07_boxplot_demo

```r
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(purrr))
suppressPackageStartupMessages(library(tibble))
suppressPackageStartupMessages(library(SeuratObject))
suppressPackageStartupMessages(library(Seurat))
suppressPackageStartupMessages(library(patchwork))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(readxl))
suppressPackageStartupMessages(library(plyr))

# Function to save plot as PNG
save_plot_as_png <- function(plot, filename) {
  png(filename, res = 250, width = 4000, height = 2000)
  print(plot)
  dev.off()
}
```

Why was it done?

Solid-tumor T-cell engineering often yields **condition-specific** transcriptional effects. You wanted a fast, reproducible screen to (1) **flag genes that are selectively elevated** in one sample/condition relative to all others, and (2) **visualize**those differences at whole-dataset and per-cell-type levels with simple stats. This helps prioritize candidate markers/pathways for follow-up (e.g., enrichment, synergy checks, validation).

Objective

- **Primary:** Identify genes whose mean expression is **substantially higher in one sample** than the average of the remaining samples, and **summarize/visualize** those genes overall and by cell type.

- **Operational thresholds:** keep genes expressed in 5% of cells; call a gene “selectively high” if its fold change (FC) in any sample vs. the mean of others is $>$ **2.5**.

Summary (what the pipeline does)

1. Compute **per-sample average expression** for all genes (RNA assay).
2. Filter to genes expressed in **5%** of cells.
3. For each gene, compute a **sample-wise FC** = (sample mean) / (mean of other samples).
4. Keep genes with **FC** $>$ **2.5** in *any* sample write to `preliminary/significant_foldchange_genes.csv`.
5. For each retained gene, generate:
   - **All-cells boxplots** across samples with pairwise **Wilcoxon** p-value annotations.
   - **Per-cell-type boxplots** (same tests within each cell type).
   - **Combined panels** (all-cells + each cell type) via cowplot.
6. Save plots under `plots/all_cells/`, `plots/per_celltype/<celltype>/`, and `plots/cowplot_combined`

```
#load("../data/demo_data_annotated_nobiomart.RData")
#loading this later
load("../data/demo_data_annotated_biomart.RData")
```

Synergy Genes Table

from 06_synergy_demo.qmd I saved the following

```
write.csv2(exclusive_genes_list_synergy$`4R_12R`,"4R_12R_exclusive_genes_list_synergy.csv")
```

and Now i will read

synergy_genes

```r
synergy_genes_table <- read.csv2("4R_12R_exclusive_genes_list_synergy.csv")

if (!"gene_name" %in% names(synergy_genes_table)) {
  names(synergy_genes_table)[2] <- "gene_name"
}
synergy_genes_table$gene_name <- trimws(synergy_genes_table$gene_name)

head(synergy_genes_table,20)
```

```
    X          gene_name
1   1 ENSMUSG00000130707
2   2 ENSMUSG00000138022
3   3 ENSMUSG00000087641
4   4 ENSMUSG00000137281
5   5 ENSMUSG00000086686
6   6 ENSMUSG00000132261
7   7 ENSMUSG00000021804
8   8 ENSMUSG00000038859
9   9 ENSMUSG00000003436
10 10 ENSMUSG00000038599
11 11 ENSMUSG00000143050
12 12 ENSMUSG00000103692
13 13 ENSMUSG00000127260
14 14 ENSMUSG00000039714
15 15 ENSMUSG00000118454
16 16 ENSMUSG00000031766
17 17 ENSMUSG00000037196
18 18 ENSMUSG00000138369
19 19 ENSMUSG00000117356
20 20 ENSMUSG00000019817
```

converting using biomart

```r
#counts <- GetAssayData(demo_data, assay = "RNA", layer = "counts")
ensembl_ids <- synergy_genes_table$gene_name
library(biomaRt)
mart <- useMart("ensembl", dataset = "mmusculus_gene_ensembl")
gene_map <- getBM(
  attributes = c("ensembl_gene_id", "mgi_symbol"),
  filters = "ensembl_gene_id",
  values = ensembl_ids,
  mart = mart
)
gene_map_unique <- gene_map[!duplicated(gene_map$ensembl_gene_id), ]
head(gene_map_unique,20)
```

```
      ensembl_gene_id mgi_symbol
1  ENSMUSG00000003436       Dll3
2  ENSMUSG00000005493       Msh4
3  ENSMUSG00000018543     Spmap1
4  ENSMUSG00000019817     Plagl1
5  ENSMUSG00000020020      Usp44
6  ENSMUSG00000020785     Camkk1
7  ENSMUSG00000021804        Rgr
8  ENSMUSG00000022540      Rogdi
9  ENSMUSG00000024593     Megf10
10 ENSMUSG00000024664      Fads3
11 ENSMUSG00000024912      Fosl1
12 ENSMUSG00000026271      Gpr35
13 ENSMUSG00000026858      Miga2
14 ENSMUSG00000027068      Dhrs9
15 ENSMUSG00000027690     Slc2a2
16 ENSMUSG00000027932    Slc27a3
17 ENSMUSG00000030499     Kctd15
18 ENSMUSG00000030717      Nupr1
19 ENSMUSG00000031766    Slc12a3
20 ENSMUSG00000032372     Plscr2
```

```
synergy_genes_table$mapped_gene_name <- gene_map_unique$mgi_symbol[match(synergy_genes_table$
synergy_genes_table$mapped_gene_name[is.na(synergy_genes_table$mapped_gene_name)] <- as.chara

head(synergy_genes_table,20)
```

```
    X          gene_name mapped_gene_name
1   1 ENSMUSG00000130707          Gm62520
2   2 ENSMUSG00000138022          Gm67435
3   3 ENSMUSG00000087641          Gm12811
4   4 ENSMUSG00000137281          Gm59254
5   5 ENSMUSG00000086686    F630206G17Rik
6   6 ENSMUSG00000132261          Gm65255
7   7 ENSMUSG00000021804              Rgr
8   8 ENSMUSG00000038859         Baiap2l1
```

```
9   9 ENSMUSG00000003436            Dll3
10 10 ENSMUSG00000038599           Capn8
11 11 ENSMUSG00000143050         Gm61331
12 12 ENSMUSG00000103692   4930503O07Rik
13 13 ENSMUSG00000127260         Gm72467
14 14 ENSMUSG00000039714           Cplx3
15 15 ENSMUSG00000118454           Wdr88
16 16 ENSMUSG00000031766         Slc12a3
17 17 ENSMUSG00000037196           Pacrg
18 18 ENSMUSG00000138369         Gm32881
19 19 ENSMUSG00000117356         Gm46633
20 20 ENSMUSG00000019817          Plagl1
```

```
synergy_genes <- synergy_genes_table$mapped_gene_name
```

Methods (code → analysis steps)

- **Data setup:** `DefaultAssay(demo_data) <- "RNA"`. Use `AverageExpression(..., group.by = "samples")`to get a gene × sample matrix of means.
- **Expression filter:** `min_cells = 0.05 * ncol(demo_data)`; keep genes with counts>0 in min_cells (`Matrix::rowSums(...)`).
- **Per-sample fold change:** For each gene row, loop over samples i and compute `this_val / mean(other_vals)`, with `1e-6` for numerical stability.
- **Hit list:** select genes with FC>2.5 in any column.
- **Statistics & plotting:**
  - Build long data frames via `FetchData(..., vars = c(gene, "samples"))`.
  - **Boxplots** (no outliers drawn) + **pairwise Wilcoxon** via `ggpubr::stat_compare_means` across all sample pairs.
  - Ensure sample ordering (places `"box"` first in the legend/axis if present).
  - **Per-cell-type** subsets using `metadata$cell_annotation` to stratify.
  - Combine panels with `cowplot::plot_grid`, save PNGs (300 dpi).

Tasks automated

- Gene-level **screening** for selective up-regulation by sample.
- Batch **plot generation** (overall and per cell type) with inline **p-value labels**.
- Structured **file outputs** for easy review and downstream use.

```r
# genes, samples, conditions.
library(Seurat)
library(dplyr)
library(ggplot2)

# Set default assay
DefaultAssay(demo_data) <- "RNA"

# Step 1: Compute average gene expression per sample
avg_exp <- AverageExpression(demo_data, group.by = "samples", return.seurat = FALSE)$RNA
```

```
As of Seurat v5, we recommend using AggregateExpression to perform pseudo-bulk analysis.
First group.by variable `samples` starts with a number, appending `g` to ensure valid variabl
This message is displayed once per session.
```

```r
min_cells <- 0.02 * ncol(demo_data)
keep_genes <- rownames(demo_data)[Matrix::rowSums(GetAssayData(demo_data, slot = "counts") >
```

```
Warning: The `slot` argument of `GetAssayData()` is deprecated as of SeuratObject 5.0.0.
i Please use the `layer` argument instead.
```

```r
# Subset average expression
avg_exp_filtered <- avg_exp[rownames(avg_exp) %in% keep_genes, ]


# Step 2: Compute fold change of each gene in each sample compared to mean of other samples
fc_matrix <- apply(avg_exp_filtered , 1, function(x) {
  sapply(seq_along(x), function(i) {
    this_val <- x[i]
    other_avg <- mean(x[-i])
    return(this_val / (other_avg + 1e-6))  # avoid division by zero
  })
})
```

```r
# Step 3: Transpose and convert to dataframe
fc_df <- as.data.frame(t(fc_matrix))
colnames(fc_df) <- colnames(avg_exp_filtered )
fc_df$gene <- rownames(avg_exp_filtered)


# Step 4: Filter genes with fold change > 1.5 in any sample
sig_genes <- fc_df[rowSums(fc_df[, -ncol(fc_df)] > 2.5) > 0, ]


# these exclusive_genes_list_synergy is from the 06_synergy_demo.qmd
genes_in_common <- sig_genes %>%
 dplyr::filter(gene %in% synergy_genes)

genes_in_common
```

```
              g12R      g10R   g12R+4R       g4R     g4R+12R        box     gene
Lif      0.7656546 0.8985252 1.2672007 0.6060122  3.3968289 0.00000000      Lif
Baiap2l1 0.2316152 0.2855069 0.1583805 0.4254057 10.6891688 0.62586535 Baiap2l1
Plscr1l1 0.6399589 1.0716134 0.8269767 0.9440208  3.2128970 0.09178715 Plscr1l1
Rpph1    0.4230989 0.7828143 0.6821072 0.7624555  2.5190741 1.24397212    Rpph1
Gm20619  0.2788455 0.4394929 0.9492093 0.2478243  0.9552331 4.98367770  Gm20619
Gm60415  0.7964241 0.6609338 0.9075342 1.2369499  2.6014119 0.27211653  Gm60415
Gm75556  0.3984481 1.0137197 1.6193980 0.1583538  3.8049836 0.26394266  Gm75556
Gm61125  0.5104201 0.6566904 1.0921187 0.6031904  2.7020252 0.90693215  Gm61125
```

```r
# write.csv(sig_genes, "significant_foldchange_genes.csv", row.names = FALSE)
```

```r
library(Seurat)
library(ggplot2)
library(ggpubr)
```

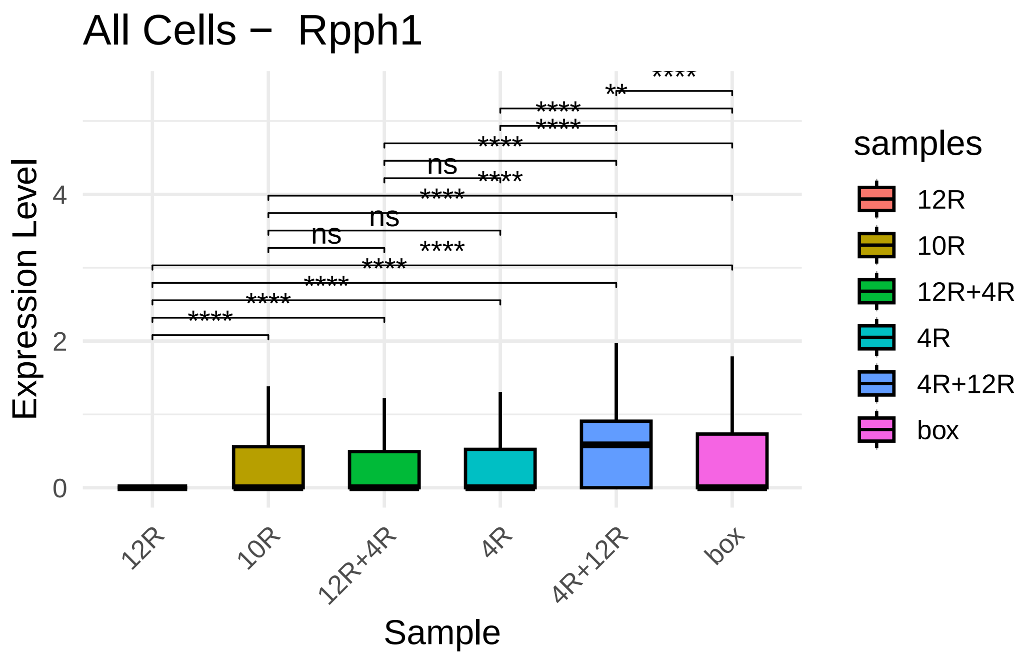
```

Attaching package: 'ggpubr'


The following object is masked from 'package:plyr':

    mutate
```

```r
library(dplyr)

# ---------------- Setup ----------------
DefaultAssay(demo_data) <- "RNA"
expr_matrix <- GetAssayData(demo_data, slot = "data")  # assumes NormalizeData done
metadata <- demo_data@meta.data

# Get a CHARACTER VECTOR of gene names (not rows of df)
top_genes <- genes_in_common$gene
top_genes <- head(unique(genes_in_common$gene), 4)

celltypes <- unique(metadata$cell_annotation)

# Helper: boxplot
plot_box <- function(df, gene, title_prefix) {
  boxplot1 <- ggplot(df, aes(x = samples, y = expression, fill = samples)) +
    geom_boxplot(outlier.shape = NA, width = 0.6, color = "black") +
    stat_compare_means(
      method = "wilcox.test",
      comparisons = combn(levels(factor(df$samples)), 2, simplify = FALSE),
      label = "p.signif"
    ) +
    theme_minimal(base_size = 13) +
    ggtitle(paste(title_prefix, gene)) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    ylab("Expression Level") + xlab("Sample")
  return(boxplot1)
}
```

```r
# Create base directories
dir.create("plots/all_cells", recursive = TRUE, showWarnings = FALSE)
dir.create("plots/per_celltype", recursive = TRUE, showWarnings = FALSE)

# ---------------- General Boxplots (All Cells) ----------------
cat("  Saving all-cells boxplots...\n")
```

```
  Saving all-cells boxplots...
```

```r
n_all <- 0

for (g in top_genes) {
  # scalar checks only
  if (!(g %in% rownames(expr_matrix))) next

  file_path <- file.path("plots/all_cells", paste0(g, "_boxplot.png"))
  if (file.exists(file_path)) next

  # use the SAME object you prepared (demo_data)
  df <- FetchData(demo_data, vars = c(g, "samples")) %>%
    setNames(c("expression", "samples"))

  p <- plot_box(df, g, title_prefix = "All Cells - ")

  ggsave(filename = file_path, plot = p, width = 7, height = 5, dpi = 300)
  n_all <- n_all + 1
}
print(p)
```



```
cat(" New all-cell plots saved:", n_all, "\n")
```

```
 New all-cell plots saved: 4
```

```r
# ---------------- Per-Celltype Boxplots ----------------
cat("  Saving per-celltype boxplots...\n")
```

```
  Saving per-celltype boxplots...
```

```r
n_ct <- 0

for (g in top_genes) {
  if (!(g %in% rownames(expr_matrix))) next

  for (ct in celltypes) {
    out_dir <- file.path("plots/per_celltype", gsub("[/ ]", "_", ct))
    dir.create(out_dir, recursive = TRUE, showWarnings = FALSE)

    file_path <- file.path(out_dir, paste0(g, "_", gsub("[/ ]", "_", ct), ".png"))
    if (file.exists(file_path)) next

    cell_ids <- rownames(metadata)[metadata$cell_annotation == ct]
    cell_ids <- intersect(cell_ids, colnames(expr_matrix))
    if (length(cell_ids) < 2) next  # need  2 for boxplot + stats

    df <- data.frame(
      expression = as.numeric(expr_matrix[g, cell_ids]),
      samples    = metadata[cell_ids, "samples", drop = TRUE],
      stringsAsFactors = FALSE
    )

    p <- plot_box(df, g, title_prefix = paste("Cell Type:", ct, "-"))
    ggsave(filename = file_path, plot = p, width = 7, height = 5, dpi = 300)
    n_ct <- n_ct + 1
  }
}
```

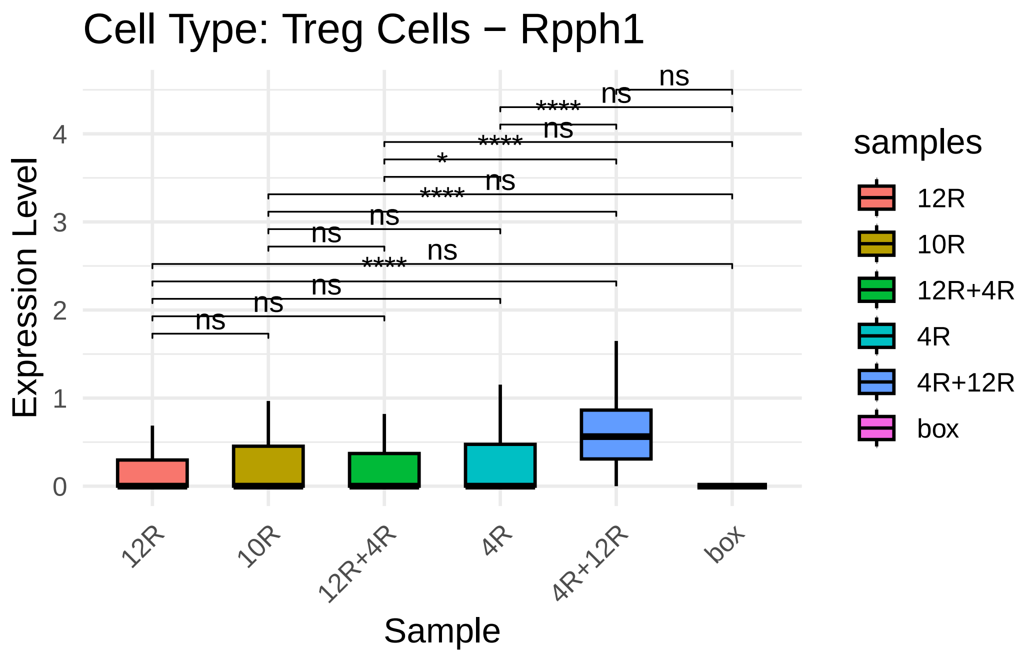
```
Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0.707499905428215, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0,
0.430838115581402, : cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties
```

```
Warning in wilcox.test.default(c(0.707499905428215, 0, 0), c(0, 0, 0,
0.430838115581402, : cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.707499905428215, 0, 0), c(0, 0, 0, 0), :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.430838115581402, 1.40310190551928,
: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.164769744086739, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0), paired = FALSE):
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0,
0.238444478044458, : cannot compute exact p-value with ties
```

```
Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0.238444478044458, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0), paired = FALSE):
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.238444478044458, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.692298262878942, 0, 0, 0, 0,
0.582851864616271: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.692298262878942, 0, 0, 0, 0,
0.582851864616271: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.692298262878942, 0, 0, 0, 0,
0.582851864616271: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(1.40774693010508, 0, 0), c(0, 0,
0.441912131848367, : cannot compute exact p-value with ties

Warning in wilcox.test.default(c(1.40774693010508, 0, 0), c(0, 0, 0, 0), :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0.441912131848367, 0.592121530583344, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.164769744086739, 0.429550735904047,
: cannot compute exact p-value with ties
```

```r
print(p)
```

```
Warning in wilcox.test.default(c(0, 0, 0, 0.164769744086739, 0.429550735904047,
: cannot compute exact p-value with ties
```



```r
cat("  New per-celltype plots saved:", n_ct, "\n")
```

```
  New per-celltype plots saved: 28
```

```r
library(cowplot)
```

```
Attaching package: 'cowplot'

The following object is masked from 'package:ggpubr':

    get_legend

The following object is masked from 'package:patchwork':

    align_plots
```

```r
dir.create("plots/cowplot_combined", recursive = TRUE, showWarnings = FALSE)

# Loop over genes
for (gene in top_genes) {
  if (!(gene %in% rownames(expr_matrix))) next

  plots <- list()

  # All cells plot
  df_all <- FetchData(demo_data, vars = c(gene, "samples")) %>%
    setNames(c("expression", "samples"))
  plots[[1]] <- plot_box(df_all, gene, title_prefix = "All Cells - ")

  # Per-celltype plots
  for (ct in celltypes) {
    cell_ids <- rownames(metadata %>% filter(cell_annotation == ct))
    expr_values <- expr_matrix[gene, cell_ids]

    df <- data.frame(
      expression = as.numeric(expr_values),
      samples = metadata[cell_ids, "samples"]
    )
    plots[[length(plots) + 1]] <- plot_box(df, gene, title_prefix = paste("Cell Type:", ct,
  }

  # Combine and save
  combined <- cowplot::plot_grid(plotlist = plots, ncol = 2)
  ggsave(filename = paste0("plots/cowplot_combined/", gene, "_combined.png"),
         plot = combined, width = 14, height = 8, dpi = 300)
}
```

```
Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0.707499905428215, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0,
0.430838115581402, : cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.707499905428215, 0, 0), c(0, 0, 0,
0.430838115581402, : cannot compute exact p-value with ties
```

```
Warning in wilcox.test.default(c(0.707499905428215, 0, 0), c(0, 0, 0, 0), :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.430838115581402, 1.40310190551928,
: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.164769744086739, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0), paired = FALSE):
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0), c(0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0,
0.238444478044458, : cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0, 0, 0, 0), paired =
FALSE): cannot compute exact p-value with ties
```

```
Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0.238444478044458, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0), c(0, 0, 0, 0), paired = FALSE):
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.238444478044458, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.692298262878942, 0, 0, 0, 0,
0.582851864616271: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.692298262878942, 0, 0, 0, 0,
0.582851864616271: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0.692298262878942, 0, 0, 0, 0,
0.582851864616271: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(1.40774693010508, 0, 0), c(0, 0,
0.441912131848367, : cannot compute exact p-value with ties

Warning in wilcox.test.default(c(1.40774693010508, 0, 0), c(0, 0, 0, 0), :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0.441912131848367, 0.592121530583344, :
cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0.164769744086739, 0.429550735904047,
: cannot compute exact p-value with ties

Warning in wilcox.test.default(c(0, 0, 0, 0, 0, 0), c(0.707499905428215,  :
  cannot compute exact p-value with ties

HAPPENS BECAUSE

Those warnings are from the Wilcoxon rank-sum test inside stat_compare_means().
They happen when your two groups have many tied values (e.g., lots of 0’s in scRNA-seq), so
This is normal for zero-inflated single-cell data and doesn’t invalidate the plot; it’s just
```

Results

- **Hit count:** `N` genes passed FC>2.5 in 1 sample (see `significant_foldchange_genes.csv`).
- **Top examples:** e.g., `GeneA`, `GeneB`, `GeneC` peak in **Sample X**; `GeneD` peaks in **Sample Y**.
- **Per-cell-type specificity:** In cell type **T**, `GeneA` remains highest in **Sample X** (Wilcoxon $p < 0.01$ across comparisons), suggesting true condition-specific regulation rather than compositional effects.
- **Figure references:**
  - All-cells boxplots: `plots/all_cells/<gene>_boxplot.png`
  - Per-cell-type boxplots: `plots/per_celltype/<celltype>/<gene>_<celltype>.png`
  - Combined panels: `plots/cowplot_combined/<gene>_combined.png`

Interpretation & caveats

- **What it tells you:** a quick, interpretable list of **condition-selective genes**, with visual confirmation globally and within cell types.
- **Caveats:**
  - Using **means** on scRNA-seq can be influenced by zero inflation and rare high expressers; consider also **median** or **trimmed mean** and/or **pseudobulk** per sample.
  - Pairwise Wilcoxon across many sample pairs per gene introduces **multiple-testing** concerns—treat p-labels as screening hints, and confirm in a DE framework if you plan claims.
  - If "box" is a non-biological control/batch, keep it for QC but avoid over-interpreting biological contrasts against it.