Extraccction Insc by fly Cell Atlas

Code ▾

Oscar Mendoza

2025-04-11

#load library

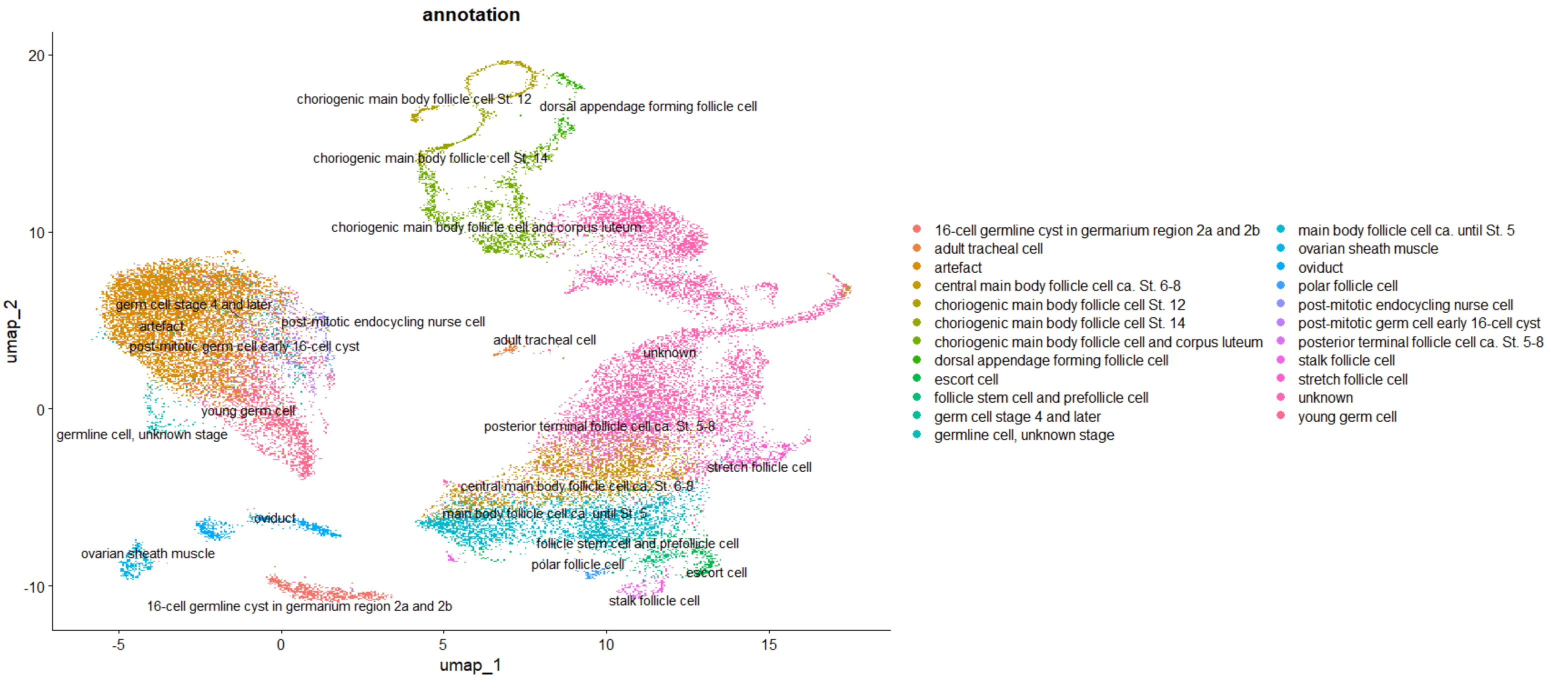
Hide

```
library(VennDiagram)
library(Seurat)
library(ggplot2)
library(ggrepel)
library(SeuratDisk)
```

Load data

Hide

```
#adata <- LoadH5Seurat("D:/Datos_Loom/New_section/1_try/ovary.h5seurat")
#check Data
DimPlot(adata, reduction = "umap", group.by = "annotation", label = TRUE, pt.size = 1, raster = T, repel =
T)
```



Hide

```
NA
NA
```

Identify cells expressing "insc" > 0.5

Hide

```

gene_of_interest <- "insc"
annotation_of_interest <- "young germ cell"

# Identificar células que expresan el gen de interés > 0.5
gene_expression <- adata@assays$RNA@data[gene_of_interest, ]
annotation_cells <- colnames(adata)[adata$annotation == annotation_of_interest]
expressers_positive <- gene_expression[annotation_cells] > 0.5
```

Hide

```
# Create two subsets: insc+ and insc-
annotation_with_expression <- annotation_cells[expressers_positive]
annotation_without_expression <- annotation_cells[!expressers_positive]
```

Hide

```
# Extract the expression matrix
group1_expression <- adata@assays$RNA@data[, annotation_with_expression]
group2_expression <- adata@assays$RNA@data[, annotation_without_expression]
```

Hide

```
# List of genes
all_genes <- rownames(adata@assays$RNA@data)
```

Hide

```
# Calculate the fold change and p-values
results <- data.frame(Gene = all_genes, logFC = numeric(length(all_genes)), p.value = numeric(length(all_genes)),
                      mean_expressers = numeric(length(all_genes)), mean_non_expressers = numeric(length(all_genes)))
```

T-test code

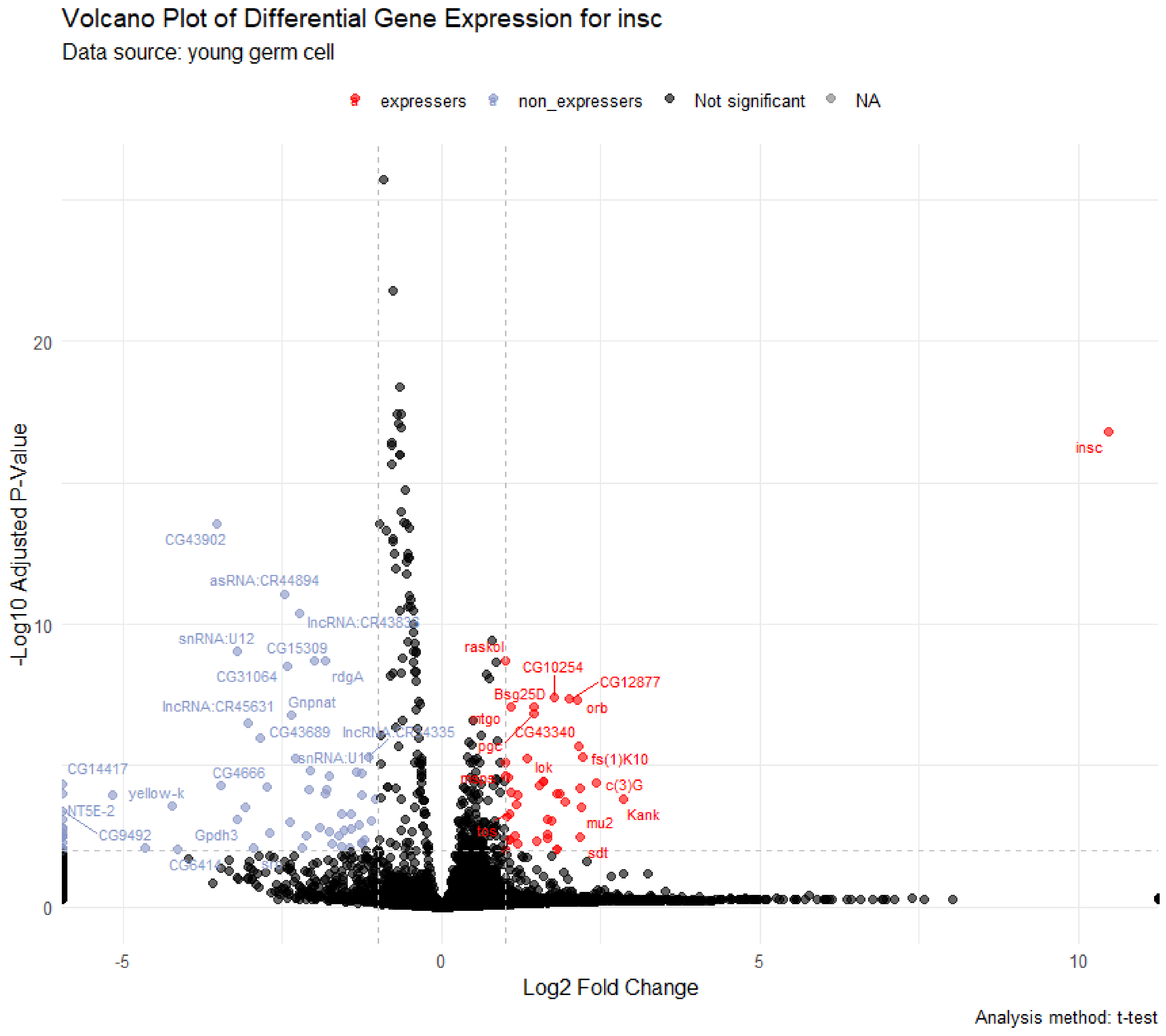
Hide

```
for (gene in all_genes) {
  t_test_result <- t.test(group2_expression[gene, ],group1_expression[gene, ])
  fold_change <-  mean(group1_expression[gene, ])/mean(group2_expression[gene, ])
  log2_fold_change <- log2(fold_change)
  results[results$Gene == gene, "logFC"] <- log2_fold_change
  results[results$Gene == gene, "p.value"] <- t_test_result$p.value
}

# Adjust p-values
results$adj.p.value <- p.adjust(results$p.value, method = "BH")


results$annotation <- ifelse(abs(results$logFC) > 1 & results$adj.p.value < 0.05,
                             ifelse(results$logFC > 0, "with_insc", "without_insc"
                                    ),
                             "Not significant")
```

Graphic

Hide

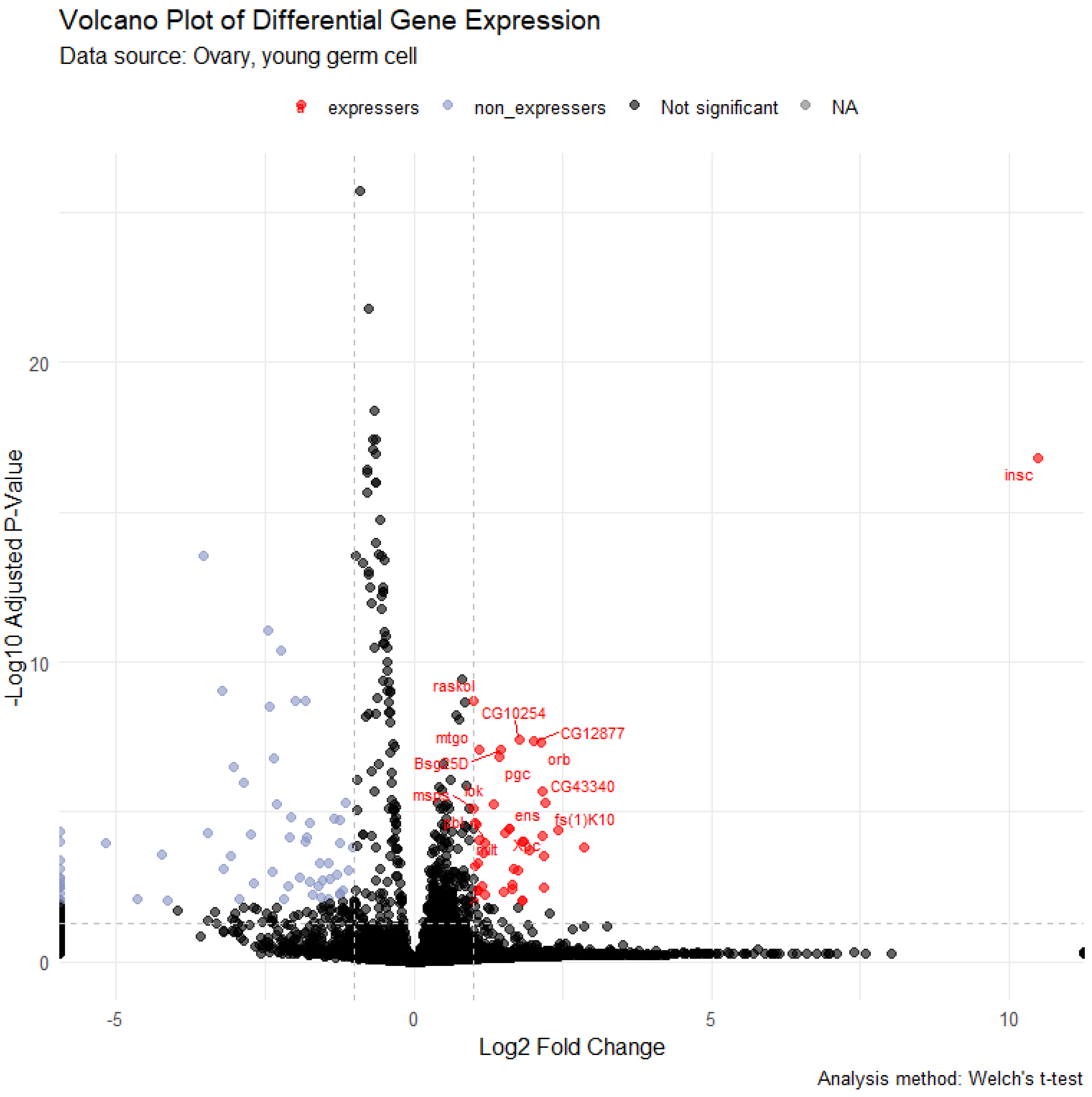
```
# Volcano Plot Chart
p <- ggplot(results, aes(x = logFC, y = -log10(adj.p.value), color = annotation)) +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("non_expressers" = "#8998C8",
                                "expressers" = "red",
                                "Not significant" = "black")) +
  theme_minimal() +
  labs(
    title = paste("Volcano Plot of Differential Gene Expression for", gene_of_interest),
    subtitle = paste("Data source:", annotation_of_interest),
    x = "Log2 Fold Change",
    y = "-Log10 Adjusted P-Value",
    caption = "Analysis method: t-test"
  ) +
  geom_hline(yintercept = -log10(0.01), color = "grey", linetype = "dashed") +
  geom_vline(xintercept = c(-1, 1), color = "grey", linetype = "dashed") +
  geom_text_repel(data = subset(results, abs(logFC) > 1 & -log10(adj.p.value) > 2),
                  aes(label = Gene), size = 3) +
  theme(
    text = element_text(family = "Arial", size = 12),
    legend.position = "top",
    legend.title = element_blank()
  )

print(p)
```



Dan image:

By gene significance in Insc

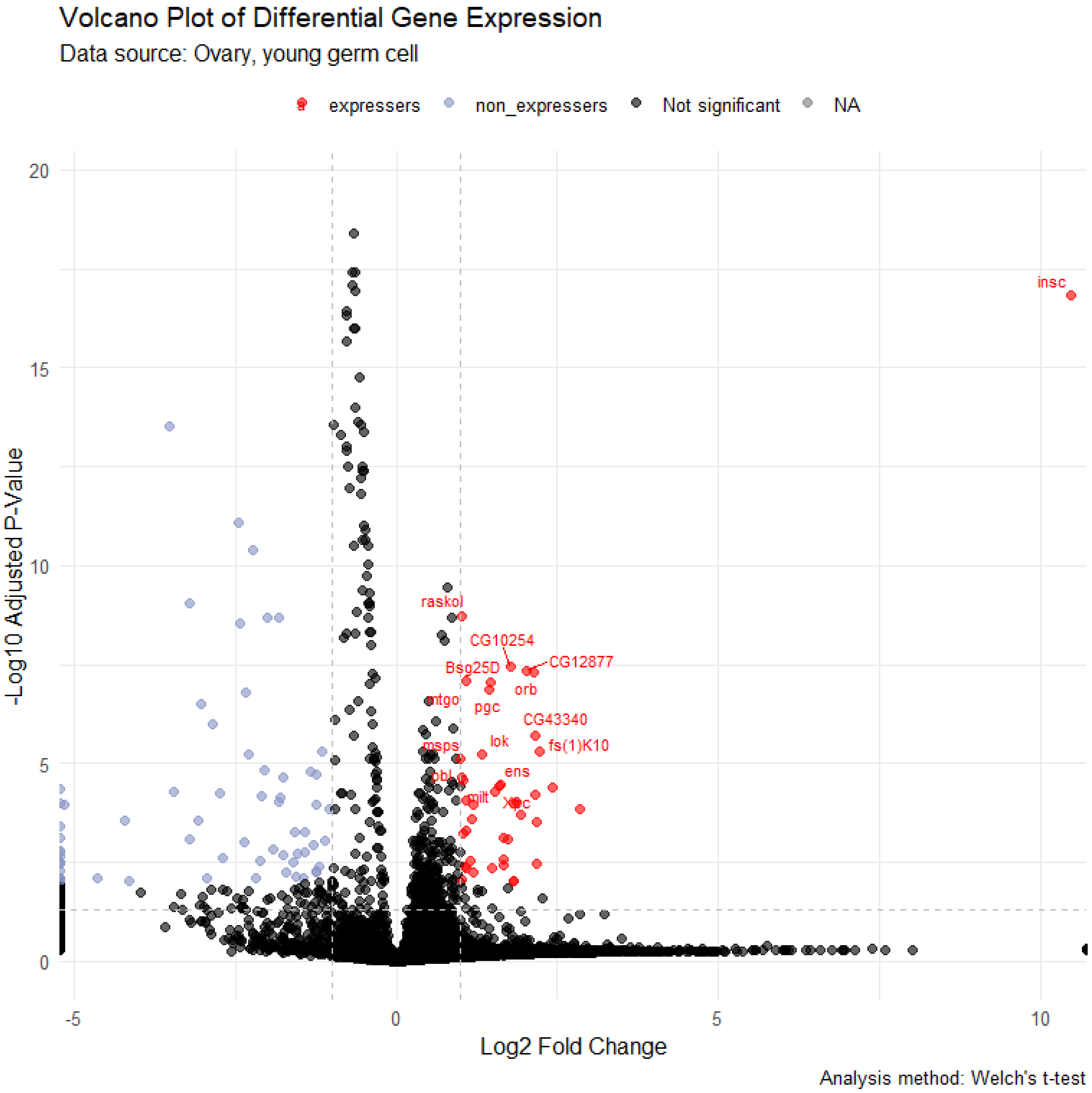
```
# Filter out only significant with_insc genes and sort by tight p
with_insc_top <- subset(results, annotation == "expressers" & abs(logFC) > 1 & -log10(adj.p.value) > 2)
with_insc_top <- with_insc_top[order(with_insc_top$adj.p.value), ][1:16, ]  # Top 20 significativos

# Volcano Plot with labels only for those 20
p <- ggplot(results, aes(x = logFC, y = -log10(adj.p.value), color = annotation)) +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("non_expressers" = "#8998C8", "expressers" = "red", "Not significant" = "bl
ack")) +
  theme_minimal() +
  labs(
    title = "Volcano Plot of Differential Gene Expression",
    subtitle = "Data source: Ovary, young germ cell",
    x = "Log2 Fold Change",
    y = "-Log10 Adjusted P-Value",
    caption = "Analysis method: Welch's t-test"
  ) +
  geom_hline(yintercept = -log10(0.05), color = "grey", linetype = "dashed") +
  geom_vline(xintercept = c(-1, 1), color = "grey", linetype = "dashed") +
  geom_text_repel(data = with_insc_top, aes(label = Gene), size = 3) +
  theme(
    text = element_text(family = "Arial", size = 12),
    legend.position = "top",
    legend.title = element_blank()
  )

print(p)
```



By statistical significance

Hide

```r
# Ensure insc is always labeled and drawn
insc_row <- subset(results, Gene == "insc")

# Filter the top 16 con_incs for tagging
with_insc_top <- subset(results, annotation == "expressers" & abs(logFC) > 1 & -log10(adj.p.value) > 2)
with_insc_top <- with_insc_top[order(with_insc_top$adj.p.value), ][1:16, ]

# Add 'insc' if it is not already included
if (!"insc" %in% with_insc_top$Gene) {
  with_insc_top <- rbind(with_insc_top, insc_row)
}

# Plot with visual limits but without cutting points
p <- ggplot(results, aes(x = logFC, y = -log10(adj.p.value), color = annotation)) +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("non_expressers" = "#8998C8", "expressers" = "red", "Not significant" = "black")) +
  theme_minimal() +
  labs(
    title = "Volcano Plot of Differential Gene Expression",
    subtitle = "Data source: Ovary, young germ cell",
    x = "Log2 Fold Change",
    y = "-Log10 Adjusted P-Value",
    caption = "Analysis method: Welch's t-test"
  ) +
  geom_hline(yintercept = -log10(0.05), color = "grey", linetype = "dashed") +
  geom_vline(xintercept = c(-1, 1), color = "grey", linetype = "dashed") +
  geom_text_repel(data = with_insc_top, aes(label = Gene), size = 3, max.overlaps = Inf) +
  coord_cartesian(xlim = c(-4.5, 10), ylim = c(0, 19.5)) +
  theme(
    text = element_text(family = "Arial", size = 12),
    legend.position = "top",
    legend.title = element_blank()
  )

print(p)
```



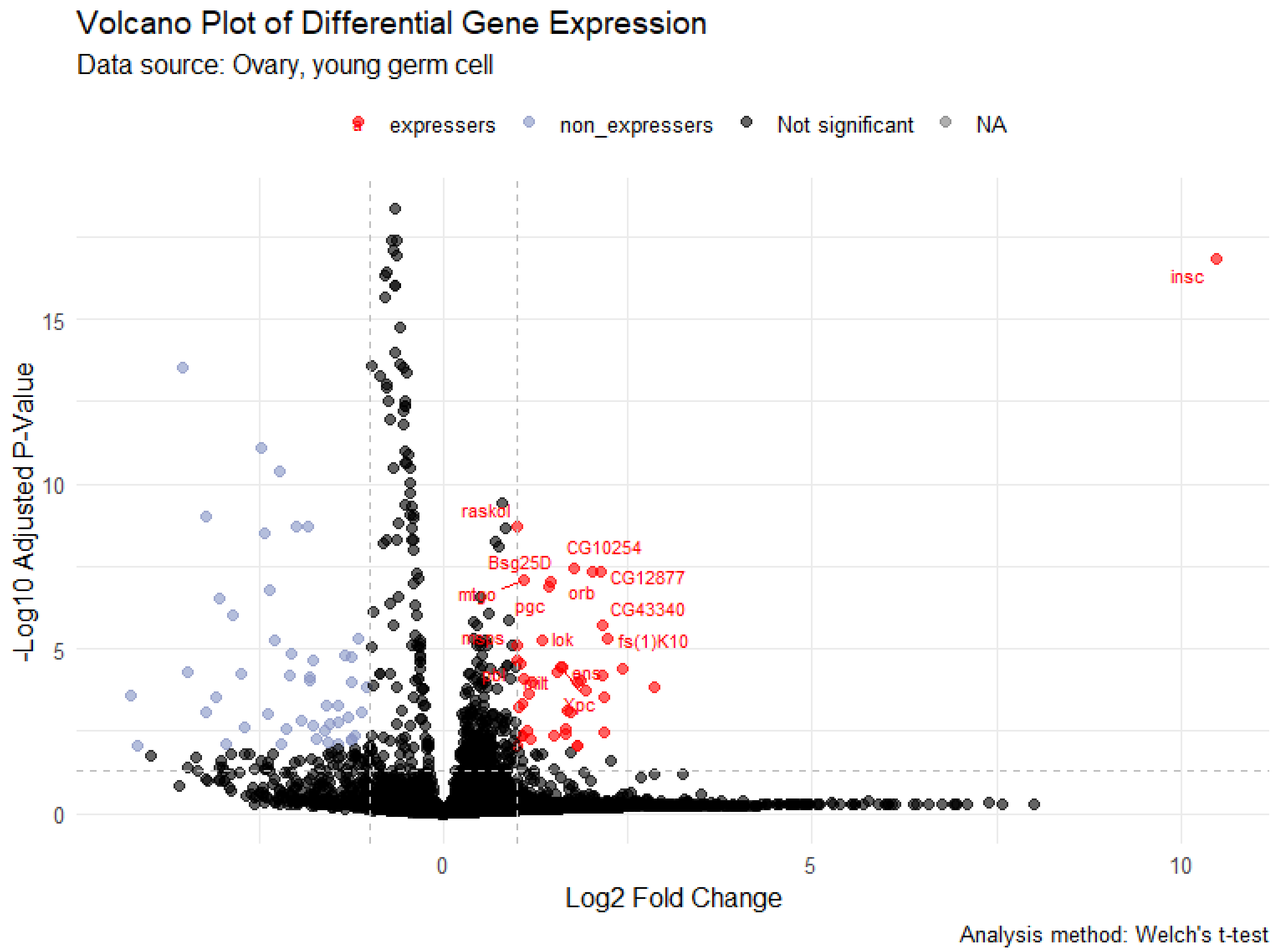
Hide

```
NA
NA
```

Final Volcano plot with filtering

Hide

```r
# Define visual boundaries
x_lim <- c(-4.5, 11)
y_lim <- c(0, 20)

# 1. Filter all genes within the range of the graph.
results_plot <- results[
  results$logFC >= x_lim[1] & results$logFC <= x_lim[2] &
  -log10(results$adj.p.value) >= y_lim[1] & -log10(results$adj.p.value) <= y_lim[2],
]

# 2. Filter out only genes annotated as “expressers”.
with_insc_genes <- results_plot[results_plot$annotation == "expressers", ]

# 3. Select the 16 most significant genes by adj.p.value.
top16_with_insc <- with_insc_genes[order(with_insc_genes$adj.p.value), ][1:16, ]

# Volcano Plot
p <- ggplot(results_plot, aes(x = logFC, y = -log10(adj.p.value), color = annotation)) +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("non_expressers" = "#8998C8", "expressers" = "red", "Not significant" = "black")) +
  theme_minimal() +
  labs(
    title = "Volcano Plot of Differential Gene Expression",
    subtitle = "Data source: Ovary, young germ cell",
    x = "Log2 Fold Change",
    y = "-Log10 Adjusted P-Value",
    caption = "Analysis method: Welch's t-test"
  ) +
  geom_hline(yintercept = -log10(0.05), color = "grey", linetype = "dashed") +
  geom_vline(xintercept = c(-1, 1), color = "grey", linetype = "dashed") +
  geom_text_repel(
    data = top16_with_insc,
    aes(label = Gene),
    size = 3,
    max.overlaps = Inf
  ) +
  theme(
    text = element_text(family = "Arial", size = 12),
    legend.position = "top",
    legend.title = element_blank()
  )

print(p)
```



#extraction in excel fiile

Hide

```
library(openxlsx)

# Crear un nuevo archivo Excel
wb <- createWorkbook()

# Añadir pestañas y escribir datos
blue_data <- results[results$annotation == "non_expressers",]
blue_data <- blue_data[order(row.names(blue_data)),]
addWorksheet(wb, "blue")
writeData(wb, "blue", blue_data)

red_data <- results[results$annotation == "expressers",]
red_data <- red_data[order(row.names(red_data)),]
addWorksheet(wb, "red")
writeData(wb, "red", red_data)

black_data <- results[results$annotation == "Not significant",]
black_data <- black_data[order(row.names(black_data)),]
addWorksheet(wb, "black")
writeData(wb, "black", black_data)

# Guardar el archivo Excel
saveWorkbook(wb, "results-insc-poster.xlsx", overwrite = TRUE)
```

Final Part

the second, make a list of cells expressing the 3 genes (“c(3)G”, “c(2)M”, “mu2”) and extract their genes.

#verification

Hide

```
# Definir los genes de interés y la anotación
gene_of_interest <- "insc"
genes_of_interest <- c("mu2", "c(2)M", "c(3)G")
annotation_of_interest <- "young germ cell"
expression_threshold <- 0.5

# Filtrar por anotación de interés
adata_filtered <- subset(adata, subset = annotation == annotation_of_interest)

# Identificar células que expresan el gen de interés (insc) > 0.5
insc_expression <- adata_filtered@assays$RNA@data[gene_of_interest, ]
annotation_cells <- colnames(adata_filtered)
insc_positive <- annotation_cells[insc_expression > expression_threshold]

# Identificar las células que expresan los tres genes de interés por encima del umbral
gene_expressions <- adata_filtered@assays$RNA@data[genes_of_interest, ]
cells_expressing_all_genes <- colnames(adata_filtered)[apply(gene_expressions > expression_threshold, 2, all)]

# Encontrar la intersección de estos dos conjuntos de células
overlapping_cells <- intersect(insc_positive, cells_expressing_all_genes)

# Imprimir el resultado
cat("Number of cells expressing 'insc':", length(insc_positive), "\n")
```

```
Number of cells expressing 'insc': 167
```

Hide

```
cat("Number of cells expressing 'mu2', 'c(2)M' y 'c(3)G':", length(cells_expressing_all_genes), "\n")
```

```
Number of cells expressing 'mu2', 'c(2)M' y 'c(3)G': 55
```

Hide

```
cat("Number of cells expressing ambos:", length(overlapping_cells), "\n")
```

```
Number of cells expressing ambos: 31
```

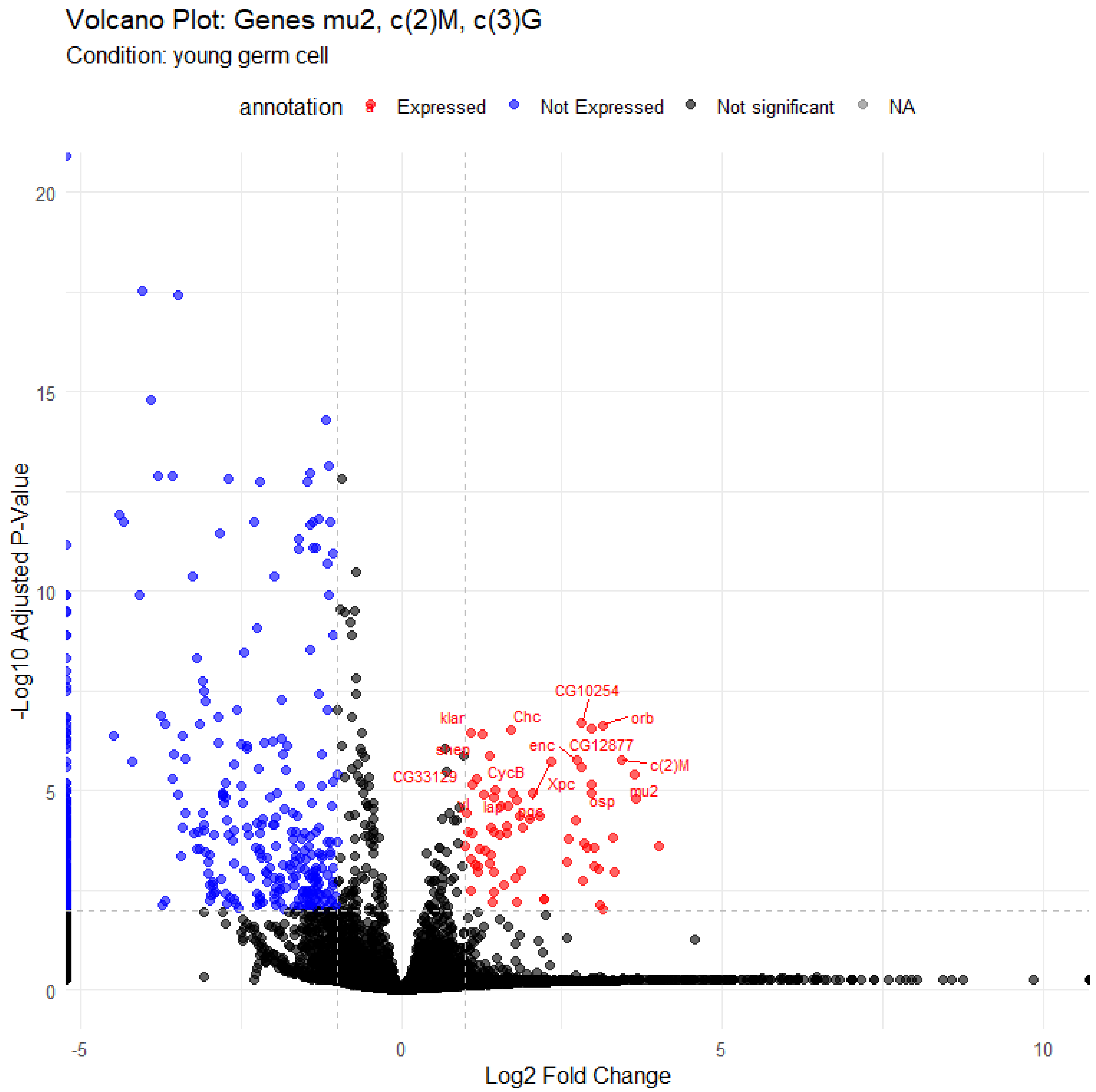
#2nd list (“mu2”, “c(2)M”, “c(3)G”)

Hide

```
# --- PART ("mu2", "c(2)M", "c(3)G") ---
# Filter out cells expressing ALL genes of interest
genes_of_interest <- c("mu2", "c(2)M", "c(3)G")
annotation_of_interest <- "young germ cell"
expression_threshold <- 0.5

adata_filtered_3g <- subset(adata, subset = annotation == annotation_of_interest)

cells_expressing_all_genes <- colnames(adata_filtered_3g)[
  apply(adata_filtered_3g@assays$RNA@data[genes_of_interest, ] > expression_threshold, 2, all)
]

# Identify cells that do not express all genes of interest above the threshold.
cells_not_expressing_all_genes <- colnames(adata_filtered_3g)[
  apply(adata_filtered_3g@assays$RNA@data[genes_of_interest, ] <= expression_threshold, 2, any)
]

# Create subsets of data for the two cell populations.
adata_expressing_3genes <- subset(adata_filtered_3g, cells = cells_expressing_all_genes)
adata_not_expressing_3genes <- subset(adata_filtered_3g, cells = cells_not_expressing_all_genes)

# Verificar si hay células repetidas entre los dos subconjuntos
stopifnot(length(intersect(cells_expressing_all_genes, cells_not_expressing_all_genes)) == 0)

# Calculate fold change, p-values and expression averages
all_genes <- rownames(adata@assays$RNA@data)
results_3genes <- data.frame(Gene = all_genes, logFC = NA, p.value = NA,
                             mean_expressers = NA, mean_non_expressers = NA)
for (gene in all_genes) {
  g1 <- adata_expressing_3genes@assays$RNA@data[gene, ]
  g2 <- adata_not_expressing_3genes@assays$RNA@data[gene, ]
  ttest <- t.test(g1, g2)
  logFC <- log2((mean(g1)) / (mean(g2)))
  results_3genes[results_3genes$Gene == gene, ] <- list(
    gene, logFC, ttest$p.value, mean(g1), mean(g2)
  )
}

# p-value adjustment and annotation
results_3genes$adj.p.value <- p.adjust(results_3genes$p.value, method = "BH")
results_3genes$annotation <- ifelse(abs(results_3genes$logFC) > 1 & results_3genes$adj.p.value < 0.01,
                                    ifelse(results_3genes$logFC > 0, "Expressed", "Not Expressed"),
                                    "Not significant")
```

Hide

```r
# 1. Filtrar los genes significativos expresados
top_genes <- subset(results_3genes, annotation == "Expressed" & abs(logFC) > 1 & -log10(adj.p.value) > 2)

# 2. Ordenar por adj.p.value (más significativos primero)
top_genes <- top_genes[order(top_genes$adj.p.value), ][1:16, ]  # Top 16

# 3. Volcano Plot con etiquetas
p3g <- ggplot(results_3genes, aes(x = logFC, y = -log10(adj.p.value), color = annotation)) +
  geom_point(alpha = 0.6, size = 2) +
  geom_text_repel(
    data = top_genes,
    aes(label = Gene),
    size = 3,
    max.overlaps = Inf
  ) +
  scale_color_manual(values = c("Expressed" = "red", "Not Expressed" = "blue", "Not significant" = "blac
k")) +
  labs(
    title = "Volcano Plot: Genes mu2, c(2)M, c(3)G",
    subtitle = "Condition: young germ cell",
    x = "Log2 Fold Change",
    y = "-Log10 Adjusted P-Value"
  ) +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "gray") +
  geom_hline(yintercept = -log10(0.01), linetype = "dashed", color = "gray") +
  coord_cartesian(xlim = c(-4.5, 10), ylim = c(0, 20)) +
  theme_minimal() +
  theme(text = element_text(size = 12), legend.position = "top")

print(p3g)
```



Hide

```
NA
NA
```

Hide

```
# Create a new Excel file
wb <- createWorkbook()

# Add tabs and write data
blue_data <- results[results$annotation == "non_expressers",]
blue_data <- blue_data[order(row.names(blue_data)),]
addWorksheet(wb, "blue")
writeData(wb, "blue", blue_data)

red_data <- results[results$annotation == "expressers",]
red_data <- red_data[order(row.names(red_data)),]
addWorksheet(wb, "red")
writeData(wb, "red", red_data)

black_data <- results[results$annotation == "Not significant",]
black_data <- black_data[order(row.names(black_data)),]
addWorksheet(wb, "black")
writeData(wb, "black", black_data)

# Save the Excel file
saveWorkbook(wb, "results-3-genes-poster.xlsx", overwrite = TRUE)
```

Hide

```
sessionInfo()
```

```
R version 4.4.3 (2025-02-28 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 26100)

Matrix products: default


locale:
[1] LC_COLLATE=English_United States.utf8  LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8 LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

time zone: America/Chicago
tzcode source: internal

attached base packages:
[1] grid      stats4    stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] ggvenn_0.1.10               dplyr_1.1.4                 openxlsx_4.2.8              VennDiagram_1.7.3
 [5] futile.logger_1.4.3         DESeq2_1.46.0               SummarizedExperiment_1.36.0 Biobase_2.66.0
 [9] MatrixGenerics_1.18.1       matrixStats_1.5.0           GenomicRanges_1.58.0        GenomeInfoDb_1.42.
3
[13] IRanges_2.40.1              S4Vectors_0.44.0            BiocGenerics_0.52.0         SeuratDisk_0.0.0.9
021
[17] ggrepel_0.9.6               ggplot2_3.5.1               Seurat_5.2.1                SeuratObject_5.0.2
[21] sp_2.2-0

loaded via a namespace (and not attached):
  [1] RColorBrewer_1.1-3      rstudioapi_0.17.1       jsonlite_1.9.1          magrittr_2.0.3
  [5] spatstat.utils_3.1-3    farver_2.1.2            rmarkdown_2.29          zlibbioc_1.52.0
  [9] vctrs_0.6.5             ROCR_1.0-11             spatstat.explore_3.4-2  tinytex_0.56
 [13] S4Arrays_1.6.0          htmltools_0.5.8.1       lambda.r_1.2.4          SparseArray_1.6.2
 [17] sass_0.4.9              sctransform_0.4.1       parallelly_1.43.0       bslib_0.9.0
 [21] KernSmooth_2.23-26      htmlwidgets_1.6.4       ica_1.0-3               plyr_1.8.9
 [25] futile.options_1.0.1    cachem_1.1.0            plotly_4.10.4           zoo_1.8-13
 [29] igraph_2.1.4            mime_0.13               lifecycle_1.0.4         pkgconfig_2.0.3
 [33] Matrix_1.7-2            R6_2.6.1                fastmap_1.2.0           GenomeInfoDbData_1.2.13
 [37] fitdistrplus_1.2-2      future_1.34.0           shiny_1.10.0            digest_0.6.37
 [41] colorspace_2.1-1        patchwork_1.3.0         tensor_1.5              RSpectra_0.16-2
 [45] irlba_2.3.5.1           labeling_0.4.3          progressr_0.15.1        spatstat.sparse_3.1-0
 [49] httr_1.4.7              polyclip_1.10-7         abind_1.4-8             compiler_4.4.3
 [53] bit64_4.6.0-1           withr_3.0.2             BiocParallel_1.40.0     fastDummies_1.7.5
 [57] MASS_7.3-64             DelayedArray_0.32.0     tools_4.4.3             lmtest_0.9-40
 [61] zip_2.3.2               httpuv_1.6.15           future.apply_1.11.3     goftest_1.2-3
 [65] glue_1.8.0              nlme_3.1-167            promises_1.3.2          rsconnect_1.3.4
 [69] Rtsne_0.17              cluster_2.1.8           reshape2_1.4.4          generics_0.1.3
 [73] hdf5r_1.3.12            gtable_0.3.6            spatstat.data_3.1-6     tidyr_1.3.1
 [77] data.table_1.17.0       XVector_0.46.0          spatstat.geom_3.3-6     RcppAnnoy_0.0.22
 [81] RANN_2.6.2              pillar_1.10.2           stringr_1.5.1           spam_2.11-1
 [85] RcppHNSW_0.6.0          later_1.4.1             splines_4.4.3           lattice_0.22-6
 [89] survival_3.8-3          bit_4.6.0               deldir_2.0-4            tidyselect_1.2.1
 [93] locfit_1.5-9.12         miniUI_0.1.1.1          pbapply_1.7-2           knitr_1.50
 [97] gridExtra_2.3           scattermore_1.2         xfun_0.51               stringi_1.8.7
[101] UCSC.utils_1.2.0        lazyeval_0.2.2          yaml_2.3.10             evaluate_1.0.3
[105] codetools_0.2-20        tibble_3.2.1            cli_3.6.4               uwot_0.2.3
[109] xtable_1.8-4            reticulate_1.42.0       jquerylib_0.1.4         munsell_0.5.1
[113] Rcpp_1.0.14             globals_0.16.3          spatstat.random_3.3-3   png_0.1-8
[117] spatstat.univar_3.1-2   parallel_4.4.3          dotCall64_1.2           listenv_0.9.1
[121] viridisLite_0.4.2       scales_1.3.0            ggridges_0.5.6          purrr_1.0.4
[125] crayon_1.5.3            rlang_1.1.5             formatR_1.14            cowplot_1.1.3
```