

Bioinformatics · Practical notes

Bulk Transcriptomics

From Raw SRA Reads to Differential Expression,
Functional Enrichment, Networks, and Figures

A practical, reference-based bulk RNA-seq pipeline built for reproducibility. It covers data retrieval, quality control, a choice of trimmer, *optional* ribosomal-RNA depletion and a contamination screen, alignment and gene-level quantification (with alternative aligners and counters, or an alignment-free route), differential expression with DESeq2 alongside edgeR and limma-voom cross-checks and an equivalence test, functional enrichment for model *and* non-model organisms (GO, KEGG, GSEA, g:Profiler, and custom gene sets), protein-interaction networks, and the figures most commonly used to communicate results. Three alternative entry points (a count matrix, a differential-expression table, or a GEO microarray series) join the same downstream analysis. Each phase explains its decisions, inputs, outputs and common failure modes. The companion provides readable scripts and environment specifications; verification status and execution limits are stated explicitly.

Code and environments:
github.com/tunabirgun/bioinformatics-notes

Tuna Birgun
github.com/tunabirgun

Original note · 2 July 2026

Contents

1 Overview & scope **3**
1.1 Decisions that precede tool selection . . . 3

2 How to use this worksheet **4**

3 Pipeline at a glance **5**

4 Software & packages **5**

5 Environment setup **7**
5.1 Operating system: Linux, WSL2, or macOS . . . 7
5.2 Install Miniconda and configure channels . . . 8
5.3 Create a dedicated environment and install the tools . . . 8
5.4 Create the project directory layout . . . 9
5.5 Companion execution contract . . . 9

6 From raw reads to a count matrix **10**
6.1 Find and fetch a dataset from the terminal . . . 10
6.2 Download raw data (SRA Toolkit) . . . 11
6.3 Initial quality control (FastQC to MultiQC) . . . 12
6.4 Contamination screen (FastQ Screen) Optional . . . 12
6.5 Adapter & quality trimming (fastp) . . . 13
6.6 Quality control after trimming . . . 13
6.7 Alternative trimmers: Trim Galore and Trimmomatic . . . 14
6.8 rRNA removal (SortMeRNA) . . . 15
6.9 Reference-free rRNA removal (RiboDetector) . . . 16
6.10 Re-pair reads (BBMap `repair.sh`) . . . 16
6.11 Build the genome index (STAR) . . . 17
6.12 Alignment (STAR to sorted BAM) . . . 18
6.13 Alternative aligner: HISAT2 . . . 18
6.14 Determine library strandedness . . . 20
6.15 Gene-level quantification (featureCounts) . . . 20
6.16 QC summary: mapping and assignment rates . . . 21
6.17 Post-alignment RNA QC (RSeQC) . . . 22
6.18 Alternative counter: htseq-count . . . 22
6.19 Alternative counter: STAR gene counts (`--quantMode GeneCounts`) . . . 23
6.20 Alignment-free quantification (Salmon) . . . 24
6.21 Single-end reads . . . 25

7 Alternative entry points **25**
7.1 Starting from a count matrix . . . 26
7.2 Starting from a differential-expression results table . . . 26
7.3 GEO microarray series with limma . . . 27

8 Differential expression with DESeq2 **27**
8.1 Import counts and build sample metadata . . . 27
8.2 Build the dataset, pre-filter, set the reference level . . . 28
8.3 Run DESeq2 and extract results . . . 29
8.4 Equivalence testing: effects within a meaningful margin Optional . . . 30
8.5 Transform counts and export . . . 31
8.6 Annotate genes (AnnotationDbi and biomaRt) . . . 31
8.7 Inspect genes of interest . . . 32
8.8 Cross-checking DE engines: edgeR and limma-voom . . . 32

9 Functional enrichment **34**
9.1 Map gene IDs and define the gene universe . . . 34
9.2 GO and KEGG over-representation (ORA) . . . 35
9.3 GSEA on the ranked gene list . . . 36

9.4 Enrichment without an OrgDb (non-model organisms) . . . 36
9.5 Bring your own gene sets (any organism) . . . 38
9.6 Visualising enrichment . . . 39
9.7 Other gene-set resources and length bias . . . 39
9.8 Per-sample gene-set activity (GSVA) Optional . . . 40

10 Protein interaction networks (STRING) 40

11 Figures for transcriptomics 42
11.1 PCA (sample structure) . . . 42
11.2 UMAP (optional, many-sample structure) . . . 43
11.3 Sample-distance heatmap & dendrogram . . . 43
11.4 MA plot (effect size vs expression) . . . 44
11.5 Volcano plot (significance vs effect size) . . . 44
11.6 Heatmap of top differentially-expressed genes . . . 46

12 Supplementary 47
12.1 Beyond this worksheet: *de novo* assembly and variant calling . . . 47
12.2 Extended QC and companion tools . . . 47
12.3 Looping & batch processing . . . 48
12.4 Threads, memory & critical parameters (quick reference) . . . 49
12.5 Reproducibility . . . 49
12.6 Troubleshooting . . . 50
12.7 Sample correlations and sensitivity checks . . . 50
12.8 Cross-study synthesis . . . 51
12.9 Annotation transfer and unavailable results . . . 51

References 52

1 Overview & scope

Bulk RNA-seq measures RNA pooled across cells in each biological sample. For example, comparing independently treated and control animals asks whether relative gene abundance differs between groups; a tissue-level difference may reflect cell composition, expression within cells, or both. Define the independent biological units before collecting reads or choosing software.

This worksheet follows a core route with explicit alternatives. The shell stages (Sections 5–6) take you from raw sequencing runs to a *count matrix*; the R stages (Sections 8–11) take that matrix to differential-expression tables, enriched pathways, and figures. Optional stages carry an Optional badge. Choose one trimmer, one alignment/quantification route, one counter, and one primary DE engine; do not run every numbered script in sequence.

Practical note

Running example. Throughout, we assume **paired-end Illumina** reads, a two-group design (`treated` vs `control`), and that a **reference genome (FASTA) and annotation (GTF)** are available for your organism. Commands assume a `bash` shell on Linux. Replace anything in `<angle brackets>` with your own values.

What this pipeline is. A *reference-based, gene-level* workflow: reads are aligned to a genome and summarised per gene. This is the standard route when a good reference exists and gene-level differential expression is the goal.

What it is not. It does not perform transcript/isoform-level quantification or *de novo* assembly. If you need transcript-level estimates or a route without genome alignment, use Salmon or kallisto and import with `tximport` [22]; the DESeq2, enrichment, and figure sections below then apply to the gene-level handoff, with the input-specific adapters described in Section 7. For an organism with no reference genome, assemble transcripts *de novo* instead (Section 12.1).

Where this sits. The stage order here follows the consensus of widely-used RNA-seq best-practice surveys and community pipelines [24, 25, 26, 27]: experimental design and library choice, quality control, read trimming, alignment (or alignment-free quantification), gene-level counting, normalisation and differential expression, then functional interpretation. The tool choices below are common, well-cited defaults; equivalent alternatives are flagged where the choice matters (Sections 6.20, 9.7).

Entry points. The default route starts from raw reads, but the analysis can join later: a ready count matrix, a pre-computed differential-expression table, or a GEO microarray series each enters downstream of alignment (Section 7).

Practical note

Design first. The statistics are only as good as the design. Use biological replicates, not technical ones, and plan replication from the expected variability and effect size; three per group is a teaching minimum, not a power guarantee. Beyond a point, biological replication contributes more to differential-expression power than additional sequencing depth. Read-depth requirements depend on organism, library complexity, expression level and the question; use study-specific power and saturation evidence rather than a universal depth target. Match the library to the biology: poly-A selection enriches mature mRNA, whereas total-RNA with rRNA depletion also retains lncRNA, pre-mRNA, and histone mRNAs and can be better suited to degraded or FFPE input. [24, 27]

1.1 Decisions that precede tool selection

A gene-level count model does not separate cell-composition changes from within-cell expression changes. Define the biological unit, target population and comparison before choosing software. Randomize samples across extraction, library preparation and sequencing batches; block known sources of variation and retain the assignment record. Sequencing lanes and repeated measurements from one donor do not create

independent biological replicates. For paired donors, preserve the pairing in the model rather than applying an unpaired test. These design principles remain central in the recent experimental-design Perspective [77].

The 2016 best-practice survey and the 2019 RNA-seq review remain foundational [24, 25]. A 2026 experimental-planning review extends the discussion of RNA classes, sample quality and platform choice [76]; it is not a benchmark proving one aligner or DE engine universally superior. Choose the library preparation for the RNA class and material quality, then document read layout, strandedness, reference assembly and annotation release. Total-RNA and poly(A)-selected libraries sample different RNA populations. Their differences are not removed merely by normalizing library sizes.

Choose the estimand. Gene-level differential expression asks whether gene abundance differs for a declared contrast. Differential transcript usage, splice-event changes and transcript discovery require other response variables, annotations and methods. A successful gene-level workflow does not validate those separate analyses. Likewise, an adjusted p-value is not an effect size or the probability that a particular gene is false. State the tested null hypothesis, effect direction and multiplicity family beside the result.

The DE section distinguishes an ordinary zero-effect test, an exploratory observed-effect screen and a threshold hypothesis (Section 8.3); engine-specific input contracts are discussed beside the alternatives (Section 8.8).

2 How to use this worksheet

Code appears in two kinds of boxes: **SHELL** boxes (command line) and **R** boxes. R code uses sequential steps so every object is easy to inspect and rerun. The body shows commands for *one* sample for clarity; Section 12.3 shows how to wrap each command in a loop and parameterise threads, references, and paths. Readable companion scripts, organised by phase, and environment specifications are in the companion repository `github.com/tunabirgun/bioinformatics-notes`.

Watch out for

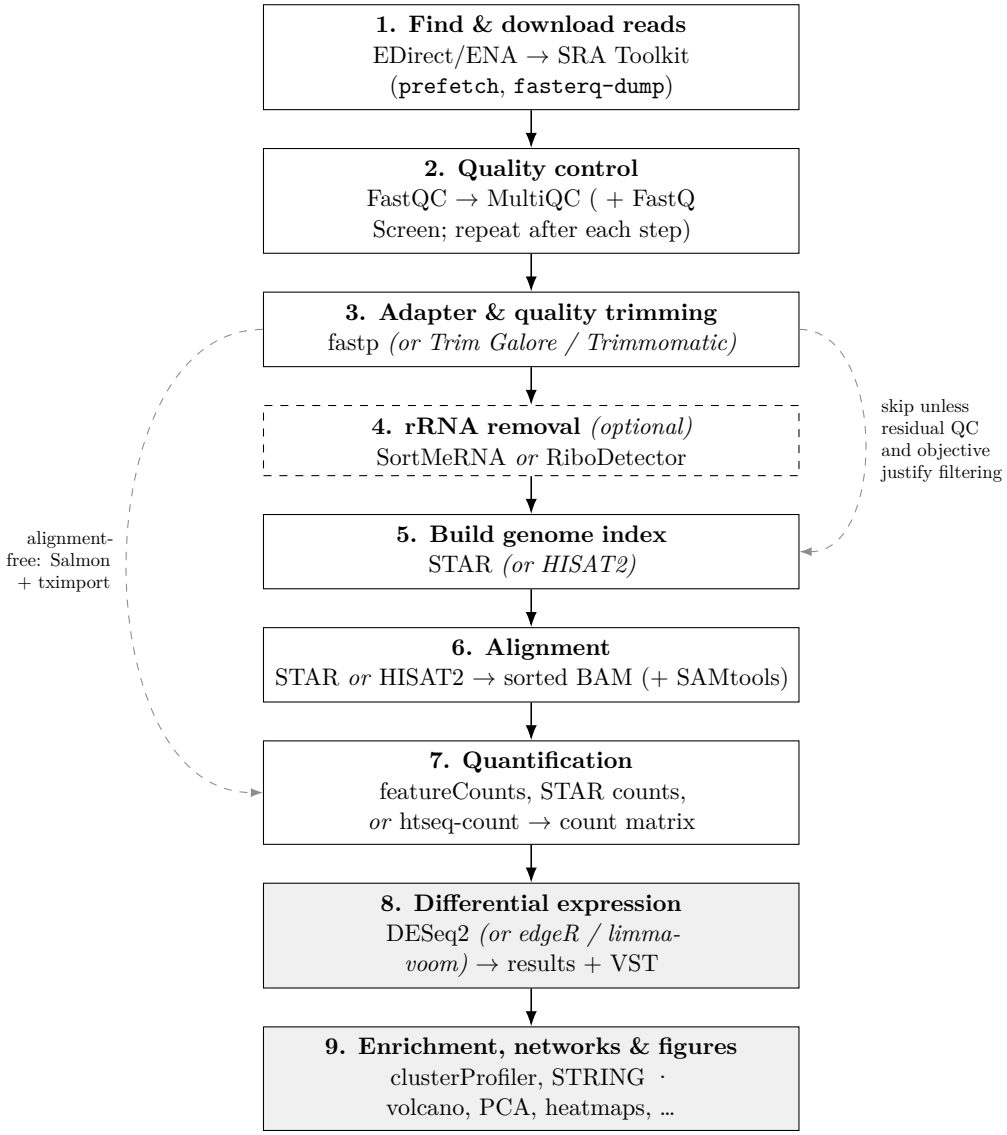
Verification status. The revised scripts have been syntax-checked and their input contracts tested with small synthetic fixtures. A fixed-seed synthetic count-matrix route with DESeq2/apeglm and an identifiable batch covariate passed checks of configured design, original inferential columns, missingness and figure-input membership. This is a bounded execution check, not biological validation. The complete analysis stack, real-data reproduction and generated figures have not been executed for this edition. Version constraints are supplied; a solved conda lock and a tested `renv.lock` are not bundled. Treat optional neighboring workflows as illustrations until their declared prerequisites are met.

Practical note

New to the command line? A **SHELL** box is typed into a terminal: Terminal on macOS, the Ubuntu/WSL2 shell on Windows (Section 5.1), or any Linux terminal. An **R** box is run in R or RStudio. The terminal shows a *prompt* (often `$` or `(rnaseq) $`); paste one command, press Enter, and let it finish before the next. A trailing `\` only continues one command across several lines, so paste the whole block together. Lines that begin with `#` are comments and do nothing. Replace anything in `<angle brackets>` with your own value, and run `conda activate rnaseq` first so the tools are on your PATH.

The Optional badge marks a stage you may skip; ✓/✗ mark recommended/avoid choices.

3 Pipeline at a glance



4 Software & packages

The command-line tools are installed with **conda** (Miniconda) from the **Bioconda** and **conda-forge** channels [23]; R packages come from CRAN and Bioconductor. The versions below are those this worksheet was written against; record the versions actually installed in your environment (Section 12.5). The table describes the intended tool set, not a validated multi-platform solve. Full citations are in the References.

Command-line tools

Tool	Ver.	Purpose (one line)
SRA Toolkit (prefetch, fasterq-dump)	—	Downloads raw runs from NCBI SRA and converts them to FASTQ. [1]
Entrez Direct (esearch, efetch)	—	Queries NCBI SRA/GEO from the shell; resolves accessions and metadata. [28]
FastQC	0.12.1	Per-sample read-quality metrics (base quality, adapters, duplication). [2]
MultiQC	1.30	Aggregates QC logs from many tools/samples into one HTML report. [3]
fastp	1.0.1	Adapter trimming, quality filtering and read QC in one fast pass. [4]
Trim Galore **Optional**	—	Cutadapt wrapper: adapter/quality trimming with automatic adapter detection. [56]
Trimmomatic **Optional**	0.39	Adapter clipping and sliding-window quality trimming against an explicit adapter FASTA. [57]
FastQ Screen **Optional**	—	Screens a read subset against a genome panel to flag contamination and species mix. [58]
SortMeRNA **Optional**	4.3.4	Filters ribosomal-RNA reads against curated rRNA databases. [5]
RiboDetector **Optional**	—	Reference-free rRNA read removal with a CPU neural-network classifier. [59]
BBMap (repair.sh) **Optional**	38.96	Re-synchronises paired reads after rRNA filtering. [6]
STAR	2.7.11b	Splice-aware aligner: maps RNA-seq reads to a reference genome. [7]
Salmon **Optional**	—	Alignment-free transcript quantification; fast and low-memory. [40]
SAMtools	1.22.1	Sorts, indexes and summarises SAM/BAM alignment files. [8]
RSeQC (infer_experiment) **Optional**	—	RNA-seq QC: strandedness, read distribution, gene-body coverage. [34]
UCSC (gtfToGenePred, genePredToBed) **Optional**	—	Converts a GTF to the 12-column BED gene model RSeQC needs. [68]
Cytoscape **Optional**	—	Desktop viewer for the exported protein-interaction network (GraphML; from cytoscape.org). [67]
Subread (featureCounts)	2.1.1	Assigns aligned reads to genes; builds the count matrix. [9]
Miniconda + Bioconda/conda-forge	—	Package manager and channels used to install the tools. [23]

R / Bioconductor packages

Package	Purpose (one line)
`DESeq2`	Model-based differential expression from raw counts (negative-binomial GLMs). [10]
`apeglm`	Shrinkage estimator for $\log_2$ fold changes, used by `lfcShrink`. [11]
`edgeR`	Negative-binomial GLMs with quasi-likelihood tests; alternative DE engine. [42]
`limma`	Moderated-t linear models; drives limma-voom (RNA-seq) and the microarray route. [60]
`clusterProfiler`	Over-representation and GSEA enrichment for GO, KEGG and custom gene sets. [12]
`enrichplot`	Visualises enrichment results (dotplot, cnetplot, emapplot, ridgeplot). [13]
`fgsea`	Fast pre-ranked gene-set enrichment analysis (GSEA). [45]
`msigdbr`	MSigDB Hallmark / curated / ontology gene sets as a tidy table. [44]
`gprofiler2`	OrgDb-free GO/KEGG/Reactome enrichment through the g:Profiler service. [63]
`GSVA`	Per-sample gene-set activity scores from an expression matrix. [48]
`STRINGdb`	Queries STRING and maps genes to protein-protein interactions. [64]
`igraph`	Graph construction, community detection and centrality measures. [65]
`ggraph`	Grammar-of-graphics layouts for plotting networks. [66]
`org.Hs.eg.db`	Genome-wide annotation and gene-ID mapping (via AnnotationDbi). [14]
`biomaRt`	Online Ensembl annotation and ID mapping: biotype, coordinates, lengths. [33]
`GEOquery`	Downloads GEO series expression matrices and sample metadata into R. [30]
`affy` Optional	Reads Affymetrix CEL files and RMA-normalises them. [61]
`tximport`	Imports Salmon/kallisto transcript estimates to gene level for DESeq2. [22]
`ggplot2`	Grammar-of-graphics system underlying all custom figures. [15]
`ggrepel`	Repels overlapping text labels (gene names) in scatter plots. [16]
`ggtext`	Renders Markdown/HTML (italics, subscripts) in plot text. [17]
`EnhancedVolcano`	Produces a styled volcano plot in one call. [18]
`pheatmap`	Annotated, clustered heatmaps with row scaling. [19]
`ComplexHeatmap`	Highly customisable heatmaps with multiple annotation tracks. [20]
`uwot`	UMAP: non-linear dimensionality reduction for sample structure. [21]

5 Environment setup

5.1 Operating system: Linux, WSL2, or macOS

The command-line stages assume a Unix shell, but they do not need a dedicated Linux machine. The intended platforms below require separate environment solves; availability of one package does not establish availability of the complete stack.

- **Linux** (native, or a lab server/HPC): the reference environment. Check the required package versions for the machine architecture.
- **Windows via WSL2**: install a distribution (`wsl --install -d Ubuntu`) and keep the project inside the Linux home (`~/`), not under `/mnt/c/`, because cross-filesystem I/O on `/mnt/c` is slow. Bioconda tools then behave as on native Linux. WSL2 allocates 50% of host RAM by default; raise it with `memory=` in `.wslconfig` if a mammalian STAR index needs more.
- **macOS** (Intel or Apple Silicon): check native R/Bioconductor and command-line package availability for the exact architecture and version. If a required tool is available only for x86-64, use an isolated compatible environment or a supported Linux host; do not mix architectures in one environment.

Shell

```
# Apple Silicon: most tools install natively (see the main env in the next subsection).
# For a tool with no arm64 build (e.g. SortMeRNA), make a small emulated x86-64 env for it:
CONDA_SUBDIR=osx-64 conda create -n rrna_x86 -y sortmerna
conda activate rrna_x86
conda config --env --set subdir osx-64        # keep this env on osx-64 for later installs
```

Watch out for

The binding constraint is usually memory rather than the operating system: STAR's human index needs ∼30 GB RAM. A primary-assembly FASTA and `--genomeSAsparseD 2` can reduce index size, but they do not guarantee that alignment and coordinate sorting fit 24 GB. Sorting allocates additional memory. Choose the assembly appropriate to the scientific question, measure peak use, and leave headroom for the operating system. [72] Otherwise run STAR on a shared server, or use an alignment-free quantifier (Section 6.20), which needs only a few GB. The emulated osx-64 route also requires a compatible Rosetta installation; verify its support on the actual macOS release.

5.2 Install Miniconda and configure channels

Conda provides isolated environments; explicit constraints and a solved export record the versions selected. Configure the channels *once*; channel order and `strict` priority decide which builds you get.

Shell

```
# 1. Download and install Miniconda (accept the licence, allow conda init)
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda.sh
bash ~/miniconda.sh
source ~/.bashrc
conda --version                        # verify it works

# 2. Add channels in the recommended order, then enforce strict priority
conda config --add channels bioconda
conda config --add channels conda-forge
conda config --set channel_priority strict
```

Watch out for

Add `bioconda` *before* `conda-forge` (the channel added last has the highest priority). With `channel_priority strict`, conda refuses incompatible mixed-channel combinations that often break at runtime. This is the documented Bioconda priority order. The companion setup uses the channel order in its YAML without changing global settings. [70]

5.3 Create a dedicated environment and install the tools

Install the core route first; add only the alternatives you choose. The companion provides a core YAML and a separate optional-tool inventory. Neither is a solved lock.

Shell

```
conda create -n rnaseq -y \
  sra-tools entrez-direct fastqc fastp star samtools subread
conda activate rnaseq

# MultiQC is happiest installed with pip inside the same environment
pip install multiqc

# RiboDetector pulls PyTorch (~2 GB); install it only if you use the reference-free rRNA
    ↪ route
conda install -n rnaseq -y ribodetector
```

Practical note

`multiqc` can clash with strict conda Python pins; installing it with pip inside the active environment can reduce these conflicts. Confirm each tool with `<tool> -version` after installation.

5.4 Create the project directory layout

A consistent layout keeps inputs, intermediates, and results separate and makes loops predictable.

Shell

```
mkdir -p rnaseq_project/{raw_data,trimmed,rrna_clean,ref,mapped,counts,results,config} \
         rnaseq_project/qc/{pre_trim,post_trim,post_rrna,fastq_screen,rseqc} \
         rnaseq_project/results/ppi
cd rnaseq_project
```

Project directories

```
rnaseq_project/
|-- raw_data/            # SRA downloads + FASTQ (.fastq.gz)
|-- qc/
|   |-- pre_trim/        # FastQC before trimming
|   |-- post_trim/       # FastQC after trimming
|   |-- post_rrna/       # OPTIONAL: FastQC after rRNA removal
|   |-- fastq_screen/    # OPTIONAL: contamination-screen reports
|   `-- rseqc/           # OPTIONAL: read distribution + gene-body coverage
|-- trimmed/             # trimmer output (clean reads)
|-- rrna_clean/          # OPTIONAL: SortMeRNA or RiboDetector output
|-- ref/                 # genome FASTA + GTF + STAR index (+ genes.bed)
|-- mapped/              # STAR sorted BAM files (+ .bai)
|-- counts/              # featureCounts matrix + summary
`-- results/             # DESeq2 tables, enrichment, figures, ppi/
```

5.5 Companion execution contract

Keep the code checkout separate from the analysis directory. Set `BULK_CODE` to the companion folder, run the directory setup once, then edit the copied `config/samples.tsv` and `config/analysis.R`. Each sample ID must identify one biological analysis unit; combine technical lanes before inference or document their aggregation. The metadata file supplies the condition and any design covariates. Counts are matched to metadata by sample ID, never by column position.

Shell

```
export BULK_CODE=/path/to/main_scripts/bulk_transcriptomics
export BULK_PROJECT="$HOME/rnaseq_project"
bash "$BULK_CODE/scripts/00_setup/01_project_dirs.sh"
cd "$BULK_PROJECT"
# Edit config/samples.tsv and config/analysis.R before analysis.
# Set after inspecting protocol and strand evidence:
export BULK_STRANDED=2
Rscript "$BULK_CODE/scripts/05_differential_expression/01_deseq2.R"
```

The final command starts from an existing featureCounts table; it does not execute the earlier shell phases. Choose the alternative entry script when starting from another format. The supplied R helper loads a companion-local renv project when its lock exists, so the code library can be used while the working directory contains the data. Install and validate that library before running an R phase. Do not treat the environment specification or a generated lock as proof of successful reproduction.

6 From raw reads to a count matrix

6.1 Find and fetch a dataset from the terminal

Public RNA-seq lives in three linked archives: NCBI GEO (series `GSE`, samples `GSM`), NCBI SRA (studies `SRP`, runs `SRR`), and the EMBL-EBI ENA, which mirrors SRA and serves ready-made FASTQ. All three are queryable from the shell, which keeps data retrieval scriptable and reproducible.

Query metadata with NCBI EDirect. EDirect drives the NCBI E-utilities from the command line: `esearch` runs the query, `efetch`/`esummary` pull records, `xtract` turns the XML into columns [28].

Shell

```
conda install -c bioconda entrez-direct          # installs esearch, efetch, xtract, ...

# resolve a GEO series to its SRA run accessions + a metadata table (RunInfo CSV)
esearch -db sra -query "GSE183947" | efetch -format runinfo > runinfo.csv
python "$BULK_CODE/scripts/01_data_retrieval/03_parse_runinfo.py" runinfo.csv SRR_Acc_List.
    ↪ txt

# check layout/strategy/platform before committing disk and compute
esearch -db sra -query "PRJNA670204" | efetch -format runinfo > runinfo.csv
python "$BULK_CODE/scripts/01_data_retrieval/03_parse_runinfo.py" runinfo.csv SRR_Acc_List.
    ↪ txt
#   -> Run, LibraryStrategy, LibraryLayout (PAIRED/SINGLE), Platform
```

Download the runs. Two routes retrieve deposited sequencing data; verify file provenance and layout rather than assuming byte-identical representations. The SRA Toolkit fetches the `.sra` container and expands it (next subsection); the ENA serves gzipped FASTQ directly, which skips the conversion step and is often faster.

Shell

```
# ENA Portal API: turn an accession into its FASTQ URLs (and md5 checksums)
curl -s "https://www.ebi.ac.uk/ena/portal/api/filereport?accession=SRR16356854&result=
    ↪ read_run&fields=run_accession,fastq_ftp,fastq_md5,library_layout&format=tsv"

# fetch every FASTQ the report lists (loop this over SRR_Acc_List.txt)
curl -s "https://www.ebi.ac.uk/ena/portal/api/filereport?accession=SRR16356854&result=
    ↪ read_run&fields=fastq_ftp&format=tsv" \
  | tail -n +2 | tr ';' '\n' | sed 's#^#https://#' | wget -i - -P raw_data/
```

One-step helpers resolve accessions and download in a single command:

- `pysradb` converts between accession types and fetches files, e.g. `pysradb gse-to-srr GSE183947` or `pysradb metadata SRP324458`. [31]
- `ffq` returns JSON metadata and direct links: `ffq GSE183947`, `ffq --ftp SRR16356854`. [32]
- `fastq-dl -a PRJNA670204` pulls every FASTQ in a project from ENA or SRA.
- `nf-core/fetchngs` turns an accession list into FASTQ plus a tidy sample sheet. [26]
- From R, `GEOquery` reads GEO series metadata and author-deposited processed files (e.g. a counts matrix). [30]

R

```
library(GEOquery)
gse   <- getGEO("GSE183947", GSEMatrix = TRUE)     # series metadata (+ matrix if present)
pheno <- pData(gse[[1]])                            # sample-level phenotype table
getGEOSuppFiles("GSE183947")                        # author files, e.g. a counts matrix
```

Watch out for

A GEO `GSE` holds processed data and metadata; the raw reads sit in the linked SRA `SRP`/`SRR` records, so most pipelines start from the `SRR` list. Confirm `LibraryLayout` (`PAIRED` vs `SINGLE`) and `LibraryStrategy` (`RNA-Seq`) before downloading. Verify each ENA file against its published `fastq_md5` with `md5sum`.

6.2 Download raw data (SRA Toolkit)

`prefetch` fetches the compressed `.sra` container; `fasterq-dump` expands it to FASTQ. For paired-end runs it writes `<SRR>_1.fastq` and `<SRR>_2.fastq`.

Shell

```
# one accession (repeat / loop for more -- see Supplementary)
prefetch SRR1234567 --max-size 50g -O raw_data/
fasterq-dump raw_data/SRR1234567 -e 8 -O raw_data/ --split-files
gzip raw_data/SRR1234567_*.fastq          # compress to .fastq.gz
```

Key parameters

`--max-size 50g` raises `prefetch`'s default 20 GB size cap so large runs are not silently skipped. `-e 8` (= `--threads`) speeds up `fasterq-dump`. `--split-files` writes R1/R2 to separate files (paired-end). `-O` sets the output directory.

Inputs and outputs

In: an SRA accession (e.g. `SRR1234567`). **Out:** `raw_data/SRR1234567_1.fastq.gz` and `_2.fastq.gz`.

Watch out for

`fasterq-dump` needs temporary scratch up to ∼10× the `.sra` size; point `-t <tmp>` at a fast disk with room. Confirm the run is really paired-end: a single output file means single-end, which the rest of this pipeline does not assume.

6.3 Initial quality control (FastQC to MultiQC)

Initial QC establishes the read-quality baseline and identifies problems before trimming. Run FastQC on every FASTQ, then combine the reports with MultiQC to compare samples.

Shell

```
fastqc raw_data/*.fastq.gz -o qc/pre_trim/ -t 8
multiqc qc/pre_trim/ -o qc/pre_trim/ -n multiqc_pre_trim
```

Key parameters

For FastQC, `-o` specifies an existing output directory; `-t` runs one thread per file (threads beyond the number of files sit idle). For MultiQC, `-o` sets the output directory and `-n` the report name; add `-f` to overwrite.

Practical note

Read the *Per-base sequence quality*, *Adapter content*, and *Overrepresented sequences* modules first; they tell you what trimming must fix. MultiQC also parses fastp, STAR, and featureCounts logs, so rerun it at the end to summarise the whole pipeline.

6.4 Contamination screen (FastQ Screen) Optional

FastQ Screen maps a small read subset against a panel of reference genomes and reports the fraction that aligns to each. It answers a different question from FastQC: whether the library comes from the species you think it does. It can flag mouse/human cross-contamination, adapter or vector carry-over, and a heavy rRNA fraction. It is a diagnostic, not a filter, so no reads are removed.

Shell

```
# one-time: download the bundled genome panel (human, mouse, rRNA, adapters, vectors, ...)
fastq_screen --get_genomes --outdir ref/fastq_screen_db

# screen one sample's R1 with an initial 100k-read diagnostic subset
fastq_screen --aligner bowtie2 \
  --conf ref/fastq_screen_db/FastQ_Screen_Genomes/fastq_screen.conf \
  --subset 100000 --threads 8 \
  --outdir qc/fastq_screen/ \
  raw_data/SRR1234567_1.fastq.gz
```

Key parameters

`--get_genomes` downloads the reference panel and writes a matching `fastq_screen.conf` once. `--conf` points at that config (the genome list and their bowtie2 indices). `--aligner bowtie2` selects the mapper, which must be installed in the environment. `--subset 100000` samples 100k reads; `--threads` sets the thread count and `--force` overwrites an old report. R1 alone is an economical first screen; compare mates when asymmetric contamination or technical artifacts are suspected.

Inputs and outputs

In: raw FASTQ. **Out:** `qc/fastq_screen/*_screen.txt` and an HTML bar chart; MultiQC has a native module.

Watch out for

The genome panel is a several-GB one-time download; fetch it once and reuse the path. Run the screen on *raw* reads so contamination shows before trimming hides it. A high hit on the rRNA row is a reason to consider rRNA depletion (Section 6.8); a large unexpected-species fraction suggests a sample swap or contamination, which trimming will not fix.

6.5 Adapter & quality trimming (fastp)

fastp detects adapters for paired-end data, filters reads by base-quality content, drops too-short reads, and emits an HTML/JSON report, all in a single pass.

Shell

```
fastp \
  -i raw_data/SRR1234567_1.fastq.gz -I raw_data/SRR1234567_2.fastq.gz \
  -o trimmed/SRR1234567_1.trim.fq.gz -O trimmed/SRR1234567_2.trim.fq.gz \
  --detect_adapter_for_pe \
  -q 15 -u 40 -l 36 \
  -h qc/post_trim/SRR1234567.html -j qc/post_trim/SRR1234567.json \
  --thread 8
```

Key parameters

`-i`/`-I` specify input R1/R2, and `-o`/`-O` their corresponding outputs. `--detect_adapter_for_pe` enables paired-end adapter detection. `-q 15` sets the minimum base quality; `-u 40` sets the maximum allowed % of low-quality bases; `-l 36` discards reads shorter than 36 bp after trimming. `-h`/`-j` specify the reports; `--thread` sets the thread count (cap 64 in v1.0.1). For 2-colour chemistry (NextSeq/NovaSeq) add `-g` to trim poly-G tails.

The displayed values are teaching starting settings. With only this quality rule considered, a 100-base read containing 41 bases below Q15 fails `-u 40`, whereas one containing 40 does not. Raising `-q` or lowering `-u` rejects more reads; these options do not themselves trim each low-quality base. Raising `-l` discards more short remnants after trimming, potentially reducing alignable fragments as well as ambiguous short reads. Inspect paired-read retention and mapping/assignment after a change; stricter settings do not guarantee a more informative experiment. [4]

Inputs and outputs

In: raw paired FASTQ. **Out:** `trimmed/*_1.trim.fq.gz`, `*_2.trim.fq.gz` + per-sample HTML/JSON.

Watch out for

Never write trimmed output onto the input file names. The command trims adapters and filters reads by quality; sliding-window end cutting requires separate flags. Avoid over-trimming; check the before/after quality and the fraction of reads passing the length filter in the fastp report.

6.6 Quality control after trimming

Repeat FastQC + MultiQC on the trimmed reads to confirm adapters are gone and quality improved.

Shell

```
fastqc trimmed/*.fq.gz -o qc/post_trim/ -t 8
multiqc qc/post_trim/ -o qc/post_trim/ -n multiqc_post_trim
```

6.7 Alternative trimmers: Trim Galore and Trimmomatic

fastp is the default, but two other trimmers are common and are sometimes required by a lab standard or a reviewer. All three remove adapters and low-quality bases and drop too-short reads, but their settings and read-retention rules can change counts. Whichever you pick, rename the output to the canonical `trimmed/<sample>_{1,2}.trim.fq.gz` so the alignment loop (Section 12.3) is unchanged.

Trim Galore wraps Cutadapt and FastQC; it auto-detects the adapter and needs no adapter FASTA.

Shell

```
trim_galore --paired --gzip --cores 4 \
  --quality 15 --length 36 \
  --basename SRR1234567 --output_dir trimmed/ \
  raw_data/SRR1234567_1.fastq.gz raw_data/SRR1234567_2.fastq.gz

# Trim Galore names its output *_val_1.fq.gz / *_val_2.fq.gz; rename to the canonical form
mv trimmed/SRR1234567_val_1.fq.gz trimmed/SRR1234567_1.trim.fq.gz
mv trimmed/SRR1234567_val_2.fq.gz trimmed/SRR1234567_2.trim.fq.gz
```

Key parameters

`--paired` enables paired-end trimming (drops a pair if either mate falls below `--length`). `--quality 15` sets the Phred cutoff and `--length 36` the minimum retained length. `--cores` scales effectively up to about ∼4 cores. `--basename` sets the output prefix.

Trimmomatic clips against an explicit adapter FASTA and trims with a sliding window. Its argument *order* is the processing order, and it writes four files (paired + unpaired for each mate).

Shell

```
# Set BULK_ADAPTER_FASTA to the actual bundled or study-selected adapter FASTA first.
trimmomatic PE -threads 8 \
  raw_data/SRR1234567_1.fastq.gz raw_data/SRR1234567_2.fastq.gz \
  trimmed/SRR1234567_1.trim.fq.gz trimmed/SRR1234567_1.unpaired.fq.gz \
  trimmed/SRR1234567_2.trim.fq.gz trimmed/SRR1234567_2.unpaired.fq.gz \
  "ILLUMINACLIP:$BULK_ADAPTER_FASTA:2:30:10" LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN
    ↪ :36
```

Key parameters

Output order: *R1 paired, R1 unpaired, R2 paired, R2 unpaired*; feed only the two *paired* files downstream. `ILLUMINACLIP:<fa>:2:30:10` adapter file : seed mismatches : palindrome-clip : simple-clip thresholds. `LEADING`/`TRAILING` 3 trim leading/trailing bases below quality 3; `SLIDINGWINDOW:4:15` cut when a 4-base window's mean quality drops below 15; `MINLEN:36` drop reads shorter than 36.

Inputs and outputs

In: raw paired FASTQ. **Out:** `trimmed/*_1.trim.fq.gz`, `*_2.trim.fq.gz` (canonical names after the rename or the paired outputs).

Watch out for

Trim Galore's `*_val_1.fq.gz` names must be renamed or the alignment loop will not find them. Trimmomatic needs the adapter FASTA on the path (use the one bundled in the conda env) and discards the *unpaired* orphans for a standard paired-end count route; its steps run in the order written, so keep `ILLUMINACLIP` first. Choose **Trim Galore** for hands-off adapter detection close to fastp's defaults, **Trimmomatic** when you need explicit control of the adapter set and quality window.

6.8 rRNA removal (SortMeRNA)

Optional

Wet-lab poly(A) selection or rRNA depletion does not establish that the sequenced library has no residual rRNA. Consider computational removal when QC shows residual rRNA that interferes with the intended non-rRNA analysis; retain rRNA if it is itself part of the question. Record whether computational filtering has already been applied and its retention report before deciding on another pass. The protocol label alone is neither a reason to filter nor to skip. Removing rRNA-classified pairs changes the retained fragment pool; inspect loss and pair synchronization before alignment.

Download the default database from the SortMeRNA **4.3.4** release (`database.tar.gz`) and use `smr_v4.3_default_db.fasta`. Record the database release and provenance. Accuracy estimates depend on the benchmark sequences and error conditions; a release-page benchmark is not a guarantee for this organism or library.

Shell

```
DB=~/sortmerna/db/smr_v4.3_default_db.fasta
WORK=sortmerna_work/SRR1234567   # a new, sample-specific directory

sortmerna \
  --ref "$DB" \
  --reads trimmed/SRR1234567_1.trim.fq.gz \
  --reads trimmed/SRR1234567_2.trim.fq.gz \
  --paired_in \
  --fastx \
  --aligned rrna_clean/SRR1234567_rRNA \
  --other    rrna_clean/SRR1234567_norRNA \
  --out2 \
  --threads 8 \
  --workdir "$WORK"

# Keep this work directory; use a different one for the next sample.
```

Key parameters

`--ref` specifies the rRNA database; give `--reads` *twice* for paired-end input. `--paired_in`: if *either* mate is rRNA, both go to `--aligned` (conservative removal). Use `--paired_out` instead to keep a pair when *either* mate is non-rRNA (retains more reads). `--fastx` writes FASTA/FASTQ; `--aligned`/`--other` are the rRNA / kept-read prefixes; `--out2` writes paired reads as *separate* R1/R2 files; `--threads` sets the thread count and `--workdir` the working directory.

Inputs and outputs

In: trimmed paired FASTQ. **Out (with `--out2`):** `*_norRNA_fwd.fq[.gz]` and `*_norRNA_rev.fq[.gz]` (reads you keep), plus `*_rRNA_*` (rRNA you discard).

Watch out for

The `--workdir` contains indexes and sample-specific state. Use a separate new work directory per sample; do not recursively clear a shared directory. Inspect an existing run before choosing a documented resume or a new output location. Because `--out2` already yields tidy R1/R2 files, the next step (BBMap re-pairing) is only needed if you produced *interleaved* output or your pairs became desynchronised.

6.9 Reference-free rRNA removal (RiboDetector)

Optional

RiboDetector classifies reads as rRNA or not with a trained neural network instead of aligning them to an rRNA database. It needs no reference download, runs on CPU, and is fast. Use it in place of SortMeRNA when you lack a curated rRNA database for your organism or prefer not to maintain one. It writes tidy paired output directly, so no BBMap re-pairing follows.

Shell

```
# -l is the read length; set it to your data (here 150 bp)
ribodetector_cpu -t 8 -l 150 \
  -i trimmed/SRR1234567_1.trim.fq.gz trimmed/SRR1234567_2.trim.fq.gz \
  -e norrna --chunk_size 256 \
  -o rrna_clean/SRR1234567_1.fq.gz rrna_clean/SRR1234567_2.fq.gz
```

Key parameters

`-i` specifies R1 and R2 (one file for single-end); `-o` specifies the matching non-rRNA outputs. `-l` is the **read length**; the model expects the true length. `-e norrna` keeps reads confidently classed as non-rRNA (the conservative *ensure* mode); `--chunk_size` sets reads per batch (lower it if memory is tight), and `-t` sets the thread count.

Inputs and outputs

In: trimmed paired FASTQ. **Out:** `rrna_clean/*_1.fq.gz`, `*_2.fq.gz` (non-rRNA reads, already paired).

Watch out for

Set `-l` to the real read length; a wrong value degrades the classifier. RiboDetector installs a ∼2 GB PyTorch stack, so add it only when you use this route. Because the output is already paired and named per mate, skip the BBMap re-pair step below and feed `rrna_clean/` (the companion standardises both rRNA routes to `<sample>_1.trim.fq.gz`/`_2.trim.fq.gz`) straight into STAR. Prefer **SortMeRNA** for database-backed, auditable removal; **RiboDetector** when no suitable rRNA reference exists. A contamination screen (Section 6.4) shows how much rRNA is present before you decide to run either.

6.10 Re-pair reads (BBMap `repair.sh`)

Optional

Use this only if your rRNA-filtered reads are interleaved or out of order. `repair.sh` restores matched R1/R2 order and sends orphaned reads to a singletons file.

Shell

```
repair.sh \
  in=rrna_clean/SRR1234567_norRNA.fq.gz \
  out1=rrna_clean/SRR1234567_R1.fq.gz \
  out2=rrna_clean/SRR1234567_R2.fq.gz \
  outs=rrna_clean/SRR1234567_singletons.fq.gz \
  overwrite=t
```

Key parameters

`in=` interleaved input (or `in1=`/`in2=` for two files); `out1=`/`out2=` re-paired R1/R2; `outs=` single- tons (*unpaired*); `overwrite=t` allows overwriting. Set Java memory with `-Xmx` (e.g. `repair.sh -Xmx8g ...`) for large files.

Watch out for

If you skipped SortMeRNA, skip this too and feed the `trimmed/` reads straight into STAR. Keep file naming consistent so the alignment loop can find R1/R2 automatically.

6.11 Build the genome index (STAR)

Alignment requires a reference genome and its matching annotation. Download the genome (FASTA) and annotation (GTF) from NCBI or Ensembl, then build a STAR index once per genome.

Shell

```
# Set READLEN from the protocol and SAINDEX from FASTA length (formula below).
# The companion index script computes SAINDEX from the uncompressed FASTA.
STAR --runMode genomeGenerate \
  --genomeDir ref/ \
  --genomeFastaFiles ref/genome.fna \
  --sjdbGTFfile ref/genome.gtf \
  --sjdbOverhang "$((READLEN-1))" \
  --genomeSAindexNbases "$SAINDEX" \
  --runThreadN 12
```

Key parameters

`--genomeDir` specifies the index output, `--genomeFastaFiles` the genome FASTA, and `--sjdbGTFfile` the annotation used for splice junctions. `--sjdbOverhang` should be *read length* -1 (e.g. 149 for 150 bp reads; the default 100 is fine for $\sim$100–150 bp). `--genomeSAindexNbases` **must be reduced for small genomes**: $\min(14,\ \lfloor \log_2(\text{GenomeLength})/2 - 1 \rfloor)$; e.g. $\sim$11 for a 35 Mb fungal genome, 14 for mammals. `--runThreadN` sets the thread count.

Inputs and outputs

In: `genome.fna` + `genome.gtf`. **Out:** index files in `ref/` (SA, SAindex, Genome, …).

Watch out for

For small non-mammalian genomes, leaving `--genomeSAindexNbases` at 14 can produce warnings and mapping problems, so compute it from genome length. If your GTF uses non-standard attribute names, set `--sjdbGTFtagExonParentGene` (for GFF3, `--sjdbGTFtagExonParentTranscript Parent`). Memory depends on assembly, index settings, threads and sorting. A sparse suffix array can reduce index memory, but assess alignment and sort peaks separately (Section 5.1).

6.12 Alignment (STAR to sorted BAM)

Align each sample's paired reads to the indexed genome and write a coordinate-sorted BAM. If you ran the optional rRNA steps, point `--readFilesIn` at the `rrna_clean/` reads instead of `trimmed/`. STAR is fast but memory-hungry; for a lower-memory splice-aware alternative see HISAT2 (Section 6.13).

Shell

```
STAR --runMode alignReads \
  --genomeDir ref/ \
  --readFilesIn trimmed/SRR1234567_1.trim.fq.gz trimmed/SRR1234567_2.trim.fq.gz \
  --readFilesCommand zcat \
  --outSAMtype BAM SortedByCoordinate \
  --quantMode GeneCounts \
  --runThreadN 12 \
  --outFileNamePrefix mapped/SRR1234567_

samtools index mapped/SRR1234567_Aligned.sortedByCoord.out.bam
samtools flagstat mapped/SRR1234567_Aligned.sortedByCoord.out.bam
```

Key parameters

`--readFilesIn` takes R1 followed by R2; `--readFilesCommand zcat` reads `.gz` on the fly. `--outSAMtype BAM SortedByCoordinate` writes a sorted BAM directly. `--quantMode GeneCounts` also writes `*ReadsPerGene.out.tab` (a per-gene count you can use as a cross-check against featureCounts, or as the count matrix directly, Section 6.19). `--outFileNamePrefix` **must include the path and a trailing separator** (here a trailing `_`). Add `--twopassMode Basic` for more sensitive novel-junction detection.

Inputs and outputs

In: clean paired FASTQ + STAR index. **Out:** `mapped/SRR1234567_Aligned.sortedByCoord.out.bam` (+ `.bai`), `*_Log.final.out` (alignment stats → MultiQC), `*_SJ.out.tab`.

Watch out for

Mismatched genome/GTF versions, or a wrong `--readFilesCommand`, are the usual causes of poor mapping; check `*_Log.final.out` (*Uniquely mapped %*). Always index the BAM before viewing in IGV. `samtools merge` is only for combining lanes/technical replicates of the same sample, not R1/R2.

6.13 Alternative aligner: HISAT2

HISAT2 is a splice-aware aligner like STAR, built on a hierarchical graph FM index. Its main practical advantage is memory: a human index aligns in a few GB of RAM against STAR's ∼30 GB, so it can be appropriate for lower-memory hardware [49]. Install with `conda install -c bioconda hisat2` (native on Linux/WSL2; on Apple Silicon use the emulated `osx-64` env from Section 5.1 if no arm64

build resolves). A genome-aligned BAM then feeds featureCounts or htseq-count exactly as the STAR route does.

Genome route (recommended). Extract splice sites from the GTF once, build a plain index (modest RAM), then supply the splice sites at alignment time, which avoids the very large memory a full graph-index build needs for big genomes.

Shell

```
# 1. splice sites from the annotation (script ships with HISAT2)
hisat2_extract_splice_sites.py ref/genome.gtf > ref/genome.ss

# 2. build a plain genome index -> ref/genome_idx.1.ht2 ... .8.ht2 (8 files, current dir; no
    ↪ folder)
hisat2-build -p 8 ref/genome.fna ref/genome_idx

# 3. align (paired-end, dUTP reverse); supply known junctions here, sort in one pipe
hisat2 -p 8 --rna-strandness RF \
  --known-splicesite-infile ref/genome.ss \
  -x ref/genome_idx \
  -1 trimmed/SRR1234567_1.trim.fq.gz -2 trimmed/SRR1234567_2.trim.fq.gz \
  --summary-file mapped/SRR1234567.hisat2.summary \
  | samtools sort -@ 4 -o mapped/SRR1234567.sorted.bam -
samtools index mapped/SRR1234567.sorted.bam
```

Key parameters

`--dta` (optional) tunes junction reporting for StringTie/Cufflinks assembly; omit it for a plain featureCounts/DESeq2 count route. `--rna-strandness RF` specifies dUTP/reverse paired-end input; FR specifies forward, and R/F are used for single-end input. `--known-splicesite-infile` adds known junctions at align time, cheaper than baking them into the index. `--summary-file` writes a MultiQC-readable report. Piping straight into `samtools sort` skips a large intermediate SAM.

Compact-genome shortcut. A transcriptome-coordinate alignment is a separate illustrative route, including for compact genomes. Its usefulness depends on annotation completeness and transcript ambiguity; short introns alone do not establish equivalence to genome alignment. Prefer the supported Salmon/tximport branch for transcript abundance estimation. Download the `*.cdna.all.fa` from Ensembl, then:

Shell

```
gunzip Fusarium_graminearum.ASM24013v3.cdna.all.fa.gz
hisat2-build Fusarium_graminearum.ASM24013v3.cdna.all.fa cdna_idx      # 8 .ht2 files,
    ↪ working dir
hisat2 -x cdna_idx -1 trim_R1.fastq -2 trim_R2.fastq \
  | samtools sort -o kont1.sorted.bam -
```

Watch out for

`hisat2-build` writes the eight `.ht2` files into the **current directory** under the basename you give; it does not create a folder. A cDNA-aligned BAM is in *transcript* coordinates, so you cannot count it against the genome GTF (that yields near-zero counts): count per transcript and aggregate with a `tx2gene` map, or prefer the genome route for vertebrate or intron-rich genomes. The shortcut also cannot discover novel junctions. Building a full graph index with `--ss`/`--exon` for a mammalian genome needs very large RAM (∼160 GB for human); on a 24 GB machine use a prebuilt HISAT2 index or the plain-index plus `--known-splicesite-infile` route above.

6.14 Determine library strandedness

Stranded protocols record which genomic strand each read came from, and the counter must be told the convention or it can discard most reads (the silent error flagged below). The common Illumina chemistry (TruSeq Stranded, NEBNext) marks the second cDNA strand with dUTP and then digests it, so the sequenced fragment reports the *reverse* complement of the mRNA. In counter terms that is reverse-stranded: `featureCounts -s 2`, `htseq-count --stranded reverse`, Salmon ISR. Unstranded libraries use `-s 0`; the rarer forward (FR) protocols use `-s 1`. Knowing the strand also resolves overlapping genes on opposite strands to the correct one.

Use an aligned sample to gather evidence for the setting, and compare it with the documented library protocol.

Shell

```
# Option A -- RSeQC: compare each read's strand to the annotated transcript strand
infer_experiment.py -r ref/genes.bed -i mapped/SRR16356854_Aligned.sortedByCoord.out.bam

# Option B -- read it off STAR's per-gene counts (needs --quantMode GeneCounts):
#   col2 = unstranded, col3 = forward (-s 1), col4 = reverse (-s 2)
#   Compare col3 versus col4; do not choose the largest of all three.
grep -v "^N_" mapped/SRR16356854_ReadsPerGene.out.tab \
  | awk '{u+=$2; f+=$3; r+=$4} END{print "unstranded="u" forward="f" reverse="r}'
```

Key parameters

`infer_experiment.py` prints the fraction of reads explained by each convention: ∼50/50 means *unstranded*; for paired-end data a dominant `"1++,1--,2+-,2-+"` supports *forward* (`-s 1`), while `"1+-,1-+,2++,2--"` supports *reverse* (`-s 2`). Review the undetermined fraction and the documented library protocol. [71] `-r` wants a 12-column BED of gene models; make it once from the GTF (e.g. UCSC `gtfToGenePred` then `genePredToBed`).

Practical note

Salmon can infer this automatically: `salmon quant -l A` infers the library type and records it in `lib_format_counts.json`. Whatever the method, fix the strandedness once and reuse it for every sample in the study.

6.15 Gene-level quantification (featureCounts)

featureCounts summarises aligned read *pairs* into a gene × sample matrix using the GTF. Pass all BAMs at once to build a single matrix. htseq-count is the common alternative (Section 6.18).

Shell

```
featureCounts \
  -a ref/genome.gtf \
  -o counts/counts.txt \
  -T 6 \
  -p --countReadPairs \
  -t exon -g gene_id \
  -s 2 \
  -Q 10 \
  mapped/*_Aligned.sortedByCoord.out.bam
```

Key parameters

`-a` specifies the GTF annotation, `-o` the output matrix, and `-T` the thread count. `-p` declares paired-end input; in Subread ≥2.0.2 add `--countReadPairs` to count *fragments* rather than reads. `-t exon -g gene_id` count over exons grouped by gene (the defaults). `-s` strandedness: 0 unstranded, 1 stranded, 2 reverse-stranded. `-Q 10` sets the minimum mapping quality.
This is a teaching threshold for alignment confidence, not base-call quality. Raising it can reduce uncertain assignments while also removing informative fragments in difficult regions; lowering it retains more alignments of uncertain origin. Compare assignment categories and retained counts after changing it. Mapping-quality scales and special values depend on the aligner, so a fixed threshold is not a universal uniqueness guarantee.

Inputs and outputs

In: sorted BAMs + GTF. **Out:** `counts/counts.txt` (6 annotation columns: Geneid, Chr, Start, End, Strand, Length, then one count column per BAM) and `counts/counts.txt.summary` (assignment stats → MultiQC).

Watch out for

Strandedness is an important source of silent errors. The wrong `-s` can dump most reads into `Unassigned_NoFeatures`. Most Illumina dUTP libraries are *reverse*-stranded (`-s 2`); if unsure, compare the two strand-specific conventions and the protocol, using RSeQC or STAR forward/reverse totals. The unstranded column can have the largest total even for a stranded library; maximising over 0/1/2 does not infer strandedness. [71] Remember the first six columns when importing into R.

6.16 QC summary: mapping and assignment rates

Two useful QC summaries are: the *overall alignment rate* (the fraction of reads that mapped) and the *assignment rate* (the fraction of processed input fragments or reads assigned to genes, according to counting mode). STAR writes the first to `*_Log.final.out`, featureCounts the second to `counts.txt.summary`. To pull them into a table:

Shell

```
# uniquely + multi-mapped % from each STAR log -> a tab-separated table
for f in mapped/*_Log.final.out; do
  s=$(basename "$f" _Log.final.out)
  uniq=$(grep "Uniquely mapped reads %" "$f" | grep -o "[0-9.]*%")
  multi=$(grep "mapped to multiple loci" "$f" | grep -o "[0-9.]*%" | head -1)
  printf "%s\t%s\t%s\n" "$s" "$uniq" "$multi"
done | sort | column -t

# overall mapping rate straight from a BAM
samtools flagstat mapped/SRR1234567_Aligned.sortedByCoord.out.bam | grep "mapped ("

# featureCounts assignment per sample (Assigned counts; MultiQC converts these to %)
grep -E "^(Status|Assigned)" counts/counts.txt.summary | column -t
```

Practical note

MultiQC already assembles these into `multiqc_data/multiqc_general_stats.txt`, a spreadsheet-ready table of per-sample metrics from FastQC, fastp, STAR, and featureCounts. Run `multiqc .` at the end and read that file (or the HTML) rather than re-parsing each log. Flag samples with low unique mapping (genome/GTF mismatch or contamination) or low assignment (usually the wrong `-s` strandedness).

6.17 Post-alignment RNA QC (RSeQC)

Optional

RSeQC provides additional diagnostics of RNA and library quality beyond mapping rate.

Read distribution is reported by `read_distribution.py`. It classifies reads as exonic, intronic, UTR or intergenic; high intronic or intergenic fractions can indicate DNA contamination or pervasive pre-mRNA.

Gene-body coverage is reported by `geneBody_coverage.py`. It measures 5' to 3' coverage evenness; a 3' skew can indicate degraded RNA. Both commands require a BAM and a 12-column BED gene model.

Shell

```
# build the 12-column BED once from the GTF (the same model infer_experiment.py used, the
    ↪ strandedness section)
gtfToGenePred ref/genome.gtf ref/genome.genePred
genePredToBed ref/genome.genePred ref/genes.bed

read_distribution.py -i mapped/SRR1234567_Aligned.sortedByCoord.out.bam -r ref/genes.bed \
  > qc/rseqc/SRR1234567.read_dist.txt
geneBody_coverage.py -i mapped/SRR1234567_Aligned.sortedByCoord.out.bam -r ref/genes.bed \
  -o qc/rseqc/SRR1234567
multiqc qc/rseqc/ -o qc/rseqc/ -n multiqc_rseqc
```

Key parameters

`-i` supplies the sorted, indexed BAM; `-r` supplies the 12-column BED gene model built once with the UCSC `gtfToGenePred` → `genePredToBed` tools. For `geneBody_coverage.py`, `-o` sets the output prefix for a per-sample curve and a MultiQC-readable summary.

Inputs and outputs

In: sorted BAM (+ `.bai`) and `ref/genes.bed`. **Out:** read-distribution tables and gene-body-coverage curves in `qc/rseqc/` (→ MultiQC).

Watch out for

These reports help investigate degraded RNA (3' bias) or genomic contamination (high intergenic fraction); they do not filter reads. `geneBody_coverage.py` is slow, so it can be limited to a few representative samples. Build the BED from the *same* GTF used for alignment and counting.

6.18 Alternative counter: htseq-count

htseq-count is the original union-mode counter from the DESeq lineage and the usual alternative to featureCounts [50, 51]. It reads a sorted BAM plus the GTF and writes one count per gene, counting a read only when it maps unambiguously to a single gene: reads over two genes go to `__ambiguous` and multimappers to `__alignment_not_unique`, so the counts stay valid input for DESeq2's model.

Shell

```
htseq-count -f bam -r pos -s reverse -t exon -i gene_id -m union \
  mapped/SRR1234567_Aligned.sortedByCoord.out.bam ref/genome.gtf \
  > counts/SRR1234567.tsv

# if you assemble the matrix by hand, drop the trailing summary rows first:
grep -v "^__" counts/SRR1234567.tsv > counts/SRR1234567.clean.tsv
```

Key parameters

`-f bam -r pos` read a coordinate-sorted BAM (use `-r name` on a name-sorted BAM for constant-memory paired counting). `-s reverse` dUTP/TruSeq strandedness (= `featureCounts -s 2`); `-s yes` = `-s 1`; `-s no` = `-s 0`. `-t exon -i gene_id` feature type and grouping attribute. `-m union` overlap mode (also `intersection-strict`, `intersection-nonempty`). HTSeq 2 adds `-n <N>` for multicore counting.

R

```
# read straight into DESeq2 (strips the trailing __ rows automatically)
library(DESeq2)
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
cfg <- read_analysis()
samples <- read_samples(cfg = cfg)
samples$sampleName <- samples$sample
samples$fileName <- paste0(samples$sample, ".tsv")
samples <- samples[, c("sampleName", "fileName", "condition",
                       setdiff(names(samples), c("sampleName", "fileName", "condition")))]
dds <- DESeqDataSetFromHTSeqCount(sampleTable = samples,   # cols: sampleName, fileName,
                                                           # condition
                                  directory   = "counts",
                                  design      = cfg$design)
```

Watch out for

`htseq-count` defaults to `-s yes` (stranded), not unstranded. On reverse-stranded data, leaving that default can assign genuine signal to `__no_feature`, so always set `-s` explicitly, just as with featureCounts. `DESeqDataSetFromHTSeqCount` drops the trailing summary rows (`__no_feature`, `__ambiguous`, `__alignment_not_unique`, …) automatically; strip them by hand (above) only when building the matrix yourself. htseq-count is slower than featureCounts and runs on Linux and macOS (use WSL2 on Windows).

6.19 Alternative counter: STAR gene counts (`--quantMode GeneCounts`)

The STAR alignment already ran with `--quantMode GeneCounts` (Section 6.12), so each sample carries a `*_ReadsPerGene.out.tab` of per-gene counts in three strand conventions. Assembling those skips the featureCounts pass. STAR uses htseq `union` counting semantics, so the result is valid DESeq2 input and can be compared with featureCounts.

R

```
#### Build the count matrix from STAR ReadsPerGene tables ####
files <- list.files("mapped", "_ReadsPerGene.out.tab$", full.names = TRUE)
names(files) <- sub("_ReadsPerGene.out.tab$", "", basename(files))

scol <- 4                                    # 2 = unstranded, 3 = forward (-s 1), 4 = reverse
                                              # (-s 2)
read_star <- function(f) {
  x <- read.delim(f, header = FALSE, skip = 4)   # drop N_unmapped/N_multimapping/N_
    ↪ noFeature
                                                  # /N_ambiguous
  setNames(x[[scol]], x[[1]])
}
mats <- lapply(files, read_star)
stopifnot(length(unique(lapply(mats, names))) == 1)   # identical gene order in every sample
cts  <- do.call(cbind, mats)
library(DESeq2)
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
cfg <- read_analysis()
cts <- check_counts(cts)
coldata <- read_samples(colnames(cts), cfg)
dds <- DESeqDataSetFromMatrix(countData = cts, colData = coldata, design = cfg$design)
```

Key parameters

`skip = 4` drops STAR's four summary rows before the genes. `scol` picks the strand column and must match the strandedness from Section 6.14: 2 unstranded, 3 forward, 4 reverse. `check_counts()` validates integer counts before changing their storage mode; `read_samples()` aligns independently authored metadata and checks the configured design.

Watch out for

The STAR index must have been built with `--sjdbGTFfile`, or `ReadsPerGene.out.tab` is empty. These tables have *no* annotation columns, so do **not** route them through the featureCounts importer that strips six leading columns (Section 8), which would delete real samples. Pick one `scol` for the whole study. Different counters use different assignment and multimapping rules; compare their outputs only with matched annotations and settings, and investigate material disagreements.

6.20 Alignment-free quantification (Salmon)

Salmon and kallisto skip base-level genome alignment: they map reads against the *transcriptome* and estimate transcript abundances directly. This route is fast and can use a few GB of RAM instead of STAR's tens of GB, and Salmon also models fragment GC and sequence bias. Import the transcript-level estimates to gene level with tximport, after which the appropriate DESeq2 constructor supplies transcript-length offsets. The companion includes a dedicated Salmon entry script.

Shell

```
# Transcript-only illustration; use a decoy-aware transcriptome for selective alignment.
# Pass --decoys decoys.txt with a matching transcriptome-plus-decoy FASTA.
salmon index -t ref/transcripts.fa -i ref/salmon_idx -k 31
salmon quant -i ref/salmon_idx -l A \
  -1 trimmed/SRR16356854_1.trim.fq.gz -2 trimmed/SRR16356854_2.trim.fq.gz \
  --gcBias -p 8 -o quant/SRR16356854
```

R

```
library(tximport); library(DESeq2)
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
cfg <- read_analysis()
coldata <- read_samples(cfg = cfg)
files <- setNames(file.path("quant", coldata$sample, "quant.sf"), coldata$sample)
tx2gene <- read.delim("config/tx2gene.tsv", colClasses = "character")
txi <- tximport(files, type = "salmon", tx2gene = tx2gene)   # tx2gene: transcript -> gene
                                                              # map
dds <- DESeqDataSetFromTximport(txi, colData = coldata, design = cfg$design)
```

Watch out for

For differential expression, feed *raw counts*, or let `tximport` supply the length offset that corrects for transcript length. Salmon recommends genomic decoys to reduce spurious mapping in selective alignment; transcript-only indexing remains a simplified illustration. [73] Do not run count-based DE on TPM or FPKM: those are within-sample normalisations and throw away the count-level information the models depend on.

6.21 Single-end reads

The body assumes paired-end data. Single-end (SE) libraries change only the read-handling flags; the stages from the count matrix onward (DESeq2, enrichment, figures) use the same framework. The per-step changes:

- **Download.** An SE run yields one FASTQ (no `_1`/`_2`); fasterq-dump's default `--split-3` already handles it.
- **fastp.** Use `-i`/`-o` only (no `-I`/`-O`) and drop `--detect_adapter_for_pe` (it only enables the paired- end overlap detection). SE adapter trimming is on by default (fastp finds the overrepresented 3' adapter); pass `--adapter_sequence <SEQ>` if auto-detection misses it. `-q`/`-l` are unchanged.
- **STAR.** Give one file to `--readFilesIn reads.fq.gz`.
- **HISAT2.** Use `-U reads.fq.gz` (not `-1`/`-2`) with `--rna-strandness R` (or `F`).
- **featureCounts.** Remove `-p` and `--countReadPairs` (both are paired-end only); keep `-s`. htseq-count needs no change.
- **Strandedness.** RSeQC `infer_experiment.py` still works; for SE it reports `"++,--"` vs `"+-,-+"` rather than the read1/read2 patterns.

Watch out for

Leaving `-p`/`--countReadPairs` on single-end BAMs is a common error that makes featureCounts miscount. SE is cheaper and fine for gene-level DE; paired-end resolves isoforms, splice junctions, and multimapping reads better, so prefer it when isoform-level questions matter.

7 Alternative entry points

The pipeline so far went from raw reads to a count matrix. Three other starting points join the same downstream analysis. A ready count matrix or a GEO microarray series enters at differential expression (Section 8); a pre-computed results table enters at enrichment and figures (Sections 9, 11), because its per-gene statistics already exist.

7.1 Starting from a count matrix

If someone already produced a gene × sample count matrix, or you downloaded one, skip the shell stages and read it straight into DESeq2. Continue at *Build the dataset, pre-filter, set the reference level* (Section 8), not at the featureCounts importer, so you do not strip columns a plain matrix does not have.

R

```
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
cfg <- read_analysis()
cts <- read_count_matrix("counts/matrix.tsv")
coldata <- read_samples(colnames(cts), cfg)
dds <- DESeqDataSetFromMatrix(cts, coldata, design = cfg$design)
```

Watch out for

Feed *raw integer* counts. Decimal values can be normalised quantities or estimated counts; establish their provenance. Import supported transcript estimates with `tximport` (Section 6.20), and never round TPM into DESeq2. Strip any trailing htseq/STAR summary rows first. Note the gene-ID type (Ensembl, symbol, Entrez) now: the annotation and enrichment steps (Sections 8.6, 9) depend on it.

7.2 Starting from a differential-expression results table

If the differential-expression statistics already exist (a collaborator's DESeq2/edgeR/limma table, or a supplementary file), go straight to enrichment, networks, and the results-only figures without recomputing anything. Standardise the column names to what the downstream code expects.

R

```
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
cfg <- read_analysis()
res_df <- read.delim("input/de_table.tsv", check.names = FALSE)
stopifnot("gene_id" %in% names(res_df))
check_ids(res_df$gene_id, "Result gene IDs")
rownames(res_df) <- res_df$gene_id
# rename to the canonical columns used from here on
names(res_df)[names(res_df) == "logFC"]     <- "log2FoldChange"   # edgeR / limma name
names(res_df)[names(res_df) == "adj.P.Val"] <- "padj"             # limma name
names(res_df)[names(res_df) == "FDR"]       <- "padj"             # edgeR name
stopifnot(all(c("log2FoldChange", "padj") %in% names(res_df)))
res <- check_results(res_df)  # retain gene identities; use the companion result-only
    ↪ adapter
```

Watch out for

The executable upload route requires an explicit `gene_id` column, declared namespace and valid numeric statistics. Confirm the fold-change *direction* matches your intended contrast (treated relative to control); a flipped sign inverts every downstream call. Count-based figures (PCA, sample distance, the expression heatmap, genes of interest) cannot run without the counts, so skip them and keep the volcano and enrichment. An MA plot additionally needs a compatible abundance measure; a generic DE table may not provide one. If the gene IDs are symbols, set `fromType = "SYMBOL"` in the enrichment mapping (Section 9) and drop the Ensembl version-strip line there.

7.3 GEO microarray series with limma

A GEO microarray series is analysed with limma rather than DESeq2, but it writes the same `res` table. The companion adapts its expression matrix and sample metadata for figures; count-specific DESeq2 methods do not apply to that matrix. Fetch the normalised series matrix, decide its documented preprocessing and expression scale, select one platform, collapse unambiguous probes to genes, then fit the declared design and contrast.

R

```
library(GEOquery); library(limma)
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
cfg <- read_analysis()
series <- getGEO("GSE5583", GSEMatrix = TRUE, AnnotGPL = TRUE)
platform <- "<GPL accession>"  # inspect annotation for each returned platform
selected <- vapply(series, function(x) annotation(x) == platform, logical(1))
stopifnot(sum(selected) == 1)
gse <- series[[which(selected)]]
expr <- exprs(gse)                          # probes x samples (usually already normalised)

# log2 only if the values look like linear intensities (GEO2R heuristic)
scale <- "log2"  # explicitly set from the deposit/platform documentation
if (scale == "linear") {
  stopifnot(all(is.finite(expr)), all(expr > 0))
  expr <- log2(expr)
}

# collapse probes to gene symbols, keeping one value per gene
sym <- fData(gse)$"Gene symbol"               # column name varies by platform
keep <- !is.na(sym) & nzchar(sym) & !grepl("///|;|,", sym)
expr <- avereps(expr[keep, , drop = FALSE], ID = sym[keep])

coldata <- read_samples(colnames(expr), cfg) # explicit sample IDs and condition
design <- model.matrix(cfg$design, coldata)
fit    <- eBayes(lmFit(expr, design), trend = TRUE, robust = TRUE)   # limma-trend
res    <- topTable(fit, coef = "conditiontreated", number = Inf) # verify against colnames(
    ↪ design)
names(res)[names(res) %in% c("logFC", "adj.P.Val")] <- c("log2FoldChange", "padj")
```

The log2 `expr` matrix plays the role of the variance-stabilised matrix for the sample-structure figures in Section 11.

Watch out for

Most GEO series matrices are already normalised, so do not re-normalise or double-log them: use the submitter's preprocessing record and platform documentation. Value ranges are diagnostic clues, not a sufficient transformation rule. The gene-symbol column name varies (`Gene symbol`, `Gene Symbol`, `gene_assignment`); inspect `fData(gse)`. Collapse multi-probe genes so each gene appears once. For raw Affymetrix CEL files, read them with `affy::ReadAffy()` then `affy::rma()` to get the normalised matrix.

8 Differential expression with DESeq2

8.1 Import counts and build sample metadata

The model interprets each count column using the corresponding sample's condition and design information. Read the featureCounts matrix, drop its six annotation columns, and build a `colData` table whose row order *exactly* matches the count columns.

R

```
#### Setup (run once) ####
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install(c("DESeq2", "apeglm"))
library(DESeq2)
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
cfg <- read_analysis()

#### Import the featureCounts matrix ####
fc            <- read.delim("counts/counts.txt", comment.char = "#", check.names = FALSE)
rownames(fc) <- fc$Geneid
cts           <- as.matrix(fc[, -(1:6)])     # drop Geneid,Chr,Start,End,Strand,Length
cts           <- check_counts(cts)          # reject decimals, missing/negative counts first

# featureCounts names columns after the BAM paths -- tidy them
colnames(cts) <- gsub("_Aligned.sortedByCoord.out.bam$", "", basename(colnames(cts)))

#### Sample metadata: identity, design, and covariates ####
coldata <- read_samples(colnames(cts), cfg)
stopifnot(identical(rownames(coldata), colnames(cts)))
```

Watch out for

Match independently authored sample metadata to count columns by ID. A guard is ineffective if metadata row names were created from those columns while conditions were assigned by position. The companion rejects duplicate or missing IDs, mismatched sample sets, missing covariates and confounded designs before fitting. Counts must be raw integers, not TPM/FPKM/CPM: DESeq2 models the counts and normalises internally.

8.2 Build the dataset, pre-filter, set the reference level

The configured design is `~ condition` by default, so it models expression by group. Use the same `cfg$design`, reference and contrast throughout import, fitting and export; edit the configuration when an identifiable batch or paired-donor term is required. Pre-filtering removes near-empty genes before fitting.

R

```
dds <- DESeqDataSetFromMatrix(countData = cts,
                              colData   = coldata,
                              design    = cfg$design)

# keep genes with >= 10 reads in at least the smallest group's size
smallestGroup <- min(table(coldata$condition))
keep          <- rowSums(counts(dds) >= 10) >= smallestGroup
dds           <- dds[keep, ]

# define the baseline level explicitly
dds$condition <- relevel(dds$condition, ref = cfg$reference)
```

Key parameters

`cfg$design` supplies the reviewed model; for example `~ batch + condition` adjusts for an identifiable batch effect. The ten-count rule is a teaching starting point. Raising it reduces the fitted gene set and computation but may discard genuinely changing low-abundance genes; lowering it retains more weakly informative tests. It changes the eligible testing family, so inspect retained genes and low-count diagnostics rather than expecting a monotonic gain in power. `cfg$reference` sets the denominator level. [69]

Practical note

With a known batch/lane effect, put it *before* the variable of interest in the design (`~ batch + condition`). DESeq2 then estimates the condition effect adjusted for batch when the design is identifiable. A batch fully confounded with condition cannot be repaired by adding a term. VST does not remove batch effects from PCA; any display-only adjustment must preserve the biological design. [69]

8.3 Run DESeq2 and extract results

Different library depths and RNA compositions can change counts without the same change in gene abundance. DESeq2 estimates size factors to place each gene's samples on a common relative scale, assuming enough stable genes or a justified control set. This is not an absolute per-cell RNA measurement. Replicates then reveal variability beyond count sampling; dispersion describes that extra variability in the negative-binomial model. The design links the expected counts to covariates, and its condition coefficient estimates the declared $\log_2$ fold change with uncertainty. Normalization cannot recover a condition effect confounded with batch. `DESeq()` estimates these components and fits the model; `results()` returns the ordinary Wald test against zero effect, and `lfcShrink()` supplies more stable display effects. [69]

R

```
dds <- DESeq(dds)
resultsNames(dds)                          # inspect the coefficient names

#### Wald test: treated vs control ####
res <- results(dds,
               contrast = cfg$contrast,
               alpha    = cfg$alpha)

#### Shrunken log2 fold changes (for ranking, MA & volcano) ####
stopifnot(cfg$coefficient %in% resultsNames(dds))
resLFC <- lfcShrink(dds,
                    coef = cfg$coefficient, res = res,
                    type = "apeglm")

summary(res)
sum(res$padj < cfg$alpha, na.rm = TRUE)         # number of significant genes
```

Key parameters

`cfg$contrast` declares numerator versus denominator; the default is treated versus control. `cfg$alpha` is 0.05 and supplies the independent-filtering target as well as the reporting cutoff. Pass the matching `res` to `lfcShrink(type = "apeglm")` with the exact configured coefficient, so shrinkage does not regenerate a different inferential table. Retain `res` for tests and import only shrunken LFCs for display. [11, 69]

Watch out for

`padj = NA` can reflect independent filtering or an outlier rule; retain this missingness. Ordinary DE selection uses `res$padj < cfg$alpha`. An additional observed `abs(res$log2FoldChange) > cfg$lfc_cutoff` is the exploratory screen used for downstream lists here; that subset does not automatically inherit the original FDR guarantee. It is distinct from testing whether the true effect exceeds a margin, described below. Shrunken values set display positions, while original effects and standard errors remain available for inference and pooling.

Hypothetical result. Suppose a gene has raw $\log_2$FC = 1.5, `lfcSE` = 0.3 and `padj` = 0.004 for treated versus control. The estimated relative abundance is about $2^{1.5} = 2.83$ times higher in treated samples; reversing the comparison gives about 0.35 times as much. The standard error describes uncertainty on the $\log_2$ scale; a larger value at the same effect means less precise estimation. A hypothetical shrunken LFC of 1.2 moves the display point without replacing the original test. The row passes the ordinary 0.05 cutoff and the observed-effect screen of 1, but this does not establish that the true LFC exceeds 1.

Use an effect threshold as a hypothesis when required. Selecting genes with ordinary DE `padj < 0.05` and then screening observed $|\log_2 FC| > 1$ does not automatically give that selected subset the original FDR guarantee. If the scientific question requires an effect exceeding a prespecified margin, use a corresponding threshold test, such as DESeq2's `lfcThreshold` with `altHypothesis = "greaterAbs"`, or edgeR's `glmTreat`. The opposite question, evidence that an effect lies inside a margin, uses the equivalence test in the differential-expression section. Neither test establishes exact zero change. Check the current method documentation before changing the primary test [69, 78].

8.4 Equivalence testing: effects within a meaningful margin Optional

A non-significant ordinary test means *not enough evidence of change*; it does not establish *no change*. To identify effects within a declared tolerance, DESeq2 offers a two one-sided tests (TOST) equivalence test: it asks whether the $\log_2$ fold change lies *inside* a band $\pm$ `lfcThreshold` rather than *outside* it [62].

R

```r
# Choose this margin from biological relevance before inspecting results.
res_eq <- results(dds, contrast = cfg$contrast,
                  lfcThreshold = cfg$equivalence_margin,
                  altHypothesis = "lessAbs", alpha = cfg$alpha)
equivalent <- subset(as.data.frame(res_eq), padj < cfg$alpha)
write.csv(equivalent, "results/equivalent_within_margin.csv")
```

Key parameters

`lfcThreshold = 1` the example equivalence band ($|\log_2 FC| < 1$, a two-fold change). Set a biologically justified tolerance before analysis. `altHypothesis = "lessAbs"` tests that the effect is *smaller* in absolute value than the threshold (the TOST alternative). `padj < alpha` then flags evidence that the selected contrast lies within the declared margin, not evidence of an exactly zero effect. [69]

Inputs and outputs

In: the fitted `dds`. **Out:** `results/equivalent_within_margin.csv`, genes whose fold change is significantly within the band.

Watch out for

A gene that passes neither criterion is inconclusive under these tests; this can reflect uncertainty or an effect near a margin. Do not widen the margin after seeing how many genes pass. Plan the margin, replication and precision together; if the study cannot distinguish the relevant effects, retain that uncertainty. Use it when a claim of equivalence within a meaningful tolerance matters (for example a housekeeping control), which an ordinary large p-value cannot support.

8.5 Transform counts and export

A variance-stabilising transform (VST) puts counts on a roughly homoscedastic, log-like scale for PCA, clustering, and heatmaps. Then write the results table.

R

```
vsd <- varianceStabilizingTransformation(dds, blind = FALSE) # also works with small gene
    ↪ panels

# ordered results table to disk
res_out <- as.data.frame(res)
res_out <- res_out[order(res_out$padj), ]
write.csv(res_out, "results/DE_results.csv")
save_analysis(dds, res, resLFC, assay(vsd), coldata, cfg)
saveRDS(vsd, "results/vsd.rds")
```

Practical note

`vst()` is a faster wrapper that subsets genes to estimate the trend; its default subset is unsuitable for very small gene panels. The full `varianceStabilizingTransformation()` avoids that subset-size constraint. `rlog()` is another useful transform, particularly when size factors vary widely, but becomes slower as sample count grows. Set `blind = FALSE` so the transform is aware of the design.

8.6 Annotate genes (AnnotationDbi and biomaRt)

Counts are keyed by stable gene IDs (Ensembl `ENSG...`, or Entrez numbers). Gene *symbols* can make result tables easier to read; enrichment requires the identifier type expected by the selected database or gene set. Additional annotations can include gene *biotype* and *length*. Two interfaces cover most of these annotation needs.

AnnotationDbi with an OrgDb (offline, fast). `mapIds()` queries the prebuilt organism database (`org.Hs.eg.db` for human; swap the species package for others).

R

```
library(AnnotationDbi)
library(org.Hs.eg.db)

ids <- sub("\\..*$", "", rownames(res))      # strip Ensembl version suffix: ENSG..."."7" ->
    ↪ ENSG...
single_target <- function(x) {
  x <- unique(x[!is.na(x)])
  if (length(x) == 1L) x else NA_character_
}
res$symbol <- mapIds(org.Hs.eg.db, keys = ids, keytype = "ENSEMBL",
                     column = "SYMBOL", multiVals = single_target)
res$entrez <- mapIds(org.Hs.eg.db, keys = ids, keytype = "ENSEMBL",
                     column = "ENTREZID", multiVals = single_target)
columns(org.Hs.eg.db)                          # all fields you can map to
```

biomaRt (online Ensembl, richest attributes). Use it for fields an `OrgDb` does not hold: gene biotype, chromosome and coordinates, GC content, transcript lengths, and cross-species homology.

R

```
library(biomaRt)
mart <- useEnsembl(biomart = "genes", dataset = "hsapiens_gene_ensembl")
ann <- getBM(
  attributes = c("ensembl_gene_id", "external_gene_name", "gene_biotype",
                 "chromosome_name", "start_position", "end_position"),
  filters    = "ensembl_gene_id",
  values     = ids,
  mart       = mart)
```

Watch out for

Strip Ensembl version suffixes (the .7 in `ENSG00000141510.7`) before mapping, or the keys will not match. Gene biotype supports feature classification, including `protein_coding` selection. Estimating rRNA or mitochondrial abundance also requires appropriate feature annotations and expression counts. One-to-many and many-to-one matches are possible. `multiVals = "first"` selects one target when a source has several; it does not resolve biological ambiguity. The companion exports every annotation match and leaves ambiguous single-value annotations missing rather than choosing an arbitrary target. The public Ensembl host has occasional downtime, so pin a release with `useEnsembl(..., version = 110)` or a mirror, and keep `AnnotationHub`/`ensembldb` (`EnsDb`) as a fully offline, versioned fallback.

8.7 Inspect genes of interest

Beyond the genome-wide table, you often want the expression of a few named genes (markers, pathway members, candidates). Work from *normalised* counts so libraries of different depth stay comparable.

R

```
#### One gene as a labelled count plot ####
goi_id <- rownames(res)[match("TP53", res$symbol)]    # symbol -> Ensembl ID (from the
    ↪ annotation step)
plotCounts(dds, gene = goi_id, intgroup = "condition") # base-R dotplot of normalised counts

#### A panel of genes: normalised counts (genes x samples) ####
goi <- c("TP53", "MDM2", "CDKN1A")
ids <- rownames(res)[match(goi, res$symbol)]
counts(dds, normalized = TRUE)[ids, ]

#### Their effect sizes and significance, from the results table ####
res[ids, c("log2FoldChange", "pvalue", "padj")]
```

Practical note

Normalised counts (size-factor scaled) make a gene's values comparable across samples, unlike *raw* counts. For a figure, pass `returnData = TRUE` to `plotCounts` and rebuild it with ggplot2 (a `geom_jitter` over condition, optionally log-scaled). Read significance from `padj`, not from the visual gap between groups.

8.8 Cross-checking DE engines: edgeR and limma-voom

The same count matrix runs through edgeR or limma-voom instead of DESeq2; the three are mature and broadly concordant. edgeR fits a negative-binomial GLM with a quasi-likelihood test [42]; limma-voom turns counts into log-CPM with precision weights and fits a linear model [43]. Select an engine from its

model, design support and diagnostics; sample count alone does not determine the appropriate method. Both start from the same `cts` and `coldata` built in Section 8.

Package capability and this companion's contract differ. edgeR v4 adds fractional-count support and revises quasi-likelihood estimation for small counts [78]. This does not authorize arbitrary normalized expression as count input. The shared integer-matrix entry point here deliberately rejects fractional values; estimated transcript counts use the documented `tximport` route. Retain the quantifier, offset strategy and fitted engine in the analysis record rather than rounding TPM or silently changing the input contract.

edgeR (quasi-likelihood F-test).

R

```
library(edgeR)
design <- model.matrix(cfg$design, data = coldata)
coefficient <- paste0(cfg$contrast[1], cfg$contrast[2])
stopifnot(coefficient %in% colnames(design), cfg$contrast[3] == cfg$reference)
y    <- DGEList(counts = cts, group = coldata$condition)    # raw integer counts
keep <- filterByExpr(y, design = design)                   # filter for the declared model
y    <- y[keep, , keep.lib.sizes = FALSE]
y    <- normLibSizes(y)                                     # TMM normalisation (older edgeR
    ↪ : calcNormFactors)

y      <- estimateDisp(y, design)                           # gene-wise + trended
    ↪ dispersions
fit    <- glmQLFit(y, design)                               # quasi-likelihood negative-
    ↪ binomial GLM
qlf    <- glmQLFTest(fit, coef = coefficient)               # declared condition effect
topTags(qlf, n = 20)                                        # ranked table, BH-adjusted FDR
```

limma-voom (reuses the filtered, TMM-normalised `y` and `design` above).

R

```
library(limma)
v   <- voom(y, design, plot = TRUE)       # mean-variance trend -> per-observation precision
    ↪  weights
fit <- lmFit(v, design)                   # weighted least-squares linear model per gene
fit <- eBayes(fit, robust = TRUE)         # moderated t-statistics, robust to hypervariable
    ↪ genes
topTable(fit, coef = coefficient, number = 20)      # ranked results
```

Key parameters

`filterByExpr` drops weakly expressed genes by the design (the recommended edgeR/limma pre-filter). `normLibSizes` (older name `calcNormFactors`) computes TMM size factors. `glmQLFit` + `glmQLFTest` are edgeR's quasi-likelihood test, also used at small n. `coefficient` selects the configured numerator level against its reference by name, so adding a covariate does not silently change the tested column. `voom` models the mean–variance trend so limma's linear model fits count data.

Watch out for

Feed these engines *raw* counts (or, for the alignment-free route, the `tximport` length offset), never TPM or FPKM: those are within-sample normalisations that discard the count-level information the models depend on. Agreement depends on design, filtering, dispersion estimation and the data. Cross-engine agreement is a sensitivity check, not proof that the experiment identifies the intended effect. Match `coef`/`contrast` to your design, *exactly* as with the DESeq2 contrast.

9 Functional enrichment

The input representation determines the enrichment question. *Over-representation analysis* (ORA) asks whether a category is enriched in your selected-gene *list*. *GSEA* uses your *ranked* gene list without a hard significance cut-off and asks whether a category's genes concentrate near its top or bottom. Both run from clusterProfiler [12]. GO over-representation and GSEA (and disease ontology) need a Bioconductor `OrgDb`, whose organism coverage must be checked. KEGG needs only a KEGG genome code, and g:Profiler or your own gene sets cover organisms with neither (Section 9.4).

9.1 Map gene IDs and define the gene universe

Enrichment functions expect ENTREZ IDs and a sensible *universe* (all genes eligible for the selection rule in this contrast). Strip Ensembl version suffixes before mapping.

R

```
#### IDs ####
library(clusterProfiler)
library(org.Hs.eg.db)                        # swap for your organism's OrgDb

res_df         <- as.data.frame(res)
res_df$ENSEMBL <- sub("\\..*$", "", rownames(res_df))   # ENSG000..."."7" -> ENSG000...

# map every tested gene to ENTREZ (this becomes the universe)
map_all <- bitr(res_df$ENSEMBL, fromType = "ENSEMBL",
                toType = "ENTREZID", OrgDb = org.Hs.eg.db)
write.csv(map_all, "results/enrichment_mapping_all.csv", row.names = FALSE)
one_target <- !duplicated(map_all$ENSEMBL) & !duplicated(map_all$ENSEMBL, fromLast = TRUE)
map <- map_all[one_target, , drop = FALSE]
res_map <- merge(res_df, map, by = "ENSEMBL")
universe <- unique(res_map$ENTREZID[!is.na(res_map$padj)])

# Exploratory observed-effect screen, not a threshold-null test
sig_entrez <- unique(res_map$ENTREZID[which(res_map$padj < cfg$alpha &
                                      abs(res_map$log2FoldChange) > cfg$lfc_cutoff)])
```

Watch out for

Always pass `universe =` the genes you tested; omitting it substitutes the database background and can change the scientific comparison. Match `keyType` to your IDs, and use the correct `OrgDb`/organism code (`hsa` human, `mmu` mouse, . . .). A small `bitr` drop-rate is normal; a large one requires investigation of species, release coverage and namespace.

The selected list uses the exploratory ordinary-padj plus observed-effect screen from Section 8.3; it is not a threshold-null test or a new gene-level FDR guarantee. Changing that screen changes the ORA question. Preserve the mapped eligible background and report mapping losses for both selected and eligible genes.

9.2 GO and KEGG over-representation (ORA)

R

```
#### GO biological process ####
ego <- enrichGO(gene          = sig_entrez,
                universe      = universe,
                OrgDb         = org.Hs.eg.db,
                keyType       = "ENTREZID",
                ont           = "BP",          # BP / MF / CC
                pAdjustMethod = "BH",
                pvalueCutoff = 0.05,
                qvalueCutoff = 0.20,
                readable      = TRUE)          # show gene SYMBOLs

#### KEGG pathways ####
ekegg <- enrichKEGG(gene         = sig_entrez,
                    organism     = "hsa",
                    universe     = universe,
                    pvalueCutoff = 0.05)
ekegg <- setReadable(ekegg, OrgDb = org.Hs.eg.db, keyType = "ENTREZID")

head(as.data.frame(ego))
```

Key parameters

`ont` selects the GO sub-ontology: BP for biological process, MF for molecular function, or CC for cellular component. `pAdjustMethod = "BH"` applies Benjamini–Hochberg FDR correction. `pvalueCutoff`/`qvalueCutoff` set the significance thresholds. `readable = TRUE` / `setReadable()` convert ENTREZ back to gene symbols.

`minGSSize`/`maxGSSize` bounds apply after usable overlap: very small sets can be driven by a few genes, while broad sets can combine distinct processes. The companion's minimum of 2 serves its toy GMT, and its maximum of 500 limits broad sets; neither is a validated universal choice. Stricter size bounds remove hypotheses rather than necessarily improving them. `pvalueCutoff`/`qvalueCutoff` control reported categories and can hide a tested term; an empty table needs inspection of overlap, size filters and evidence, not an activation claim. [12]

Hypothetical ORA counts. After mapping and the resource's annotation restrictions, suppose 1,000 eligible genes remain, 100 pass the exploratory screen, and a term contains 50 eligible genes, including 10 selected genes. Its `GeneRatio` is 10/100 and `BgRatio` is 50/1000: five overlaps are expected under random selection, so the observed overlap is twice that expectation. These counts alone do not supply an adjusted p-value; the test and correction across tested terms are still required. The denominators must be the actual resource-specific eligible counts.

9.3 GSEA on the ranked gene list

GSEA needs *all* genes ranked by a signed statistic (here raw signed $\log_2$FC; use the same declared metric throughout), named by ENTREZ, sorted decreasing.

R

```
#### Build the ranked list ####
unique_rank <- !duplicated(res_map$ENTREZID) & !duplicated(res_map$ENTREZID, fromLast = TRUE
    ↪ )
ranked <- res_map[unique_rank & is.finite(res_map$log2FoldChange), ]
geneList <- sort(setNames(ranked$log2FoldChange, ranked$ENTREZID), decreasing = TRUE)
# Report excluded ambiguous mappings; do not select an arbitrary duplicate row.

#### Run GSEA (reproducible) ####
set.seed(42)
gse <- gseGO(geneList     = geneList,
             OrgDb        = org.Hs.eg.db,
             ont          = "BP",
             keyType      = "ENTREZID",
             pvalueCutoff = 0.05,
             eps          = 0,              # accurate small p-values
             seed         = TRUE)
```

Watch out for

The ranking metric must be *signed* (keep direction) and the list *complete* (not just significant genes). `set.seed()` together with `seed = TRUE` supports reproducible permutation p-values. `eps = 0` avoids flooring very small p-values.

Raw LFC ranking emphasizes estimated effect magnitude without weighting its precision. Hypothetically, LFC 2 with SE 2 ranks above LFC 1.5 with SE 0.25, although their Wald statistics are 1 and 6. A signed Wald-statistic ranking would reverse their order and answer a different, uncertainty-sensitive question. Declare the metric before interpretation and do not switch it simply to obtain a preferred pathway. [69]

9.4 Enrichment without an OrgDb (non-model organisms)

The routes above need a Bioconductor `OrgDb`, whose coverage depends on the organism and annotation source. Check availability before choosing the route. Three routes cover them, chosen by what the organism has:

1. **A KEGG genome code** (check the KEGG organism catalogue): KEGG over-representation and GSEA run directly on the organism's own gene IDs.
2. **g:Profiler support** (many Ensembl / EnsemblGenomes organisms): GO, KEGG and Reactome through the g:Profiler web service, no OrgDb needed.
3. **Neither**: supply your own gene sets (Section 9.5).

First put the IDs in the namespace each service expects. KEGG uses NCBI Gene IDs or organism-specific locus tags; crops and fungi do not share a universal conversion rule. RefSeq LOC+digits, rice LOC_Os identifiers and fungal locus tags are distinct conventions. Use a reviewed conversion table, declare its namespace and retain unmapped/ambiguous records. [74]

R

```
# A complete reviewed one-to-one map for this example; NA explicitly marks unmapped genes.
res_df <- as.data.frame(res)
res_df$gene_id <- rownames(res_df)
map <- read.delim("config/kegg_mapping.tsv", colClasses = "character")
ids <- map$kegg_id[match(res_df$gene_id, map$gene_id)]
sig <- res_df$gene_id[which(res_df$padj < cfg$alpha & abs(res_df$log2FoldChange) > cfg$lfc_
    ↪ cutoff)]
kegg_ids <- ids[match(sig, res_df$gene_id)]
kegg_ids <- kegg_ids[!is.na(kegg_ids)]
universe_kegg <- unique(ids[!is.na(res_df$padj) & !is.na(ids)])
keep <- !is.na(ids) & is.finite(res_df$log2FoldChange)
ranklist <- sort(setNames(res_df$log2FoldChange[keep], ids[keep]), decreasing = TRUE)
```

KEGG for any organism with a code (rice `osa`, *Fusarium graminearum* `fgr`, . . .):

R

```
ekegg <- enrichKEGG(gene = kegg_ids, organism = "osa", keyType = "ncbi-geneid",
                    universe = universe_kegg, pvalueCutoff = 0.05)
set.seed(42)
kgse <- gseKEGG(geneList = ranklist, organism = "osa", keyType = "ncbi-geneid", # ranklist
    ↪ in same ID namespace
                eps = 0, seed = TRUE, pvalueCutoff = 0.05)
```

g:Profiler for GO/KEGG/Reactome when the organism is supported:

R

```
library(gprofiler2)
gp <- gost(query    = sig,
           organism = "osativa",                    # g:Profiler organism id
           sources = c("GO:BP", "KEGG", "REAC"),
           custom_bg = res_df$gene_id[!is.na(res_df$padj)],             # the tested genes
    ↪ as the background
           user_threshold    = 0.05,
           correction_method = "g_SCS")             # g:Profiler's own multiple-testing
    ↪ correction
head(gp$result[, c("source", "term_name", "p_value", "intersection_size")])
```

Key parameters

`organism = "osa"` specifies the KEGG code; `keyType = "ncbi-geneid"` declares NCBI Gene IDs for this example; use `"kegg"` only for validated native KEGG keys. For g:Profiler, `organism` is the service's own id (e.g. `osativa`); `sources` selects the databases, `custom_bg` supplies the tested-gene background, and `correction_method = "g_SCS"` selects its default correction. If g:Profiler does not recognise your IDs, convert them first with `gconvert()`.

Watch out for

Zero reported terms can be a valid negative result, reflect set-size or significance filters, or reveal an identifier mismatch. Inspect mapping coverage and eligible set sizes before interpretation. KEGG key types are organism-specific; g:Profiler must recognise the submitted namespace (inspect `gconvert()` where needed). Confirm the KEGG organism code and, for STRING later (Section 10), a taxon supported by the selected release. A successful mapping does not guarantee enrichment, and a negative enrichment does not prove that the mapping failed.

9.5 Bring your own gene sets (any organism)

When no database covers your organism, or you have a curated set of your own (a GMT file, or an identifier-to-term table), test it with the same clusterProfiler machinery. It is organism-agnostic and only needs the gene sets and your gene list to share an ID namespace.

This independent branch reloads the saved result table and defines the query, eligible universe and ranking in `custom_cfg$gene_namespace`; Ensembl versions are removed only under the ENSEMBL convention. It uses `config/genesets/example_sets.gmt` from the analysis project, rather than inheriting ENTREZ/KEGG objects. Replace the toy GMT with a reviewed file in that same namespace and record its provenance. A no-overlap guard exposes incompatible identifiers before testing; partial coverage still needs review.

R

```
library(clusterProfiler)
source(file.path(Sys.getenv("BULK_CODE"), "R/contracts.R"))
obj <- read_analysis_object(); custom_res <- obj$res; custom_cfg <- obj$config
ids <- native_gene_ids(rownames(custom_res), custom_cfg$gene_namespace)
sig <- ids[match(significant_ids(custom_res, custom_cfg), rownames(custom_res))]
universe_custom <- ids[!is.na(custom_res$padj)]
rank_custom <- ranked_genes(custom_res, ids)
gmt <- read.gmt("config/genesets/example_sets.gmt")   # term, native gene ID
if (!length(intersect(gmt$gene, universe_custom))) {
  stop("Gene sets and eligible genes have no shared IDs; check the namespace")
}
eora <- NULL
if (length(sig)) eora <- enricher(sig, TERM2GENE = gmt, universe = universe_custom,
    minGSSize = custom_cfg$geneset_min_size, maxGSSize = custom_cfg$geneset_max_size)
set.seed(42)
egse <- GSEA(rank_custom, TERM2GENE = gmt, minGSSize = custom_cfg$geneset_min_size,
             maxGSSize = custom_cfg$geneset_max_size, eps = 0, seed = TRUE)
```

Watch out for

The gene sets and your gene list must use the *same* identifiers; unchecked mismatches can yield empty results; this branch stops when no eligible IDs overlap (Section 9.4). `read.gmt` returns two columns (term, gene); a two-column data frame must put the term first and gene identifier second. The bundled four-gene toy sets use an explicit minimum size of two for demonstration; choose biologically justified production bounds. An empty significant-gene list produces no ORA result and is not a namespace failure. Set a sensible `universe` so the test is not compared against the whole genome.

A hypothetical GSEA row with `NES` $= 1.8$ and adjusted $p = 0.02$ indicates that set members concentrate toward the positive end of this treated-versus-control ranking. NES normalizes the enrichment score relative to its null distribution; it is not a fold change. For positive enrichment, `core_enrichment` identifies the leading-edge members encountered up to the running-score peak, the subset driving that signal. A negative NES points toward the other end. Inspect those genes and the rank metric; enrichment alone does not establish biochemical pathway activation. [12]

9.6 Visualising enrichment

R

```
library(enrichplot)

dotplot(ego, showCategory = 15)                          # top categories: gene ratio + size
barplot(ego, showCategory = 15)                          # counts per category
ego2 <- pairwise_termsim(ego)                            # needed before emapplot
emapplot(ego2, showCategory = 25)                        # network of related terms
cnetplot(ego, showCategory = 5, foldChange = geneList) # genes <-> categories
ridgeplot(gse, showCategory = 15)                        # GSEA: score distributions
gseaplot2(gse, geneSetID = 1:3)                          # GSEA running-score curves
```

Practical note

`dotplot` provides a compact summary (category vs gene ratio, dot size = gene count, colour = adjusted p). `emapplot` groups redundant GO terms into themes; `cnetplot` shows which genes drive which categories.

9.7 Other gene-set resources and length bias

GO and KEGG are starting points. The same selected-gene set and ranked list feed several other widely used resources, all driven from the objects built above (`sig_entrez`, `universe`, `geneList`).

R

```
#### MSigDB Hallmark sets + fast pre-ranked GSEA (fgsea) ####
library(msigdbr); library(fgsea)
h        <- msigdbr(species = "Homo sapiens", collection = "H")     # 50 Hallmark gene sets
pathways <- split(h$ncbi_gene, h$gs_name)                            # sets as lists of
    ↪ ENTREZ ids
fgseaRes <- fgsea(pathways = pathways, stats = geneList)             # geneList = signed,
    ↪ ranked, named

#### Reactome pathways ####
library(ReactomePA)
react <- enrichPathway(gene = sig_entrez, universe = universe,
                       organism = "human", readable = TRUE)
```

R

```
#### Any MSigDB collection through clusterProfiler (e.g. C2 canonical pathways) ####
library(clusterProfiler)
c2    <- msigdbr(species = "Homo sapiens", collection = "C2", subcollection = "CP:REACTOME")
emsig <- enricher(sig_entrez, TERM2GENE = c2[, c("gs_name", "ncbi_gene")],
                  universe = universe)
```

Key parameters

`msigdbr` ships MSigDB as a tidy table [44]: `H` Hallmark (broad, denoised processes), `C2` curated pathways (Reactome/KEGG/BioCarta), `C5` ontologies (GO). `fgsea` runs fast permutation GSEA on a named, signed, ranked statistic [45]. `enricher`/`GSEA` apply ORA/GSEA to any custom `TERM2GENE` table. Recent `msigdbr` (≥ 10) renamed `category`/`subcategory` to `collection`/`subcollection` and the column `entrez_gene` to `ncbi_gene`; on older installs use the former (check `colnames()`).

Watch out for

Gene-length bias. Read counts depend on transcript length, so longer genes can reach significance more easily; a plain GO over-representation test can consequently over-report categories rich in long genes. When this matters, use `goseq`, which weights the test by a length-based probability (`nullp()` → `goseq()`) [47]. Ranked methods avoid a hard DE-list cutoff, but their ranking statistics and annotation coverage can still introduce bias; they are not a general length-bias correction.

9.8 Per-sample gene-set activity (GSVA) Optional

The tests above ask whether a gene set differs *between groups*. GSVA instead scores each *sample* against each gene set, turning the expression matrix into a set × sample activity matrix that you can correlate with a covariate, cluster, or model with a suitable continuous-score method such as limma, or use in plotting [48]. It is descriptive, not a significance test.

R

```r
library(GSVA)
mat <- assay(vsd)                               # a normalised matrix (samples in columns)
gene_sets <- read.gmt("sets/my_pathways.gmt") # two columns: term, gene
gene_sets <- split(gene_sets$gene, gene_sets$term)   # -> a named list of gene-id vectors

par    <- gsvaParam(mat, gene_sets, kcdf = "Gaussian", minSize = 2, maxSize = 500)
scores <- gsva(par)                             # gene sets x samples
write.csv(scores, "results/gsva_scores.csv")

# heatmap of the most variable pathways
library(pheatmap)
v <- apply(scores, 1, var)
sel <- head(order(v, decreasing = TRUE), 40)
pheatmap(scores[sel, , drop = FALSE], scale = "row", cluster_rows = length(sel) > 1)
```

Key parameters

`gsvaParam(...)` then `gsva()` is the current two-step GSVA API (version ≥ 1.50; earlier versions call `gsva()` directly). `kcdf = "Gaussian"` for a log-scale continuous matrix (VST, log-CPM, log2 intensity); use `"Poisson"` for raw integer counts. `minSize`/`maxSize` bound gene-set size.

Watch out for

A GSVA score is a per-sample activity, not a raw count or a differential test. Do not feed it into DESeq2; to compare groups, run limma or a t-test on the score matrix afterwards. The gene sets must share the matrix's identifier namespace, or no usable overlap may remain (Section 9.4). Match `kcdf` to the data scale.

10 Protein interaction networks (STRING)

STRING places selected genes in a functional-association network, where connected modules and highly connected nodes can be examined [64]. Map the selected list from Section 8 through the online service to construct the graph. This uses the same exploratory adjusted-p-value and observed-effect screen as enrichment; it is not a threshold-null test.

R

```
library(STRINGdb); library(igraph)
sig_sym <- res_map$symbol[which(res_map$padj < cfg$alpha &
                                abs(res_map$log2FoldChange) > cfg$lfc_cutoff)]
sig_sym <- unique(sig_sym[!is.na(sig_sym)])

# Use a taxon supported by the selected STRING release (see the watchbox).
sdb    <- STRINGdb$new(version = "12.0", species = 9606, score_threshold = 400)
mapped <- sdb$map(data.frame(gene = sig_sym), "gene", removeUnmappedRows = TRUE)
edges <- sdb$get_interactions(mapped$STRING_id)
```

R

```
# build the graph, find modules and hubs
weighted <- data.frame(from = edges$from, to = edges$to, weight = edges$combined_score /
    ↪ 1000)
g <- graph_from_data_frame(weighted, directed = FALSE,
                           vertices = data.frame(name = unique(mapped$STRING_id)))
g <- simplify(g, edge_attr_combine = list(weight = "max", "ignore"))

set.seed(42)
comm <- cluster_louvain(g)                    # functional modules
V(g)$module <- membership(comm)
V(g)$degree <- degree(g)
hubs <- names(sort(V(g)$degree, decreasing = TRUE))[1:15]
```

R

```
# export for Cytoscape, then draw a figure
write_graph(g, "results/ppi/string_ppi.graphml", format = "graphml")

library(ggraph)
set.seed(42)
p <- ggraph(g, layout = "fr") +
       geom_edge_link(alpha = 0.3) +
       geom_node_point(aes(size = degree, colour = factor(module))) +
       theme_void()
ggsave("results/ppi/string_ppi.png", p, width = 7, height = 6, dpi = 300)
ggsave("results/ppi/string_ppi.svg", p, width = 7, height = 6)
```

Key parameters

`version = "12.0"` pins the STRING release; `species` specifies the NCBI taxid; `score_threshold = 400` keeps medium-confidence edges (700 for high). `combined_score / 1000` rescales STRING's 0–1000 score to a 0–1 edge weight. `cluster_louvain` finds modules by modularity; node `degree` ranks hubs.

Raising the STRING threshold from 400 to 700 keeps fewer supported associations for the same mapped seeds; degrees, connected components and modules can change, and isolated seeds need not disappear. It trades coverage for evidence confidence rather than proving a more biologically valid network. Record the threshold alongside the release. [75]

Inputs and outputs

In: the selected-gene symbols. **Out:** `results/ppi/string_ppi.graphml` (open in Cytoscape [67]) and a PNG/SVG network figure.

Watch out for

Use a taxon supported by the selected release. STRING keys strains separately, and a species-level id can return a 404: for *Fusarium graminearum* PH-1 use **229533**, not the species 5518. For a genome with no gene symbols (RefSeq locus tags), seed on the gene ID instead of the symbol. STRING is a live web service, so pin its `version` and cache the mapped edges for reproducibility. Preserve exact identifier mappings; do not case-fold symbols universally. The default network represents functional associations and can include indirect evidence. Degree ranks nodes only within the selected induced graph; it does not establish causal importance. Retain mapped seeds with no edges and report mapping losses. [75]

11 Figures for transcriptomics

The count-route examples below use `dds`, `res`, `resLFC`, and `vsd`. The companion scripts read a common result/expression/metadata handoff, allowing log2 microarray expression in sample-structure plots and imported DE tables in volcano plots. Results-only input cannot supply expression-based figures; MA requires a compatible mean-abundance measure: DESeq2 baseMean is shown on a log axis, whereas limma AveExpr and edgeR logCPM are already logged. Each is exported as both PNG (raster, for slides) and SVG (vector, for editing) at the end; titles are kept out of the figure and belong in the caption.

11.1 PCA (sample structure)

Principal component analysis (PCA) on the most variable genes in the VST output shows sample separation by condition and helps investigate outliers or batch effects.

R

```
#### PCA ####
library(ggplot2)
pca <- plotPCA(vsd, intgroup = "condition", ntop = 500, returnData = TRUE)
pv <- round(100 * attr(pca, "percentVar"))

p_pca <- ggplot(pca, aes(PC1, PC2, colour = condition)) +
  geom_point(size = 3, alpha = 0.9) +
  ggrepel::geom_text_repel(aes(label = name), size = 3, seed = 1, show.legend = FALSE) +
  labs(x = paste0("PC1 (", pv[1], "%)"),
       y = paste0("PC2 (", pv[2], "%)"), colour = NULL) +
  coord_fixed() +
  theme_bw(base_size = 12)

ggsave("results/pca.png", p_pca, width = 6, height = 5, dpi = 300)
ggsave("results/pca.svg", p_pca, width = 6, height = 5)
```

Key parameters

`ntop = 500` uses the 500 most variable genes. `returnData = TRUE` hands back a data frame so you can fully restyle the plot. `percentVar` (an attribute) gives the variance explained per axis. `coord_fixed()` keeps the axes on a comparable scale.

Watch out for

Run PCA on `vst/rlog` output, never raw counts. If PC1 separates by batch rather than condition, include an identifiable batch term in the DE model. This adjusts inference but does not remove batch variation from VST/PCA. Any display-only batch removal needs a design that preserves condition. [69]

`ntop = 500` is a teaching choice for the most variable VST genes. A smaller set emphasizes fewer dominant patterns; a larger set introduces additional, sometimes weak, variation and can rotate the axes or change explained variance. PCA feature selection changes this diagnostic view, not the fitted DE model. Compare plausible feature sets and metadata before explaining a separation.

11.2 UMAP (optional, many-sample structure)

UMAP is a non-linear embedding [21]. It is informative only with *many* samples; for typical small bulk designs, PCA and clustering are more reliable.

R

```
#### UMAP (uwot) ####
library(uwot)
library(matrixStats)

v   <- rowVars(assay(vsd))
sel <- head(order(v, decreasing = TRUE), 1000)    # top 1000 variable genes
mat <- t(assay(vsd)[sel, ])                   # samples x genes

set.seed(42)
um <- umap(mat, n_neighbors = 15, min_dist = 0.1, metric = "euclidean")

um_df <- data.frame(UMAP1 = um[, 1], UMAP2 = um[, 2],
                    condition = vsd$condition, sample = colnames(vsd))
ggplot(um_df, aes(UMAP1, UMAP2, colour = condition)) +
  geom_point(size = 3) + theme_bw(base_size = 12)
```

Watch out for

`n_neighbors` **must be smaller than the number of samples**; with only a handful of samples UMAP is unstable and misleading, so prefer PCA here. Always `set.seed()`: UMAP is stochastic. `min_dist` controls how tightly points pack.

11.3 Sample-distance heatmap & dendrogram

A heatmap of sample-to-sample distances (with the matching dendrogram) is an overview of sample relationships under the chosen distance and linkage.

R

```
#### Sample distances ####
library(pheatmap)
library(RColorBrewer)

sampleDists <- dist(t(assay(vsd)), method = "euclidean") # between samples
mat         <- as.matrix(sampleDists)
rownames(mat) <- colnames(mat) <- paste(vsd$condition, colnames(vsd), sep = " | ")
cols        <- colorRampPalette(rev(brewer.pal(9, "Greys")))(255)

pheatmap(mat,
         clustering_distance_rows = sampleDists,   # cluster on the SAME metric
         clustering_distance_cols = sampleDists,
         clustering_method        = "ward.D2",
         col = cols)

# stand-alone dendrogram
plot(hclust(sampleDists, method = "ward.D2"),
     labels = colnames(vsd), main = "", xlab = "", sub = "")
```

Key parameters

`dist(t(assay(vsd)))` computes Euclidean distances between samples; transposition puts samples in rows. `clustering_distance_* = sampleDists` passes the *precomputed* distance object so heatmap and dendrogram agree. `clustering_method = "ward.D2"` selects a commonly used linkage that forms compact clusters.

11.4 MA plot (effect size vs expression)

The MA plot shows $\log_2$FC against mean expression; shrinkage reduces the extent to which low-count genes dominate its edges.

R

```r
#### MA plot ####
ma_display <- res                              # original inference and missingness
ma_display$log2FoldChange <- resLFC$log2FoldChange[match(rownames(res), rownames(resLFC))]
plotMA(ma_display, ylim = c(-4, 4), alpha = cfg$alpha,
       colSig = "black", colNonSig = "grey70")     # base-R version
abline(h = c(-1, 1), col = "grey50", lty = 2)

# ggplot version
ma <- as.data.frame(ma_display)
ma <- ma[!is.na(ma$padj), ]
ma$sig <- ma$padj < cfg$alpha
ggplot(ma, aes(baseMean, log2FoldChange, colour = sig)) +
  geom_point(size = 1, alpha = 0.6) +
  geom_hline(yintercept = 0, colour = "grey40") +
  scale_x_log10() +
  scale_colour_manual(values = c("FALSE" = "grey75", "TRUE" = "black"),
                      name = paste("padj <", cfg$alpha)) +
  labs(x = "mean of normalised counts", y = "log2 fold change") +
  theme_bw(base_size = 12)
```

Watch out for

Build the display from `res`, preserving its adjusted p-values and NA state, and import only the matching shrunken LFC by gene ID. This reduces extreme low-count display effects without changing the declared test. The x-axis shows mean normalized counts on a log scale.

11.5 Volcano plot (significance vs effect size)

A volcano plot displays effect size and statistical evidence: $\log_2$FC on x, $-\log_{10}$ adjusted p on y, with selected genes labelled. This is the most detailed recipe in the worksheet: `ggrepel` keeps labels readable and `ggtext` typesets proper subscripts/italics.

R

```r
#### Volcano plot (ggplot2 + ggrepel + ggtext) ####
library(ggplot2)
library(ggrepel)
library(ggtext)

res_df              <- as.data.frame(res)              # original inference table
res_df$raw_lfc      <- res_df$log2FoldChange
res_df$log2FoldChange <- resLFC$log2FoldChange[match(rownames(res_df), rownames(resLFC))]
res_df$gene         <- rownames(res_df)
res_df              <- res_df[!is.na(res_df$padj), ]   # drop NA padj (filtered)
res_df$neglog10padj <- -log10(pmax(res_df$padj, .Machine$double.xmin))

# exploratory screen using original adjusted p-values and raw effects
res_df$direction <- "Other"
res_df$direction[res_df$padj < cfg$alpha & res_df$raw_lfc > cfg$lfc_cutoff] <- "Up"
res_df$direction[res_df$padj < cfg$alpha & res_df$raw_lfc < -cfg$lfc_cutoff] <- "Down"

# label the 15 lowest-padj genes passing the exploratory screen
lab <- res_df[res_df$direction != "Other", ]
lab <- head(lab[order(lab$padj), , drop = FALSE], 15)

# greyscale-friendly: encode direction by BOTH colour and shape
pal <- c(Up = "black", Down = "grey55", "Other" = "grey80")
shp <- c(Up = 17,      Down = 16,       "Other" = 16)

p_volcano <- ggplot(res_df, aes(log2FoldChange, neglog10padj)) +
  geom_hline(yintercept = -log10(cfg$alpha), linetype = "dashed",
             colour = "grey60", linewidth = 0.3) +
  geom_point(aes(colour = direction, shape = direction),
             size = 1.6, alpha = 0.8) +
  geom_text_repel(data = lab, aes(label = gene),
                  size = 3, fontface = "italic",
                  box.padding = 0.4, point.padding = 0.2,
                  min.segment.length = 0, max.overlaps = Inf,
                  segment.size = 0.3, segment.colour = "grey50",
                  seed = 42) +
  scale_colour_manual(values = pal, name = NULL) +
  scale_shape_manual(values = shp, name = NULL) +
  scale_x_continuous(breaks = scales::pretty_breaks(6)) +
  labs(x = "shrunken log<sub>2</sub> fold change",
       y = "-log<sub>10</sub> adjusted *p*") +
  theme_classic(base_size = 12) +
  theme(axis.title.x = element_markdown(),
        axis.title.y = element_markdown(),
        legend.position = "top")

ggsave("results/volcano.png", p_volcano, width = 6, height = 5, dpi = 300)
ggsave("results/volcano.svg", p_volcano, width = 6, height = 5)
```

Key parameters

ggrepel: `box.padding`/`point.padding` set spacing around labels and points; `min.segment.length = 0` always draws a leader line; `max.overlaps = Inf` keeps every requested label; `seed` fixes the label layout for reproducible figures. **ggtext:** `element_markdown()` renders the `<sub>` subscripts and `*...*` italics in the axis titles. **scales:** `pretty_breaks(6)` chooses clean x ticks. **encoding:** colour and shape both mark direction, so the figure survives grayscale printing.

Watch out for

The Up/Down labels mark the exploratory joint screen on original `padj` and raw LFC, not a threshold-null test or an additional FDR guarantee; Other means the screen was not passed and can include significant but small effects. Shrinkage changes only plotted positions, so raw-effect cutoffs do not define vertical boundaries here. Keep original p-values and missingness for plotting membership. Cap labels (here 15); for `EnhancedVolcano`, pass the same inference-preserving display table rather than obtaining p-values from an independently generated shrinkage table. [18]

11.6 Heatmap of top differentially-expressed genes

A z-scored heatmap of the top genes shows expression *patterns* across samples and groups them by similarity.

R

```r
#### Top-DE-gene heatmap (z-scored VST) ####
library(pheatmap)

ids <- rownames(res)[which(res$padj < cfg$alpha)]
ids <- head(ids[order(res[ids, "padj"])], 30)
mat <- assay(vsd)[ids, , drop = FALSE]
mat <- mat[apply(mat, 1, var) > 0, , drop = FALSE]
stopifnot(nrow(mat) > 0)
mat <- t(scale(t(mat)))                      # z-score per gene (row)
ann <- as.data.frame(colData(dds)[, "condition", drop = FALSE])

pheatmap(mat,
         scale            = "none",           # already z-scored
         annotation_col   = ann,
         show_rownames    = TRUE,
         clustering_method = "ward.D2", cluster_rows = nrow(mat) > 1,
         color = colorRampPalette(c("#2C7BB6", "white", "#C0392B"))(255),
         border_color = NA, fontsize_row = 7)
```

Practical note

For more control (multiple annotation tracks, split rows/columns), use `ComplexHeatmap` [20]: `Heatmap(mat, name = "z", col = circlize::colorRamp2(c(-2,0,2), c("#2C7BB6","white","#C0392B")), top_annotation = HeatmapAnnotation(condition = vsd$condition))`.

Watch out for

Z-score per gene (across samples) so genes on different expression scales become comparable; with data already scaled, set `scale = "none"` to avoid double scaling. Keep the gene set small enough to read (here 30); use a diverging palette centred at zero.

A row z score centers and divides each gene's transformed values by its own sample standard deviation. Hypothetical VST values 8, 10 and 12 have mean 10 and sample SD 2, giving z scores $-1, 0, 1$. Thus a value of 1 means one within-gene SD above its mean, not a two-fold change or higher expression than another gene. Because the top genes were chosen using this same condition comparison, their heatmap is descriptive and its group separation is not independent validation. The code selects by ordinary adjusted p-value only; it does not apply the extra observed-effect screen.

Focused heatmap (genes of interest). To inspect a curated set (a pathway, marker panel, or candidate list) instead of the top-DE genes, swap the `top` selection for your own IDs and keep their order.

R

```
#### Heatmap of a chosen gene panel ####
goi <- c("CD8A", "GZMB", "IFNG", "FOXP3", "IL2RA")          # genes of interest (symbols)
ids <- rownames(res)[match(goi, res$symbol)]                # res$symbol from the annotation
    ↪ step
ids <- ids[!is.na(ids)]
m   <- scaled_rows(assay(vsd)[ids, , drop = FALSE])                      # z-score per gene
    ↪ (row)
rownames(m) <- res$symbol[match(rownames(m), rownames(res))] # label rows by symbol
pheatmap(m, scale = "none", annotation_col = ann,
         cluster_rows = FALSE,                              # preserve the panel's order
         color = colorRampPalette(c("#2C7BB6", "white", "#C0392B"))(255),
         border_color = NA)
```

12 Supplementary

12.1 Beyond this worksheet: *de novo* assembly and variant calling

Two adjacent tasks come up often but sit outside the reference-based, gene-level scope here.

No reference genome? Assemble transcripts *de novo*. Trinity is an established RNA-seq assembler and rnaSPAdes is a common alternative [52]; MEGAHIT is fast but is a metagenome/genome assembler, not RNA-seq-specific [53]. The usual route is: assemble, predict ORFs with TransDecoder, quantify with Salmon against the assembly, then annotate (e.g. Trinotate).

Shell

```
# Trinity (paired-end)
Trinity --seqType fq --left trimmed/R1.fq.gz --right trimmed/R2.fq.gz \
  --max_memory 20G --CPU 8 --output trinity_out
```

Variants from RNA-seq. You can call SNPs/indels from the BAM, but the naive `mpileup` route is not appropriate for RNA-seq. The following DNA-style syntax is an illustration of the commands, not a validated RNA-seq variant pipeline:

Shell

```
bcftools mpileup -f ref/genome.fna mapped/SRR1234567.sorted.bam \
  | bcftools call -mv -Ob -o calls.bcf
bcftools view calls.bcf | less
```

Watch out for

For trustworthy RNA-seq variants follow the GATK best practice: STAR 2-pass, mark duplicates, then `SplitNCigarReads` (which splits reads at the N CIGAR gaps that introns create and remaps STAR's MAPQ 255 to 60 so the caller keeps them) before `HaplotypeCaller` [54]. Splice junctions, allele-specific expression, and RNA editing (A-to-I, read as A $\rightarrow$ G) all confound a plain DNA-style caller.

12.2 Extended QC and companion tools

Additional command-line tools provide quality checks used in workflows and evaluations. Check platform-specific installation and MultiQC module support for each chosen tool; not every output is parsed automatically.

Tool	What it adds (and the biology it reports)	Typical call
SAMtools [8]	Mapping rate (`flagstat`); per-chromosome read counts, including the mitochondrial fraction (`idxstats`); summary stats (`stats`); duplicate marking (`markdup`).	`samtools idxstats`
Picard [36]	`MarkDuplicates` (PCR/optical duplicates, high in low-input libraries); `CollectRnaSeqMetrics` (% coding/UTR/intronic/intergenic and 5'/3' bias).	`picard CollectRnaSeqMetrics`
RSeQC [34]	Strandedness (`infer_experiment.py`); read distribution over exons/introns/UTRs; gene-body 5'→3' coverage evenness, which can flag RNA degradation.	`read_distribution.py`
Qualimap [35]	One-shot HTML RNA-seq QC: genomic origin of reads, coverage bias, transcript saturation.	`qualimap rnaseq`
preseq [37]	Library-complexity / saturation curve: would deeper sequencing recover more distinct molecules?	`preseq lc_extrap`
deepTools [38]	Normalised bigWig coverage tracks for genome browsers (`bamCoverage`); sample-to-sample correlation (`multiBamSummary` + `plotCorrelation`).	`bamCoverage`
gffread [39]	GTF/GFF conversion; extract a transcript FASTA for Salmon or kallisto.	`gffread -w tx.fa`

Practical note

Gene-body coverage (RSeQC `geneBody_coverage.py`) and `CollectRnaSeqMetrics` provide quick checks for degraded RNA: partially degraded poly-A libraries can show a 3' coverage skew because reverse transcription primes from the poly-A tail.

Watch out for

Unlike DNA-seq, standard RNA-seq differential expression does not deduplicate reads: a highly expressed gene genuinely yields many identical fragments, so duplicate rates can reflect genuine expression as well as technical amplification. Assess expression dependence with dupRadar; UMI libraries require a suitable molecule-aware deduplication method. Do not remove non-UMI duplicates automatically without a study-specific justification.

12.3 Looping & batch processing

The body ran one sample. In practice, derive a sample name with `basename` and loop, keeping *threads, reference, and paths* in variables at the top so the script is portable.

Shell

```
#### Parameters (edit once) ####
THREADS=8
REF=ref/genome.fna
GTF=ref/genome.gtf

#### fastp over every paired sample ####
for r1 in raw_data/*_1.fastq.gz; do
  s=$(basename "$r1" _1.fastq.gz)          # sample name from the R1 file
  echo ">> trimming $s"
  fastp \
     -i raw_data/${s}_1.fastq.gz -I raw_data/${s}_2.fastq.gz \
     -o trimmed/${s}_1.trim.fq.gz -O trimmed/${s}_2.trim.fq.gz \
     --detect_adapter_for_pe -q 15 -l 36 \
     -h qc/post_trim/${s}.html -j qc/post_trim/${s}.json \
     --thread "$THREADS"
done
```

Shell

```
# Same idea with GNU parallel (run 2 samples at a time, 8 threads each)
ls raw_data/*_1.fastq.gz | sed 's#_1.fastq.gz##' | \
  parallel -j 2 'fastp -i {}_1.fastq.gz -I {}_2.fastq.gz \
    -o trimmed/$(basename {})_1.trim.fq.gz -O trimmed/$(basename {})_2.trim.fq.gz \
    --detect_adapter_for_pe -q 15 -l 36 --thread 8'
```

Watch out for

Do not oversubscribe the CPU: *parallel jobs* × *threads per job* should not exceed the number of cores. STAR is memory-hungry, so run *one* STAR job at a time unless you have RAM for several genome indices in memory.

12.4 Threads, memory & critical parameters (quick reference)

Tool	Key flags	Note
fasterq-dump	`-e` <threads>, `-t` <tmp>	temp needs ∼10× the .sra size
FastQC	`-t` <n>	one thread per file
fastp	`--thread` (≤64 in v1.0.1), `-q` `-l`, `-g`	`-g` for NextSeq/NovaSeq poly-G
SortMeRNA	`--threads`, `--out2`, `--paired_in/out`	use separate workdirs per sample
STAR index	`--runThreadN`, `--sjdbOverhang`, `--genomeSAindexNbases`	overhang = readlen−1; reduce SA index for small genomes; ∼30 GB RAM (human)
STAR align	`--runThreadN`, `--readFilesCommand` zcat, `--outSAMtype` BAM SortedByCoordinate	prefix needs a trailing path separator
featureCounts	`-T`, `-p` `--countReadPairs`, `-s` 0/1/2	set `-s` correctly (usually 2)

12.5 Reproducibility

Recording versions and sessions allows a colleague to reconstruct the environment and assess agreement. Pin versions, record sessions, and seed every stochastic step. Seeds alone do not guarantee numerical identity across versions or platforms.

Shell

```
# Capture the exact tool versions of the conda environment
conda env export -n rnaseq > environment.lock.yml      # recreate with: conda env create -f
    ↪ environment.lock.yml

# A lighter, human-readable record of the key tools
conda list -n rnaseq | grep -E "star|fastp|subread|samtools|sortmerna|sra-tools|multiqc"
```

R

```
#### Record the R environment + fix seeds ####
set.seed(42)              # before UMAP / GSEA / anything random
writeLines(capture.output(sessionInfo()), "results/sessionInfo.txt")
# For full project-level R reproducibility, consider renv::init() / renv::snapshot()
```

Practical note

Keep `environment.yml`, `sessionInfo.txt`, your scripts, and a short README under version control (`git`). Pin tool versions explicitly rather than relying on "latest": results can shift between releases of aligners and annotation databases. Record the exact genome/GTF build (source, release, date) alongside the code.

12.6 Troubleshooting

Symptom	Likely cause	Fix
STAR: `genomeSAindexNbases` too large	Default 14 used on a small genome	Recompute `--genomeSAindexNbases` from genome length
Very low featureCounts `Assigned` %	Wrong strandedness or annotation mismatch	Check protocol and forward/reverse evidence; do not maximise over 0/1/2
SortMeRNA reports existing state	Reused work directory	Inspect prior state; use a new per-sample workdir
fasterq-dump runs out of space	Temp dir too small	Point `-t` at a disk with ∼10× the .sra size
Many reads unmapped in STAR	Genome/GTF mismatch or residual adapters	Verify matching builds; re-check fastp/MultiQC
DESeq2: *every gene has at least one zero*	Cannot compute size factors (sparse data)	Assess sparsity and design; `type="poscounts"` is an explicit alternative normalization choice
`bitr` drops most genes	Wrong `keyType` / ID format	Strip Ensembl versions; check ID type and `OrgDb`

12.7 Sample correlations and sensitivity checks

Pearson correlation compares linear similarity of sample expression profiles; Spearman correlation compares their ranks. Use the same finite transformed-expression matrix as for PCA, and inspect sample labels and known covariates. The companion's `07_figures/05_sample_correlations.R` exports both matrices from a gene-by-sample TSV. High sample correlation can coexist with a strong treatment effect or a sample swap; correlation alone neither confirms identity nor validates a model. This is sample-level quality assessment, not a gene coexpression network such as WGCNA.

An unpaired Wilcoxon rank-sum test on transformed expression can provide a limited exploratory sensitivity check for two independent groups. It does not preserve batch adjustment, pairing or the count mean–variance model. Use `stats::wilcox.test` only when its sampling assumptions match the question; declare tie handling and adjust across every tested gene. Disagreement with DESeq2 calls for

examination of the design and data, not majority voting between tests. No Wilcoxon phase is substituted for the primary model here.

12.8 Cross-study synthesis

A joint model such as `~ dataset + condition` estimates a common condition contrast while accounting for an additive study term. It requires an identifiable design and comparable measurement/annotation. It cannot recover a condition completely confounded with study. A study-of-origin field must represent independent studies, not sequencing lanes or duplicate deposits. Review overlapping donors and controls before combining public datasets; reused samples invalidate an assumption of independent studies.

Meta-analysis instead fits each study separately and combines compatible statistics. `HTSFilter` provides a filtering method for replicated count data; `metaRNASeq` implements p-value combination, including inverse-normal combination [79]. Combining evidence does not estimate a shared effect magnitude or prove replication in every study. Concordant signs do not establish that each study is significant. A direction-selected family needs a justified directional testing procedure; do not assume ordinary BH control follows merely from selecting matching signs.

For effect synthesis, `metafor::rma.uni` combines unshrunken log2 effects with their standard errors. Common-effect (`FE`) and random-effects (for example `REML` or `DL`) models make different assumptions; random effects do not repair incompatible biology or dependent studies. With few studies, heterogeneity and uncertainty are difficult to estimate. Choose the target model before looking at significance and inspect study-level effects and intervals.

`08_cross_study/01_pool_effects.R` starts from reviewed study-level tables, not raw reads. Its TSV contains `gene_id`, `study`, `numerator`, `denominator`, `log2FoldChange` and `lfcSE`. All studies must use the same organism, gene namespace, contrast direction and log2 scale; standard errors must correspond to the unshrunken estimate. Provide the complete reviewed common tested gene set, not a union of significant lists. The script rejects missing study rows, duplicates, incompatible directions and nonpositive standard errors. Its two-sided pooled tests receive BH adjustment across all supplied genes; the sign-agreement column is descriptive. FE runs in base R; REML/DL require `metafor`, and this teaching route requires at least three studies for them.

FE/REML/DL passed independent synthetic checks with R 4.6.1 and `metafor` 5.2-1 on native macOS aarch64; real-data and full-stack validation remain untested.

12.9 Annotation transfer and unavailable results

For organisms without a curated OrgDb, imported GO/KEGG or STRING orthology-transferred annotations can be represented as `TERM2GENE` or GMT and used by the custom enrichment route. Retain the taxon, release, source evidence, orthology relation and all mapping losses. Transferred membership is a hypothesis based on the mapping, not direct experimental annotation in the target species. MSigDB sets obtained through `msigdbr` also require a declared species and collection; cross-species projection changes the interpretation. A set overlap is descriptive unless a declared tested universe and inferential model support a p-value.

Distinguish a successful enrichment calculation with zero retained terms from an unavailable result caused by missing mapping, unsupported species, a network-service failure or a failed model. A blank table or placeholder figure cannot distinguish them. Keep an execution-status record alongside the complete tested table, effective annotated background and mapping ledger. Report unavailable analyses as unavailable; do not interpret failure as absence of pathway involvement.

Shrinkage for general contrasts. `apeglm` shrinks a selected model coefficient; `ashr` is an alternative shrinkage method usable with supported general contrasts through DESeq2's `lfcShrink`. The companion uses its declared coefficient with `apeglm`; a complex contrast requires explicit implementation and review, not changing the displayed label alone. Shrinkage remains distinct from the primary hypothesis test [69].

References

[1] Leinonen R, Sugawara H, Shumway M; INSDC. The Sequence Read Archive. *Nucleic Acids Res.* 2011;39(Database issue):D19–D21. https://doi.org/10.1093/nar/gkq1019

[2] Andrews S. *FastQC: A Quality Control Tool for High Throughput Sequence Data.* Babraham Bioinformatics; 2010. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

[3] Ewels P, Magnusson M, Lundin S, Käller M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics.* 2016;32(19):3047–3048. https://doi.org/10.1093/bioinformatics/btw354

[4] Chen S, Zhou Y, Chen Y, Gu J. fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics.* 2018;34(17):i884–i890. https://doi.org/10.1093/bioinformatics/bty560

[5] Kopylova E, Noé L, Touzet H. SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. *Bioinformatics.* 2012;28(24):3211–3217. https://doi.org/10.1093/bioinformatics/bts611

[6] Bushnell B. *BBMap: A Fast, Accurate, Splice-Aware Aligner.* Lawrence Berkeley National Laboratory; 2014. LBNL-7065E.

[7] Dobin A, Davis CA, Schlesinger F, et al. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics.* 2013;29(1):15–21. https://doi.org/10.1093/bioinformatics/bts635

[8] Danecek P, Bonfield JK, Liddle J, et al. Twelve years of SAMtools and BCFtools. *GigaScience.* 2021;10(2):giab008. https://doi.org/10.1093/gigascience/giab008

[9] Liao Y, Smyth GK, Shi W. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics.* 2014;30(7):923–930. https://doi.org/10.1093/bioinformatics/btt656

[10] Love MI, Huber W, Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.* 2014;15(12):550. https://doi.org/10.1186/s13059-014-0550-8

[11] Zhu A, Ibrahim JG, Love MI. Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences. *Bioinformatics.* 2019;35(12):2084–2092. https://doi.org/10.1093/bioinformatics/bty895

[12] Wu T, Hu E, Xu S, et al. clusterProfiler 4.0: A universal enrichment tool for interpreting omics data. *The Innovation.* 2021;2(3):100141. https://doi.org/10.1016/j.xinn.2021.100141

[13] Yu G. *enrichplot: Visualization of Functional Enrichment Result.* R/Bioconductor package. https://doi.org/10.18129/B9.bioc.enrichplot

[14] Carlson M. *org.Hs.eg.db: Genome wide annotation for Human.* R/Bioconductor package. https://doi.org/10.18129/B9.bioc.org.Hs.eg.db

[15] Wickham H. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York; 2016. https://doi.org/10.1007/978-3-319-24277-4

[16] Slowikowski K. *ggrepel: Automatically Position Non-Overlapping Text Labels with ggplot2.* R package. https://cran.r-project.org/package=ggrepel

[17] Wilke CO, Wiernik BM. *ggtext: Improved Text Rendering Support for ggplot2.* R package. https://cran.r-project.org/package=ggtext

[18] Blighe K, Rana S, Lewis M. *EnhancedVolcano: Publication-ready volcano plots with enhanced colouring and labeling.* R/Bioconductor package. https://doi.org/10.18129/B9.bioc.EnhancedVolcano

[19] Kolde R. *pheatmap: Pretty Heatmaps.* R package. https://doi.org/10.32614/CRAN.package.pheatmap

[20] Gu Z, Eils R, Schlesner M. Complex heatmaps reveal patterns and correlations in multidimensional genomic data. *Bioinformatics.* 2016;32(18):2847–2849. https://doi.org/10.1093/bioinformatics/btw313

[21] McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426; 2018. (R: uwot, Melville J.)

[22] Soneson C, Love MI, Robinson MD. Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences. *F1000Research.* 2015;4:1521. https://doi.org/10.12688/f1000research.7563.2

[23] Grüning B, Dale R, Sjödin A, et al. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat Methods.* 2018;15(7):475–476. https://doi.org/10.1038/s41592-018-0046-7

[24] Conesa A, Madrigal P, Tarazona S, et al. A survey of best practices for RNA-seq data analysis. *Genome Biol.* 2016;17:13. https://doi.org/10.1186/s13059-016-0881-8

[25] Stark R, Grzelak M, Hadfield J. RNA sequencing: the teenage years. *Nat Rev Genet.* 2019;20(11):631–656. https://doi.org/10.1038/s41576-019-0150-2

[26] Ewels PA, Peltzer A, Fillinger S, et al. The nf-core framework for community-curated bioinformatics pipelines. *Nat Biotechnol.* 2020;38(3):276–278. https://doi.org/10.1038/s41587-020-0439-x

[27] ENCODE Project Consortium. *ENCODE Guidelines and Best Practices for RNA-Seq.* ENCODE Data Coordinating Center. https://www.encodeproject.org/data-standards/rna-seq/long-rnas/

[28] Kans J. *Entrez Direct: E-utilities on the Unix Command Line.* In: Entrez Programming Utilities Help. Bethesda (MD): NCBI; 2013 (updated regularly). https://www.ncbi.nlm.nih.gov/books/NBK179288/

[29] Burgin J, Ahamed A, Cummins C, et al. The European Nucleotide Archive in 2022. *Nucleic Acids Res.* 2023;51(D1):D121–D125. https://doi.org/10.1093/nar/gkac1051

[30] Davis S, Meltzer PS. GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. *Bioinformatics.* 2007;23(14):1846–1847. https://doi.org/10.1093/bioinformatics/btm254

[31] Choudhary S. pysradb: A Python package to query next-generation sequencing metadata and data from NCBI Sequence Read Archive. *F1000Research.* 2019;8:532. https://doi.org/10.12688/f1000research.18676.1

[32] Gálvez-Merchán Á, Min KHJ, Pachter L, Booeshaghi AS. Metadata retrieval from sequence databases with ffq. *Bioinformatics.* 2023;39(1):btac667. https://doi.org/10.1093/bioinformatics/btac667

[33] Durinck S, Spellman PT, Birney E, Huber W. Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. *Nat Protoc.* 2009;4(8):1184–1191. https://doi.org/10.1038/nprot.2009.97

[34] Wang L, Wang S, Li W. RSeQC: quality control of RNA-seq experiments. *Bioinformatics.* 2012;28(16):2184–2185. https://doi.org/10.1093/bioinformatics/bts356

[35] Okonechnikov K, Conesa A, García-Alcalde F. Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data. *Bioinformatics.* 2016;32(2):292–294. https://doi.org/10.1093/bioinformatics/btv566

[36] Broad Institute. *Picard Toolkit.* GitHub repository; 2019. https://broadinstitute.github.io/picard/

[37] Daley T, Smith AD. Predicting the molecular complexity of sequencing libraries. *Nat Methods.* 2013;10(4):325–327. https://doi.org/10.1038/nmeth.2375

[38] Ramírez F, Ryan DP, Grüning B, et al. deepTools2: a next generation web server for deep-sequencing data analysis. *Nucleic Acids Res.* 2016;44(W1):W160–W165. https://doi.org/10.1093/nar/gkw257

[39] Pertea G, Pertea M. GFF Utilities: GffRead and GffCompare. *F1000Research.* 2020;9:304. https://doi.org/10.12688/f1000research.23297.2

[40] Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C. Salmon provides fast and bias-aware quantification of transcript expression. *Nat Methods.* 2017;14(4):417–419. https://doi.org/10.1038/nmeth.4197

[41] Bray NL, Pimentel H, Melsted P, Pachter L. Near-optimal probabilistic RNA-seq quantification. *Nat Biotechnol.* 2016;34(5):525–527. https://doi.org/10.1038/nbt.3519

[42] Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics.* 2010;26(1):139–140. https://doi.org/10.1093/bioinformatics/btp616

[43] Law CW, Chen Y, Shi W, Smyth GK. voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biol.* 2014;15(2):R29. https://doi.org/10.1186/gb-2014-15-2-r29

[44] Liberzon A, Birger C, Thorvaldsdóttir H, et al. The Molecular Signatures Database Hallmark gene set collection. *Cell Syst.* 2015;1(6):417–425.

[45] Korotkevich G, Sukhov V, Budin N, et al. Fast gene set enrichment analysis. *bioRxiv.* 2021;060012 (version 3, posted 1 February 2021). https://www.biorxiv.org/content/10.1101/060012v3

[46] Yu G, He QY. ReactomePA: an R/Bioconductor package for reactome pathway analysis and visualization. *Mol BioSyst.* 2016;12(2):477–479. https://doi.org/10.1039/c5mb00663e

[47] Young MD, Wakefield MJ, Smyth GK, Oshlack A. Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biol.* 2010;11(2):R14. https://doi.org/10.1186/gb-2010-11-2-r14

[48] Hänzelmann S, Castelo R, Guinney J. GSVA: gene set variation analysis for microarray and RNA-seq data. *BMC Bioinformatics.* 2013;14:7. https://doi.org/10.1186/1471-2105-14-7

[49] Kim D, Paggi JM, Park C, Bennett C, Salzberg SL. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nat Biotechnol.* 2019;37(8):907–915. https://doi.org/10.1038/s41587-019-0201-4

[50] Anders S, Pyl PT, Huber W. HTSeq—a Python framework to work with high-throughput sequencing data. *Bioinformatics.* 2015;31(2):166–169. https://doi.org/10.1093/bioinformatics/btu638

[51] Putri GH, Anders S, Pyl PT, Pimanda JE, Zanini F. Analysing high-throughput sequencing data in Python with HTSeq 2.0. *Bioinformatics.* 2022;38(10):2943–2945. https://doi.org/10.1093/bioinformatics/btac166

[52] Grabherr MG, Haas BJ, Yassour M, et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotechnol.* 2011;29(7):644–652. https://doi.org/10.1038/nbt.1883

[53] Li D, Liu CM, Luo R, Sadakane K, Lam TW. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics.* 2015;31(10):1674–1676. https://doi.org/10.1093/bioinformatics/btv033

[54] Van der Auwera GA, O'Connor BD. *Genomics in the Cloud: Using Docker, GATK, and WDL in Terra.* O'Reilly Media; 2020. (GATK Best Practices for RNA-seq short variant discovery.)

[55] Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal.* 2011;17(1):10–12.

[56] Krueger F. *Trim Galore: A Wrapper Around Cutadapt and FastQC.* Babraham Bioinformatics. https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/

[57] Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics.* 2014;30(15):2114–2120. https://doi.org/10.1093/bioinformatics/btu170

[58] Wingett SW, Andrews S. FastQ Screen: a tool for multi-genome mapping and quality control. *F1000Research.* 2018;7:1338. https://doi.org/10.12688/f1000research.15931.2

[59] Deng ZL, Münch PC, Mreches R, McHardy AC. Rapid and accurate identification of ribosomal RNA sequences via deep learning. *Nucleic Acids Res.* 2022;50(10):e60. https://doi.org/10.1093/nar/gkac112

[60] Ritchie ME, Phipson B, Wu D, et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res.* 2015;43(7):e47. https://doi.org/10.1093/nar/gkv007

[61] Gautier L, Cope L, Bolstad BM, Irizarry RA. affy—analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics.* 2004;20(3):307–315. https://doi.org/10.1093/bioinformatics/btg405

[62] Schuirmann DJ. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *J Pharmacokinet Biopharm.* 1987;15(6):657–680. https://doi.org/10.1007/bf01068419

[63] Kolberg L, Raudvere U, Kuzmin I, et al. g:Profiler—interoperable web service for functional enrichment analysis and gene identifier mapping (2023 update). *Nucleic Acids Res.* 2023;51(W1):W207–W212. https://doi.org/10.1093/nar/gkad347

[64] Szklarczyk D, Kirsch R, Koutrouli M, et al. The STRING database in 2023: protein-protein association networks and functional enrichment analyses for any sequenced genome of interest. *Nucleic Acids Res.* 2023;51(D1):D638–D646. https://doi.org/10.1093/nar/gkac1000

[65] Csardi G, Nepusz T. The igraph software package for complex network research. *InterJournal Complex Systems.* 2006;1695.

[66] Pedersen TL. *ggraph: An Implementation of Grammar of Graphics for Graphs and Networks.* R package. https://cran.r-project.org/package=ggraph

[67] Shannon P, Markiel A, Ozier O, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* 2003;13(11):2498–2504. https://doi.org/10.1101/gr.1239303

[68] Kent WJ, Sugnet CW, Furey TS, et al. The human genome browser at UCSC. *Genome Res.* 2002;12(6):996–1006. https://doi.org/10.1101/gr.229102

[69] Love MI, Anders S, Huber W. *Analyzing RNA-seq data with DESeq2.* Bioconductor package vignette. https://www.bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

[70] Bioconda contributors. *Bioconda documentation: usage and channel priority.* https://bioconda.github.io/

[71] Dobin A. *STAR stranded gene-count conventions and unstranded totals.* Upstream explanations, issues 842 and 1589. https://github.com/alexdobin/STAR/issues/1589

[72] STAR developers. *STAR parameter reference: index and BAM-sort memory settings.* https://github.com/alexdobin/STAR/blob/master/source/parametersDefault

[73] Salmon developers. *Salmon documentation: selective alignment and decoy-aware transcriptomes.* https://salmon.readthedocs.io/en/stable/salmon.html

[74] KEGG. *KEGG GENES database and identifier conventions.* https://www.genome.jp/kegg/genes.html

[75] STRING consortium. *What is a STRING network?* https://en.string-db.org/help/scores/

[76] Antoszewski K, Chmielewska K, Jagiello K, Puzyn T. Designing RNA sequencing experiments: A practical guide to reproducible gene expression analysis. *Comput Struct Biotechnol J.* 2026;31:101–119. Published online 18 December 2025. https://doi.org/10.1016/j.csbj.2025.12.015

[77] Wagner MR, Kleiner M. How thoughtful experimental design can empower biologists in the omics era. *Nat Commun.* 2025;16:7263. https://doi.org/10.1038/s41467-025-62616-x

[78] Chen Y, Chen L, Lun ATL, Baldoni PL, Smyth GK. edgeR v4: powerful differential analysis of sequencing data with expanded functionality and improved support for small counts and larger datasets. *Nucleic Acids Res.* 2025;53(2):gkaf018. https://doi.org/10.1093/nar/gkaf018

[79] Rau A, Marot G, Jaffrézic F. Differential meta-analysis of RNA-seq data from multiple studies. *BMC Bioinformatics.* 2014;15:91. https://doi.org/10.1186/1471-2105-15-91