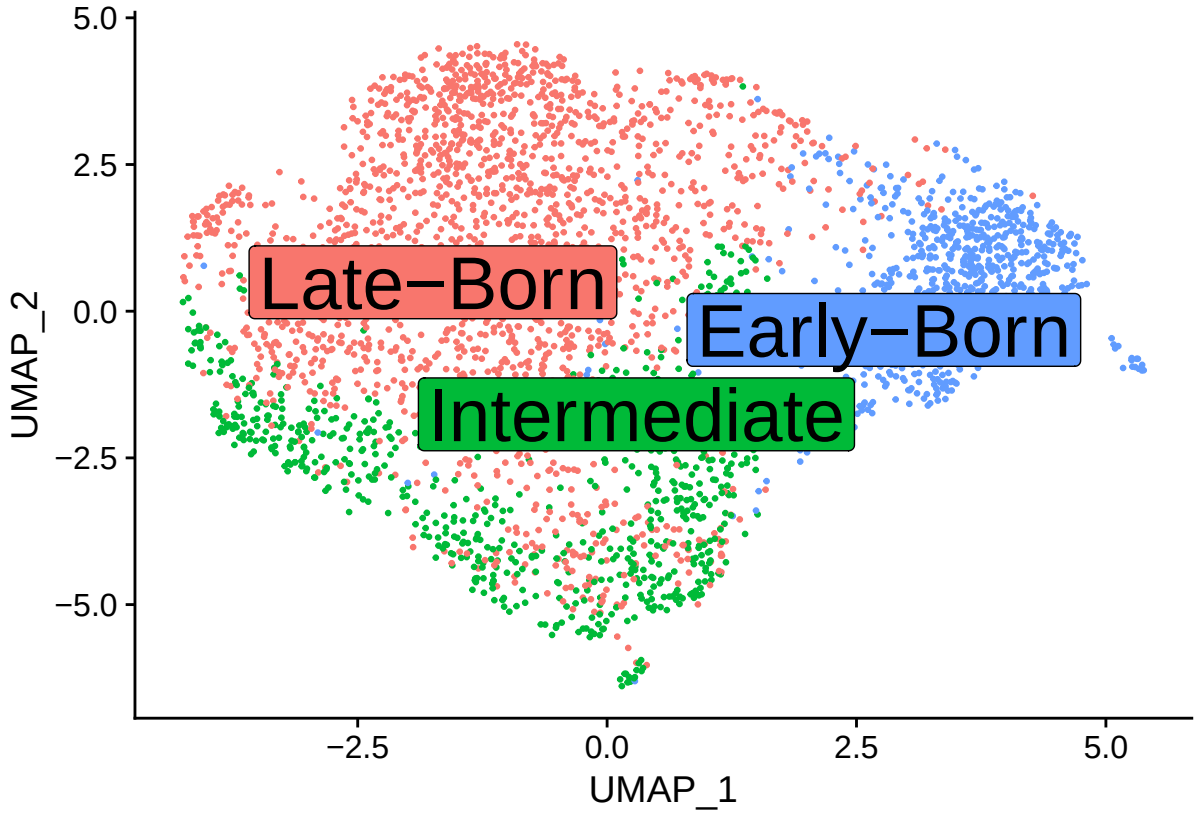

```
DimPlot(p10_subset_mOSN, reduction = "umap", label = TRUE, repel = TRUE)
```




```
marker_genes_subset_mOSN <- c("Bcl2","Neurod1", "Gng8", "Gap43")

# Reorder levels to make DotPlot in correct order
levels(p10_subset_mOSN) <- c("0", "2", "1")
new.cluster.ids <- c("Late-Born", "Intermediate", "Early-Born")
names(new.cluster.ids) <- levels(p10_subset_mOSN)
p10_subset_mOSN <- RenameIdents(p10_subset_mOSN, new.cluster.ids)

# DotPlot used to determine cluster identity. Theme() arguments used to make the graph presentable for
DotPlot(p10_subset_mOSN, dot.scale = 20, scale.by = "size", features = marker_genes_subset_mOSN, cluste
```




```
# Figure 5c Code
DimPlot(p10_subset_mOSN, reduction = "umap", label = TRUE, pt.size = 0.5, repel = TRUE, label.size = 10
```


Late−Born
Early−Born
Intermediate
UMAP_2
UMAP_1
5.0
2.5
0.0
−2.5
−5.0
−2.5
0.0
2.5
5.0