

# WB01\_sc\_KB\_to\_Seurat\_object\_3\_8\_21

June 30, 2022

```
[1]: #THIS SCRIPT WILL PERFORM EMPTYDROPS ANALYSIS, SEURAT OBJECT CREATION, AND
      ↪MULTIPLSET IDENTIFICATION

suppressMessages(library(BUSpaRse))
suppressMessages(library(Matrix))
suppressMessages(library(tidyverse))
suppressMessages(library(Seurat))
suppressMessages(library(DropletUtils))
suppressMessages(library(DoubletFinder))

proto_genes=read.csv("../data/bulk_data/protoplasting.csv")
proto_list=as.character(proto_genes[abs(proto_genes$logFC) > 2,]$genes)

# Slightly modified from BUSpaRse, just to avoid installing a few dependencies
↪not used here
read_count_output <- function(dir, name) {
  dir <- normalizePath(dir, mustWork = TRUE)
  m <- readMM(paste0(dir, "/", name, ".mtx"))
  m <- Matrix::t(m)
  m <- as(m, "dgCMatrix")
  # The matrix read has cells in rows
  ge <- ".genes.txt"
  genes <- readLines(file(paste0(dir, "/", name, ge)))
  barcodes <- readLines(file(paste0(dir, "/", name, ".barcodes.txt")))
  colnames(m) <- barcodes
  rownames(m) <- genes
  return(m)
}
```

```
[2]: sessionInfo()
```

R version 3.6.3 (2020-02-29)

Platform: x86\_64-conda\_cos6-linux-gnu (64-bit)

Running under: Ubuntu 20.04.2 LTS

Matrix products: default

BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r\_3/lib/libopenblas-p-r0.3.9.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] parallel stats4 stats graphics grDevices utils datasets
[8] methods base
```

other attached packages:

```
[1] DoubletFinder_2.0.3      DropletUtils_1.6.1
[3] SingleCellExperiment_1.8.0 SummarizedExperiment_1.16.1
[5] DelayedArray_0.12.3      BiocParallel_1.20.1
[7] matrixStats_0.61.0      Biobase_2.46.0
[9] GenomicRanges_1.38.0     GenomeInfoDb_1.22.1
[11] IRanges_2.20.2           S4Vectors_0.24.4
[13] BiocGenerics_0.32.0      Seurat_3.1.5
[15] forcats_0.5.0            stringr_1.4.0
[17] dplyr_1.0.7              purrr_0.3.4
[19] readr_1.4.0              tidyr_1.1.3
[21] tibble_3.1.6             ggplot2_3.3.5
[23] tidyverse_1.3.0          Matrix_1.2-18
[25] BUSpaRse_1.0.0
```

loaded via a namespace (and not attached):

```
[1] readxl_1.3.1             uuid_0.1-4              backports_1.4.1
[4] BiocFileCache_1.10.2     plyr_1.8.6              igraph_1.2.5
[7] repr_1.1.0               lazyeval_0.2.2          splines_3.6.3
[10] listenv_0.8.0            digest_0.6.29           ensemblDb_2.10.2
[13] htmltools_0.5.0         fansi_0.5.0             magrittr_2.0.1
[16] memoise_1.1.0           BSgenome_1.54.0         cluster_2.1.2
[19] ROCR_1.0-11             limma_3.42.2            globals_0.12.5
[22] Biostrings_2.54.0       modelr_0.1.6            RcppParallel_5.1.5
[25] R.utils_2.9.2           askpass_1.1             prettyunits_1.1.1
[28] colorspace_2.0-2        blob_1.2.1              rvest_0.3.5
[31] rappdirs_0.3.3          ggrepel_0.9.1           haven_2.3.1
[34] crayon_1.4.2            RCurl_1.98-1.2          jsonlite_1.7.2
[37] zeallot_0.1.0           survival_3.1-12         zoo_1.8-8
[40] ape_5.3                 glue_1.6.0              gtable_0.3.0
[43] zlibbioc_1.32.0         XVector_0.26.0          leiden_0.3.9
[46] plyranges_1.6.10        Rhdf5lib_1.8.0          future.apply_1.6.0
[49] HDF5Array_1.14.4        scales_1.1.1            edgeR_3.28.1
[52] DBI_1.1.2               Rcpp_1.0.7              viridisLite_0.4.0
[55] progress_1.2.2          dqrng_0.3.0            reticulate_1.16
[58] rsvd_1.0.3             bit_4.0.4               tsne_0.1-3
```

[61]	htmlwidgets_1.5.1	httr_1.4.2	RColorBrewer_1.1-2
[64]	ellipsis_0.3.2	ica_1.0-2	R.methodsS3_1.8.1
[67]	pkgconfig_2.0.3	XML_3.99-0.3	uwot_0.1.8
[70]	dbplyr_1.4.2	locfit_1.5-9.4	utf8_1.2.2
[73]	reshape2_1.4.4	tidyselect_1.1.1	rlang_0.4.12
[76]	AnnotationDbi_1.48.0	munsell_0.5.0	cellranger_1.1.0
[79]	tools_3.6.3	cli_3.1.0	generics_0.1.1
[82]	RSQLite_2.2.0	broom_0.5.5	ggribbles_0.5.2
[85]	evaluate_0.14	bit64_0.9-7	fs_1.5.2
[88]	fitdistrplus_1.1-1	RANN_2.6.1	AnnotationFilter_1.10.0
[91]	pbapply_1.5-0	future_1.18.0	nlme_3.1-147
[94]	R.oo_1.23.0	xml2_1.3.1	biomaRt_2.42.1
[97]	compiler_3.6.3	rstudioapi_0.13	png_0.1-7
[100]	plotly_4.9.2.1	curl_4.3	reprex_0.3.0
[103]	stringi_1.7.6	GenomicFeatures_1.38.2	lattice_0.20-45
[106]	IRdisplay_0.7.0	ProtGenerics_1.18.0	vcrrs_0.3.8
[109]	pillar_1.6.4	lifecycle_1.0.1	lmtest_0.9-38
[112]	RcppAnnoy_0.0.19	data.table_1.14.2	cowplot_1.1.1
[115]	bitops_1.0-7	irlba_2.3.5	patchwork_1.1.1
[118]	rtracklayer_1.46.0	R6_2.5.1	gridExtra_2.3
[121]	KernSmooth_2.23-20	codetools_0.2-18	MASS_7.3-54
[124]	assertthat_0.2.1	rhdf5_2.30.1	openssl_1.4.6
[127]	withr_2.4.3	sctransform_0.2.1	GenomicAlignments_1.22.1
[130]	Rsamtools_2.2.3	GenomeInfoDbData_1.2.2	hms_1.1.0
[133]	grid_3.6.3	IRkernel_1.1	Rtsne_0.15
[136]	pbdZMQ_0.3-6	lubridate_1.7.8	base64enc_0.1-3

```
[2]: files <- list.files(path="../data/scKB_outs/", full.names=TRUE, recursive=FALSE)
files_base = list.files(path="../data/scKB_outs/", full.names=FALSE,
↪recursive=FALSE)
```

```
[ ]: #prevent warnings from printing
defaultW <- getOption("warn")
options(warn = -1)

seu_list = list()

#loop through all files and perform empty drops quantification and make seuat_
↪objects
for (i in 1:length(files)){
  sample = files[i]

  #read in spliced matrix and retrain only Arabidopsis gene counts (important_
↪for species mixing experiments)
  spliced = read_count_output(sample, "spliced")
```

```

    spliced = spliced[grepl("AT",unlist(spliced@Dimnames[1])), fixed=TRUE],,
↳drop=FALSE]

    #read in unspliced matrix and retrain only Arabidopsis gene counts
    unspliced = read_count_output(sample, "unspliced")
    unspliced = unspliced[grepl("AT",unlist(unspliced@Dimnames[1])),
↳fixed=TRUE],, drop=FALSE]

    #Find barcodes identified in both spliced and unspliced count matrices
    shared = intersect(colnames(spliced), colnames(unspliced))

    #filter to barcodes present in both spliced and unspliced matrices
    spliced = spliced[,shared]
    unspliced = unspliced[,shared]

    combined = spliced + unspliced

    #run empty drops on combined count matrix
    empty_drops = emptyDrops(combined[grepl(pattern = "AT[1-5]",
↳unlist(spliced@Dimnames[1])),, drop=FALSE], ignore = 500, lower = 300)

    #We take all cells from spliced and unspliced that were called by
↳emptydrops, and then we sum them into combined matrix
    shared = intersect(intersect(colnames(spliced), colnames(unspliced)),
↳rownames(empty_drops[!is.na(empty_drops$FDR) & empty_drops$FDR < .001,]))

    spliced = spliced[,shared]
    unspliced = unspliced[,shared]

    combined = spliced + unspliced

    #create seurat object
    seu_obj <- CreateSeuratObject(combined, min.cells = 3)

    #add spliced and unspliced as assays
    spliced_assay <- CreateAssayObject(counts = spliced)
    unspliced_assay <- CreateAssayObject(counts = unspliced)

    seu_obj[["spliced"]] = spliced_assay
    seu_obj[["unspliced"]] = unspliced_assay

    #mito and plastid read percent
    seu_obj = PercentageFeatureSet(seu_obj, pattern = "ATM", col.name =
↳"percent.mito", assay = "RNA")
    seu_obj = PercentageFeatureSet(seu_obj, pattern = "ATC", col.name =
↳"percent.cp", assay = "RNA")

```

```

#doublet finder
df_seu <- NormalizeData(seu_obj)
df_seu <- FindVariableFeatures(df_seu, selection.method = "vst", nfeatures_
↪= 2000)
df_seu <- ScaleData(df_seu)
df_seu <- RunPCA(df_seu)
df_seu <- RunTSNE(df_seu, dims = 1:15)

#simple approximate expected doublet rate based on equation from 10x data:
↪doublet_percent = .004/500 * #_cells
nExp_poi <- round((0.004 /500*(nrow(df_seu@meta.data))) * nrow(df_seu@meta.
↪data))

sweep.sweep.df_seu <- paramSweep_v3(df_seu, PCs = 1:15, sct = FALSE)
sweep.stats_df_seu <- summarizeSweep(sweep.sweep.df_seu, GT = FALSE)
bcmvn_sweep.df_seu <- find.pK(sweep.stats_df_seu)

pK = double(bcmvn_sweep.df_seu[max(bcmvn_sweep.
↪df_seu$BCmetric)==bcmvn_sweep.df_seu$BCmetric,2])
df_seu <- doubletFinder_v3(df_seu, PCs = 1:15, pN = 0.25, pK = pK, nExp =
↪nExp_poi, reuse.pANN = FALSE, sct = FALSE)

seu_obj <- subset(seu_obj, subset = (percent.mito < 10) & df_seu@meta.
↪data[,dim(df_seu@meta.data)[2]] == "Singlet")

#set original experiment
seu_obj@meta.data$orig.ident = files_base[i]

#set genotype
if (files_base[i] %in% c("sc_101", "sc_103", "sc_26_combined", "sc_67",
↪"sc_69")) {
  seu_obj@meta.data$geno = "WT"
}
else {
  seu_obj@meta.data$geno = "mutant"
}

#set experiment
if (files_base[i] %in% c("sc_101", "sc_102", "sc_103", "sc_104", "sc_69",
↪"sc_70")) {
  seu_obj@meta.data$experiment = "sorted"
}

```

```

else {
  seu_obj@meta.data$experiment = "nonsorted"
}

print(sample)
print("original # cells: ")
print(nrow(df_seu@meta.data))
print("singlet # cells: ")
print(nrow(seu_obj@meta.data))

saveRDS(seu_obj, file = paste("../data/seurat_objects/seurat_raw_1_4_22/",
↪files_base[i], ".rds", sep=""))
}

```

# WB02\_WT\_mut\_integration

June 30, 2022

```
[1]: #THIS SCRIPT PERFORMS SCTransform AND THEN INTEGRATES THE WT AND MUTANT OBJECTS  
↳ INTO A SINGLE SEURAT OBJECT  
  
suppressMessages(library(Seurat))  
library(ggplot2)  
  
proto_genes=read.csv("../data/bulk_data/protoplasting.csv")  
proto_list=as.character(proto_genes[abs(proto_genes$logFC) > 1,]$genes)  
  
#FUNCTION USED IN THIS SCRIPT  
#takes a list of Seurat objects with SCTransform already run  
seu_integrate <- function(..., filename, nfeatures){  
  seu.list <- list(...) # THIS WILL BE A LIST STORING EVERYTHING:  
  
  ref.genes = rownames(seu.list[[1]]@assays$RNA)  
  assay_list <- rep("SCT", length(seu.list))  
  
  # integration  
  rc.features <- SelectIntegrationFeatures(object.list = seu.list, nfeatures_  
↳= nfeatures)  
  
  #remove genes influenced by protoplasting to a high degree as well as  
↳ plastid/mitochondrial genes  
  rc.features <- rc.features[(!c(grepl("ATMG",rc.features) | grepl("ATCG",rc.  
↳ features) | rc.features%in%proto_list))]  
  
  seu.list <- PrepSCTIntegration(object.list = seu.list, anchor.features = rc.  
↳ features, verbose = TRUE, assay = assay_list)  
  seu.list <- lapply(X = seu.list, FUN = RunPCA, verbose = FALSE, features =_  
↳ rc.features)  
  rc.anchors <- FindIntegrationAnchors(object.list = seu.list, normalization.  
↳ method = "SCT", anchor.features = rc.features, verbose = TRUE, reference=1,  
↳ reduction = "rpca")  
  to_integrate <- Reduce(intersect, lapply(rc.anchors@object.list, rownames))  
  
  #integrate data and keep full geneset
```

```

rc.integrated <- IntegrateData(anchorset = rc.anchors, features.to.
→integrate = to_integrate, normalization.method = "SCT", verbose = TRUE)
rc.integrated <- RunPCA(rc.integrated, npcs = 50, verbose = FALSE, approx =
→FALSE)

#save object
saveRDS(rc.integrated, file = paste("../data/intd_seu_objects/",filename,".
→rds", sep = ""))
return(rc.integrated)
# }
}

```

```
[2]: sessionInfo()
```

```

R version 3.6.3 (2020-02-29)
Platform: x86_64-conda_cos6-linux-gnu (64-bit)
Running under: Ubuntu 20.04.2 LTS

Matrix products: default
BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r_3/lib/libopenblas-r0.3.9.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] ggplot2_3.3.5 Seurat_3.1.5
```

loaded via a namespace (and not attached):

```

[1] httr_1.4.2          tidyr_1.1.3          jsonlite_1.7.2       viridisLite_0.4.0
[5] splines_3.6.3       leiden_0.3.9          ggrepel_0.9.1        globals_0.12.5
[9] pillar_1.6.4        lattice_0.20-45       glue_1.6.0           reticulate_1.16
[13] uuid_0.1-4          digest_0.6.29         RColorBrewer_1.1-2   colorspace_2.0-2
[17] cowplot_1.1.1       htmltools_0.5.0      Matrix_1.2-18        plyr_1.8.6
[21] pkgconfig_2.0.3     tsne_0.1-3           listenv_0.8.0        purrr_0.3.4
[25] patchwork_1.1.1     scales_1.1.1         RANN_2.6.1           Rtsne_0.15
[29] tibble_3.1.6        generics_0.1.1       ellipsis_0.3.2       withr_2.4.3
[33] repr_1.1.0          ROCR_1.0-11          pbapply_1.5-0        lazyeval_0.2.2
[37] survival_3.1-12     magrittr_2.0.1       crayon_1.4.2         evaluate_0.14
[41] future_1.18.0       fansi_0.5.0          nlme_3.1-147         MASS_7.3-54

```



[45]	ica_1.0-2	tools_3.6.3	fitdistrplus_1.1-1	data.table_1.14.2
[49]	lifecycle_1.0.1	stringr_1.4.0	plotly_4.9.2.1	munSELL_0.5.0
[53]	cluster_2.1.2	irlba_2.3.5	compiler_3.6.3	rsvd_1.0.3
[57]	rlang_0.4.12	grid_3.6.3	ggribges_0.5.2	pbdZMQ_0.3-6
[61]	IRkernel_1.1	RcppAnnoy_0.0.19	htmlwidgets_1.5.1	igraph_1.2.5
[65]	base64enc_0.1-3	gtable_0.3.0	codetools_0.2-18	DBI_1.1.2
[69]	reshape2_1.4.4	R6_2.5.1	gridExtra_2.3	zoo_1.8-8
[73]	dplyr_1.0.7	uwot_0.1.8	future.apply_1.6.0	utf8_1.2.2
[77]	KernSmooth_2.23-20	ape_5.3	stringi_1.7.6	parallel_3.6.3
[81]	IRdisplay_0.7.0	Rcpp_1.0.7	sctransform_0.2.1	vcrrs_0.3.8
[85]	png_0.1-7	tidyselect_1.1.1	lmtest_0.9-38	

```
[2]: # THIS IS THE PREPROCESSING TO GET TO THE INTEGRATED SEURAT OBJECT.
#WT
wt_1_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/
↳sc_26_combined.rds")
wt_2_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/sc_67.rds")
YFP_1_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/sc_101.
↳rds")
YFP_2_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/sc_103.
↳rds")

#set experimental condition
wt_1_seu@meta.data$condition = "wt_unsorted"
wt_2_seu@meta.data$condition = "wt_unsorted"
YFP_1_seu@meta.data$condition = "wt_sorted"
YFP_2_seu@meta.data$condition = "wt_sorted"

#set batch
wt_1_seu@meta.data$batch = "1"
wt_2_seu@meta.data$batch = "2"
YFP_1_seu@meta.data$batch = "3"
YFP_2_seu@meta.data$batch = "3"

# THIS IS THE PREPROCESSING TO GET TO THE INTEGRATED SEURAT OBJECT.
#MUTANT
mut_1_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/
↳sc_27_combined.rds")
mut_2_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/sc_68.
↳rds")
KE_1_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/sc_102.
↳rds")
KE_2_seu = readRDS(file = "../data/seurat_objects/seurat_raw_3_11_21/sc_104.
↳rds")

#set experimental condition
```

```
mut_1_seu@meta.data$condition = "mut_unsorted"
mut_2_seu@meta.data$condition = "mut_unsorted"
KE_1_seu@meta.data$condition = "mut_sorted"
KE_2_seu@meta.data$condition = "mut_sorted"
```

```
#set batch
mut_1_seu@meta.data$batch = "1"
mut_2_seu@meta.data$batch = "2"
KE_1_seu@meta.data$batch = "3"
KE_2_seu@meta.data$batch = "3"
```

```
[ ]: #SCtransform
#WT
wt_1_seu = SCTransform(wt_1_seu)
wt_2_seu = SCTransform(wt_2_seu)
YFP_1_seu = SCTransform(YFP_1_seu)
YFP_2_seu = SCTransform(YFP_2_seu)

#mutant
mut_1_seu = SCTransform(mut_1_seu)
mut_2_seu = SCTransform(mut_2_seu)
KE_1_seu = SCTransform(KE_1_seu)
KE_2_seu = SCTransform(KE_2_seu)
```

```
[ ]: #Integrate
seu_intd_wt_mut = seu_integrate(wt_1_seu, wt_2_seu, YFP_1_seu, YFP_2_seu,
  ↳ mut_2_seu, mut_1_seu, KE_1_seu, KE_2_seu, filename = "4_12_22_WT_mut",
  ↳ nfeatures = 3000)
```

```
[2]: #Load object
seu_intd_wt_mut = readRDS(file = "../data/intd_seu_objects/4_12_22_WT_mut.rds")
```

```
[ ]: #cluster and UMAP embed
resolution = .75
set.seed(42)
DefaultAssay(seu_intd_wt_mut) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
# Run the standard workflow for visualization and clustering
seu_intd_wt_mut <- RunPCA(seu_intd_wt_mut, npcs = 100, verbose = FALSE, approx_
  ↳ FALSE)
seu_intd_wt_mut <- FindNeighbors(seu_intd_wt_mut, dims = 1:20, verbose = FALSE)
seu_intd_wt_mut <- FindClusters(seu_intd_wt_mut, resolution = resolution,
  ↳ algorithm = 3, verbose = FALSE)
seu_intd_wt_mut <- RunUMAP(seu_intd_wt_mut, reduction = "pca", dims = 1:20,
  ↳ verbose = FALSE)
```

```
[ ]: #plot
options(repr.plot.width= 30, repr.plot.height=18)
DimPlot(seu_intd_wt_mut, reduction = "umap", label = FALSE, pt.size = 2, split.
  ↪by = "geno")#, cols = c("0" = "red"))

[ ]: #Sweep across a few clustering resolutions
res = c(seq(.25, .75, .25))
res

options(repr.plot.width= 30, repr.plot.height=18)

for (r in res) {
  resolution = r
  set.seed(42)
  DefaultAssay(seu_intd_wt_mut) <- "integrated"
  # Run the standard workflow for visualization and clustering
  seu_intd_wt_mut <- RunPCA(seu_intd_wt_mut, npcs = 100, verbose = FALSE,
  ↪approx = FALSE)
  seu_intd_wt_mut <- FindNeighbors(seu_intd_wt_mut, dims = 1:20, verbose =
  ↪FALSE)
  seu_intd_wt_mut <- FindClusters(seu_intd_wt_mut, resolution = resolution,
  ↪algorithm = 3, verbose = FALSE)
  seu_intd_wt_mut <- RunUMAP(seu_intd_wt_mut, reduction = "pca", dims = 1:20,
  ↪verbose = FALSE)
  plot = DimPlot(seu_intd_wt_mut, reduction = "umap", label = TRUE, pt.size =
  ↪2, split.by = "geno")#, cols = c("0" = "red"))
  print(plot)
  ggsave(file=paste0("../data/for_figures/UMAPs/", "res_sweep", as.
  ↪character(r), ".png"), plot=plot, width=20, height=10)
}
```

# WB03\_WT\_AZ\_annotation

June 30, 2022

```
[1]: #THIS SCRIPT PERFORMS AZ CLUSTER IDENTIFICATION

suppressMessages(library(Seurat))
library(ggplot2)

bulk_data = read.csv("../data/buckets/single_cell_bucket_3_4_21/IWT_RNA_seq/
↳scRNA_flowers/outputs/bulk_edger_10_16_20.csv")

annotations = read.csv("R_functions/gene_descriptions.csv", header = F)
colnames(annotations) = c("gene_id", "description")
annotations$gene_id = substr(annotations$gene_id, 1, 9)

bp = read.csv("../data/shiny_go_analysis/figure_3/bp.csv")
cc = read.csv("../data/shiny_go_analysis/figure_3/cc.csv")
mf = read.csv("../data/shiny_go_analysis/figure_3/mf.csv")
```

```
[2]: sessionInfo()
```

R version 3.6.3 (2020-02-29)

Platform: x86\_64-conda\_cos6-linux-gnu (64-bit)

Running under: Ubuntu 20.04.2 LTS

Matrix products: default

BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r\_3/lib/libopenblas-r0.3.9.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] ggplot2_3.3.5 Seurat_3.1.5
```

loaded via a namespace (and not attached):

[1] httr_1.4.2	tidyr_1.1.3	jsonlite_1.7.2	viridisLite_0.4.0
[5] splines_3.6.3	leiden_0.3.9	ggrepel_0.9.1	globals_0.12.5
[9] pillar_1.6.4	lattice_0.20-45	glue_1.6.0	reticulate_1.16
[13] uuid_0.1-4	digest_0.6.29	RColorBrewer_1.1-2	colorspace_2.0-2
[17] cowplot_1.1.1	htmltools_0.5.0	Matrix_1.2-18	plyr_1.8.6
[21] pkgconfig_2.0.3	tsne_0.1-3	listenv_0.8.0	purrr_0.3.4
[25] patchwork_1.1.1	scales_1.1.1	RANN_2.6.1	Rtsne_0.15
[29] tibble_3.1.6	generics_0.1.1	ellipsis_0.3.2	withr_2.4.3
[33] repr_1.1.0	ROCR_1.0-11	pbapply_1.5-0	lazyeval_0.2.2
[37] survival_3.1-12	magrittr_2.0.1	crayon_1.4.2	evaluate_0.14
[41] future_1.18.0	fansi_0.5.0	nlme_3.1-147	MASS_7.3-54
[45] ica_1.0-2	tools_3.6.3	fitdistrplus_1.1-1	data.table_1.14.2
[49] lifecycle_1.0.1	stringr_1.4.0	plotly_4.9.2.1	munsell_0.5.0
[53] cluster_2.1.2	irlba_2.3.5	compiler_3.6.3	rsvd_1.0.3
[57] rlang_0.4.12	grid_3.6.3	ggribes_0.5.2	pbdZMQ_0.3-6
[61] IRkernel_1.1	RcppAnnoy_0.0.19	htmlwidgets_1.5.1	igraph_1.2.5
[65] base64enc_0.1-3	gtable_0.3.0	codetools_0.2-18	DBI_1.1.2
[69] reshape2_1.4.4	R6_2.5.1	gridExtra_2.3	zoo_1.8-8
[73] dplyr_1.0.7	uwot_0.1.8	future.apply_1.6.0	utf8_1.2.2
[77] KernSmooth_2.23-20	ape_5.3	stringi_1.7.6	parallel_3.6.3
[81] IRdisplay_0.7.0	Rcpp_1.0.7	sctransform_0.2.1	vcvtrs_0.3.8
[85] png_0.1-7	tidyselect_1.1.1	lmtest_0.9-38	

```
[ ]: seu_intd_wt_mut = readRDS(file = "../data/intd_seu_objects/4_12_22_WT_mut.rds")
```

```
resolution = .75
set.seed(42)
DefaultAssay(seu_intd_wt_mut) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
seu_intd_wt_mut <- RunPCA(seu_intd_wt_mut, npcs = 100, verbose = FALSE, approx_
  ↪= FALSE)
seu_intd_wt_mut <- FindNeighbors(seu_intd_wt_mut, dims = 1:20, verbose = FALSE)
seu_intd_wt_mut <- FindClusters(seu_intd_wt_mut, resolution = resolution,
  ↪algorithm = 3, verbose = FALSE)
seu_intd_wt_mut <- RunUMAP(seu_intd_wt_mut, reduction = "pca", dims = 1:20,
  ↪verbose = FALSE)
```

```
[ ]: options(repr.plot.width= 20, repr.plot.height=10)
DimPlot(seu_intd_wt_mut, reduction = "umap", label = TRUE, pt.size = 2, split.
  ↪by = "geno")#, cols = c("0" = "red"))
```

```
[ ]: seu_intd_wt = subset(seu_intd_wt_mut, subset = geno == "WT")
```

```
[ ]: #findmarkers
cluster_AZ_all = FindAllMarkers(seu_intd_wt, logfc.threshold = 0, max.cells.
  ↪per.ident = 1000)

[ ]: write.csv(cluster_AZ_all, file = paste("../data/for_figures/",
  ↪"AZ_markers_WT_ALL_res_75_April_25_22", ".csv", sep=""))

[ ]: cluster_AZ_all = readRDS(paste0("../data/markers/",
  ↪"AZ_markers_WT_ALL_res_75_April_19_22", ".rds"))

[ ]: head(cluster_AZ_all[cluster_AZ_all$cluster == 11,])

[ ]: #FOR GO ANALYSIS
#write AZ specific genes as well as all genes with high enough expression to be
  ↪included in the analysis (ie the universe of genes for gene set testing)
write.csv(cluster_AZ_all[cluster_AZ_all$cluster == 11,], file = paste("../data/
  ↪for_figures/", "AZ_spec_genes_universe_WT_res_75_4_25_22", ".csv", sep=""),
  ↪row.names = FALSE)
write.csv(unique(cluster_AZ_all$gene), file = paste("../data/for_figures/",
  ↪"WT_universe_spec_genes_WT_res_75_4_25_22", ".csv", sep=""), row.names =
  ↪FALSE)

[1]: #QRT2 data
kwak_ptpms=read.csv("../data/counts/kwak_ptpms.csv")
rownames(kwak_ptpms) = kwak_ptpms$X
kwak_ptpms[,c(1,2,3)] =NULL
colnames(kwak_ptpms) = "counts"

#HAE YFP sorted
YFP_KE = read.csv("../data/counts/HAE_sorted.csv")
YFP_av = data.frame(YFP_KE[,2])
rownames(YFP_av) = YFP_KE[,1]

[ ]: DefaultAssay(seu_intd_wt) = "RNA"

[ ]: #get pseudobulk for each cluster to compare with kwak data
pbs = list()
count = 1
for (l in levels(seu_intd_wt@meta.data$seurat_clusters)) {
  pbs[[count]] = rowSums(as.matrix(GetAssayData(seu_intd_wt, slot =
  ↪"counts")[, WhichCells(seu_intd_wt, ident = 1)]))
  count = count + 1
}

saveRDS(pbs, "../data/counts/cluster_pbs_4_13_22")

[ ]: pbs = readRDS("../data/counts/cluster_pbs_4_13_22")
```

```
[ ]: #convert pseudobulk to TPM
count = 1
for (c in pbs) {
  pbs[[count]] = data.frame(pbs[[count]]/sum(data.
  ↪frame(pbs[[count]]))*1000000
  rns = rownames(pbs[[count]])
  pbs[[count]] = pbs[[count]][order(rns),, drop = FALSE]
  count = count + 1
}
```

```
[ ]: #QRT2
#set dataset
dataset = kwak_ptpms
cors_spearman = vector()
count = 1

seu_intd_wt@meta.data$kwak_cor = NULL

for (cluster in c(1:length(levels(seu_intd_wt@meta.data$seurat_clusters)))){
  test = cbind(pbs[[cluster]][intersect(rownames(pbs[[cluster]]),
  ↪rownames(dataset)),,dataset[intersect(rownames(pbs[[cluster]]),
  ↪rownames(dataset)),,])
  cors_spearman[count] = cor(log(test[,1]+.1), log(test[,2]+.1), method =
  ↪"spearman")
  count = count + 1
}

for (i in c(1:length(levels(seu_intd_wt@meta.data$seurat_clusters)))){
  seu_intd_wt@meta.data$kwak_cor[seu_intd_wt@meta.data$seurat_clusters ==
  ↪toString(i-1)] = cors_spearman[i]
}

plot = FeaturePlot(seu_intd_wt, features = "kwak_cor", pt.size = 1.5, cols =
  ↪c("gray", "red"))
print(plot)
ggsave(file=paste0("../data/for_figures/UMAPs/kwak_cor_wt_2_1_22.png"),
  ↪plot=plot, width=10, height=10)
```

```
[ ]: #HAE
#set dataset
dataset = YFP_av
cors_spearman = vector()
count = 1

seu_intd_wt@meta.data$HAE_YFP = NULL

for (cluster in c(1:length(levels(seu_intd_wt@meta.data$seurat_clusters)))){
```

```

    test = cbind(pbs[[cluster]][intersect(rownames(pbs[[cluster]]),
↪rownames(dataset)),], dataset[intersect(rownames(pbs[[cluster]]),
↪rownames(dataset)),])
    cors_spearman[count] = cor(log(test[,1]+.1), log(test[,2]+.1), method =
↪"spearman")
    count = count + 1
}

for (i in c(1:length(levels(seu_intd_wt@meta.data$seurat_clusters)))){
    seu_intd_wt@meta.data$HAE_YFP[seu_intd_wt@meta.data$seurat_clusters ==
↪toString(i-1)] = cors_spearman[i]
}

plot = FeaturePlot(seu_intd_wt, features = "HAE_YFP", pt.size = 1.5, cols =
↪c("white", "red"))
print(plot)
ggsave(file=paste0("../data/for_figures/UMAPs/HAE_YFP_cor_wt_2_1_22.png"),
↪plot=plot, width=10, height=10)

```



# WB04\_cell\_annotation

June 30, 2022

```
[1]: #THIS SCRIPT ATTEMPTS TO ANNOTATE OTHER CLUSTERS IN THE INTEGRATED OBJECT
```

```
suppressMessages(library(Seurat))
suppressMessages(library(tidyverse))
suppressMessages(library(dplyr))
suppressMessages(library(cowplot))
suppressMessages(library(data.table))
```

```
[2]: sessionInfo()
```

R version 3.6.3 (2020-02-29)

Platform: x86\_64-conda\_cos6-linux-gnu (64-bit)

Running under: Ubuntu 20.04.2 LTS

Matrix products: default

BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r\_3/lib/libopenblas-r0.3.9.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] data.table_1.14.2 cowplot_1.1.1    forcats_0.5.0    stringr_1.4.0
[5] dplyr_1.0.7        purrr_0.3.4      readr_1.4.0      tidyr_1.1.3
[9] tibble_3.1.6       ggplot2_3.3.5    tidyverse_1.3.0  Seurat_3.1.5
```

loaded via a namespace (and not attached):

```
[1] nlme_3.1-147      tsne_0.1-3       fs_1.5.2         lubridate_1.7.8
[5] RcppAnnoy_0.0.19  RColorBrewer_1.1-2 httr_1.4.2       repr_1.1.0
[9] sctransform_0.2.1 tools_3.6.3      backports_1.4.1  utf8_1.2.2
[13] R6_2.5.1          irlba_2.3.5      KernSmooth_2.23-20 uwot_0.1.8
```

[17]	DBI_1.1.2	lazyeval_0.2.2	colorspace_2.0-2	withr_2.4.3
[21]	tidyselect_1.1.1	gridExtra_2.3	compiler_3.6.3	cli_3.1.0
[25]	rvest_0.3.5	xml2_1.3.1	plotly_4.9.2.1	scales_1.1.1
[29]	lmtest_0.9-38	ggribes_0.5.2	pbapply_1.5-0	pbDZMQ_0.3-6
[33]	digest_0.6.29	base64enc_0.1-3	pkgconfig_2.0.3	htmltools_0.5.0
[37]	dbplyr_1.4.2	readxl_1.3.1	htmlwidgets_1.5.1	rlang_0.4.12
[41]	rstudioapi_0.13	generics_0.1.1	zoo_1.8-8	jsonlite_1.7.2
[45]	ica_1.0-2	magrittr_2.0.1	patchwork_1.1.1	Matrix_1.2-18
[49]	Rcpp_1.0.7	IRkernel_1.1	munsell_0.5.0	fansi_0.5.0
[53]	ape_5.3	reticulate_1.16	lifecycle_1.0.1	stringi_1.7.6
[57]	MASS_7.3-54	Rtsne_0.15	plyr_1.8.6	grid_3.6.3
[61]	parallel_3.6.3	listenv_0.8.0	ggrepel_0.9.1	crayon_1.4.2
[65]	lattice_0.20-45	haven_2.3.1	IRdisplay_0.7.0	splines_3.6.3
[69]	hms_1.1.0	pillar_1.6.4	igraph_1.2.5	uuid_0.1-4
[73]	future.apply_1.6.0	reshape2_1.4.4	codetools_0.2-18	leiden_0.3.9
[77]	reprex_0.3.0	glue_1.6.0	evaluate_0.14	modelr_0.1.6
[81]	png_0.1-7	vctrs_0.3.8	cellranger_1.1.0	gtable_0.3.0
[85]	RANN_2.6.1	future_1.18.0	assertthat_0.2.1	rsvd_1.0.3
[89]	broom_0.5.5	survival_3.1-12	viridisLite_0.4.0	cluster_2.1.2
[93]	globals_0.12.5	fitdistrplus_1.1-1	ellipsoids_0.3.2	ROCR_1.0-11

```
[ ]: seu_intd_wt_mut = readRDS(file = "../data/intd_seu_objects/4_12_22_WT_mut.rds")
```

```
[ ]: resolution = .75
set.seed(42)
DefaultAssay(seu_intd_wt_mut) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
# Run the standard workflow for visualization and clustering
#all_intd_sct <- ScaleData(all_intd_sct, verbose = FALSE)
seu_intd_wt_mut <- RunPCA(seu_intd_wt_mut, npcs = 100, verbose = FALSE, approx_
  <- FALSE)
#From RunPCA doc: Features to compute PCA on. If features=NULL, PCA will be run_
  <- using the variable features for the Assay.
#Note that the features must be present in the scaled data. Any requested_
  <- features that are not scaled or have 0 variance
#will be dropped, and the PCA will be run using the remaining features.

#previously run 20 PCs as of 2/14/22
seu_intd_wt_mut <- FindNeighbors(seu_intd_wt_mut, dims = 1:20, verbose = FALSE)
seu_intd_wt_mut <- FindClusters(seu_intd_wt_mut, resolution = resolution,
  <- algorithm = 3, verbose = FALSE)
seu_intd_wt_mut <- RunUMAP(seu_intd_wt_mut, reduction = "pca", dims = 1:20,
  <- verbose = FALSE)
```

```
[ ]: options(repr.plot.width= 20, repr.plot.height=10)
```

```
DimPlot(seu_intd_wt_mut, reduction = "umap", label = TRUE, pt.size = 2, split.
  ↪by = "geno")
```

```
[ ]: #seu_intd_wt = subset(seu_intd_wt_mut, subset = geno == "WT")
seu_intd_wt = seu_intd_wt_mut
```

```
[ ]: known.good.markers <- read.csv("../data/cell_type_markers/markers.csv", header =
  ↪F)
colnames(known.good.markers) = c("Name", "Locus", "Celltype")
known.good.markers <- known.good.markers[known.good.markers$Locus %in%
  ↪rownames(seu_intd_wt@assays$RNA),]
known.good.markers$Celltype <- gsub("abscission_zone", "Abscission Zone", known.
  ↪good.markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
known.good.markers$Celltype <- gsub("companion_cells", "Companion Cells", known.
  ↪good.markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
known.good.markers$Celltype <- gsub("xylem", "Xylem", known.good.
  ↪markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
known.good.markers$Celltype <- gsub("phloem", "Phloem", known.good.
  ↪markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
known.good.markers$Celltype <- gsub("vascular_subtype_1", "Vascular Subtype",
  ↪known.good.markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T,
  ↪useBytes = FALSE);
known.good.markers$Celltype <- gsub("epidermis", "Epidermis", known.good.
  ↪markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
known.good.markers$Celltype <- gsub("sieve_element", "Sieve Element", known.
  ↪good.markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
known.good.markers$Celltype <- gsub("tracheary_element", "Tracheary Element",
  ↪known.good.markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T,
  ↪useBytes = FALSE);
known.good.markers$Celltype <- gsub("mesophyll", "Mesophyll", known.good.
  ↪markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
known.good.markers$Celltype <- gsub("guard_cells", "Guard Cells", known.good.
  ↪markers$Celltype, ignore.case = FALSE, perl = FALSE, fixed = T, useBytes =
  ↪FALSE);
```

```
[ ]: known.good.markers = known.good.markers[!known.good.markers$Name %in% c("IDA",
  ↪"PGAZAT", "HSL2", "HAESA"),]
known.good.markers
```

```
[ ]: AZ_new = c("AT3G44550", "AT3G59850", "AT5G03820", "AT1G68320")
AZ_new_names = c("FAR5", "PLL", "GDSL", "MYB62")
AZ_new_df = data.frame(matrix(ncol = 3, nrow = 4))
colnames(AZ_new_df) = c("Name", "Locus", "Celltype")
AZ_new_df$Locus = AZ_new
AZ_new_df$Celltype = "Abscission Zone"
AZ_new_df$Name = AZ_new_names
known.good.markers = rbind(known.good.markers, AZ_new_df)
```

```
[ ]: known.good.markers
```

```
[ ]: #MAY NOT NEED THIS CELL
```

```
DefaultAssay(seu_intd_wt) = "SCT"
options(repr.plot.width=8, repr.plot.height=8)

for (g in as.character(known.good.markers$Locus)) {
  plot = (FeaturePlot(seu_intd_wt, feature = g, pt.size = 4, order = TRUE,
    ↪min = .50))
  ggsave(file=paste0("../data/for_figures/gene_plots/figure_2_pngs/
    ↪celltype_plots/", g, "_", known.good.markers[known.good.markers$Locus ==
    ↪g,]$Celltype, ".png"), plot=plot, width=10, height=10)
}
```

```
[ ]: #MAY NOT NEED THIS CELL
```

```
DefaultAssay(seu_intd_wt) = "SCT"
g = "AT3G01420"
plot = (FeaturePlot(seu_intd_wt, feature = g, pt.size = 4, order = TRUE, min =
    ↪.50))
options(repr.plot.width=8, repr.plot.height=8)
ggsave(file=paste0("../data/for_figures/gene_plots/figure_2_pngs/celltype_plots/
    ↪", g, "_silique", ".png"), plot=plot, width=10, height=10)
print(plot)
```

```
[ ]: resolution = 2
set.seed(42)
DefaultAssay(seu_intd_wt) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
# Run the standard workflow for visualization and clustering
#all_intd_sct <- ScaleData(all_intd_sct, verbose = FALSE)
seu_intd_wt <- RunPCA(seu_intd_wt, npcs = 100, verbose = FALSE, approx = FALSE)
#From RunPCA doc: Features to compute PCA on. If features=NULL, PCA will be run
    ↪using the variable features for the Assay.
#Note that the features must be present in the scaled data. Any requested
    ↪features that are not scaled or have 0 variance
```

```
#will be dropped, and the PCA will be run using the remaining features.

#previously run 20 PCs as of 2/14/22
seu_intd_wt <- FindNeighbors(seu_intd_wt, dims = 1:20, verbose = FALSE)
seu_intd_wt <- FindClusters(seu_intd_wt, resolution = resolution, algorithm = 3, verbose = FALSE)
seu_intd_wt <- RunUMAP(seu_intd_wt, reduction = "pca", dims = 1:20, verbose = FALSE)
```

```
[ ]: zscore <- function(x){(x-mean(x))/sd(x)}

msc <- c()
for (i in as.character(unique(known.good.markers$Celltype))){
  if (length(known.good.markers[which(known.good.markers$Celltype==i),]$Locus)>1){
    msc <- cbind(msc, as.numeric(apply(apply(seu_intd_wt@assays$SCT@data[known.good.markers[which(known.good.markers$Celltype== i),]$Locus,], 1, zscore), 1, mean)))
  } else {
    msc <- cbind(msc, as.numeric(zscore(seu_intd_wt@assays$SCT@data[known.good.markers[which(known.good.markers$Celltype== i),]$Locus,])))
  }
}

colnames(msc) <- as.character(unique(known.good.markers$Celltype))
rownames(msc) <- colnames(seu_intd_wt)
```

```
[ ]: DefaultAssay(seu_intd_wt) <- "integrated"
suppressMessages(suppressWarnings(
  seu_intd_wt <- FindClusters(seu_intd_wt, resolution = 2, algorithm = 3)
))
```

```
[ ]: anno <- seu_intd_wt$seurat_clusters
for (i in unique(seu_intd_wt$seurat_clusters)){
  if (max(apply(msc[which(seu_intd_wt$seurat_clusters==i),], 2, mean))>0){
    ct <- names(which.max(apply(msc[which(seu_intd_wt$seurat_clusters==i),], 2, mean)))
  } else {
    ct <- "NA"
  }
  anno <- gsub(paste0("^", i, "$"), ct, anno, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)
}
```

```
[ ]: seu_intd_wt$score.crude.anno <- anno
```

```
[ ]: # Plot marker annotation
order <- c("Abscission Zone", "Base of Sepals/Petals", "Columella", "Lateral_
↳Root Cap", "Atrichoblast", "Epidermis", "Mesophyll", "Guard Cells",_
↳"Phloem", "Sieve Element", "Xylem", "Vascular Subtype", "Companion_
↳Cells", "Phloem Pole Pericycle", "Protoxylem", "Tracheary Element", "Unknown")
palette <- c("#9400d3", "#DCD0FF", "#5ab953", "#bfef45", "#008080", "#21B6A8",_
↳"#82b6ff", "#0000FF", "#e6194b", "#dd77ec", "#9a6324", "#ffe119", "#ff9900",_
↳"#ffd4e3", "#9a6324", "#ddaa6f", "#EEEEEE")
seu_intd_wt$score.crude.anno <- factor(seu_intd_wt$score.crude.anno , levels =_
↳order[sort(match(unique(seu_intd_wt$score.crude.anno),order))])
color <- palette[sort(match(unique(seu_intd_wt$score.crude.anno),order))]
options(repr.plot.width=20, repr.plot.height=10)
DimPlot(seu_intd_wt, reduction = "umap", group.by = "score.crude.anno", split.
↳by = "geno", cols = color)+ggtitle("Z-Score Annotation Crude")
```

```
[ ]: # Find clusters, here we choose Leiden clustering algorithm with resolution 0.5.
↳ Parameter "algorithm": 1 = original Louvain algorithm; 2 = Louvain_
↳algorithm with multilevel refinement; 3 = SLM algorithm; 4 = Leiden algorithm
DefaultAssay(seu_intd_wt) <- "integrated"
suppressMessages(suppressWarnings(
  seu_intd_wt <- FindClusters(seu_intd_wt, resolution = 200, algorithm = 3)
))
```

```
[ ]: anno <- seu_intd_wt$seurat_clusters
for (i in unique(seu_intd_wt$seurat_clusters)){
  if (max(apply(msc[which(seu_intd_wt$seurat_clusters==i),],2,mean))>0){
    ct <- names(which.
↳max(apply(msc[which(seu_intd_wt$seurat_clusters==i),],2,mean)))
  } else {
    ct <- "NA"
  }
  anno <- gsub(paste0("^",i,"$"), ct, anno, ignore.case = FALSE, perl =_
↳FALSE,fixed = FALSE, useBytes = FALSE)
}

seu_intd_wt$score.anno <- anno
# Plot marker annotation
order <- c("Abscission Zone", "Base of Sepals/Petals", "Columella", "Lateral_
↳Root Cap", "Atrichoblast", "Epidermis", "Mesophyll", "Guard Cells",_
↳"Phloem", "Sieve Element", "Xylem", "Vascular Subtype", "Companion_
↳Cells", "Phloem Pole Pericycle", "Protoxylem", "Tracheary Element", "Unknown")
palette <- c("#9400d3", "#DCD0FF", "#5ab953", "#bfef45", "#008080", "#21B6A8",_
↳"#82b6ff", "#0000FF", "#e6194b", "#dd77ec", "#9a6324", "#ffe119", "#ff9900",_
↳"#ffd4e3", "#9a6324", "#ddaa6f", "#EEEEEE")
```

```

seu_intd_wt$score.anno <- factor(seu_intd_wt$score.anno , levels =
  ↪order[sort(match(unique(seu_intd_wt$score.anno),order))])
#color <- palette[sort(match(unique(seu_intd_wt$score.anno),order))]
#options(repr.plot.width=12, repr.plot.height=10)
#DimPlot(seu_intd_wt, reduction = "umap", group.by = "score.anno", cols =
  ↪color)+ggtitle("Z-Score Annotation")

```

```

[ ]: #Consensus Annotation
dat <- data.frame(seu_intd_wt$score.anno, seu_intd_wt$score.crude.anno)
seu_intd_wt$consensus.anno <- apply(dat,1,function(x){if (is.
  ↪na(x[1])){"Unknown"} else if (is.na(x[2])){"Unknown"} else if
  ↪(x[1]==x[2]){x[1]} else {"Unknown"}})
#seu_intd_wt$consensus.anno <- apply(dat,1,function(x){if(x[1]==x[2]){x[1]}else
  ↪if(x[1]=="Trichoblast" & x[2]=="Atrichoblast"){"Trichoblast"}
#   else if(x[1]=="Late Metaxylem" & x[2]=="Phloem"){"Late Metaxylem"}else
  ↪if(x[1]=="Cortex" & x[2]=="Sclerenchyma"){"Cortex"}
#   else if(x[1]=="Exodermis" & x[2]=="Sclerenchyma"){"Exodermis"}else
  ↪if(x[1]=="Exodermis" & x[2]=="Endodermis"){"Exodermis"}
#   else if(x[1]=="Cortex" & x[2]=="Endodermis"){"Cortex"}else
  ↪if(x[1]=="Pericycle" & x[2]=="Endodermis"){"Pericycle"}else
  ↪if(x[1]=="Phloem" & x[2]=="Endodermis"){"Phloem"}
#   else if(x[1]=="Late Metaxylem" & x[2]=="Endodermis"){"Late
  ↪Metaxylem"}else if(x[3]=="Maturation1"/x[3]=="Maturation2"){x[2]}else
  ↪{"Unknown"}})
order <- c("Abscission Zone", "Base of Sepals/Petals","Columella", "Lateral
  ↪Root Cap", "Atrichoblast", "Epidermis", "Mesophyll", "Guard Cells",
  ↪"Phloem", "Sieve Element", "Xylem", "Vascular Subtype", "Companion
  ↪Cells", "Phloem Pole Pericycle", "Protoxylem", "Tracheary Element", "Unknown")
palette <- c("#9400d3", "#DCD0FF", "#5ab953", "#bfe445", "#008080", "#21B6A8",
  ↪"#82b6ff", "#0000FF", "#e6194b", "#dd77ec", "#9a6324", "#ffe119", "#ff9900",
  ↪"#ffd4e3", "#9a6324", "#ddaa6f", "#EEEEEE")
seu_intd_wt$consensus.anno <- factor(seu_intd_wt$consensus.anno , levels =
  ↪order[sort(match(unique(seu_intd_wt$consensus.anno),order))])
color <- palette[sort(match(unique(seu_intd_wt$consensus.anno),order))]
options(repr.plot.width=10, repr.plot.height=10)
DimPlot(seu_intd_wt, reduction = "umap", group.by = "consensus.anno", pt.size =
  ↪3, cols = color)+ggtitle("consensus.anno")

seu_intd_wt$celltype.consensus.anno <- seu_intd_wt$consensus.anno

```

```

[ ]: options(repr.plot.width=20, repr.plot.height=10)
DimPlot(seu_intd_wt, reduction = "umap", group.by = "consensus.anno", pt.size =
  ↪3, split.by = "geno", cols = color)+ggtitle("consensus.anno")

seu_intd_wt$celltype.consensus.anno <- seu_intd_wt$consensus.anno

```

```
[ ]: color <- palette[sort(match( c("Mesophyll", "Phloem", "Xylem", "Guard Cells",
  ↪ "Companion Cells", "Epidermis", "Abscission Zone", "Unknown"), order))]
```

```
seu_intd_wt@meta.data[seu_intd_wt@meta.data$consensus.anno == "Sieve
  ↪ Element",]$consensus.anno = "Unknown"
```

```
seu_intd_wt@meta.data[seu_intd_wt@meta.data$consensus.anno == "Tracheary
  ↪ Element",]$consensus.anno = "Unknown"
```

```
seu_intd_wt@meta.data[seu_intd_wt@meta.data$consensus.anno == "Vascular
  ↪ Subtype",]$consensus.anno = "Unknown"
```

```
[ ]: #c("#9400d3", , "#5ab953", , , "#21B6A8", "#0000FF", , "#dd77ec", "#9a6324",
  ↪ "#ffe119", "#ff9900", "#ffd4e3", "#9a6324", "#ddaa6f", "#EEEEEE")
```

```
options(repr.plot.width= 20, repr.plot.height=10)
```

```
palette <- c("#EEEEEE", "#dd77ec", "#42f5ef", "#f542ef", "#ff9900", "#42f548",
  ↪ "#0000FF", "#f56642")
```

```
plot = DimPlot(seu_intd_wt, reduction = "umap", group.by = "consensus.anno",
  ↪ split.by = "geno", order = c("Mesophyll", "Phloem", "Xylem", "Guard Cells",
  ↪ "Companion Cells", "Epidermis", "Abscission Zone", "Unknown"), pt.size = 4,
  ↪ cols =palette)+ggtitle("consensus.anno")
```

```
ggsave(file=paste0("../data/for_figures/gene_plots/figure_2_pngs/celltype_plots/
  ↪ celltype_UMAP.png"), plot=plot, width=20, height=10)
```

```
print(plot)
```



# WB05\_pseudo\_bulk

June 30, 2022

```
[1]: #THIS SCRIPT PERFORMS PSEUDOBULK ANALYSIS ON THE AZ IN WT AND MUTANT

suppressMessages(library(Seurat))
library(here)
source(here("R_functions", "edgeR_function.R"))

annotations = read.csv("R_functions/gene_descriptions.csv", header = F)
colnames(annotations) = c("gene_id", "description")
annotations$gene_id = substr(annotations$gene_id, 1, 9)

proto_genes=read.csv("../data/bulk_data/protoplasting.csv")
proto_list=as.character(proto_genes[abs(proto_genes$logFC) > 1,]$genes)
```

here() starts at /home/robotmessenger810/sc\_analysis/code

```
[2]: sessionInfo()
```

R version 3.6.3 (2020-02-29)

Platform: x86\_64-conda\_cos6-linux-gnu (64-bit)

Running under: Ubuntu 20.04.2 LTS

Matrix products: default

BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r\_3/lib/libopenblas-r0.3.9.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] here_0.1      Seurat_3.1.5
```

loaded via a namespace (and not attached):

[1] httr_1.4.2	tidyr_1.1.3	jsonlite_1.7.2	viridisLite_0.4.0
[5] splines_3.6.3	leiden_0.3.9	ggrepel_0.9.1	globals_0.12.5
[9] pillar_1.6.4	lattice_0.20-45	glue_1.6.0	reticulate_1.16
[13] uuid_0.1-4	digest_0.6.29	RColorBrewer_1.1-2	colorspace_2.0-2
[17] cowplot_1.1.1	htmltools_0.5.0	Matrix_1.2-18	plyr_1.8.6
[21] pkgconfig_2.0.3	tsne_0.1-3	listenv_0.8.0	purrr_0.3.4
[25] patchwork_1.1.1	scales_1.1.1	RANN_2.6.1	Rtsne_0.15
[29] tibble_3.1.6	generics_0.1.1	ggplot2_3.3.5	ellipsis_0.3.2
[33] repr_1.1.0	ROCR_1.0-11	pbapply_1.5-0	lazyeval_0.2.2
[37] survival_3.1-12	magrittr_2.0.1	crayon_1.4.2	evaluate_0.14
[41] future_1.18.0	fansi_0.5.0	nlme_3.1-147	MASS_7.3-54
[45] ica_1.0-2	tools_3.6.3	fitdistrplus_1.1-1	data.table_1.14.2
[49] lifecycle_1.0.1	stringr_1.4.0	plotly_4.9.2.1	munsell_0.5.0
[53] cluster_2.1.2	irlba_2.3.5	compiler_3.6.3	rsvd_1.0.3
[57] rlang_0.4.12	grid_3.6.3	ggribes_0.5.2	pbdZMQ_0.3-6
[61] IRkernel_1.1	RcppAnnoy_0.0.19	htmlwidgets_1.5.1	igraph_1.2.5
[65] base64enc_0.1-3	gtable_0.3.0	codetools_0.2-18	DBI_1.1.2
[69] reshape2_1.4.4	R6_2.5.1	gridExtra_2.3	zoo_1.8-8
[73] dplyr_1.0.7	uwot_0.1.8	future.apply_1.6.0	utf8_1.2.2
[77] rprojroot_2.0.2	KernSmooth_2.23-20	ape_5.3	stringi_1.7.6
[81] parallel_3.6.3	IRdisplay_0.7.0	Rcpp_1.0.7	sctransform_0.2.1
[85] vctrs_0.3.8	png_0.1-7	tidyselect_1.1.1	lmtest_0.9-38

```
[ ]: seu_intd_wt_mut = readRDS(file = "../data/intd_seu_objects/4_12_22_WT_mut.rds")

resolution = .75
set.seed(42)
DefaultAssay(seu_intd_wt_mut) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
seu_intd_wt_mut <- RunPCA(seu_intd_wt_mut, npcs = 100, verbose = FALSE, approx_
  ↪= FALSE)
seu_intd_wt_mut <- FindNeighbors(seu_intd_wt_mut, dims = 1:20, verbose = FALSE)
seu_intd_wt_mut <- FindClusters(seu_intd_wt_mut, resolution = resolution,
  ↪algorithm = 3, verbose = FALSE)
seu_intd_wt_mut <- RunUMAP(seu_intd_wt_mut, reduction = "pca", dims = 1:20,
  ↪verbose = FALSE)
```

```
[ ]: cluster = "11" #AZ cluster
seu_intd_wt_mut@active.assay = "RNA"

wt_1_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset = orig.
  ↪ident == "sc_26_combined"), slot = "counts"),
  ↪WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_26_combined"),
  ↪ident = cluster)))
```

```
wt_2_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset = orig.
  ↳ ident == "sc_67"), slot = "counts")[, WhichCells(subset(seu_intd_wt_mut,
  ↳ subset = orig.ident == "sc_67"), ident = cluster)]))
YFP_1_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset =
  ↳ orig.ident == "sc_101"), slot = "counts")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_101"), ident =
  ↳ cluster)]))
YFP_2_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset =
  ↳ orig.ident == "sc_103"), slot = "counts")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_103"), ident =
  ↳ cluster)]))
```

```
[ ]: cluster = "11"
seu_intd_wt_mut@active.assay = "RNA"

mut_1_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset =
  ↳ orig.ident == "sc_27_combined"), slot = "counts")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_27_combined"),
  ↳ ident = cluster)]))
mut_2_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset =
  ↳ orig.ident == "sc_68"), slot = "counts")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_68"), ident =
  ↳ cluster)]))
KE_1_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset = orig.
  ↳ ident == "sc_102"), slot = "counts")[, WhichCells(subset(seu_intd_wt_mut,
  ↳ subset = orig.ident == "sc_102"), ident = cluster)]))
KE_2_AZ <- rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_mut, subset = orig.
  ↳ ident == "sc_104"), slot = "counts")[, WhichCells(subset(seu_intd_wt_mut,
  ↳ subset = orig.ident == "sc_104"), ident = cluster)]))
```

```
[ ]: gene_intersection = intersect(names(wt_1_AZ), names(mut_1_AZ))

wt_1_AZ = wt_1_AZ[gene_intersection]
wt_2_AZ = wt_2_AZ[gene_intersection]
YFP_1_AZ = YFP_1_AZ[gene_intersection]
YFP_2_AZ = YFP_2_AZ[gene_intersection]
mut_1_AZ = mut_1_AZ[gene_intersection]
mut_2_AZ = mut_2_AZ[gene_intersection]
KE_1_AZ = KE_1_AZ[gene_intersection]
KE_2_AZ = KE_2_AZ [gene_intersection]
```

```
[ ]: pb_df = data.frame(cbind(wt_1_AZ , wt_2_AZ, YFP_1_AZ, YFP_2_AZ, mut_1_AZ ,
  ↳ mut_2_AZ, KE_1_AZ, KE_2_AZ))
colnames(pb_df) = c("WT1", "WT2", "YFP1", "YFP2", "mut1", "mut2", "KE1", "KE2")
rownames(pb_df) = gene_intersection
```

```
[ ]: write.csv(pb_df, "../data/pseudo_bulk_data/AZ_pbs_4_19_22.csv")

[ ]: pb_df = read.csv("../data/pseudo_bulk_data/AZ_pbs_4_19_22.csv")
      rownames(pb_df) = pb_df[,1]
      pb_df[,1] <- NULL

[ ]: #account for factors in experiment
      phenotype=as.factor(c("wt", "wt", "wt", "wt", "mut", "mut", "mut", "mut"))
      batch=as.factor(c(0,0,1,1,0,0,1,1))
      design <- model.matrix(~phenotype+batch)

      #double check design matrix isn't singular
      print(paste("determinant of XT*X of design matrix is: ",
        ↪det(t(design)%*%(design))))

      #making contrast matrix for tests of interest
      my.contrasts <- makeContrasts(s1_v_s2=phenotypewt, levels=design)

[ ]: my.contrasts
      design

[ ]: #put experimental covariates in
      bulk_edger_1 = edgeR_2_sample(pb_df, "WT", "mut", c(1,2,3,4), c(5,6,7,8),
        ↪annotations, design, my.contrasts)

[ ]: WT_higher_1 = bulk_edger_1[bulk_edger_1$FDR < .2 & bulk_edger_1$logFC > 1,]
      WT_lower_1 = bulk_edger_1[bulk_edger_1$FDR < .2 & bulk_edger_1$logFC < -1,]

[ ]: dim(WT_higher_1)

[ ]: write.csv(bulk_edger_1, "../data/pseudo_bulk_data/AZ_edger_4_19_22_factors.csv")

[ ]: bulk_edger_1 = read.csv("../data/pseudo_bulk_data/AZ_edger_4_19_22_factors.csv")
      write.csv(bulk_edger_1, "../data/for_figures/AZ_edger_4_19_22_factors.csv")

[ ]: dim(bulk_edger_1)
```

# WB06\_subclustering

June 30, 2022

```
[1]: #THIS SCRIPT PERFORMS SUBCLUSTERING OF THE WT AND MUTANT AZ CELLS, IDENTIFIES,  
→PUTATIVE SECESSION AND RESIDUUM CELLS, AND PERFORMS GENE ENRICHMENT ANALYSIS.
```

```
library(here)
library(Matrix)
library(tidyverse)
library(Seurat)
library(edgeR)
library(limma)
source(here("R_functions", "edgeR_function.R"))

annotations = read.csv("R_functions/gene_descriptions.csv", header = F)
colnames(annotations) = c("gene_id", "description")
annotations$gene_id = substr(annotations$gene_id, 1, 9)

proto_genes=read.csv("../data/bulk_data/protoplasting.csv")
proto_list=as.character(proto_genes[abs(proto_genes$logFC) > 1,]$genes)
bulk_data = read.csv("/home/robotmessenger810/data/buckets/  
→single_cell_bucket_3_4_21/IWT_RNA_seq/scRNA_flowern/outputs/  
→bulk_edger_10_16_20.csv")
```

```
[2]: sessionInfo()
```

R version 3.6.3 (2020-02-29)

Platform: x86\_64-conda\_cos6-linux-gnu (64-bit)

Running under: Ubuntu 20.04.2 LTS

Matrix products: default

BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r\_3/lib/libopenblas-r0.3.9.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] edgeR_3.28.1    limma_3.42.2    Seurat_3.1.5    forcats_0.5.0
[5] stringr_1.4.0    dplyr_1.0.7     purrr_0.3.4     readr_1.4.0
[9] tidyr_1.1.3     tibble_3.1.6    ggplot2_3.3.5    tidyverse_1.3.0
[13] Matrix_1.2-18    here_0.1
```

loaded via a namespace (and not attached):

```
[1] Rtsne_0.15      colorspace_2.0-2 ellipsis_0.3.2    ggribes_0.5.2
[5] rprojroot_2.0.2 IRdisplay_0.7.0    base64enc_0.1-3    fs_1.5.2
[9] rstudioapi_0.13 leiden_0.3.9        listenv_0.8.0      ggrepel_0.9.1
[13] fansi_0.5.0     lubridate_1.7.8    xml2_1.3.1         codetools_0.2-18
[17] splines_3.6.3   IRkernel_1.1       jsonlite_1.7.2     broom_0.5.5
[21] ica_1.0-2       cluster_2.1.2      dbplyr_1.4.2       png_0.1-7
[25] uwot_0.1.8      sctransform_0.2.1 compiler_3.6.3     httr_1.4.2
[29] backports_1.4.1 assertthat_0.2.1   lazyeval_0.2.2     cli_3.1.0
[33] htmltools_0.5.0 tools_3.6.3        rsvd_1.0.3         igraph_1.2.5
[37] gtable_0.3.0    glue_1.6.0         RANN_2.6.1         reshape2_1.4.4
[41] Rcpp_1.0.7      cellranger_1.1.0   vctrs_0.3.8        ape_5.3
[45] nlme_3.1-147    lmtest_0.9-38      globals_0.12.5     rvest_0.3.5
[49] lifecycle_1.0.1 irlba_2.3.5        future_1.18.0      MASS_7.3-54
[53] zoo_1.8-8       scales_1.1.1       hms_1.1.0          parallel_3.6.3
[57] RColorBrewer_1.1-2 reticulate_1.16    pbapply_1.5-0      gridExtra_2.3
[61] stringi_1.7.6   repr_1.1.0         rlang_0.4.12       pkgconfig_2.0.3
[65] evaluate_0.14   lattice_0.20-45    ROCR_1.0-11        patchwork_1.1.1
[69] htmlwidgets_1.5.1 cowplot_1.1.1      tidyselect_1.1.1   RcppAnnoy_0.0.19
[73] plyr_1.8.6      magrittr_2.0.1     R6_2.5.1           generics_0.1.1
[77] pbdZMQ_0.3-6    DBI_1.1.2          pillar_1.6.4       haven_2.3.1
[81] withr_2.4.3     fitdistrplus_1.1-1 survival_3.1-12     future.apply_1.6.0
[85] tsne_0.1-3      modelr_0.1.6       crayon_1.4.2       uuid_0.1-4
[89] KernSmooth_2.23-20 utf8_1.2.2         plotly_4.9.2.1     locfit_1.5-9.4
[93] grid_3.6.3      readxl_1.3.1       data.table_1.14.2  reprex_0.3.0
[97] digest_0.6.29   munsell_0.5.0     viridisLite_0.4.0
```

```
[3]: seu_intd_wt_mut = readRDS(file = "../data/intd_seu_objects/4_12_22_WT_mut.rds")
```

```
[ ]: resolution = .75
set.seed(42)
DefaultAssay(seu_intd_wt_mut_mut) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
# Run the standard workflow for visualization and clustering
seu_intd_wt_mut_mut <- RunPCA(seu_intd_wt_mut_mut, npcs = 100, verbose = FALSE,
  ↪approx = FALSE)
```

```

seu_intd_wt_mut_mut <- FindNeighbors(seu_intd_wt_mut_mut, dims = 1:20, verbose = FALSE)
seu_intd_wt_mut_mut <- FindClusters(seu_intd_wt_mut_mut, resolution = resolution, algorithm = 3, verbose = FALSE)
seu_intd_wt_mut_mut <- RunUMAP(seu_intd_wt_mut_mut, reduction = "pca", dims = 1:20, verbose = FALSE)

```

```

[ ]: cluster = "11"

wt_1_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_26_combined")[, WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_26_combined"), ident = cluster)]
wt_2_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_67")[, WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_67"), ident = cluster)]
YFP_1_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_101")[, WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_101"), ident = cluster)]
YFP_2_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_103")[, WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_103"), ident = cluster)]

```

```

[ ]: wt_1_seu = SCTransform(wt_1_AZ)
wt_2_seu = SCTransform(wt_2_AZ)
YFP_1_seu = SCTransform(YFP_1_AZ)
YFP_2_seu = SCTransform(YFP_2_AZ)

```

```

[ ]: seu_intd_wt_AZ = seu_integrate(wt_1_seu, wt_2_seu, YFP_1_seu, YFP_2_seu, filename = "AZ_only_WT_4_19_22", nfeatures = 3000)

```

```

[2]: seu_intd_wt_AZ = readRDS("../data/intd_seu_objects/AZ_only_WT_4_19_22.rds")

```

```

[ ]: resolution = .1
set.seed(42)
DefaultAssay(seu_intd_wt_AZ) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
# Run the standard workflow for visualization and clustering
seu_intd_wt_AZ <- ScaleData(seu_intd_wt_AZ, verbose = FALSE)
seu_intd_wt_AZ <- RunPCA(seu_intd_wt_AZ, npcs = 100, verbose = FALSE, approx = FALSE)
seu_intd_wt_AZ <- FindNeighbors(seu_intd_wt_AZ, dims = 1:20, verbose = FALSE)
seu_intd_wt_AZ <- FindClusters(seu_intd_wt_AZ, resolution = resolution, algorithm = 3, verbose = FALSE)
seu_intd_wt_AZ <- RunUMAP(seu_intd_wt_AZ, reduction = "pca", dims = 1:20, verbose = FALSE)

```

```
[23]: options(repr.plot.width=8, repr.plot.height=8)
DefaultAssay(seu_intd_wt_AZ) = "SCT"
plot = FeaturePlot(seu_intd_wt_AZ, "AT4G28490", min = .5, pt.size = 6, order = 1,
  ↪T)
ggsave(file="../data/for_figures/UMAPs/AZ_WT_HAE_featureplot.png", plot=plot,
  ↪width=8, height=8)
plot = FeaturePlot(seu_intd_wt_AZ, "AT1G68765", min = .5, pt.size = 6, order = 1,
  ↪T)
ggsave(file="../data/for_figures/UMAPs/AZ_WT_IDA_featureplot.png", plot=plot,
  ↪width=8, height=8)
```

```
[ ]: options(repr.plot.width= 10, repr.plot.height=10)
plot = DimPlot(seu_intd_wt_AZ, reduction = "umap", label = TRUE, pt.size = 4)
print(plot)
ggsave(file="../data/for_figures/UMAPs/AZ_WT_UMAP.png", plot=plot, width=10,
  ↪height=10)
```

```
[ ]: DefaultAssay(seu_intd_wt_AZ) <- "RNA"

#get pseudobulk for each cluster to compare with kwak data
pbs = list()
count = 1
for (l in levels(seu_intd_wt_AZ@meta.data$seurat_clusters)) {
  pbs[[count]] = rowSums(as.matrix(GetAssayData(seu_intd_wt_AZ, slot = "counts"),
  ↪WhichCells(seu_intd_wt_AZ, ident = l))))
  count = count + 1
}

#saveRDS(pbs, "../data/counts/AZ_wt_cluster_pbs_3_1_22")
```

```
[ ]: #convert pseudobulk to TPM
count = 1
for (c in pbs) {
  pbs[[count]] = data.frame(pbs[[count]])/sum(data.
  ↪frame(pbs[[count]]))*1000000
  rns = rownames(pbs[[count]])
  pbs[[count]] = pbs[[count]][order(rns),, drop = FALSE]
  count = count + 1
}
```

```
[ ]: #secession
kwak_ptpms_raw=read.csv("../data/counts/kwak_ptpms.csv")
rownames(kwak_ptpms_raw) = kwak_ptpms_raw$X
kwak_ptpms = kwak_ptpms_raw
kwak_ptpms[,c(1,3,4)] =NULL

#secession
```



```

#set dataset
dataset = kwak_ptpms

cors_spearman = vector()
count = 1

seu_intd_wt_AZ@meta.data$kwak_cor = NULL

for (cluster in c(1:length(levels(seu_intd_wt_AZ@meta.data$seurat_clusters)))){
  test = cbind(pbs[[cluster]][intersect(rownames(pbs[[cluster]]),
↪rownames(dataset)),], dataset[intersect(rownames(pbs[[cluster]]),
↪rownames(dataset)),])
  cors_spearman[count] = cor(log(test[,1]+.1), log(test[,2]+.1), method =
↪"spearman")
  count = count + 1
}

for (i in c(1:length(levels(seu_intd_wt_AZ@meta.data$seurat_clusters)))){
  seu_intd_wt_AZ@meta.data$kwak_cor[seu_intd_wt_AZ@meta.data$seurat_clusters
↪== toString(i-1)] = cors_spearman[i]
}

plot = FeaturePlot(seu_intd_wt_AZ, features = "kwak_cor", pt.size = 4, cols =
↪c("light gray", "red"))
print(plot)
ggsave(file="../data/for_figures/UMAPs/AZ_WT_sec_UMAP.png", plot=plot,
↪width=10, height=10)

```

```

[ ]: #residuum
kwak_ptpms_raw=read.csv("../data/counts/kwak_ptpms.csv")
rownames(kwak_ptpms_raw) = kwak_ptpms_raw$X
kwak_ptpms = kwak_ptpms_raw
kwak_ptpms[,c(1,2,4)] =NULL

#residuum
#set dataset
dataset = kwak_ptpms

cors_spearman = vector()
count = 1

seu_intd_wt_AZ@meta.data$kwak_cor = NULL

for (cluster in c(1:length(levels(seu_intd_wt_AZ@meta.data$seurat_clusters)))){
  test = cbind(pbs[[cluster]][intersect(rownames(pbs[[cluster]]),
↪rownames(dataset)),], dataset[intersect(rownames(pbs[[cluster]]),
↪rownames(dataset)),])

```

```

    cors_spearman[count] = cor(log(test[,1]+.1), log(test[,2]+.1), method =
↪ "spearman")
    count = count + 1
}

for (i in c(1:length(levels(seu_intd_wt_AZ@meta.data$seurat_clusters)))){
    seu_intd_wt_AZ@meta.data$kwak_cor[seu_intd_wt_AZ@meta.data$seurat_clusters
↪ == toString(i-1)] = cors_spearman[i]
}

plot = FeaturePlot(seu_intd_wt_AZ, features = "kwak_cor", pt.size = 4, cols =
↪ c("light gray", "red"))
print(plot)
ggsave(file="../data/for_figures/UMAPs/AZ_WT_res_UMAP.png", plot=plot,
↪ width=10, height=10)

```

```

[ ]: DefaultAssay(seu_intd_wt_AZ) <- "RNA"
wt_sec_v_rec = data.frame(matrix(ncol = 8, nrow
↪ =dim(seu_intd_wt_AZ@assays$RNA)[1]))
wt_sec_v_rec_red = data.frame(matrix(ncol = 6, nrow
↪ =dim(seu_intd_wt_AZ@assays$RNA)[1]))

res1_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
↪ ident == "sc_26_combined"), slot = "counts")[,
↪ WhichCells(subset(seu_intd_wt_AZ, subset = orig.ident == "sc_26_combined"),
↪ ident = "0"))))
res2_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
↪ ident == "sc_67"), slot = "counts")[, WhichCells(subset(seu_intd_wt_AZ,
↪ subset = orig.ident == "sc_67"), ident = "0"))))
res3_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
↪ ident == "sc_101"), slot = "counts")[, WhichCells(subset(seu_intd_wt_AZ,
↪ subset = orig.ident == "sc_101"), ident = "0"))))
res4_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
↪ ident == "sc_103"), slot = "counts")[, WhichCells(subset(seu_intd_wt_AZ,
↪ subset = orig.ident == "sc_103"), ident = "0"))))

sec1_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
↪ ident == "sc_26_combined"), slot = "counts")[,
↪ WhichCells(subset(seu_intd_wt_AZ, subset = orig.ident == "sc_26_combined"),
↪ ident = "1"))))
sec2_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
↪ ident == "sc_67"), slot = "counts")[, WhichCells(subset(seu_intd_wt_AZ,
↪ subset = orig.ident == "sc_67"), ident = "1"))))
sec3_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
↪ ident == "sc_101"), slot = "counts")[, WhichCells(subset(seu_intd_wt_AZ,
↪ subset = orig.ident == "sc_101"), ident = "1"))))

```

```

sec4_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_wt_AZ, subset = orig.
  ↳ ident == "sc_103"), slot = "counts")[, WhichCells(subset(seu_intd_wt_AZ,
  ↳ subset = orig.ident == "sc_103"), ident = "1")]))

wt_sec_v_rec[,1:8] = c(res1_1, res2_1, res3_1, res4_1, sec1_1, sec2_1, sec3_1,
  ↳ sec4_1)
wt_sec_v_rec_red[,1:6] = c(res1_1, res2_1, res3_1 + res4_1, sec1_1, sec2_1,
  ↳ sec3_1 + sec4_1)

```

```

[ ]: colnames(wt_sec_v_rec) = c(rep("res",4), rep("sec",4))
colnames(wt_sec_v_rec_red) = c(rep("res",3), rep("sec",3))
rownames(wt_sec_v_rec) = names(res1_1)
rownames(wt_sec_v_rec_red) = names(res1_1)

```

```

[ ]: zone=as.factor(c(rep("res",4), rep("sec",4)))
design <- model.matrix(~zone)#+insertion

#check design matrix isn't singular
print(paste("determinant of XT*X of design matrix is: ",
  ↳ det(t(design)%*%(design))))

#making contrast matrix for tests of interest
my.contrasts <- makeContrasts(s1_v_s2=zonesec, levels=design)
wt_zone_edger_1 = edgeR_2_sample(wt_sec_v_rec, "res", "sec", c(1,2,3,4),
  ↳ c(5,6,7,8), annotations, design, my.contrasts)

```

```

[ ]: zone=as.factor(c(rep("res",3), rep("sec",3)))
sort=as.factor(c("u","u","s","u","u","s"))
design <- model.matrix(~zone + sort)#+insertion

#check design matrix isn't singular
print(paste("determinant of XT*X of design matrix is: ",
  ↳ det(t(design)%*%(design))))

#making contrast matrix for tests of interest
my.contrasts <- makeContrasts(s1_v_s2=zonesec, levels=design)
wt_zone_edger_red = edgeR_2_sample(wt_sec_v_rec_red, "res", "sec", c(1,2,3),
  ↳ c(4,5,6), annotations, design, my.contrasts)

```

```

[ ]: head(wt_zone_edger_red[wt_zone_edger_red$FDR < .05,],20)

```

```

[ ]: write.csv(wt_zone_edger_1, "../data/for_figures/wt_zone_edger_4_21_22.csv")
write.csv(wt_zone_edger_red, "../data/for_figures/wt_zone_edger_red_4_21_22.
  ↳ csv")

```

```

[ ]: wt_zone_edger_1[wt_zone_edger_1$genes=="AT1G01610",]

```

```
[ ]: cluster = "11"

mut_1_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_27_combined")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_27_combined"),
  ↳ ident = cluster)]
mut_2_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_68")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_68"), ident =
  ↳ cluster)]
KE_1_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_102")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_102"), ident =
  ↳ cluster)]
KE_2_AZ <- subset(seu_intd_wt_mut, subset = orig.ident == "sc_104")[,
  ↳ WhichCells(subset(seu_intd_wt_mut, subset = orig.ident == "sc_104"), ident =
  ↳ cluster)]
```

```
[ ]: mut_1_seu = SCTransform(mut_1_AZ)
mut_2_seu = SCTransform(mut_2_AZ)
KE_1_seu = SCTransform(KE_1_AZ)
KE_2_seu = SCTransform(KE_2_AZ)
```

```
[ ]: seu_intd_mut_AZ = seu_integrate(mut_1_seu, mut_2_seu, KE_1_seu, KE_2_seu,
  ↳ filename = "AZ_only_mut_3_1_22", nfeatures = 3000)
```

```
[ ]: resolution = .1
set.seed(42)
DefaultAssay(seu_intd_mut_AZ) <- "integrated"
options(repr.plot.width=12, repr.plot.height=12)
# Run the standard workflow for visualization and clustering
seu_intd_mut_AZ <- ScaleData(seu_intd_mut_AZ, verbose = FALSE)
seu_intd_mut_AZ <- RunPCA(seu_intd_mut_AZ, npcs = 100, verbose = FALSE, approx
  ↳ = FALSE)
seu_intd_mut_AZ <- FindNeighbors(seu_intd_mut_AZ, dims = 1:20, verbose = FALSE)
seu_intd_mut_AZ <- FindClusters(seu_intd_mut_AZ, resolution = resolution,
  ↳ algorithm = 3, verbose = FALSE)
seu_intd_mut_AZ <- RunUMAP(seu_intd_mut_AZ, reduction = "pca", dims = 1:20,
  ↳ verbose = FALSE)
```

```
[ ]: options(repr.plot.width= 10, repr.plot.height=10)
plot = DimPlot(seu_intd_mut_AZ, reduction = "umap", label = TRUE, pt.size = 4)
print(plot)
ggsave(file="../data/for_figures/UMAPs/AZ_mut_UMAP.png", plot=plot, width=10,
  ↳ height=10)
```

```
[ ]: DefaultAssay(seu_intd_mut_AZ) <- "RNA"

#get pseudobulk for each cluster to compare with kwak data
```

```

pbs_mut = list()
count = 1
for (l in levels(seu_intd_mut_AZ@meta.data$seurat_clusters)) {
  pbs_mut[[count]] = rowSums(as.matrix(GetAssayData(seu_intd_mut_AZ, slot = "counts"), WhichCells(seu_intd_mut_AZ, ident = 1))))
  count = count + 1
}

```

```

[ ]: #convert pseudobulk to TPM
count = 1
for (c in pbs_mut) {
  pbs_mut[[count]] = data.frame(pbs_mut[[count]])/sum(data.frame(pbs_mut[[count]]))*1000000
  rns = rownames(pbs_mut[[count]])
  pbs_mut[[count]] = pbs_mut[[count]][order(rns),, drop = FALSE]
  count = count + 1
}

```

```

[ ]: #secession
kwak_ptpms_raw=read.csv("../data/counts/kwak_ptpms.csv")
rownames(kwak_ptpms_raw) = kwak_ptpms_raw$X
kwak_ptpms = kwak_ptpms_raw
kwak_ptpms[,c(1,3,4)] =NULL

#secession
#set dataset
dataset = kwak_ptpms

cors_spearman = vector()
count = 1

seu_intd_mut_AZ@meta.data$kwak_cor = NULL

for (cluster in c(1:length(levels(seu_intd_mut_AZ@meta.data$seurat_clusters)))){
  test = cbind(pbs_mut[[cluster]][intersect(rownames(pbs_mut[[cluster]]), rownames(dataset)),], dataset[intersect(rownames(pbs_mut[[cluster]]), rownames(dataset)),])
  cors_spearman[count] = cor(log(test[,1]+.1), log(test[,2]+.1), method = "spearman")
  count = count + 1
}

for (i in c(1:length(levels(seu_intd_mut_AZ@meta.data$seurat_clusters)))){
  seu_intd_mut_AZ@meta.data$kwak_cor[seu_intd_mut_AZ@meta.data$seurat_clusters == toString(i-1)] = cors_spearman[i]
}

```

```

plot = FeaturePlot(seu_intd_mut_AZ, features = "kwak_cor", pt.size = 4, cols =
  ↪c("light gray", "red"))
print(plot)
ggsave(file="../data/for_figures/UMAPs/AZ_mut_sec_UMAP.png", plot=plot,
  ↪width=10, height=10)

```

```

[ ]: #residuum
kwak_ptpms_raw=read.csv("../data/counts/kwak_ptpms.csv")
rownames(kwak_ptpms_raw) = kwak_ptpms_raw$X
kwak_ptpms = kwak_ptpms_raw
kwak_ptpms[,c(1,2,4)] =NULL

#residuum
#set dataset
dataset = kwak_ptpms

cors_spearman = vector()
count = 1

seu_intd_mut_AZ@meta.data$kwak_cor = NULL

for (cluster in c(1:length(levels(seu_intd_mut_AZ@meta.data$seurat_clusters)))){
  test = cbind(pbs_mut[[cluster]][intersect(rownames(pbs_mut[[cluster]]),
  ↪rownames(dataset)),],dataset[intersect(rownames(pbs_mut[[cluster]]),
  ↪rownames(dataset)),])
  cors_spearman[count] = cor(log(test[,1]+.1), log(test[,2]+.1), method =
  ↪"spearman")
  count = count + 1
}

for (i in c(1:length(levels(seu_intd_mut_AZ@meta.data$seurat_clusters)))){
  seu_intd_mut_AZ@meta.data$kwak_cor[seu_intd_mut_AZ@meta.
  ↪data$seurat_clusters == toString(i-1)] = cors_spearman[i]
}

plot = FeaturePlot(seu_intd_mut_AZ, features = "kwak_cor", pt.size = 4, cols =
  ↪c("light gray", "red"))
print(plot)
ggsave(file="../data/for_figures/UMAPs/AZ_mut_res_UMAP.png", plot=plot,
  ↪width=10, height=10)

```

```

[ ]: DefaultAssay(seu_intd_mut_AZ) <- "RNA"
mut_sec_v_rec = data.frame(matrix(ncol = 8, nrow
  ↪=dim(seu_intd_mut_AZ@assays$RNA)[1]))

```

```

mut_sec_v_rec_red = data.frame(matrix(ncol = 6, nrow =
  ↪dim(seu_intd_mut_AZ@assays$RNA)[1]))

res1_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_27_combined"), slot = "counts")[,
  ↪WhichCells(subset(seu_intd_mut_AZ, subset = orig.ident == "sc_27_combined"),
  ↪ident = "0"))))
res2_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_68"), slot = "counts")[, WhichCells(subset(seu_intd_mut_AZ,
  ↪subset = orig.ident == "sc_68"), ident = "0"))))
res3_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_102"), slot = "counts")[, WhichCells(subset(seu_intd_mut_AZ,
  ↪subset = orig.ident == "sc_102"), ident = "0"))))
res4_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_104"), slot = "counts")[, WhichCells(subset(seu_intd_mut_AZ,
  ↪subset = orig.ident == "sc_104"), ident = "0"))))

sec1_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_27_combined"), slot = "counts")[,
  ↪WhichCells(subset(seu_intd_mut_AZ, subset = orig.ident == "sc_27_combined"),
  ↪ident = "1"))))
sec2_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_68"), slot = "counts")[, WhichCells(subset(seu_intd_mut_AZ,
  ↪subset = orig.ident == "sc_68"), ident = "1"))))
sec3_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_102"), slot = "counts")[, WhichCells(subset(seu_intd_mut_AZ,
  ↪subset = orig.ident == "sc_102"), ident = "1"))))
sec4_1 = rowSums(as.matrix(GetAssayData(subset(seu_intd_mut_AZ, subset = orig.
  ↪ident == "sc_104"), slot = "counts")[, WhichCells(subset(seu_intd_mut_AZ,
  ↪subset = orig.ident == "sc_104"), ident = "1"))))

mut_sec_v_rec[,1:8] = c(res1_1, res2_1, res3_1, res4_1, sec1_1, sec2_1, sec3_1,
  ↪sec4_1 )
mut_sec_v_rec_red[,1:6] = c(res1_1, res2_1, res3_1 + res4_1, sec1_1, sec2_1,
  ↪sec3_1 + sec4_1 )

```

```

[ ]: colnames(mut_sec_v_rec) = c(rep("res",4), rep("sec",4))
colnames(mut_sec_v_rec_red) = c(rep("res",3), rep("sec",3))
rownames(mut_sec_v_rec) = names(res1_1)
rownames(mut_sec_v_rec_red) = names(res1_1)

```

```

[ ]: zone=as.factor(c(rep("res",4), rep("sec",4)))
design <- model.matrix(~zone)#+insertion)

```

```

#check design matrix isn't singular
print(paste("determinant of XT*X of design matrix is: ",
  ↪det(t(design)%*%(design))))

#making contrast matrix for tests of interest
my.contrasts <- makeContrasts(s1_v_s2=zonesec, levels=design)
mut_zone_edger_1 = edgeR_2_sample(mut_sec_v_rec, "res", "sec", c(1,2,3,4),
  ↪c(5,6,7,8), annotations, design, my.contrasts)

```

```

[ ]: #combined sorted samples
zone=as.factor(c(rep("res",3), rep("sec",3)))
sort=as.factor(c("u","u","s","u","u","s"))
design <- model.matrix(~zone + sort)#+insertion)

#check design matrix isn't singular
print(paste("determinant of XT*X of design matrix is: ",
  ↪det(t(design)%*%(design))))

#making contrast matrix for tests of interest
my.contrasts <- makeContrasts(s1_v_s2=zonesec, levels=design)
mut_zone_edger_red = edgeR_2_sample(mut_sec_v_rec_red, "res", "sec", c(1,2,3),
  ↪c(4,5,6), annotations, design, my.contrasts)

```

```

[ ]: head(mut_zone_edger_red[mut_zone_edger_red$FDR<.05,])

```

```

[ ]: write.csv(mut_zone_edger_1, "../data/for_figures/mut_zone_edger_4_21_22.csv")
write.csv(mut_zone_edger_red, "../data/for_figures/mut_zone_edger_red_4_21_22.
  ↪csv")

```

```

[ ]: kwak = read.csv("../data/for_figures/KWAK_data.csv")
rownames(kwak) = kwak[,1]
kwak = kwak[,c(5:10)]
colnames(kwak) = c("res", "res", "res", "sec", "sec", "sec")
kwak = kwak[c(1:33602),]

```

```

[ ]: zone=as.factor(c(rep("res",3), rep("sec",3)))
design <- model.matrix(~zone)#+insertion)

#check design matrix isn't singular
print(paste("determinant of XT*X of design matrix is: ",
  ↪det(t(design)%*%(design))))

#making contrast matrix for tests of interest
my.contrasts <- makeContrasts(s1_v_s2=zonesec, levels=design)

```

```

[ ]: kwak_edger_1 = edgeR_2_sample(kwak, "res", "sec", c(1,2,3), c(4,5,6),
  ↪annotations, design, my.contrasts)

```



```
[ ]: write.csv(kwak_edger_1, "../data/for_figures/kwak_edger_4_21_22.csv")
```

```
[ ]: #takes a list of Seurat objects with SCT transform run
seu_integrate <- function(..., filename, nfeatures){
  seu.list <- list(...) # THIS WILL BE A LIST STORING EVERYTHING:

  ref.genes = rownames(seu.list[[1]]@assays$RNA)
  assay_list <- rep("SCT", length(seu.list))

  # integration
  rc.features <- SelectIntegrationFeatures(object.list = seu.list, nfeatures =
  nfeatures)
  rc.features <- rc.features[(!c(grepl("ATMG",rc.features) | grepl("ATCG",rc.
  features) | rc.features%in%proto_list))]

  seu.list <- PrepSCTIntegration(object.list = seu.list, anchor.features = rc.
  features, verbose = TRUE, assay = assay_list)
  seu.list <- lapply(X = seu.list, FUN = RunPCA, verbose = FALSE, features =
  rc.features)
  rc.anchors <- FindIntegrationAnchors(object.list = seu.list, normalization.
  method = "SCT", anchor.features = rc.features, verbose = TRUE, reference=1,
  reduction = "rpca")

  to_integrate <- Reduce(intersect, lapply(rc.anchors@object.list, rownames))
  # integrate data and keep full geneset

  rc.integrated <- IntegrateData(anchorset = rc.anchors, features.to.
  integrate = to_integrate, normalization.method = "SCT", verbose = TRUE)
  rc.integrated <- RunPCA(rc.integrated, npcs = 50, verbose = FALSE, approx =
  FALSE)

  #save object
  saveRDS(rc.integrated, file = paste("../data/intd_seu_objects/",filename,".
  rds", sep = ""))
  return(rc.integrated)
# }
}
```

# WB07\_sc\_vs\_bulk\_RNA\_seq

June 30, 2022

```
[1]: #THIS SCRIPT CALCULATES THE INTERSECTION OF DE GENES OF PRIOR BULK COMPARISON
      ↪ OF WT AND MUTANT COMPARED TO SINGLE-CELL ANALYSIS

library(edgeR)
library(here)
source(here("R_functions", "edgeR_function.R"))

annotations = read.csv("R_functions/gene_descriptions.csv", header = F)
colnames(annotations) = c("gene_id", "description")
annotations$gene_id = substr(annotations$gene_id, 1, 9)
```

Loading required package: limma

here() starts at /home/robotmessenger810/sc\_analysis/code

```
[2]: sessionInfo()
```

R version 3.6.3 (2020-02-29)

Platform: x86\_64-conda\_cos6-linux-gnu (64-bit)

Running under: Ubuntu 20.04.2 LTS

Matrix products: default

BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r\_3/lib/libopenblas-r0.3.9.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] here_0.1      edgeR_3.28.1  limma_3.42.2
```

loaded via a namespace (and not attached):

```
[1] Rcpp_1.0.7      locfit_1.5-9.4  lattice_0.20-45 fansi_0.5.0
[5] rprojroot_2.0.2 digest_0.6.29   utf8_1.2.2      crayon_1.4.2
[9] IRdisplay_0.7.0 grid_3.6.3      repr_1.1.0      lifecycle_1.0.1
[13] jsonlite_1.7.2  evaluate_0.14   pillar_1.6.4    rlang_0.4.12
[17] uuid_0.1-4      vctrs_0.3.8     ellipsis_0.3.2  IRkernel_1.1
[21] tools_3.6.3     compiler_3.6.3  base64enc_0.1-3 pbdZMQ_0.3-6
[25] htmltools_0.5.0
```

```
[ ]: #bulk samples
bulk = counts_to_reads_df("../data/bulk_data/Col_v_h3h3_bulk_counts_first" )

#remove no_feature, ambiguous, etc reads
bulk = bulk[1:(dim(bulk)[1]-5),]

#account for factors in experiment
phenotype=as.factor(c("wt", "wt", "wt", "wt", "wt", "wt", "mut", "mut", "mut", "mut", "mut", "mut", "mut"))
batch=as.factor(c(0,0,0,1,1,1,0,0,0,1,1,1))
design <- model.matrix(~phenotype+batch)#+insertion)

#double check design matrix isn't singular
print(paste("determinant of XT*X of design matrix is: ",
  det(t(design)%*%(design))))

#making contrast matrix for tests of interest
my.contrasts <- makeContrasts(s1_v_s2=phenotypewt, levels=design)

#run edgeR
bulk_edger_2 = edgeR_2_sample(bulk, "WT", "mut", c(1,2,3,4,5,6),
  c(7,8,9,10,11,12), annotations, design, my.contrasts)
```

```
[ ]: #pseudo bulk samples
pb = read.csv("../data/pseudo_bulk_data/AZ_pbs_4_19_22.csv")
rownames(pb) = pb[,1]
pb[,1] <- NULL

#account for factors in experiment
phenotype=as.factor(c("wt", "wt", "wt", "wt", "wt", "mut", "mut", "mut", "mut"))
method=as.factor(c(0,0,1,1,0,0,1,1))
design <- model.matrix(~phenotype+method)#+insertion)

#double check design matrix isn't singular
print(paste("determinant of XT*X of design matrix is: ",
  det(t(design)%*%(design))))
```

```

#making contrast matrix for tests of interest
my.contrasts <- makeContrasts(s1_v_s2=phenotypewt, levels=design)

#run edgeR
pb_edger_1 = edgeR_2_sample(pb, "WT", "mut", c(1,2,3,4), c(5,6,7,8),
  ↪ annotations, design, my.contrasts)

```

```

[ ]: sig_intersect_WT_up = intersect(pb_edger_1[pb_edger_1$FDR<
  ↪ 0.05 & pb_edger_1$logFC>1,]$genes, bulk_edger_2[bulk_edger_2$FDR<
  ↪ 0.05 & bulk_edger_2$logFC>1,]$genes)
length(sig_intersect_WT_up)
write.csv(sig_intersect_WT_up, "../data/pseudo_bulk_data/
  ↪ pb_bulk_sig_intersect_WT_up.csv")

```

# WB08\_fal

June 30, 2022

```
[ ]: #THIS SCRIPT PLOTS A HEATMAP OF BULK RNA-SEQ GENE EXPRESSION VALUES FOR WT,  
      ↪MUTANT, AND THE FAL SUPPRESSORS USING THE LIST OF GENES GENERATED FROM  
      ↪SCRIPT 6
```

```
library(limma)
library(edgeR)
library(here)
library(ggplot2)
library(gplots)
source(here("R_functions", "edgeR_function.R"))
annotations = read.csv("R_functions/gene_descriptions.csv", header = F)
colnames(annotations) = c("gene_id", "description")
annotations$gene_id = substr(annotations$gene_id, 1, 9)

library(pheatmap)
```

```
[2]: sessionInfo()
```

R version 3.6.3 (2020-02-29)

Platform: x86\_64-conda\_cos6-linux-gnu (64-bit)

Running under: Ubuntu 20.04.2 LTS

Matrix products: default

BLAS/LAPACK: /home/robotmessenger810/anaconda3/envs/r\_3/lib/libopenblas-r0.3.9.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] pheatmap_1.0.12 gplots_3.1.1    ggplot2_3.3.5   here_0.1
```

```
[5] edgeR_3.28.1    limma_3.42.2
```

loaded via a namespace (and not attached):

```
[1] Rcpp_1.0.7          RColorBrewer_1.1-2 pillar_1.6.4      compiler_3.6.3
[5] bitops_1.0-7        base64enc_0.1-3    tools_3.6.3       digest_0.6.29
[9] uuid_0.1-4          jsonlite_1.7.2     evaluate_0.14     lifecycle_1.0.1
[13] tibble_3.1.6        gtable_0.3.0       lattice_0.20-45   pkgconfig_2.0.3
[17] rlang_0.4.12        DBI_1.1.2          IRdisplay_0.7.0   IRkernel_1.1
[21] withr_2.4.3         repr_1.1.0         dplyr_1.0.7       caTools_1.18.0
[25] gtools_3.9.2        generics_0.1.1     vctrs_0.3.8       tidyselect_1.1.1
[29] locfit_1.5-9.4      rprojroot_2.0.2    grid_3.6.3        glue_1.6.0
[33] R6_2.5.1            fansi_0.5.0        pbdZMQ_0.3-6      purrr_0.3.4
[37] magrittr_2.0.1      scales_1.1.1       ellipsis_0.3.2    htmltools_0.5.0
[41] colorspace_2.0-2    KernSmooth_2.23-20 utf8_1.2.2         munsell_0.5.0
[45] crayon_1.4.2
```

```
[ ]: #bulk_pb_intersection
bulk_pb_intersection = read.csv("../data/pseudo_bulk_data/
  ↳pb_bulk_sig_intersect_WT_up.csv")

[ ]: #bulk samples
bulk = counts_to_reads_df("../data/bulk_data/Col_h3h3_fals")

[3]: colnames(bulk) =_
  ↳c("Col_1", "fal7_1", "Col_2", "fal7_2", "Col_3", "fal7_3", "Col_4", "fal7_4", "h3h3_1", "fal3_1", "h3
  ↳h3h3_3", "fal3_3", "h3h3_4", "fal3_4")

[ ]: #remove poor h3h3 sample
bulk2 = bulk[,-13]
head(bulk2)

[ ]: #make DGElist
x3 <- DGEList(counts = bulk2, genes = rownames(bulk2))

#reads per library uniquely mapped to a gene:

#make cpm and lcpm
cpm <- cpm(x3)
lcpm <- cpm(x3, log=TRUE)

#keep only genes that are expressed. "Expressed" here means counts observed in_
  ↳at least 3 samples
dim(x3)
keep.exprs <- rowMeans(cpm)>=.5
x3 <- x3[keep.exprs,, keep.lib.sizes=FALSE]
```

```
dim(x3) #compare to dim(x) above

#normalize data after removing low expressed genes
x3 <- calcNormFactors(x3)

#cpm, lcpm of normalized values
cpm <- cpm(x3)
lcpm <- cpm(x3, log=TRUE)

de_sub = lcpm[rownames(lcpm) %in% unlist(bulk_pb_intersection$x),]
```

```
[ ]: head(cpm)
write.csv(cpm, "../data/bulk_data/fal.csv")
```

```
[ ]: de_sub_av = cbind(rowMeans(de_sub[,c(1,3,5,7)]), rowMeans(de_sub[,c(9,11,14)]),
  ↳ rowMeans(de_sub[,c(2,4,6,8)]), rowMeans(de_sub[,c(10,12,13, 15)]))
```

```
[ ]: colnames(de_sub_av) = c("wt", "h3h3", "fal7", "fal3")
```

```
[ ]: de_sub_av = de_sub_av[,c(1,2,4,3)]
fals = de_sub_av
```

```
[ ]: dim(de_sub)
```

```
[ ]: #only WT up genes
mat = de_sub_av-rowMeans(de_sub_av[,c(1,2)])
hmp = heatmap(mat, Rowv=NA, Colv=NA)
mat = mat[rev(hmp$rowInd),]
hmp = heatmap(mat, Rowv=NA, Colv=NA)
```

```
[ ]: pheatmap(mat, cluster_rows=FALSE, cluster_cols=FALSE)
```

```
[ ]: for_plot = data.frame(matrix(ncol = 2, nrow = length(colSums(de_sub))))
colnames(for_plot) = c("genotype", "value")
for_plot$value= colSums(de_sub)
for_plot$genotype = substr(names(colSums(de_sub)),1,4)
for_plot
ggplot(data = for_plot, aes(x=genotype, y = value)) +geom_point()
```