BIOF3 组学数据分析

bulk RNA-seq 实践手册

BioF3 bulk RNA-seq 实践教程导出版

导出日期：2026年6月27日

扫码关注微信公众号【生信F3】

获取文章完整内容，分享生物信息学最新知识。

BIOF3 组学数据分析

bulk RNA-seq 实践手册目录

BioF3 bulk RNA-seq 实践教程导出版

01 bulk RNA-seq 实践教程

bulk RNA-seq 把一份组织或一群细胞打成混合样本测序，得到的是每个基因在样本里的平均表达量。多数实验室的转录组项目仍然围绕它展开：几十个样本、两三个处理组、一篇文章的主线分析。

本专栏打算把 bulk RNA-seq 从"测完序之后怎么办"这一步开始讲，按真实项目常见的顺序走一遍：质控、比对或准比对、定量、差异分析、富集解释。尽量给出能在自己电脑上跑完的数据和脚本，而不是只列工具名。

bulk vs 单细胞 vs 空间：什么时候选哪个

在动手之前先回答一个问题：你的科学问题真的需要 bulk RNA-seq 吗？

问题	推荐技术	理由
两组样本之间整体差异	**bulk RNA-seq**	便宜、统计成熟、3-5 个样本就够
不同细胞类型的响应不同	单细胞 RNA-seq	bulk 看到的是各类细胞的加权平均，细节被抹掉
表达定位（哪一块组织在做什么）	空间转录组	bulk 和单细胞都没有空间信息
稀有亚群（< 5%）的变化	单细胞	bulk 里 5% 的信号被 95% 稀释
大样本 + 临床/流行病学问题	**bulk RNA-seq**	TCGA 上千样本量级，单细胞做不起
转录起始位点 / 可变剪接	bulk + 长读长	标准 bulk 也能做，但 Iso-seq / Nanopore 直接

bulk 还是大多数项目的默认起点：便宜、流程成熟、统计学经过 20 年验证、可以做大样本。除非你的问题明显需要细胞分辨率或空间信息，否则先做 bulk。

一个项目大致是什么样子

假设你做了一个时间点比较的实验：对照组三个样本、处理组三个样本、每个样本一次双端测序。下机之后你会拿到 12 个 FASTQ 文件，加起来可能有几十 GB。

从这里到"一张火山图 + 一张 GO 富集结果 + 一句能写进论文的结论"之间，要经过的步骤大致是：

步骤	典型产物	常用工具
原始数据质控	每个样本的 QC 报告	FastQC、MultiQC
比对到参考基因组	每个样本的 BAM	STAR、HISAT2
基因或转录本定量	counts 矩阵 或 TPM 矩阵	featureCounts、Salmon、kallisto
样本层面检查	PCA 图、样本相关性	DESeq2、edgeR、limma
差异表达分析	差异基因表	DESeq2、edgeR、limma-voom
功能富集	GO、KEGG、GSEA 结果	clusterProfiler、fgsea
结果交付	表格、图、方法段	R Markdown、Quarto

"比对 + featureCounts" 和 "Salmon/kallisto 准比对" 是两条常见路线。前者产出 BAM、对可视化和变异分析都友好；后者跳过显式比对，直接估计转录本丰度，速度快、磁盘友好。新项目里如果只关心基因层面的差异表达，用 Salmon 是更经济

的选择。

常见工具栈

在 BioF3 的例子里会优先使用下面这套组合，原因是文档、社区和版本都比较稳定：

阶段	工具	运行环境
质控	FastQC、MultiQC	bash
比对	STAR、HISAT2	bash
准比对定量	Salmon、tximport	bash + R
差异分析	DESeq2、edgeR、limma-voom	R
富集分析	clusterProfiler、fgsea	R
可视化	ggplot2、ComplexHeatmap、EnhancedVolcano	R

其他工具（Kallisto、RSEM、StringTie、pheatmap 等）都是合理替代，但这套组合能覆盖大部分教学和小型真实项目。

推荐公开数据集

学这个专栏不一定要用自己的测序数据。下面这几份公开数据都能用来练手，且每份都有配套文献可对照。

数据集	规模	适合	入口
Pasilla（果蝇 RNA-seq）	7 个样本	最小差异分析例子，Bioconductor 官方包 `pasilla` 里直接带	pasilla
airway（人肺上皮，4 对照 + 4 处理）	8 个样本	DESeq2 官方教程数据，做 glucocorticoid 处理	airway
GTEx 子集	可选子集	多组织 / 多样本练习归一化和批次	GTEx
Recount3 项目	任选一个 SRP	预处理好的 counts 矩阵，跳过比对	Recount3

前两个数据集在 R 里装一下包就能直接用，不需要下载 FASTQ，非常适合当作入门。GTEx 和 Recount3 的优点是真实样本量足够做归一化和批次演示。

最小可跑的例子

下面这段代码从安装 DESeq2 到跑出一份差异基因结果一共不到 20 行，数据就是上面 airway 公共数据集。把它粘到 RStudio 里，装好包就能跑：

```r
# 一次性安装依赖（如果还没装）
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install(c("DESeq2", "airway"))

library(DESeq2)
library(airway)

# 加载示例数据：8 个样本 x ~64000 个基因的 counts 矩阵
data(airway)
se <- airway
se$dex <- relevel(se$dex, ref = "untrt")

# 构建 DESeq 对象并拟合模型
dds <- DESeqDataSet(se, design = ~ cell + dex)
dds <- DESeq(dds)

# 提取差异基因（trt vs untrt）
res <- results(dds, contrast = c("dex", "trt", "untrt"))
summary(res)

# 看一下最显著的几个基因
head(res[order(res$padj), ])
```

运行之后 `summary(res)` 会告诉你上下调基因各多少个，`head(res[order(res$padj), ])` 会列出按校正 p 值排的前几个基因。把这几行读懂，基本就知道 bulk RNA-seq 差异分析是在做什么了。

真实项目比这段要多两件事：样本多、要考虑批次和协变量；定量步骤之前还有比对和质控。这也是后面几个模块要展开讲的内容。

专栏模块规划

模块	主题	状态
01	项目结构、数据入口与 Salmon 定量	已上线
02	DESeq2 差异表达分析	已上线
03	GO / KEGG / GSEA 富集分析	已上线
04	火山图、热图与富集可视化	已上线
05	多时间点与 LRT 检验	已上线
06	批次效应：ComBat / SVA / batch-in-design	已上线
07	DESeq2 / edgeR / limma-voom 三工具对比	已上线
08	可复现分析报告与结果交付	已上线

所有 8 个模块都已上线，前七个模块带一份可以直接跑的 R 脚本。

推荐前置知识

- 编程基础：R、Python、Bash 学习路径

- R 数据整理与 ggplot2 可视化
- 公共数据库与数据检索

单细胞项目里的质控、归一化和差异分析思路和 bulk 有重叠，已经看过单细胞实践教程的读者会觉得很多概念上手更快。

常见误区

新手做 bulk RNA-seq 经常掉的几个坑：

误区 1：只做差异分析就交付

差异基因列表 ≠ 完整结果。**至少还要补：PCA / 样本距离的 QC 图、富集分析、关键基因的表达分布图**。只给一份 padj 排序的 csv 给合作者，对方看不出哪些是真信号、哪些是边缘 noise。

误区 2：不看 PCA 直接跑差异

PCA 是"测序质量 + 设计是否合理"的总体检查。**PC1 不分组 = 处理效应弱（或者 design 错了）；某个样本远离同组其他样本 = 该样本可能有问题**。这两件事不解决，跑出来的差异基因没法解释。

误区 3：把 |LFC| > 1 当成生物学显著的硬标准

`|log2FC| > 1` 即 fold change > 2 倍。**这个阈值在不同项目里意义完全不同**：转录因子表达 1.5 倍可能就是大事，组织特异性基因 10 倍变化才有意义。先看自己数据的整体分布，再选阈值。

误区 4：富集结果只看 top 5 通路

GO / KEGG 跑出来几十条通路，只看 top 5 容易被冗余 term（"细胞增殖" / "正向调控细胞增殖" / "细胞周期"基本是同一件事）误导。**用 cnetplot / emapplot 把通路网络化，看真正的几个独立模块**。

误区 5：发表前不锁版本

DESeq2 / clusterProfiler / Salmon 半年一更，**包升级有时会改算法默认值**。发表后审稿人或读者重跑结果不一致就麻烦。**用 `renv` 或 `conda env.yaml` 锁版本**，08 章会展开。

参考资源

- DESeq2 官方教程
- edgeR 用户指南
- limma 用户手册
- RNA-seq workflow (Bioconductor)
- Recount3
- GTEx Portal

02 从数据到项目结构

这一章的目标是让读者在真正动手做 bulk RNA-seq 之前，先把"拿到什么输入、打算得到什么输出"想清楚。很多项目一开始跑得很顺，到了复盘阶段才发现样本表有问题、目录结构乱、中间文件丢了。bulk RNA-seq 的数据量比单细胞小得多，但分析链条更长，更值得一开始就把项目结构定下来。

一个典型项目的输入和输出

以最常见的两组比较为例：对照 vs 处理，每组 3 个生物学重复，每个样本一次双端测序。你手上会有的东西和最后要交付的东西大致是：

阶段	输入	输出
测序中心交付	12 个 FASTQ 文件（6 样本 × 2 端）+ 基本 QC 报告	—
分析起点	上面这些 FASTQ	本地目录结构 + 样本表
比对 / 准比对	FASTQ + 参考基因组 + GTF	每个样本的 BAM 或 Salmon quant 目录
定量	BAM / Salmon	一份 gene × sample 的 counts 矩阵
差异分析	counts + 样本表	差异基因表（gene, log2FC, padj）
功能解释	差异基因表	GO / KEGG / GSEA 结果
交付	所有上面的结果	图 + 表 + Methods 段

"样本表"是整条链路的中心。它至少要有：样本 ID、条件、批次、测序时间、FASTQ 路径。后面每一步都要回到这张表。

项目目录

长期维护的目录结构没有标准答案，但至少应该满足"6 个月后回来还知道每个文件是什么"。一个可用的结构：

```
rnaseq_project/
├── README.md                 # 项目背景、样本来源、关键决策
├── metadata/
│   └── samples.tsv           # 样本表，唯一可信来源
├── data/
│   ├── raw/                  # 原始 FASTQ（不动）
│   ├── qc/                   # FastQC + MultiQC 报告
│   ├── align/                # BAM 或 Salmon quant
│   └── counts/               # 合并后的 counts 矩阵
├── analysis/
│   ├── 01-qc.R
│   ├── 02-de.R               # DESeq2 / edgeR
│   ├── 03-enrichment.R
│   └── 04-figures.R
├── results/
│   ├── tables/
│   └── figures/
└── envs/
    └── env.yaml              # conda 环境，锁定版本
```

几个经验：

- `data/raw/` 只读，永远不直接修改，新文件写到其他目录
- 分析脚本按步骤命名（`01-`，`02-` ...），从上到下能复现整个流程
- `results/` 只放最终交付物，中间产物放 `data/`
- `envs/env.yaml` 锁版本，过两个月再跑也能装回原环境

样本表怎么写

样本表是一个纯文本 TSV，越简单越好。一个最小可用版本：

```
sample_id         condition         batch    r1       r2
CTRL_1    untreated         1        data/raw/CTRL_1_R1.fastq.gz         data/raw/CTRL_1_R2.fastq.gz
CTRL_2    untreated         1        data/raw/CTRL_2_R1.fastq.gz         data/raw/CTRL_2_R2.fastq.gz
CTRL_3    untreated         2        data/raw/CTRL_3_R1.fastq.gz         data/raw/CTRL_3_R2.fastq.gz
TRT_1     treated  1        data/raw/TRT_1_R1.fastq.gz data/raw/TRT_1_R2.fastq.gz
TRT_2     treated  1        data/raw/TRT_2_R1.fastq.gz data/raw/TRT_2_R2.fastq.gz
TRT_3     treated  2        data/raw/TRT_3_R1.fastq.gz data/raw/TRT_3_R2.fastq.gz
```

`batch` 这一列不一定用得到，但有分析经验之后你会知道：**一开始就记录下来是最便宜的，事后补很贵**。测序批次、试剂批次、操作人、文库类型，只要是可能引入系统偏差的变量都值得留一列。DESeq2 的 design 会直接用到这些列：

```
design = ~ batch + condition
```

参考基因组和注释

人和小鼠的参考基因组有几个主流来源：

来源	特点	推荐用途
GENCODE	人 / 小鼠注释的金标准	绝大多数项目，和 Ensembl ID 对得上
Ensembl	覆盖物种最多	非模式物种
UCSC / NCBI	字段丰富	做浏览器可视化、变异分析

基因组和注释**必须来自同一个来源、同一个版本**，不然基因 ID 和坐标就对不上。推荐直接下载 GENCODE 的 primary assembly + 主注释 GTF，搭配 Salmon 或 STAR 都能跑：

```
# 人 v44 作为例子
wget -c https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_44/GRCh38.primary_assembly
wget -c https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_44/gencode.v44.primary_ass
```

这两个文件解压后占 6~7 GB，落到 `data/reference/` 里，长期项目里这份东西是共享的，不跟单个项目耦合。

比对 vs 准比对

两条主流路线：

路线	典型工具	速度	磁盘	适用场景
全比对到基因组	STAR / HISAT2 → featureCounts	慢	大（BAM）	需要 BAM 做可视化 / SNV / 新转录本
准比对到转录本	Salmon / kallisto → tximport	快	小	只做差异分析和富集

新项目如果只做差异表达和富集，用 Salmon 几乎是最佳选择 —— 笔记本上几分钟就能定量一个样本。下面这段是 Salmon 的典型调用：

```
# 建立索引（只做一次）
salmon index \
  -t gencode.v44.transcripts.fa.gz \
  -i salmon_index \
  --gencode

# 定量每个样本
for s in CTRL_1 CTRL_2 CTRL_3 TRT_1 TRT_2 TRT_3; do
  salmon quant \
    -i salmon_index \
    -l A \
    -1 data/raw/${s}_R1.fastq.gz \
    -2 data/raw/${s}_R2.fastq.gz \
    -p 8 --validateMappings \
    -o data/align/${s}
done
```

定量完成后每个样本会得到一个目录，里面的 `quant.sf` 就是转录本级别的定量结果。后续在 R 里用 `tximport` 聚合到基因级就进入差异分析：

```
library(tximport)
library(readr)

samples <- read.table("metadata/samples.tsv", header = TRUE, sep = "\t")
files <- file.path("data/align", samples$sample_id, "quant.sf")
names(files) <- samples$sample_id

# tx2gene 映射（从 GTF 生成一次就好）
tx2gene <- readr::read_tsv("metadata/tx2gene.tsv")

txi <- tximport(files, type = "salmon", tx2gene = tx2gene)
# txi$counts 就是 gene x sample 的 counts 矩阵
```

快速跳过：直接用 airway / pasilla

上面这套是面向真实项目的；如果只是想学习差异分析，**完全可以跳过 FASTQ → BAM → counts 这一段**，直接用 Bioconductor 已经打包好的 counts 数据：

- `airway` — 人肺上皮 8 样本（4 对照 + 4 糖皮质激素处理），DESeq2 官方教程数据
- `pasilla` — 果蝇 7 样本，DESeq2 最小例子

后面 02、03、04 的真实示例都用 `airway`，装上包就能跑，不用碰比对流程。

常见坑

坑 1：样本表用 Excel 编辑导致格式错乱

Excel 会自动把 `MARCH1`（基因名）改成日期、把数字前补 0 删掉、改空格成 unicode。**永远用纯文本编辑器或 R / Python 维护样本表**，要看的时候再 `read.delim` 进来。

坑 2：参考基因组版本和注释不匹配

下载 GRCh38 fasta 配 GENCODE v44 GTF 没问题；但配 v25 GTF 会出现"基因名对得上、坐标对不上"的诡异错误。**fa 和 gtf 必须来自同一来源同一版本**，不要混用。

坑 3：把 raw counts 当成已归一化的 TPM

featureCounts 出来的是 raw counts（整数），可以直接喂 DESeq2 / edgeR；但 Salmon / kallisto 的默认输出是 TPM 或估计 counts，**给 DESeq2 时要用 `tximport(..., type = "salmon")` 转成 counts，不是直接读 quant.sf**。

坑 4：Salmon 索引用错链路

`--gencode` 标志只对 GENCODE 来源的转录组用，**Ensembl 来源的转录组不要加 `--gencode`**。否则 transcript ID 在去版本号时会出错。

坑 5：把样本目录命名成数字开头

Linux 下排序看上去 `1.fastq` `2.fastq` 没问题，但 R 读进来当数据框列名时会自动加 `X` 前缀（变成 `X1`、`X2`）。**样本 ID 一律用字母开头**（`S1`、`CTRL_1`），下游脚本省心。

下一步

接着深入：

- 02 DESeq2 差异表达分析 — counts 准备好之后的第一步，airway 公开数据可以直接跑

横向延伸：

- 03 GO / KEGG / GSEA 富集 — 如果你已经有差异基因表，可以直接跳到富集分析
- 06 批次效应 — 多批次样本要先想清楚 batch 怎么处理，再设计 design 公式

参考资源

- GENCODE
- Salmon 官方文档
- tximport 教程
- RNA-seq workflow (Bioconductor)

速答

Q: DESeq2 为什么用负二项分布而不是泊松? A: 泊松假设均值=方差，RNA-seq 数据存在过度离散（方差 > 均值），负二项分布多一个离散度参数能拟合这种过度离散。

Q: DESeq2 的 padj 和 pvalue 有什么区别? A: padj 是 Benjamini-Hochberg 校正后的 p 值，控制假阳性率。多基因检验时必须用 padj（通常 < 0.05），不能用原始 pvalue。

Q: log2FC 阈值设多少? A: 常用 |log2FC| > 1（即 2 倍变化）。宽松可降到 0.58（1.5 倍），严格可到 1.5（2.8 倍）。结合 padj 一起筛。

Q: LFC shrinkage 是什么，要不要做? A: 对 log2FC 做收缩估计（apeglm / ashr），减少低表达基因的极端 fold change。推荐做，结果更可信、火山图更好看。

Q: DESeq2 和 edgeR 怎么选? A: DESeq2 文档完善、默认参数稳健，适合大多数场景；edgeR 更灵活、速度更快，适合有经验的分析者。结果通常高度一致。

03 DESeq2 差异表达分析

DESeq2 是 bulk RNA-seq 差异分析里最常用的工具。它假设 counts 服从负二项分布，用每个基因的表达均值和离散度估计显著性，加上 Benjamini-Hochberg 校正给出最终的 padj。流程非常规整，稳到几乎是默认选择。

这一章用 DESeq2 官方教程数据 airway 走一遍完整流程。数据是 8 个人类气道上皮细胞样本 —— 4 个对照 + 4 个糖皮质激素（地塞米松，dex）处理，来自 Himes et al. 2014（PLoS ONE 9(6): e99625）。

为什么是负二项分布而不是泊松或正态

DESeq2 模型选择不是随便拍脑袋决定的，理解这一点能让你看懂很多默认行为：

- **不是正态分布**：counts 是非负整数，且方差不等于均值（高表达基因的方差比泊松预期的大很多）
- **不是泊松分布**：泊松假设方差 = 均值，但真实 RNA-seq 数据 over-dispersed —— 同一条件下生物学重复之间方差远大于均值
- **是负二项分布**：在泊松基础上加一个 dispersion 参数，描述"额外的方差"。这个 dispersion 在低表达基因数据少时估不准，DESeq2 用"所有基因共同拟合一条 mean-dispersion 曲线"来稳定估计

这意味着什么：

1. counts 矩阵必须是**未归一化的整数**（raw counts），不能是 TPM / RPKM / 已 log 变换的值
2. DESeq2 需要至少 2 个生物学重复才能估 dispersion；3 个起步、4-6 个稳
3. 强行用单样本（1 vs 1）跑 DESeq2 会强制把 dispersion 拉到全局均值，结果不可信

核心流程是三步

DESeq2 的调用界面非常简洁。真正的 API 就三步：

```
dds <- DESeqDataSet(se, design = ~ batch + condition)
dds <- DESeq(dds)
res <- results(dds, contrast = c("condition", "trt", "untrt"))
```

1. 构建 `DESeqDataSet` 对象，声明设计公式（`design` 最后一列是要检验的变量）
2. `DESeq()` 一个调用里做完归一化、离散度估计、Wald 检验
3. `results()` 提取指定对比的差异结果

真实项目的工作重点不在这三行代码上，而在**围绕它的 QC 和解释**：样本是否正常、模型是否合适、结果是否可信。下面按这个顺序走。

加载 airway 并构建 DESeq 对象

```
library(DESeq2)
library(airway)

data(airway)
se <- airway
# 把 untrt 设为参考水平，这样 log2FC 是 "trt 相对 untrt"
se$dex <- relevel(se$dex, ref = "untrt")

# design: 控制细胞系差异，检验 dex 处理效应
dds <- DESeqDataSet(se, design = ~ cell + dex)

# 预过滤：counts 总和 < 10 的基因直接丢掉
# 不影响统计正确性，但能加速并减少内存
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep, ]
```

airway 的 `colData` 里 `cell` 是细胞系（4 个不同来源），`dex` 是处理变量。`design = ~ cell + dex` 的意思是"在控制细胞系差异之后看 dex 的影响"。设计公式里变量的顺序无所谓，但**要检验的变量一般放最后**，`results()` 默认会拿最后一个变量的第一个对比。

跑 DESeq 并看总体结果

```
dds <- DESeq(dds)

res <- results(dds, contrast = c("dex", "trt", "untrt"))
summary(res)
```

`summary(res)` 会告诉你上下调基因各有多少个（默认阈值 padj < 0.1）。对 airway 这份数据，通常会看到几千个显著差异基因 —— 糖皮质激素是强效应通路，差异很明显。

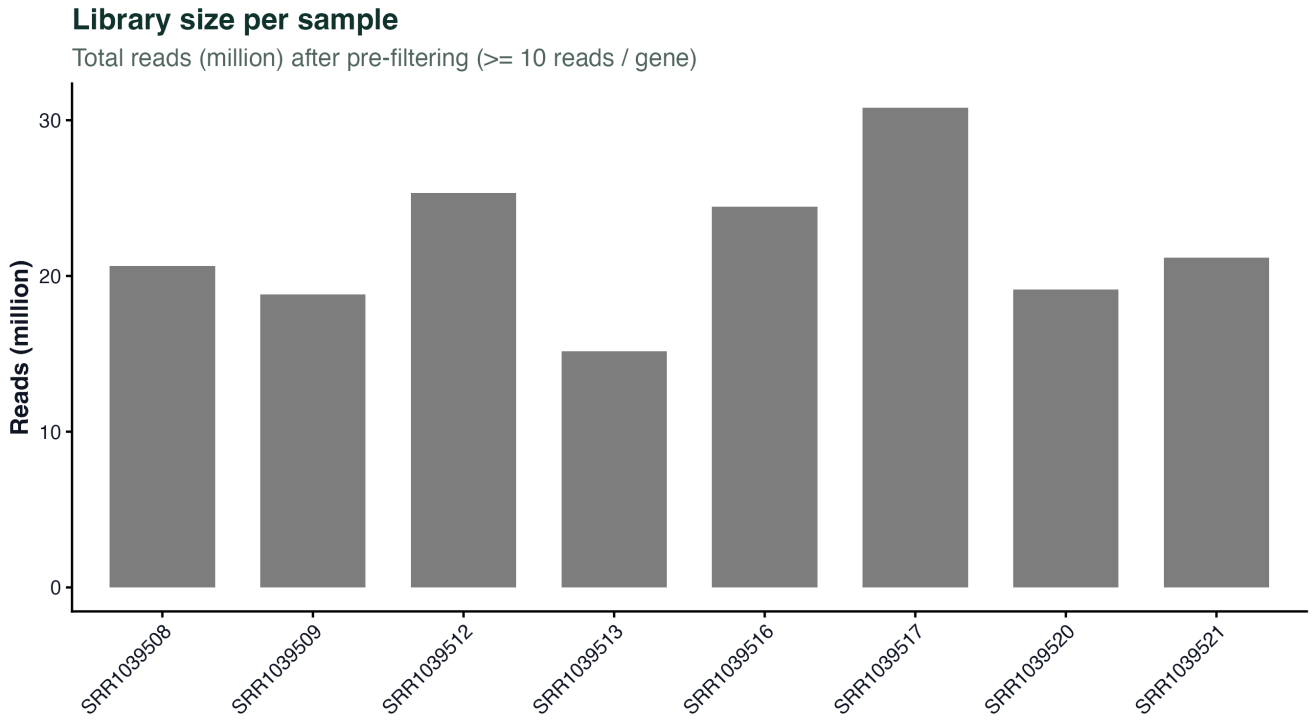
真实示例：airway 上的完整 DESeq2 分析

配套脚本 `bulk02_deseq2_sci.R` 把上面的流程完整跑了一遍，输出 6 张真实数据图和一份 DE 基因表。装好 `DESeq2 + airway + apeglm` 就能直接跑，不需要下载 FASTQ：

```
Rscript scripts/bulk02_deseq2_sci.R
```

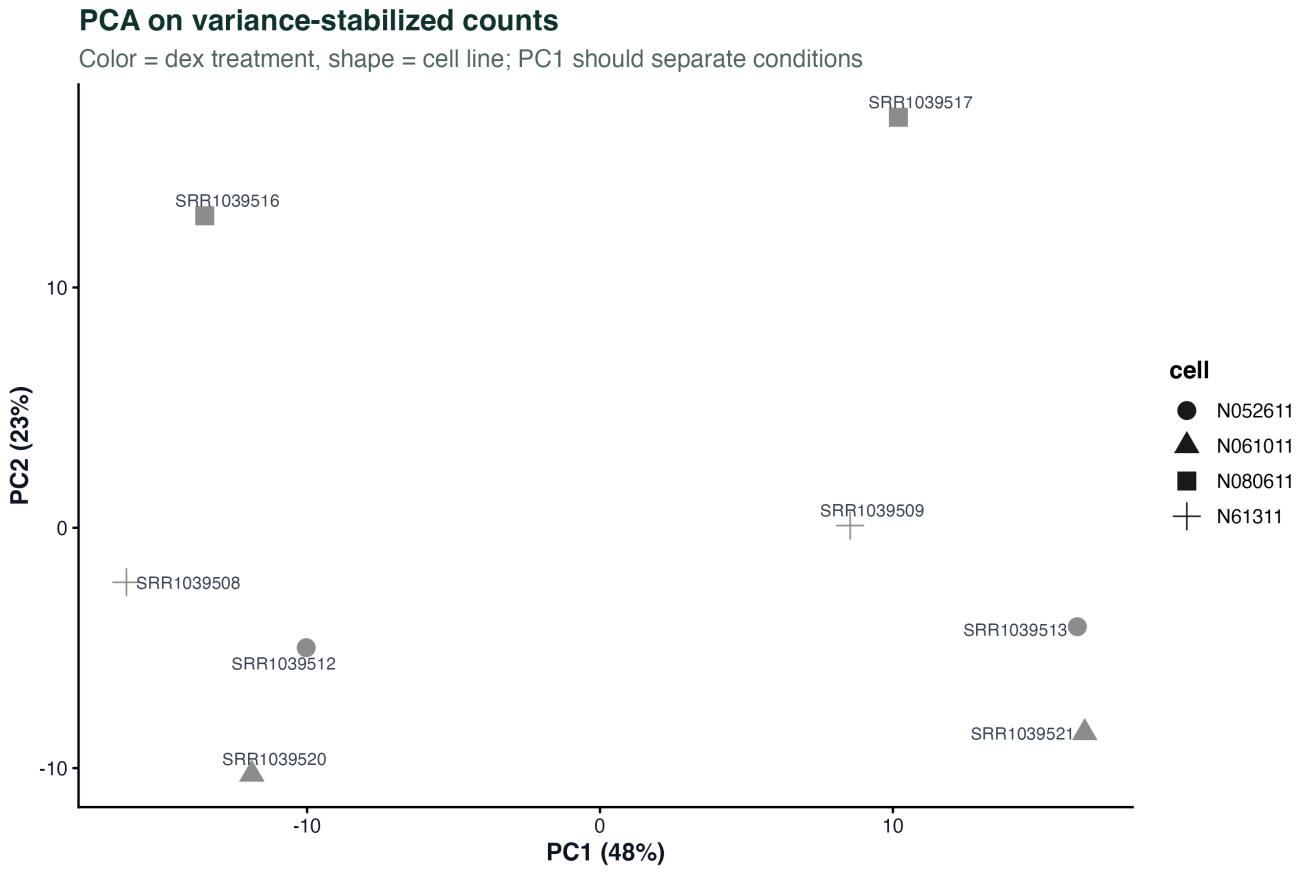
脚本里做的事：预过滤 → `DESeq()` → `results()` → `lfcShrink(apeglm)` → 6 张图 → 输出 DE 表。下面逐张看。

图 1：每个样本的库大小



横轴是样本 ID，纵轴是预过滤之后每个样本的总 reads（百万）。这张图是最基础的 QC：库大小应该在一个量级，如果有某个样本只有其他样本的 1/5，就要回去看测序报告。airway 里 8 个样本大小相当，没问题。

图 2：VST 归一化后的 PCA

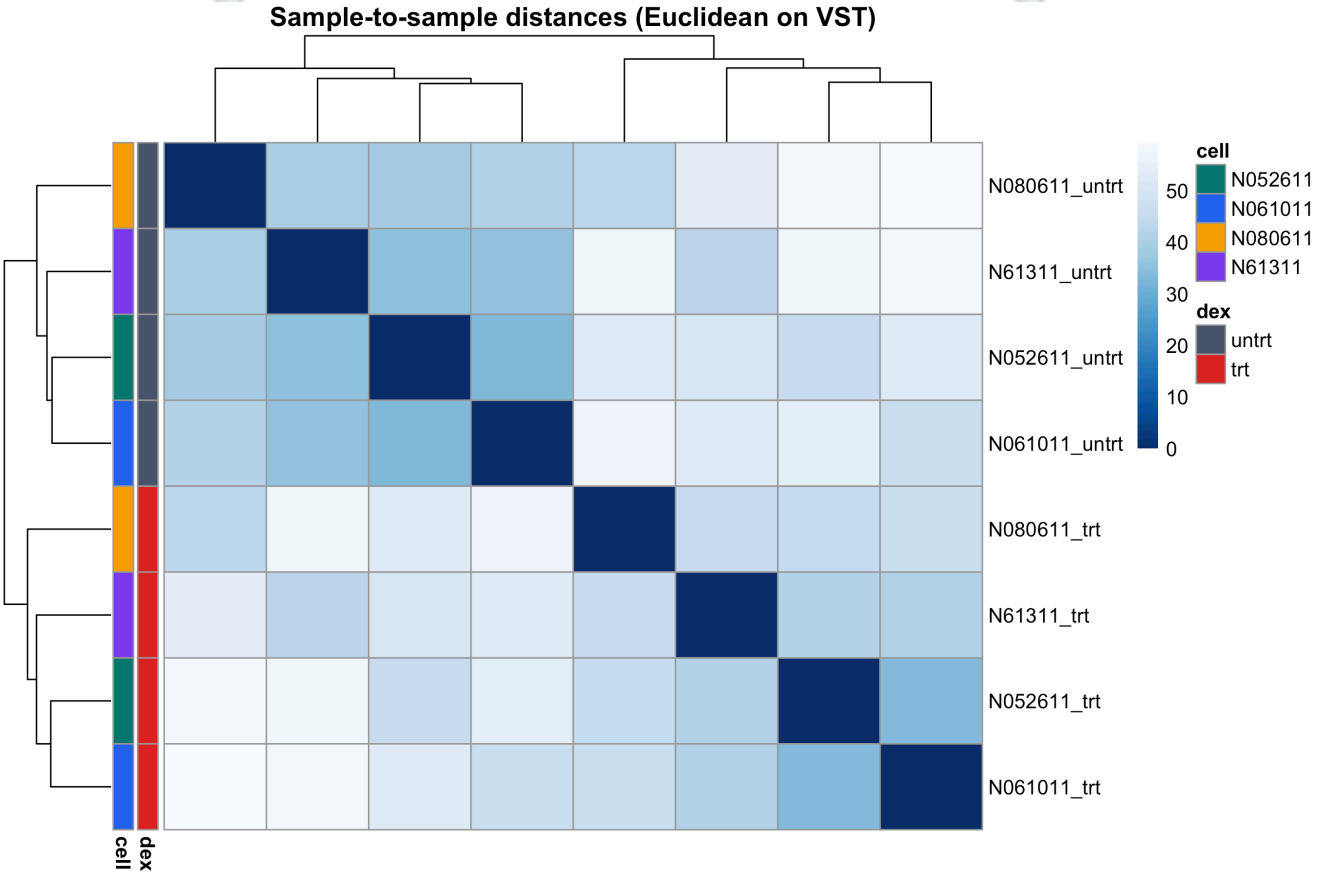


`vst(dds, blind = FALSE)` 做方差稳定变换 —— 让高表达和低表达基因的方差都差不多，这样 PCA 才不被少数高表达基因主导。颜色是 dex 处理，形状是细胞系。

这张图回答两个问题：

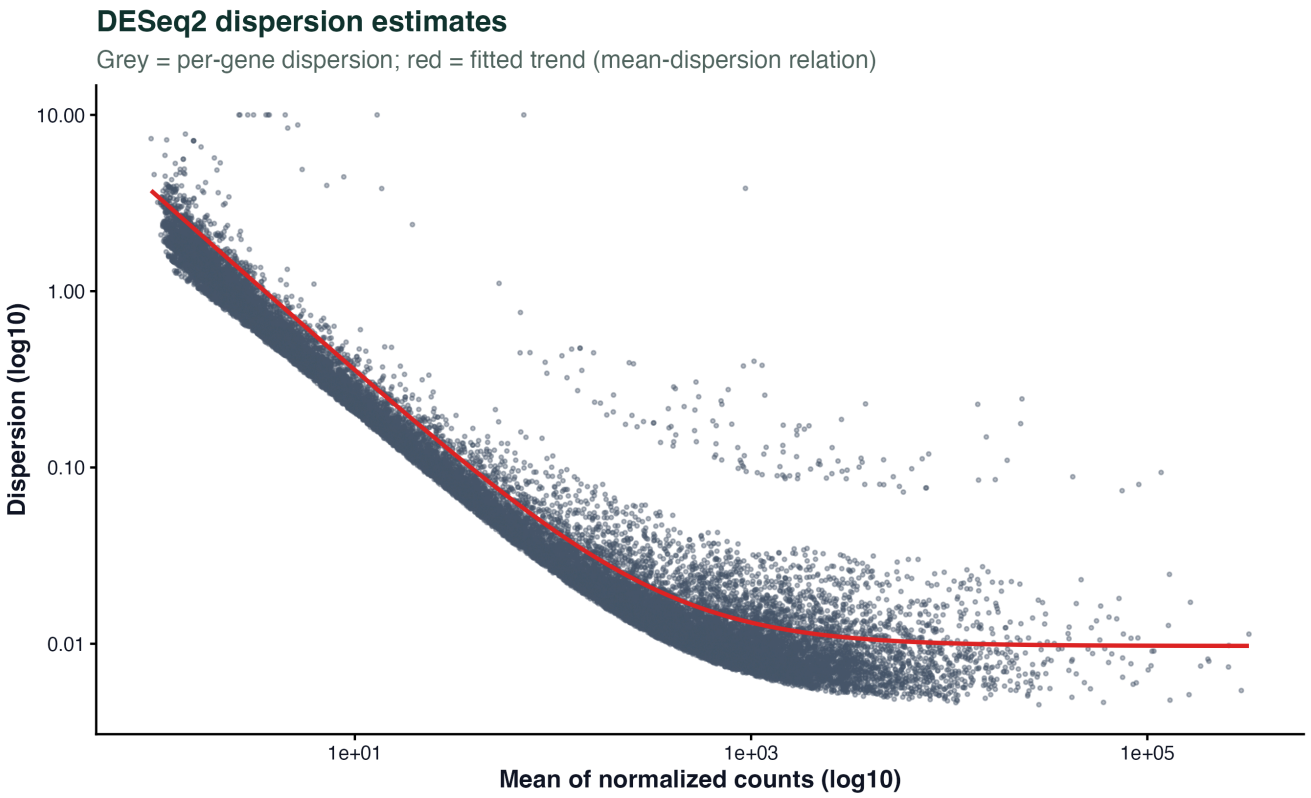
- **PC1 是什么**：理想情况下 PC1 直接把处理组和对照组分开。airway 里 PC1 正是 dex 轴。
- **细胞系差异大不大**：同样颜色但不同形状的点会偏开，说明细胞系间有个体差异。这就是 design 里加 `cell` 的原因。

图 3：样本间欧式距离



样本两两之间的欧式距离（在 VST 空间）画成热图，再聚一次类。同一条件 + 同一细胞系的样本应该靠在一起。如果某个样本跟同组的样本距离反而大过别组，要小心标签弄错或者该样本有问题。

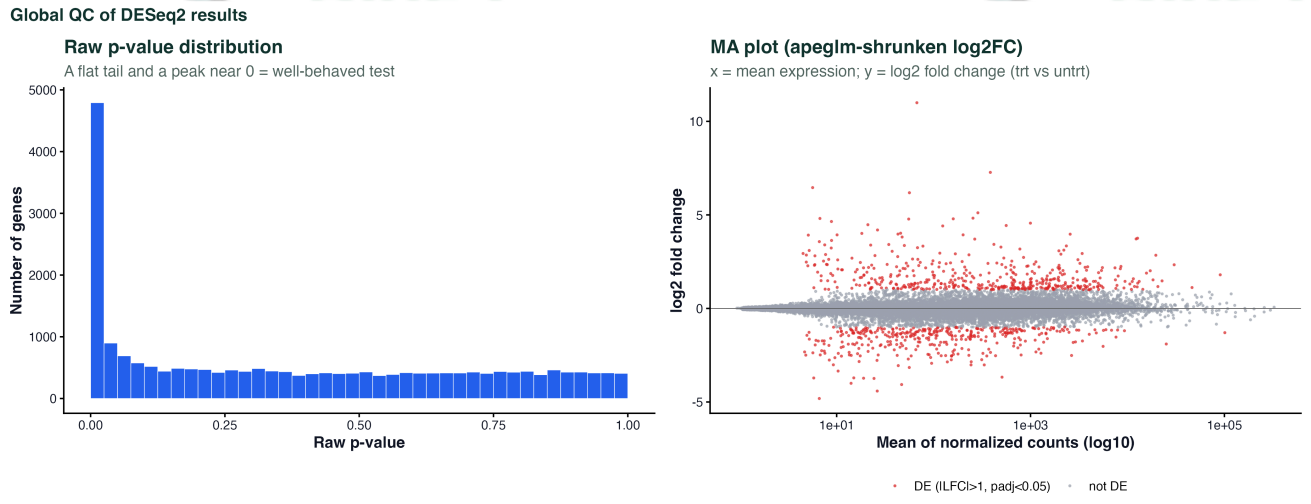
图 4：离散度估计



DESeq2 的核心是"用所有基因一起拟合 mean-dispersion 关系"，低表达基因数据少、方差估计不稳，就借用趋势回到那条红色的拟合曲线上。灰色是每个基因的原始离散度，红线是拟合趋势。

看这张图主要是确认：**趋势是不是随均值单调下降**。如果红线走势很乱，可能说明样本里有严重的技术偏差。正常的曲线应该像这张一样，左下右上缓慢下降。

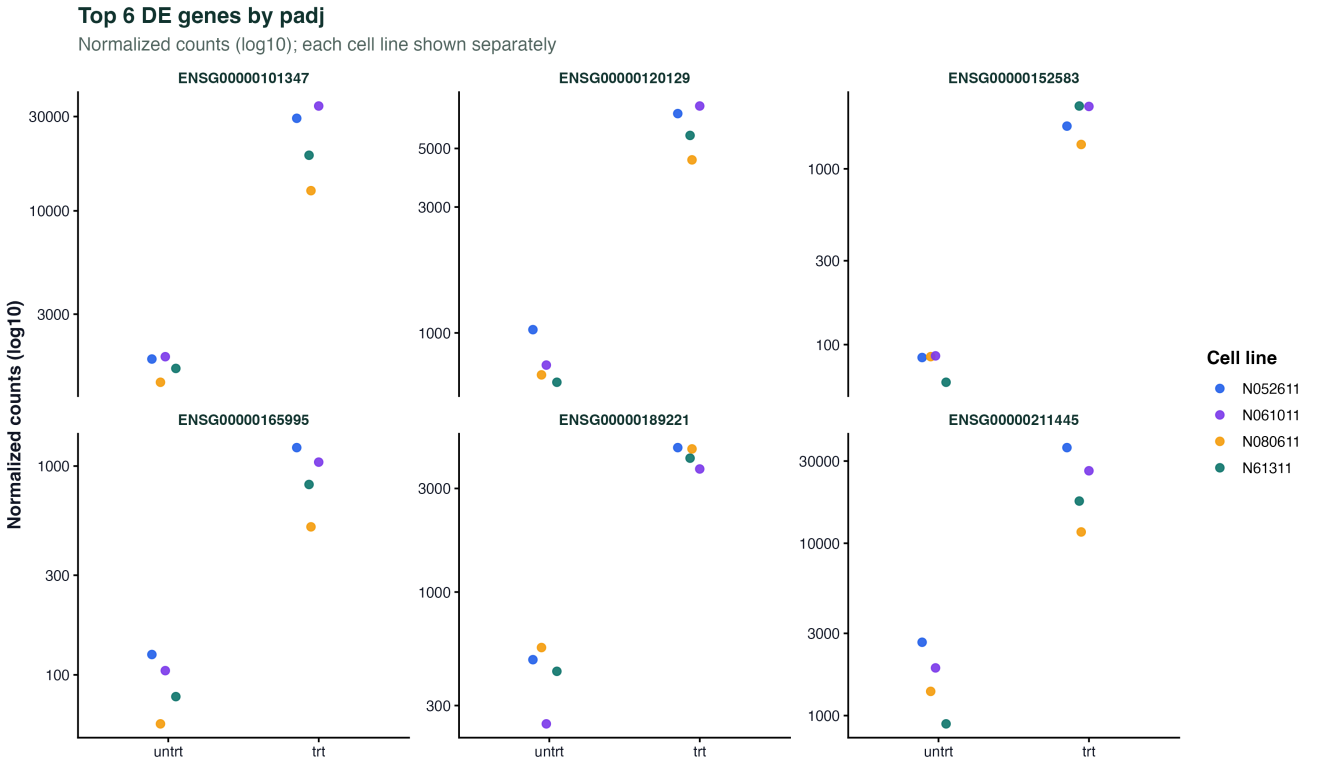
图 5：p 值分布与 MA 图



左边是原始 p 值直方图。一张健康的直方图长这样：0~0.05 一段突出（真正显著的基因），0.05~1 平坦分布（无效假设下的均匀分布）。如果直方图两端高中间低，说明离散度估计有问题；如果整体偏左（p 值都很小），可能是 design 漏了重要的协变量。

右边是 MA plot，横轴 `log10(均值表达量)`，纵轴 `log2 fold change`。红点是显著 DE 基因（|LFC| > 1 且 padj < 0.05）。这里用的是 `apeglm` shrunken LFC，低表达基因（左侧）的 fold change 被往 0 拉，避免"表达量低 + 噪声大 → fold change 虚高"的假象。

图 6：Top DE 基因的归一化 counts



按 padj 排最前面的 6 个基因的 normalized counts，每条细胞系画成一个颜色。y 轴是 log10。每个基因都能看到 4 个 untrt 和 4 个 trt 点，处理组和对照组在 y 轴上分得很开，且不同细胞系的点虽然绝对值有差异，但处理效应方向一致 —— 这是 DESeq2 在控制了 `cell` 之后仍然能拿到强显著性的原因。

提取结果表

脚本末尾把 shrunken log2FC + 原始 padj 写成一张 TSV：

```
de_tbl <- as.data.frame(res_shrink) |>
  tibble::rownames_to_column("gene_id") |>
  mutate(padj_raw = results(dds, contrast = c("dex", "trt", "untrt"))$padj) |>
  arrange(padj_raw)

write.table(de_tbl, "de_genes_airway.tsv", sep = "\t", quote = FALSE, row.names = FALSE)
```

这张表的列：`gene_id` 、 `baseMean` 、 `log2FoldChange` (shrunken)、 `lfcSE` 、 `pvalue` 、 `padj_raw` 。进 `clusterProfiler` 或 `fgsea` 做富集分析就用这张表。

套到自己数据上

脚本的前 15 行把 `airway` 换成自己的数据就行。常见两种输入：

```r
# 输入 1：tximport 的结果
library(tximport)
txi <- tximport(files, type = "salmon", tx2gene = tx2gene)
dds <- DESeqDataSetFromTximport(txi, colData = samples, design = ~ batch + condition)

# 输入 2：featureCounts 的 counts 矩阵
count_mat <- read.delim("counts.tsv", row.names = 1)
samples <- read.delim("metadata/samples.tsv", row.names = 1)
dds <- DESeqDataSetFromMatrix(count_mat, colData = samples, design = ~ batch + condition)
```

其余的 `DESeq()` / `results()` / `lfcShrink()` 调用不用改。真实项目里最容易踩的坑：

- 样本表里 `rownames` 要和 counts 矩阵的列名严格一致（包括顺序）
- 如果样本有批次（不同测序 run、不同试剂盒批次），一定要加进 design
- 小于 3 对 3 的实验 DESeq2 能跑，但统计力很弱，不建议做复杂 design

常见坑

坑 1：把 TPM 当 counts 喂给 DESeq2

`DESeqDataSetFromMatrix` 检查不到这种错误 —— 因为 TPM 也是数值矩阵。**但 dispersion 估计会完全失真**，p 值可信度归零。Salmon / kallisto 输出的是 TPM 和 estimated counts，**用 `tximport(..., type = "salmon")` 拿 counts 列**，不要直接读 quant.sf 的 TPM。

坑 2：design 公式里把要检验的变量放前面

`design = ~ dex + cell` 和 `design = ~ cell + dex` 的拟合系数完全一样，但 **`results()` 默认提取最后一个变量的对比**。要检验的变量放最后是社区惯例，否则 results() 会拿错对比。

坑 3：用未 relevel 的因子默认顺序

R 里因子默认按字母顺序排（"trt" 在 "untrt" 前面），这导致 DESeq2 把 "trt" 当参考、"untrt" 当对比 —— **log2FC 符号正好反过来**。养成习惯：建立 dds 之前显式 `relevel(factor, ref = "untrt")`，或在 `results()` 里 `contrast = c("dex","trt","untrt")` 写清楚。

坑 4：用原始 log2FC 做下游而不是 shrunken

低表达基因（baseMean < 10）的 LFC 噪声很大，会出现 |LFC| > 5 但 padj 接近 1 的情况。**做下游分析（GSEA 排序、可视化、报告）一律用 `lfcShrink(..., type = "apeglm")`**，原始 LFC 只在写 supp table 时保留。

坑 5：直接信 padj < 0.05，不看 p 值分布

p 值直方图是一个免费的"模型质量检查"：

- 健康：左侧高峰 + 右侧均匀
- U 型：模型欠拟合（design 漏了变量）或数据有奇异样本
- 整体偏左：有未控制的混杂因子

跑完 DESeq2 务必 `hist(res$pvalue)` 一秒钟看一眼，比单看 padj 信息量大得多。

下载资源

bulk02_deseq2_sci.R
12 KB

下载 airway DESeq2 完整脚本 ↗

不想本地装环境？在 BioF3 上跑

下一步

接着深入：

- 03 功能富集：GO / KEGG / GSEA — 拿到差异基因表后第一件事是富集分析，把基因翻译成通路
- 04 火山图、热图与富集可视化 — 同一份 DE 结果可以做出能直接发表的 6 张图

横向延伸：

- 05 多时间点与 LRT — 你的设计里有时间序列 / 剂量响应时，Wald 检验不够，要换 LRT
- 06 批次效应 — 多批次数据的 design 公式怎么写
- 07 多工具对比 — 想知道 edgeR / limma-voom 的结果会不会很不一样

参考资源

- DESeq2 官方教程
- RNA-seq workflow (Bioconductor)
- apeglm 方法
- Himes et al. 2014, airway 数据源

速答

Q: ORA 和 GSEA 有什么区别? A: ORA 需要先筛差异基因（有阈值），再看基因集是否富集；GSEA 不筛阈值，用全部基因排序看基因集在排序里的分布。GSEA 更灵敏，能发现 ORA 漏掉的温和变化。

Q: 基因 ID 转换 SYMBOL / ENSEMBL / ENTREZID 哪个用? A: clusterProfiler 的 enrichGO / gseKEGG 需要 ENTREZID；画图展示用 SYMBOL。用 `clusterProfiler::bitr()` 或 `AnnotationDbi::mapIds()` 转换。

Q: GO 的 BP / MF / CC 选哪个? A: 生物学过程（BP）最常报告，分子功能（MF）和细胞组分（CC）作为补充。三者都跑，重点解读 BP。

Q: GSEA 的 NES 和 pvalue 怎么解读? A: NES（Normalized Enrichment Score）> 1 正向富集，< -1 负向富集；|NES| 越大越显著。pvalue < 0.05 或 padj < 0.25 视为显著。

04 功能富集：GO / KEGG / GSEA

差异基因本身只是中间产物。真正有用的是"这些差异基因对应哪些通路、哪些功能"。功能富集分析用的就是**已经注释好的基因集合**，看差异基因列表和这些集合有没有非随机的重叠。

本章用 02 DESeq2 差异表达 里 airway 的差异结果，走一遍最常用的三种富集：

- **GO**（Gene Ontology）：基因的生物过程 / 分子功能 / 细胞组分
- **KEGG**：手工整理的代谢和信号通路
- **GSEA**（Gene Set Enrichment Analysis）：不依赖阈值，直接看排序里的分布

ORA vs GSEA 两种思路

这两种方法是互补的，值得花一段说清楚它们的区别。

维度	过表达分析（ORA）	基因集富集（GSEA）
输入	一张"显著差异基因"名单	所有基因的排序（按某个 statistic）
需要阈值	需要（padj < 0.05、|LFC| > 1 之类）	不需要
擅长回答	强效应基因扎堆在哪些通路	整条通路整体是否往一个方向偏
典型工具	`enrichGO` 、 `enrichKEGG`	`gseGO` 、 `fgsea`
弱点	阈值非常敏感	对排序 statistic 选择敏感

工程上的建议：两种都跑，交叉印证。ORA 给出"非常确定的、强效应的通路"；GSEA 能抓到那些"每个基因效应不大，但整条通路一致下降"的情况（比如线粒体电子传递链）。

举一个具体例子说明为什么两种都要跑：

某次实验做药物处理 vs 对照。用 padj < 0.05 & |LFC| > 1 阈值得到 50 个 DE 基因。

- **ORA 结果**：top 通路 "免疫激活"、"细胞凋亡" — 来源是 50 个 DE 基因里恰好有 8 个属于这两条通路
- **GSEA 结果**：top 通路 "线粒体电子传递链 (OXPHOS)" — 这条通路上 100 多个基因每个 LFC 只有 0.3-0.5，没一个过阈值，但**整条通路一致下调**

这种"分散的、每个基因小而全通路一致"的信号，**ORA 完全看不见**。OXPHOS 在很多药物处理项目里的下调正是这种类型。漏掉 GSEA 等于漏掉这一类机制。

基因 ID 的坑

clusterProfiler 的 `enrichKEGG` 、 `enrichGO` 都要求 **Entrez ID** 或能通过 `OrgDb` 映射到 Entrez。airway 的 counts 矩阵用的是 Ensembl gene ID（`ENSG00000000003` 这种），需要先做一步转换：

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

id_map <- AnnotationDbi::select(
  org.Hs.eg.db,
  keys     = rownames(dds),
  keytype  = "ENSEMBL",
  columns  = c("ENTREZID", "SYMBOL")
)
```

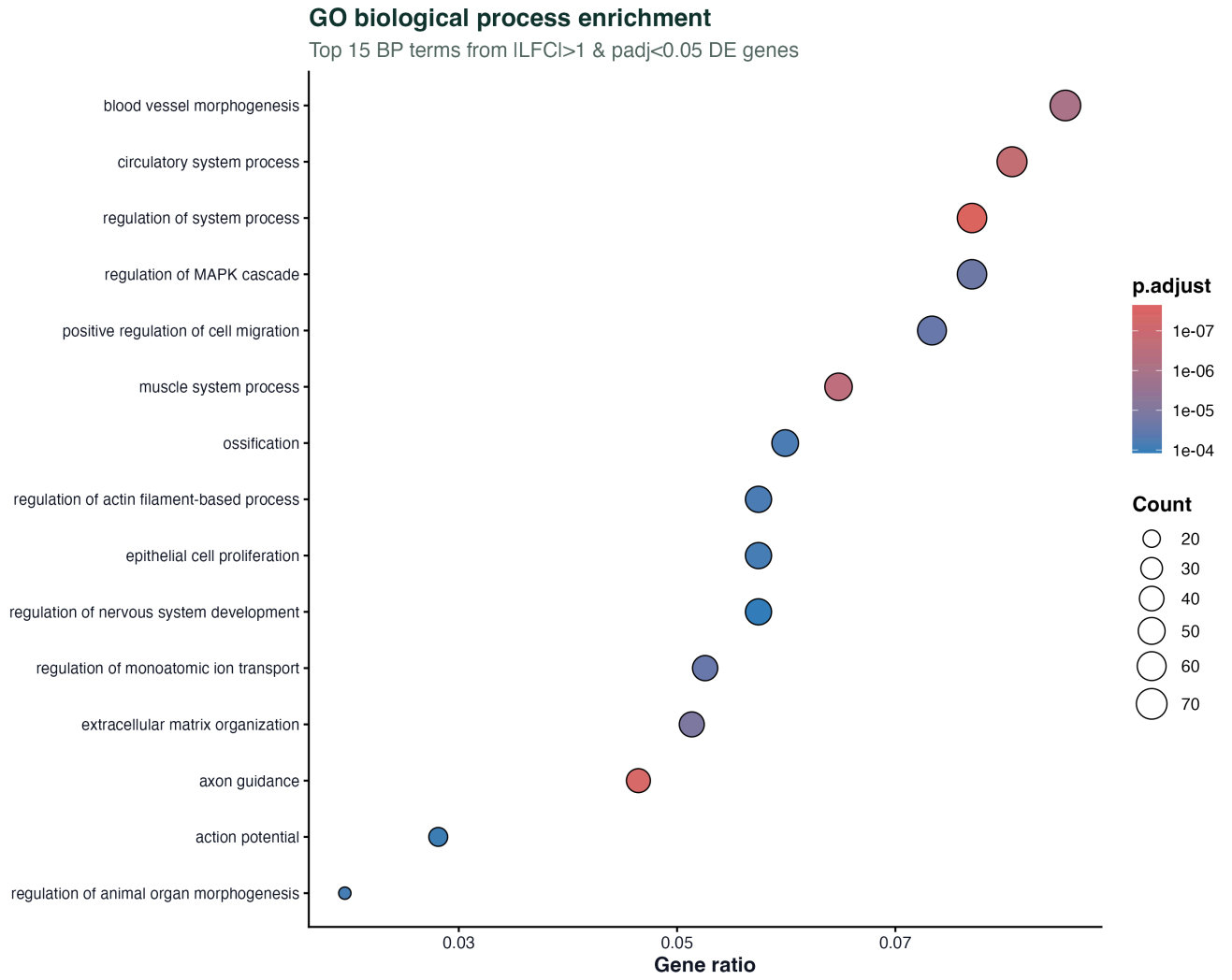
一个 Ensembl ID 可能对应多个 Entrez ID（历史重命名、重复注释），也可能对应 `NA` 。做富集前要 `distinct()` 和 `filter(!is.na())` 清理一遍。

真实示例：airway 上的 GO + KEGG + GSEA

配套脚本 `bulk03_enrichment_sci.R` 接着上一章 DESeq2 的结果走：把 DE 结果的 Ensembl 映射成 Entrez → 对显著 DE 基因做 GO BP/MF/CC 和 KEGG 的 ORA → 对全部基因按 `sign(LFC) * -log10(p)` 排序做 GSEA。

```
Rscript scripts/bulk03_enrichment_sci.R
```

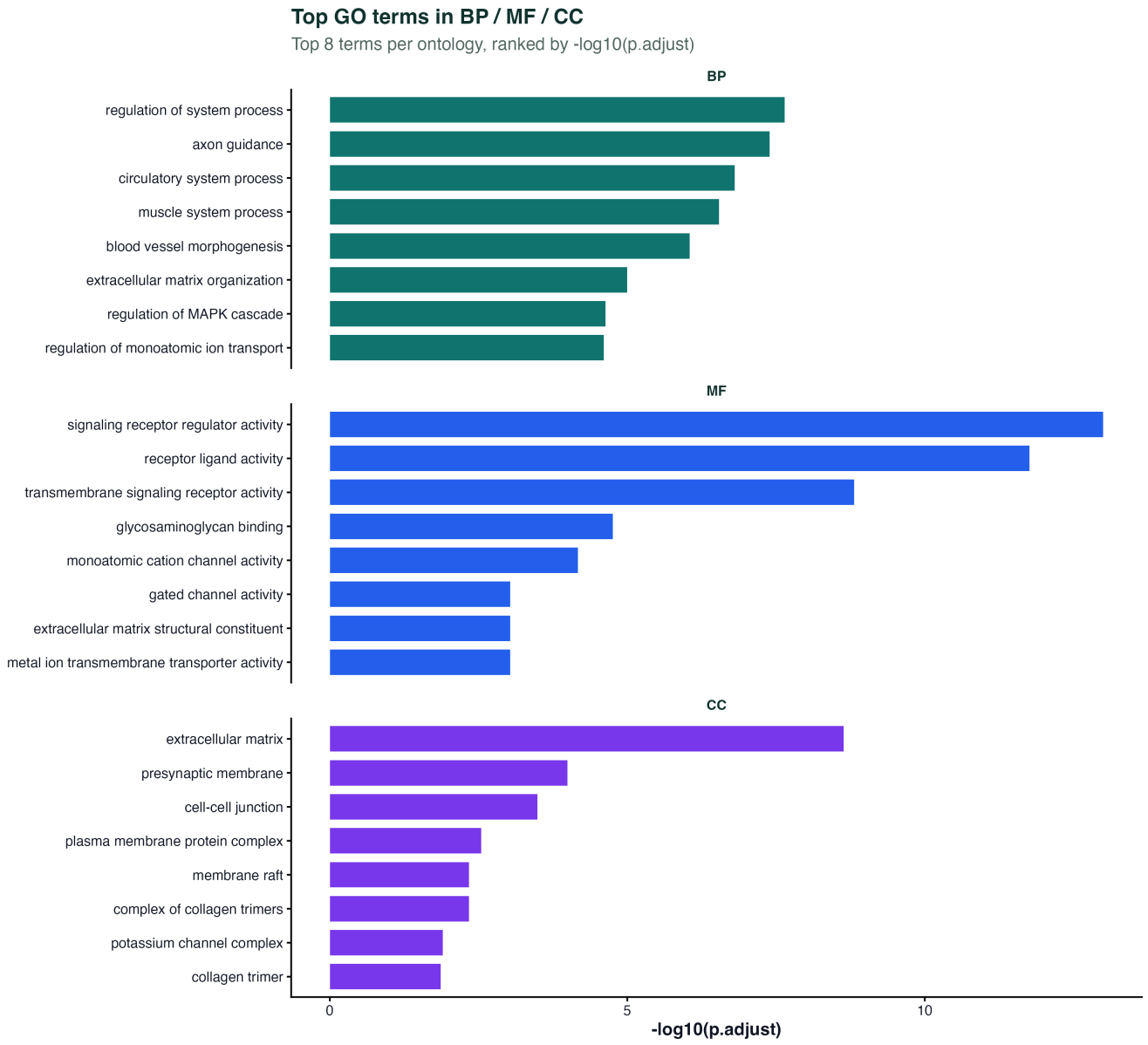
图 1：GO BP 富集 dotplot



显著 DE 基因在 GO biological process 里的 top 15 个 term。横轴 `Gene ratio` 是"term 里命中的 DE 基因数 / 输入的 DE 基因总数"，点大小也代表命中数量，颜色是 p.adjust。

airway 是地塞米松处理的气道上皮，跑出来能看到典型的糖皮质激素响应：类固醇激素响应、细胞增殖调控、上皮细胞分化之类。这种"药理作用 → 预期通路"的对应就是富集分析最直观的验证。

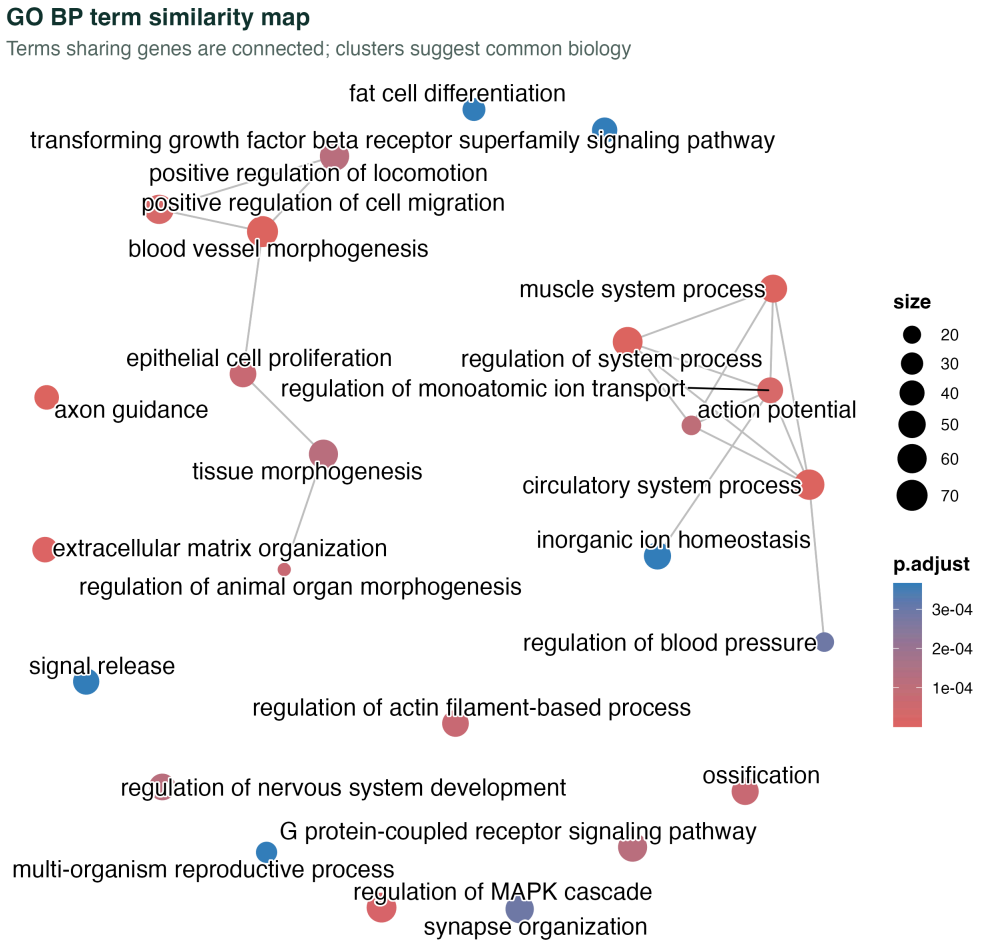
图 2：GO 三大类（BP/MF/CC）的 top 条目



GO 分三个子本体：BP（生物过程）、MF（分子功能）、CC（细胞组分）。三个一起看能给出更全的解读：BP 告诉你"在做什么"，MF 告诉你"通过什么分子机制"，CC 告诉你"主要发生在哪里"。

图里每个子本体取 top 8，颜色区分。写 Methods 段的时候这张图能同时列出三类核心 term，比单独看 BP 更完整。

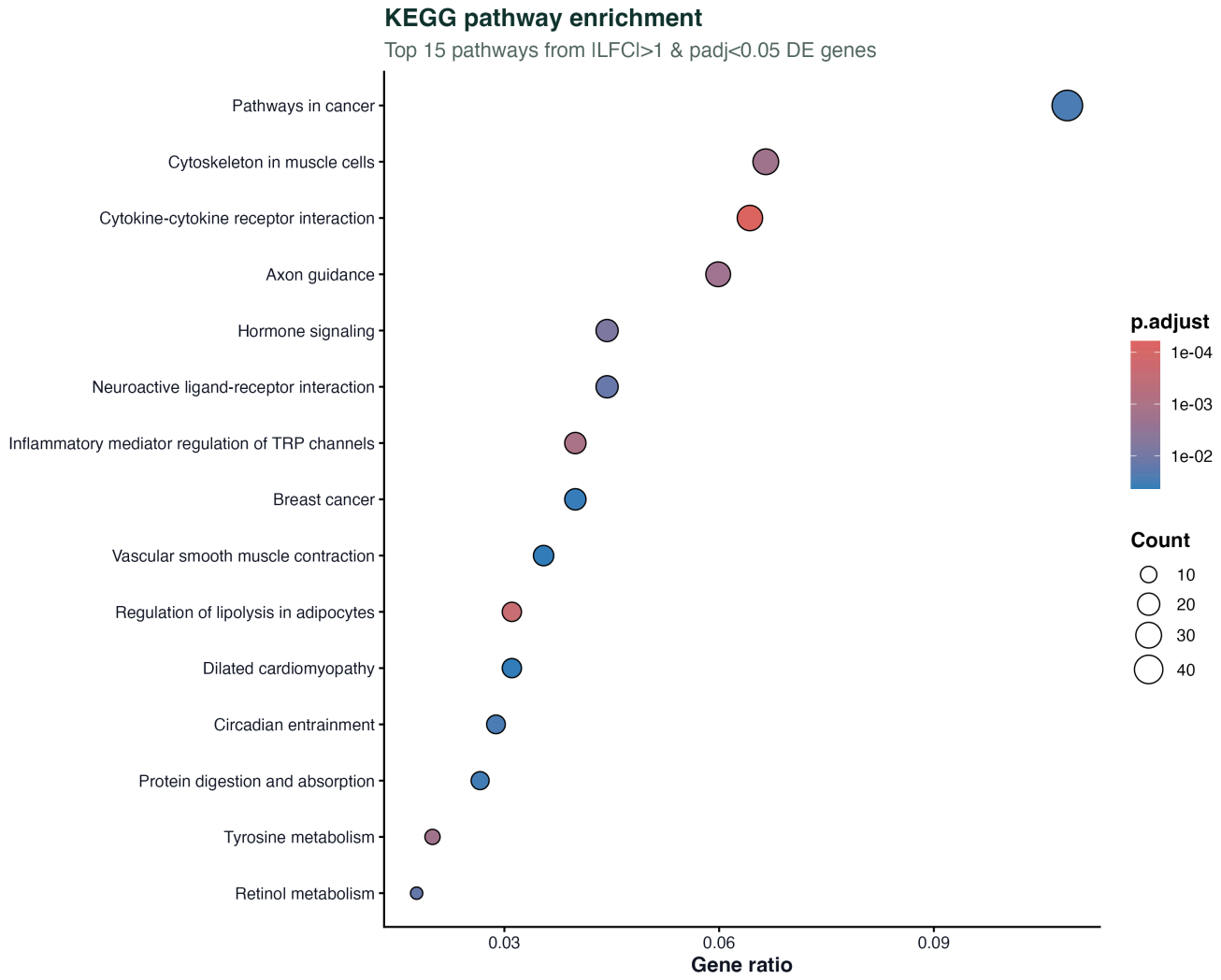
图 3：GO term 相似度网络



很多 GO term 之间共享基因（比如 "regulation of cell proliferation" 和 "positive regulation of cell proliferation"）。`emapplot` 把 term 之间的基因重叠画成网络，聚在一起的 term 通常讲的是同一组生物学。

这张图的价值是**去冗余**：直接看 top 15 dotplot 容易发现很多 term 其实在说同一件事；emapplot 能帮你把它们归成几个有意义的簇。

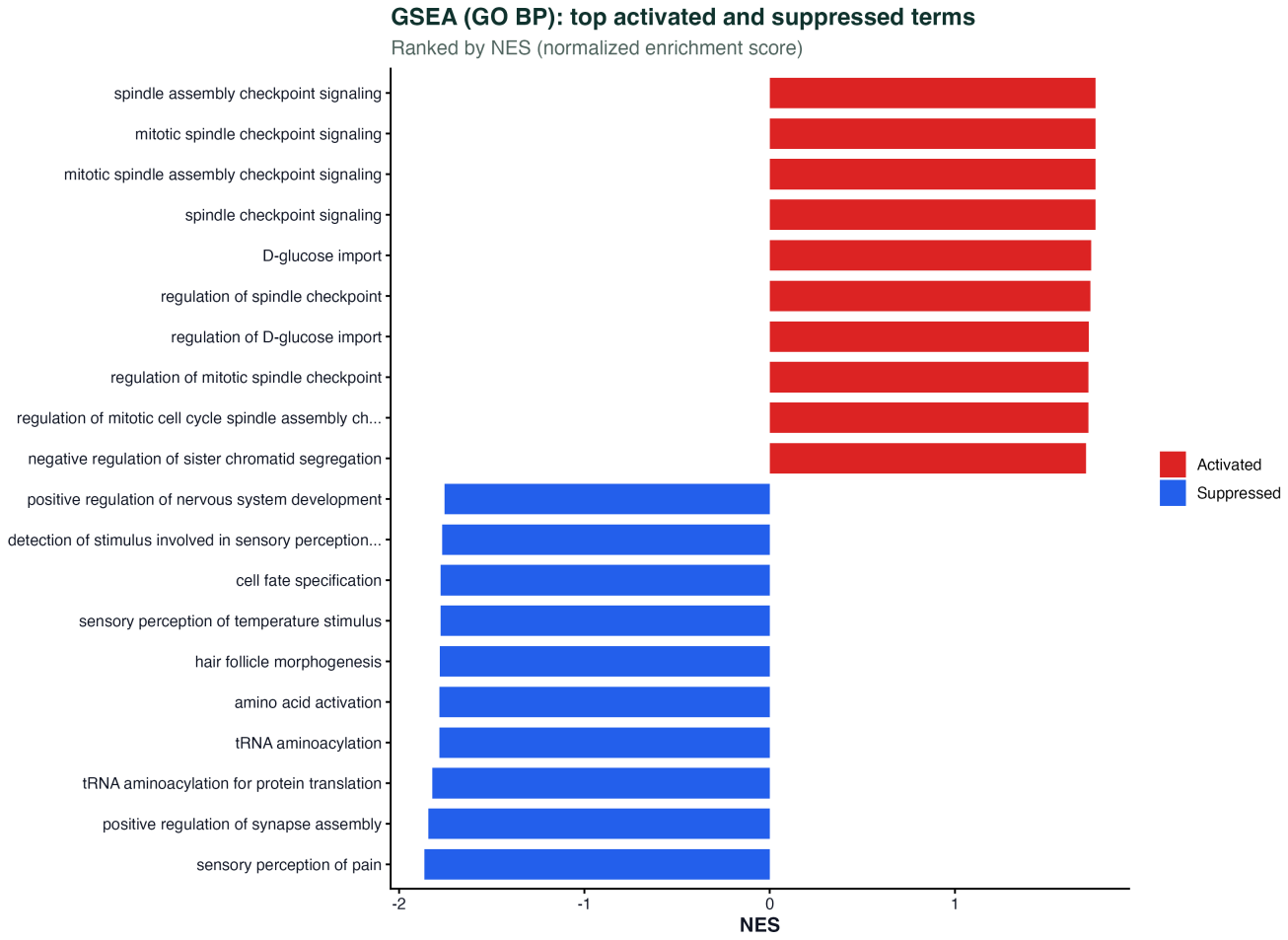
图 4：KEGG pathway 富集



KEGG 比 GO 更贴近"通路"的直觉 —— 每个 pathway 都有一张官方示意图（比如 hsa04110 是 cell cycle），非常适合写报告和做插图。`enrichKEGG` 默认在线查 KEGG 的 REST API，所以这一步需要能访问 `rest.kegg.jp` 。

注意 KEGG 的 pathway ID 前缀是物种代号（`hsa` 是人），做小鼠要换成 `mmu` 。

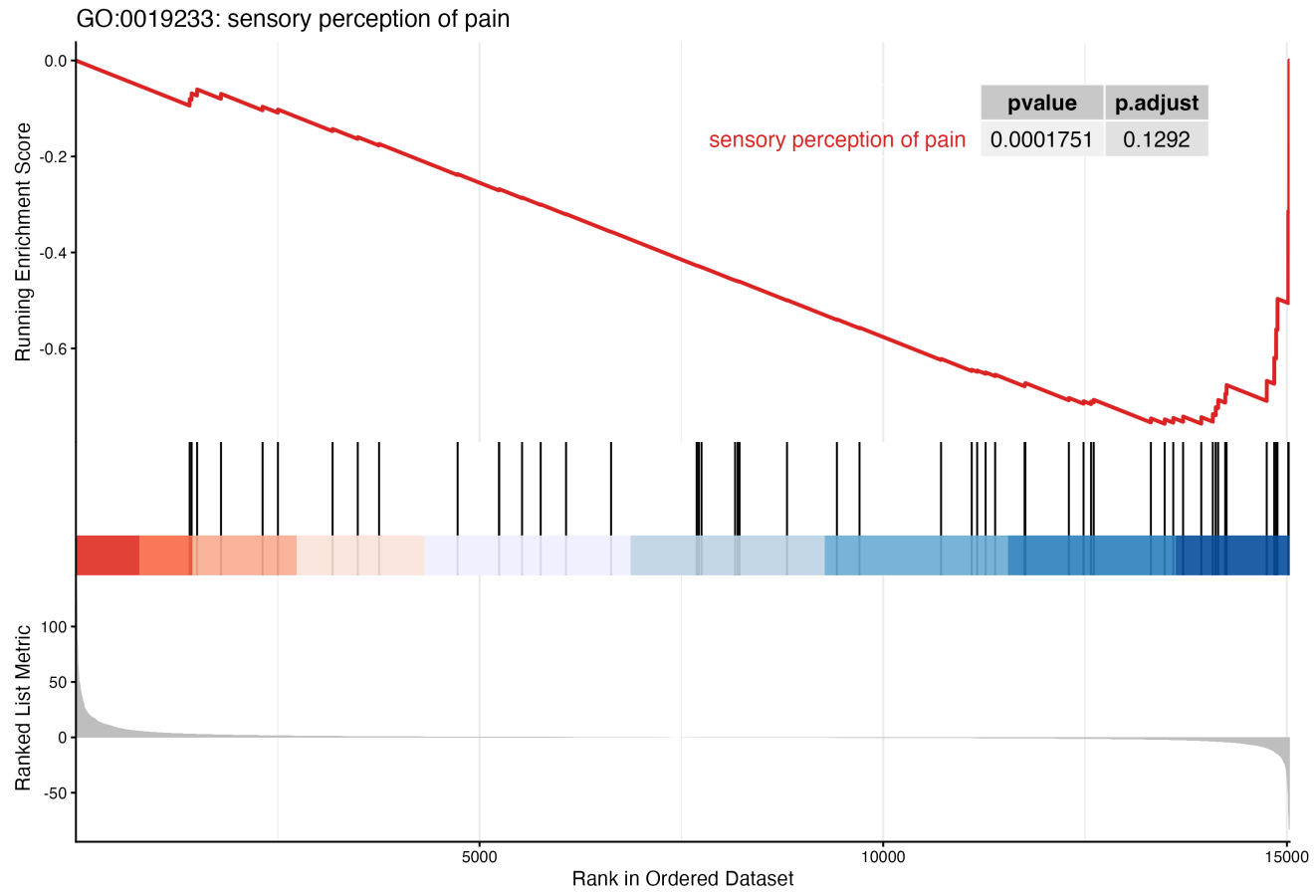
图 5：GSEA top 通路（瀑布图）



GSEA 按 NES（normalized enrichment score）给每个 term 一个"整体方向性"的打分。红色（上调富集）表示这条通路的基因整体在处理组里往上走；蓝色（下调富集）表示整体往下。

对 airway，可以看到糖皮质激素相关的响应通路在正方向显著富集，和 ORA 的 dotplot 互相验证。

图 6：GSEA running-score 图



对最强 NES 的那个 term 画的经典 GSEA 图。上半部分的曲线是"在排序里走到这个基因时累计的 enrichment score"，峰值越偏离 0 越显著；中间条形表示每次击中一个该通路里的基因时的位置；下半部分是原始 ranking score 的分布。

这张图最核心的是峰值出现在**排序的哪一端**。峰在左边（高排名那端）= 通路整体上调；峰在右边 = 整体下调；峰在中间没有倾向 = 这条通路不显著。

套到自己数据上

脚本的入口是 `res` 对象（`DESeq2::results()` 返回），从其他工具（edgeR 的 `glmQLFTest`、limma 的 `topTable`）得到的差异表只要有 `log2FoldChange` / `pvalue` / `padj` 这些列也一样跑。

几个需要根据实际调的参数：

- **物种**：`org.Hs.eg.db` 换成自己物种的 OrgDb（小鼠是 `org.Mm.eg.db`），`enrichKEGG(organism = "mmu")`
- **阈值**：默认 `padj < 0.05 & |LFC| > 1` 比较严。小项目或效应较弱的数据可以放宽到 `padj < 0.1 & |LFC| > 0.58`
- **基因集规模**：`minGSSize = 15` / `maxGSSize = 500` 过滤极端 term，数据小的时候可以降下界
- **GSEA 排序统计量**：这里用的是 `sign(LFC) * -log10(pvalue)`；也有人用 `stat`（Wald）或 `log2FoldChange * -log10(padj)`

MSigDB 和 fgsea

`clusterProfiler::gseGO` 和 `enrichGO` 的基因集只来自 GO。要用 MSigDB 的 Hallmarks、C2（canonical pathways）、C7（immune signatures）这些非常实用的集合，最稳的搭配是 `fgsea + msigdbr`：

```
library(msigdbr)
library(fgsea)

# Hallmarks
hallmarks <- msigdbr(species = "Homo sapiens", collection = "H") |>
  split(x = .$entrez_gene, f = .$gs_name)

fgsea_res <- fgsea(pathways = hallmarks, stats = rank_vec, minSize = 15, maxSize = 500)
```

注意 `msigdbr` 26.x 把基因集改成运行时从 zenodo 下载。国内直连 zenodo 经常慢或失败，可以手动下载一份 `msigdb.v2024.1.Hs.json.gz` 放到本地，或用 `msigdbr::msigdbr_download_db()` 指定镜像。

常见坑

坑 1：用 SYMBOL 输入 enrichKEGG

`enrichKEGG` 期望 ENTREZID。直接喂 SYMBOL 会跑得动但**结果几乎全空**（命中率 < 5%）—— 因为 KEGG 内部用的是 ENTREZID。**永远先 `bitr()` 转一道**。

坑 2：把 background 当成"全部基因"

ORA 的 universe（背景集）应该是**这次实验里所有被检测到的基因**，不是物种全部基因。设错背景会让显著性虚高（小池子里抓鱼当然显著）。`enrichGO(universe = rownames(dds))` 显式传，不要省。

坑 3：GSEA 的 ranking 用了 padj 而不是 LFC

`-log10(padj)` 都是正值，没有方向，GSEA 跑出来的 enrichment 就只有"越早出现越富集"的方向，看不出激活/抑制。**正确做法是 `sign(LFC) * -log10(pvalue)` 或直接 `stat`** —— 必须有正负。

坑 4：富集结果不去冗余直接报

GO 出来的 top 30 可能只代表 5 个独立的生物学模块，剩下都是冗余 term。**emapplot / clusterProfiler::simplify() 去冗余后再写报告**。读者看到 30 条很相似的 term 会觉得在凑数。

坑 5：KEGG 在线 API 时不时挂

`enrichKEGG(use_internal_data = FALSE)` 默认在线查 rest.kegg.jp，有时国内访问慢或断流。**长期项目缓存一份本地 KEGG.db 或者用 `download_KEGG()` 离线**，关键分析不要依赖外部 API 当场可用。

下载资源

bulk03_enrichment_sci.R
12 KB

下载 airway 富集分析完整脚本 ↗

不想本地装环境？在 BioF3 上跑

GO/KEGG ORA 和 GSEA 各有独立的在线工具，都支持从 DESeq2 结果一键导入。

下一步

接着深入：

- 04 火山图、热图与富集可视化 — 富集结果出来后怎么做成可发表的 6 张图

横向延伸：

- 02 DESeq2 差异表达分析 — 富集结果质量取决于 DE 表质量，回头检查 DE 是否做对了
- Subramanian 2005 GSEA 原始论文 — GSEA 算法的方法论起点
- clusterProfiler book — 富集分析的全套工具，超出本章范围的工具都在这里

参考资源

- clusterProfiler 教程
- GSEA 官方
- MSigDB
- fgsea 文档
- enrichplot 文档

05 火山图、热图与富集可视化

前三章把数据、差异和富集跑完，这一章只做一件事：**把结果变成能直接用于论文和汇报的图**。每张图都配一个最常见的用途，源码一起放出来。

配套脚本 `bulk04_visualization_sci.R` 把 DESeq2 + 富集的流程重跑一次（继承自 02 和 03），然后输出 6 张可视化成品：

```
Rscript scripts/bulk04_visualization_sci.R
```

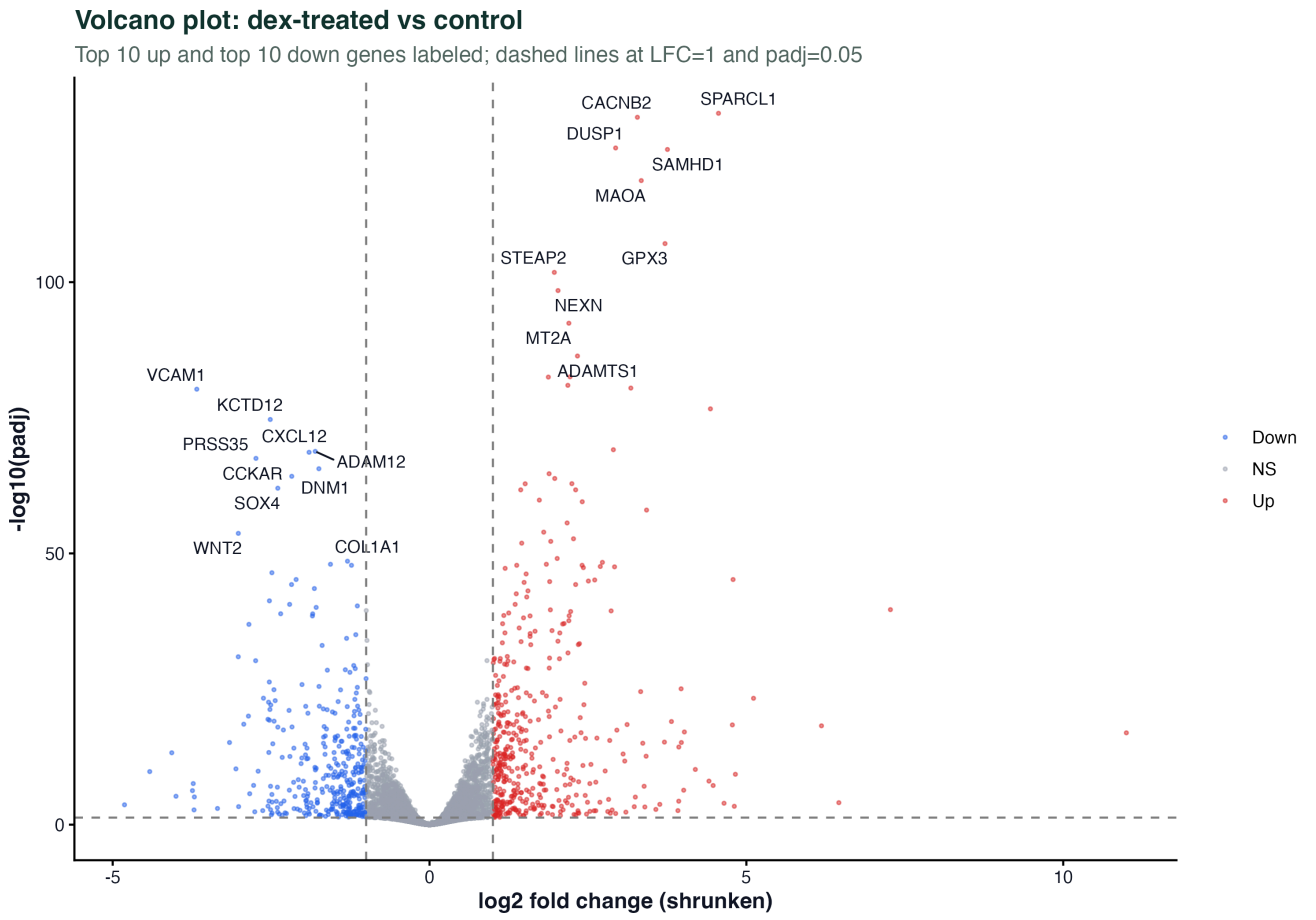
一篇论文的 figure 怎么排

具体画图之前，先想清楚每张图在论文里要承担的角色。一篇典型的 RNA-seq 文章 main figure 大约 4-6 张图，结构通常是：

Figure	内容	本章对应
Fig 1	**样本概览 + QC** — PCA、距离热图、库大小	02 章的图 1-3
Fig 2	**差异分析整体** — 火山图 + top DE 热图	本章 图 1-2
Fig 3	**关键基因深入** — 已知通路基因表达分布	本章 图 3
Fig 4	**功能富集** — KEGG / GO bubble + cnetplot	本章 图 4-5
Fig 5	**跨数据集验证**	看项目

一张图回答一个问题，不要堆 8 个 panel 的 supplementary 怪物。每张图问自己："读者看完这张图能讲清楚什么？" 讲不清楚就拆开。

图 1：火山图



火山图是每篇转录组文章的标配 —— 横轴是 log2 fold change，纵轴是 `-log10(padj)`。每个点是一个基因，颜色表示它在哪个象限。两条虚线标出 padj = 0.05 和 |LFC| = 1 的阈值。

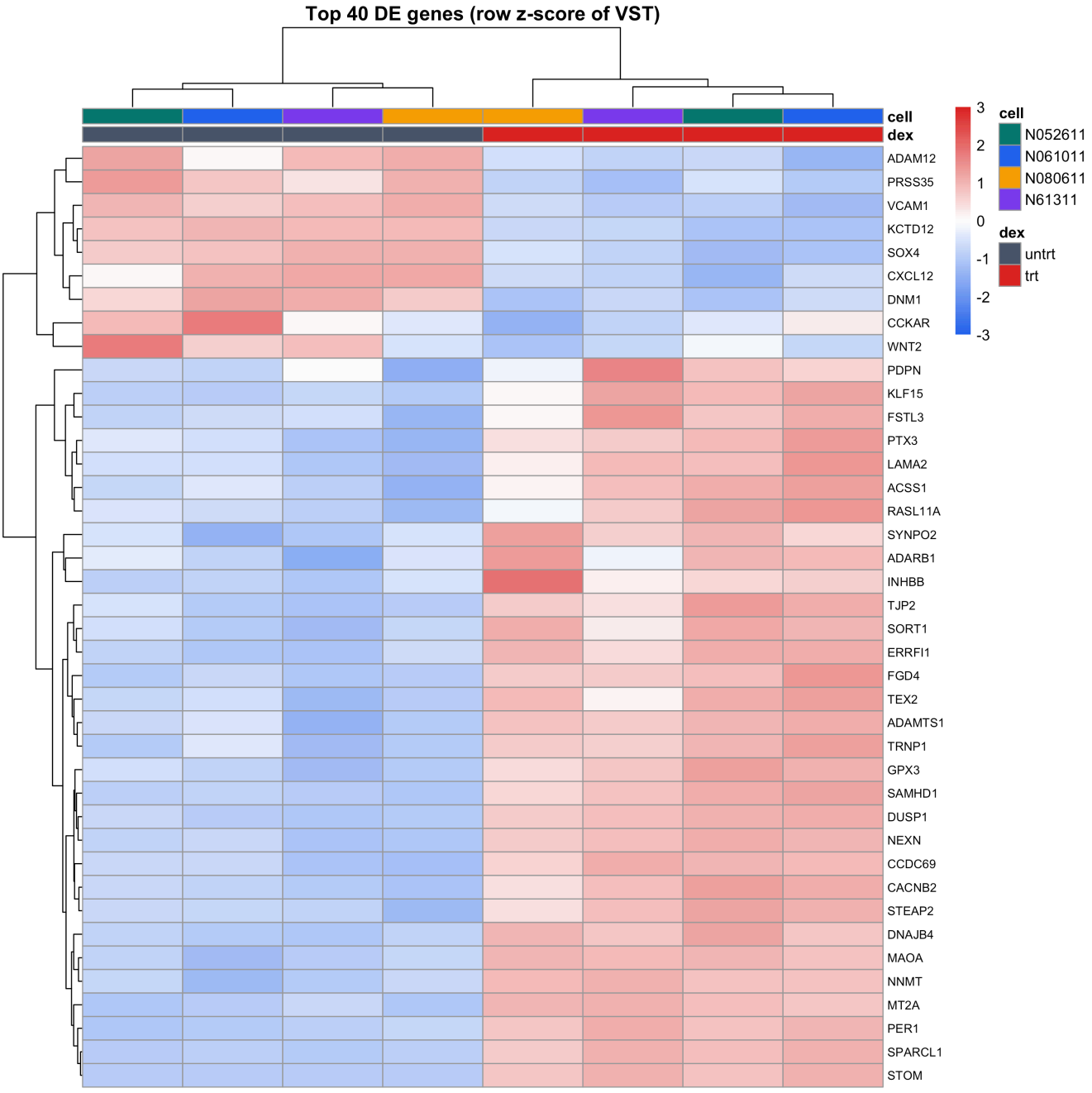
这张图的关键是**标注**：默认标全部显著基因会完全盖掉点云。脚本里只标 top 10 up + top 10 down（按 padj 排序），用 `ggrepel` 自动避让。

核心代码：

```
ggplot(vol, aes(x = log2FoldChange, y = -log10(padj), color = sig)) +
  geom_point(size = 0.6, alpha = 0.5) +
  geom_text_repel(data = label_df, aes(label = SYMBOL), ...) +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed") +
  geom_hline(yintercept = -log10(0.05), linetype = "dashed")
```

改 `sig` 列的阈值、`head(10)` 的数量、颜色都很方便。

图 2：Top DE 基因热图



取 padj 最小的 40 个基因，在 VST 归一化后做 row z-score，画热图。列注释标出了处理状态和细胞系。

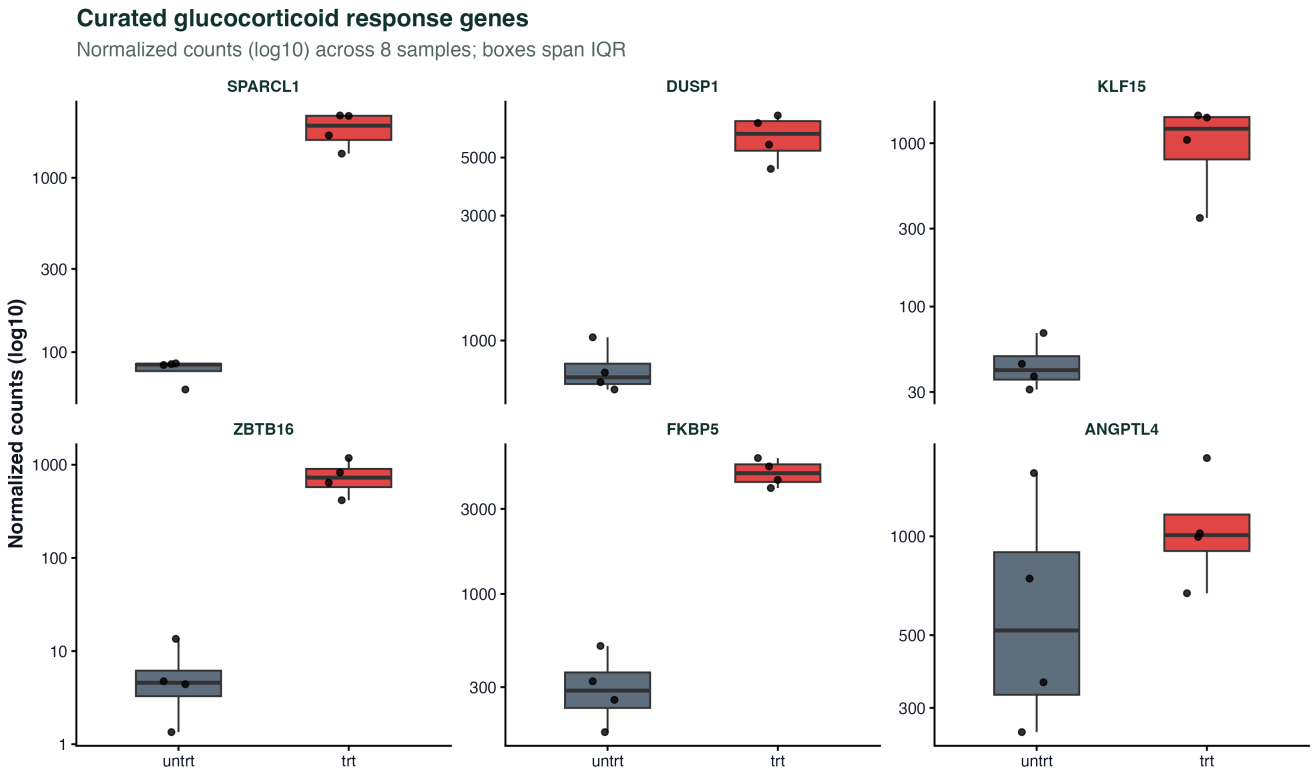
这张图在报告里的作用是给火山图"补一刀" —— 火山图只能看统计显著性，热图能直接看到"这些基因在每个样本里到底是什么表达水平"。列聚类会把处理样本和对照样本各自聚到一起；行聚类会把上调和下调基因分开。

用 `pheatmap`：

```
mat_z <- t(scale(t(mat)))  # row z-score
pheatmap(mat_z,
  annotation_col = anno_col,
  color = colorRampPalette(c("blue", "white", "red"))(100),
  breaks = seq(-3, 3, length.out = 101))
```

`breaks` 参数很重要：限制颜色范围能避免个别极端基因把色阶拉歪。`[-3, 3]` 是 z-score 的常用范围。

图 3：精选基因的表达分布

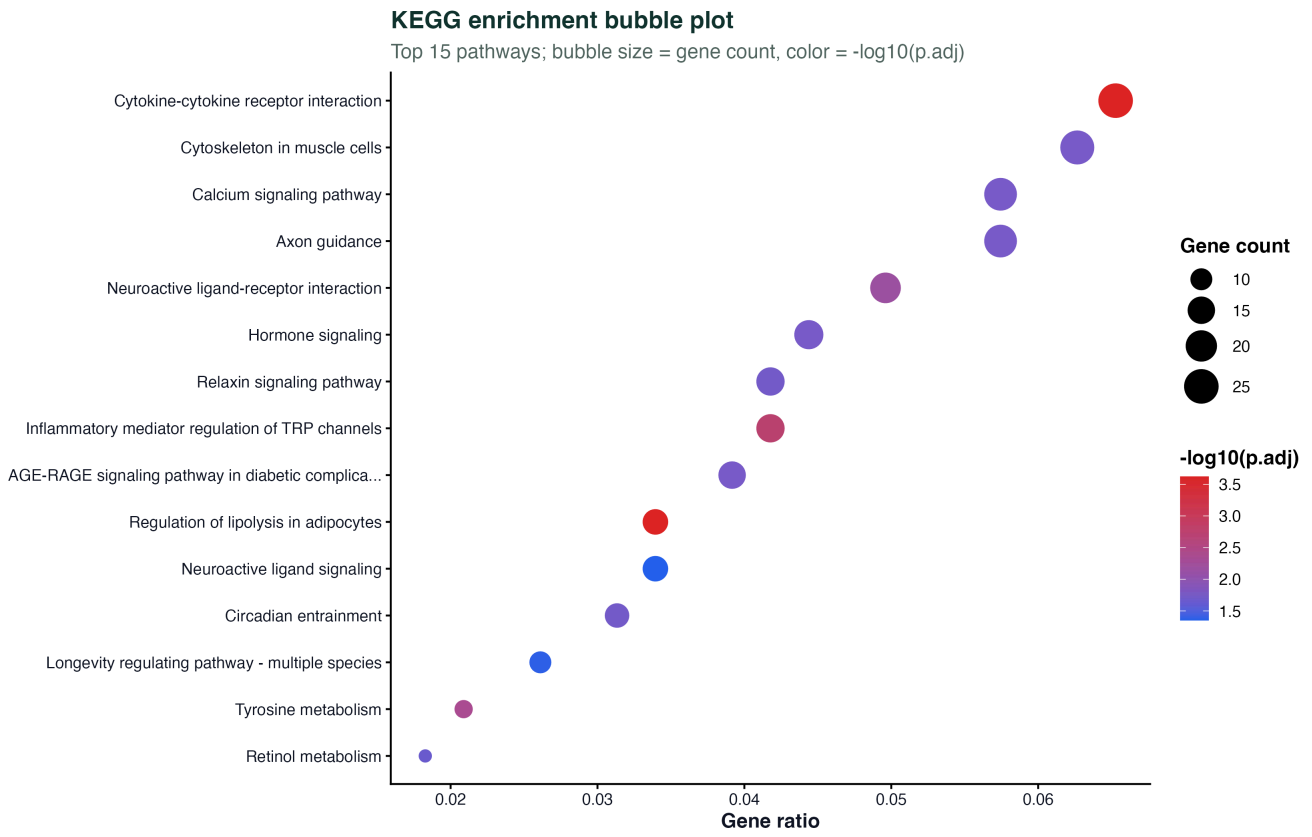


选几个已知的糖皮质激素响应基因（`DUSP1` / `FKBP5` / `ANGPTL4` / `KLF15` / `SPARCL1` / `ZBTB16`）画 boxplot + jitter，y 轴是 log10 的 normalized counts。

这种"已知通路验证"的图在讨论部分特别有用：**文献说这些基因是地塞米松响应基因，我们的数据里也确实看到了这个响应**，能直接把自己的数据和已有知识对接上。

```
ggplot(count_long, aes(x = dex, y = count, fill = dex)) +
  geom_boxplot() + geom_jitter() +
  facet_wrap(~ symbol, scales = "free_y") + scale_y_log10()
```

图 4：KEGG 富集气泡图

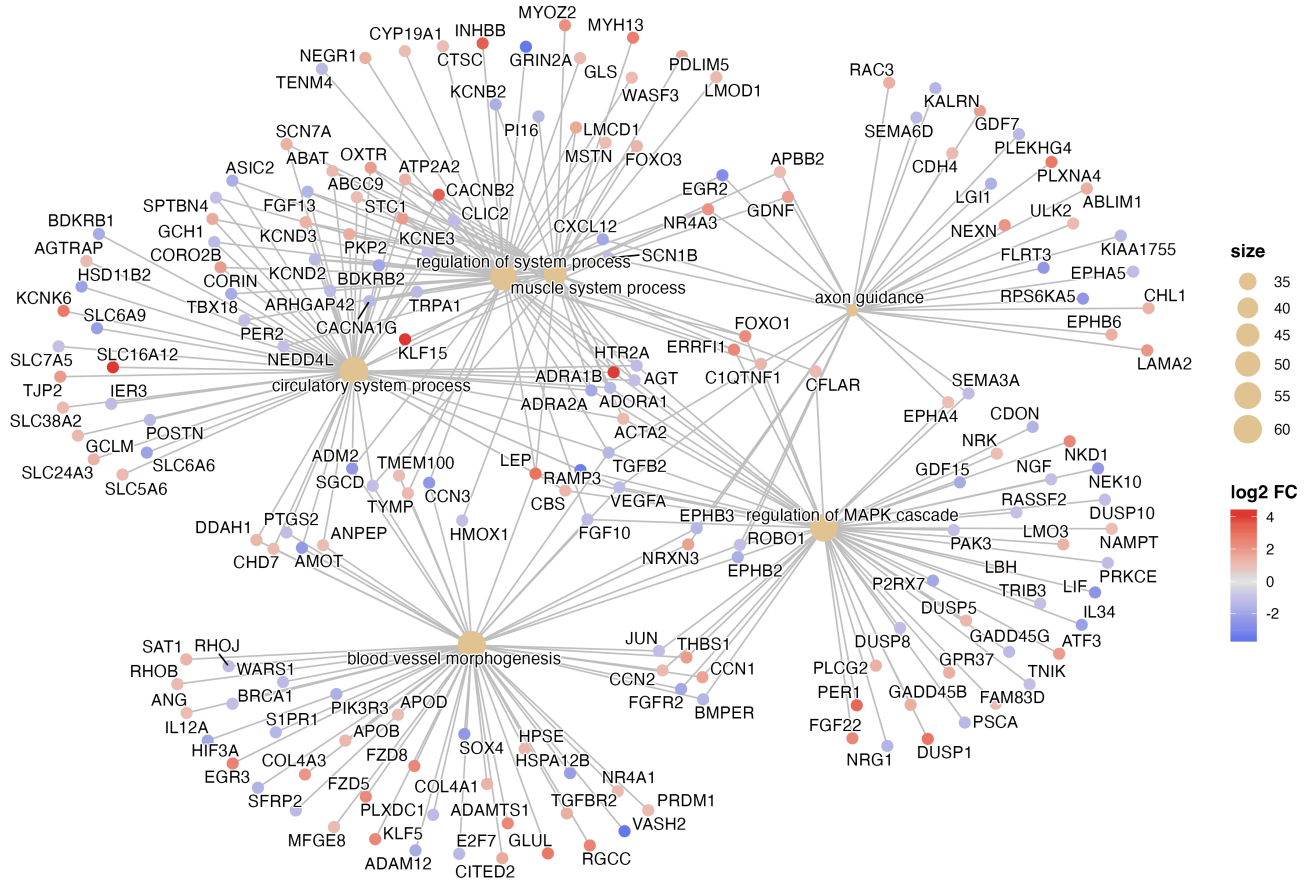


比 `dotplot(ekegg)` 的默认版本更有信息量：点大小是 gene count，颜色是 `-log10(p.adj)`，横轴是 gene ratio。一张图同时显示"通路覆盖度"、"显著性"、"通路里命中了多少基因"三个维度。

这个 bubble 是纯 ggplot 写的，不依赖 enrichplot，完全可以移植到任何富集结果（`enrichGO`、`enrichPathway`、`GSEA` 都行）。

```
ggplot(ekegg_df,
       aes(x = GeneRatio_num, y = reorder(Description, GeneRatio_num),
           size = Count, color = -log10(p.adjust))) +
  geom_point() +
  scale_color_gradient(low = "blue", high = "red") +
  scale_size_continuous(range = c(3, 9))
```

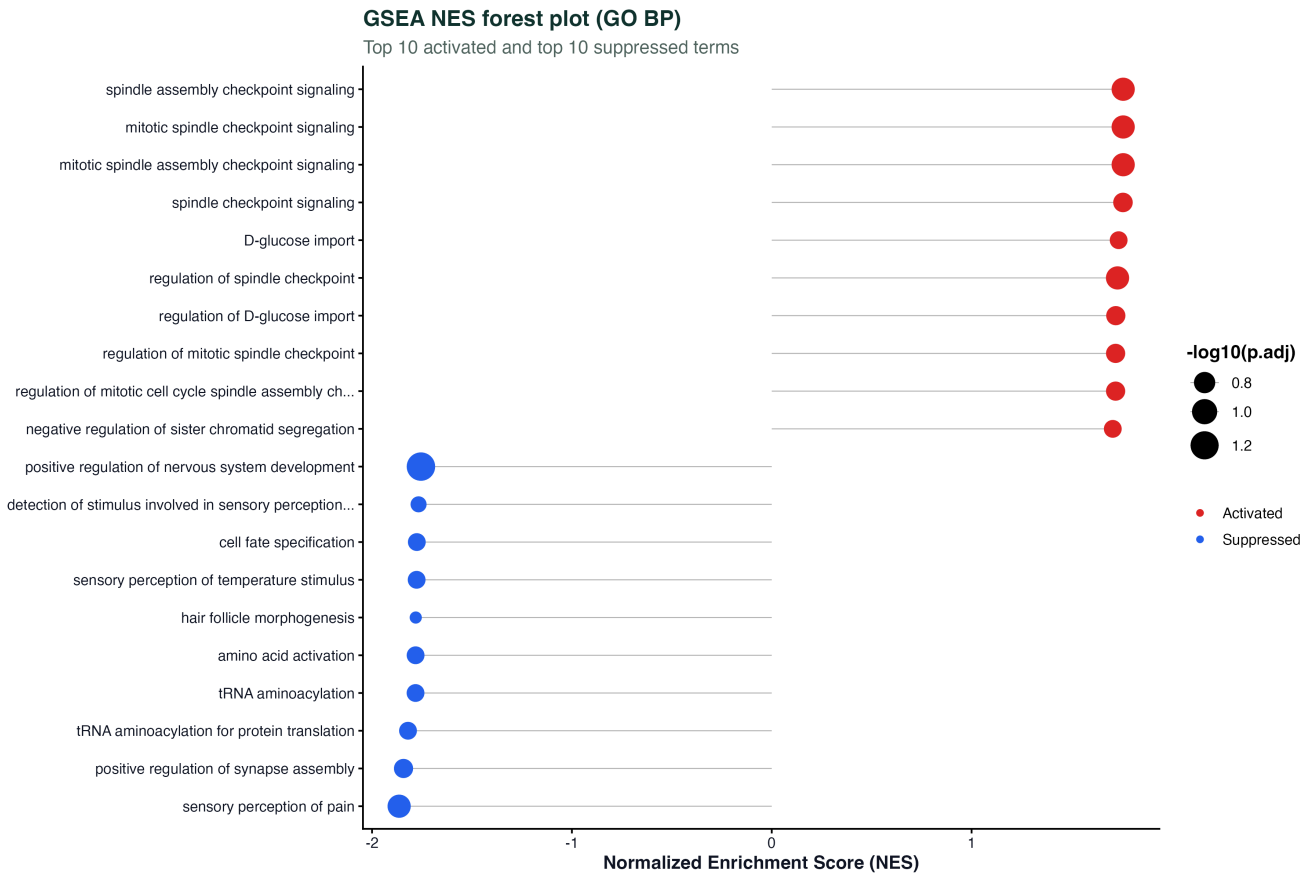
图 5：cnetplot 基因-通路网络



`enrichplot::cnetplot` 把 GO term 和驱动这些 term 的基因画成二部图：浅灰色大圆点是 term，彩色小点是基因，颜色按 log2FC 着色。

这张图的价值是**一眼看到"哪些基因被多条通路共享"** —— 共享基因往往是生物学上最关键的 master regulator。讨论部分里可以直接点出"图里 A 基因被 3 条通路共享，说明它很可能是这次处理的上游枢纽"。

图 6：GSEA NES forest 图



GSEA 的瀑布图能表达排序，但多个 term 一起看、带显著性的信息，forest 图更合适。横轴是 NES，纵轴是 term 名字，点大小是 `-log10(p.adj)`，颜色区分激活 / 抑制。

相比 waterfall，这张图额外能读出 **NES 到 0 的距离**（segment 线段）和 **显著性强度**（点大小），信息密度更高。

常见坑

- **颜色**：火山图用蓝 / 红 / 灰三色是惯例，别换成 viridis 这种连续色；读者会困惑"到底哪边是上调"
- **标注**：标签超过 30 个就没法读；能标 top 10-20 就够了
- **基因 ID**：热图 / 火山图要标 SYMBOL 不是 Ensembl。一定要在最后可视化之前把 SYMBOL 映射上
- **坐标轴范围**：火山图如果有极端 padj（比如 1e-300），用 `pmax(padj, 1e-300)` 封顶避免压扁整张图
- **导出**：论文图建议 `dpi = 300` 直接存 PNG；不要存 JPG（有压缩伪影）；有时需要 TIFF / PDF 向量图，见 `ggsave()` 的 `device` 参数

补充几条容易踩的坑：

坑：热图行做 z-score 但没限制 breaks

`scale(t(mat))` 出来的 z-score 范围可能 [-10, 30]，pheatmap 默认会把整个范围映射到色阶 — 结果 99% 的基因颜色都接近白色，几个极端基因把图"拉歪"。**`breaks = seq(-3, 3, length.out = 101)`** 是常用上限，超出范围的值压到端点。

坑：火山图 padj = 0 的基因显示为 Inf

部分 DE 工具会输出 padj = 0（数值下溢），`-log10(0) = Inf` 让 ggplot 报错或丢点。**`pmax(padj, 1e-300)` 封顶**再画。

坑：cnetplot 的基因数太多挤成一团

`cnetplot(showCategory = 5)` 默认显示前 5 条 term，如果 term 之间共享基因多，几十个基因挤成一坨可读性归零。**显式 `showCategory = 3` + `node_label = "category"` 只标 term 不标基因**，更清晰。

坑：ggsave PDF 字体丢失

`ggsave(file = "fig.pdf")` 在某些字体下会输出"被吃掉部分字体"的 PDF，投稿时被拒。**用 `device = cairo_pdf` 强制嵌入字体**，或者直接存 png 300dpi。

坑：facet_wrap 的 free_y 让 boxplot 跨基因不可比

`facet_wrap(~ symbol, scales = "free_y")` 让每个基因的 y 轴独立 — 视觉上"差异都看上去差不多大"。**展示绝对幅度差异时用 `scales = "fixed"`**，让读者能看出"基因 A 的 fold 比基因 B 大多少"。

下载资源

bulk04_visualization_sci.R
16 KB

下载 airway 可视化完整脚本 ↗

下一步

bulk RNA-seq 专栏目前到这一章为止。后面会补：

- 多时间点或剂量响应的差异分析（`LRT` vs `Wald`）
- 批次效应更复杂的场景（RUVseq、SVA）
- 面向发表的结果组织（fgsea + MSigDB Hallmarks + Reactome）
- 单细胞 vs bulk 的方法学对照

接着深入：

- 05 多时间点与 LRT 检验 — 设计里有时间序列 / 剂量梯度时，本章的火山图 + 热图都要换思路
- 08 可复现分析报告与结果交付 — 这些图怎么组织进 R Markdown 报告，怎么交付给合作者

横向延伸：

- FigCode 火山图 — 在线把你的 DE 表替换 demo 数据出图
- FigCode 热图 — 复杂热图（多列 annotation、自定义聚类）的样板代码

参考资源

- ggplot2 官方
- ggrepel
- pheatmap
- ComplexHeatmap
- EnhancedVolcano
- enrichplot 可视化库

06 多时间点与 LRT 检验

前几章做的都是"两组比较"：对照 vs 处理。真实项目里经常遇到更复杂的设计 —— 一条时间曲线，剂量梯度，多个处理条件 —— 这时候两两 Wald 检验就不够了。DESeq2 给出了两个重要工具：

- **设计公式里加交互项**：`~ strain + minute + strain:minute`，同时估计主效应和交互
- **LRT（似然比检验）**：检验"完整模型相比简化模型是否显著更好拟合"，一次拿到所有交互项的综合 p 值

本章用 fission 数据做演示 —— 裂殖酵母在氧化应激下的时间序列：野生型 vs `atf21del` 突变体，6 个时间点（0/15/30/60/120/180 min），每个组合 3 个生物学重复，共 36 个样本。

多因子设计的核心问题

时间序列、剂量梯度、多条件实验都属于"多因子设计"。要写对 design 公式，先想清楚两个问题：

1. 你想检验的是主效应、交互效应，还是某个特定对比？

问题	公式	检验方式
"时间有影响吗？"（不管菌株）	`~ minute`	LRT, reduced = `~ 1`
"菌株有影响吗？"（不管时间）	`~ minute + strain`	LRT, reduced = `~ minute`
"菌株对时间响应有影响吗？"	`~ strain + minute + strain:minute`	LRT, reduced = `~ strain + minute`
"30 分钟时菌株差异多大？"	上面同公式	Wald on `strainmut.minute30`

2. LRT 还是 Wald？

- **Wald**：一个 contrast 一个 p 值。"A vs B 在某条件下"
- **LRT**：一组 contrast 一个综合 p 值。"任何 contrast 是否显著"

LRT 适合"我不知道哪个具体对比显著，但想知道有没有任何不一样"。Wald 适合"我已经知道我要看哪个对比"。

时间序列项目典型流程：**先 LRT 筛出"响应基因"，再 Wald 锁定具体时间点**。本章脚本图 5 就是这个用法。

Wald 检验不够用的时候

DESeq2 的默认 `results()` 做的是 Wald test，回答"某一个对比（contrast）的效应是否显著"。如果问题是：

- "是否存在任何一个时间点上两个菌株有差异？" — Wald 做不了，要对每一对时间点都检验再合并 p 值，麻烦且不严格
- "药物剂量和基因表达是否有剂量-响应关系？" — Wald 只能看某两个剂量的对比
- "某条通路的变化在治疗组和对照组里是否不同？" — 这就是典型的交互项问题

LRT 的解决办法是：**给一个"完整模型"和一个"简化模型"，问数据在完整模型下的似然是否显著高于简化模型**。如果是，说明完整模型里多出来的那些参数（通常是交互项）承载了真实信号。

具体到 fission：

```
# full 包含交互项、reduced 不包含
dds <- DESeqDataSet(fission, design = ~ strain + minute + strain:minute)
dds <- DESeq(dds, test = "LRT", reduced = ~ strain + minute)
res_lrt <- results(dds)
```

`res_lrt` 里每个基因的 padj 回答的是：**这个基因对时间的响应，是否依赖菌株？** 答案显著的基因就是"时间响应被 atf21 影响的基因"。

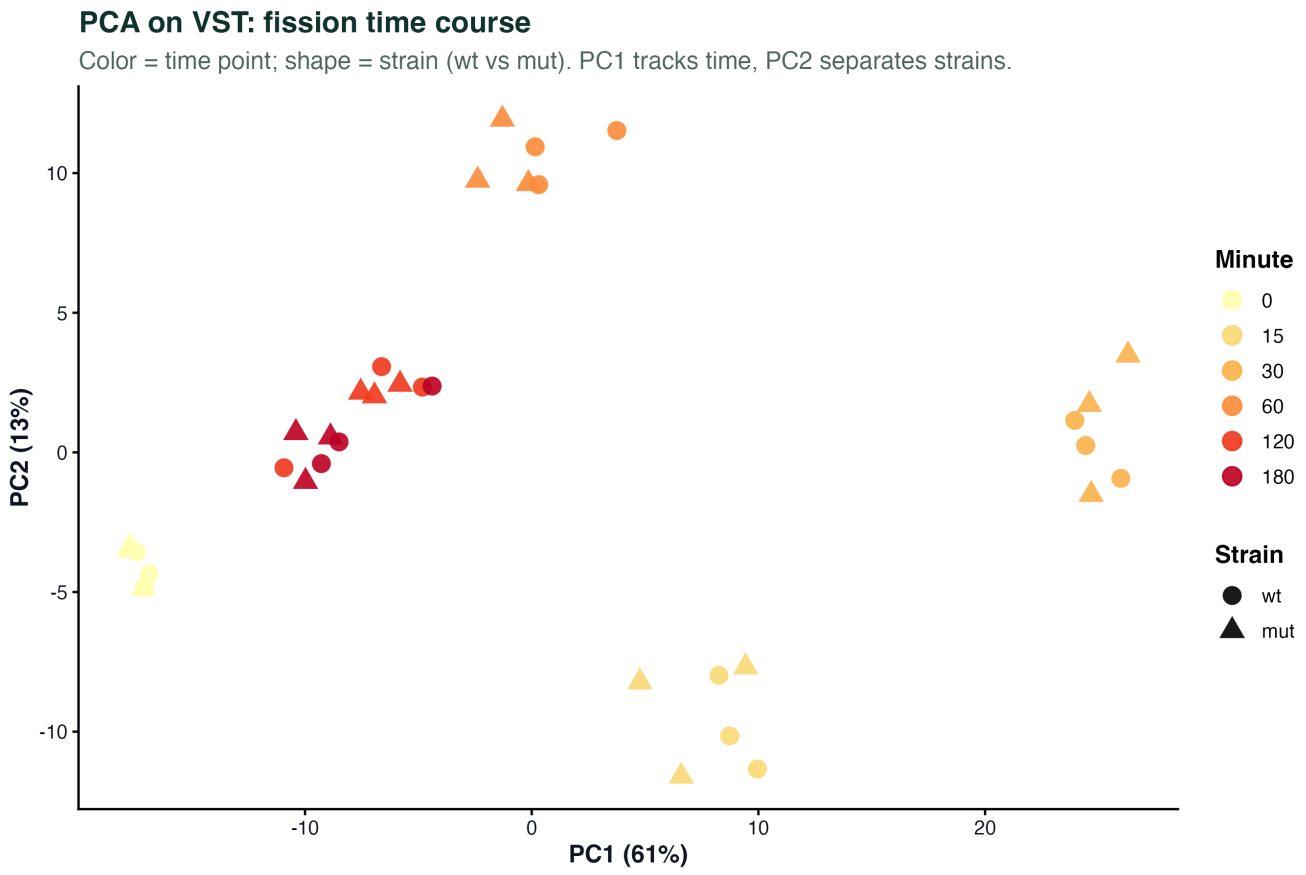
真实示例：fission 时间序列

配套脚本 `bulk05_timecourse_sci.R` 加载 `fission` → 建立含交互的设计 → LRT 检验 → 画 6 张图：

```
Rscript scripts/bulk05_timecourse_sci.R
```

fission 这份数据在"时间主效应"上信号非常强（细胞受到氧化应激），但 `strain:minute` 的交互信号比较弱（atf21 对响应的影响不那么戏剧）。实际跑下来 LRT 在 padj < 0.05 下只过了几个基因 —— 这本身就是一个重要的教学点：**强交互不是所有生物学问题都存在**。

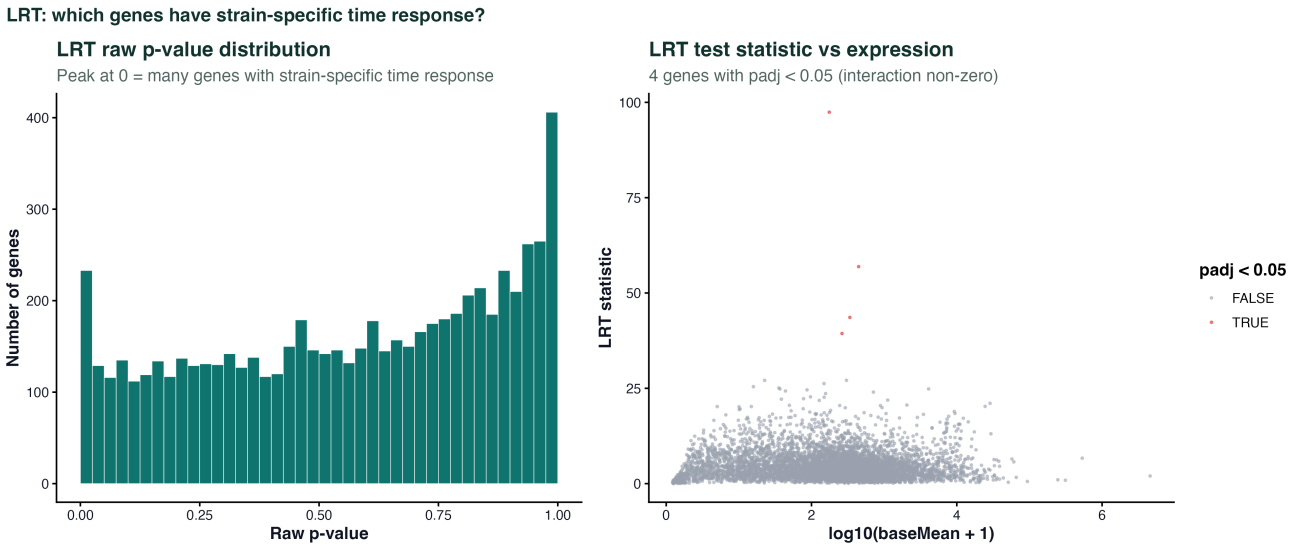
图 1：PCA 看整体结构



颜色表示时间，形状区分菌株。PC1 几乎就是时间轴 —— 从 0 min（浅色）到 180 min（深红），样本沿着一条轨迹排开。PC2 上两个菌株有一定分离，但比 PC1 上的时间效应弱得多。

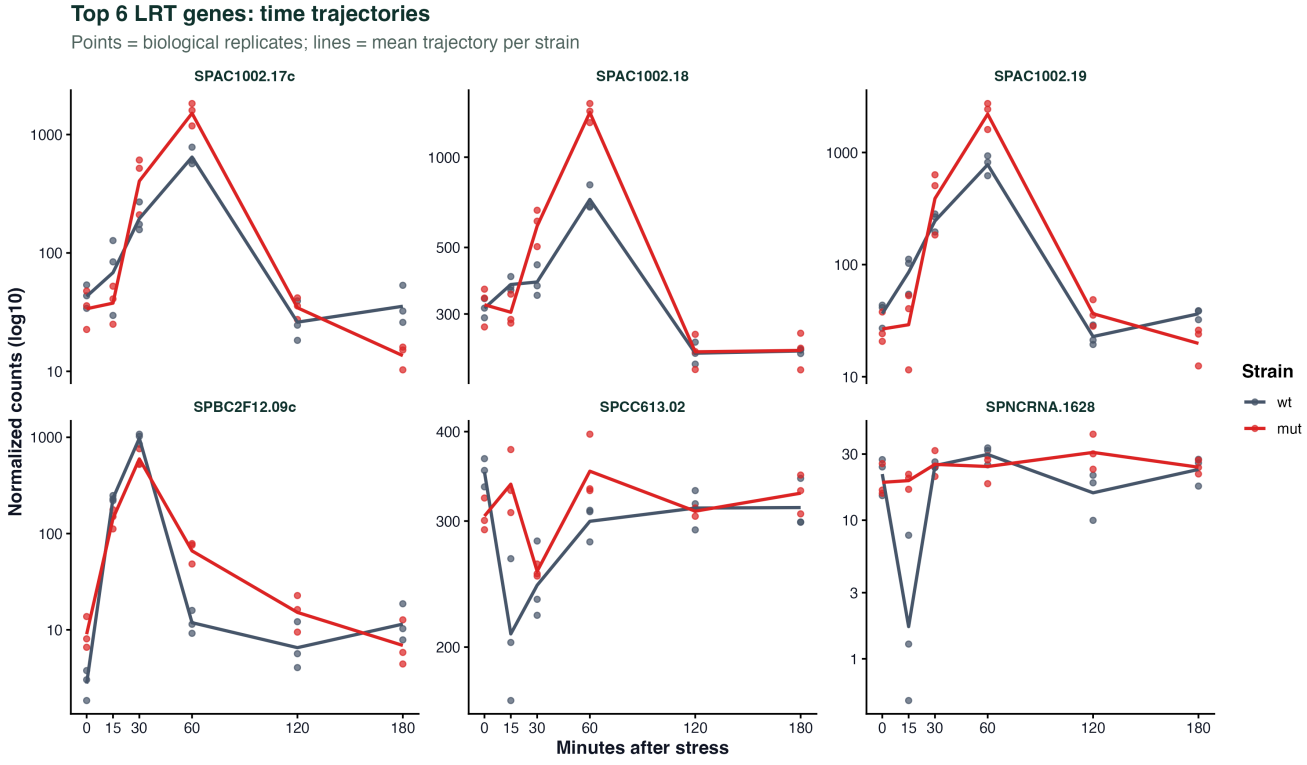
这张图回答的第一个问题：时间是最主要的变异来源，说明设计公式里把 `minute` 当主效应是对的。

图 2：LRT p 值分布与统计量



左边是 LRT 原始 p 值的直方图。和 Wald 的直方图一样：0 附近有 peak 说明有真信号，剩余部分接近均匀。 右边横轴是 log10(baseMean)，纵轴是 LRT 统计量。LRT 统计量越大说明"完整模型比简化模型好"的程度越强。高表达基因（右侧）通常统计量更大 —— 这是 RNA-seq 的常见现象：高表达基因数据多、功效高。

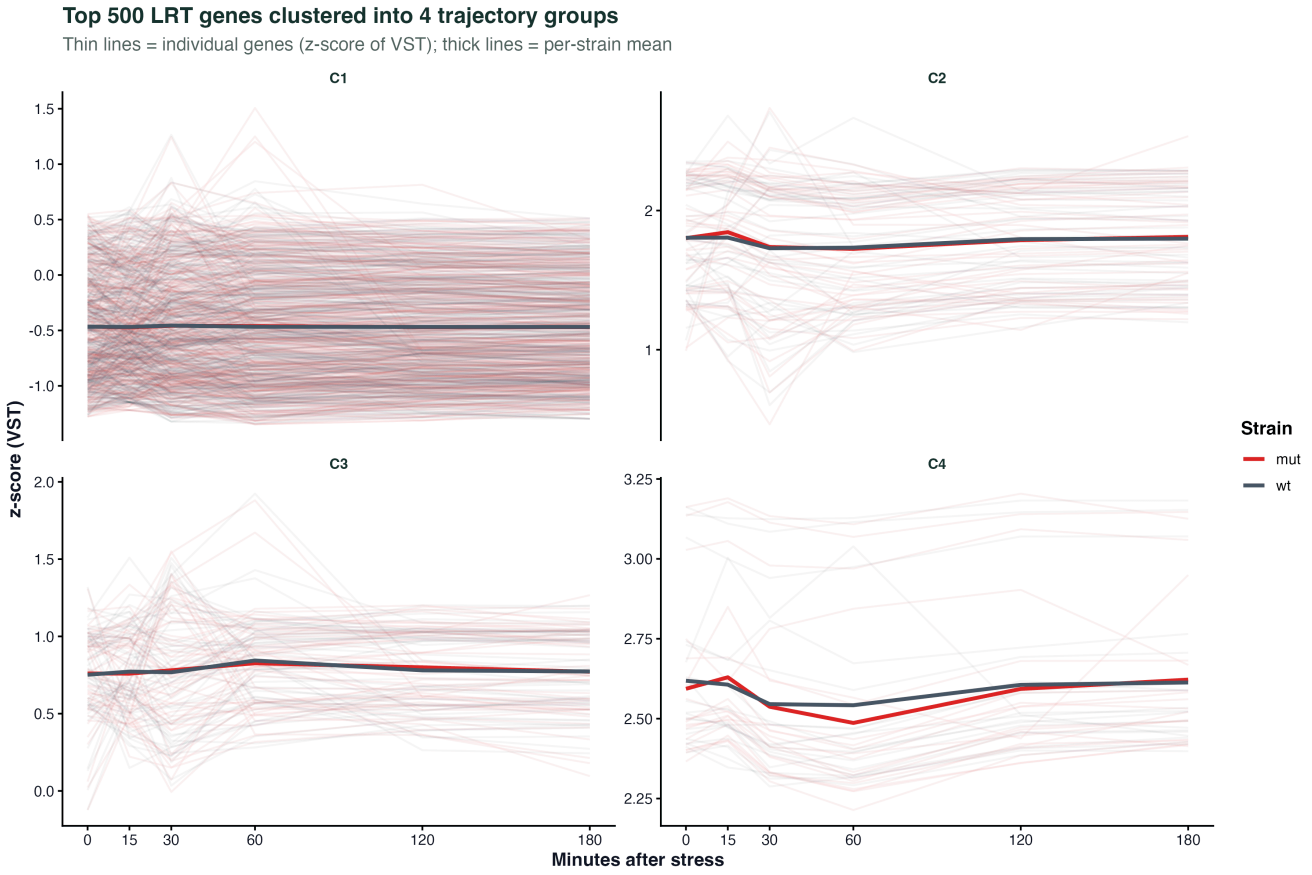
图 3：Top LRT 基因的时间轨迹



对 padj 最小的 6 个基因，把两个菌株的 normalized counts 画成时间曲线（点 = 重复，粗线 = 同时间点的均值）。可以看到 wt 和 mut 的曲线在某些时间点"分叉" —— 这就是 LRT 捕获的信号：两条曲线的形状不一样。

这种图在写论文时是"讲故事"最核心的图：把抽象的 LRT 统计量具象成"这个基因在 30 分钟的时候开始在突变体里下降"。

图 4：聚类的时间轨迹



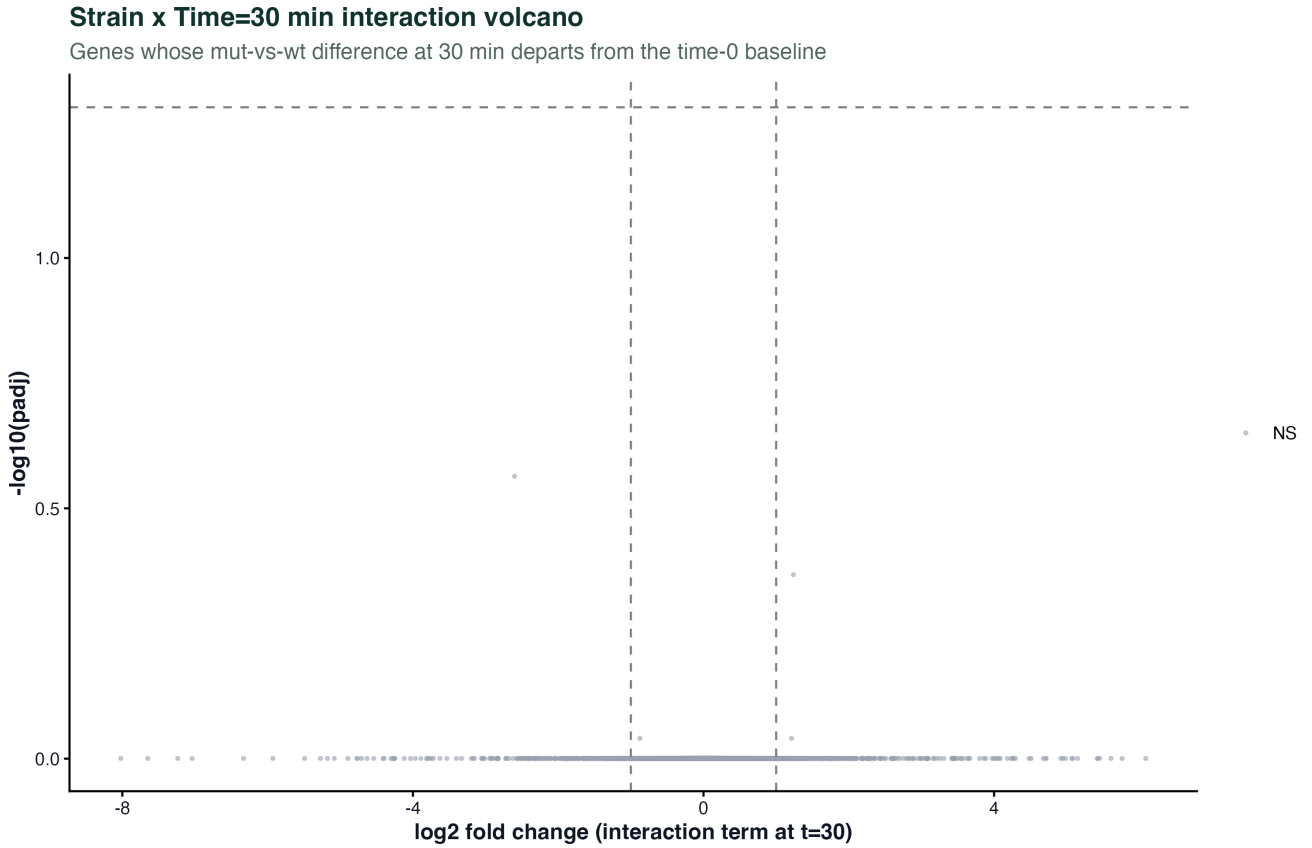
取 LRT 排名前 500 的基因，做一次层次聚类，分 4 组。每组里每个基因画一条半透明细线（z-score），最上面的粗线是两个菌株各自的均值。

这张图的信息量很大：

- 哪些基因"早响应、早下降"（C1）
- 哪些"晚响应、持续升高"（C3）
- 哪些在两个菌株间方向一致但幅度不同（C2）
- 哪些在一个菌株里响应、另一个菌株里不响应（C4）

真实项目里这张图往往比 dotplot 更能直接生成生物学假设。

图 5：特定时间点的交互 volcano

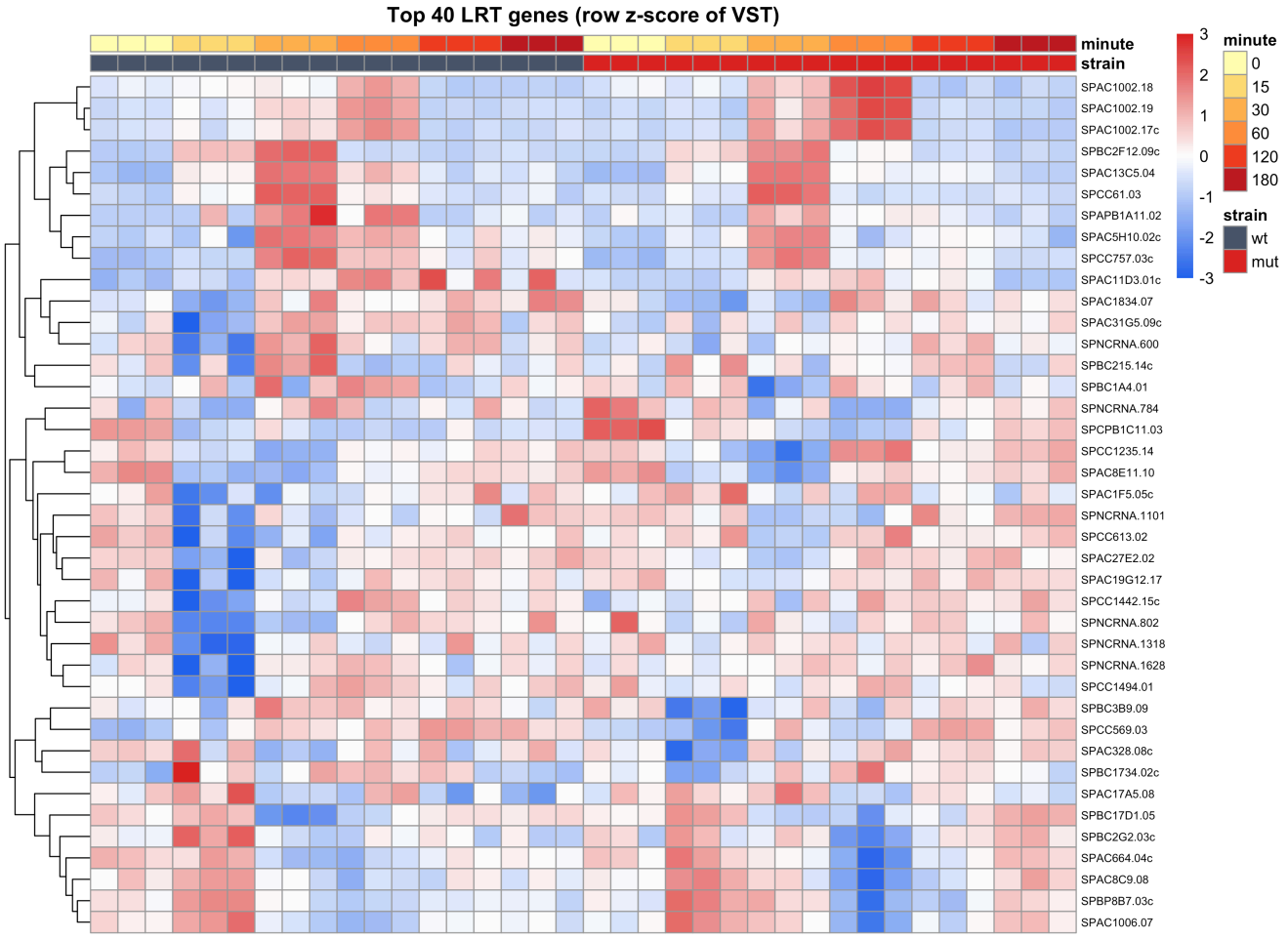


LRT 告诉你"哪些基因的响应依赖菌株"，但不告诉你**在哪个时间点差异最大**。要定位具体时间点，改用 Wald 检验提取单独一个交互项的效应：

```
dds_wald <- DESeq(dds)  # default test = "Wald"
# 查 resultsNames(dds_wald) 看有哪些系数
res30 <- results(dds_wald, name = "strainmut.minute30")
```

`strainmut.minute30` 这个系数回答的是：**在 30 分钟时，mut 和 wt 之间的差异，相对于时间 0 的基线差异，是否显著？**正值 = mut 在 30 分钟比 wt 升得更多；负值 = mut 升得更少或反而下降。

图 6：Top LRT 基因热图（按时间排序）



列按菌株 + 时间排序（不做列聚类），行是 top 40 LRT 基因的 row z-score。横向左到右能看到每个基因沿时间的变化；纵向看哪些基因变化模式相似。

其他常见的复杂设计

设计公式的写法对应生物学问题。一些速查：

问题	design	检验方式
两组比较	`~ condition`	`results(dds, contrast = c("condition","B","A"))`
控制批次	`~ batch + condition`	同上
时间效应	`~ minute` （因子）	LRT: reduced = `~ 1`
时间 × 菌株交互	`~ strain + minute + strain:minute`	LRT: reduced = `~ strain + minute`
连续剂量响应	`~ dose` （数值变量）	Wald on dose coefficient
多条件 + 批次	`~ batch + tissue + condition`	results(..., contrast = ...)

两个关键原则：

1. 要检验的变量**一般放 design 最后**，results() 默认取它

2. LRT 时 `reduced` 要和 `full` 只差你关心的那些项 —— 差得越多，LRT 就是在同时检验那些项的组合，解读会变乱

和 single-cell 的对比

单细胞里做"轨迹推断 + 沿拟时序的差异"（比如 Slingshot + tradeSeq），思路和 bulk 里 LRT 很像：都在问"基因表达是否沿着某个连续变量系统性变化"。主要差别：

- bulk 的 minute 是**实验里设好的时间点**，维度离散、样本多；单细胞的 pseudotime 是**推断出的**连续变量，每个细胞一个值
- bulk 的 LRT 基于 counts 的负二项分布；`tradeSeq` 用 GAM 拟合连续轨迹

常见坑

坑 1：把时间当数值变量

`design = ~ minute`（minute 是 numeric）会假设"基因表达沿时间线性变化"。但生物学响应通常不是线性的（先升后降、滞后峰值）。**默认把 minute 转成 factor**：`dds$minute <- factor(dds$minute, levels = c(0, 15, 30, 60, 120, 180))`，让模型自己估计每个时间点的水平。

只有在你确实想假设"剂量响应是单调线性"时才把 dose 留作 numeric。

坑 2：reduced 和 full 差太多导致 LRT 解读混乱

LRT 检验的是 "full 比 reduced 多出来那些项的整体显著性"。如果 full = `~ batch + strain + minute + strain:minute`、reduced = `~ 1`，LRT 同时检验 4 件事的综合显著性 — **结果显著也说不清是哪个驱动。reduced 只比 full 少你关心的那一项**，结果才好解读。

坑 3：LRT 之后用 Wald 时忘了重新跑 DESeq

LRT 模式下 `dds <- DESeq(dds, test = "LRT", reduced = ...)` 拟合的是 LRT 模型；要做 Wald 必须 `dds_wald <- DESeq(dds, test = "Wald")` 重新跑一遍，**不能直接 `results(dds, name = ...)` 取系数**，那会用 LRT 时估的 SE，结果不对。

坑 4：交互项基因数比预期少很多

时间×菌株交互的强信号在很多生物学体系里**就是稀疏的** — 大部分基因对时间的响应在两组里其实差不多，只是少数关键基因有差异。fission 这份数据里 LRT padj < 0.05 也只过几十个基因，是正常的，不是分析做错了。**先看 PCA 是否有明显的菌株分离**判断信号强度。

坑 5：用 vst 后的值做时间轨迹但忽略了 batch

`vst(dds, blind = FALSE)` 会用 design 信息去除 batch 影响；`blind = TRUE` 不会。**做时间轨迹可视化时务必 `blind = FALSE`**，否则 batch 噪声会把每条轨迹搞乱。

下载资源

bulk05_timecourse_sci.R
12 KB

下载 fission 时间序列 LRT 完整脚本 ↗

下一步

接着深入：

- 06 批次效应 — 时间序列实验经常跨多个测序批次，下一步是把 batch 也加进 design
- 07 多工具对比 — limma 的 `contrasts.fit` 在多因子设计下有时比 DESeq2 LRT 更灵活

横向延伸：

- 单细胞 05 轨迹推断 — bulk 的 LRT 与单细胞的 pseudotime 在思路上互通
- DESeq2 时间序列 vignette — 官方的更详细示例

参考资源

- DESeq2 vignette: time course
- fission 数据
- Leong et al. 2014 原始研究
- DESeq2 LRT vs Wald

07 批次效应：ComBat / SVA / RUV

批次效应是 RNA-seq 里最容易让人上当的误差来源之一：同一天测完的样本可能更像；同一个技师做的样本可能更像；同一个 lane 出来的样本可能更像。真实的生物学差异如果被这种技术变量遮住，得到的"差异基因"就全是噪声。

处理批次有三条主流路线：

方法	思路	典型场景
把批次放进设计公式	`design = ~ batch + condition`	批次已知且与目标条件不完全混淆
ComBat（sva 包）	直接修改表达矩阵，把批次效应"减掉"	批次已知；需要下游工具无法建模批次时
SVA（Surrogate Variable Analysis）	从表达矩阵里估计"看不到的"批次变量	批次未知，或还有其他技术变异
RUVSeq	用 control genes 估计不良变异	有 spike-in 或已知稳定基因时

优先级：能加进设计就加进设计（最干净）。设计不方便或者你不知道所有批次时再上 ComBat / SVA。

项目开始就该想的事：批次设计的预防

在数据已经测出来之后，再聪明的方法也只是补救。**真正的批次问题应该在实验设计阶段就避免**：

错误做法	后果	正确做法
第一周测对照、第二周测处理	batch 和 condition 完全混淆，无法分离	**每个 batch 都要含两组样本**
同一个人只测一组	操作人成混杂因子	操作随机化或每人测同样比例
试剂买错版本（v2 vs v3）	chemistry 差异是技术 batch	同一项目用同一批试剂
一个样本只跑一次测序	没法估技术重复噪声	至少几个样本做技术重复
上机日期是按组排的	上机批次就是组别	randomize 上机顺序

最重要的一条规则：实验设计阶段 condition 和 batch 必须 **balanced**（平衡）—— 每个 batch 里每个 condition 都有样本。混淆了就是混淆了，统计上 unfixable。

经典教学例子：bladderbatch

bladderbatch 是 sva 作者提供的 microarray 数据，正因为**批次和癌症状态重度混淆**，成了批次校正教学的标配：

```
        batch
cancer    1  2  3  4  5
  Biopsy  0  0  0  5  4
  Cancer 11 14  0  0 15
  Normal  0  4  4  0  0
```

注意 batch 1、2、3 几乎全是某一类样本 —— 这就是"混淆设计"（confounded design）。混淆严重到一定程度，再聪明的方法也救不回来。但 bladderbatch 里留了足够的重叠，能看出不同校正方法的效果。

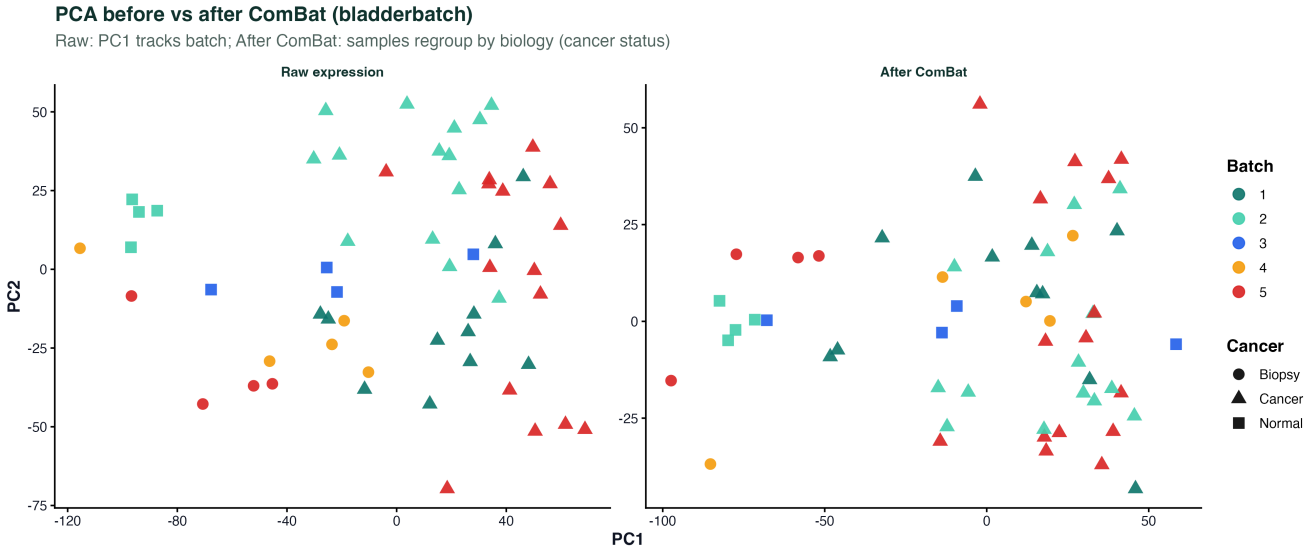
虽然是 microarray 数据，批次校正的原理和 RNA-seq 完全一样（数据经过 `voom` 或 VST 变换之后，两者本质上都是近似正态的连续数据矩阵），不妨碍教学。

真实示例：三种策略对比

配套脚本 `bulk06_batch_sci.R` 同时演示 bladderbatch 上的 ComBat / SVA / batch-in-design，以及 airway 上的"把 cell 当批次"。

```
Rscript scripts/bulk06_batch_sci.R
```

图 1：ComBat 前后的 PCA



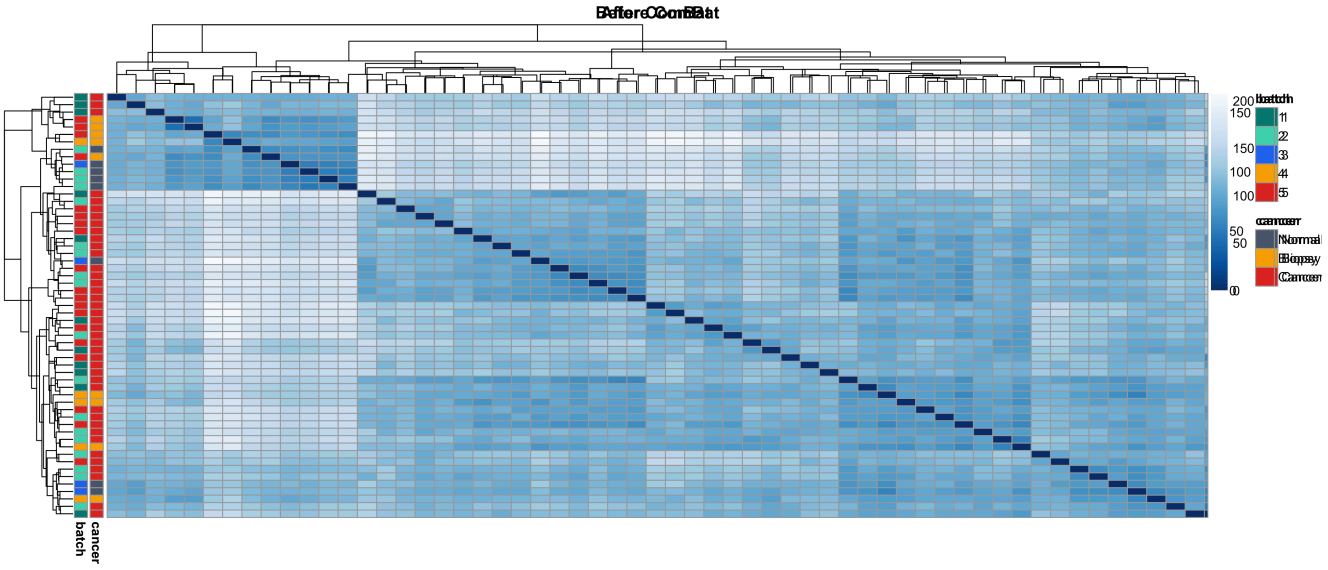
左边 raw：PC1 几乎被 batch 主宰 —— 同色的点（同批次）紧挨着，不同形状（癌症状态）散落在各处。右边 ComBat 之后：同形状的点（同癌症状态）开始靠拢，颜色（batch）反而变得弥散。

这张图把批次效应的本质直观化了：**原始矩阵里批次变异 > 生物学变异**。ComBat 做的就是把这个顺序掰回来。

```
library(sva)
mod <- model.matrix(~ as.factor(cancer), data = pheno)
combat_edata <- ComBat(dat = edata, batch = batch, mod = NULL)
```

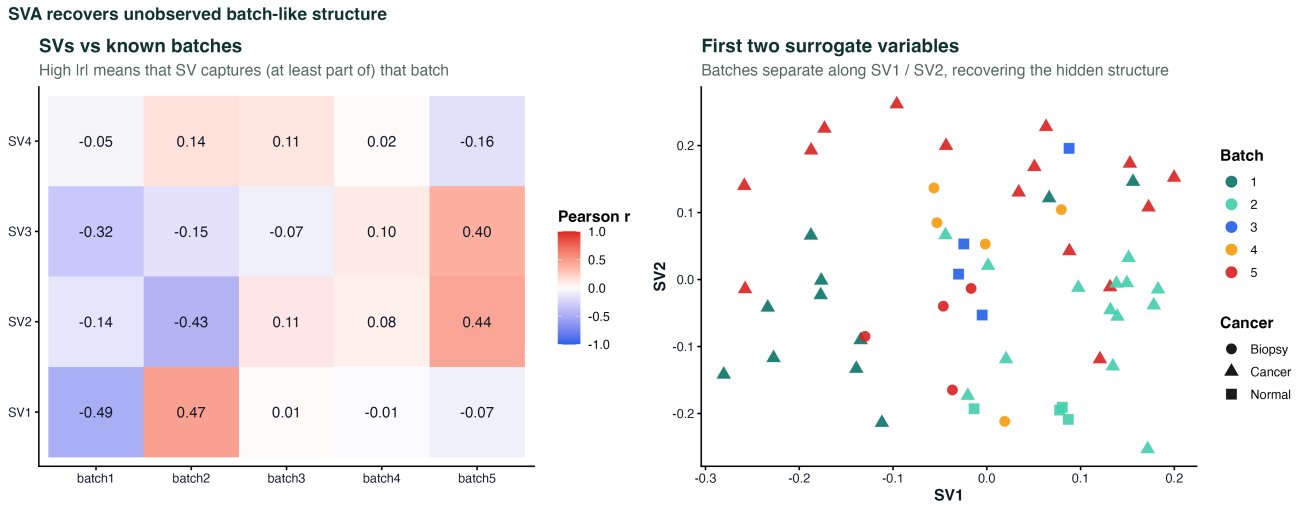
`mod` 这里传 NULL 表示"不保护任何生物学变量"，保守用法。如果确实担心 ComBat 把生物信号一起抹掉，可以传入 `mod = model.matrix(~ cancer, ...)`，告诉 ComBat "这列是要保留的"。

图 2：样本距离热图



跟 PCA 互补的角度：样本两两之间的欧式距离。校正前样本按 batch 聚在一起；校正后按 cancer 状态聚。写 Methods 段时把 PCA + 距离热图一起放，说服力会更强。

图 3：SVA 估计出的隐变量



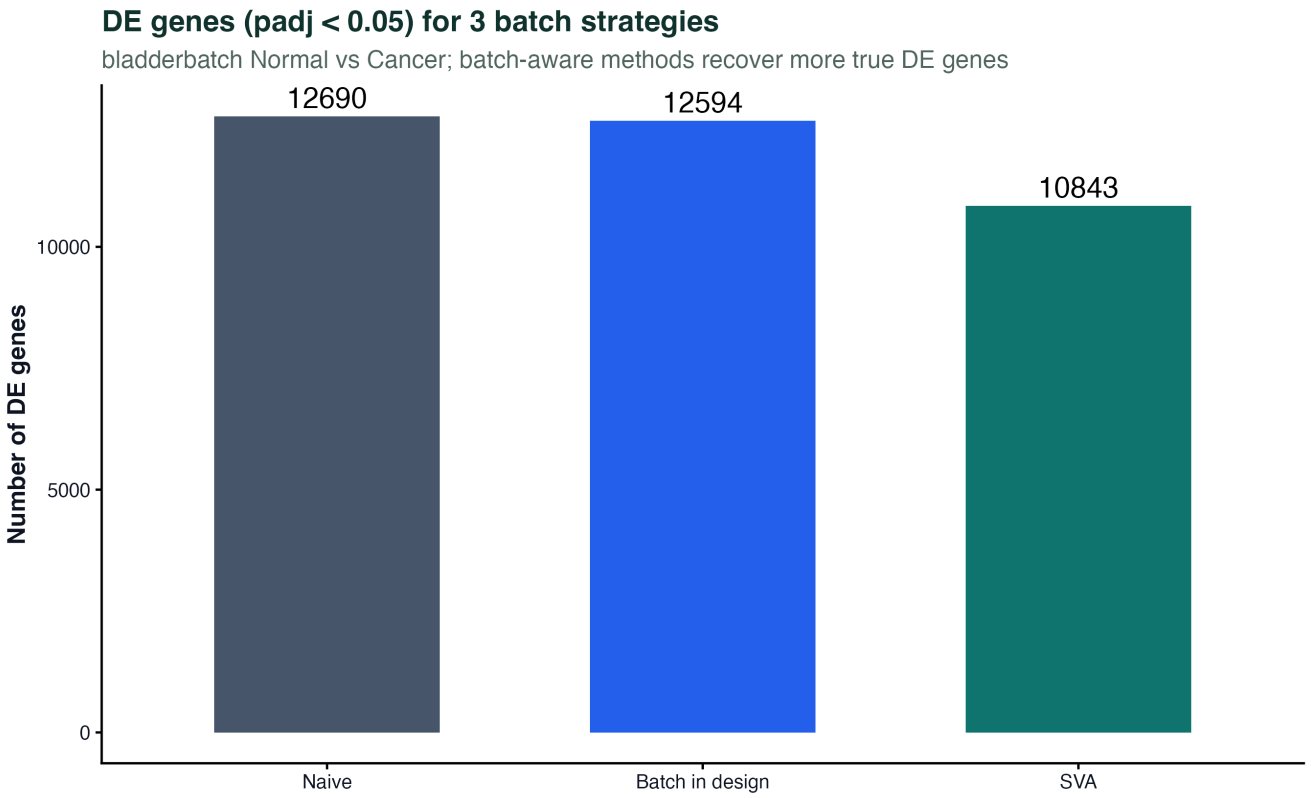
SVA 不依赖 batch 标签 —— 它直接从表达矩阵里估计"和生物学变量无关的系统性变异"，结果叫 **surrogate variable (SV)**。

左图是 SV 和已知 batch 的相关性矩阵。每个 SV 都和 **某一个 batch** 相关性很强 —— 意味着 SVA 成功把隐藏的批次结构找出来了。右图把 SV1 vs SV2 画成散点，点按真实批次着色：同批次的样本在 SVA 空间里也聚到一起。

```
mod0 <- model.matrix(~ 1, data = pheno)
n_sv <- num.sv(edata, mod, method = "leek")
svobj <- sva(edata, mod, mod0, n.sv = n_sv)
# 把 svobj$sv 当成新列加进 limma / DESeq2 的设计矩阵
```

SVA 什么时候用：完全不知道 batch、或者知道 batch 但怀疑还有别的隐变量（试剂批次、操作人、机器 drift）时。

图 4：三种策略的 DE 基因数

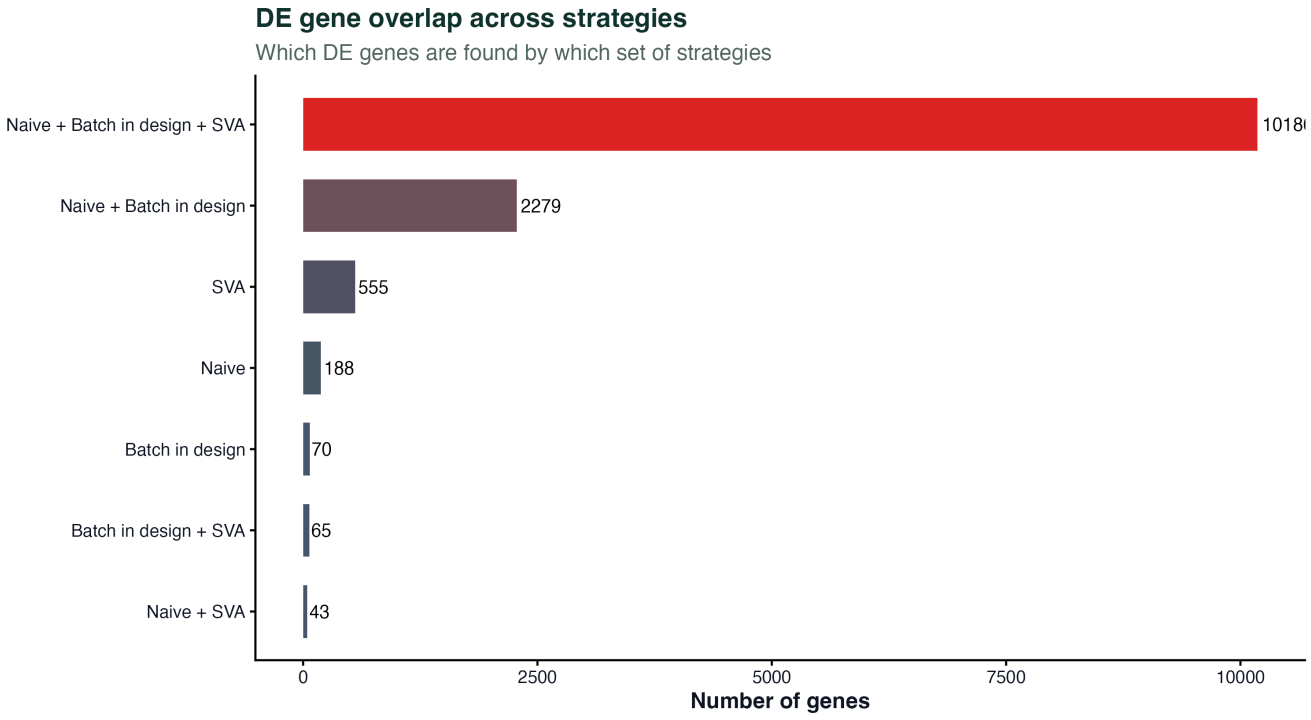


对 bladderbatch 里 Normal vs Cancer 的比较，三种策略（naive、把 batch 加进设计、用 SVA）分别跑 limma，统计 padj < 0.05 的差异基因数：

- Naive（忽略批次）：显著基因数被批次噪声稀释，本来该显著的信号被掩盖
- Batch 放进 design：效力显著提高
- SVA：进一步挖出额外的隐变量，效力最高

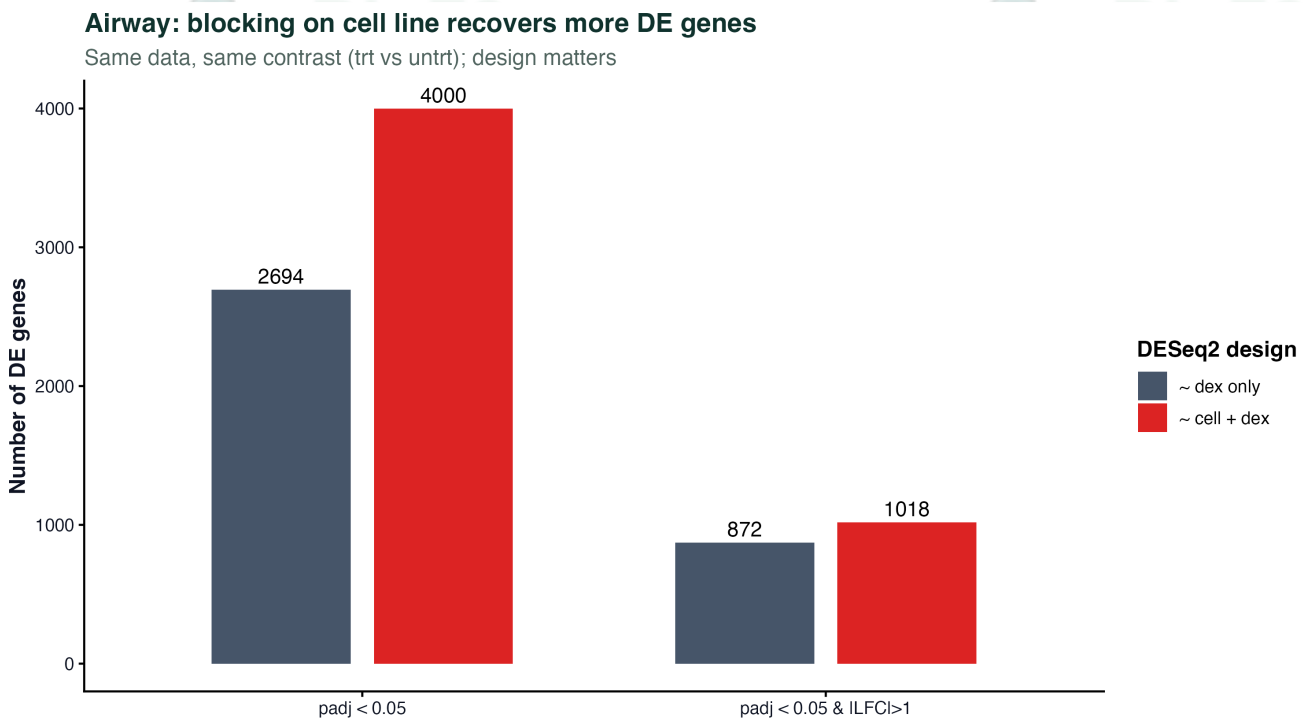
这张图最直接地说明批次校正**不是在"变出"基因**，而是在"让真正的生物学信号从噪声里浮出来"。

图 5：三种策略的 DE 基因重叠



DE 基因列表在三种策略之间的重叠。最重要的是检查：**naive 独有的那一部分基因，会不会其实是批次带来的假阳性？** 同时看**三种策略都发现的基因**是信号最稳的那批，适合重点关注。

图 6：airway 里把 cell 当批次



airway 有 4 个细胞系，每个细胞系都有 trt 和 untrt 两个样本。细胞系之间的基线表达差异是"技术批次"的一种 —— 只不过它是已知的。

- `design = ~ dex` （忽略 cell）：把细胞系间的变异算进残差，误差估得偏大，DE 基因变少
- `design = ~ cell + dex` （把 cell 放进去）：细胞系间的基线被模型吃掉，检验 dex 效应的功效更高

在 `padj < 0.05 & |LFC|>1` 这个实用阈值下，把 cell 加进 design 可以把 DE 基因数从几百个提到数千个。**这说明 DESeq2 里"batch 放不放进 design"不是个形式问题，是真的能改变结果的。**

如何选择策略

实用建议：

情形	推荐做法
batch 已知、未与条件混淆	`~ batch + condition`
batch 已知、部分混淆	同上 + 仔细检查 PCA / distance
batch 已知、完全混淆	**设计本身有问题**，尽量重测，其次 SVA
batch 未知	SVA，估出 SV 之后加进 design
有 spike-in 或 housekeeping	RUVg 或 RUVs
下游工具不接受 design（老版 GSVA 之类）	ComBat 直接校正矩阵

绝对不要做的事：在已经很干净的数据上"保险起见跑一遍 ComBat"。ComBat 会让样本间的真实生物学变异也变平，显著增加 type II error。

和单细胞整合的关系

单细胞实践 04 多样本整合 里的 Harmony / Seurat v5 integration 思路和这里一致：**在某个隐空间里把批次的那一部分方向抹掉，保留生物学方向**。只不过单细胞是在 PCA 空间上做，bulk 是在原始 log 表达矩阵上做。

常见坑

坑 1：批次和条件完全混淆还硬上 ComBat

如果你的 batch 1 全是 control、batch 2 全是 treatment，**ComBat / SVA / 任何方法都救不回来**。模型分不清"组间差异"和"批次差异"。这种情况只能重新设计实验。**用 `table(batch, condition)` 先看一眼分布**，如果某行某列全是 0，停止分析。

坑 2：ComBat 不传 mod 参数

`ComBat(dat, batch)` 不传 `mod` 时是"保守模式"，假定所有变异都是批次。**但如果生物学差异恰好和批次有点相关**（比如 batch 1 的样本恰巧大多是处理组），ComBat 会把生物学信号也当批次抹掉。**正确做法 `ComBat(dat, batch, mod = model.matrix(~ condition))`**，告诉它哪一列要保留。

坑 3：SVA 估出的 SV 数量过多

`num.sv(method = "leek")` 在小样本数据里有时给出 10+ 个 SV，全加进 design 会消耗自由度、降低统计力。**SV 数量经验上 ≤ samples/3**，超过这个先回头检查数据本身有没有别的问题。

坑 4：用 ComBat 之后还把 batch 加进 design

ComBat 已经把 batch 信号从矩阵里去掉了，DESeq2 / limma 的 design 不应再含 batch（含了会双重校正）。**ComBat → design = `~ condition`**；不用 ComBat → design = `~ batch + condition`。两选一，不能同时用。

坑 5：以为 batch 校正后样本之间所有差异都是生物学

校正完看 PCA 漂亮了 — 这只是说明"batch 这个变量被消了"，不代表"剩下的差异全是生物学"。可能还有别的混杂因子（性别、年龄、采样时间）。**校正后还是要逐个 metadata 列做 PCA 染色**，确认没有别的隐藏批次。

下载资源

bulk06_batch_sci.R
14 KB

下载批次效应演示完整脚本 ↗

下一步

接着深入：

- 07 多工具对比 — DESeq2 / edgeR / limma 在批次校正后的结果会有差别吗?
- 08 可复现分析报告 — Methods 段怎么写清楚 batch 处理方式

横向延伸：

- 单细胞 04 多样本整合 — 单细胞里 Harmony / Seurat integration 是同一思路在 PCA 空间的版本
- Leek 2010 batch effects 综述 — 批次效应统计学背景，是经典阅读

参考资源

- sva 包教程
- Leek et al. 2012 SVA 方法

- Johnson et al. 2007 ComBat 方法
- RUVSeq vignette
- bladderbatch 数据背景

08 多工具对比：DESeq2 / edgeR / limma-voom

bulk RNA-seq 差异分析有三个"标配"工具 —— DESeq2、edgeR、limma-voom。它们建模思路不同，但在绝大多数真实数据上结果高度一致。本章做一次 head-to-head 对比：同一份 airway 数据、同一条 design、同一个对比，跑三种工具，看结果是不是真的如传说那样一致。

三种工具的建模差异

工具	模型	特点
DESeq2	负二项 GLM + 共享离散度估计 + Wald / LRT	默认的归一化（median of ratios）稳；LFC shrinkage 是一大卖点
edgeR	负二项 GLM + 经验贝叶斯估计离散度	TMM 归一化，`glmQLFTest` 的 F 检验统计学上最稳
limma-voom	先 `voom` 把 counts 变成 log-CPM + 权重，再走 limma（线性模型 + 经验贝叶斯）	速度最快，适合大规模比较；对样本量大的数据尤其稳

几个经验：

- 样本量 < 10 对 10，差异不大，三者谁都能用
- 样本量 > 50，limma-voom 的速度优势很明显
- 要做复杂的 contrast（比如多因子设计），limma 的接口最灵活
- 要做单细胞数据的 pseudobulk DE，目前最主流的搭配是 **edgeR**

为什么 bulk RNA-seq 没有"最佳工具"

很多新手希望有一个"最准的"工具能一锤定音。这不存在，原因是：

1. 三种工具数学上接近等价

DESeq2 和 edgeR 都建模负二项分布，只是离散度估计的先验不同；limma-voom 走线性模型路线，但 voom 的权重设计让它在 RNA-seq 上几乎等价。**真实数据上 80%+ 的 DE 基因被三种工具都识别**。

2. 各自有各自的最佳场景

- DESeq2 的 LFC shrinkage 在做下游可视化和 GSEA 排序时最稳
- edgeR 的 QLF 检验在 FDR 控制上最严格
- limma-voom 在 100+ 样本规模下速度优势是数量级的

3. 真正影响结果的是上游 QC 和 design

`design = ~ condition` vs `design = ~ batch + condition` 带来的差异，远大于"DESeq2 vs edgeR"的差异。**先把 QC、批次、设计公式想清楚，工具选哪个其实不那么重要。**

实务建议：新项目用 DESeq2（社区最大、教程最多）；样本量 > 50 用 limma-voom；做单细胞 pseudobulk 用 edgeR。一线发表都接受。

三条流水线怎么写

DESeq2、edgeR、limma-voom 的 API 有些差别，但核心都是"声明 design + 拟合模型 + 提取 contrast"：

```r
# DESeq2
dds <- DESeqDataSetFromMatrix(cnt, colData, design = ~ cell + dex)
dds <- DESeq(dds)
res_deseq <- results(dds, contrast = c("dex", "trt", "untrt"))

# edgeR
dge <- DGEList(cnt) |> calcNormFactors(method = "TMM")
design <- model.matrix(~ cell + dex, data = coldata)
dge <- estimateDisp(dge, design)
fit_edg <- glmQLFit(dge, design)
qlf <- glmQLFTest(fit_edg, coef = "dextrt")
tt_edg <- topTags(qlf, n = Inf)$table

# limma-voom
v <- voom(dge, design, plot = FALSE)
fit_v <- lmFit(v, design) |> eBayes()
tt_lim <- topTable(fit_v, coef = "dextrt", number = Inf)
```

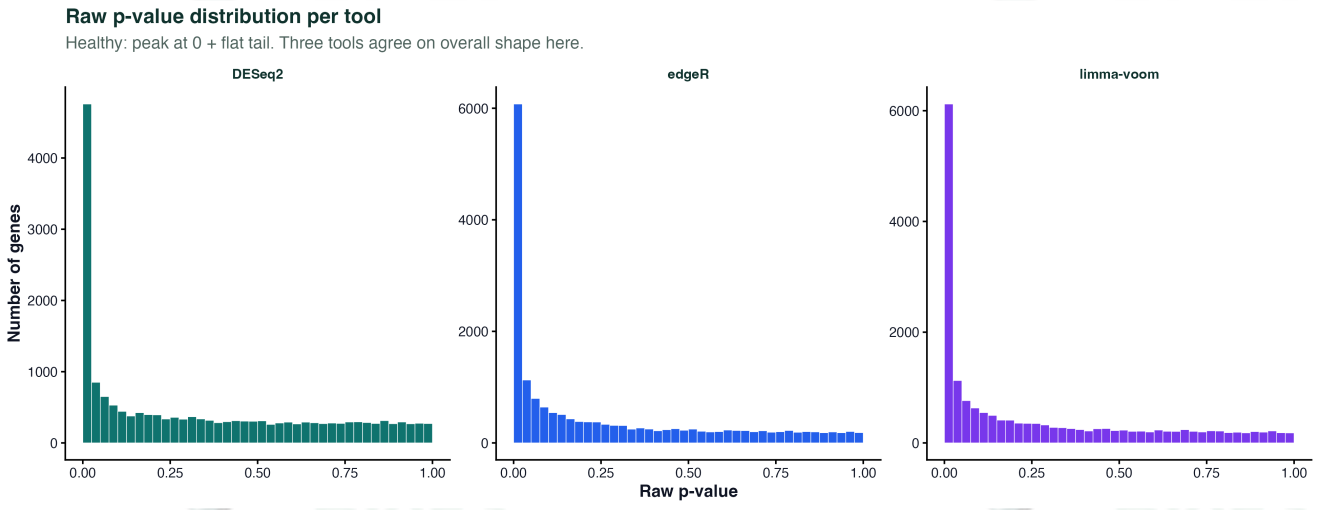
这三段都用同一份 `cnt` counts 矩阵和 `coldata` 样本表。关键在**让设计矩阵一致**：都是 `~ cell + dex`，都检验 `dex` 的 `trt vs untrt` 效应。

真实示例：airway 上的三工具对比

配套脚本 `bulk07_tools_sci.R` 在同一份 airway 数据上跑 DESeq2 / edgeR / limma-voom，做完后对比 6 个维度：

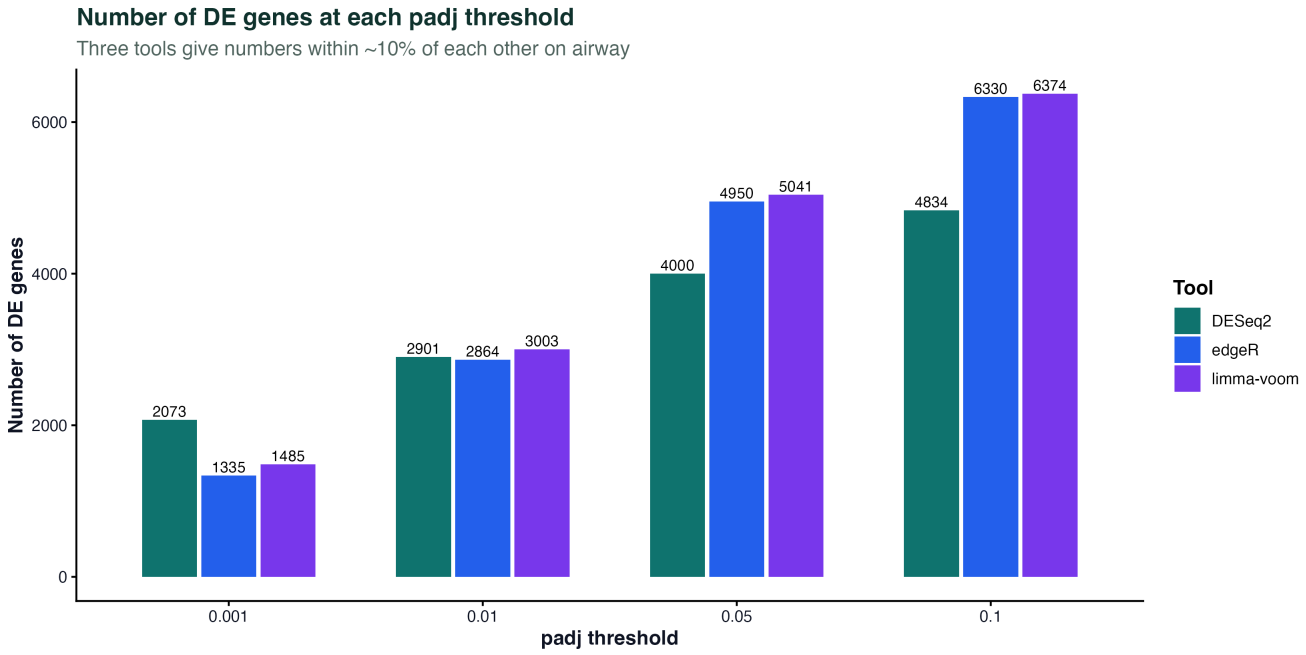
```
Rscript scripts/bulk07_tools_sci.R
```

图 1：每个工具的 p 值分布



三个工具各自的原始 p 值分布。健康的直方图都有"0 附近 peak + 剩余均匀"的形状。如果某个工具的直方图出现 U 型或者整体左偏，说明它在这份数据上模型拟合不好。airway 是个"好数据"，三个工具都给出了教科书级的 p 值分布。

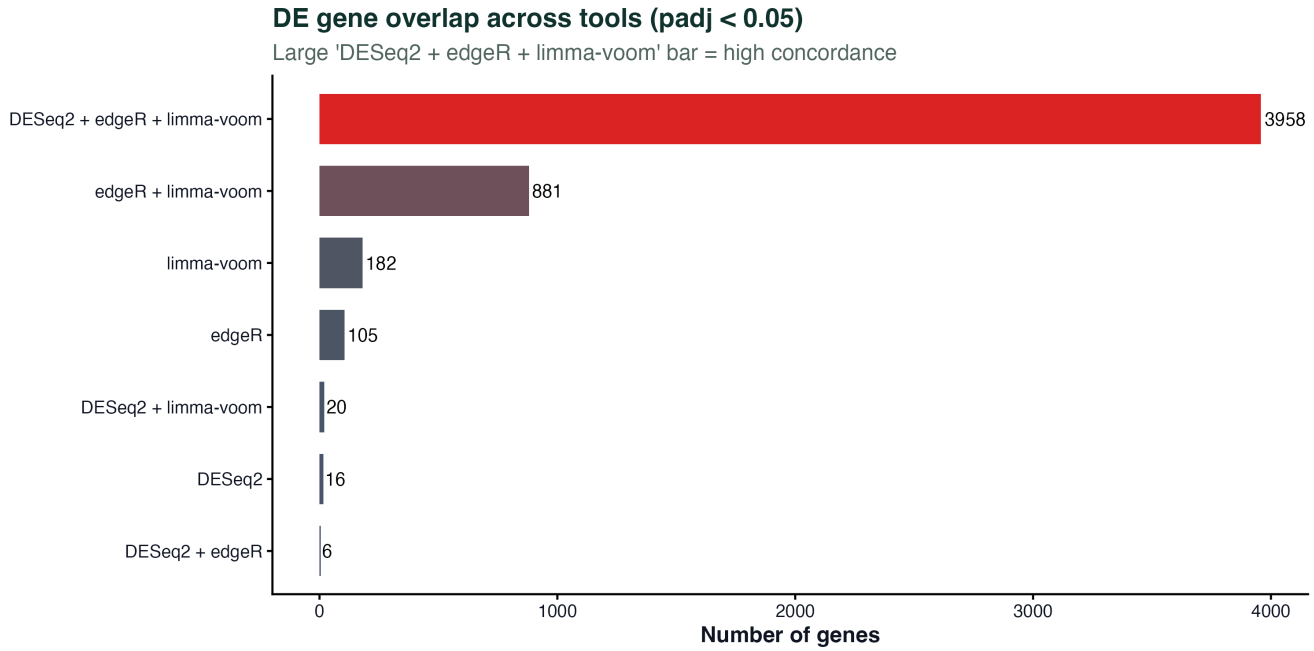
图 2：不同 padj 阈值下的 DE 基因数



同一个对比，三种工具在 padj < 0.001 / 0.01 / 0.05 / 0.1 四个阈值上的显著基因数。数值差距通常在 10% 以内 —— 这就是"三个工具大差不差"的具体含义。

值得注意的是 edgeR 的 QLF 检验（`glmQLFTest`）在统计学上是**比 DESeq2 的 Wald 更严格**的检验，所以 edgeR 的 DE 数经常比 DESeq2 略少。这不是 edgeR 少找到，而是它更严谨。

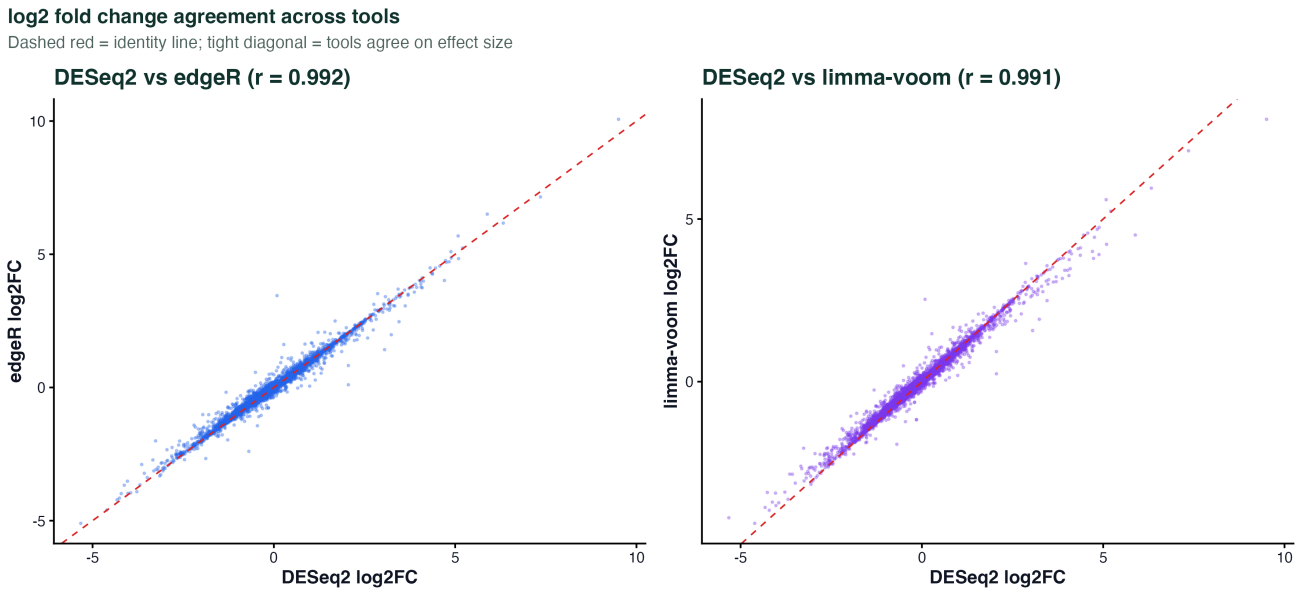
图 3：DE 基因集的重叠



三个工具的 DE 基因集（padj < 0.05）的重叠分布。最大的那一条"DESeq2 + edgeR + limma-voom"说明绝大多数 DE 基因被三个工具同时发现。只被某一个工具独有发现的基因数相对很少，通常是边界基因（接近阈值）。

实用经验：三个工具都说是 DE 的基因，大概率是真的信号；只被一个工具说是 DE 的，要多看一眼是不是 QC 问题或边界效应。

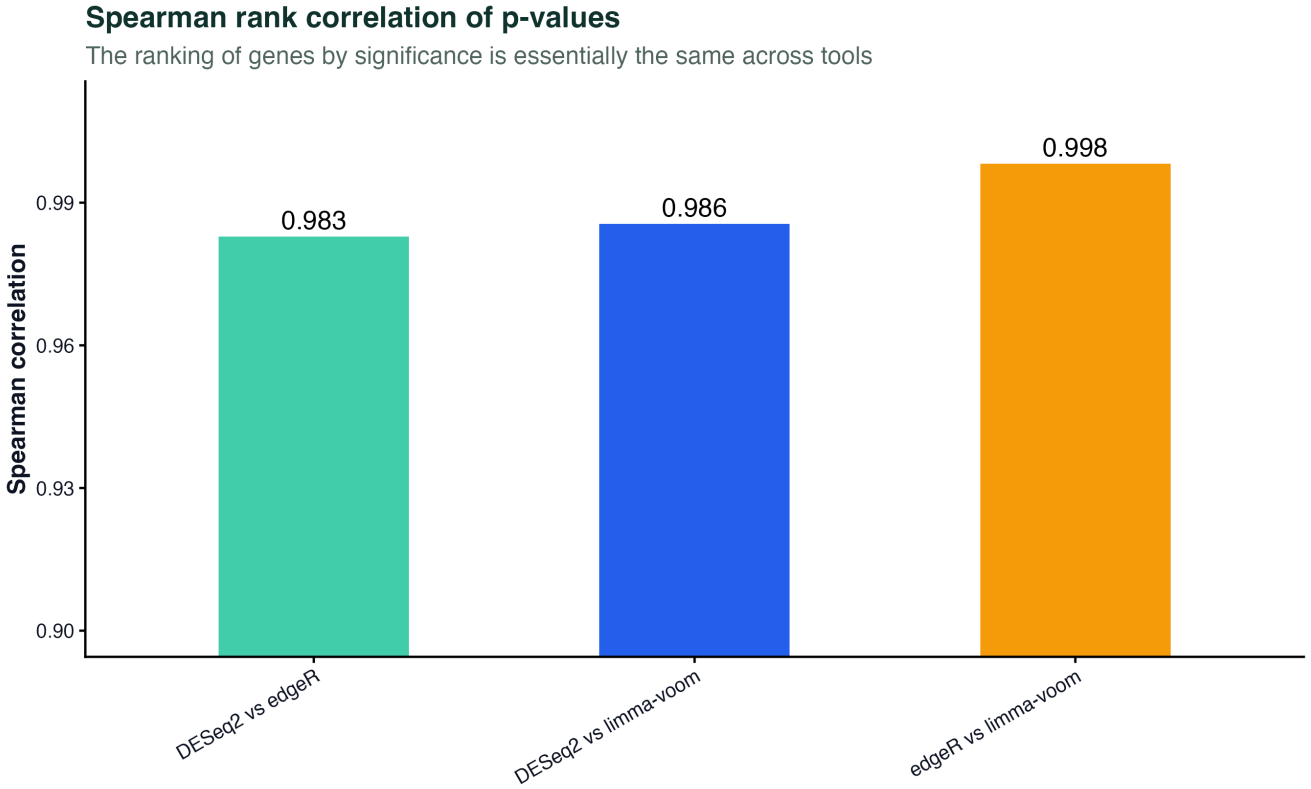
图 4：log2FC 的散点对比



三个工具估计的 log2 fold change 两两散点，红色虚线是 y = x 的对角线。点紧贴对角线 = 三个工具对效应大小的估计几乎一致。

DESeq2 vs edgeR 的相关系数通常在 0.99 左右；DESeq2 vs limma-voom 略低一点（limma-voom 的 LFC 在低表达区会稍微更平），但整体仍 > 0.98。

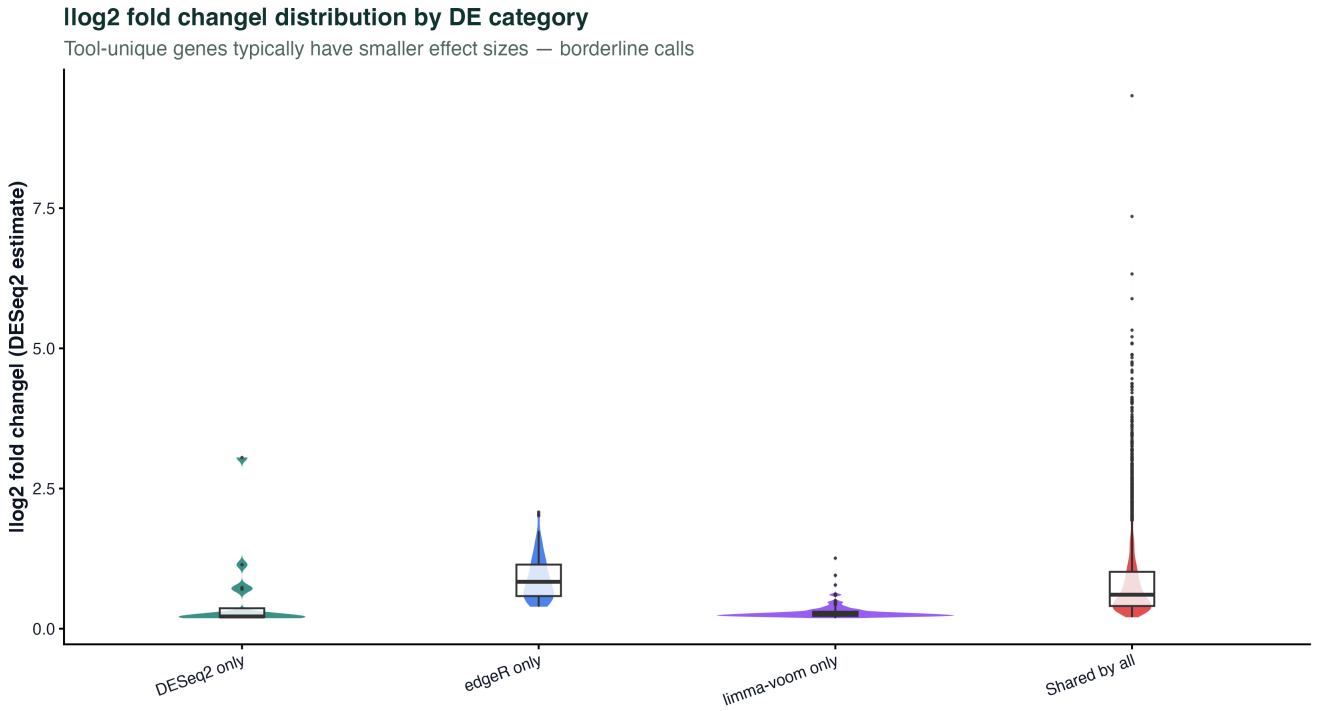
图 5：p 值的 Spearman 排序相关



三个工具两两之间，对所有基因按 p 值排序的 Spearman 相关。三个相关系数都 > 0.95。意思是：**如果你把"基因按显著性从高到低排名"当作输出（比如用来做 GSEA），三个工具给出的排名几乎完全一样。**

这个指标比"DE 基因集重叠"更稳 —— 因为后者会被阈值人为影响，而排序相关和阈值无关。

图 6：工具独有基因的效应大小



最后一个角度：三种工具独有的那些基因有什么共同点？把每类（A only、B only、C only、三个都有）的 `|log2FC|` 画成小提琴图。

- **Shared by all**（三个都有）：效应大，中位 |LFC| 可能在 1.5 以上 —— 这些是强信号基因
- **某一个工具独有**：效应小，大部分 |LFC| 在阈值附近 —— 边界基因

这张图最重要的解读：**工具间的分歧主要发生在"勉强显著"的基因上**。真正的强信号，三个工具都抓得到。

什么时候选哪个

实用建议（可以背下来）：

场景	推荐
样本量小（< 6 对 6）、新项目	**DESeq2**，默认最稳
样本量大（> 30 对 30）	**limma-voom**，速度最快
要做 ANOVA-like 检验（LRT / 多因子交互）	DESeq2 的 LRT 或 limma 的 `contrasts.fit`
单细胞 pseudobulk	**edgeR**，目前最成熟
要做严格的 FDR 控制（需要 FPR < 5%）	edgeR QLF，最保守
要做 LFC shrinkage 做下游	**DESeq2** 配 `apeglm`

都跑一遍是一个好习惯。在一份新数据上，先用三种工具各跑一遍，如果结果相似，报告其中一种；如果结果差很多，说明数据或设计有问题，要回头看 QC。

除了这三个

其他常见选择：

- **NOISeq**：非参数方法，对非常态分布友好

- **sleuth**：搭配 kallisto/Salmon 的 bootstrap 结果使用，能同时建模定量不确定性
- **BASiCS**：有 spike-in 的数据可用
- **DREAM**（variancePartition）：处理复杂混合效应模型（样本重复测量、配对设计）

刚上手时没必要学这些；先把 DESeq2 / edgeR / limma-voom 中一个摸熟。

常见坑

坑 1：design 公式不一致就直接对比

DESeq2 用 `~ cell + dex` 、edgeR 用 `~ dex` ，对比出来 DE 基因差很多 — 当然，**模型不一样**。三工具对比的前提是 design 矩阵完全一致，否则比的是 design 不是工具。

坑 2：不同工具的 LFC 直接对比

DESeq2 的 LFC 是 shrunken 版本，edgeR 的是 raw，limma-voom 是基于 log-CPM 的回归系数。**单位和尺度都不完全一样**，散点图能看出趋势但绝对值有偏差。要对比结果一致性，更稳的是**比较 p 值排序**（Spearman correlation）。

坑 3：拿"工具独有的基因"当差异点报道

DESeq2 独有的 50 个基因里，大概率是 padj 在 0.04-0.06 边缘的基因，换个工具阈值偏一点就过不了。**不要把工具差异当成"哪个工具更敏感"的证据**，那只是阈值附近的随机扰动。

坑 4：limma-voom 跑出 pvalue = 0

`eBayes()` 的 t 统计量极端时会输出 `pvalue = 0` （数值下溢）。 **`pvalue < 1e-300` 的基因要 cap 到 1e-300 再做下游**，否则 `-log10(0) = Inf` 让画图崩溃。

坑 5：edgeR 跑 LRT 但忘了 estimateGLMTagwiseDisp

旧版 edgeR 教程用 `estimateCommonDisp + estimateTagwiseDisp` ，新版要用 `estimateDisp` （一步到位）。**两套 API 不混用**，老教程的代码在新版可能跑不通或报错。

下载资源

bulk07_tools_sci.R
12 KB

下载三工具对比完整脚本 ↗

下一步

接着深入：

- 08 可复现分析报告与结果交付 — bulk RNA-seq 专栏的最后一篇，怎么把分析交付出去

横向延伸：

- 02 DESeq2 差异表达分析 — 如果某个工具的结果很奇怪，回头检查 DESeq2 的 QC 图
- Soneson 2018 RNA-seq 工具基准评测 — Nature Methods 的系统对比

参考资源

- DESeq2 vignette
- edgeR user's guide
- limma user's guide
- Soneson & Robinson 2018, benchmarking

- Schurch et al. 2016, how many replicates?

09 可复现分析报告与结果交付

分析做完不是终点，**能被别人（包括 6 个月后的自己）重新跑一遍**才是。这一章讨论 bulk RNA-seq 项目的收尾环节：怎么组织代码和输出、怎么写分析报告、怎么交付给合作者。没有什么"最佳实践"一定对，但有一些经验能让你后续工作顺得多。

为什么可复现性是真的硬要求

很多人把可复现当成"投稿前才需要做的事"。这是滞后思维。真正的成本在三个时间点：

1. **3 个月后**：审稿人问"figure 3 的数据怎么生成的"。能否一个命令重跑？不能 → 重跑要花 2-3 天。
2. **半年后**：合作者要在你的数据上加一个新分析。能否拿到一致环境？不能 → 装包失败、版本不匹配，光环境就调两天。
3. **一年后**：你自己写下一篇文章想引用这次结果。代码还跑得通吗？跑不通 → 引用了一份自己都不能复现的数据。

可复现不是为了别人，是为了未来的自己。投入 1 天做好，省下未来 5 天的 debug。

可复现的三个层次

从最基本到最严格：

层次	含义	实现方式
相同数据 + 相同代码 → 相同结果	基础要求	代码版本控制 + 随机种子
相同数据 + 相同环境 → 相同结果	跨机器复现	conda / renv / docker 固定依赖
相同实验 + 相同分析流程 → 相似结论	最高要求	清晰的样本 metadata + 分析日志

前两条是技术问题；第三条是组织问题。

目录结构回顾

01 章 里给过一份推荐的目录结构。在收尾阶段加一件事：**把最终的可交付产物单独放进 `deliverables/`**。

```
rnaseq_project/
├── README.md
├── metadata/samples.tsv
├── data/
│   ├── raw/
│   ├── qc/
│   └── counts/
├── analysis/
│   ├── 01-qc.R
│   ├── 02-de.R
│   ├── 03-enrichment.R
│   └── 04-figures.R
├── results/                # 中间结果
│   ├── tables/
│   └── figures/
├── deliverables/           # 交付给合作者的最终产物
│   ├── report.html
│   ├── report.pdf
│   ├── figures_final/      # 发表级图
│   └── tables_final/       # 带 README 的表格
└── envs/
    └── env.yaml / renv.lock
```

`results/` 是 "所有代码跑出来的东西"； `deliverables/` 是 "我要给别人看的那一小部分"。两者要分开，因为 `results/` 里通常有很多中间文件、测试用的图、探索性结果，不适合直接给合作者。

R Markdown / Quarto 做分析报告

bulk RNA-seq 最常见的交付形式是一份 HTML 报告，包含**样本表 + QC 图 + 差异基因结果 + 富集 + 讨论**。R Markdown（传统）或 Quarto（新）是目前最成熟的方案。

一个最小可用的 R Markdown 模板：

```
---
title: "RNA-seq analysis of dex-treated airway epithelium"
author: "Your Name"
date: "`r Sys.Date()`"
output:
  html_document:
    toc: true
    toc_float: true
    code_folding: show
    theme: flatly
---

# Background

(简要描述实验目的、样本来源、分析目标)

# Sample overview

```{r samples, message=FALSE}
library(DT)
samples <- read.delim("metadata/samples.tsv")
datatable(samples, rownames = FALSE)
```

QC

```
knitr::include_graphics("results/figures/01-library-size.png")
```

... PCA、距离热图等

Differential expression

```
library(DESeq2); library(dplyr); library(DT)
# ...
```

Enrichment

...

Session info

```
sessionInfo()
```
```

几个实用技巧：

- `code_folding: show`：读者能默认看到代码，但可以随时折叠
- `toc: true` + `toc_float: true`：左侧浮动目录，长报告必备
- `DT::datatable()` 嵌入交互式表格：合作者可以自己搜索和排序
- 最后一定要有 `sessionInfo()`：记录所有包版本

Quarto 是 R Markdown 的继任者，支持多语言（R、Python、Julia），渲染更快。生态还在迁移期，但新项目推荐直接用 Q

包版本管理

"我半年前跑过的代码，今天重跑结果变了"是一个很常见的困扰。R 这边最主流的两个方案：

renv

`renv` 会在项目里记录一份 lockfile，锁定所有包的版本：

```r
install.packages("renv")
renv::init()        # 初始化，生成 renv.lock
renv::snapshot()    # 任何包版本变了，重新 snapshot
renv::restore()     # 别人拿到项目，装回原来的所有包版本
```

对于发表前要归档的项目，强烈建议用 `renv` 。

conda env.yaml

如果项目里混了 bash 工具（Salmon、STAR、FastQC）和 R 包，用 conda 统一管理更方便：

```yaml
# envs/env.yaml
name: bulkrnaseq
channels:
  - conda-forge
  - bioconda
dependencies:
  - r-base=4.3.2
  - bioconductor-deseq2=1.42.0
  - bioconductor-edger=4.0.16
  - bioconductor-limma=3.58.1
  - bioconductor-clusterprofiler=4.10.0
  - salmon=1.10.2
  - fastqc=0.12.1
  - multiqc=1.21
  - r-tidyverse
```

用 `conda env create -f envs/env.yaml` 或者更快的 `mamba env create -f envs/env.yaml` ，就能在任何机器上装回一样的环境。

交付给合作者的结果表

给合作者的 CSV / Excel 表不能只是 `write.csv(res, ...)` 。至少要做三件事：

1. 加一份 README

每张表都配一个 `_README.txt` 或 `_README.md`，说明列的含义：

```
de_genes_airway.tsv
====================

8 个 airway 样本，trt (dex-treated) vs untrt (control)。
用 DESeq2 v1.48.2 做差异分析，design = ~ cell + dex。

列：
  gene_id          — Ensembl gene ID
  gene_symbol      — HGNC symbol（可能为空）
  baseMean         — DESeq2 归一化后的平均表达量
  log2FoldChange   — apeglm shrunken log2FC，正值 = 在 trt 里更高
  lfcSE            — LFC 的标准误
  pvalue           — raw p-value from Wald test
  padj             — Benjamini-Hochberg 校正后的 q-value

分析者：xxx
分析日期：2024-xx-xx
```

2. 加入 gene_symbol

矩阵里基因 ID 一般是 Ensembl，但合作者关心的是 SYMBOL。一定要把 SYMBOL 合并进去再交付：

```r
library(org.Hs.eg.db); library(AnnotationDbi)
id_map <- AnnotationDbi::select(
  org.Hs.eg.db,
  keys = rownames(res), keytype = "ENSEMBL",
  columns = c("SYMBOL")
)
res_out <- merge(as.data.frame(res), id_map, by.x = 0, by.y = "ENSEMBL")
```

3. 分层交付

别把所有 60000 行都给合作者。分成三张：

- `de_all_genes.tsv` — 全部基因，用于后续分析
- `de_significant.tsv` — padj < 0.05 的，通常几百到几千行
- `de_top_genes.xlsx` — top 50 padj + 显著通路里的关键基因，带格式化的 Excel

Methods 段怎么写

投稿时 Methods 段的 RNA-seq 部分通常要覆盖：

1. **样本来源**：几个对照 / 处理、生物学重复数、测序平台
2. **比对 / 定量工具和版本**：`STAR v2.7.11` 、 `Salmon v1.10.2` 等
3. **参考基因组版本**：`GRCh38, GENCODE v44`
4. **定量方式**：featureCounts 或 tximport from Salmon
5. **DE 工具 + 设计公式 + 阈值**：`DESeq2 v1.48.2, design = ~ batch + condition, padj < 0.05 & |LFC| > 1`
6. **富集工具 + 数据库版本**：`clusterProfiler v4.16.0, GO BP, MSigDB Hallmarks v2024.1`

两三段话就够。不要模糊处理（别写"我们做了标准的差异分析"），每个工具都要带版本号。审稿人常问的是"你用的 DESeq2 是哪个版本，padj 阈值是多少"。

常见坑

最后几条经验：

- **随机种子**：任何用到随机算法的地方（UMAP、聚类、抽样）都 `set.seed(42)`，不然每次跑结果都不一样
- **绝对路径**：别在代码里写 `/Users/你的名字/xxx`，用 `here::here()` 或 `file.path()` 相对路径
- **数据不进 git**：raw FASTQ / BAM / 大 counts 文件别 commit，用 `.gitignore` 屏蔽。如果要共享，用 Zenodo / figshare / GSA / SRA
- **日志**：长分析用 `sink()` 把 `sessionInfo()` 和关键输出存一份
- **测试**：`bulk07` 那个"三工具对比"本身就是一种测试 —— 新分析完成后用不同工具验证一遍，结果差太多时回头查

补充几条新人最常踩的：

坑：把数据 / 中间文件 commit 进 git

`git add .` 把几 GB 的 BAM / counts 矩阵也提交了，一次性把仓库吹大几倍，clone 半小时。**`.gitignore` 一定要早设**，包含 `data/raw/`、`data/align/`、`*.bam`、`*.h5ad`、`*.rds`（除非很小）。

坑：renv 锁了 R 包但没锁 R 本身的版本

R 4.2 和 R 4.3 的 base 函数行为微妙不同（factor 默认值、`dplyr` 兼容性等）。**README 显式记录 R 版本**：`R version 4.3.2 (2023-10-31) -- "Eye Holes"`。

坑：Methods 段写"按 DESeq2 默认参数"

读者不知道默认参数是什么版本的默认。**显式写出关键参数**：`design = ~ batch + condition`、`padj < 0.05`、`|log2FoldChange| > 1`、`apeglm shrinkage`。

坑：交付的 Excel 表里基因名被自动改成日期

Excel 是基因名杀手：`MARCH1` → `1-Mar`、`SEPT2` → `2-Sep`。**给合作者文件优先用 TSV / CSV**，必须 Excel 时把那一列设成"文本"格式。HUGO 已经把这些基因重命名了（MARCH1 → MARCHF1），但旧数据里的老名字还得用。

坑：报告 HTML 用了相对路径但发邮件后图片不显示

`include_graphics("results/figures/01.png")` 在本地正常，但发邮件单独一个 HTML 没有 `results/` 文件夹就空白。**给合作者发报告前用 `rmarkdown::render(..., self_contained = TRUE)`** 把图片 inline 进去，或干脆压缩整个目录发。

工具小清单

每个环节推荐的工具：

任务	推荐
报告	`R Markdown` → 过渡到 `Quarto`
表格交互	`DT::datatable`
环境	`renv`（纯 R）或 `conda`（混合工具）
项目模板	`usethis::create_project + here`
代码风格	`styler + lintr`
git 钩子	`precommit` 自动格式化

下载资源

本章不配脚本。给合作者的交付物相关模板（README / Methods 段 / R Markdown 骨架）可以从 BioF3 GitHub 仓库 找到（规划中）。

下一步

bulk RNA-seq 专栏 8 个模块到这里就完成了。后续可能扩展：

- 转录本级别分析（DTU、DTE）
- 拼接与可变剪接
- 小 RNA / miRNA / lncRNA 专题
- 单细胞 pseudobulk 与 bulk 的互补

也可以回到 单细胞实践教程 或其他组学专栏。

接着深入：

- 单细胞实践 03 QC + 聚类 — 学完 bulk 之后看单细胞 QC，会发现很多概念是相通的，但单细胞要面对的稀疏度大得多
- 多组学整合实践 — bulk 转录组配合甲基化、ATAC、蛋白组做整合分析

横向延伸：

- Quarto 官方文档 — R Markdown 的继任者，新项目推荐
- renv 教程 — R 包版本管理的标配

参考资源

- R Markdown: The Definitive Guide
- Quarto 官方文档
- renv 文档
- Bioconda
- RNA-seq workflow Bioconductor
- GEO / SRA 数据提交指南