A dopamine-induced gene expression signature regulates neuronal function and cocaine response -Primary Striatal Culture Analysis

02/02/2020

## Load Libraries

To begin, all packages/libraries that will be needed for the analysis will be loaded

suppressPackageStartupMessages(library("dplyr"))  
suppressPackageStartupMessages(library("Seurat"))  
suppressPackageStartupMessages(library("svMisc"))  
suppressPackageStartupMessages(library("cowplot"))  
suppressPackageStartupMessages(library("ggplot2"))  
suppressPackageStartupMessages(library("pheatmap"))  
suppressPackageStartupMessages(library("reshape2"))  
suppressPackageStartupMessages(library("gridExtra"))  
suppressPackageStartupMessages(library("RColorBrewer"))

## Workflow

The adult analysis includes additional information regarding details about our analysis. This document will include essential code for the culture analysis.

Veh <- Read10X(data.dir = "~/Bioinformatics/JD0034/95\_GTF/Veh\_Output/")  
DA <- Read10X(data.dir = "~/Bioinformatics/JD0034/95\_GTF/DA\_Output/")  
SKF <- Read10X(data.dir = "~/Bioinformatics/JD0034/95\_GTF/SKF\_Output/")  
KCl <- Read10X(data.dir = "~/Bioinformatics/JD0034/95\_GTF/KCL\_Output/")  
  
#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.   
Veh <- CreateSeuratObject(counts = Veh,min.cells = 1,min.features = 1)

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

DA <- CreateSeuratObject(counts = DA,min.cells = 1,min.features = 1)

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

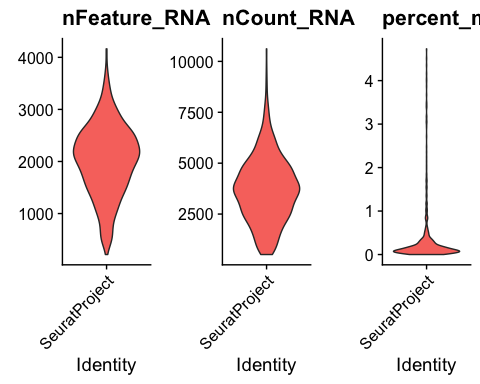
SKF <- CreateSeuratObject(counts = SKF,min.cells = 1,min.features = 1)

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

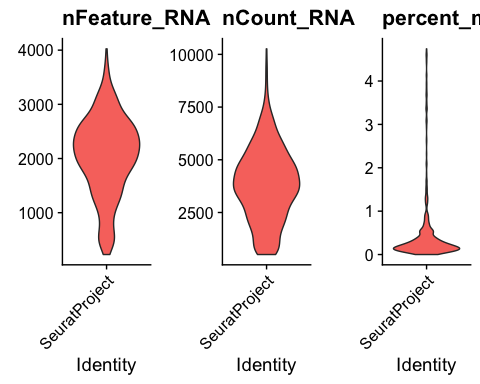
KCl <- CreateSeuratObject(counts = KCl,min.cells = 1,min.features = 1)

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

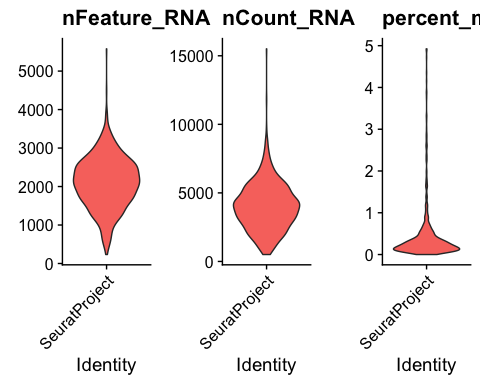
#3601 nuclei total   
  
#Identify the percentage of reads mapping to mitochondrial genes   
Veh <- PercentageFeatureSet(Veh, pattern = "^Mt-", col.name = "percent\_mito")  
DA <- PercentageFeatureSet(DA, pattern = "^Mt-", col.name = "percent\_mito")  
SKF <- PercentageFeatureSet(SKF, pattern = "^Mt-", col.name = "percent\_mito")  
KCl <- PercentageFeatureSet(KCl, pattern = "^Mt-", col.name = "percent\_mito")  
  
#Subset data to have greater than 200 features and less than 5% of reads mapping to mitochondrial genes   
Veh <- subset(x = Veh, subset = nFeature\_RNA > 200 & percent\_mito < 5)   
DA <- subset(x = DA, subset = nFeature\_RNA > 200 & percent\_mito < 5)   
SKF <- subset(x = SKF, subset = nFeature\_RNA > 200 & percent\_mito < 5)   
KCl <- subset(x = KCl, subset = nFeature\_RNA > 200 & percent\_mito < 5)   
  
  
# #Replot to visualize the QC metrics following the subset   
VlnPlot(Veh, features = c("nFeature\_RNA", "nCount\_RNA", "percent\_mito"), ncol = 3,pt.size = 0)



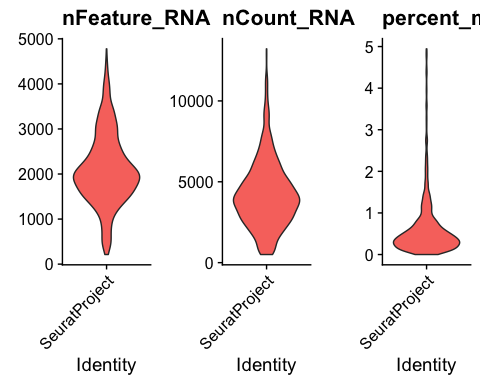
VlnPlot(DA, features = c("nFeature\_RNA", "nCount\_RNA", "percent\_mito"), ncol = 3,pt.size = 0)



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



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



Veh <- NormalizeData(Veh, normalization.method = "LogNormalize", scale.factor = 10000)  
DA <- NormalizeData(DA, normalization.method = "LogNormalize", scale.factor = 10000)  
SKF <- NormalizeData(SKF, normalization.method = "LogNormalize", scale.factor = 10000)  
KCl <- NormalizeData(KCl, normalization.method = "LogNormalize", scale.factor = 10000)  
  
Veh <- FindVariableFeatures(Veh, selection.method = "vst", nfeatures = 2000)  
DA <- FindVariableFeatures(DA, selection.method = "vst", nfeatures = 2000)  
SKF <- FindVariableFeatures(SKF, selection.method = "vst", nfeatures = 2000)  
KCl <- FindVariableFeatures(KCl, selection.method = "vst", nfeatures = 2000)  
  
#Integrate all datasets   
Veh$Stim <- "Veh"  
DA$Stim <- "DA"  
SKF$Stim <- "SKF"  
KCl$Stim <- "KCl"  
  
  
#Now integrate the data  
Culture\_log <- FindIntegrationAnchors(object.list = list(Veh,DA,SKF,KCl), dims = 1:10)

## 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 2037 anchors

## Filtering anchors

## Retained 1973 anchors

## Extracting within-dataset neighbors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 2163 anchors

## Filtering anchors

## Retained 2067 anchors

## Extracting within-dataset neighbors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 2162 anchors

## Filtering anchors

## Retained 2118 anchors

## Extracting within-dataset neighbors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 2237 anchors

## Filtering anchors

## Retained 2073 anchors

## Extracting within-dataset neighbors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 2173 anchors

## Filtering anchors

## Retained 2046 anchors

## Extracting within-dataset neighbors

## Running CCA

## Merging objects

## Finding neighborhoods

## Finding anchors

## Found 2336 anchors

## Filtering anchors

## Retained 2185 anchors

## Extracting within-dataset neighbors

Culture\_log <- IntegrateData(anchorset = Culture\_log,dims = 1:10)

## Merging dataset 2 into 3

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

## Merging dataset 1 into 4

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

## Merging dataset 3 2 into 4 1

## Extracting anchors for merged samples

## Finding integration vectors

## Finding integration vector weights

## Integrating data

DefaultAssay(Culture\_log) <- "integrated"  
  
# Run the standard workflow for visualization and clustering  
Culture\_log <- ScaleData(Culture\_log,verbose = FALSE)  
Culture\_log <- RunPCA(Culture\_log,npcs = 10 ,verbose = FALSE) #Compute 50 npcs by default  
# Dimensionality reduction and Clustering  
Culture\_log <- RunUMAP(Culture\_log,reduction = "pca", dims = 1:10)  
Culture\_log <- FindNeighbors(Culture\_log, reduction = "pca", dims = 1:10)

## Computing nearest neighbor graph

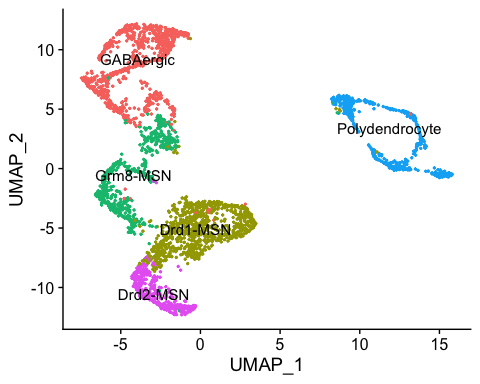
## Computing SNN

Culture\_log <- FindClusters(Culture\_log, resolution = 0.1)

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck  
##   
## Number of nodes: 3339  
## Number of edges: 104965  
##   
## Running Louvain algorithm...  
## Maximum modularity in 10 random starts: 0.9656  
## Number of communities: 5  
## Elapsed time: 0 seconds

#Add the cell-type identities   
Culture\_log <- RenameIdents(object = Culture\_log,  
 "0" = "GABAergic",  
 "1" = "Drd1-MSN",  
 "2" = "Grm8-MSN",  
 "3" = "Polydendrocyte",  
 "4" = "Drd2-MSN")  
#Make a metadata column for Celltype  
Culture\_log$CellType <- Idents(Culture\_log)  
  
#Plot UMAP  
DimPlot(object = Culture\_log,reduction = "umap",label = TRUE) + NoLegend()

## Warning: Using `as.character()` on a quosure is deprecated as of rlang 0.3.0.  
## Please use `as\_label()` or `as\_name()` instead.  
## This warning is displayed once per session.



sessionInfo()

## R version 3.6.0 (2019-04-26)  
## Platform: x86\_64-apple-darwin15.6.0 (64-bit)  
## Running under: macOS Mojave 10.14.6  
##   
## Matrix products: default  
## BLAS: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib  
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib  
##   
## locale:  
## [1] en\_US.UTF-8/en\_US.UTF-8/en\_US.UTF-8/C/en\_US.UTF-8/en\_US.UTF-8  
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] RColorBrewer\_1.1-2 gridExtra\_2.3 reshape2\_1.4.3 pheatmap\_1.0.12   
## [5] ggplot2\_3.2.1 cowplot\_1.0.0 svMisc\_1.1.0 Seurat\_3.0.2   
## [9] dplyr\_0.8.3   
##   
## loaded via a namespace (and not attached):  
## [1] tsne\_0.1-3 nlme\_3.1-142 bitops\_1.0-6   
## [4] httr\_1.4.1 sctransform\_0.2.0 tools\_3.6.0   
## [7] backports\_1.1.5 R6\_2.4.1 irlba\_2.3.3   
## [10] KernSmooth\_2.23-16 lazyeval\_0.2.2 colorspace\_1.4-1   
## [13] withr\_2.1.2 npsurv\_0.4-0 tidyselect\_0.2.5   
## [16] compiler\_3.6.0 plotly\_4.9.1 labeling\_0.3   
## [19] caTools\_1.17.1.3 scales\_1.0.0 lmtest\_0.9-37   
## [22] ggridges\_0.5.1 pbapply\_1.4-2 stringr\_1.4.0   
## [25] digest\_0.6.22 rmarkdown\_1.17 R.utils\_2.9.0   
## [28] pkgconfig\_2.0.3 htmltools\_0.4.0 bibtex\_0.4.2   
## [31] htmlwidgets\_1.5.1 rlang\_0.4.1 zoo\_1.8-6   
## [34] jsonlite\_1.6 ica\_1.0-2 gtools\_3.8.1   
## [37] R.oo\_1.23.0 magrittr\_1.5 Matrix\_1.2-17   
## [40] Rcpp\_1.0.3 munsell\_0.5.0 ape\_5.3   
## [43] reticulate\_1.13 lifecycle\_0.1.0 R.methodsS3\_1.7.1   
## [46] stringi\_1.4.3 yaml\_2.2.0 gbRd\_0.4-11   
## [49] MASS\_7.3-51.4 gplots\_3.0.1.1 Rtsne\_0.15   
## [52] plyr\_1.8.4 grid\_3.6.0 parallel\_3.6.0   
## [55] gdata\_2.18.0 listenv\_0.7.0 ggrepel\_0.8.1   
## [58] crayon\_1.3.4 lattice\_0.20-38 splines\_3.6.0   
## [61] SDMTools\_1.1-221.1 zeallot\_0.1.0 knitr\_1.26   
## [64] pillar\_1.4.2 igraph\_1.2.4.1 future.apply\_1.3.0   
## [67] codetools\_0.2-16 glue\_1.3.1 evaluate\_0.14   
## [70] lsei\_1.2-0 metap\_1.1 data.table\_1.12.8   
## [73] vctrs\_0.2.0 png\_0.1-7 Rdpack\_0.11-0   
## [76] gtable\_0.3.0 RANN\_2.6.1 purrr\_0.3.3   
## [79] tidyr\_1.0.0 future\_1.15.0 assertthat\_0.2.1   
## [82] xfun\_0.11 rsvd\_1.0.2 survival\_3.1-7   
## [85] viridisLite\_0.3.0 tibble\_2.1.3 cluster\_2.1.0   
## [88] globals\_0.12.4 fitdistrplus\_1.0-14 ROCR\_1.0-7