An Atlas of Transcriptionally Defined Cell Types in the Rat Ventral Tegmental Area

Supplemental Code

Robert A. Phillips III, Jennifer J. Tuscher, Samantha L. Black, Lara Ianov, Jeremy J. Day

## Load Libraries

suppressPackageStartupMessages(library("data.table"))  
suppressPackageStartupMessages(library("pheatmap"))  
suppressPackageStartupMessages(library("reldist"))  
suppressPackageStartupMessages(library("ggplot2"))  
suppressPackageStartupMessages(library("cowplot"))  
suppressPackageStartupMessages(library("ggrepel"))  
suppressPackageStartupMessages(library("Seurat"))   
suppressPackageStartupMessages(library("tibble"))  
suppressPackageStartupMessages(library("tidyr"))  
suppressPackageStartupMessages(library("dplyr"))

## Identification of Major Cell Types within the Rat Ventral Tegmental Area

The code within this chunk closely follows the workflow demonstrated on the Satija lab website <https://satijalab.org/seurat/articles/pbmc3k_tutorial.html> and from a previously published analysis with code available at <https://gitlab.rc.uab.edu/day-lab>. Additional code comments are included in the chunk. A similar workflow is used to identify VTA neuronal subsets.

#set seed for reproducibility  
set.seed(1234)  
########VTA###########  
Fem1\_data <- Read10X(data.dir = "/data/project/daylab/2020-JD-0044/F1\_output/outs/filtered\_feature\_bc\_matrix/")  
Fem2\_data <- Read10X(data.dir = "/data/project/daylab/2020-JD-0044/F2\_output/outs/filtered\_feature\_bc\_matrix/")  
Male1\_data <- Read10X(data.dir = "/data/project/daylab/2020-JD-0044/M1\_output/outs/filtered\_feature\_bc\_matrix/")  
Male2\_data <- Read10X(data.dir = "/data/project/daylab/2020-JD-0044/M2\_output/outs/filtered\_feature\_bc\_matrix/")  
  
#Create the Seurat object   
#Using arbitrary cutoffs here. This allows us to interrogate the quality of every cell, while reserving the right to remove some at   
#a later QC point.   
Fem1 <- CreateSeuratObject(counts = Fem1\_data,min.cells = 1,min.features = 1) #5873 nuclei

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

Fem2 <- CreateSeuratObject(counts = Fem2\_data,min.cells = 1,min.features = 1) #5150 nuclei

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

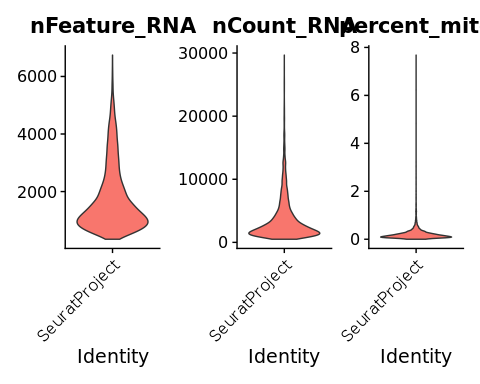
Male1 <- CreateSeuratObject(counts = Male1\_data,min.cells = 1,min.features = 1) #3628 nuclei

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

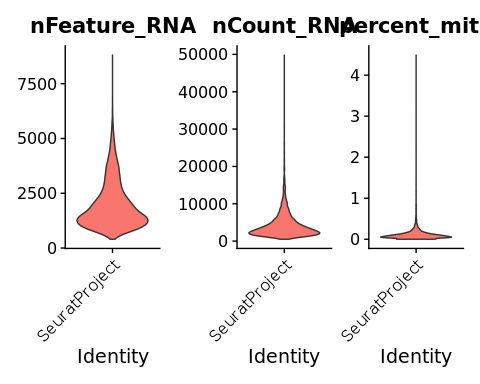
Male2 <- CreateSeuratObject(counts = Male2\_data,min.cells = 1,min.features = 1) #6955 nuclei

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

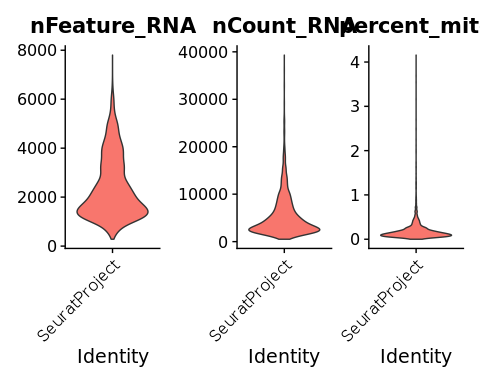
#21606  
  
#Identify the percentage of reads mapping to mitochondrial genes   
Fem1 <- PercentageFeatureSet(Fem1, pattern = "^Mt-", col.name = "percent\_mito")  
Fem2 <- PercentageFeatureSet(Fem2, pattern = "^Mt-", col.name = "percent\_mito")  
Male1 <- PercentageFeatureSet(Male1, pattern = "^Mt-", col.name = "percent\_mito")  
Male2 <- PercentageFeatureSet(Male2, pattern = "^Mt-", col.name = "percent\_mito")  
  
#VlnPlots to visualize QC metrics  
VlnPlot(Fem1, features = c("nFeature\_RNA", "nCount\_RNA", "percent\_mito"), ncol = 3,pt.size = 0)



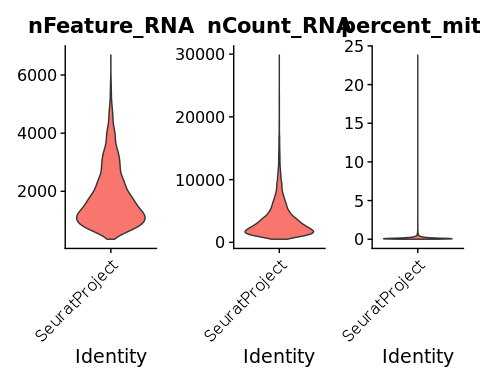
VlnPlot(Fem2, features = c("nFeature\_RNA", "nCount\_RNA", "percent\_mito"), ncol = 3,pt.size = 0)



VlnPlot(Male1, features = c("nFeature\_RNA", "nCount\_RNA", "percent\_mito"), ncol = 3,pt.size = 0)



VlnPlot(Male2, features = c("nFeature\_RNA", "nCount\_RNA", "percent\_mito"), ncol = 3,pt.size = 0)



#Subset data to have greater than 200 features and less than 5% of reads mapping to mitochondrial genes   
Fem1 <- subset(x = Fem1, subset = nFeature\_RNA > 200 & percent\_mito < 5) #5871 nuclei   
Fem2 <- subset(x = Fem2, subset = nFeature\_RNA > 200 & percent\_mito < 5) #5150 nuclei   
Male1 <- subset(x = Male1, subset = nFeature\_RNA > 200 & percent\_mito < 5) #3628 nuclei   
Male2 <- subset(x = Male2, subset = nFeature\_RNA > 200 & percent\_mito < 5) #6951 nuclei  
#21600  
  
#Log Normalize the data  
Fem1 <- NormalizeData(Fem1, normalization.method = "LogNormalize", scale.factor = 10000)  
Fem2 <- NormalizeData(Fem2, normalization.method = "LogNormalize", scale.factor = 10000)  
Male1 <- NormalizeData(Male1, normalization.method = "LogNormalize", scale.factor = 10000)  
Male2 <- NormalizeData(Male2, normalization.method = "LogNormalize", scale.factor = 10000)  
  
#Find variable features  
Fem1 <- FindVariableFeatures(Fem1, selection.method = "vst", nfeatures = 2000)  
Fem2 <- FindVariableFeatures(Fem2, selection.method = "vst", nfeatures = 2000)  
Male1 <- FindVariableFeatures(Male1, selection.method = "vst", nfeatures = 2000)  
Male2 <- FindVariableFeatures(Male2, selection.method = "vst", nfeatures = 2000)  
  
#Add Metadata attributes  
Fem1$GEM\_Well <- "Fem1"  
Fem2$GEM\_Well <- "Fem2"  
Male1$GEM\_Well <- "Male1"  
Male2$GEM\_Well <- "Male2"  
  
Fem1$Sex <- "Female"  
Fem2$Sex <- "Female"  
Male1$Sex <- "Male"  
Male2$Sex <- "Male"  
  
#Integrate the data  
VTA\_adult <- FindIntegrationAnchors(object.list = list(Fem1,Fem2,Male1,Male2), dims = 1:25)

## Warning in CheckDuplicateCellNames(object.list = object.list): Some cell names  
## are duplicated across objects provided. Renaming to enforce unique cell names.

## Computing 2000 integration features

## Scaling features for provided objects

## Finding all pairwise anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 13281 anchors

## Filtering anchors

## Retained 7664 anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 10571 anchors

## Filtering anchors

## Retained 7679 anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 9724 anchors

## Filtering anchors

## Retained 6693 anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 15208 anchors

## Filtering anchors

## Retained 8255 anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 13575 anchors

## Filtering anchors

## Retained 6928 anchors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 11338 anchors

## Filtering anchors

## Retained 6699 anchors

VTA\_adult <- IntegrateData(anchorset = VTA\_adult,dims = 1:25)

## Merging dataset 3 into 1

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

## Merging dataset 2 into 1 3

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

## Merging dataset 4 into 1 3 2

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

## Warning: Adding a command log without an assay associated with it

#Change the default assay  
DefaultAssay(VTA\_adult) <- "integrated"  
  
# Run the standard workflow for visualization and clustering  
VTA\_adult <- ScaleData(VTA\_adult,verbose = FALSE)  
VTA\_adult <- RunPCA(VTA\_adult,npcs = 25,verbose = FALSE) #Compute 50 npcs by default  
# Dimensionality reduction and Clustering  
VTA\_adult <- RunUMAP(VTA\_adult, reduction = "pca", dims = 1:25)

## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R-native UWOT using the cosine metric  
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'  
## This message will be shown once per session

## 15:58:47 UMAP embedding parameters a = 0.9922 b = 1.112

## 15:58:47 Read 21600 rows and found 25 numeric columns

## 15:58:47 Using Annoy for neighbor search, n\_neighbors = 30

## 15:58:47 Building Annoy index with metric = cosine, n\_trees = 50

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

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

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|  
## 15:58:54 Writing NN index file to temp file /tmp/RtmprfBzDp/file5ba165c7b838  
## 15:58:54 Searching Annoy index using 1 thread, search\_k = 3000  
## 15:59:04 Annoy recall = 100%  
## 15:59:05 Commencing smooth kNN distance calibration using 1 thread  
## 15:59:07 Initializing from normalized Laplacian + noise  
## 15:59:14 Commencing optimization for 200 epochs, with 1000998 positive edges  
## 15:59:48 Optimization finished

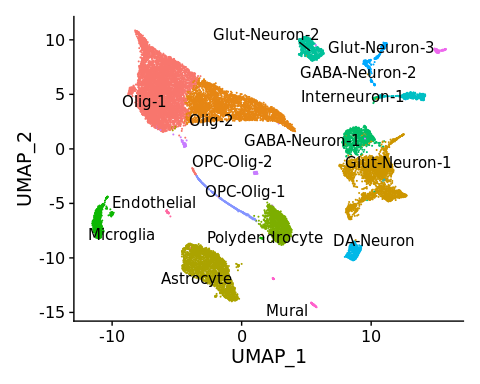
VTA\_adult <- FindNeighbors(VTA\_adult, reduction = "pca", dims = 1:25)

## Computing nearest neighbor graph  
## Computing SNN

VTA\_adult <- FindClusters(VTA\_adult, resolution = 0.2)

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck  
##   
## Number of nodes: 21600  
## Number of edges: 1109939  
##   
## Running Louvain algorithm...  
## Maximum modularity in 10 random starts: 0.9609  
## Number of communities: 16  
## Elapsed time: 5 seconds

#Rename the identities  
VTA\_adult <- RenameIdents(object = VTA\_adult,  
 "0" = "Olig-1",  
 "1" = "Olig-2",  
 "2" = "Glut-Neuron-1",  
 "3" = "Astrocyte",  
 "4" = "Polydendrocyte",  
 "5" = "Microglia",  
 "6" = "GABA-Neuron-1",  
 "7" = "Glut-Neuron-2",  
 "8" = "Interneuron-1",  
 "9" = "DA-Neuron",  
 "10"= "GABA-Neuron-2",  
 "11"= "OPC-Olig-1",  
 "12"= "OPC-Olig-2",   
 "13"= "Glut-Neuron-3",  
 "14"= "Mural",  
 "15"= "Endothelial")  
  
  
  
#Make a celltype column  
VTA\_adult$CellType <- Idents(VTA\_adult)  
  
#Plot the UMAP  
DimPlot(object = VTA\_adult,reduction = "umap",label = TRUE,repel = TRUE) + NoLegend()



## DEG Testing for Major Cell Populations

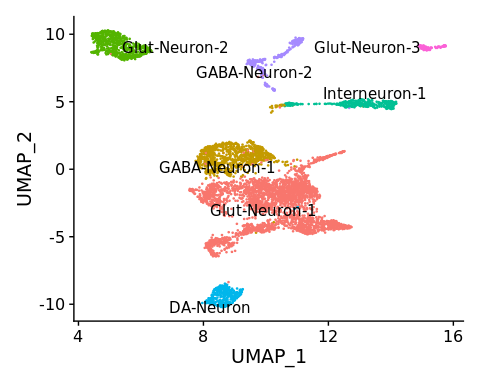
##DEG Testing  
################DEG TESTING MAJOR CELTYPES#########################  
DefaultAssay(VTA\_adult) <- "RNA"  
#DEG Analysis  
#Build a list to input DEGs into.  
Cluster\_Lists <- vector(mode = "list",length = length(unique(Idents(VTA\_adult))))  
#Change the names of the list to the cell types  
names(Cluster\_Lists) <- as.character(unique(Idents(VTA\_adult)))  
#Build a dataframe that is the number of genes in each cell type specific dataframe  
No\_Genes <- data.frame(CellType = names(Cluster\_Lists),  
 No\_Genes = NA)  
#Using modified Seurat source code  
for(i in names(Cluster\_Lists)){  
 #Print the element so we can actually see the cell type as we loop through  
 print(i)  
 #Create a dataframe in each element of the list where FC will be the log2FC and gene will be the name  
 Cluster\_Lists[[i]] <- data.frame(log2FC = (log2(rowMeans(expm1(x = as.matrix(GetAssayData(object = VTA\_adult,slot = "data",assay = "RNA")[,WhichCells(object = VTA\_adult,idents = i)]))))) -  
 log2(rowMeans(expm1(x = as.matrix(GetAssayData(object = VTA\_adult,slot = "data",assay = "RNA")[,WhichCells(object = VTA\_adult,idents = as.character(unique(Idents(VTA\_adult))[which(unique(Idents(VTA\_adult))!=i)]))])))),  
 gene = row.names(as.matrix(GetAssayData(object = VTA\_adult,slot = "data",assay= "RNA"))[,WhichCells(object = VTA\_adult,idents = i)]))  
 #Remove infinite values  
 Cluster\_Lists[[i]] <- Cluster\_Lists[[i]][!is.infinite(Cluster\_Lists[[i]]$log2FC),]  
 #remove NaN values  
 Cluster\_Lists[[i]] <- Cluster\_Lists[[i]][!is.nan(Cluster\_Lists[[i]]$log2FC),]  
 #paste the number of genes left  
 No\_Genes[which(No\_Genes$CellType ==i),"No\_Genes"] <- nrow(Cluster\_Lists[[i]])  
}  
  
write.table(x = No\_Genes,  
 file = "/data/project/daylab/2020-JD-0044/Analysis/No\_of\_Genes\_Tested\_Per\_Cluster\_VTA\_Major\_Clusters.txt",  
 sep = "\t",  
 col.names = TRUE,  
 row.names = FALSE,  
 quote = FALSE)  
  
#Now test genes for significance  
for(i in names(Cluster\_Lists)){  
 print(i)  
 #Create a data.frame where the rownames are the cells within the clusters  
 group.info <- data.frame(row.names = c(WhichCells(object = VTA\_adult,idents = i),  
 WhichCells(object = VTA\_adult,idents = as.character(unique(Idents(VTA\_adult))[which(unique(Idents(VTA\_adult))!=i)]))))  
 #Create a group column where cocaine is Group1 and saline is group2. The way to do that is to search the rows names for Cocaine and Saline cell identities  
 group.info[WhichCells(object = VTA\_adult,idents = i), "group"] <- "Group1"  
 group.info[WhichCells(object = VTA\_adult,idents = as.character(unique(Idents(VTA\_adult))[which(unique(Idents(VTA\_adult))!=i)])), "group"] <- "Group2"  
 #Make the group column a factor so it can be tested with the wilcox  
 group.info[, "group"] <- factor(x = group.info[, "group"])  
 #Create an expression matrix where the row.names are the genes found within that cluster - do this by running row.names(Cluster\_Lists[[i]])  
 #the columns should be row.names(group.info) which are the cells within that cluster  
 #Drop maintains the structure of the matrix when subsetting, which is needed when pulling the expression matrix  
 #Pull counts from log-normalized matrix  
 data.use <- GetAssayData(object = VTA\_adult,slot = "data",assay = "RNA")[row.names(Cluster\_Lists[[i]]), rownames(x = group.info), drop = FALSE]  
 #Loop through the rows in the expression matrix which are the genes  
 for(l in 1:length(row.names(data.use))){  
 #Calculate p-values for every gene  
 #Do this by entering the element of the list for the specific cluster that we are running. i will always be the cell type  
 #row.names(data.use)[l] will be the gene name  
 #wilcox.test will test gene by group to see if there are any differences. Then by adding $p.value we can pull only the p-value from the test  
 Cluster\_Lists[[i]][row.names(data.use)[l],"p.val"] <- wilcox.test(data.use[row.names(data.use)[l], ] ~ group.info[,"group"])$p.value  
 Cluster\_Lists[[i]][row.names(data.use)[l],"p.adj"] <- p.adjust(Cluster\_Lists[[i]][row.names(data.use)[l],"p.val"],  
 method = "bonferroni",  
 n = nrow(Cluster\_Lists[[i]]))  
 }  
 write.table(x = Cluster\_Lists[[i]],  
 file = paste0("/data/project/daylab/2020-JD-0044/Analysis/Major\_Cluster\_Marker\_Genes/",i,"\_DEGs.txt"),  
 col.names = TRUE,  
 row.names = FALSE,  
 quote = FALSE,  
 sep = "\t")  
}

## Sex Differences Analysis

#Create a metadata column combining cell type and sex information.   
VTA\_adult$CellType\_Sex <- paste(VTA\_adult$CellType, VTA\_adult$Sex, sep = "\_")  
############# Sex differences analysis ###############  
#Plot UMAP by sex using sex index colors  
DimPlot(VTA\_adult, reduction = "umap", group.by = "Sex", pt.size = 0.1, cols = c("#FF2F90","#0094FF")) + coord\_equal()  
## Calculate fraction of cells in each sex group  
#Make Idents celltype sex and build dataframe with every cluster and columns for male and female cells  
Idents(VTA\_adult) <- VTA\_adult$CellType\_Sex  
Idents\_Sex <- as.data.frame(matrix(nrow = 16,ncol = 3))  
names(Idents\_Sex) <- c("CellType","Male","Female")  
#Get male and female number of cells   
for(i in 1:length(levels(VTA\_adult$CellType))){  
 Idents\_Sex[i,"CellType"] <- levels(VTA\_adult$CellType)[i]  
 Idents\_Sex[i,"Male"] <- length(WhichCells(object = VTA\_adult,idents = paste(levels(VTA\_adult$CellType)[i],"Male",sep = "\_")))  
 Idents\_Sex[i,"Female"] <- length(WhichCells(object = VTA\_adult,idents = paste(levels(VTA\_adult$CellType)[i],"Female",sep = "\_")))  
}  
#Calculate proportions  
for(i in 1:nrow(Idents\_Sex)){  
 Idents\_Sex[i,"Male\_Frac"] <- (Idents\_Sex[i,"Male"]/as.numeric(table(VTA\_adult$Sex)["Male"]))  
 Idents\_Sex[i,"Female\_Frac"] <- (Idents\_Sex[i,"Female"]/as.numeric(table(VTA\_adult$Sex)["Female"]))  
}  
#Write out file  
write.csv(Idents\_Sex, "Proportions\_by\_sex.csv")

## Identifying Transcriptionally Distinct Neuronal Subpopulations

############Subclustering DA/GABA/Glut Neurons  
DefaultAssay(VTA\_adult) <- "RNA"  
#Switch the identities back to celltype   
Idents(VTA\_adult) <- VTA\_adult$CellType  
#Subset for DA/GABA/Glut Neurons  
VTA\_subset <- subset(VTA\_adult,idents = c("Glut-Neuron-1","Glut-Neuron-2","Glut-Neuron-3","Interneuron-1","GABA-Neuron-1","GABA-Neuron-2","DA-Neuron"))  
#Look at the umap  
DimPlot(object = VTA\_subset,label = TRUE,repel = TRUE) + NoLegend()



VTA\_subset <- FindVariableFeatures(VTA\_subset,selection.method = "vst",nfeatures = 2000)  
VTA\_subset <- ScaleData(VTA\_subset,verbose = FALSE)  
VTA\_subset <- RunPCA(VTA\_subset,npcs = 15,verbose = FALSE) #Compute 50 npcs by default  
# Dimensionality reduction and Clustering  
VTA\_subset <- RunUMAP(VTA\_subset, reduction = "pca", dims = 1:15)

## 16:00:15 UMAP embedding parameters a = 0.9922 b = 1.112

## 16:00:15 Read 5950 rows and found 15 numeric columns

## 16:00:15 Using Annoy for neighbor search, n\_neighbors = 30

## 16:00:15 Building Annoy index with metric = cosine, n\_trees = 50

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

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

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|  
## 16:00:17 Writing NN index file to temp file /tmp/RtmprfBzDp/file5ba14ee8a066  
## 16:00:17 Searching Annoy index using 1 thread, search\_k = 3000  
## 16:00:19 Annoy recall = 100%  
## 16:00:20 Commencing smooth kNN distance calibration using 1 thread  
## 16:00:21 Initializing from normalized Laplacian + noise  
## 16:00:21 Commencing optimization for 500 epochs, with 237482 positive edges  
## 16:00:43 Optimization finished

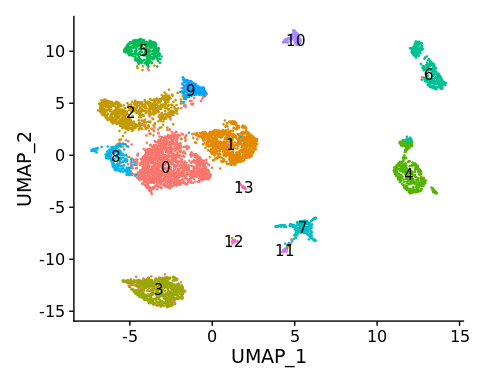
VTA\_subset <- FindNeighbors(VTA\_subset, reduction = "pca", dims = 1:15)

## Computing nearest neighbor graph  
## Computing SNN

VTA\_subset <- FindClusters(VTA\_subset, resolution = 0.2)

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck  
##   
## Number of nodes: 5950  
## Number of edges: 189560  
##   
## Running Louvain algorithm...  
## Maximum modularity in 10 random starts: 0.9573  
## Number of communities: 14  
## Elapsed time: 0 seconds

DimPlot(object = VTA\_subset,label = TRUE) + NoLegend()



## Generating Heatmaps for Dopamine,Glutamate, and GABA Marker Genes in VTA Neuronal Subclusters

#New subset  
Cluster\_Lists <- vector(mode = "list",length = 11)  
#name every element of the list the numeric identity from subclustering  
names(Cluster\_Lists) <- c(0:10)  
#Within each element of the list, create a dataframe that will contain the percentage of cells within the cluster expressing two genes   
for(i in names(Cluster\_Lists)){  
 Cluster\_Lists[[i]] <- data.frame(Th = rep(NA,10),  
 Ddc = rep(NA,10),  
 Slc6a3 = rep(NA,10),  
 Slc18a2 = rep(NA,10),  
 Slc17a6 = rep(NA,10),  
 Grm2 = rep(NA,10),  
 Gad1 = rep(NA,10),  
 Gad2 = rep(NA,10),  
 Slc32a1 = rep(NA,10),  
 Slc6a1 = rep(NA,10),  
 row.names = c("Th","Ddc","Slc6a3","Slc18a2", #DA  
 "Slc17a6","Grm2",  
 "Gad1","Gad2","Slc32a1","Slc6a1"))  
   
}  
#Calculate the percentage of cells expressing two genes   
for(i in c(0:10)){  
 #Subset the VTA\_subset object for a single identity  
 VTA\_subset2 <- subset(VTA\_subset,idents = i)  
 #Pull the count values for the 11 genes and transpose the dataframe so that cells are rows and genes are columns   
 x <- as.data.frame(t(as.matrix(GetAssayData(object = VTA\_subset2,assay = "RNA")[c("Th","Ddc","Slc6a3","Slc18a2", #DA  
 "Slc17a6","Grm2","Slc17a7",#Glut  
 "Gad1","Gad2","Slc32a1","Slc6a1"),])))  
 #Now loop through the 11 genes of interest by looping through the column names of each dataframe created in the first step   
 for(l in colnames(Cluster\_Lists[[as.character(i)]])){  
 #Get all of the genes that are not equal to the iteration of the loop  
 Other\_Genes <- colnames(Cluster\_Lists[[as.character(i)]])[which(colnames(Cluster\_Lists[[as.character(i)]])!=l)]  
 for(k in Other\_Genes){  
 #Calculate the percentage of cells expressing the l and k, both of which are some combination of the 11 genes  
 #Dividing by ncol(VTA\_subset2) divides by the number of cells within the cluster of interest   
 Cluster\_Lists[[as.character(i)]][l,k] <- length(which(x[,l] > 0 & x[,k] > 0))/ncol(VTA\_subset2)  
 }  
 }  
}  
  
for(i in names(Cluster\_Lists)){  
pdf(file = paste0("/data/project/daylab/2020-JD-0044/Analysis/Neurotransmitter\_Heatmaps/Fig",i,"\_heatmap.pdf"),  
 height = 8,  
 width = 8)  
 breaksList <- seq(0, 1, by = 0.05)  
 pheatmap(mat = Cluster\_Lists[[i]],  
 color = colorRampPalette(c("gray96","peachpuff","goldenrod1","orange","darkorange","red","red4"))(length(breaksList)),  
 breaks = breaksList,  
 cluster\_rows = FALSE,  
 cluster\_cols = FALSE,  
 main = paste("Cluster",i,sep = " "))  
 dev.off()  
}

## Identification of VTA Neuronal Subset DEGs

######DEGS   
VTA\_subset <- RenameIdents(object = VTA\_subset,  
 "0" = "Zero",  
 "1" = "One",  
 "2" = "Two",  
 "3" = "Three",  
 "4" = "Four",  
 "5" = "Five",  
 "6" = "Six",  
 "7" = "Seven",  
 "8" = "Eight",  
 "9" = "Nine",  
 "10" = "Ten")  
#VTA\_subset$Character\_Ident <- Idents(VTA\_subset)  
#Change the default assay to RNA  
DefaultAssay(VTA\_subset) <- "RNA"  
#DEG Analysis   
#Build a list to input DEGs into.   
Cluster\_Lists <- vector(mode = "list",length = 11)  
#Change the names of the list to the cell types   
names(Cluster\_Lists) <- levels(Idents(VTA\_subset))[1:11]  
#Build a dataframe that is the number of genes in each cell type specific dataframe   
No\_Genes <- data.frame(CellType = names(Cluster\_Lists),  
 No\_Genes = NA)  
#Using modified Seurat source code   
for(i in names(Cluster\_Lists)){  
 #Print the element so we can actually see the cell type as we loop through  
 print(i)  
 #Create a dataframe in each element of the list where FC will be the log2FC and gene will be the name  
 Cluster\_Lists[[i]] <- data.frame(log2FC = (log2(rowMeans(expm1(x = as.matrix(GetAssayData(object = VTA\_subset,slot = "data",assay = "RNA")[,WhichCells(object = VTA\_subset,idents = i)]))))) -  
 log2(rowMeans(expm1(x = as.matrix(GetAssayData(object = VTA\_subset,slot = "data",assay = "RNA")[,WhichCells(object = VTA\_subset,idents = as.character(unique(Idents(VTA\_subset))[which(unique(Idents(VTA\_subset))!=i)]))])))),  
 gene = row.names(as.matrix(GetAssayData(object = VTA\_subset,slot = "data",assay= "RNA"))[,WhichCells(object = VTA\_subset,idents = i)]))  
 #Remove infinite values  
 Cluster\_Lists[[i]] <- Cluster\_Lists[[i]][!is.infinite(Cluster\_Lists[[i]]$log2FC),]  
 #remove NaN values  
 Cluster\_Lists[[i]] <- Cluster\_Lists[[i]][!is.nan(Cluster\_Lists[[i]]$log2FC),]  
 #paste the number of genes left   
 No\_Genes[which(No\_Genes$CellType ==i),"No\_Genes"] <- nrow(Cluster\_Lists[[i]])  
}  
  
  
write.table(x = No\_Genes,  
 file = "/data/project/daylab/2020-JD-0044/Analysis/No\_of\_Genes\_Tested\_Per\_Cluster\_VTA\_Neuronal\_Subset.txt",  
 sep = "\t",  
 col.names = TRUE,  
 row.names = FALSE,  
 quote = FALSE)  
  
#Now test genes for significance   
for(i in names(Cluster\_Lists)){  
 print(i)  
 #Create a data.frame where the rownames are the cells within the clusters   
 group.info <- data.frame(row.names = c(WhichCells(object = VTA\_subset,idents = i),  
 WhichCells(object = VTA\_subset,idents = as.character(unique(Idents(VTA\_subset))[which(unique(Idents(VTA\_subset))!=i)]))))  
 #Create a group column where cocaine is Group1 and saline is group2. The way to do that is to search the rows names for Cocaine and Saline cell identities  
 group.info[WhichCells(object = VTA\_subset,idents = i), "group"] <- "Group1"  
 group.info[WhichCells(object = VTA\_subset,idents = as.character(unique(Idents(VTA\_subset))[which(unique(Idents(VTA\_subset))!=i)])), "group"] <- "Group2"  
 #Make the group column a factor so it can be tested with the wilcox   
 group.info[, "group"] <- factor(x = group.info[, "group"])  
 #Create an expression matrix where the row.names are the genes found within that cluster - do this by running row.names(Cluster\_Lists[[i]])  
 #the columns should be row.names(group.info) which are the cells within that cluster   
 #Drop maintains the structure of the matrix when subsetting, which is needed when pulling the expression matrix  
 #Pull counts from log-normalized matrix  
 data.use <- GetAssayData(object = VTA\_subset,slot = "data",assay = "RNA")[row.names(Cluster\_Lists[[i]]), rownames(x = group.info), drop = FALSE]  
 #Loop through the rows in the expression matrix which are the genes  
 for(l in 1:length(row.names(data.use))){  
 #Print the progress of the loop  
 #progress(l,max.value = length(row.names(data.use)))  
 #Calculate p-values for every gene   
 #Do this by entering the element of the list for the specific cluster that we are running. i will always be the cell type  
 #row.names(data.use)[l] will be the gene name  
 #wilcox.test will test gene by group to see if there are any differences. Then by adding $p.value we can pull only the p-value from the test   
 Cluster\_Lists[[i]][row.names(data.use)[l],"p.val"] <- wilcox.test(data.use[row.names(data.use)[l], ] ~ group.info[,"group"])$p.value  
 Cluster\_Lists[[i]][row.names(data.use)[l],"p.adj"] <- p.adjust(Cluster\_Lists[[i]][row.names(data.use)[l],"p.val"],  
 method = "bonferroni",  
 n = nrow(Cluster\_Lists[[i]]))  
 }  
 write.table(x = Cluster\_Lists[[i]],  
 file = paste0("/data/project/daylab/2020-JD-0044/Analysis/Neuronal\_Subset\_Cluster/",i,"\_DEGs.txt"),  
 col.names = TRUE,  
 row.names = FALSE,  
 quote = FALSE,  
 sep = "\t")  
}

## Calculation of Gini Coefficient and Plots for Figure 4

###Read in the DEG dataframe for cluster 5   
cluster5 <- read.table(file = "/data/project/daylab/2020-JD-0044/Analysis/Neuronal\_Subset\_Cluster/Five\_DEGs.txt",  
 sep = "\t",  
 header = TRUE)  
cluster9 <- read.table(file = "/data/project/daylab/2020-JD-0044/Analysis/Neuronal\_Subset\_Cluster/Nine\_DEGs.txt",  
 sep = "\t",  
 header = TRUE)  
#Figure out which genes are shared   
Shared\_Genes <- intersect(subset(cluster5,subset=(p.adj < 0.05 & log2FC > 0.5))$gene,  
 subset(cluster9,subset=(p.adj < 0.05 & log2FC > 0.5))$gene)  
#Make a shared column  
cluster9$shared <- ifelse(cluster9$gene %in% Shared\_Genes,  
 "Yes",  
 "No")  
cluster5$shared <- ifelse(cluster5$gene %in% Shared\_Genes,  
 "Yes",  
 "No")  
#Set the Seurat assay to RNA  
DefaultAssay(VTA\_subset) <- "RNA"  
#Set identities back to the integer values instead of the characters used for DEG testing  
Idents(VTA\_subset) <- VTA\_subset$RNA\_snn\_res.0.2  
#Calculate the cluster-specific average expression values for every gene  
CellAverages <- AverageExpression(VTA\_subset)  
#Pull the RNA values  
RNA <- CellAverages$RNA  
#Calculate the Gini coefficient, maximum/mean gene expression value, and ratio  
Gini\_calc <- apply(RNA, 1, gini)  
max\_RNA <- apply(RNA, 1, max)  
mean\_RNA <- apply(RNA, 1, mean)  
Ratio <- max\_RNA/mean\_RNA  
#Create a dataframe consisting of the Gini coefficient, maximum/mean gene expression value, and ratio  
Gini\_dataframe <- data.frame(Gini\_calc, max\_RNA, mean\_RNA, Ratio, RNA)  
#Create a column of gene names  
Gini\_dataframe$Gene <- rownames(Gini\_dataframe)  
#Remove all NAs  
Gini\_dataframe <- na.omit(Gini\_dataframe)  
#Create a logX5 df  
Gini\_dataframe\_logX5 <- Gini\_dataframe  
#calculate log10 for cluster 5  
Gini\_dataframe\_logX5$log\_X5 <- log10(Gini\_dataframe\_logX5$X5)  
#Remove the Infinite values  
Gini\_dataframe\_logX5 <- Gini\_dataframe\_logX5[!is.infinite(Gini\_dataframe\_logX5$log\_X5),]  
#Now merge   
Gini\_dataframe\_logX5 <- merge(x = Gini\_dataframe\_logX5,  
 y = cluster5,  
 by.x = "Gene",  
 by.y = "gene")  
#plot  
p1 <- ggplot(data = Gini\_dataframe\_logX5, aes(x = log\_X5,y = Gini\_calc)) +  
 geom\_point(color = "lightgrey") +  
 geom\_point(data = subset(Gini\_dataframe\_logX5,subset=(p.adj < 0.05 & log2FC > 0.5 & shared == "No")), color = "#f3766e") +  
 geom\_point(data = subset(Gini\_dataframe\_logX5,subset=(p.adj < 0.05 & log2FC > 0.5 & shared == "Yes")),color = "gray50") +  
 geom\_density2d(color = "red") +  
 geom\_text\_repel(data = subset(Gini\_dataframe\_logX5, subset= (Gene == "Gch1" | Gene == "Th" | Gene == "Drd2" | Gene == "Slc18a2" | Gene == "Slc6a3")),  
 label = subset(Gini\_dataframe\_logX5, subset= (Gene == "Gch1" | Gene == "Th" | Gene == "Drd2" | Gene == "Slc18a2" | Gene == "Slc6a3"))$Gene,  
 min.segment.length = 0,  
 box.padding = 0.5) +  
 theme\_bw() +  
 ylim(c(0,1)) +  
 NoGrid() +  
 ylab("Gini coefficient") +  
 xlab("log10(Cluster 5 Expression Values)") +  
 ggtitle("Classically-defined DA Neurons") +  
 theme(plot.title = element\_text(hjust = 0.5))  
#Create a logX9 df  
Gini\_dataframe\_logX9 <- Gini\_dataframe  
#calculate log10 for cluster 5  
Gini\_dataframe\_logX9$log\_X9 <- log10(Gini\_dataframe\_logX9$X9)  
#Remove the infinite values  
Gini\_dataframe\_logX9 <- Gini\_dataframe\_logX9[!is.infinite(Gini\_dataframe\_logX9$log\_X9),]  
#Now merge   
Gini\_dataframe\_logX9 <- merge(x = Gini\_dataframe\_logX9,  
 y = cluster9,  
 by.x = "Gene",  
 by.y = "gene")  
p2 <- ggplot(data = Gini\_dataframe\_logX9, aes(x = log\_X9,y = Gini\_calc)) +  
 geom\_point(color = "lightgrey") +  
 geom\_point(data = subset(Gini\_dataframe\_logX9,subset=(p.adj < 0.05 & log2FC > 0.5 & shared == "No")), color = "#e18a26") +  
 geom\_point(data = subset(Gini\_dataframe\_logX9,subset=(p.adj < 0.05 & log2FC > 0.5 & shared == "Yes")),color = "gray50") +  
 geom\_density2d(color = "red") +  
 geom\_text\_repel(data = subset(Gini\_dataframe\_logX9, subset= (Gene == "Slc26a7" | Gene == "Wnt2" | Gene == "Crhbp")),  
 label = subset(Gini\_dataframe\_logX9, subset= (Gene == "Slc26a7" | Gene == "Wnt2" | Gene == "Crhbp"))$Gene,  
 min.segment.length = 0,  
 box.padding = 0.5) +  
 theme\_bw() +  
 NoGrid() +  
 ylim(c(0,1)) +  
 ylab("Gini coefficient") +  
 xlab("log10(Cluster 9 Expression Values)") +  
 ggtitle("Combinatorial Neurons") +  
 theme(plot.title = element\_text(hjust = 0.5))  
   
   
p3 <- p1 + p2  
ggsave(plot = p3,  
 filename = "/data/project/daylab/2020-JD-0044/Analysis/Gini\_Coefficient\_Plots.pdf",  
 height = 8,  
 width = 12)

## Generation of Gene Lists for MAGMA

This analysis utilizes a curated file of vertebrate homology from the JAX lab. This file is updated regularly. Thus, some rat-human gene transfers may differ. However, we have performed the analysis with updated versions of the JAX vertebrate homology file and final MAGMA statistics have not changed

###FINAL Generation of MAGMA gene list  
#########Calculate the percent of cells expressing a gene  
#Create a list  
Pct\_Exp\_List <- vector(mode = "list",length = length(levels(VTA\_adult)))  
#Name each element   
names(Pct\_Exp\_List) <- levels(Idents(VTA\_adult))  
#Run the calculations  
for(i in levels(Idents(VTA\_adult))){  
 #Create a dataframe where the first column is the gene and the second column is the pct\_expressed  
 Pct\_Exp\_List[[i]] <- data.frame(Gene = NA,  
 Pct\_Expressed = NA)  
 #Pull the counts   
 data <- as.matrix(GetAssayData(VTA\_adult, assay = "RNA",slot = "data")[, WhichCells(VTA\_adult, ident = i)])  
 for(l in 1:nrow(data)){  
 #Create a dataframe to merge with the dataframe within each list  
 x <- data.frame(Gene = rownames(data)[l], #provides gene name  
 Pct\_Expressed = as.numeric(table(data[l,]>0)["TRUE"])/ncol(data)) #number of cells expressing gene/total number of cells  
 Pct\_Exp\_List[[i]] <- rbind(Pct\_Exp\_List[[i]],x)  
 }  
 #Remove the first row by omitting NAs  
 Pct\_Exp\_List[[i]] <- na.omit(Pct\_Exp\_List[[i]])  
 #Subset for only those genes that are expressed in >10% of cells.   
 Pct\_Exp\_List[[i]] <- subset(Pct\_Exp\_List[[i]],subset=(Pct\_Expressed > 0.10))  
}  
  
#Build a list to input DEGs into.   
Cluster\_Lists <- vector(mode = "list",length = length(unique(Idents(VTA\_adult))))  
#Change the names of the list to the cell types   
names(Cluster\_Lists) <- as.character(unique(Idents(VTA\_adult)))  
#Make a list for subset too  
Cluster\_Lists\_subset <- vector(mode = "list",length = length(unique(Idents(VTA\_adult))))  
#Change the names of the list to the cell types   
names(Cluster\_Lists\_subset) <- as.character(unique(Idents(VTA\_adult)))  
  
#Now read in everything  
for(i in names(Cluster\_Lists)){  
 #Read in the DEG lists  
 Cluster\_Lists[[i]] <- read.table(paste0("/data/project/daylab/2020-JD-0044/Analysis/Major\_Cluster\_Marker\_Genes/",i,"\_DEGs.txt"),sep = "\t",header = TRUE)  
 #Subset for genes that are enriched (log2FC >0.5) and with a bonferroni adjusted p-value < 1e-12  
 #MAGMA will take a two column dataframe as input where the first column is the "Set" (CellType) and the second column is the gene  
 Cluster\_Lists\_subset[[i]] <- data.frame(Set = rep(i,nrow(subset(Cluster\_Lists[[i]],subset=(p.adj < 1e-12 & log2FC > 0.5)))),  
 Gene = subset(Cluster\_Lists[[i]],subset=(p.adj < 1e-12 & log2FC > 0.5))$gene)  
 #Merge with the pct\_Expressed dataframe with the same name to keep only those DEGs that are expressed in at least 10% of cells.   
 Cluster\_Lists\_subset[[i]] <- merge(x = Cluster\_Lists\_subset[[i]],  
 y = Pct\_Exp\_List[[i]],  
 by = "Gene")  
}  
  
  
#Now merge the lists to create a large dataframe  
Cluster\_Lists\_subset\_df\_all <- do.call(what = "rbind",Cluster\_Lists\_subset)  
  
#From LieberInstitute github, thanks y'all  
hom <- read.delim("http://www.informatics.jax.org/downloads/reports/HOM\_AllOrganism.rpt",  
 as.is=TRUE)  
#Pull human and rat  
hom\_hs <- hom[hom$Common.Organism.Name == "human", ]  
hom\_rat <- hom[hom$Common.Organism.Name == "rat", ]  
  
#Create new columns for the dataframes   
Cluster\_Lists\_subset\_df\_all$Human\_Gene\_EntrezGene <- NA  
#Now map to the human homolog  
#for the whole gene set  
for(i in 1:nrow(Cluster\_Lists\_subset\_df\_all)){  
 #Only run the code if a homolog is found.   
 if(length(hom\_hs[hom\_hs$HomoloGene.ID %in% hom\_rat[hom\_rat$Symbol %in% Cluster\_Lists\_subset\_df\_all[i,"Gene"],"HomoloGene.ID"],"EntrezGene.ID"]) > 0){  
 Cluster\_Lists\_subset\_df\_all[i,"Human\_Gene\_EntrezGene"] <- hom\_hs[hom\_hs$HomoloGene.ID %in% hom\_rat[hom\_rat$Symbol %in% Cluster\_Lists\_subset\_df\_all[i,"Gene"],"HomoloGene.ID"],"EntrezGene.ID"][1]  
 }else{  
 next  
 }  
}  
  
#Some rat genes will not have a human ortholog. Thus, we omit all NA values.   
Cluster\_Lists\_subset\_df\_all <- na.omit(Cluster\_Lists\_subset\_df\_all)  
  
#write out the gene list  
write.table(x = Cluster\_Lists\_subset\_df\_all[,c("Set","Human\_Gene\_EntrezGene")], #Just need the Set/CellType and the entrez gene id  
 file = "/data/project/daylab/2020-JD-0044/Analysis/MAGMA/CellType\_HumanMarkerGenes.txt",  
 sep = "\t",  
 row.names = FALSE,  
 quote = FALSE,  
 col.names = TRUE)

## Generate Heatmap from MAGMA Gene-Set Analysis

#First create a dataframe in which the columns are the GWAS Phenotypes and the rows are the cell types  
gwas\_df<- as.data.frame(matrix(ncol = 11,  
 nrow = 16))  
  
colnames(gwas\_df) <- c("AD",  
 "Adhd",  
 "ASD",  
 "BIP",  
 "PD",  
 "SCZ",  
 "AgeofInit",  
 "CigsPerDay",  
 "DrinksPerWeek",  
 "SmokingCessation",  
 "SmokingInitiation")  
rownames(gwas\_df) <- c("DA-Neuron",  
 "Glut-Neuron-1",  
 "Glut-Neuron-2",  
 "Glut-Neuron-3",  
 "GABA-Neuron-1",  
 "GABA-Neuron-2",  
 "Interneuron-1",  
 "Astrocyte",  
 "Microglia",  
 "Olig-1",  
 "Olig-2",  
 "OPC-Olig-1",  
 "OPC-Olig-2",  
 "Polydendrocyte",  
 "Mural",  
 "Endothelial")  
  
#Read in the MAGMA output into the dataframe created above  
for(i in colnames(gwas\_df)){  
 x <- read.table(file = paste0("/data/project/daylab/2020-JD-0044/Analysis/MAGMA/MAGMA\_Results/10kb/",i,".gsa.out"),  
 comment.char = "#",  
 header = TRUE)  
 rownames(x) <- x$VARIABLE  
 for(l in rownames(gwas\_df)){  
 gwas\_df[l,i] <- x[l,"P"]  
 }  
}  
  
#Change the dataframe so that the columns are celltype, GWAS Phenotype, and empirical p-value  
dt <- gwas\_df %>%  
 rownames\_to\_column() %>%  
 gather(colname, value, -rowname)  
  
#Create a new column containing the -log10(empirical p-value)  
dt$log10p <- -log10(dt$value)  
  
#make a new column to input FDR values and stars  
dt$FDR <- NA  
dt$stars <- NA  
#generate a column for stars and a p.adjusted column  
for(i in 1:nrow(dt)){  
 dt[i,"FDR"] <- p.adjust(p = dt$value[i],method = "fdr",n = length(levels(VTA\_adult)))  
 if(dt[i,"FDR"] <= 0.0001){  
 dt[i,"stars"] <- "\*\*\*\*"  
 }else{  
 if( dt[i,"FDR"] <=0.001){  
 dt[i,"stars"] <- "\*\*\*"  
 }else{  
 if( dt[i,"FDR"] <=0.01){  
 dt[i,"stars"] <- "\*\*"  
 }else{  
 if( dt[i,"FDR"] <=0.05){  
 dt[i,"stars"] <- "\*"  
 }else{  
 dt[i,"stars"] <- ""  
 }  
 }  
 }  
 }  
}  
  
#Make the column and rownames factors and put in order  
dt$rowname <- factor(x = dt$rowname,levels = rev(c("DA-Neuron",  
 "Glut-Neuron-1",  
 "Glut-Neuron-2",  
 "Glut-Neuron-3",  
 "GABA-Neuron-1",  
 "GABA-Neuron-2",  
 "Interneuron-1",  
 "Astrocyte",  
 "Microglia",  
 "Olig-1",  
 "Olig-2",  
 "OPC-Olig-1",  
 "OPC-Olig-2",  
 "Polydendrocyte",  
 "Mural",  
 "Endothelial")))  
  
#Now we are going to change up the some of the names of the GWAS to something a bit more ~\*spicy\*~  
#dt$colname <- gsub(x = dt$colname,pattern = "CUD",replacement = "Cannabis Use Disorder")  
dt$colname <- gsub(x = dt$colname,pattern = "AD",replacement = "Alzheimer's Disease")  
dt$colname <- gsub(x = dt$colname,pattern = "Adhd",replacement = "ADHD")  
dt$colname <- gsub(x = dt$colname,pattern = "ASD",replacement = "Autsim Spectrum Disorder")  
dt$colname <- gsub(x = dt$colname,pattern = "BIP",replacement = "Bipolar Disorder")  
dt$colname <- gsub(x = dt$colname,pattern = "PD",replacement = "Parkinson's Disease")  
dt$colname <- gsub(x = dt$colname,pattern = "SCZ",replacement = "Schizophrenia")  
dt$colname <- gsub(x = dt$colname,pattern = "AgeofInit",replacement = "Age of Initiation\nof Regular Smoking")  
dt$colname <- gsub(x = dt$colname,pattern = "CigsPerDay",replacement = "Cigarettes per day")  
dt$colname <- gsub(x = dt$colname,pattern = "DrinksPerWeek",replacement = "Drinks Per Week")  
dt$colname <- gsub(x = dt$colname,pattern = "SmokingCessation",replacement = "Smoking Cessation")  
dt$colname <- gsub(x = dt$colname,pattern = "SmokingInitiation",replacement = "Smoking Initiation")  
#Refactor the gwas   
dt$colname <- factor(x = dt$colname,levels = c("Alzheimer's Disease",  
 "Parkinson's Disease",  
 "Bipolar Disorder",  
 "Schizophrenia",  
 "ADHD",  
 "Autsim Spectrum Disorder",  
 "Age of Initiation\nof Regular Smoking",  
 "Cigarettes per day",  
 "Drinks Per Week",  
 "Smoking Cessation",  
 "Smoking Initiation"))  
#Generate the heatmap  
HM <- ggplot(dt,aes(x = colname, y = rowname, fill = log10p)) +  
 geom\_tile(color = "black",size = 0.5) +  
 geom\_text(aes(label = stars)) +  
 scale\_fill\_gradient(low="white",high="red") +  
 labs(x = "GWAS Phenotype",  
 y = "Cell Type",  
 fill = "-log10(Empircial p)",  
 title = "10kb Window") +  
 theme\_bw() +  
 theme(plot.title = element\_text(hjust = 0.5),  
 axis.text.x = element\_text(angle = 45,hjust = 1)) +  
 NoGrid()   
  
  
ggsave(filename = "/data/project/daylab/2020-JD-0044/Analysis/MAGMA/MAGMA\_Heatmap.pdf",  
 plot = HM,  
 height = 14,  
 width = 14)  
  
  
###Now edit the dt dataframe to make a supplemental table   
dt$colname <- gsub(x = dt$colname,pattern = "Age of Initiation\nof Regular Smoking",replacement = "Age of Initiation of Regular Smoking")  
colnames(dt)[c(1:4,6)] <- c("CellType","GWAS Phenotype","Empirical p-Value","-log10(Empirical p-value)","Level of Significance")  
write.table(x = dt,  
 file = "/data/project/daylab/2020-JD-0044/Analysis/MAGMA/MAGMA\_Output\_Stats.txt",  
 col.names = TRUE,  
 row.names = FALSE,  
 quote = FALSE,  
 sep = "\t")

## Generation of Heatmaps Displaying Risk Genes for Selected GWAS

# Make heatmap with selected list of gene markers for MAGMA figure  
# Generate AD gene list  
AD\_genes <- c("Abca7",  
 "Adam10",  
 "Adamts4",  
 "Alpk2",  
 "Aph1b",  
 "Apoe",  
 "Bcam",  
 "Bcl3",  
 "Bin1",  
 "Bloc1s3",  
 "Cass4",  
 "Cblc",  
 "Cd2ap",  
 "Cd33",  
 "Ceacam20",  
 "Chrna2",  
 "Ckm",  
 "Clasrp",  
 "Clptm1",  
 "Clu",  
 "Cnn2",  
 "Cntnap2",  
 "Cr1l",  
 "Exoc3l2",  
 "Gemin7",  
 "Hesx1",  
 "Igsf23",  
 "Il17rd",  
 "Inpp5d",  
 "Kat8",  
 "Mark4",  
 "Meiosin",  
 "Ms4a2",  
 "Ms4a4a",  
 "Nectin2",  
 "Nkpd1",  
 "Picalm",  
 "Pilra",  
 "Ptk2b",  
 "Pvr",  
 "Relb",  
 "Slc24a4",  
 "Sorl1",  
 "Stag3",  
 "Tomm40",  
 "Trem2",  
 "Unc5cl",  
 "Zcwpw1")  
  
# Generate SCZ gene lists - one from PGC 2014 Nature paper   
SCZ\_genes <- c("Adamtsl3",  
 "Atp2a2",  
 "Bank1",  
 "Bcl11b",  
 "Brinp2",  
 "Cacna1c",  
 "Cacnb2",  
 "Chrna3",  
 "Clcn3",  
 "Cnot1",  
 "Cntn4",  
 "Coa8",  
 "Csmd1",  
 "Cul3",  
 "Cyp26b1",  
 "Dgki",  
 "Dpp4",  
 "Drd2",  
 "Efnb1",  
 "Etf1",  
 "Furin",  
 "Galnt10",  
 "Gatad2a",  
 "Gid4",  
 "Gigyf2",  
 "Gpm6a",  
 "Gramd1b",  
 "Grin2a",  
 "Grm3",  
 "Hcn1",  
 "Igsf9b",  
 "Immp2l",  
 "Itih3",  
 "Itih4",  
 "Kdm4a",  
 "Klc1",  
 "Man2a1",  
 "Mau2",  
 "Mphosph9",  
 "Nfatc3",  
 "Nosip",  
 "Nrgn",  
 "Nxph4",  
 "Pak6",  
 "Pja1",  
 "Plch2",  
 "Ppp1r16b",  
 "Prkd1",  
 "Prrg2",  
 "Ptgis",  
 "Ptprf",  
 "Rgs6",  
 "Rims1",  
 "Satb2",  
 "Sdccag8",  
 "Selenoh",  
 "Sf3b1",  
 "Slc38a7",  
 "Slc39a8",  
 "Slc4a10",  
 "Srpk2",  
 "Srr",  
 "Stag1",  
 "Tbc1d5",  
 "Tcf20",  
 "Tcf4",  
 "Thoc7",  
 "Tmprss5",  
 "Tmtc1",  
 "Tsnare1",  
 "Zfp536",  
 "Zfp804a")  
  
# Generate ADHD gene list  
ADHD\_genes <- c("Cadps2",  
 "Fezf1",  
 "Foxp2",  
 "Mettl15",  
 "Ptprf",  
 "Sorcs3",  
 "Spag16")  
  
# Generate Smoking Initiation gene list  
SmokingInit\_genes <- c("Acp1",  
 "Actn2",  
 "Actr1b",  
 "Adam15",  
 "Adgrb2",  
 "Adgrb3",  
 "Aff3",  
 "Alk",  
 "Ankub1",  
 "Arap2",  
 "Arid5b",  
 "Armh4",  
 "As3mt",  
 "Ascc3",  
 "Auts2",  
 "Bcl11b",  
 "Bdnf",  
 "Bend7",  
 "Birc6",  
 "Brinp1",  
 "Brwd1",  
 "Btrc",  
 "Bysl",  
 "C1qtnf7",  
 "Cabp1",  
 "Cacna1d",  
 "Cadm2",  
 "Cadps2",  
 "Cahm",  
 "Calb1",  
 "Camta1",  
 "Car10",  
 "Cbln4",  
 "Cc2d2a",  
 "Ccnd3",  
 "Celf2",  
 "Chchd3",  
 "Chd3",  
 "Chrna2",  
 "Chrna4",  
 "Cilk1",  
 "Clybl",  
 "Cntn3",  
 "Cntn4",  
 "Cpeb4",  
 "Cpne4",  
 "Cpsf6",  
 "Cpxm2",  
 "Ctnna2",  
 "Cul3",  
 "Cyb5b",  
 "Dab1",  
 "Dcc",  
 "Dgkz",  
 "Diaph3",  
 "Dnajc1",  
 "Dync1i2",  
 "Eed",  
 "Efna5",  
 "Elavl4",  
 "Elfn1",  
 "Ell",  
 "Elovl3",  
 "Elovl7",  
 "Entpd6",  
 "Ephx2",  
 "Erbb3",  
 "Erc2",  
 "Exoc4",  
 "Fam163a",  
 "Fam168a",  
 "Fat3",  
 "Fbxl17",  
 "Fbxl22",  
 "Fezf1",  
 "Fhit",  
 "Focad",  
 "Foxo3",  
 "Foxp1",  
 "Gabra6",  
 "Gapvd1",  
 "Gbf1",  
 "Ggact",  
 "Golga3",  
 "Gpm6a",  
 "Grid2",  
 "Grin2a",  
 "Hdgfl2",  
 "Hectd4",  
 "Herc1",  
 "Hhla2",  
 "Hs6st3",  
 "Ift57",  
 "Igf1r",  
 "Igsf21",  
 "Il11",  
 "Immp2l",  
 "Inpp4b",  
 "Ip6k2",  
 "Itprip",  
 "Jade2",  
 "Kansl1",  
 "Kcnj3",  
 "Kdm4b",  
 "Lingo1",  
 "Lmo3",  
 "Lmtk2",  
 "Lrguk",  
 "Lrpprc",  
 "Lrrc4c",  
 "Mad1l1",  
 "Magi2",  
 "Maip1",  
 "Maml3",  
 "Mast2",  
 "Mcrs1",  
 "Mctp1",  
 "Med27",  
 "Mettl15",  
 "Mfsd4b",  
 "Mgst1",  
 "Mllt10",  
 "Mms22l",  
 "Msra",  
 "Mterf1",  
 "Ncam1",  
 "Nfat5",  
 "Nlgn1",  
 "Nol4",  
 "Nolc1",  
 "Nrap",  
 "Nrxn1",  
 "Nrxn2",  
 "Nrxn3",  
 "Nsf",  
 "Nt5c2",  
 "Ntm",  
 "Nyap2",  
 "Olfm1",  
 "Ovol1",  
 "Pcdh15",  
 "Pcdh9",  
 "Pde4b",  
 "Pgpep1",  
 "Phc2",  
 "Phf21a",  
 "Pigl",  
 "Pipox",  
 "Plcl2",  
 "Pmfbp1",  
 "Pnmt",  
 "Polr2f",  
 "Postn",  
 "Prickle2",  
 "Prrc2b",  
 "Ptger3",  
 "Ptprd",  
 "Ptprf",  
 "Qser1",  
 "Ranbp17",  
 "Rasgrf2",  
 "Rbfox1",  
 "Rere",  
 "Rev3l",  
 "Rfx3",  
 "Rhot2",  
 "Rnf13",  
 "Robo2",  
 "Rps6ka4",  
 "Runx1t1",  
 "Ryr2",  
 "Scn2a",  
 "Sdhd",  
 "Sdk1",  
 "Sema3f",  
 "Sema6d",  
 "Setbp1",  
 "Sez6",  
 "Sh3yl1",  
 "Shtn1",  
 "Slc4a10",  
 "Smad3",  
 "Smarcc1",  
 "Smg6",  
 "Sorcs3",  
 "Sorl1",  
 "Spg7",  
 "St6galnac3",  
 "Stimate",  
 "Syt14",  
 "Syt3",  
 "Tcf20",  
 "Tenm2",  
 "Terf2ip",  
 "Thbs4",  
 "Thsd7b",  
 "Tmem18",  
 "Tmem242",  
 "Tmtc4",  
 "Tnni3k",  
 "Tnrc6a",  
 "Tox",  
 "Trim8",  
 "Trpc4",  
 "Ttc29",  
 "Ttl",  
 "Tulp4",  
 "Ubap2l",  
 "Uhrf1",  
 "Vrk2",  
 "Wdpcp",  
 "Xkr6",  
 "Xylt1",  
 "Zbtb16",  
 "Zbtb20",  
 "Zbtb46",  
 "Zcchc14",  
 "Zfhx3",  
 "Zfp423",  
 "Zic4")  
  
#Subsample dataset for nicer heatmap and set cluster order  
VTA\_adult\_small <- subset(VTA\_adult, downsample = 75)  
DefaultAssay(VTA\_adult\_small) <- "RNA"  
RevOrder <- c("DA-Neuron",  
 "Glut-Neuron-1",  
 "Glut-Neuron-2",  
 "Glut-Neuron-3",  
 "GABA-Neuron-1",  
 "GABA-Neuron-2",  
 "Interneuron-1",  
 "Astrocyte",  
 "Microglia",  
 "Olig-1",  
 "Olig-2",  
 "OPC-Olig-1",  
 "OPC-Olig-2",  
 "Polydendrocyte",  
 "Mural",  
 "Endothelial")  
levels(VTA\_adult\_small) <-RevOrder  
  
#Make heatmaps for MAGMA figure  
DoHeatmap(object = VTA\_adult\_small,features = AD\_genes,disp.min=0,disp.max=3,group.bar=TRUE,slot="data",label = TRUE,draw.lines = TRUE, raster = FALSE) +   
 theme(axis.text.y = element\_text(size = 4)) +   
 theme(axis.text.x = element\_text(size = 5)) +   
 scale\_fill\_gradientn(colors = c("lightgray", "red"))  
  
DoHeatmap(object = VTA\_adult\_small,features = SCZ\_genes,disp.min=0,disp.max=3,group.bar=TRUE,slot="data",label = TRUE,draw.lines = TRUE, raster = FALSE) +   
 theme(axis.text.y = element\_text(size = 4)) +   
 theme(axis.text.x = element\_text(size = 5)) +   
 scale\_fill\_gradientn(colors = c("lightgray", "red"))  
  
DoHeatmap(object = VTA\_adult\_small,features = ADHD\_genes,disp.min=0,disp.max=3,group.bar=TRUE,slot="data",label = TRUE,draw.lines = TRUE, raster = FALSE) +   
 theme(axis.text.y = element\_text(size = 4)) +   
 theme(axis.text.x = element\_text(size = 5)) +   
 scale\_fill\_gradientn(colors = c("lightgray", "red"))  
  
DoHeatmap(object = VTA\_adult\_small,features = SmokingInit\_genes,disp.min=0,disp.max=3,group.bar=TRUE,slot="data",label = TRUE,draw.lines = TRUE, raster = FALSE) +   
 theme(axis.text.y = element\_text(size = 4)) +   
 theme(axis.text.x = element\_text(size = 5)) +   
 scale\_fill\_gradientn(colors = c("lightgray", "red"))

## Generation of Violin Plots and Heatmaps for Supplementary Figure 1

## Heatmap and count/gene violin plots for Supplementary Figure 1   
# Find cell-specific markers and make heatmap  
DefaultAssay(VTA\_adult) <- "integrated"  
VTA\_markers <- FindAllMarkers(VTA\_adult, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)  
Genes\_Use <- VTA\_markers %>% group\_by(cluster) %>% top\_n(n = 10, wt = avg\_logFC)  
#Subsample for nicer heatmap  
VTA\_adult\_small <- subset(VTA\_adult, downsample = 75)  
DefaultAssay(VTA\_adult\_small) <- "integrated"  
DoHeatmap(object = VTA\_adult\_small,features = Genes\_Use$gene,label = FALSE,draw.lines = TRUE, raster = FALSE) +   
 theme(axis.text.y = element\_text(size = 4)) +   
 scale\_fill\_gradientn(colors = c("steelblue4", "white", "red"))  
# Plot nFeature\_RNA and nCount\_RNA by CellType  
levels(VTA\_adult) <- NewOrder  
Idents(VTA\_adult) <- VTA\_adult$CellType  
p1 <- ggplot(VTA\_adult@meta.data, aes(Idents(VTA\_adult), nFeature\_RNA, colour = CellType, label=FALSE)) +   
 geom\_violin() +   
 theme\_bw() +  
 theme(legend.position = "none") +   
 scale\_y\_log10() +  
 coord\_flip()  
p2 <- ggplot(VTA\_adult@meta.data, aes(Idents(VTA\_adult), nCount\_RNA, colour = CellType, label=FALSE)) +   
 geom\_violin() +   
 theme\_bw() +  
 theme(legend.position = "none") +   
 scale\_y\_log10() +  
 coord\_flip()  
plot\_grid(p1, p2, labels = c('a', 'b'))

## Session Info

sessionInfo()

## R version 4.0.2 (2020-06-22)  
## Platform: x86\_64-pc-linux-gnu (64-bit)  
## Running under: Ubuntu 16.04.6 LTS  
##   
## Matrix products: default  
## BLAS: /usr/lib/openblas-base/libblas.so.3  
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so  
##   
## locale:  
## [1] LC\_CTYPE=en\_US.UTF-8 LC\_NUMERIC=C   
## [3] LC\_TIME=en\_US.UTF-8 LC\_COLLATE=en\_US.UTF-8   
## [5] LC\_MONETARY=en\_US.UTF-8 LC\_MESSAGES=en\_US.UTF-8   
## [7] LC\_PAPER=en\_US.UTF-8 LC\_NAME=C   
## [9] LC\_ADDRESS=C LC\_TELEPHONE=C   
## [11] LC\_MEASUREMENT=en\_US.UTF-8 LC\_IDENTIFICATION=C   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] dplyr\_1.0.2 tidyr\_1.1.2 tibble\_3.0.4 Seurat\_3.2.2   
## [5] ggrepel\_0.8.2 cowplot\_1.1.0 ggplot2\_3.3.2 reldist\_1.6-6   
## [9] pheatmap\_1.0.12 data.table\_1.13.0  
##   
## loaded via a namespace (and not attached):  
## [1] Rtsne\_0.15 colorspace\_1.4-1 deldir\_0.1-29   
## [4] ellipsis\_0.3.1 ggridges\_0.5.2 htmlTable\_2.1.0   
## [7] base64enc\_0.1-3 rstudioapi\_0.11 spatstat.data\_1.4-3   
## [10] farver\_2.0.3 leiden\_0.3.3 listenv\_0.8.0   
## [13] RSpectra\_0.16-0 codetools\_0.2-16 splines\_4.0.2   
## [16] knitr\_1.30 polyclip\_1.10-0 Formula\_1.2-4   
## [19] jsonlite\_1.7.1 ica\_1.0-2 cluster\_2.1.0   
## [22] png\_0.1-7 uwot\_0.1.8 shiny\_1.5.0   
## [25] sctransform\_0.3.1 compiler\_4.0.2 httr\_1.4.2   
## [28] backports\_1.1.10 Matrix\_1.2-18 fastmap\_1.0.1   
## [31] lazyeval\_0.2.2 later\_1.1.0.1 htmltools\_0.5.0   
## [34] tools\_4.0.2 rsvd\_1.0.3 igraph\_1.2.6   
## [37] gtable\_0.3.0 glue\_1.4.2 RANN\_2.6.1   
## [40] reshape2\_1.4.4 rappdirs\_0.3.1 Rcpp\_1.0.5   
## [43] spatstat\_1.64-1 vctrs\_0.3.4 nlme\_3.1-149   
## [46] lmtest\_0.9-38 xfun\_0.18 stringr\_1.4.0   
## [49] globals\_0.13.1 mime\_0.9 miniUI\_0.1.1.1   
## [52] lifecycle\_0.2.0 irlba\_2.3.3 goftest\_1.2-2   
## [55] future\_1.19.1 MASS\_7.3-53 zoo\_1.8-8   
## [58] scales\_1.1.1 promises\_1.1.1 spatstat.utils\_1.17-0  
## [61] parallel\_4.0.2 RColorBrewer\_1.1-2 yaml\_2.2.1   
## [64] reticulate\_1.16 pbapply\_1.4-3 gridExtra\_2.3   
## [67] rpart\_4.1-15 latticeExtra\_0.6-29 stringi\_1.5.3   
## [70] checkmate\_2.0.0 rlang\_0.4.8 pkgconfig\_2.0.3   
## [73] matrixStats\_0.57.0 evaluate\_0.14 lattice\_0.20-41   
## [76] tensor\_1.5 ROCR\_1.0-11 purrr\_0.3.4   
## [79] labeling\_0.3 patchwork\_1.0.1 htmlwidgets\_1.5.2   
## [82] tidyselect\_1.1.0 RcppAnnoy\_0.0.16 plyr\_1.8.6   
## [85] magrittr\_1.5 R6\_2.4.1 generics\_0.0.2   
## [88] Hmisc\_4.4-1 pillar\_1.4.6 foreign\_0.8-80   
## [91] withr\_2.3.0 mgcv\_1.8-33 fitdistrplus\_1.1-1   
## [94] abind\_1.4-5 survival\_3.2-7 nnet\_7.3-14   
## [97] future.apply\_1.6.0 crayon\_1.3.4 KernSmooth\_2.23-17   
## [100] plotly\_4.9.2.1 rmarkdown\_2.7 jpeg\_0.1-8.1   
## [103] grid\_4.0.2 digest\_0.6.26 xtable\_1.8-4   
## [106] httpuv\_1.5.4 munsell\_0.5.0 viridisLite\_0.3.0