Individual dataset processing

Ricardo Martins-Ferreira

2023-06-28

Example for the Grubman et al. 2019 (PMID: 31768052)

```
libs <- c("Seurat", "tidyverse", "stringr", "data.table", "readxl")
suppressMessages(
  suppressWarnings(sapply(libs, require, character.only = TRUE))
)
```

```
##     Seurat  tidyverse    stringr data.table     readxl
##       TRUE       TRUE       TRUE       TRUE       TRUE
```

The data consists of snRNA-seq from autopsied entorhinal cortex from patients with Alzheirme's Disease and controls.

It is available at NCBI's Gene Expression Omnibus database under the accession code GSE138852.

The input data consists of a processed count matrix 13,214 nuclei and 10,850 genes.

Create Seurat object from count matrix

```
# upload count matrix
count.matrix <- read.csv("~/windows_ricardo/Desktop/rstudio_data/HuMicA/Public_datasets/Grubman_2019/In

# Create Seurat object acounting for genes present in at least 3 nuclei.
Seurat = CreateSeuratObject(counts =count.matrix, min.cells=3)
```

Filter low quality nuclei

Remove nuclei with the following features:

- Feature count lower than 200 and higher than 5000;
- UMI count lower than 500 and higher than 20000;
- Percentage of mithocondrial genes higher than 20%;
- Percentage of ribosomal genes higher than 5%.

```
# Calculate percentage of ribosomal genes
C<-GetAssayData(object = Seurat, slot = "counts")
rb.genes <- rownames(Seurat)[grep("^RP[SL]",rownames(Seurat))]
percent.ribo <- colSums(C[rb.genes,])/Matrix::colSums(C)*100
Seurat <- AddMetaData(Seurat, percent.ribo, col.name = "percent.ribo")

# Calculate percentage of ribosomal genes
Seurat[["percent.mt"]] <- PercentageFeatureSet(Seurat, pattern = "^MT-")


# Filter low quality nuclei
Seurat <- subset(Seurat, subset = nFeature_RNA > 200 & nFeature_RNA < 5000
                 & nCount_RNA > 500 & nCount_RNA < 20000
                 & percent.mt < 10 & percent.ribo < 5)

Seurat
```

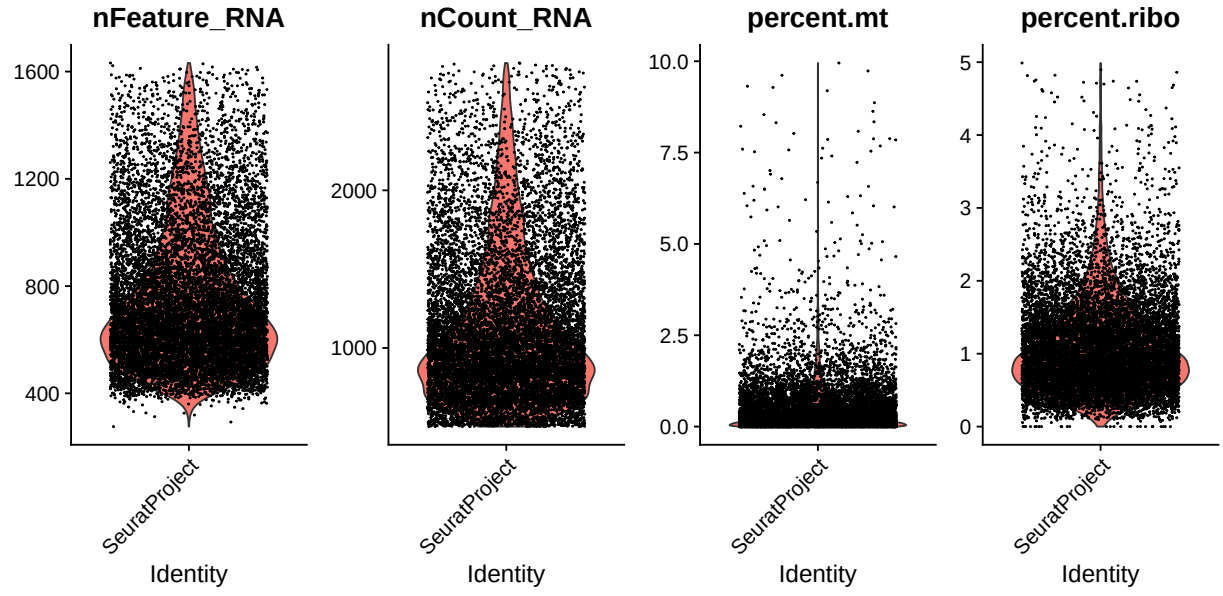
```
## An object of class Seurat
## 10850 features across 11440 samples within 1 assay
## Active assay: RNA (10850 features, 0 variable features)
```

```
#check mean genes per nuclei
mean(Seurat@meta.data$nFeature_RNA)
```

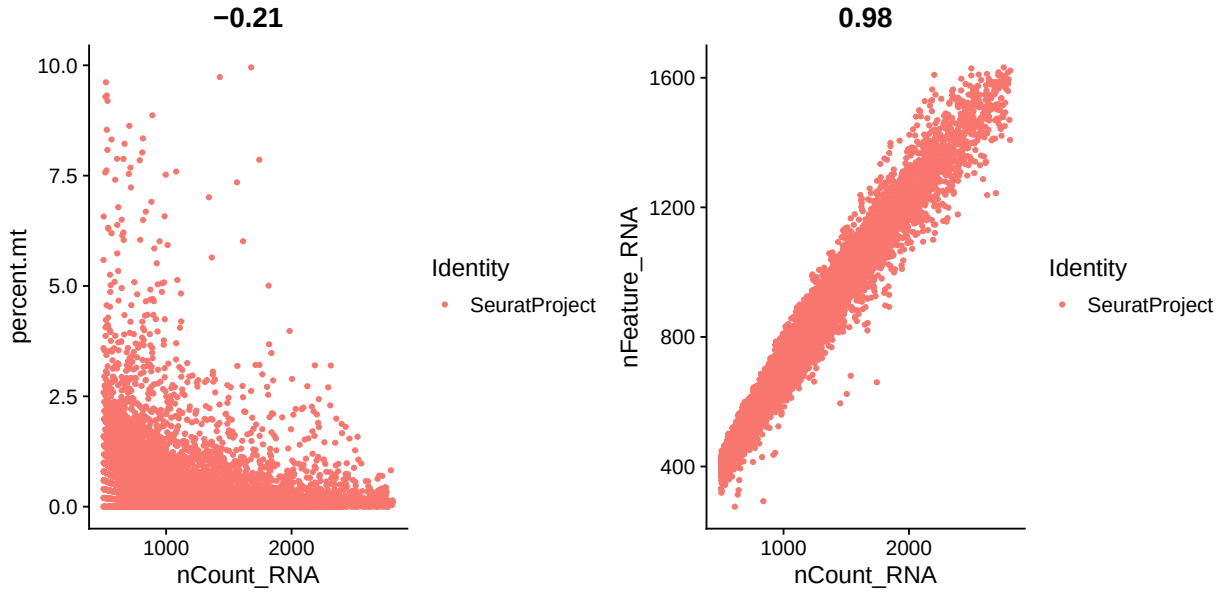
```
## [1] 756.6617
```

A Seurat object with 11,440 nuclei, 10,850 genes and approximately 757 genes per nuclei.

```
# Visualize QC metrics as a violin plot
print(VlnPlot(Seurat, features = c("nFeature_RNA", "nCount_RNA",
                                   "percent.mt","percent.ribo"), ncol = 4))
```



```
#FeatureScatter is typically used to visualize feature-feature relationships
plot1 = FeatureScatter(Seurat, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 = FeatureScatter(Seurat, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")

print(plot1 + plot2)
```



Annotate nuclei by Subject

```
#the name of each sample is merged to each nuclei's annotation
Seurat@meta.data$TAG <- rownames(Seurat@meta.data)

#Split the sample name from the full nuclei name
x<-as.data.frame(str_split_fixed(Seurat@meta.data$TAG,"_",2))
Seurat@meta.data$Subject <- x$V2

#remove percent.ribo and percent.mt columns fro metadata
Seurat@meta.data <- Seurat@meta.data[,-c(4,5)]
```

Filter out genes associated with postmortem

Remove the list of genes associated with postmortem interval in bulk RNA-seq in cerebral cortex (Zhu et. al 2017; PMID: 28710439)

```
PMI_genes <- read_csv("~/windows_ricardo/Desktop/rstudio_data/HuMicA/Public_datasets/PMI-genes.csv",
                      col_names = FALSE)$X1 %>% as.vector()

keep= c(!rownames(Seurat) %in% c(PMI_genes))
Seurat<- subset(x = Seurat,features =c(1:(dim(Seurat)[1]))[keep])
```

```
Seurat
```

```
## An object of class Seurat
## 10850 features across 11440 samples within 1 assay
## Active assay: RNA (10850 features, 0 variable features)
```

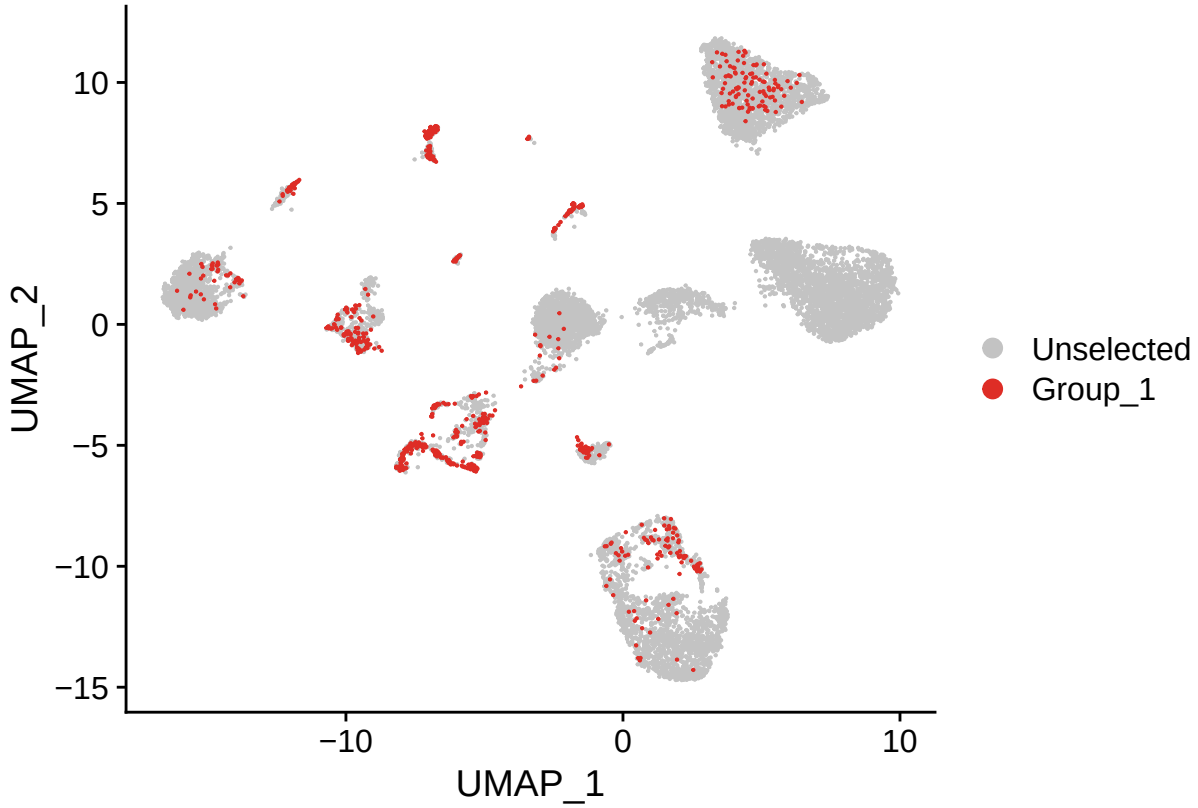
```
#check mean genes per nuclei
mean(Seurat@meta.data$nFeature_RNA)
```

```
## [1] 756.6617
```

Doublet removal

Doublets (n=4074) had already been calculated using DoubletFinder.

```
#upload doublet nuclei
list <- list.files(pattern = ".csv", path ="~/windows_ricardo/Desktop/rstudio_data/HuMicA/Public_datase
                   ,full.names =TRUE)

named.list <- lapply(list, read.csv)
doublets <- do.call(rbind, named.list)

#Plot doublets (normalization and dimensionality reduction are required)
Seurat <- SCTransform(Seurat, verbose = FALSE) #defaul variable features = 3000
Seurat <- RunPCA(Seurat, verbose = FALSE)
Seurat <- RunUMAP(Seurat,  reduction = "pca",dims = 1:30, verbose = FALSE)


#Visualization
DimPlot(Seurat, reduction = "umap",  cells.highlight = doublets$x,pt.size = 0.1,
        sizes.highlight = 0.1)
```




```
#remove doublets
toRemove <- doublets$x
Seurat <- Seurat[,!colnames(Seurat) %in% toRemove]

Seurat
```

```
## An object of class Seurat
## 21700 features across 10582 samples within 2 assays
## Active assay: SCT (10850 features, 3000 variable features)
##  1 other assay present: RNA
##  2 dimensional reductions calculated: pca, umap
```

```
#check mean genes per nuclei
mean(Seurat@meta.data$nFeature_RNA)
```

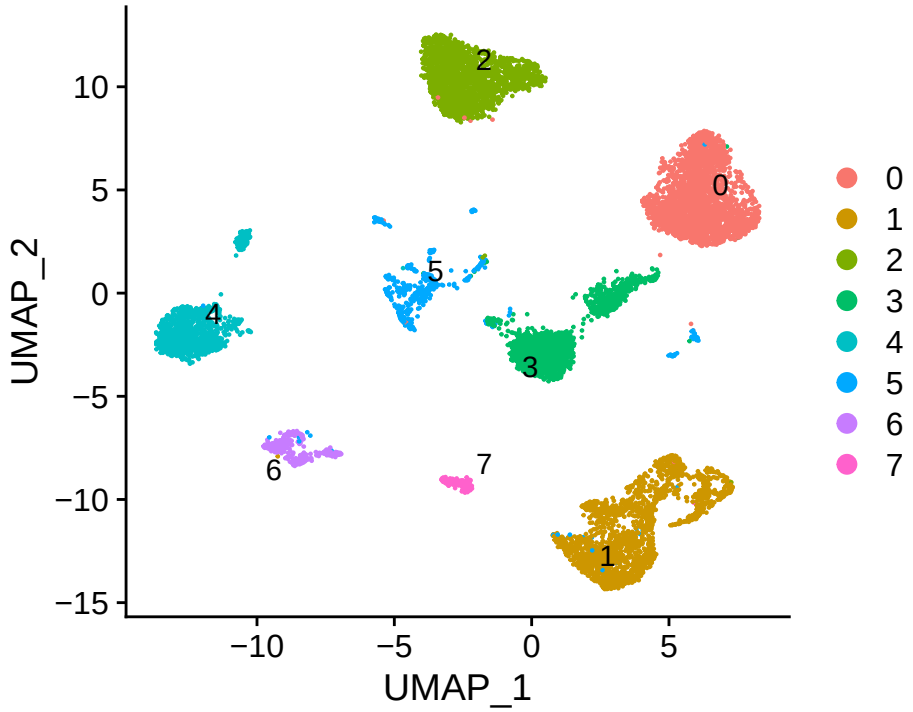
```
## [1] 756.4957
```

(Re)Normalization, dimensionality reduction and clustering

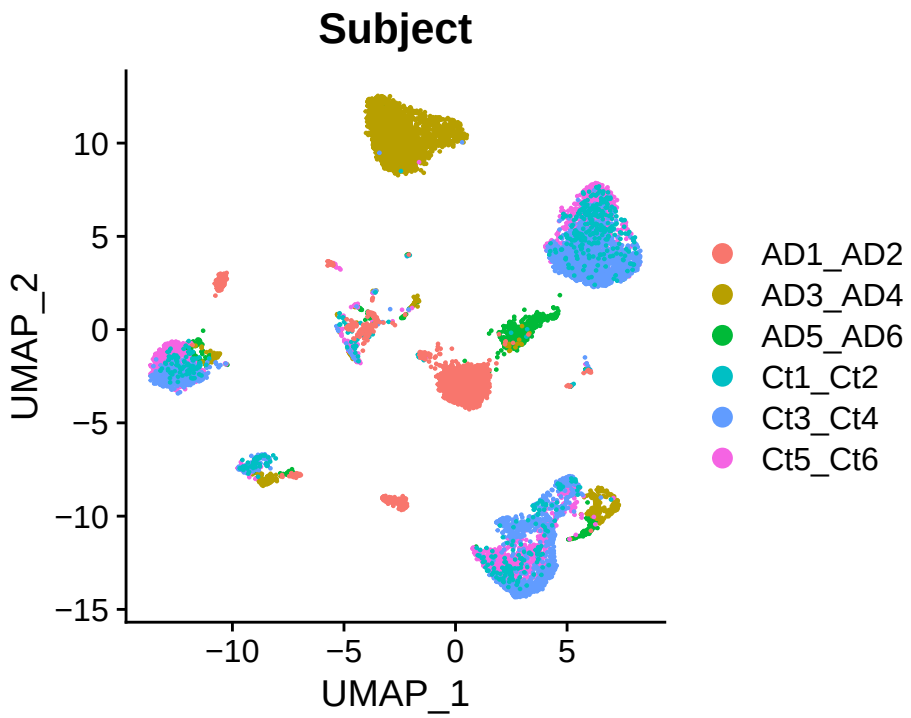
```
Seurat <- SCTransform(Seurat, verbose = FALSE) #defaul variable features = 3000
Seurat <- RunPCA(Seurat, verbose = FALSE)
Seurat <- RunUMAP(Seurat,  reduction = "pca",dims = 1:30, verbose = FALSE)
Seurat <- FindNeighbors(Seurat,  reduction = "pca",dims = 1:30)
Seurat <- FindClusters(Seurat,resolution = 0.05)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 10582
## Number of edges: 427492
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9889
## Number of communities: 8
## Elapsed time: 1 seconds
```

```
#UMAP
DimPlot(Seurat, reduction = "umap", label = TRUE, repel = TRUE)
```



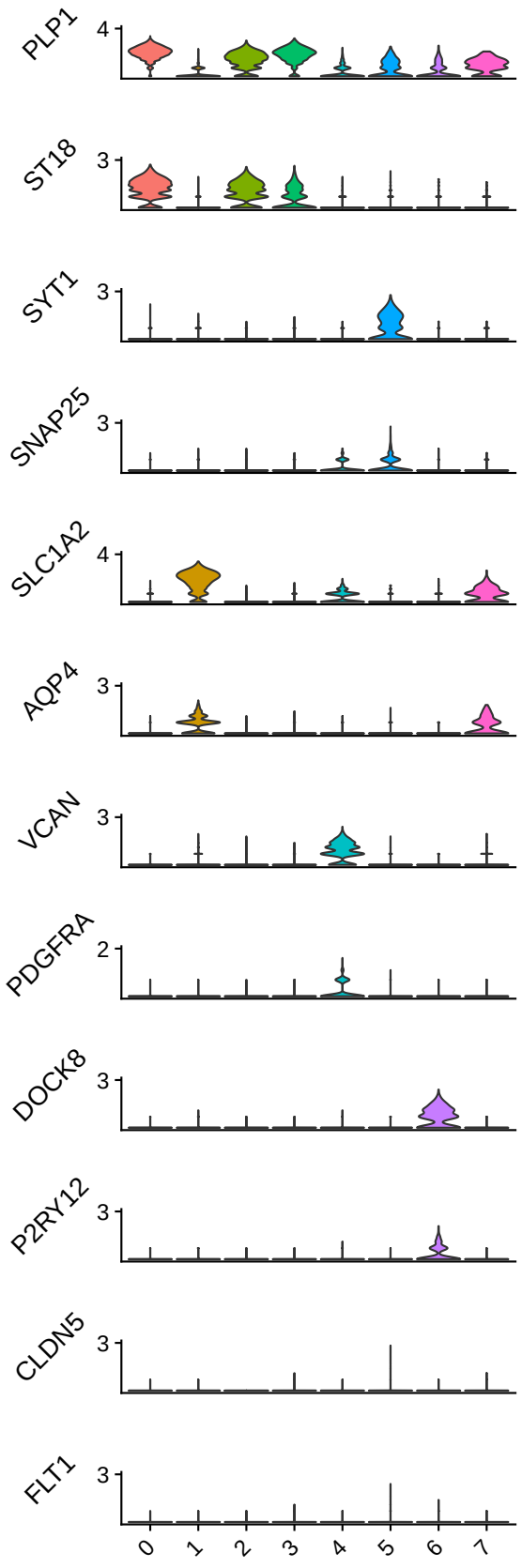

```
DimPlot(Seurat, reduction = "umap", label = FALSE, repel = TRUE, group.by = "Subject")
```



Calculate gene markers for each cluster

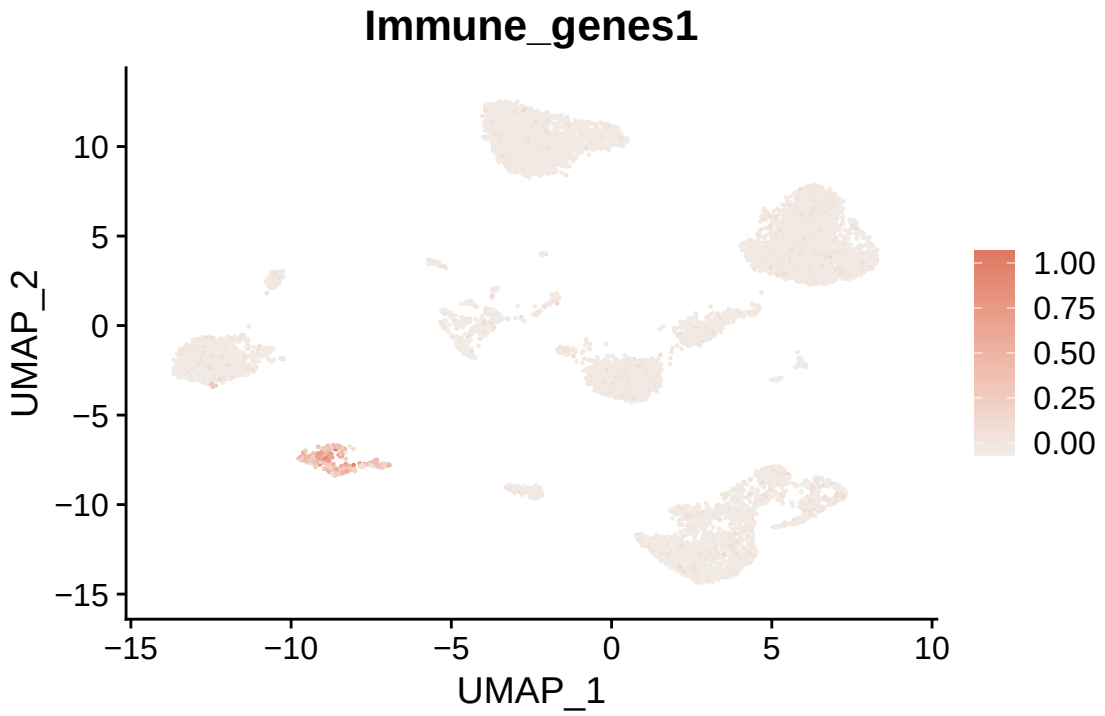
```
Seurat.markers <- FindAllMarkers(Seurat, only.pos = TRUE, min.pct = 0.1, logfc.threshold = 0.25)

#upload excell with cell type markers from the literature manually curated
Brain_cell_markers <- read_excel("~/windows_ricardo/Desktop/rstudio_data/HuMicA/Public_datasets/Brain_c
                                 col_names = FALSE)
colnames(Brain_cell_markers)<- c("Cell_type", "gene","source")

##Check for known markers in Seurat.markers
known_markers <- merge(Seurat.markers, Brain_cell_markers, by="gene") #128 overlaps


#Stacked violin plot function
## remove the x-axis text and tick
## plot.margin to adjust the white space between each plot.
## ... pass any arguments to VlnPlot in Seurat

modify_vlnplot<- function(obj,
                          feature,
                          pt.size = 0,
                          plot.margin = unit(c(-0.75, 0, -0.75, 0), "cm"),
                          ...) {
  p<- VlnPlot(obj, features = feature, pt.size = pt.size #,cols = UMAP_colors
              , ... )  +
    xlab("") + ylab(feature) + ggtitle("") +
    theme(legend.position = "none",
```

```
          axis.text.x = element_blank(),
          axis.ticks.x = element_blank(),
          axis.title.y = element_text(size = rel(1), angle = 45),
          axis.text.y = element_text(size = rel(1)),
          plot.margin = plot.margin )
  return(p)
}
## extract the max value of the y axis
extract_max<- function(p){
  ymax<- max(ggplot_build(p)$layout$panel_scales_y[[1]]$range$range)
  return(ceiling(ymax))
}


## main function
StackedVlnPlot<- function(obj, features,
                          pt.size = 0,
                          plot.margin = unit(c(-0.75, 0, -0.75, 0), "cm"),
                          ...) {

  plot_list<- purrr::map(features, function(x) modify_vlnplot(obj = obj,feature = x, ...))

  # Add back x-axis title to bottom plot. patchwork is going to support this?
  plot_list[[length(plot_list)]]<- plot_list[[length(plot_list)]] +
    theme(axis.text.x=element_text(angle = 45, hjust = 1), axis.ticks.x = element_line())

  # change the y-axis tick to only max value
  ymaxs<- purrr::map_dbl(plot_list, extract_max)
  plot_list<- purrr::map2(plot_list, ymaxs, function(x,y) x +
                            scale_y_continuous(breaks = c(y)) +
                            expand_limits(y = y))

  p<- patchwork::wrap_plots(plotlist = plot_list, ncol = 1)
  return(p)
}

#stacked violin plots
markers<- c("PLP1","ST18",                      #OLIGO
              "SYT1", "SNAP25",                 #NEURONS
              "SLC1A2", "AQP4",                 #ASTROCYTES
              "VCAN", "PDGFRA",                 #OPCS
              "DOCK8", "P2RY12",                #IMMUNE CELLS
              "CLDN5", "FLT1")                  #ENDOT

#Plot
StackedVlnPlot(obj = Seurat, features = markers)
```

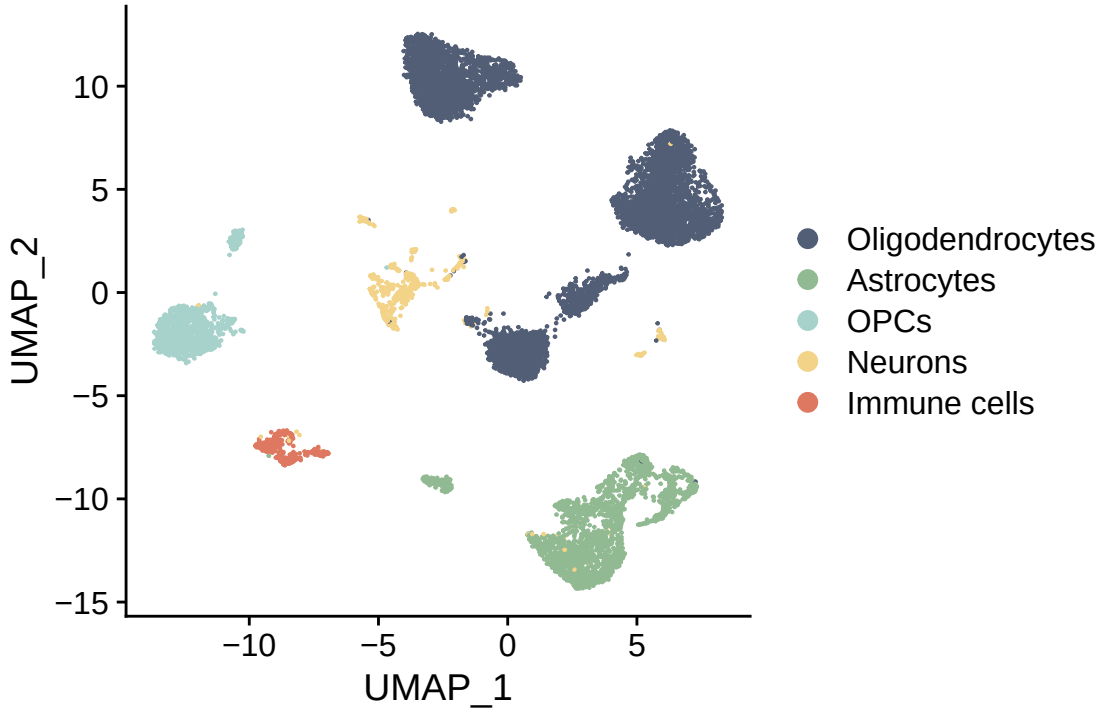

PLP1
4
ST18
3
SYT1
3
SNAP25
3
SLC1A2
4
AQP4
3
VCAN
3
PDGFRA
2
DOCK8
3
P2RY12
3
CLDN5
3
FLT1
3
0
1
2
3
4
5
6
7

Assign new cluster ids for cell types based on marker expresion

```
#Assign new cluster ids manually based on marker expresion
new.cluster.ids <- c("Oligodendrocytes", "Astrocytes","Oligodendrocytes",
                     "Oligodendrocytes","OPCs","Neurons",
                     "Immune cells","Astrocytes")

names(new.cluster.ids) <- levels(Seurat)
Seurat <- RenameIdents(Seurat, new.cluster.ids)

#Feature plot for immune cell markers (Module Score)
DefaultAssay(Seurat) <- "SCT"
Immune_genes <-c( "P2RY12","DOCK8","CD74","CX3CR1","C1QB","C3","AIF1","HLA-DRA")
Seurat <- AddModuleScore(Seurat,
                         features =list(Immune_genes),
                         name='Immune_genes')

FeaturePlot(Seurat, features = "Immune_genes1",cols = c("#F1EEE9","#DF7861"))
```




```
#UMAP of the annotated object

UMAP_colors<-c("#525E75", #Oligodendrocytes
               "#92BA92", #Astrocytes
               "#A7D2CB", #OPCs
               "#F2D388", #Neurons
               "#DF7861"  #Immune cells
               )
DimPlot(Seurat, reduction = "umap", label = FALSE, repel = F ,cols = UMAP_colors)
```



Subset the immune cell cluter/s from the Seurat object

```
Immune_Seurat<- Seurat[,Seurat@active.ident=="Immune cells"]
```

Check for Subject outliers

Check for individual samples with low cell number

```
md <- Immune_Seurat@meta.data %>% as.data.table
Cell_number_subject <- md[, .N, by = c("Subject")]

Cell_number_subject
```

```
##    Subject  N
## 1: AD5_AD6 30
## 2: AD3_AD4 84
## 3: Ct5_Ct6 56
## 4: Ct3_Ct4 84
## 5: Ct1_Ct2 39
## 6: AD1_AD2 47
```

All individual subjects/samples have similar number of nuclei.

Remove subjects with number of nuclei too low

In this case, this does not apply. The following is an example of how to remove samples "subjectX" and "subjectY"

```
#Nuclei_SubjectX<- Immune_Seurat[,Immune_Seurat@meta.data$Subject=="SubjectX"]%>%rownames()
#Nuclei_SubjectY<- Immune_Seurat[,Immune_Seurat@meta.data$Subject=="SubjectY"]%>%rownames()

#toRemove <- c(Nuclei_SubjectX, Nuclei_SubjectY)

#Immune_Seurat <- Immune_Seurat[,!colnames(Immune_Seurat) %in% toRemove]
```