RNA-seq

-

2025-09-05

Table of contents

1 Create genome database

- Genome sequencing (   )
- Genome annotation NCBI
- GenBank Batch Parsing and Multi-format Export (bioconda)
- KEGG annotation (KEGG Web service)
- Create and save gson dbfile

1.1 Genome sequencing (   )

1.2 Genome annotation NCBI

1.3 GenBank Batch Parsing and Multi-format Export (bioconda)

```python
from pathlib import Path
from Bio import SeqIO
```

```
ModuleNotFoundError: No module named 'Bio'
```

```python
import pandas as pd
```

```
ModuleNotFoundError: No module named 'pandas'
```

```python
def gb_to_fasta_gtf_protein(gb_path):
    gb_path = Path(gb_path)
    out_dir = gb_path.parent

    fasta_file = out_dir / f"{gb_path.stem}.fasta"
    gtf_file = out_dir / f"{gb_path.stem}.gtf"
    protein_file = out_dir / f"{gb_path.stem}_protein.fasta"
    #   GenBank
    records = list(SeqIO.parse(gb_path, "genbank"))

    #
    SeqIO.write(records, fasta_file, "fasta")

    #   GTF
    gtf_rows = []
    protein_records = []

    for rec in records:
        for feature in rec.features:
            if feature.type in ["gene", "CDS"]:
                start = int(feature.location.start) + 1  # GTF   1-based
                end = int(feature.location.end)
                strand = "+" if feature.strand == 1 else "-"
                attr_parts = []

                # gene_id
                gene_id = feature.qualifiers.get("locus_tag", ["NA"])[0]
                attr_parts.append(f'gene_id "{gene_id}"')

                # gene_name
                if "gene" in feature.qualifiers:
                    attr_parts.append(f'gene_name "{feature.qualifiers["gene"][0]}"')

                # product
                if "product" in feature.qualifiers:
                    attr_parts.append(f'product "{feature.qualifiers["product"][0]}"')

                attributes = "; ".join(attr_parts)

                gtf_rows.append([
                    rec.id, "GenBank", feature.type, start, end, ".", strand, ".", attribute
                ])

                #
                if feature.type == "CDS" and "translation" in feature.qualifiers:
                    protein_seq = feature.qualifiers["translation"][0]
                    protein_records.append(
                        SeqIO.SeqRecord(
                            seq=feature.qualifiers["translation"][0],
                                id=gene_id,
                                description=feature.qualifiers.get("product", [""])[0]
                            )
                        )

    #   GTF
    gtf_df = pd.DataFrame(gtf_rows, columns=[
        "seqname", "source", "feature", "start", "end", "score", "strand", "frame", "attribu
    ])
    gtf_df.to_csv(gtf_file, sep="\t", index=False, header=False)

    #
    with open(protein_file, "w") as f:
        for rec in protein_records:
            f.write(f">{rec.id} {rec.description}\n{rec.seq}\n")

    print(f"  \n- {fasta_file}\n- {gtf_file}\n- {protein_file}")

#
gb_files = [
    r"rawdata\Pantoea\ncbi\pantoea.gb",
    r"rawdata\Burkholderia\ncbi\Burkholderia.gb"
]

for gb in gb_files:
    gb_to_fasta_gtf_protein(gb)
```

```
NameError: name 'SeqIO' is not defined
```

1.4 KEGG annotation (KEGG Web service)

1.5 Create and save gson dbfile

```
library(KEGGREST)
library(dplyr)
```

```
   'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(jsonlite)

process_ko_file <- function(ko_file) {
  ko_file <- normalizePath(ko_file)
  out_dir <- dirname(ko_file)

  # 1.   KO
  df <- read.table(ko_file,
                   header = FALSE, sep = "", stringsAsFactors = FALSE,
                   fill = TRUE, quote = "", comment.char = "")
  if (ncol(df) == 1) df$V2 <- NA_character_
  colnames(df) <- c("GeneID", "KO")

  # 1.1   KO
  df$KO <- trimws(df$KO)
  df <- df[!is.na(df$KO) & df$KO != "", ]
  df$KO <- toupper(df$KO)
  df <- df[grepl("^K\\d{5}$", df$KO), , drop = FALSE]

  #
  df <- distinct(df)

  # 2.   KO → Pathway
  kos <- unique(df$KO)
  batch_size <- 200
  link_list <- list()

  for (i in seq(1, length(kos), by = batch_size)) {
    batch <- kos[i:min(i + batch_size - 1, length(kos))]
    tmp <- keggLink("pathway", paste0("ko:", batch))
    link_list[[length(link_list) + 1]] <- tmp
    Sys.sleep(0.2)
  }
```

```r
links <- unlist(link_list)
ko2path <- data.frame(
  KO = sub("^ko:", "", names(links)),
  Pathway = sub("^path:", "", links),
  stringsAsFactors = FALSE
)

merged <- merge(df, ko2path, by = "KO")

# 3.   Pathway
path_info <- keggList("pathway", "ko")
path_df <- data.frame(
  Pathway = sub("^path:", "", names(path_info)),
  Name = as.vector(path_info),
  stringsAsFactors = FALSE
)

merged <- merge(merged, path_df, by = "Pathway")

# 4.   GSON
gson <- merged %>%
  group_by(Pathway, Name) %>%
  summarise(gene = list(unique(GeneID)), .groups = "drop") %>%
  transmute(id = Pathway, name = Name, gene = gene)

gson_file <- file.path(out_dir, "kegg_user.gson")
write_json(gson, gson_file, pretty = TRUE, auto_unbox = TRUE)

# 5.   GMT
gmt_lines <- merged %>%
  group_by(Pathway, Name) %>%
  summarise(genes = paste(unique(GeneID), collapse = "\t"), .groups = "drop") %>%
  mutate(line = paste(Pathway, Name, genes, sep = "\t")) %>%
  pull(line)

gmt_file <- file.path(out_dir, "kegg_user.gmt")
writeLines(gmt_lines, gmt_file)

cat("  ", ko_file, "\n",
    "  ", gson_file, "\n",
    "  ", gmt_file, "\n",
    "  KO  ", length(kos), "\n",
      "     ", nrow(merged), "\n\n")
}

#
ko_files <- c(
  "rawdata/Pantoea/kegg/Pantoea_ko.txt",
  "rawdata/Burkholderia/kegg/Burkholderia_ko.txt"
)

lapply(ko_files, process_ko_file)
```

```
  D:\ \      \  \rna-seq\rawdata\Pantoea\kegg\Pantoea_ko.txt
   D:/ /      /  /rna-seq/rawdata/Pantoea/kegg/kegg_user.gson
   D:/ /      /  /rna-seq/rawdata/Pantoea/kegg/kegg_user.gmt
   KO   2381
       2643

  D:\ \      \  \rna-seq\rawdata\Burkholderia\kegg\Burkholderia_ko.txt
   D:/ /      /  /rna-seq/rawdata/Burkholderia/kegg/kegg_user.gson
   D:/ /      /  /rna-seq/rawdata/Burkholderia/kegg/kegg_user.gmt
   KO   2246
       3772

[[1]]
NULL

[[2]]
NULL
```

2 RNA-seq data process

- Sample and design
- Sequencing results
- Mapping and quantification
- DESeq2 analysis
- DEG results (Excel/CSV table) ## Sample and design

2.1 Sequencing results

2.1.1

```
#!/usr/bin/env bash
set -Eeuo pipefail
shopt -s nullglob

################################################################################
# Command-line flags
################################################################################
# 1 = run mapping; 0 = skip mapping entirely
RUN_MAPPING=1

while [[ $# -gt 0 ]]; do
  case $1 in
    --skip-map)
      RUN_MAPPING=0
      shift
      ;;
    *)
      break
      ;;
  esac
done
```

```
################################################################################
# Configuration
################################################################################
WORKDIR="$HOME/ /     /  /TPL2025061860/CleanData"
OUTDIR="$WORKDIR/results"
IDXDIR="$OUTDIR/idx"
SAMD="$OUTDIR/sam"
BAMD="$OUTDIR/bam"
CNTD="$OUTDIR/counts"

BUR_FNA="$WORKDIR/burkholderia.fasta"
PAN_FNA="$WORKDIR/pantoea.fasta"
BUR_GTF="$WORKDIR/burkholderia.gtf"
PAN_GTF="$WORKDIR/pantoea.gtf"
COMBO_FNA="$WORKDIR/combo.fasta"
COMBO_GTF="$WORKDIR/combo.gtf"

BUR_IDX="$IDXDIR/burkholderia_idx"
PAN_IDX="$IDXDIR/pantoea_idx"
COM_IDX="$IDXDIR/combo_idx"

SAMPLES_B=(B_1 B_2 B_3 CK_B_1 CK_B_2 CK_B_3)
SAMPLES_P=(P_1 P_2 P_3 CK_P_1 CK_P_2 CK_P_3)
SAMPLES_PB=(PB_1 PB_2 PB_3)

R1_SUFFIX="_clean_R1.fq.gz"
R2_SUFFIX="_clean_R2.fq.gz"

THREADS="$(nproc)"
MAX_THREADS=64
(( THREADS > MAX_THREADS )) && THREADS="$MAX_THREADS"
MAPQ=10
STRAND=0

PB_B_PREFIX="ACSMXP_"
PB_P_PREFIX="ACSMXK_"

# force featureCounts to count this feature type (exon/CDS/transcript/gene)
FEATURE_TYPE="${FEATURE_TYPE:-exon}"

################################################################################
# Helper functions
```

```
################################################################################
log()   { echo "[$(date '+%H:%M:%S')] $*"; }
warn()  { echo "[$(date '+%H:%M:%S')] [WARN] $*" >&2; }
die()   { echo "[$(date '+%H:%M:%S')] [ERROR] $*" >&2; exit 1; }

need_file() {
  [[ -f "$1" ]] || die "Missing file: $1"
}

build_idx() {
  local fa="$1" prefix="$2"
  mkdir -p "$IDXDIR"
  local missing=0
  for i in {1..8}; do
    [[ -f "${prefix}.${i}.ht2" ]] || missing=1
  done
  if (( missing == 0 )); then
    log "[SKIP] HISAT2 index exists: $(basename "$prefix")"
  else
    log "[RUN ] hisat2-build: $fa → $prefix"
    hisat2-build "$fa" "$prefix"
    log "[DONE] index built: $prefix"
  fi
}

map_and_sort() {
  local sample="$1" idx="$2"
  local bam="$BAMD/${sample}.sorted.bam"
  local r1="$WORKDIR/${sample}${R1_SUFFIX}"
  local r2="$WORKDIR/${sample}${R2_SUFFIX}"
  local logf="$BAMD/${sample}.hisat2.log"

  if [[ -f "$bam" ]]; then
    log "[SKIP] BAM exists: $bam"
    return
  fi
  if [[ ! -f "$r1" || ! -f "$r2" ]]; then
    warn "Missing FASTQ for $sample: $r1 or $r2"
    return
  fi

  log "[RUN ] hisat2 → BAM: $sample"
  set -o pipefail
  hisat2 -p "$THREADS" -x "$idx" -1 "$r1" -2 "$r2" 2> "$logf" \
    | samtools view -@ "$THREADS" -bS - \
    | samtools sort  -@ "$THREADS" -o "$bam" -
  set +o pipefail

  samtools index -@ "$THREADS" "$bam"
  log "[DONE] sorted & indexed: $bam"
}

run_featureCounts() {
  local label="$1"; shift
  local gtf="$1"; shift
  local outdir="$CNTD/$label"
  mkdir -p "$outdir"

  local ftype="$FEATURE_TYPE"
  log "[INFO] $label: using feature type = $ftype"

  # Validate that the GTF's third column contains the feature
  if ! awk -F $'\t' -v t="$ftype" '$3==t { found=1; exit } END { exit !found }' "$gtf"; then
    die "GTF $gtf does not contain feature '$ftype'; recheck or set FEATURE_TYPE"
  fi

  local bams=()
  for x in "$@"; do
    [[ -f "$x" ]] && bams+=("$x")
  done
  (( ${#bams[@]} == 0 )) && { warn "$label: no BAMs found"; return; }

  local outfile="$outdir/counts_${label}.txt"
  log "[RUN ] featureCounts → $outfile"
  featureCounts \
    -F GTF \
    -t "$ftype" \
    -g "transcript_id" \
    -T "$THREADS" \
    -p -B -C \
    -Q "$MAPQ" \
    -s "$STRAND" \
    -a "$gtf" \
    -o "$outfile" \
    "${bams[@]}"
  log "[DONE] featureCounts output: $outfile"
}

split_pb_counts() {
  local pb_dir="$CNTD/PB_on_combo"
  local pb_counts="$pb_dir/counts_PB_on_combo.txt"
  [[ -f "$pb_counts" ]] || { warn "PB_on_combo counts not found"; return; }

  local out_b="$pb_dir/counts_PB_Burkholderia.txt"
  local out_p="$pb_dir/counts_PB_Pantoea.txt"

  awk -v p="$PB_B_PREFIX" 'NR==1 || $1~("^"p)' "$pb_counts" > "$out_b"
  awk -v p="$PB_P_PREFIX" 'NR==1 || $1~("^"p)' "$pb_counts" > "$out_p"
  log "[SPLIT] PB_on_combo → $(basename $out_b), $(basename $out_p)"
}

################################################################################
# Main workflow
################################################################################
log "Threads = $THREADS"
cd "$WORKDIR"

mkdir -p "$IDXDIR" "$SAMD" "$BAMD" "$CNTD"
need_file "$BUR_FNA"
need_file "$PAN_FNA"
need_file "$BUR_GTF"
need_file "$PAN_GTF"

build_idx "$BUR_FNA" "$BUR_IDX"
build_idx "$PAN_FNA" "$PAN_IDX"

if [[ ! -f "${COM_IDX}.1.ht2" ]]; then
  [[ -f "$COMBO_FNA" ]] || { cat "$BUR_FNA" "$PAN_FNA" > "$COMBO_FNA"; log "[GEN ] combo.fas
  [[ -f "$COMBO_GTF" ]] || { cat "$BUR_GTF" "$PAN_GTF" > "$COMBO_GTF"; log "[GEN ] combo.gtf
  build_idx "$COMBO_FNA" "$COM_IDX"
else
  log "[SKIP] combo index exists"
fi

if (( RUN_MAPPING )); then
  log "=== Mapping Burkholderia samples ==="
  for s in "${SAMPLES_B[@]}";  do map_and_sort "$s" "$BUR_IDX"; done

  log "=== Mapping Pantoea samples ==="
  for s in "${SAMPLES_P[@]}";  do map_and_sort "$s" "$PAN_IDX"; done

  log "=== Mapping PB_on_combo samples ==="
  for s in "${SAMPLES_PB[@]}"; do map_and_sort "$s" "$COM_IDX"; done
else
  log "[SKIP] mapping stage (--skip-map)"
fi

log "=== Running featureCounts ==="
run_featureCounts burkholderia "$BUR_GTF"    "$BAMD"/B_*.sorted.bam  "$BAMD"/CK_B_*.sorted.ba
run_featureCounts pantoea      "$PAN_GTF"   "$BAMD"/P_*.sorted.bam  "$BAMD"/CK_P_*.sorted.bar
run_featureCounts PB_on_combo  "$COMBO_GTF" "$BAMD"/PB_*.sorted.bam

split_pb_counts

log "All done."
```

```
[15:05:49] Threads = 8
/usr/bin/bash:  170  cd: /c/Users/99374/Documents/ /     /  /TPL2025061860/CleanData: No suc
```

2.2 DESeq2 analysis

```
library(DESeq2)
```

```
     S4Vectors

     stats4

     BiocGenerics


   'BiocGenerics'
```

```
The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs


The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min



   'S4Vectors'


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

    findMatches


The following objects are masked from 'package:base':

    expand.grid, I, unname


     IRanges



   'IRanges'


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

    windows


     GenomicRanges


     GenomeInfoDb


     SummarizedExperiment
     MatrixGenerics


     matrixStats



   'MatrixGenerics'


The following objects are masked from 'package:matrixStats':

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

     Biobase


Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.



   'Biobase'


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

    rowMedians
```

```
The following objects are masked from 'package:matrixStats':

    anyMissing, rowMedians
```

```
library(openxlsx)

#
out_dir <- "D:/ /      /  /rna-seq/rawdata/DEG_result"
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)

run_deseq_and_export <- function(count_file, pattern_treat, pattern_ctrl,
                                  treat_label, ctrl_label, prefix) {
  # 1.
  counts <- read.table(count_file,
                       header = TRUE, comment.char = "#", row.names = 1,
                       sep = "\t", check.names = FALSE)

  # 2.    5
  counts <- counts[, 6:ncol(counts)]

  # 3.
  sample_info <- data.frame(
    row.names = colnames(counts),
    condition = factor(ifelse(grepl(pattern_treat, colnames(counts)),
                              treat_label, ctrl_label),
                       levels = c(ctrl_label, treat_label))
  )

  # 4.   DESeqDataSet
  dds <- DESeqDataSetFromMatrix(countData = counts,
                                colData = sample_info,
                                design = ~ condition)

  # 5.
  keep <- rowSums(counts(dds)) >= 10
  dds <- dds[keep, ]

  # 6.   DESeq
  dds <- DESeq(dds)

  # 7.   dds
  saveRDS(dds, file = file.path(out_dir, paste0("dds_", prefix, ".rds")))
```

```r
  # 8.
  res <- results(dds, contrast = c("condition", treat_label, ctrl_label))
  res_df <- as.data.frame(res)
  res_df$gene_id <- rownames(res_df)

  # 9.   CSV
  write.csv(res_df, file.path(out_dir, paste0("DESeq2_results_", prefix, ".csv")),
            row.names = FALSE)

  # 10.   Excel
  wb <- createWorkbook()
  addWorksheet(wb, prefix)
  writeData(wb, prefix, res_df)
  saveWorkbook(wb, file.path(out_dir, paste0("DESeq2_results_", prefix, ".xlsx")),
               overwrite = TRUE)

  message("  ", prefix, ":    DESeq       ", out_dir)
}

# Pantoea
run_deseq_and_export(
  count_file = "D:/ /      /  /work/counts/pantoea/counts_pantoea.txt",
  pattern_treat = "^P_", pattern_ctrl = "^CK_P",
  treat_label = "P", ctrl_label = "CK_P",
  prefix = "pantoea"
)
```

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

  pantoea:    DESeq       D:/ /      /  /rna-seq/rawdata/DEG_result
```

```
# Burkholderia
run_deseq_and_export(
  count_file = "D:/ /     /  /work/counts/burkholderia/counts_burkholderia.txt",
  pattern_treat = "^B_", pattern_ctrl = "^CK_B",
  treat_label = "B", ctrl_label = "CK_B",
  prefix = "burkholderia"
)
```

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

  burkholderia:    DESeq      D:/ /     /  /rna-seq/rawdata/DEG_result
```

2.3 DEG results (Excel/CSV table)

```
##   DESeq2    rawdata/DEG_result     Excel
library(DESeq2)
library(openxlsx)

#
out_dir <- "D:/ /     /  /rna-seq/rawdata/DEG_result"
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)

#
dds_p <- readRDS("dds_p.rds")
```

```
Warning in gzfile(file, "rb"):      'dds_p.rds'    'No such
file or directory'
```

```
Error in gzfile(file, "rb"):
```

```r
dds_b <- readRDS("dds_b.rds")
```

```
Warning in gzfile(file, "rb"):     'dds_b.rds'   'No such
file or directory'
```

```
Error in gzfile(file, "rb"):
```

```r
#
res_p <- results(dds_p, contrast = c("condition", "P", "CK_P"))
```

```
Error:    'dds_p'
```

```r
res_b <- results(dds_b, contrast = c("condition", "B", "CK_B"))
```

```
Error:    'dds_b'
```

```r
#   data.frame   gene_id
res_p_df <- as.data.frame(res_p)
```

```
Error in h(simpleError(msg, call)):  'as.data.frame'     'x'   :    'res_p'
```

```r
res_p_df$gene_id <- rownames(res_p_df)
```

```
Error in h(simpleError(msg, call)):  'rownames'     'x'   :    'res_p_df'
```

```r
res_b_df <- as.data.frame(res_b)
```

```
Error in h(simpleError(msg, call)):  'as.data.frame'     'x'   :    'res_b'
```

```r
res_b_df$gene_id <- rownames(res_b_df)
```

```
Error in h(simpleError(msg, call)):  'rownames'     'x'   :    'res_b_df'
```

```
##   CSV
write.csv(res_p_df, file.path(out_dir, "DESeq2_results_pantoea.csv"), row.names = FALSE)
```

```
Error in eval(expr, p):    'res_p_df'
```

```
write.csv(res_b_df, file.path(out_dir, "DESeq2_results_burkholderia.csv"), row.names = FALSE
```

```
Error in eval(expr, p):    'res_b_df'
```

```
##   Excel
wb_p <- createWorkbook()
addWorksheet(wb_p, "Pantoea")
writeData(wb_p, "Pantoea", res_p_df)
```

```
Error:    'res_p_df'
```

```
saveWorkbook(wb_p, file.path(out_dir, "DESeq2_results_pantoea.xlsx"), overwrite = TRUE)

wb_b <- createWorkbook()
addWorksheet(wb_b, "Burkholderia")
writeData(wb_b, "Burkholderia", res_b_df)
```

```
Error:    'res_b_df'
```

```
saveWorkbook(wb_b, file.path(out_dir, "DESeq2_results_burkholderia.xlsx"), overwrite = TRUE)

message("           Excel ",
        "\n- ", file.path(out_dir, "DESeq2_results_pantoea.csv"),
        "\n- ", file.path(out_dir, "DESeq2_results_burkholderia.csv"),
        "\n- ", file.path(out_dir, "DESeq2_results_pantoea.xlsx"),
        "\n- ", file.path(out_dir, "DESeq2_results_burkholderia.xlsx"))
```

```
            Excel
- D:/ /      /  /rna-seq/rawdata/DEG_result/DESeq2_results_pantoea.csv
- D:/ /      /  /rna-seq/rawdata/DEG_result/DESeq2_results_burkholderia.csv
- D:/ /      /  /rna-seq/rawdata/DEG_result/DESeq2_results_pantoea.xlsx
- D:/ /      /  /rna-seq/rawdata/DEG_result/DESeq2_results_burkholderia.xlsx
```

3 Mechanism analysis

- combine annotation and deg
- analysis pathway enrichment
- perform GSEA
- data visulization

3.1 combine annotation and deg

```r
##   DESeq2    KEGG       Pantoea & Burkholderia   .tXX
library(jsonlite)
library(DESeq2)
```

```
     S4Vectors

     stats4

     BiocGenerics


   'BiocGenerics'

The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs

The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min
   'S4Vectors'

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

    findMatches

The following objects are masked from 'package:base':

    expand.grid, I, unname

     IRanges


   'IRanges'

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

    windows

     GenomicRanges

     GenomeInfoDb

     SummarizedExperiment

     MatrixGenerics

     matrixStats


   'MatrixGenerics'
```

```
The following objects are masked from 'package:matrixStats':

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

     Biobase

Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.


   'Biobase'

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

    rowMedians

The following objects are masked from 'package:matrixStats':

    anyMissing, rowMedians
```

```r
# 1.   gson
pantoea_gson <- fromJSON("rawdata/Pantoea/kegg/kegg_user.gson")
burkholderia_gson <- fromJSON("rawdata/Burkholderia/kegg/kegg_user.gson")
```

```r
make_gene_sets <- function(df) {
  #   .tXX
  df$gene <- lapply(df$gene, function(g) sub("\\.t\\d+$", "", g))
  sets <- setNames(df$gene, df$id)
  names(sets) <- paste(df$id, df$name, sep = ": ")
  return(sets)
}

pantoea_sets <- make_gene_sets(pantoea_gson)
burkholderia_sets <- make_gene_sets(burkholderia_gson)

# 2.   DESeq2
dds_p <- readRDS("rawdata/DEG_result/dds_pantoea.rds")
dds_b <- readRDS("rawdata/DEG_result/dds_burkholderia.rds")

res_p <- results(dds_p)
res_b <- results(dds_b)

# 3.   geneList     .tXX
geneList_p <- res_p$log2FoldChange
names(geneList_p) <- sub("\\.t\\d+$", "", rownames(res_p))
geneList_p <- sort(geneList_p, decreasing = TRUE)

geneList_b <- res_b$log2FoldChange
names(geneList_b) <- sub("\\.t\\d+$", "", rownames(res_b))
geneList_b <- sort(geneList_b, decreasing = TRUE)

# 4.        /
check_name_match <- function(geneList, gene_sets, label, n_show = 10) {
  genes_in_list <- names(geneList)
  genes_in_sets <- unique(unlist(gene_sets, use.names = FALSE))
  common_genes <- intersect(genes_in_list, genes_in_sets)
  unmatched_genes <- setdiff(genes_in_list, genes_in_sets)

  cat("\n====", label, "====\n")
  cat("geneList   ", length(genes_in_list), "\n")
  cat("     ", length(genes_in_sets), "\n")
  cat("    ", length(common_genes), "\n")
  cat("     geneList ",
      round(length(common_genes) / length(genes_in_list) * 100, 2), "%\n")

  cat("\n ", n_show, "     :\n")
  print(head(common_genes, n_show))

  cat("\n ", n_show, "      :\n")
  print(head(unmatched_genes, n_show))

  invisible(list(common = common_genes, unmatched = unmatched_genes))
}

# 5.
match_p <- check_name_match(geneList_p, pantoea_sets, "Pantoea", n_show = 10)
```

```
==== Pantoea ====
geneList    4564
      887
     884
     geneList  19.37 %

 10       :
 [1] "ACSMXK_13025" "ACSMXK_13030" "ACSMXK_00125" "ACSMXK_10330" "ACSMXK_00140"
 [6] "ACSMXK_13040" "ACSMXK_13035" "ACSMXK_07470" "ACSMXK_07475" "ACSMXK_00135"

 10        :
 [1] "ACSMXK_21765" "ACSMXK_21775" "ACSMXK_21770" "ACSMXK_21780" "ACSMXK_16465"
 [6] "ACSMXK_10325" "ACSMXK_21790" "ACSMXK_08620" "ACSMXK_21785" "ACSMXK_21795"
```

```r
match_b <- check_name_match(geneList_b, burkholderia_sets, "Burkholderia", n_show = 10)
```

```
==== Burkholderia ====
geneList    7304
      1101
     1100
     geneList  15.06 %

 10       :
 [1] "ACSMXP_35140" "ACSMXP_22575" "ACSMXP_34115" "ACSMXP_00960" "ACSMXP_18535"
 [6] "ACSMXP_24400" "ACSMXP_34225" "ACSMXP_22570" "ACSMXP_34175" "ACSMXP_29725"

 10        :
 [1] "ACSMXP_20900" "ACSMXP_20885" "ACSMXP_32285" "ACSMXP_20895" "ACSMXP_35135"
```

```
[6] "ACSMXP_20880" "ACSMXP_35150" "ACSMXP_35130" "ACSMXP_32320" "ACSMXP_32300"
```