

```

# Please install these packages before the library command. Seurat installation instruction is
https://satijalab.org/seurat/articles/install\_v5
# ggplot2 and tidyverse installation is here https://ggplot2.tidyverse.org/
# Use code install.packages("gridExtra") to install gridExtra

#Load the installed packages by using the following commands. It needs to be done every time you open a new R section.
library(Seurat)
library(ggplot2)
library(tidyverse)
library(gridExtra)

#Set your own working directory. All input and output files will come from/go to the working directory unless specified.
setwd("~/Desktop")

#Load the unanalyzed rds files that already went through QC using the following commands. The files are provided in the
folder.
#glass_wpp is gl mutant, and wpp is wild type. These data are generated from eye disc cells at white prepupal stages

glass <- readRDS("~/Desktop/glass_wpp.rds")
wppgl<- readRDS("~/Desktop/wppgl.rds")

#The glass and wppgl dataset will be merged to enable data integration in later steps. Commands for merging:
#1 Add a column to indicate glass mutant data or wt data
new <-rep("gl mutant", 10225)
glass@meta.data$condition <-new
new2 <-rep('wt',26669)
wpp@meta.data$condition <-new2
#2 Merge
wpp_gl_combined = merge(wpp, y = glass, add.cell.ids = c("wt", "gl_mutant"), project = "gene_condition", merge.data = TRUE)

#Run preparation for PCA, and umap visualization of merged dataset prior to data integration
wpp_gl_combined <-NormalizeData(wpp_gl_combined)
wpp_gl_combined <- FindVariableFeatures(wpp_gl_combined)
wpp_gl_combined <-ScaleData(wpp_gl_combined)
wpp_gl_combined <-RunPCA(wpp_gl_combined)
ElbowPlot(wpp_gl_combined)
wpp_gl_combined <-RunUMAP(wpp_gl_combined, dims = 1:20, reduction = 'pca')
#You will get the umap prior to integration after running the following command
DimPlot(wpp_gl_combined, reduction = 'umap', group.by = 'condition')

#Now seurat integration procedures
#Split the merged dataset based on condition
obj_list <- SplitObject(wpp_gl_combined, split.by = "condition")
for(i in 1:length(obj_list)){
  obj_list[[i]]<-NormalizeData(object = obj_list[[i]])
  obj_list[[i]]<- FindVariableFeatures(object = obj_list[[i]])
}
#start integration process
features <-SelectIntegrationFeatures(object.list = obj_list)
#It will take roughly 15min 13s to finish FindIntegrationAnchors function, this step takes the longest in this code. If you
install R on your own computer, it may run faster than using an online server/co-lab.
anchors <-FindIntegrationAnchors(object.list = obj_list, anchor.features = features)
#Now making a new rds object that has the integrated data. This may take a minute
wppgl <- IntegrateData(anchorset = anchors)
#Processing of integrated data
wppgl <-ScaleData(object = wppgl)
wppgl <-RunPCA(object = wppgl)
wppgl <-FindNeighbors(wppgl, dim = 1:50)
#Resolution here is finely tuned in this paper here at 0.55 to balance the cluster sizes to match the physiological presenting
cell type numbers.
#Resolution is also set to make consistent cluster output with the previous publication https://doi.org/10.1038/s41467-023-43037-0 by Raja. et al
wppgl <-FindClusters (wppgl, resolution = c(0.5, 0.55, 0.6))
wppgl <-RunUMAP(object = wppgl, dim = 1:50, seed.use =12)
Idents(wppgl) <- "integrated_snn_res.0.55"

#Generate a graph to show dimension plot after integration. Please use the Zoom icon in the Plots section to plot the graph
bigger. You can maximize the Plot Zoom window.
#If you don't know how to zoom you may want to change the element text to a smaller number also reduce the pt.size
p <-DimPlot(wppgl, reduction = 'umap', split.by = "condition",label= "False", pt.size = 1) & theme(axis.text =
element_text(size = 30),axis.title.x = element_text(size = 30), axis.title = element_text(size = 30), plot.title =
element_text(size = 30), plot.tag = element_text(size = 30), text = element_text(size = 30, face = "bold"))
LabelClusters(p, id = "ident", size =12, fontface = "bold", split.by = "condition")

#The cluster names are automatically generated based on cluster cell number sizes, 0 being the largest and 24 the smallest

```

```

#Here we partially renamed clusters to match the wild type that is published by Raja. et al https://doi.org/10.1038/s41467-023-43037-0
new.cluster.ids <- c("0", "1", "2", "3", "4", "Furrow", "6", "Preproneural", "8", "9", "10", "11", "SMW", "R1/6", "diff.PR",
                    "15", "R8", "R3/4", "18", "R7", "R2/5",
                    "21", "22", "23", "24")
names(new.cluster.ids) <- levels(wppgl)
wppgl <- RenameIdents(object = wppgl, new.cluster.ids)

#Replot Dimplot with new cluster ID
# "Warning messages: 1: ggrepel: 6 unlabeled data points (too many overlaps). Consider increasing max.overlaps 2: ggrepel: 4
unlabeled data points (too many overlaps). Consider increasing max.overlaps
#may occur, but they are due to the big text sizes. They shouldn't affect visualization.
p <- DimPlot(wppgl, reduction = 'umap', split.by = "condition", label = "False", pt.size = 1) & theme(axis.text =
element_text(size = 30), axis.title.x = element_text(size = 30), axis.title = element_text(size = 30), plot.title =
element_text(size = 30), plot.tag = element_text(size = 30), text = element_text(size = 30, face = "bold"))
LabelClusters(p, id = "ident", size = 12, fontface = "bold", split.by = "condition")

#You can also plot one without label to rid of the warning message, and manually add the cell types later.
DimPlot(wppgl, reduction = 'umap', split.by = "condition", label = "False", pt.size = 1) & theme(axis.text = element_text(size =
30), axis.title.x = element_text(size = 30), axis.title = element_text(size = 30), plot.title = element_text(size = 30),
plot.tag = element_text(size = 30), text = element_text(size = 30, face = "bold"))

#The cluster names are verified with FindMarkers function to tell you what genes are specifically enriched in one cluster
compared to the other selected clusters. For example:
#You can see gene sens pop out in R8 compared to all the other photoreceptor types.
markersexample <- FindMarkers(object = wppgl, ident.1 = 'R8', ident.2 = c("R7", "R1/6", "R2/5", "R3/4"))
head(x = markersexample)

#FindMarkers function can also be used in a way to let you know what are the genes that changed most in their expression level
in gl mutant compared to wild type, in the cluster you specify.
#These marker lists can be exported into an excel readable format, For example:

marker2_5 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident =
"R2/5")
write.csv(marker2_5, file = "R2_5.csv")

marker3_4 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident =
"R3/4")
write.csv(marker3_4, file = "R3_4.csv")

marker8 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident =
"R8")
write.csv(marker8, file = "R8.csv")

marker1_6 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident =
"R1/6")
write.csv(marker1_6, file = "R1_6.csv")

marker7 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident =
"R7")
write.csv(marker7, file = "R7.csv")

marker11 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident =
"11")
write.csv(marker11, file = "11.csv")

marker21 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident =
"21")
write.csv(marker21, file = "21.csv")

marker6 <- FindMarkers(wppgl, ident.1 = "wt", ident.2 = "gl mutant", verbose = TRUE, group.by = "condition", subset.ident = "6")
write.csv(marker6, file = "6.csv")

```