kirc_tcga_data_visualization

Alejandro Sanchez

2025-11-05

renv_guard

Setup and checks

```r
# ---- setup_and_checks ----
# Packages
if (!requireNamespace("here", quietly = TRUE)) install.packages("here")
suppressPackageStartupMessages({
  library(here); library(data.table)
})

# Knit defaults + Unicode sanitizer (keeps pdflatex happy)
set.seed(1234)
knitr::opts_chunk$set(message = FALSE, warning = FALSE, fig.width = 7, fig.height = 5)
knitr::knit_hooks$set(output = function(x, options) {
  repl <- list(
    "\u2265"=">=","\u2264"="<=","\u2013"="-","\u2014"="--","\u00B1"="+/-",
    "\u2018"="'","\u2019"="'","\u201C"="\"","\u201D"="\""
  )
  for (k in names(repl)) x <- gsub(k, repl[[k]], x, fixed = TRUE)
  x
})

# Choose contrast
contrast_label <- "ccrcc_vs_normal_kidney"  # edit if needed
allowed <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc",
             "hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_normal_kidney")
if (!contrast_label %in% allowed) stop("Invalid contrast_label: ", contrast_label, call. = FALSE)

# Core files from previous Rmd
bundle_path <- here::here("outputs","kirc_clean_tmm_objects.rds")
samples_tsv <- here::here("data","tcga_kirc_samples.tsv")
de_csv      <- here::here("results", sprintf("kirc_%s_limma.csv", contrast_label))
gsea_tsv    <- here::here("results", sprintf("kirc_%s_gsea_hallmark.tsv", contrast_label)) # optional

# Existence checks (fail early)
missing <- c(bundle_path, samples_tsv, de_csv)[!file.exists(c(bundle_path, samples_tsv, de_csv))]
if (length(missing)) stop("Missing required file(s):\n- ", paste(normalizePath(missing, winslash = "/")
if (!file.exists(gsea_tsv)) message("[Note] GSEA file not found (expected later): ", basename(gsea_tsv)

# Read DE now so downstream chunks can assume `de` exists
de <- tryCatch(data.table::fread(de_csv), error = function(e) stop("Failed to read DE CSV: ", de_csv, "
stopifnot(is.data.frame(de), all(c("logFC") %in% names(de)))  # minimal schema guard

# Echo for sanity
cat("[Paths]\n",
    "- contrast_label: ", contrast_label, "\n",
    "- bundle:         ", normalizePath(bundle_path, winslash = "/"), "\n",
    "- samples:        ", normalizePath(samples_tsv, winslash = "/"), "\n",
    "- DE csv:         ", normalizePath(de_csv, winslash = "/"), "\n",
    "- GSEA (opt):     ", normalizePath(gsea_tsv, winslash = "/"), "\n", sep = "")
```

[Paths]

- contrast_label: ccrcc_vs_normal_kidney

- bundle: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work/KIRC_TCGA/outputs/kirc_clean_tmm_objects.rds

- samples: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work/KIRC_TCGA/data/tcga_kirc_samples.tsv

- DE csv: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work/KIRC_TCGA/results/kirc_ccrcc_vs_normal_kidney_limma.c

- GSEA (opt): /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work/KIRC_TCGA/results/kirc_ccrcc_vs_normal_kidney_gsea_ha

```r
cat("[DE] rows:", nrow(de), " | cols:", ncol(de), "\n")
```

[DE] rows: 18278 | cols: 9

Volcano

```r
# ---- volcano_single_contrast ----
suppressPackageStartupMessages({ library(here); library(data.table); library(ggplot2) })

stopifnot(exists("de"), exists("contrast_label"))

# Pick p-value column robustly
pick_col <- function(dt, cands) { hit <- intersect(cands, names(dt)); if (length(hit)) hit[1] else NA_c
p_col <- pick_col(de, c("P.Value","pvalue","pval","PValue","p"))
if (is.na(p_col)) stop("No p-value column found in DE table.", call. = FALSE)
pval <- de[[p_col]]

# Labels: prefer gene_name (non-empty), else gene_id
lab <- if ("gene_name" %in% names(de) && any(nzchar(de$gene_name))) de$gene_name else de$gene_id

# Try EnhancedVolcano; else ggplot fallback
has_ev <- requireNamespace("EnhancedVolcano", quietly = TRUE)
if (has_ev) {
  p <- EnhancedVolcano::EnhancedVolcano(
    de,
    lab = lab,
    x = "logFC",
    y = p_col,
    pCutoff = 1e-5,
    FCcutoff = 1.5,
    drawConnectors = FALSE,
    title = sprintf("Volcano - %s", contrast_label)
  )
} else {
  thr_p  <- 0.05
  thr_fc <- log2(1.5)
  de$neglog10p <- -log10(pval)
  de$signif <- is.finite(de$neglog10p) & is.finite(de$FDR) & de$FDR < thr_p & is.finite(de$logFC) & abs
  p <- ggplot(de, aes(logFC, neglog10p)) +
    geom_point(alpha = 0.4, size = 0.7, aes(shape = signif)) +
    geom_hline(yintercept = -log10(thr_p), linetype = 2) +
    geom_vline(xintercept = c(-thr_fc, thr_fc), linetype = 2) +
    labs(title = sprintf("Volcano - %s", contrast_label),
         x = "log2 fold change", y = "-log10(p)")
}

# Output paths
dir.create(here("plots","viz"), recursive = TRUE, showWarnings = FALSE)
out_pdf <- here("plots","viz", sprintf("%s_volcano.pdf", contrast_label))
out_png <- here("plots","viz", sprintf("%s_volcano.png", contrast_label))

# Safe saves: PNG always, PDF with fallbacks
# 1) PNG
ggsave(out_png, plot = p, width = 7, height = 6, dpi = 300)

# 2) PDF - Cairo → base pdf() → SVG fallback
pdf_ok <- FALSE; alt_svg <- NA_character_
try({
  ggsave(out_pdf, plot = p, width = 7, height = 6, device = cairo_pdf)
  pdf_ok <- file.exists(out_pdf) && file.info(out_pdf)$size > 0
}, silent = TRUE)
if (!pdf_ok) try({
  grDevices::pdf(out_pdf, width = 7, height = 6, useDingbats = FALSE)
  print(p); grDevices::dev.off()
  pdf_ok <- file.exists(out_pdf) && file.info(out_pdf)$size > 0
}, silent = TRUE)
if (!pdf_ok && requireNamespace("svglite", quietly = TRUE)) {
  alt_svg <- sub("\\.pdf$", ".svg", out_pdf)
  svglite::svglite(alt_svg, width = 7, height = 6)
  print(p); grDevices::dev.off()
}

# Verify sizes
fi_pdf <- if (file.exists(out_pdf)) file.info(out_pdf)$size else 0
fi_png <- if (file.exists(out_png)) file.info(out_png)$size else 0
```

```r
cat("[Volcano] PNG:", normalizePath(out_png, winslash="/"), " size:", fi_png, "bytes\n")
```

[Volcano] PNG: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work/KIRC_TCGA/plots/viz/ccrcc_vs_normal_kidney_volcano.pn size: 696083 bytes

```r
if (pdf_ok) {
  cat("[Volcano] PDF:", normalizePath(out_pdf, winslash="/"), " size:", fi_pdf, "bytes\n")
} else if (!is.na(alt_svg)) {
  cat("[Volcano] PDF failed; wrote SVG:", normalizePath(alt_svg, winslash="/"), " size:",
      if (file.exists(alt_svg)) file.info(alt_svg)$size else 0, "bytes\n")
}
```

[Volcano] PDF: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work/KIRC_TCGA/plots/viz/ccrcc_vs_normal_kidney_volcano.pd size: 894507 bytes

```r
stopifnot(fi_png > 0)
```

All 4 volcano plots at once

```r
# ---- volcano_all_four ----
suppressPackageStartupMessages({
  library(here); library(data.table); library(ggplot2)
  library(SummarizedExperiment)
})

# Load bundle + aligned metadata
B <- readRDS(here("outputs","kirc_clean_tmm_objects.rds"))
se_filt <- B$se_clean_filtered
assn    <- B$assay_name
stopifnot(inherits(se_filt,"SummarizedExperiment"), is.character(assn))
cts  <- SummarizedExperiment::assay(se_filt, assn)
smeta <- as.data.frame(B$sample_meta); rownames(smeta) <- colnames(cts)
if (!"condition" %in% names(smeta)) {
  smeta$condition <- factor(ifelse(smeta$sample_type=="Solid Tissue Normal","Normal","Tumor"),
                            levels = c("Normal","Tumor"))
}
# Grade parsing
pick_first <- function(df, xs){ for (k in xs) if (k %in% names(df)) return(df[[k]]); NULL }
raw_grade <- toupper(as.character(pick_first(smeta, c("tumor_grade","neoplasm_histologic_grade","grade"
parse_grade <- function(x){ x <- gsub("\\s+","",x); ifelse(grepl("G3|G4|HIGH",x),"HG", ifelse(grepl("G1
smeta$grade_group <- factor(ifelse(smeta$condition=="Tumor", parse_grade(raw_grade), NA), levels=c("LG"

# Helpers
pick_col <- function(dt, cands){ hit <- intersect(cands, names(dt)); if (length(hit)) hit[1] else NA_ch
contrast_groups <- function(label) {
  switch(label,
    "ccrcc_vs_normal_kidney"    = { sub <- smeta[smeta$condition %in% c("Normal","Tumor"),];
                                    c(Normal = sum(sub$condition=="Normal"), Tumor = sum(sub$condition=
    "hg_vs_lg_ccrcc"            = { sub <- smeta[smeta$condition=="Tumor" & !is.na(smeta$grade_group),]
                                    c(LG = sum(sub$grade_group=="LG"), HG = sum(sub$grade_group=="HG"))
    "hg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), HG = sum(sub$grade_group==
    "lg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), LG = sum(sub$grade_group==
    stop(sprintf("Unknown contrast label: %s", label))
  )
}
build_volcano <- function(de_dt, label, title_suffix) {
  has_ev <- requireNamespace("EnhancedVolcano", quietly = TRUE)
  p_col <- pick_col(de_dt, c("P.Value","pvalue","pval","PValue","p")); stopifnot(!is.na(p_col))
  lab <- if ("gene_name" %in% names(de_dt) && any(nzchar(de_dt$gene_name))) de_dt$gene_name else de_dt$
  if (has_ev) {
    EnhancedVolcano::EnhancedVolcano(
      de_dt, lab = lab, x = "logFC", y = p_col,
      pCutoff = 1e-5, FCcutoff = 1.5, drawConnectors = FALSE,
      title = paste0("Volcano - ", label, " ", title_suffix)
    )
  } else {
    pval <- de_dt[[p_col]]
    de_dt$neglog10p <- -log10(pval)
    thr_p  <- 0.05; thr_fc <- log2(1.5)
    de_dt$signif <- is.finite(de_dt$neglog10p) & is.finite(de_dt$FDR) & de_dt$FDR < thr_p &
                    is.finite(de_dt$logFC) & abs(de_dt$logFC) >= thr_fc
    ggplot(de_dt, aes(logFC, neglog10p)) +
      geom_point(alpha = 0.4, size = 0.7, aes(shape = signif)) +
      geom_hline(yintercept = -log10(thr_p), linetype = 2) +
      geom_vline(xintercept = c(-thr_fc, thr_fc), linetype = 2) +
      labs(title = paste0("Volcano - ", label, " ", title_suffix),
           x = "log2 fold change", y = "-log10(p)")
  }
}
safe_save_pdf <- function(path_pdf, plot, width=7, height=6){
  ok <- FALSE
  # Cairo
  try({ ggsave(path_pdf, plot = plot, width = width, height = height, device = cairo_pdf)
      ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0 }, silent = TRUE)
  # base pdf
  if (!ok) try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    print(plot); grDevices::dev.off()
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  # SVG fallback
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height)
    print(plot); grDevices::dev.off()
    message("[Volcano] PDF failed; wrote SVG: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
```

```r
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}

# Run
dir.create(here("plots","viz"), recursive = TRUE, showWarnings = FALSE)
labels <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc","hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_normal_ki

for (lab in labels) {
  de_csv <- here("results", sprintf("kirc_%s_limma.csv", lab))
  if (!file.exists(de_csv)) { message("[Skip] Missing DE: ", basename(de_csv)); next }
  de_dt <- data.table::fread(de_csv)

  # title with group sizes
  counts <- contrast_groups(lab)
  suffix <- sprintf("(%s)", paste(paste0(names(counts), " n=", as.integer(counts)), collapse = ", "))

  p <- build_volcano(de_dt, lab, suffix)

  out_png <- here("plots","viz", sprintf("%s_volcano.png", lab))
  out_pdf <- here("plots","viz", sprintf("%s_volcano.pdf", lab))

  ggsave(out_png, plot = p, width = 7, height = 6, dpi = 300)
  pdf_res <- safe_save_pdf(out_pdf, p, width = 7, height = 6)

  ok_png <- file.exists(out_png) && file.info(out_png)$size > 0
  ok_pdf <- file.exists(out_pdf) && file.info(out_pdf)$size > 0
  cat(sprintf("[Volcano %s] PNG=%s | PDF=%s\n",
              lab, if (ok_png) "ok" else "missing",
              if (ok_pdf || (!isTRUE(pdf_res$pdf_ok))) "ok/alt" else "failed"))
  stopifnot(ok_png)
}
```

[Volcano ccrcc_vs_normal_kidney] PNG=ok | PDF=ok/alt

[Volcano hg_vs_lg_ccrcc] PNG=ok | PDF=ok/alt

[Volcano hg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok/alt

[Volcano lg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok/alt

PCA All

```r
# ---- viz_pca_all ----
suppressPackageStartupMessages({
  library(here); library(data.table); library(ggplot2)
  library(SummarizedExperiment); library(edgeR); library(matrixStats)
})

# Load bundle and aligned meta
B <- readRDS(here("outputs","kirc_clean_tmm_objects.rds"))
se_filt <- B$se_clean_filtered; assn <- B$assay_name
```

```
stopifnot(inherits(se_filt,"SummarizedExperiment"), is.character(assn))
cts  <- SummarizedExperiment::assay(se_filt, assn)

smeta <- as.data.frame(B$sample_meta); rownames(smeta) <- colnames(cts)
if (!"condition" %in% names(smeta)) {
  smeta$condition <- factor(ifelse(smeta$sample_type=="Solid Tissue Normal","Normal","Tumor"),
                            levels = c("Normal","Tumor"))
}
# Grade parsing → LG/HG
pick_first <- function(df, xs){ for (k in xs) if (k %in% names(df)) return(df[[k]]); NULL }
raw_grade <- toupper(as.character(pick_first(smeta, c("tumor_grade","neoplasm_histologic_grade","grade"
parse_grade <- function(x){ x <- gsub("\\s+","",x); ifelse(grepl("G3|G4|HIGH",x),"HG", ifelse(grepl("G1
smeta$grade_group <- factor(ifelse(smeta$condition=="Tumor", parse_grade(raw_grade), NA), levels=c("LG"

# TMM logCPM
dge <- edgeR::DGEList(counts = cts); dge <- edgeR::calcNormFactors(dge, method = "TMM")
logCPM_all <- edgeR::cpm(dge, log = TRUE, prior.count = 1)

# Helpers
contrast_keep_ids <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = rownames(smeta)[ smeta$condition %in% c("Normal","Tumor") ],
    "hg_vs_lg_ccrcc"            = rownames(smeta)[ smeta$condition=="Tumor" & !is.na(smeta$grade_group)
    "hg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tumo
    "lg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tumo
    stop(sprintf("Unknown contrast: %s", label))
  )
}
contrast_groups <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = { sub <- smeta[smeta$condition %in% c("Normal","Tumor"),];
                                    c(Normal = sum(sub$condition=="Normal"), Tumor = sum(sub$condition=
    "hg_vs_lg_ccrcc"            = { sub <- smeta[smeta$condition=="Tumor" & !is.na(smeta$grade_group),]
                                    c(LG = sum(sub$grade_group=="LG"), HG = sum(sub$grade_group=="HG"))
    "hg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), HG = sum(sub$grade_group==
    "lg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), LG = sum(sub$grade_group==
  )
}
safe_save_pdf <- function(path_pdf, plot, width=7, height=5){
  ok <- FALSE
  # Cairo
  try({ ggsave(path_pdf, plot = plot, width = width, height = height, device = cairo_pdf)
       ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0 }, silent = TRUE)
  # base pdf
  if (!ok) try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    print(plot); grDevices::dev.off()
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  # SVG fallback
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
```

```r
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height)
    print(plot); grDevices::dev.off()
    message("[PCA] PDF failed; wrote SVG: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}

dir.create(here("plots","viz"), recursive = TRUE, showWarnings = FALSE)
labels_all <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc","hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_norma

for (lab in labels_all) {
  keep_ids <- contrast_keep_ids(lab)
  keep_ids <- intersect(keep_ids, colnames(logCPM_all))
  if (length(keep_ids) < 20) { message("[PCA] Skip ", lab, " (too few samples)"); next }

  X <- logCPM_all[, keep_ids, drop = FALSE]
  vars <- matrixStats::rowVars(X)
  take <- order(vars, decreasing = TRUE)[seq_len(min(500L, sum(is.finite(vars))))]

  pca <- prcomp(t(X[take, ]), scale. = TRUE)
  var_pct <- round(100 * (pca$sdev^2 / sum(pca$sdev^2))[1:2], 1)

  grp <- switch(
    lab,
    "hg_vs_lg_ccrcc"            = smeta[keep_ids,"grade_group", drop = TRUE],
    "hg_ccrcc_vs_normal_kidney" = factor(ifelse(smeta[keep_ids,"condition"]=="Normal","Normal","HG"), l
    "lg_ccrcc_vs_normal_kidney" = factor(ifelse(smeta[keep_ids,"condition"]=="Normal","Normal","LG"), l
    smeta[keep_ids,"condition", drop = TRUE]
  )

  df <- data.frame(PC1 = pca$x[,1], PC2 = pca$x[,2], group = grp)
  gs <- contrast_groups(lab)
  ttl <- sprintf("PCA - %s (%s) [top 500 var genes]",
                 lab, paste(paste0(names(gs), " n=", as.integer(gs)), collapse = ", "))

  p <- ggplot(df, aes(PC1, PC2, shape = group)) +
    geom_point(size = 2) +
    labs(title = ttl, x = paste0("PC1 (", var_pct[1], "%)"), y = paste0("PC2 (", var_pct[2], "%)"))

  out_png <- here("plots","viz", sprintf("%s_PCA.png", lab))
  out_pdf <- here("plots","viz", sprintf("%s_PCA.pdf", lab))
  ggsave(out_png, plot = p, width = 7, height = 5, dpi = 300)
  invisible(safe_save_pdf(out_pdf, p, width = 7, height = 5))

  ok_png <- file.exists(out_png) && file.info(out_png)$size > 0
  cat(sprintf("[PCA %s] PNG=%s | PDF=%s\n",
              lab, if (ok_png) "ok" else "missing",
              if (file.exists(out_pdf) && file.info(out_pdf)$size > 0) "ok" else "failed/SVG"))
  stopifnot(ok_png)
}
```

[PCA ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[PCA hg_vs_lg_ccrcc] PNG=ok | PDF=ok

[PCA hg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[PCA lg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

MDS (limma) for all contrasts

```r
# ---- viz_mds_all ----
suppressPackageStartupMessages({
  library(here); library(ggplot2); library(SummarizedExperiment); library(edgeR); library(limma)
})

dir.create(here("plots","viz"), recursive = TRUE, showWarnings = FALSE)

# Rebuild prereqs if missing
if (!exists("se_filt") || !exists("assn") || !exists("smeta") || !exists("logCPM_all")) {
  B <- readRDS(here("outputs","kirc_clean_tmm_objects.rds"))
  se_filt <- B$se_clean_filtered; assn <- B$assay_name
  cts <- SummarizedExperiment::assay(se_filt, assn)
  smeta <- as.data.frame(B$sample_meta); rownames(smeta) <- colnames(cts)
  if (!"condition" %in% names(smeta)) {
    smeta$condition <- factor(ifelse(smeta$sample_type=="Solid Tissue Normal","Normal","Tumor"),
                              levels = c("Normal","Tumor"))
  }
  # grade groups
  pick_first <- function(df, xs){ for (k in xs) if (k %in% names(df)) return(df[[k]]); NULL }
  rawg <- toupper(as.character(pick_first(smeta, c("tumor_grade","neoplasm_histologic_grade","grade","h
  pg <- function(x){ x <- gsub("\\s+","",x); ifelse(grepl("G3|G4|HIGH",x),"HG", ifelse(grepl("G1|G2|LOW
  smeta$grade_group <- factor(ifelse(smeta$condition=="Tumor", pg(rawg), NA), levels=c("LG","HG"))
  dge <- edgeR::DGEList(counts = cts); dge <- edgeR::calcNormFactors(dge)
  logCPM_all <- edgeR::cpm(dge, log = TRUE, prior.count = 1)
}

contrast_keep_ids <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = rownames(smeta)[ smeta$condition %in% c("Normal","Tumor") ],
    "hg_vs_lg_ccrcc"            = rownames(smeta)[ smeta$condition=="Tumor" & !is.na(smeta$grade_group)
    "hg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tum
    "lg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tum
    stop(sprintf("Unknown contrast: %s", label))
  )
}

safe_save_pdf <- function(path_pdf, plot_fun, width=7, height=5) {
  ok <- FALSE
  # base pdf
  try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    plot_fun(); grDevices::dev.off()
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height)
    plot_fun(); grDevices::dev.off()
    message("[MDS] PDF failed; wrote SVG: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}

labels_all <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc","hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_norma

for (lab in labels_all) {
  keep_ids <- contrast_keep_ids(lab)
  keep_ids <- intersect(keep_ids, colnames(logCPM_all))
  if (length(keep_ids) < 20) { message("[MDS] Skip ", lab, " (too few samples)"); next }

  X <- logCPM_all[, keep_ids, drop = FALSE]
  grp <- switch(lab,
    "hg_vs_lg_ccrcc"            = smeta[keep_ids, "grade_group", drop = TRUE],
    "hg_ccrcc_vs_normal_kidney" = factor(ifelse(smeta[keep_ids,"condition"]=="Normal","Normal","HG"), le
    "lg_ccrcc_vs_normal_kidney" = factor(ifelse(smeta[keep_ids,"condition"]=="Normal","Normal","LG"), le
    smeta[keep_ids, "condition", drop = TRUE]
  )

  # PNG
  out_png <- here("plots","viz", sprintf("%s_MDS.png", lab))
  png(out_png, width = 2000, height = 1600, res = 300)
  limma::plotMDS(X, labels = as.character(grp), top = 500, gene.selection = "pairwise")
  title(main = sprintf("MDS - %s", lab))
  dev.off()
  ok_png <- file.exists(out_png) && file.info(out_png)$size > 0

  # PDF/SVG
  out_pdf <- here("plots","viz", sprintf("%s_MDS.pdf", lab))
  pdf_res <- safe_save_pdf(out_pdf, plot_fun = function(){
    limma::plotMDS(X, labels = as.character(grp), top = 500, gene.selection = "pairwise")
    title(main = sprintf("MDS - %s", lab))
  }, width = 7, height = 5)

  cat(sprintf("[MDS %s] PNG=%s | PDF=%s\n",
              lab, if (ok_png) "ok" else "missing",
              if (isTRUE(pdf_res$pdf_ok)) "ok" else "alt/failed"))
  stopifnot(ok_png)
}
```

[MDS ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[MDS hg_vs_lg_ccrcc] PNG=ok | PDF=ok

[MDS hg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[MDS lg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

QC plots per contrast using TMM logCPM inputs

```r
# ---- qc_libsize_zero_detected ----
suppressPackageStartupMessages({
  library(here); library(data.table); library(ggplot2)
  library(SummarizedExperiment); library(edgeR)
})

# Load bundle + counts + aligned metadata
B <- readRDS(here("outputs","kirc_clean_tmm_objects.rds"))
se_filt <- B$se_clean_filtered
assn    <- B$assay_name
stopifnot(inherits(se_filt, "SummarizedExperiment"), is.character(assn))
cts <- SummarizedExperiment::assay(se_filt, assn)

smeta <- as.data.frame(B$sample_meta); rownames(smeta) <- colnames(cts)
if (!"condition" %in% names(smeta)) {
  smeta$condition <- factor(ifelse(smeta$sample_type=="Solid Tissue Normal","Normal","Tumor"),
                            levels = c("Normal","Tumor"))
}
# Grade groups
pick_first <- function(df, xs){ for (k in xs) if (k %in% names(df)) return(df[[k]]); NULL }
rawg <- toupper(as.character(pick_first(smeta, c("tumor_grade","neoplasm_histologic_grade","grade","his
pg <- function(x){ x <- gsub("\\s+","", x); ifelse(grepl("G3|G4|HIGH", x), "HG",
                                          ifelse(grepl("G1|G2|LOW", x), "LG", NA)) }
smeta$grade_group <- factor(ifelse(smeta$condition=="Tumor", pg(rawg), NA), levels = c("LG","HG"))

# Contrast selectors and group counters
contrast_keep_ids <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = rownames(smeta)[ smeta$condition %in% c("Normal","Tumor") ],
    "hg_vs_lg_ccrcc"            = rownames(smeta)[ smeta$condition=="Tumor" & !is.na(smeta$grade_group)
    "hg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tum
    "lg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tum
    stop(sprintf("Unknown contrast: %s", label))
  )
}
contrast_groups <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = { sub <- smeta[smeta$condition %in% c("Normal","Tumor"),];
                                    c(Normal = sum(sub$condition=="Normal"), Tumor = sum(sub$condition=
    "hg_vs_lg_ccrcc"            = { sub <- smeta[smeta$condition=="Tumor" & !is.na(smeta$grade_group),]
                                    c(LG = sum(sub$grade_group=="LG"), HG = sum(sub$grade_group=="HG"))
    "hg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), HG = sum(sub$grade_group==
    "lg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), LG = sum(sub$grade_group==
  )
}

# Safe PDF saver (Cairo -> base pdf -> SVG fallback)
safe_save_pdf <- function(path_pdf, plot, width=7, height=5){
  ok <- FALSE
  # Try Cairo
  try({
    ggsave(path_pdf, plot = plot, width = width, height = height, device = cairo_pdf)
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  # Base PDF
  if (!ok) try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    print(plot); grDevices::dev.off()
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  # SVG fallback
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height)
    print(plot); grDevices::dev.off()
    message("[QC] PDF failed; wrote SVG instead: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}

dir.create(here("plots","viz"), recursive = TRUE, showWarnings = FALSE)
labels_all <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc","hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_norm

for (lab in labels_all) {
  keep_ids <- contrast_keep_ids(lab)
  keep_ids <- intersect(keep_ids, colnames(cts))
  if (length(keep_ids) < 20) { message("[QC] Skip ", lab, " (too few samples)"); next }

  M <- cts[, keep_ids, drop = FALSE]
  grp <- switch(lab,
    "hg_vs_lg_ccrcc"            = smeta[keep_ids, "grade_group", drop = TRUE],
    "hg_ccrcc_vs_normal_kidney" = factor(ifelse(smeta[keep_ids,"condition"]=="Normal","Normal","HG"), l
    "lg_ccrcc_vs_normal_kidney" = factor(ifelse(smeta[keep_ids,"condition"]=="Normal","Normal","LG"), l
    smeta[keep_ids, "condition", drop = TRUE]
  )

  # Metrics
  libsize   <- colSums(M)
  zero_frac <- colMeans(M == 0)
  detected  <- colSums(M > 0)

  # Titles with sample sizes
  gs <- contrast_groups(lab)
```

```r
  suffix <- sprintf("(%s)", paste(paste0(names(gs), " n=", as.integer(gs)), collapse = ", "))

  # Plot 1: library size vs zero fraction
  qc1 <- data.frame(sample = keep_ids, group = grp,
                    libsize = libsize, zero_frac = zero_frac)
  p1 <- ggplot(qc1, aes(x = log10(libsize + 1), y = zero_frac, shape = group)) +
    geom_point(size = 2) +
    labs(title = paste0("QC - ", lab, " ", suffix, ": Library size vs Zero fraction"),
         x = "log10(library size + 1)", y = "Fraction zeros")

  # Plot 2: detected genes histogram
  qc2 <- data.frame(sample = keep_ids, group = grp, detected = detected)
  p2 <- ggplot(qc2, aes(x = detected, fill = group)) +
    geom_histogram(bins = 30, alpha = 0.6, position = "identity") +
    labs(title = paste0("QC - ", lab, " ", suffix, ": Detected genes"),
         x = "Genes with count > 0", y = "Samples")

  # Save
  out1_png <- here("plots","viz", sprintf("%s_QC_libsize_zero.png", lab))
  out1_pdf <- here("plots","viz", sprintf("%s_QC_libsize_zero.pdf", lab))
  out2_png <- here("plots","viz", sprintf("%s_QC_detected_genes.png", lab))
  out2_pdf <- here("plots","viz", sprintf("%s_QC_detected_genes.pdf", lab))

  ggsave(out1_png, plot = p1, width = 7, height = 5, dpi = 300)
  ggsave(out2_png, plot = p2, width = 7, height = 5, dpi = 300)
  invisible(safe_save_pdf(out1_pdf, p1, width = 7, height = 5))
  invisible(safe_save_pdf(out2_pdf, p2, width = 7, height = 5))

  ok1 <- file.exists(out1_png) && file.info(out1_png)$size > 0
  ok2 <- file.exists(out2_png) && file.info(out2_png)$size > 0
  cat(sprintf("[QC %s] libsize/zero PNG=%s | detected PNG=%s\n",
              lab, if (ok1) "ok" else "missing", if (ok2) "ok" else "missing"))
  stopifnot(ok1, ok2)
}
```

[QC ccrcc_vs_normal_kidney] libsize/zero PNG=ok | detected PNG=ok

[QC hg_vs_lg_ccrcc] libsize/zero PNG=ok | detected PNG=ok

[QC hg_ccrcc_vs_normal_kidney] libsize/zero PNG=ok | detected PNG=ok

[QC lg_ccrcc_vs_normal_kidney] libsize/zero PNG=ok | detected PNG=ok

MA plots

```r
# ---- ma_plots_all ----
suppressPackageStartupMessages({
  library(here); library(data.table); library(ggplot2)
  library(SummarizedExperiment)
})

dir.create(here("plots","viz"), recursive = TRUE, showWarnings = FALSE)
```

```
# Bundle for sample counts in titles
B <- readRDS(here("outputs","kirc_clean_tmm_objects.rds"))
se_filt <- B$se_clean_filtered; assn <- B$assay_name
cts <- SummarizedExperiment::assay(se_filt, assn)
smeta <- as.data.frame(B$sample_meta); rownames(smeta) <- colnames(cts)
if (!"condition" %in% names(smeta)) {
  smeta$condition <- factor(ifelse(smeta$sample_type=="Solid Tissue Normal","Normal","Tumor"),
                            levels = c("Normal","Tumor"))
}
# Grade groups
pick_first <- function(df, xs){ for (k in xs) if (k %in% names(df)) return(df[[k]]); NULL }
raw_grade <- toupper(as.character(pick_first(smeta, c("tumor_grade","neoplasm_histologic_grade","grade"
parse_grade <- function(x){ x <- gsub("\\s+","",x); ifelse(grepl("G3|G4|HIGH",x),"HG", ifelse(grepl("G1
smeta$grade_group <- factor(ifelse(smeta$condition=="Tumor", parse_grade(raw_grade), NA), levels = c("L

contrast_groups <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = { sub <- smeta[smeta$condition %in% c("Normal","Tumor"),];
                                    c(Normal = sum(sub$condition=="Normal"), Tumor = sum(sub$condition=
    "hg_vs_lg_ccrcc"            = { sub <- smeta[smeta$condition=="Tumor" & !is.na(smeta$grade_group),]
                                    c(LG = sum(sub$grade_group=="LG"), HG = sum(sub$grade_group=="HG"))
    "hg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), HG = sum(sub$grade_group==
    "lg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), LG = sum(sub$grade_group==
    stop(sprintf("Unknown contrast: %s", label))
  )
}

# Safe PDF writer
safe_save_pdf <- function(path_pdf, plot, width=7, height=5){
  ok <- FALSE
  try({
    ggsave(path_pdf, plot = plot, width = width, height = height, device = cairo_pdf)
    if (file.exists(path_pdf) && file.info(path_pdf)$size > 0) ok <- TRUE
  }, silent = TRUE)
  if (!ok) try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    print(plot); grDevices::dev.off()
    if (file.exists(path_pdf) && file.info(path_pdf)$size > 0) ok <- TRUE
  }, silent = TRUE)
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height); print(plot); grDevices::dev.off()
    message("[MA] PDF failed; wrote SVG: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}

labels <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc","hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_normal_k

for (lab in labels) {
```

```r
  de_csv <- here("results", sprintf("kirc_%s_limma.csv", lab))
  if (!file.exists(de_csv)) { message("[MA] Skip (missing): ", basename(de_csv)); next }
  de_dt <- data.table::fread(de_csv)

  # Minimal schema guard
  need <- c("AveExpr","logFC")
  if (!all(need %in% names(de_dt))) { message("[MA] Missing columns in ", lab, " (need: AveExpr, logFC)

  # Title with group sizes
  gs <- contrast_groups(lab)
  ttl <- sprintf("MA - %s (%s)", lab, paste(paste0(names(gs), " n=", as.integer(gs)), collapse = ", "))

  p <- ggplot(de_dt, aes(AveExpr, logFC)) +
    geom_point(alpha = 0.4, size = 0.6) +
    geom_hline(yintercept = 0, linetype = 2) +
    labs(title = ttl, x = "Average expression", y = "log2 fold change")

  out_png <- here("plots","viz", sprintf("%s_MA.png", lab))
  out_pdf <- here("plots","viz", sprintf("%s_MA.pdf", lab))
  ggsave(out_png, plot = p, width = 7, height = 5, dpi = 300)
  invisible(safe_save_pdf(out_pdf, p, width = 7, height = 5))

  ok_png <- file.exists(out_png) && file.info(out_png)$size > 0
  ok_pdf <- file.exists(out_pdf) && file.info(out_pdf)$size > 0
  cat(sprintf("[MA %s] PNG=%s | PDF=%s\n", lab, if (ok_png) "ok" else "missing", if (ok_pdf) "ok" else
  stopifnot(ok_png)
}
```

[MA ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[MA hg_vs_lg_ccrcc] PNG=ok | PDF=ok

[MA hg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[MA lg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

Heatmaps with top DE genes

```r
# ---- heatmaps_top_de_all ----
suppressPackageStartupMessages({
  library(here); library(data.table); library(SummarizedExperiment)
  library(edgeR); library(matrixStats)
})
if (!requireNamespace("pheatmap", quietly = TRUE)) install.packages("pheatmap")
library(pheatmap)

dir.create(here("plots","viz"), recursive = TRUE, showWarnings = FALSE)

n_top <- 50

# Load bundle → counts + meta + logCPM
B <- readRDS(here("outputs","kirc_clean_tmm_objects.rds"))
```

```r
stopifnot(is.list(B), inherits(B$se_clean_filtered,"SummarizedExperiment"))
se_filt <- B$se_clean_filtered; assn <- B$assay_name
cts <- SummarizedExperiment::assay(se_filt, assn)

smeta <- as.data.frame(B$sample_meta); rownames(smeta) <- colnames(cts)
if (!"condition" %in% names(smeta)) {
  smeta$condition <- factor(ifelse(smeta$sample_type=="Solid Tissue Normal","Normal","Tumor"),
                            levels = c("Normal","Tumor"))
}
# LG/HG parsing
pick_first <- function(df, xs){ for (k in xs) if (k %in% names(df)) return(df[[k]]); NULL }
raw_grade <- toupper(as.character(pick_first(smeta, c("tumor_grade","neoplasm_histologic_grade","grade"
parse_grade <- function(x){ x <- gsub("\\s+","",x); ifelse(grepl("G3|G4|HIGH",x),"HG", ifelse(grepl("G1
smeta$grade_group <- factor(ifelse(smeta$condition=="Tumor", parse_grade(raw_grade), NA), levels = c("L

# TMM logCPM
dge <- edgeR::DGEList(counts = cts); dge <- edgeR::calcNormFactors(dge, method = "TMM")
logCPM_all <- edgeR::cpm(dge, log = TRUE, prior.count = 1)

labels <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc","hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_normal_k

contrast_keep_ids <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = rownames(smeta)[ smeta$condition %in% c("Normal","Tumor") ],
    "hg_vs_lg_ccrcc"            = rownames(smeta)[ smeta$condition=="Tumor" & !is.na(smeta$grade_group)
    "hg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tumo
    "lg_ccrcc_vs_normal_kidney" = rownames(smeta)[ (smeta$condition=="Normal") | (smeta$condition=="Tumo
    stop(sprintf("Unknown contrast: %s", label))
  )
}
contrast_groups <- function(label){
  switch(label,
    "ccrcc_vs_normal_kidney"    = { sub <- smeta[smeta$condition %in% c("Normal","Tumor"),];
                                    c(Normal = sum(sub$condition=="Normal"), Tumor = sum(sub$condition=
    "hg_vs_lg_ccrcc"            = { sub <- smeta[smeta$condition=="Tumor" & !is.na(smeta$grade_group),]
                                    c(LG = sum(sub$grade_group=="LG"), HG = sum(sub$grade_group=="HG"))
    "hg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), HG = sum(sub$grade_group==
    "lg_ccrcc_vs_normal_kidney" = { sub <- smeta[(smeta$condition=="Normal") | (smeta$condition=="Tumor
                                    c(Normal = sum(sub$condition=="Normal"), LG = sum(sub$grade_group==
  )
}

# Map DE ids to expression rows (handles Ensembl version)
map_de_to_rows <- function(de_dt, rn){
  gid <- de_dt$gene_id
  if (is.null(gid)) return(character(0))
  ok <- intersect(gid, rn)
  if (length(ok) < length(gid)) {
    strip_rn <- sub("\\.\\d+$","", rn)
    m <- match(sub("\\.\\d+$","", gid), strip_rn)
    ok <- unique(c(ok, rn[stats::na.omit(m)]))
  }
  unique(ok)
}

for (lab in labels) {
  de_csv <- here("results", sprintf("kirc_%s_limma.csv", lab))
  if (!file.exists(de_csv)) { message("[Heatmap] Skip (missing DE): ", basename(de_csv)); next }
  de_dt <- data.table::fread(de_csv)

  # Select top genes: prefer FDR<0.05 else top by |t|
de_dt <- de_dt[is.finite(FDR) & is.finite(t)]
de_dt[, t_abs := abs(t)]
data.table::setorder(de_dt, FDR, -t_abs)
  sig <- de_dt[FDR < 0.05]
  pick <- if (nrow(sig) >= 10) sig[1:min(n_top, .N)] else de_dt[1:min(n_top, .N)]
  if (nrow(pick) == 0) { message("[Heatmap] No DE rows for ", lab); next }

  keep_ids <- contrast_keep_ids(lab)
  keep_ids <- intersect(keep_ids, colnames(logCPM_all))
  if (length(keep_ids) < 20) { message("[Heatmap] Skip ", lab, " (too few samples)"); next }

  X <- logCPM_all[, keep_ids, drop = FALSE]
  gene_ids <- map_de_to_rows(pick, rownames(X))
  if (length(gene_ids) < 5) { message("[Heatmap] Only ", length(gene_ids), " matched rows for ", lab);

  M <- X[gene_ids, , drop = FALSE]

  # Row labels: prefer gene_name when available
  rowlabs <- pick[match(gene_ids, pick$gene_id), ifelse(nzchar(gene_name), gene_name, gene_id)]
  miss <- is.na(rowlabs); if (any(miss)) rowlabs[miss] <- sub("\\.\\d+$","", gene_ids[miss])
  rownames(M) <- rowlabs

  # Column annotation by contrast
  ann_col <- switch(lab,
    "hg_vs_lg_ccrcc"             = data.frame(group = smeta[keep_ids, "grade_group", drop = TRUE]),
    "hg_ccrcc_vs_normal_kidney" = data.frame(group = factor(ifelse(smeta[keep_ids,"condition"]=="Normal
    "lg_ccrcc_vs_normal_kidney" = data.frame(group = factor(ifelse(smeta[keep_ids,"condition"]=="Normal
    data.frame(group = smeta[keep_ids, "condition", drop = TRUE])
  )
  rownames(ann_col) <- keep_ids

  # Z-score per gene
  Mz <- t(scale(t(M)))

  # Title with group sizes
  gs <- contrast_groups(lab)
  ttl <- sprintf("Top DE (n=%d) - %s (%s)",
                 nrow(Mz), lab, paste(paste0(names(gs), " n=", as.integer(gs)), collapse = ", "))

  # Write PNG + PDF
  out_png <- here("plots","viz", sprintf("%s_heatmap_topDE.png", lab))
  out_pdf <- here("plots","viz", sprintf("%s_heatmap_topDE.pdf", lab))

  pheatmap::pheatmap(Mz, show_rownames = FALSE, annotation_col = ann_col,
                     main = ttl, filename = out_png, width = 8, height = 10)
  pheatmap::pheatmap(Mz, show_rownames = FALSE, annotation_col = ann_col,
                     main = ttl, filename = out_pdf, width = 8, height = 10)

  ok_png <- file.exists(out_png) && file.info(out_png)$size > 0
  ok_pdf <- file.exists(out_pdf) && file.info(out_pdf)$size > 0
  cat(sprintf("[Heatmap %s] PNG=%s | PDF=%s | genes=%d | samples=%d\n",
              lab, if (ok_png) "ok" else "missing", if (ok_pdf) "ok" else "failed",
              nrow(Mz), ncol(Mz)))
  stopifnot(ok_png)
}
```

[Heatmap ccrcc_vs_normal_kidney] PNG=ok | PDF=ok | genes=50 | samples=608

[Heatmap hg_vs_lg_ccrcc] PNG=ok | PDF=ok | genes=50 | samples=432

[Heatmap hg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok | genes=50 | samples=292

[Heatmap lg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok | genes=50 | samples=284

GSEA: helpers

```r
# ---- gsea_helpers ----
suppressPackageStartupMessages({
  library(here); library(data.table); library(ggplot2)
  library(msigdbr); library(fgsea)
})

# Directories
dir.create(here("results"), recursive = TRUE, showWarnings = FALSE)
dir.create(here("plots","gsea"), recursive = TRUE, showWarnings = FALSE)

# Build signed ranks from a DE CSV
build_ranks <- function(label){
  de_csv <- here("results", sprintf("kirc_%s_limma.csv", label))
  if (!file.exists(de_csv)) stop("Missing DE file: ", de_csv, call. = FALSE)
  dt <- data.table::fread(de_csv)

  # choose gene key: symbol preferred, else Ensembl core
  if ("gene_name" %in% names(dt) && any(nzchar(dt$gene_name))) {
    dt[, gene_key := ifelse(nzchar(gene_name), gene_name, sub("\\.\\d+$","", gene_id))]
  } else {
    dt[, gene_key := sub("\\.\\d+$","", gene_id)]
  }

  # keep finite t and non-empty key
  dt <- dt[is.finite(t) & !is.na(gene_key) & nzchar(gene_key)]
  dt[, t_abs := abs(t)]
  data.table::setorder(dt, -t_abs)  # top |t| first
  dt <- dt[!duplicated(gene_key)]   # keep best per gene

  setNames(dt$t, dt$gene_key)
}

# PDF saver with fallbacks
safe_save_pdf <- function(path_pdf, plot, width=8, height=10){
  ok <- FALSE
  try({
    ggsave(path_pdf, plot = plot, width = width, height = height, device = cairo_pdf)
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  if (!ok) try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    print(plot); grDevices::dev.off()
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height); print(plot); grDevices::dev.off()
    message("[GSEA] PDF failed; wrote SVG: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}

# msigdbr fetchers (correct API: category/subcategory)
msig_hallmark <- function() {
  m <- msigdbr(species = "Homo sapiens", category = "H")
  split(m$gene_symbol, m$gs_name)
}
msig_reactome <- function() {
  m <- msigdbr(species = "Homo sapiens", category = "C2", subcategory = "CP:REACTOME")
  split(m$gene_symbol, m$gs_name)
}
msig_kegg <- function() {
  m <- msigdbr(species = "Homo sapiens", category = "C2", subcategory = "CP:KEGG")
  split(m$gene_symbol, m$gs_name)
}
msig_go_all <- function(onts = c("BP","MF","CC")) {
  parts <- lapply(onts, function(ont) {
    m <- msigdbr(species = "Homo sapiens", category = "C5", subcategory = paste0("GO:", ont))
    split(m$gene_symbol, m$gs_name)
  })
  out <- do.call(c, parts)
  out[vapply(out, length, integer(1)) > 0]
}

# Knit-safe fgsea runner (used later by all chunks)
run_gsea_ALEJ_safe_2025 <- function(label, gs_list, libtag,
                                    minSize = 15, maxSize = 5000, nperm = 10000) {
  ranks <- build_ranks(label)
  if (!length(ranks)) stop("Empty ranks for ", label, call. = FALSE)

  # intersect genes per set; filter size
  gs <- lapply(gs_list, function(v) intersect(unique(v), names(ranks)))
  sizes <- vapply(gs, length, integer(1))
  keep <- sizes >= minSize & sizes <= maxSize
  gs <- gs[keep]
  if (!length(gs)) stop(sprintf("[%s/%s] No gene sets after size filter.", libtag, label), call. = FALSI

  set.seed(1234)
  fg <- fgsea(pathways = gs, stats = ranks, nperm = nperm, minSize = minSize, maxSize = maxSize)
  fg <- data.table::as.data.table(fg)
  fg[, NES_abs := abs(NES)]
  data.table::setorder(fg, padj, -NES_abs)

  # Write results
  out_tsv <- here("results", sprintf("kirc_%s_gsea_%s.tsv", label, tolower(libtag)))
  data.table::fwrite(fg, out_tsv, sep = "\t")

  # Leading-edge long table (if present)
  if ("leadingEdge" %in% names(fg)) {
    le_tbl <- data.table(
      pathway = rep(fg$pathway, lengths(fg$leadingEdge)),
      gene    = unlist(fg$leadingEdge),
      NES     = rep(fg$NES, lengths(fg$leadingEdge)),
      padj    = rep(fg$padj, lengths(fg$leadingEdge))
    )
    out_le <- here("results", sprintf("kirc_%s_gsea_%s_leading_edge.tsv", label, tolower(libtag)))
    data.table::fwrite(le_tbl, out_le, sep = "\t")
  }

  message(sprintf("[GSEA %s] %s -> sets=%d, padj<0.05=%d, file=%s",
                  toupper(libtag), label, nrow(fg), sum(fg$padj < 0.05, na.rm = TRUE), basename(out_tsv
  invisible(fg)
}
```

GSEA: fetch Hallmark, KEGG, Reacome and GO gene sets

```r
# ---- gsea_fetch_sets_go_fix ----
suppressPackageStartupMessages({
  library(here); library(data.table); library(msigdbr)
})

# v10 helpers (as before)
msig_hallmark <- function() {
  m <- msigdbr(species = "Homo sapiens", collection = "H")
  split(m$gene_symbol, m$gs_name)
}
msig_reactome <- function() {
  m <- msigdbr(species = "Homo sapiens", collection = "C2", subcollection = "CP:REACTOME")
  split(m$gene_symbol, m$gs_name)
}
# NEW: fetch BP/MF/CC separately, then merge
```

```r
msig_go_all <- function(onts = c("BP","MF","CC")) {
  stopifnot(length(onts) >= 1)
  parts <- lapply(onts, function(ont) {
    m <- msigdbr(species = "Homo sapiens", collection = "C5", subcollection = paste0("GO:", ont))
    split(m$gene_symbol, m$gs_name)
  })
  # merge lists (pathways are unique by name)
  out <- do.call(c, parts)
  # drop empties/dups defensively
  out <- out[vapply(out, length, integer(1)) > 0]
  out
}

# KEGG fallback (unchanged)
msig_kegg <- function() {
  try({
    m <- msigdbr(species = "Homo sapiens", collection = "C2", subcollection = "CP:KEGG")
    if (nrow(m)) return(split(m$gene_symbol, m$gs_name))
  }, silent = TRUE)

  # Fallback via KEGGREST + org.Hs.eg.db
  need_bioc <- function(pkgs){
    to_install <- setdiff(pkgs, rownames(installed.packages()))
    if (length(to_install)) {
      if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
      BiocManager::install(to_install, ask = FALSE, update = FALSE)
    }
  }
  need_bioc(c("KEGGREST","org.Hs.eg.db","AnnotationDbi"))
  suppressPackageStartupMessages({ library(KEGGREST); library(org.Hs.eg.db); library(AnnotationDbi) })

  paths <- KEGGREST::keggList("pathway", "hsa")
  p_ids <- names(paths); p_names <- as.character(paths)

  kegg_sets <- vector("list", length(p_ids)); names(kegg_sets) <- sub("^path:", "", p_ids)
  for (i in seq_along(p_ids)) {
    mem <- try(KEGGREST::keggGet(p_ids[i])[[1]]$GENE, silent = TRUE)
    if (inherits(mem, "try-error") || is.null(mem)) next
    entrez <- mem[seq(1, length(mem), by = 2)]
    syms <- AnnotationDbi::mapIds(org.Hs.eg.db, keys = entrez,
                                  column = "SYMBOL", keytype = "ENTREZID", multiVals = "first")
    kegg_sets[[i]] <- unique(na.omit(unname(syms)))
  }
  names(kegg_sets) <- paste0(names(kegg_sets), " - ",
                             sub(" - Homo sapiens .*", "", p_names))
  kegg_sets <- kegg_sets[vapply(kegg_sets, length, integer(1)) >= 10]
  if (!length(kegg_sets)) stop("KEGG fallback produced no sets.")
  kegg_sets
}

# Fetch all four collections
hallmark <- msig_hallmark()
reactome <- msig_reactome()
```

```r
go_sets  <- msig_go_all(c("BP","MF","CC"))
kegg     <- msig_kegg()

# Checks
stopifnot(length(hallmark) >= 10, length(reactome) >= 100, length(go_sets) >= 500, length(kegg) >= 50)
cat("[Sets] H:", length(hallmark),
    "| REACTOME:", length(reactome),
    "| GO(BP/MF/CC):", length(go_sets),
    "| KEGG:", length(kegg), "\n")
```

[Sets] H: 50 | REACTOME: 1787 | GO(BP/MF/CC): 10480 | KEGG: 340

```r
#[Sets] H: 50 | REACTOME: 1787 | GO(BP/MF/CC): 10480 | KEGG: 340
```

GSEA: Cache gene sets (don't have to refetch every time)

```r
# ---- gsea_cache_sets ----
suppressPackageStartupMessages({ library(here) })

# Require in-memory objects from previous chunk
need <- c("hallmark","reactome","kegg","go_sets")
missing <- need[!vapply(need, exists, logical(1))]
if (length(missing)) stop("Missing gene set objects: ", paste(missing, collapse = ", "))

dir.create(here("outputs"), recursive = TRUE, showWarnings = FALSE)

saveRDS(hallmark, here("outputs","msig_hallmark_sets.rds"))
saveRDS(reactome, here("outputs","msig_reactome_sets.rds"))
saveRDS(kegg,     here("outputs","kegg_sets.rds"))
saveRDS(go_sets,  here("outputs","msig_go_sets.rds"))

ok <- all(file.exists(
  here("outputs","msig_hallmark_sets.rds"),
  here("outputs","msig_reactome_sets.rds"),
  here("outputs","kegg_sets.rds"),
  here("outputs","msig_go_sets.rds")
))
cat("[Cache] Saved MSigDB sets → outputs/ ; ok =", ok, "\n")
```

[Cache] Saved MSigDB sets → outputs/ ; ok = TRUE

```r
stopifnot(ok)
```

gsea_build_ranks_and_smoke

```r
# ---- gsea_build_ranks_and_smoke ----
suppressPackageStartupMessages({ library(here); library(data.table) })
```

```r
# Define build_ranks() only if missing (prefer earlier helper)
if (!exists("build_ranks")) {
  build_ranks <- function(label){
    de_csv <- here("results", sprintf("kirc_%s_limma.csv", label))
    if (!file.exists(de_csv)) stop("Missing DE file: ", de_csv, call. = FALSE)
    dt <- data.table::fread(de_csv)

    # symbol preferred, else Ensembl core
    if ("gene_name" %in% names(dt) && any(nzchar(dt$gene_name))) {
      dt[, gene_key := ifelse(nzchar(gene_name), gene_name, sub("\\.\\d+$","", gene_id))]
    } else {
      dt[, gene_key := sub("\\.\\d+$","", gene_id)]
    }

    dt <- dt[is.finite(t) & !is.na(gene_key) & nzchar(gene_key)]
    dt[, t_abs := abs(t)]
    data.table::setorder(dt, -t_abs)
    dt <- dt[!duplicated(gene_key)]

    setNames(dt$t, dt$gene_key)
  }
}

# Guard: need the safe runner + Hallmark sets
need_objs <- c("run_gsea_ALEJ_safe_2025", "hallmark")
miss <- need_objs[!vapply(need_objs, exists, logical(1))]
if (length(miss)) stop("Missing objects: ", paste(miss, collapse = ", "),
                       "\nDid you run the previous GSEA helpers and fetch-sets chunks?", call. = FALSE)

# Smoke test on one contrast
lab <- "ccrcc_vs_normal_kidney"
stopifnot(file.exists(here("results", sprintf("kirc_%s_limma.csv", lab))))

fg <- run_gsea_ALEJ_safe_2025(lab, hallmark, libtag = "HALLMARK", nperm = 2000)

# Compact preview
cols <- intersect(c("pathway","NES","pval","padj","size"), names(fg))
print(utils::head(as.data.frame(fg)[, cols, drop = FALSE], 10))
```

pathway NES pval padj size

1 HALLMARK_MTORC1_SIGNALING -1.534621 0.001089325 0.04116356 198

2 HALLMARK_ESTROGEN_RESPONSE_LATE -1.529362 0.001646542 0.04116356 185

3 HALLMARK_G2M_CHECKPOINT -1.509564 0.002724796 0.04541326 196

4 HALLMARK_HEME_METABOLISM -1.486532 0.004947774 0.06184717 182

5 HALLMARK_HYPOXIA -1.447257 0.007675439 0.07675439 190

6 HALLMARK_MITOTIC_SPINDLE -1.441235 0.009809264 0.08174387 196

7 HALLMARK_IL6_JAK_STAT3_SIGNALING -1.511254 0.017661389 0.11038368 81

8 HALLMARK_UV_RESPONSE_DN -1.430516 0.017454955 0.11038368 144

9 HALLMARK_APICAL_SURFACE -1.542996 0.026530612 0.11443102 43

10 HALLMARK_DNA_REPAIR -1.393712 0.024239008 0.11443102 148

```r
cat("Wrote: ", here("results", sprintf("kirc_%s_gsea_hallmark.tsv", lab)), "\n", sep = "")
```

Wrote: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_

GSEA— Run GSEA for all 4 contrasts × {H, REACTOME, KEGG, GO}

```r
# ---- gsea_run_all ----
suppressPackageStartupMessages({ library(here); library(data.table) })

# Guards
if (!exists("run_gsea_ALEJ_safe_2025")) stop("Missing run_gsea_ALEJ_safe_2025(); run the GSEA helpers c

# Ensure gene sets are in memory; otherwise load cached copies
load_if_missing <- function(obj, path) {
  if (!exists(obj, inherits = FALSE)) {
    f <- here("outputs", path)
    if (!file.exists(f)) stop("Missing gene set object '", obj, "' and cache file: ", f, call. = FALSE)
    assign(obj, readRDS(f), envir = .GlobalEnv)
  }
}
load_if_missing("hallmark", "msig_hallmark_sets.rds")
load_if_missing("reactome", "msig_reactome_sets.rds")
load_if_missing("kegg",     "kegg_sets.rds")
load_if_missing("go_sets",  "msig_go_sets.rds")
```

```r
labels <- c("ccrcc_vs_normal_kidney",
            "hg_vs_lg_ccrcc",
            "hg_ccrcc_vs_normal_kidney",
            "lg_ccrcc_vs_normal_kidney")

dir.create(here("results"), recursive = TRUE, showWarnings = FALSE)

for (lab in labels) {
  de_csv <- here("results", sprintf("kirc_%s_limma.csv", lab))
  if (!file.exists(de_csv)) { message("[GSEA] Skip (missing DE): ", basename(de_csv)); next }

  invisible(run_gsea_ALEJ_safe_2025(lab, hallmark, libtag = "HALLMARK", minSize = 15, maxSize = 5000, np
  invisible(run_gsea_ALEJ_safe_2025(lab, reactome, libtag = "REACTOME", minSize = 15, maxSize = 5000, np
  invisible(run_gsea_ALEJ_safe_2025(lab, kegg,     libtag = "KEGG",     minSize = 15, maxSize = 5000, np
  invisible(run_gsea_ALEJ_safe_2025(lab, go_sets,  libtag = "GO",       minSize = 50, maxSize = 5000, np
}

# Quick existence ping for one output
chk <- here("results", "kirc_ccrcc_vs_normal_kidney_gsea_hallmark.tsv")
cat("[GSEA] Example file exists:", file.exists(chk), "→", normalizePath(chk, winslash = "/"), "\n")
```

[GSEA] Example file exists: TRUE → /Users/alejandrosanchez_2/Library/CloudStorage/On
UniversityofUtah/R_work/KIRC_TCGA/results/kirc_ccrcc_vs_normal_kidney_gsea_ha

Hallmark Top 10 pathways per contrast

```r
# ---- gsea_hallmark_top10 ----
suppressPackageStartupMessages({ library(here); library(data.table); library(ggplot2) })
dir.create(here("plots","gsea"), recursive = TRUE, showWarnings = FALSE)

contrast_labels <- c(
  "ccrcc_vs_normal_kidney",
  "hg_vs_lg_ccrcc",
  "hg_ccrcc_vs_normal_kidney",
  "lg_ccrcc_vs_normal_kidney"
)

normalize_gsea_table <- function(path){
  stopifnot(file.exists(path))
  dt <- data.table::fread(path)

  pick <- function(cands) { h <- intersect(cands, names(dt)); if (length(h)) h[1] else NA_character_ }

  # NES (fallback to ES)
  nes_col <- pick(c("NES","nes","NES.x","NES.y"))
  es_col  <- pick(c("ES","es","ES.x","ES.y"))
  if (is.na(nes_col)) {
    if (is.na(es_col)) stop("No NES/ES column in: ", basename(path))
    dt[, NES := as.numeric(get(es_col))]
  } else {
    data.table::setnames(dt, nes_col, "NES"); dt[, NES := as.numeric(NES)]
  }
  # padj (fallback to BH on raw p)
  padj_col <- pick(c("padj","adj.P.Val","FDR","qval","qvalue","q.val","q"))
  rawp_col <- pick(c("pval","P.Value","pvalue","p"))
  if (is.na(padj_col)) {
    if (is.na(rawp_col)) dt[, padj := NA_real_] else dt[, padj := p.adjust(as.numeric(get(rawp_col)), "
  } else {
    data.table::setnames(dt, padj_col, "padj"); dt[, padj := as.numeric(padj)]
  }

  # pathway column
  if (!"pathway" %in% names(dt)) {
    alt <- pick(c("pathway","pathways","gs","term"))
    if (is.na(alt)) stop("No pathway column in: ", basename(path))
    data.table::setnames(dt, alt, "pathway")
  }

  keep <- intersect(c("pathway","NES","padj","size"), names(dt))
  as.data.frame(dt[, ..keep])
}

safe_save_pdf <- function(path_pdf, plot, width=8, height=6){
  ok <- FALSE
  # Cairo
  try({ ggsave(path_pdf, plot = plot, width = width, height = height, device = cairo_pdf)
      ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0 }, silent = TRUE)
  # base pdf
  if (!ok) try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    print(plot); grDevices::dev.off()
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  # SVG fallback
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height); print(plot); grDevices::dev.off()
    message("[Top10] PDF failed; wrote SVG: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}

for (lab in contrast_labels) {
  f <- here("results", sprintf("kirc_%s_gsea_hallmark.tsv", lab))
  if (!file.exists(f)) { message("[Top10 HALLMARK] Missing: ", basename(f)); next }
  dt <- tryCatch(normalize_gsea_table(f), error = function(e) { message("[Top10] ", e$message); NULL })
  if (is.null(dt)) next

  # choose rows
  sig <- subset(dt, is.finite(NES) & is.finite(padj) & padj < 0.05)
  top <- if (nrow(sig) > 0) sig else subset(dt, is.finite(NES))
  if (!nrow(top)) { message("[Top10] No rows for ", lab); next }

  # rank by |NES| without expression in setorder
  top$NES_abs <- abs(top$NES)
  top <- top[order(-top$NES_abs), , drop = FALSE]
  top10 <- top[seq_len(min(10L, nrow(top))), , drop = FALSE]

  top10$pathway <- factor(top10$pathway, levels = rev(top10$pathway))
  top10$lab_txt <- sprintf("NES=%.2f\npadj=%s",
                           top10$NES,
                           ifelse(is.na(top10$padj), "NA", formatC(top10$padj, format="e", digits=2)))

  p <- ggplot(top10, aes(x = pathway, y = NES, fill = NES > 0)) +
    geom_col() +
    coord_flip() +
    geom_text(aes(label = lab_txt), hjust = ifelse(top10$NES > 0, -0.05, 1.05), size = 3) +
    labs(title = sprintf("Top 10 Hallmark pathways - %s", lab), x = "", y = "NES") +
    guides(fill = "none")

  ypad <- max(abs(top10$NES)) * 0.15
  p <- p + ylim(min(min(top10$NES), 0) - ypad, max(max(top10$NES), 0) + ypad)

  out_png <- here("plots","gsea", sprintf("%s_hallmark_top10.png", lab))
  out_pdf <- here("plots","gsea", sprintf("%s_hallmark_top10.pdf", lab))
  ggsave(out_png, plot = p, width = 8, height = max(4, 0.5 * nrow(top10) + 1), dpi = 300)
  invisible(safe_save_pdf(out_pdf, p, width = 8, height = max(4, 0.5 * nrow(top10) + 1)))

  ok_png <- file.exists(out_png) && file.info(out_png)$size > 0
  cat(sprintf("[Hallmark Top10 %s] PNG=%s | PDF=%s\n",
              lab, if (ok_png) "ok" else "missing",
              if (file.exists(out_pdf) && file.info(out_pdf)$size > 0) "ok" else "failed/SVG"))
  stopifnot(ok_png)
}
```

[Hallmark Top10 ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[Hallmark Top10 hg_vs_lg_ccrcc] PNG=ok | PDF=ok

[Hallmark Top10 hg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

[Hallmark Top10 lg_ccrcc_vs_normal_kidney] PNG=ok | PDF=ok

All 4 contrasts, top 10 pathways

```
# ---- gsea_top10_reactome_kegg_go ----
suppressPackageStartupMessages({ library(here); library(data.table); library(ggplot2) })
dir.create(here("plots","gsea"), recursive = TRUE, showWarnings = FALSE)

contrast_labels <- c(
  "ccrcc_vs_normal_kidney",
  "hg_vs_lg_ccrcc",
  "hg_ccrcc_vs_normal_kidney",
  "lg_ccrcc_vs_normal_kidney"
)
lib_tags <- c("reactome","kegg","go")  # file suffixes are lowercase
```

```r
normalize_gsea_table <- function(path){
  stopifnot(file.exists(path))
  dt <- data.table::fread(path)
  pick <- function(cands){ h <- intersect(cands, names(dt)); if (length(h)) h[1] else NA_character_ }

  # NES (fallback to ES)
  nes_col <- pick(c("NES","nes","NES.x","NES.y"))
  es_col  <- pick(c("ES","es","ES.x","ES.y"))
  if (is.na(nes_col)) {
    if (is.na(es_col)) stop("No NES/ES column in: ", basename(path))
    dt[, NES := as.numeric(get(es_col))]
  } else {
    data.table::setnames(dt, nes_col, "NES"); dt[, NES := as.numeric(NES)]
  }

  # padj (fallback to BH on raw p)
  padj_col <- pick(c("padj","adj.P.Val","FDR","qval","qvalue","q.val","q"))
  rawp_col <- pick(c("pval","P.Value","pvalue","p"))
  if (is.na(padj_col)) {
    if (is.na(rawp_col)) dt[, padj := NA_real_] else dt[, padj := p.adjust(as.numeric(get(rawp_col)), "
  } else {
    data.table::setnames(dt, padj_col, "padj"); dt[, padj := as.numeric(padj)]
  }

  # pathway column
  if (!"pathway" %in% names(dt)) {
    alt <- pick(c("pathway","pathways","gs","term"))
    if (is.na(alt)) stop("No pathway column in: ", basename(path))
    data.table::setnames(dt, alt, "pathway")
  }

  keep <- intersect(c("pathway","NES","padj","size"), names(dt))
  as.data.frame(dt[, ..keep])
}

safe_save_pdf <- function(path_pdf, plot, width=8, height=6){
  ok <- FALSE
  try({ ggsave(path_pdf, plot = plot, width = width, height = height, device = cairo_pdf)
      ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0 }, silent = TRUE)
  if (!ok) try({
    grDevices::pdf(path_pdf, width = width, height = height, useDingbats = FALSE)
    print(plot); grDevices::dev.off()
    ok <- file.exists(path_pdf) && file.info(path_pdf)$size > 0
  }, silent = TRUE)
  if (!ok && requireNamespace("svglite", quietly = TRUE)) {
    path_svg <- sub("\\.pdf$", ".svg", path_pdf)
    svglite::svglite(path_svg, width = width, height = height); print(plot); grDevices::dev.off()
    message("[Top10] PDF failed; wrote SVG: ", basename(path_svg))
    return(list(pdf_ok = FALSE, alt_svg = path_svg))
  }
  list(pdf_ok = ok, alt_svg = if (ok) NA_character_ else NA_character_)
}
```

```r
for (lab in contrast_labels) {
  for (lib in lib_tags) {
    f <- here("results", sprintf("kirc_%s_gsea_%s.tsv", lab, lib))
    if (!file.exists(f)) { message("[Top10] Missing: ", basename(f)); next }
    dt <- tryCatch(normalize_gsea_table(f), error = function(e) { message("[Top10] ", e$message); NULL
    if (is.null(dt)) next

    sig <- subset(dt, is.finite(NES) & is.finite(padj) & padj < 0.05)
    top <- if (nrow(sig) > 0) sig else subset(dt, is.finite(NES))
    if (!nrow(top)) { message("[Top10] No rows to plot for ", lab, " / ", toupper(lib)); next }

    top$NES_abs <- abs(top$NES)
    top <- top[order(-top$NES_abs), , drop = FALSE]
    top10 <- top[seq_len(min(10L, nrow(top))), , drop = FALSE]

    top10$pathway <- factor(top10$pathway, levels = rev(top10$pathway))
    top10$lab_txt <- sprintf("NES=%.2f\npadj=%s", top10$NES, ifelse(is.na(top10$padj), "NA", formatC(top

    p <- ggplot(top10, aes(x = pathway, y = NES, fill = NES > 0)) +
      geom_col() +
      coord_flip() +
      geom_text(aes(label = lab_txt), hjust = ifelse(top10$NES > 0, -0.05, 1.05), size = 3) +
      labs(title = sprintf("Top 10 %s pathways - %s", toupper(lib), lab), x = "", y = "NES") +
      guides(fill = "none")

    ypad <- max(abs(top10$NES)) * 0.15
    p <- p + ylim(min(min(top10$NES), 0) - ypad, max(max(top10$NES), 0) + ypad)

    out_png <- here("plots","gsea", sprintf("%s_%s_top10.png", lab, lib))
    out_pdf <- here("plots","gsea", sprintf("%s_%s_top10.pdf", lab, lib))
    ggsave(out_png, plot = p, width = 8, height = max(4, 0.5 * nrow(top10) + 1), dpi = 300)
    invisible(safe_save_pdf(out_pdf, p, width = 8, height = max(4, 0.5 * nrow(top10) + 1)))

    ok_png <- file.exists(out_png) && file.info(out_png)$size > 0
    cat(sprintf("[Top10 %s / %s] PNG=%s | PDF=%s\n",
                lab, toupper(lib), if (ok_png) "ok" else "missing",
                if (file.exists(out_pdf) && file.info(out_pdf)$size > 0) "ok" else "failed/SVG"))
    stopifnot(ok_png)
  }
}
```

[Top10 ccrcc__vs__normal__kidney / REACTOME] PNG=ok | PDF=ok

[Top10 ccrcc__vs__normal__kidney / KEGG] PNG=ok | PDF=ok

[Top10 ccrcc__vs__normal__kidney / GO] PNG=ok | PDF=ok

[Top10 hg__vs__lg__ccrcc / REACTOME] PNG=ok | PDF=ok

[Top10 hg__vs__lg__ccrcc / KEGG] PNG=ok | PDF=ok

[Top10 hg__vs__lg__ccrcc / GO] PNG=ok | PDF=ok

[Top10 hg__ccrcc__vs__normal__kidney / REACTOME] PNG=ok | PDF=ok

[Top10 hg__ccrcc__vs__normal__kidney / KEGG] PNG=ok | PDF=ok

[Top10 hg__ccrcc__vs__normal__kidney / GO] PNG=ok | PDF=ok

[Top10 lg__ccrcc__vs__normal__kidney / REACTOME] PNG=ok | PDF=ok

[Top10 lg__ccrcc__vs__normal__kidney / KEGG] PNG=ok | PDF=ok

[Top10 lg__ccrcc__vs__normal__kidney / GO] PNG=ok | PDF=ok

Compact file index

```r
# ---- files_index_build ----
suppressPackageStartupMessages({ library(here); library(data.table) })
dir.create(here("results"), recursive = TRUE, showWarnings = FALSE)

labels <- c("ccrcc_vs_normal_kidney","hg_vs_lg_ccrcc","hg_ccrcc_vs_normal_kidney","lg_ccrcc_vs_normal_k
libs   <- c("hallmark","reactome","kegg","go")

rows <- list(
  list(path = here("data","tcga_kirc_expression.tsv"),
       purpose = "Filtered count matrix (genes x samples)."),
  list(path = here("data","tcga_kirc_samples.tsv"),
       purpose = "Sample metadata with condition and parsed grade_group."),
  list(path = here("data","tcga_kirc_genes.tsv"),
       purpose = "Gene annotations (Ensembl ID, gene_name)."),
  list(path = here("outputs","kirc_clean_tmm_objects.rds"),
       purpose = "Bundle: filtered SE, assay name, TMM DGE, sample & gene meta.")
)

# Optional clinical (silently include if present)
rows[[length(rows)+1L]] <- list(path = here("data","tcga_kirc_clinical.tsv"),
                                 purpose = "Clinical phenotypes (optional).")

# DE CSVs + per-contrast RDS model objects
for (lb in labels) {
  rows[[length(rows)+1L]] <- list(path = here("results", sprintf("kirc_%s_limma.csv", lb)),
                                   purpose = sprintf("DE results (limma-voom) for %s.", lb))
  rows[[length(rows)+1L]] <- list(path = here("outputs", sprintf("kirc_%s_objects.rds", lb)),
                                  purpose = sprintf("Voom/design/group objects for %s.", lb))
}

# GSEA tables per library (results + leading-edge if present)
for (lb in labels) {
  for (L in libs) {
    rows[[length(rows)+1L]] <- list(path = here("results", sprintf("kirc_%s_gsea_%s.tsv", lb, L)),
                                    purpose = sprintf("GSEA %s results for %s.", toupper(L), lb))
    rows[[length(rows)+1L]] <- list(path = here("results", sprintf("kirc_%s_gsea_%s_leading_edge.tsv", l
                                    purpose = sprintf("GSEA %s leading-edge genes for %s.", toupper(L),
  }
}

DT <- rbindlist(rows)
DT[, exists := file.exists(path)]
DT[, size_bytes := fifelse(exists, file.info(path)$size, NA_integer_)]
DT[, size_mb := round(size_bytes/1024^2, 2)]
setcolorder(DT, c("path","exists","size_mb","purpose"))

# Save & print
out_idx <- here("results", "kirc_file_index_for_sphingolipid_work.tsv")
fwrite(DT, out_idx, sep = "\t")
print(DT)
```

path

1: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

2: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

3: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

4: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

5: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

6: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

7: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

8: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

9: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work

10: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

11: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

12: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

13: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

14: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

15: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

16: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

17: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

18: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

19: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

20: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

21: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

22: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

23: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

24: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

25: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

26: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

27: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_wor

```r
cat("[Index] Wrote: ", normalizePath(out_idx, winslash="/"), "\n", sep = "")
```

[Index] Wrote: /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive-UniversityofUtah/R_work/KIRC_TCGA/results/kirc_file_index_for_sphingolipid_work

```r
stopifnot(file.exists(out_idx))
```

File index checks

```r
# ---- files_index_checks ----
suppressPackageStartupMessages({ library(here); library(data.table) })

expr_f <- here("data","tcga_kirc_expression.tsv")
samp_f <- here("data","tcga_kirc_samples.tsv")
gene_f <- here("data","tcga_kirc_genes.tsv")
bundle <- here("outputs","kirc_clean_tmm_objects.rds")

stopifnot(file.exists(expr_f), file.exists(samp_f), file.exists(gene_f), file.exists(bundle))

expr_head <- fread(expr_f, nrows = 5)
samp_head <- fread(samp_f, nrows = 5)
gene_head <- fread(gene_f, nrows = 5)

stopifnot("gene_id" %in% names(expr_head))
stopifnot(all(c("sample_id","sample_type") %in% names(samp_head)))

if (!"gene_name" %in% names(gene_head)) message("[Warn] 'gene_name' not found in tcga_kirc_genes.tsv; E

B <- readRDS(bundle)
stopifnot(is.list(B), "se_clean_filtered" %in% names(B), "sample_meta" %in% names(B), "gene_meta" %in% 
cat("[Checks] Core files and columns are available.\n")
```

[Checks] Core files and columns are available.

Session summary

```r
# ---- session_summary ----
suppressPackageStartupMessages({ library(here) })

dir.create(here("logs"), recursive = TRUE, showWarnings = FALSE)
log_file <- here("logs", sprintf("sessioninfo_viz_%s.txt", format(Sys.Date(), "%Y%m%d")))
status_file <- here("logs", sprintf("renv_status_viz_%s.txt", format(Sys.Date(), "%Y%m%d")))

# 1) Full session info → file (no inline dump)
con <- file(log_file, open = "wt")
sink(con)
cat("Session summary - visualization Rmd\n===============================\n")
cat("Project root: ", normalizePath(here::here(), winslash="/"), "\n\n", sep="")
if (requireNamespace("sessioninfo", quietly = TRUE)) {
  sessioninfo::session_info()
} else {
  sessionInfo()
}
sink(); close(con)

# 2) renv::status() → strip ANSI → file (no inline dump)
if (requireNamespace("renv", quietly = TRUE)) {
  # disable ANSI colors at the source
  o <- options(cli.num_colors = 0, crayon.enabled = FALSE)
  on.exit(options(o), add = TRUE)

  rs <- tryCatch(capture.output(renv::status()), error = function(e) character())
  if (length(rs)) {
    # strip any lingering ANSI escapes
    if (requireNamespace("cli", quietly = TRUE)) rs <- cli::ansi_strip(rs)
    writeLines(rs, status_file)
  }
}

# 3) Compact inline notice only (ASCII-safe)
cat("[Session] Wrote full session info →", normalizePath(log_file,  winslash="/"), "\n")
```

[Session] Wrote full session info → /Users/alejandrosanchez_2/Library/CloudStorage/OneDr UniversityofUtah/R_work/KIRC_TCGA/logs/sessioninfo_viz_20251108.txt

```
if (file.exists(status_file)) {
  cat("[Session] Wrote renv status     →", normalizePath(status_file, winslash="/"), "\n")
}
```

[Session] Wrote renv status → /Users/alejandrosanchez_2/Library/CloudStorage/OneDrive- UniversityofUtah/R_work/KIRC_TCGA/logs/renv_status_viz_20251108.txt

Archive inputs and ouputs

```
zipfile <- here::here("results", sprintf("kirc_tcga_release_%s.zip", format(Sys.Date(), "%Y%m%d")))
files <- c(
  list.files(here::here("results"), full.names = TRUE, recursive = TRUE),
  list.files(here::here("plots"),   full.names = TRUE, recursive = TRUE),
  here::here("outputs","kirc_clean_tmm_objects.rds"),
  here::here("results","kirc_file_index_for_sphingolipid_work.tsv"),
  here::here("results","kirc_sphingolipid_starter_config.tsv")
)
utils::zip(zipfile, files[dir.exists(dirname(files)) & file.exists(files)])
```

“‘