BIOINFORMATICS

v1.0 · 2026.08

生物信息学入门教程

从分子生物学到 Linux 实战 · 面向零基础学习者

分子生物学 | 测序技术 | 数据格式 | 统计学 | 数据库 | Linux 实战

ATCG · 探索生命的数据之美

目 录

第二部分　Linux 基础入门

PART ONE

生信知识铺垫

第 1 章　分子生物学基础

1.1 中心法则：生命的"信息流水线"

分子生物学的**中心法则（Central Dogma）**描述了遗传信息在生物大分子间的流动方向：

图 1-1　中心法则示意图：*DNA* → *RNA* → 蛋白质（信息流不可逆）

用一个**工厂比喻**来理解：

阶段	生物学过程	工厂比喻	说明
DNA	储存遗传信息的"总蓝图"	档案室里的**原版设计图纸**	保存在细胞核中，不直接参与生产，是所有信息的最终来源
转录	DNA → mRNA	把图纸**复印**一份送到车间	原版图纸不能拿出来（太珍贵），所以转录一份 mRNA 副本带出细胞核
RNA	mRNA 携带遗传信息	**工作副本 / 工艺单**	mRNA 是 DNA 的"临时复印件"，可以从细胞核运到细胞质中被读取
翻译	mRNA → 蛋白质	车间按工艺单**组装产品**	核糖体读取 mRNA 上的密码子，将氨基酸逐个连接成蛋白质链
蛋白质	执行生命功能的"工人"	最终的**产品 / 零件**	蛋白质承担催化（酶）、结构（胶原蛋白）、运输（血红蛋白）等功能

> **一句话记忆**
>
> DNA 是图纸原件 → 转录出 mRNA 工作副本 → 翻译成蛋白质产品。**信息流始终是 DNA → RNA → 蛋白质**，不会反过来。

1.2 基因、转录本与基因组结构元件

1.2.1 基因（Gene）

基因是 DNA 上一段具有功能的序列，是遗传信息的基本单位。可以理解为一本"操作手册"中的一个完整章节——它包含了生产某种蛋白质（或功能性 RNA）所需的全部指令。

1.2.2 转录本（Transcript）

同一个基因通过**可变剪接（Alternative Splicing）**可以产生多种不同的 mRNA，这些不同的 mRNA 变体就叫**转录本**。

> **生活比喻**
>
> 同一个基因就像一本**菜谱**，可变剪接就像厨师根据不同场合选取不同步骤——有时加辣、有时不加——做出不同口味的菜（不同转录本），但都来自同一本菜谱。

1.2.3 外显子与内含子（Exon / Intron）

概念	定义	比喻	在转录中的命运
外显子（Exon）	基因中最终保留在成熟 mRNA 中的序列	菜谱中"真正有用的步骤"	被保留，参与翻译
内含子（Intron）	基因中在剪接过程中被去除的序列	菜谱中的"注释和说明"	在 RNA 剪接时被切掉丢弃

剪接过程：pre-mRNA（含内含子）→ 剪接去除内含子、拼接外显子 → 成熟 mRNA（只含外显子）。

1.2.4 启动子（Promoter）

启动子是基因上游的一段 DNA 序列，是 RNA 聚合酶的"停靠码头"。它像工厂的 **开工开关**——决定了基因何时、何地、以多大强度被转录。

1.2.5 UTR（非翻译区）

mRNA 上并非所有序列都会被翻译成蛋白质。成熟 mRNA 的两端各有一段不编码氨基酸的区域，称为 **UTR（Untranslated Region）**：

区域	位置	功能
5' UTR	mRNA 5' 端，翻译起始密码子上游	调控翻译效率、核糖体募集
3' UTR	mRNA 3' 端，终止密码子下游	mRNA 稳定性调控、microRNA 结合位点
CDS	起始密码子到终止密码子之间	实际编码蛋白质氨基酸序列的区域

图 1-2　基因结构与 *mRNA* 各区域示意图

1.3 密码子表与遗传密码

mRNA 上每 **3 个相邻的核苷酸**组成一个**密码子（Codon）**，对应一个氨基酸。4 种碱基（A/U/G/C）的组合产生 $4^3 = 64$ **种密码子**，编码 20 种标准氨基酸，加上起始和终止信号。

> **注意**
>
> 密码子表中的碱基是 **RNA 碱基**（A、U、G、C），不是 DNA 碱基（A、T、G、C）。在 DNA 中 T 对应 RNA 中的 U。

	U	C	A	G
U	**UUU** Phe **UUC** Phe **UUA** Leu **UUG** Leu	**UCU** Ser **UCC** Ser **UCA** Ser **UCG** Ser	UAU **Tyr** UAC **Tyr** UAA **Stop** UAG **Stop**	UGU **Cys** UGC **Cys** UGA **Stop** UGG **Trp**
C	**CUU** Leu **CUC** Leu **CUA** Leu **CUG** Leu	**CCU** Pro **CCC** Pro **CCA** Pro **CCG** Pro	**CAU** His **CAC** His **CAA** Gln **CAG** Gln	**CGU** Arg **CGC** Arg **CGA** Arg **CGG** Arg
A	AUU **Ile** AUC **Ile** AUA **Ile** AUG **Met**	**ACU** Thr **ACC** Thr **ACA** Thr **ACG** Thr	**AAU** Asn **AAC** Asn **AAA** Lys **AAG** Lys	**AGU** Ser **AGC** Ser **AGA** Arg **AGG** Arg
G	**GUU** Val **GUC** Val **GUA** Val **GUG** Val	**GCU** Ala **GCC** Ala **GCA** Ala **GCG** Ala	**GAU** Asp **GAC** Asp **GAA** Glu **GAG** Glu	**GGU** Gly **GGC** Gly **GGA** Gly **GGG** Gly

表 1-1　标准遗传密码子表（*RNA* 密码子）—— *Met** 为起始密码子，红色为终止密码子

> **关键记忆点**
>
> - **AUG** 既是起始密码子（编码甲硫氨酸 Met），也是翻译开始的信号
> - **UAA、UAG、UGA** 是终止密码子（Stop），不编码任何氨基酸
> - 遗传密码是**简并的**（degenerate）：多数氨基酸有多个密码子对应，如亮氨酸（Leu）有 6 个
> - 遗传密码是**通用的**（universal）：几乎所有生物共用同一套密码子表

1.4 序列比对：从"找相似"到"推断功能"

什么是序列比对？

序列比对（Sequence Alignment）是将两条或多条生物序列（DNA / RNA / 蛋白质）逐位置对齐，找出它们之间的对应关系和差异（替换、插入、缺失）的过程。

> **生活比喻**
>
> 就像**校对两篇文章的异同**：把两段文字逐字对齐，标出哪些字相同、哪些字被替换了、哪里多了一行或少了一行。序列比对做的是同样的事，只不过"字"变成了碱基（A/T/G/C）或氨基酸。

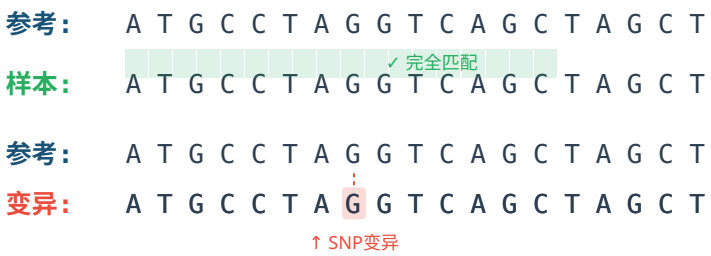

图 1-3　序列比对示意：上方为完全匹配，下方显示一个 *SNP* 变异位点

为什么需要比对？

应用场景	说明
功能推断	未知功能的序列与已知功能序列高度相似 → 大概率有类似功能
进化分析	比较不同物种的同源基因 → 推断亲缘关系和进化时间
变异检测	将测序 reads 比对到参考基因组 → 找出 SNP、Indel 等变异
基因注释	通过比对已知基因模型来注释新组装的基因组
差异表达分析	RNA-seq 的第一步就是把 reads 比对到基因组/转录组上

比对的基本类型

- **全局比对（Global Alignment）**：两条序列从头到尾全部对齐（如 Needleman-Wunsch 算法），适合长度相近的序列
- **局部比对（Local Alignment）**：只比对最相似的区域（如 BLAST、Smith-Waterman 算法），适合找保守片段
- **多重序列比对（MSA）**：同时比对三条以上序列，用于找保守位点、构建系统发育树

生信中最常用的比对工具

- **BLAST**：序列相似性搜索的"瑞士军刀"，快速从数据库中找相似序列
- **BWA / Bowtie2**：将大量短 reads 快速比对到参考基因组（DNA-seq）
- **STAR / HISAT2**：RNA-seq 专用比对工具，能处理剪接位点

—— 第 1 章结束 ——

第 2 章　测序技术简介

2.1 一代 / 二代 / 三代测序原理对比

测序（Sequencing）就是"读取 DNA/RNA 分子上的碱基排列顺序"的技术。自 1977 年 Sanger 测序法诞生以来，测序技术已经历了三代更迭。

特征	一代测序（Sanger）	二代测序（NGS / Illumina）	三代测序（PacBio / Nanopore）
原理	链终止法：用荧光标记的 ddNTP 随机终止合成，通过电泳/毛细管读取荧光信号	边合成边测序（SBS）：DNA 聚合酶逐个添加荧光标记 dNTP，高分辨率摄像头实时拍照读信号	单分子实时测序：无需 PCR 扩增，直接读取单条 DNA 分子
读长	~800–1,000 bp	50–300 bp（短读长）	10 kb – >100 kb（长读长）
通量	低（一次几十–几百条）	极高（一次数十亿条 reads）	高（一次数百万条 reads）
准确率	极高（~99.99%）	高（~99.9%，Q30+）	中–高（PacBio HiFi ~99.9%；Nanopore ~95–99%）
成本	高（$500/Mb）	极低（$0.01/Mb）	中（$1–10/Mb）
典型应用	验证测序、小片段测序、Sanger 验证金标准	RNA-seq、全基因组重测序、外显子组、ChIP-seq	基因组从头组装、结构变异检测、全长转录组
代表平台	ABI 3730xl	Illumina NovaSeq / NextSeq	PacBio Sequel IIe、Oxford Nanopore MinION/ PromethION

一句话总结三代演进

一代 准但慢且贵 → 二代 快且便宜但读长短 → 三代 读长长但准确率略低且成本中等 。实际项目中常组合使用，比如用二代做高精度检测、三代补结构变异和组装。

2.2 RNA-seq 实验流程

RNA-seq（RNA 测序）是研究基因表达量最主流的技术。完整流程如下：

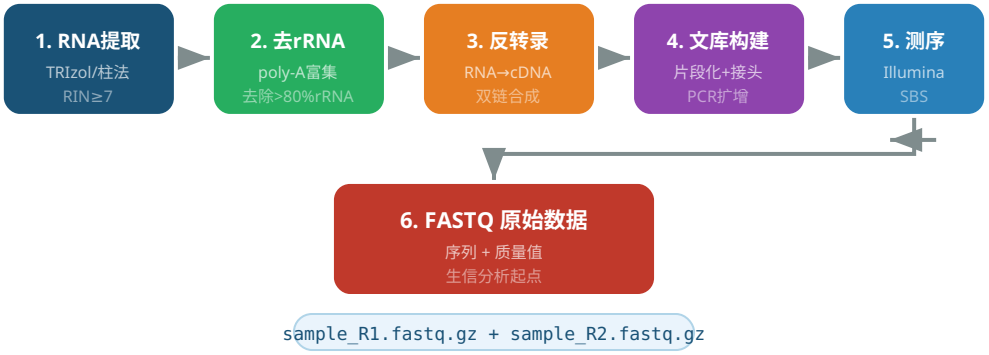

图 2-1　*RNA-seq* 实验流程：从 *RNA* 提取到 *FASTQ* 数据产出

各步骤详解

步骤	名称	做什么 & 为什么要做
1	RNA 提取	从细胞/组织中提取总 RNA。常用 TRIzol 试剂或柱式试剂盒。需检测 RNA 完整性（RIN 值 ⩾ 7 为合格）
2	去除 rRNA / 富集 mRNA	总 RNA 中 >80% 是核糖体 RNA（rRNA），不关心它。用 poly-A 磁珠富集带 poly-A 尾的 mRNA，或用探针去除 rRNA
3	反转录（RT）	测序仪只能测 DNA，所以用反转录酶把 RNA 转成互补 DNA（cDNA）。通常做双链 cDNA 合成
4	文库构建	将 cDNA 打断成短片段 → 末端修复 → 加 A 尾 → 连接接头（adapter）→ PCR 扩增富集。得到的"测序文库"可以直接上机
5	上机测序	将文库加载到测序芯片（flow cell）上，Illumina 平台通过 SBS（边合成边测序）读取每条片段的碱基序列
6	FASTQ 产出	测序仪输出 FASTQ 格式文件，包含每条 read 的序列和每个碱基的质量值。这就是生信分析的起点

2.3 读长、覆盖度与转录本定量

读长（Read Length）

每条测序读取的碱基长度。Illumina 常见读长有 PE150（双端各 150 bp）、SE50（单端 50 bp）等。**读长越长，比对越特异，能覆盖的变异类型越多。**

覆盖度（Coverage / Depth）

基因组上每个碱基平均被多少条 read 覆盖。

```
覆盖度 = 总测序碱基数 / 基因组大小

例如：人类基因组 ~3 Gb，测序得到 90 Gb 数据
覆盖度 = 90 Gb / 3 Gb = 30×
```

RNA-seq 通常不需要太高的覆盖度，10–30× 即可；全基因组测序一般需要 30× 以上。

转录本定量（Transcript Quantification）

计算每个基因/转录本表达了多少。核心思想：**一个基因被测到的 read 数越多，说明它表达量越高。**

定量方法	原理	常用工具
基于比对	先把 reads 比对到基因组，再统计每个基因区域的 read 数	STAR + featureCounts、HTSeq
基于伪比对	不传统比对，快速判断 read 属于哪个转录本，速度极快	Kallisto、Salmon、RSEM

2.4 FASTQ 与 FASTA 格式详解

2.4.1 FASTA 格式

FASTA 是最简单的序列存储格式，只包含序列信息，用于存储参考基因组、蛋白质序列等。

```
>基因标识行（以 > 开头，后面跟序列 ID 和描述）
ATCGATCGATCGATCGATCGATCGATCGATCGATCG
ATCGATCGATCGATCGATCGATCGATCGATCGATCG
（序列可以换行，每行通常 60 或 70 个字符）

# 实际示例：人类 BRCA1 基因片段
>ENST00000357654|BRCA1|Homo sapiens
ATGGATTTATCTGCTCTTCGCGTTGAAGAAGTACAAAATGTCAT
TAATGCTATGCAGAAAATCTTAGAGTGTCCCATCTGTCCAGATA
```

2.4.2 FASTQ 格式

FASTQ 在 FASTA 基础上增加了每个碱基的**质量值**，是测序仪的原始输出格式。每条 read 占 **4 行**：

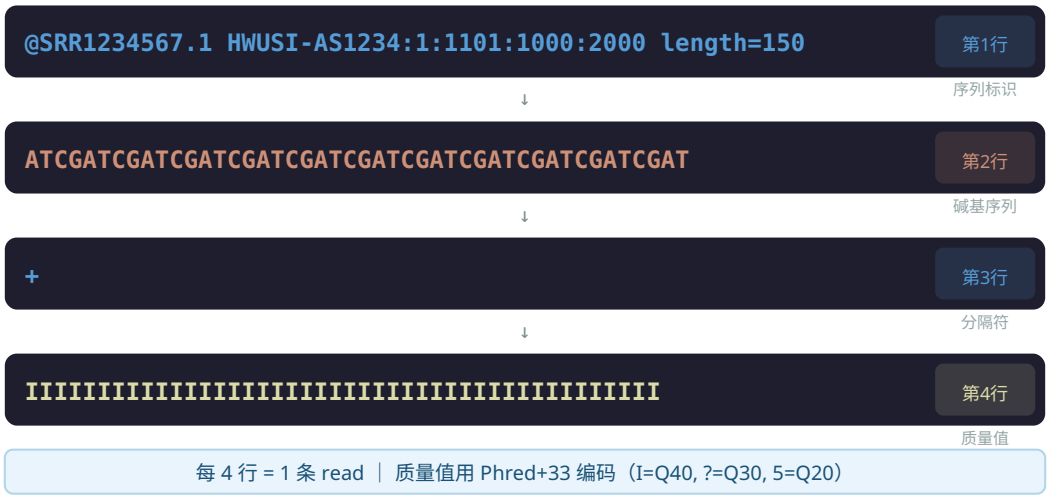

图 2-2　*FASTQ 格式图解：4 行构成一条完整的测序 read*

行号	字段	说明
第 1 行	序列标识行（@ 开头）	以 `@` 开头，包含 read ID、测序仪信息、坐标等元数据
第 2 行	碱基序列	测到的 DNA 序列，由 A/T/G/C/N 组成（N 表示无法确定）
第 3 行	分隔符（+ 开头）	仅一个 `+`，有时后面会重复 read ID（可省略）
第 4 行	质量值字符串	每个字符对应第 2 行的一个碱基的质量分值，使用 **Phred +33** 编码

质量值（Phred Score）详解

第 4 行的每个 ASCII 字符代表对应碱基的质量分值。Illumina 使用 **Phred+33** 编码：

```
Phred 值 = ASCII 码 − 33

例如字符 'I' 的 ASCII 码是 73，Phred 值 = 73 − 33 = 40
表示该碱基的错误率 = 10^(−40/10) = 10^(−4) = 0.01%
```

Phred 值	ASCII 字符	错误率	准确率	质量评级
Q10	'+'	1/10	90%	差
Q20	'5'	1/100	99%	合格
Q30	'?'	1/1000	99.9%	良好
Q40	'I'	1/10000	99.99%	优秀

质控标准

RNA-seq 数据通常要求 ≥ **80% 的碱基达到 Q30**。如果质量值偏低，需要用 `fastp` 或 `Trimmomatic` 进行过滤和裁剪。

—— 第 2 章结束 ——

第 3 章　核心数据类型

3.1 序列数据（DNA / RNA / 蛋白质）

生信分析的核心对象是**序列**。三种序列的存储和编码各有特点：

序列类型	字母表	格式	来源
DNA	A / T / G / C / N	FASTA	基因组测序、PCR 扩增
RNA	A / U / G / C / N	FASTA	RNA-seq（通常转为 cDNA 测序，故文件中仍为 T）
蛋白质	20 种氨基酸单字母代码	FASTA	基因翻译、质谱鉴定

3.2 基因组坐标与注释格式（GTF / GFF / BED）

序列数据回答"是什么序列"，注释格式回答"**序列上的某段区域是什么功能**"。

3.2.1 GTF 格式（Gene Transfer Format）

GTF 是最常用的基因注释格式，每行描述一个特征（feature）。共 **9 列（Tab 分隔）**：

```
chr1  havana  exon  11869  12227  .  +  .  gene_id "ENSG00000000005"; transcript_id
"ENST00000000412";
 |      |       |      |       |    |  |  |       |
 |      |       |      |       |    |  |  |       └── 第9列：属性键值对（gene_id,
transcript_id 等）
 |      |       |      |       |    |  |  └── 第8列：帧（0/1/2，CDS 的阅读框）
 |      |       |      |       |    |  └── 第7列：链（+/-/.）
 |      |       |      |       |    └── 第6列：得分（通常用 . 占位）
 |      |       |      |       └── 第5列：结束坐标（1-based，闭区间）
 |      |       |      └── 第4列：起始坐标（1-based，闭区间）
 |      |       └── 第3列：特征类型（gene / transcript / exon / CDS / UTR 等）
 |      └── 第2列：来源（havana / ensembl / . ）
 └── 第1列：染色体/contig 名称
```

3.2.2 GFF 格式（Generic Feature Format）

GFF 与 GTF 结构几乎一样（也是 9 列），区别在于第 9 列的属性格式不同：

```
# GTF 属性格式（键="值"，分号分隔，空格分隔键值）
gene_id "ENSG00000000005"; transcript_id "ENST00000000412";

# GFF3 属性格式（键=值，分号分隔，无空格无引号）
ID=ENST00000000412;Parent=ENSG00000000005;Name=TSPAN6
```

3.2.3 BED 格式（Browser Extensible Data）

BED 是更简洁的区间格式，最少 **3 列**，最多 12 列：

```
# 前 3 列（必须）：
chr1    11868    12227    TSPAN6    .    +

# 各列含义：
# 1. chrom：染色体名
# 2. chromStart：起始坐标（0-based，半开区间！）
# 3. chromEnd：结束坐标
# 4. name（可选）：特征名称
# 5. score（可选）：得分
# 6. strand（可选）：链（+/-.）
```

关键差异：坐标系！

- **GTF/GFF**：1-based，闭区间 → `start=100, end=200` 表示第 100 到第 200 号碱基（共 101 个碱基）
- **BED**：0-based，半开区间 → `start=99, end=200` 表示第 100 到第 200 号碱基（共 101 个碱基）
- 这是生信中最常见的**"差一个"bug**来源，务必牢记！

3.3 表达量数据（计数矩阵、TPM / FPKM / RPKM）

RNA-seq 定量后得到的是每个基因/转录本的**表达量**。有几种常见的标准化方式：

3.3.1 原始计数矩阵（Count Matrix）

最原始的表达量，是每个基因被测到的 read 数。行是基因，列是样本，值是整数。

```
            Sample_1  Sample_2  Sample_3
BRCA1           1254      2103       876
TP53             892       745      1203
EGFR             456       623       312
GAPDH          15234     14892     16001
```

3.3.2 RPKM / FPKM

指标	全称	计算公式	适用场景
RPKM	Reads Per Kilobase Million	RPKM = read_count / (gene_length_kb × total_reads_million)	单端测序 RNA-seq
FPKM	Fragments Per Kilobase Million	与 RPKM 相同，但"片段"而非"read"（双端测序一对 = 1 个 fragment）	双端测序 RNA-seq

核心思想：**同时校正基因长度和测序深度**——长基因天然更容易被测到，测序量大的样本 read 数自然多。

3.3.3 TPM（Transcripts Per Million）

TPM 是目前推荐使用的标准化方式，与 RPKM/FPKM 的区别在于**先校正长度再校正深度**：

```
TPM = (read_count / gene_length_kb) / Σ(read_count / gene_length_kb) × 10^6

# TPM 的优势：所有样本的 TPM 总和相同（= 1,000,000），便于跨样本比较
```

选择建议

- **差异表达分析**（DESeq2 / edgeR）：直接用原始 count 矩阵，软件内部会做自己的标准化
- **跨样本比较表达量**：用 TPM（推荐）或 FPKM
- **不要**用 FPKM/RPKM 做差异表达分析

3.4 差异表达结果表（log2FC、P 值、FDR）

差异表达分析（Differential Expression Analysis）的输出通常是一张表，每行一个基因，包含以下核心列：

列名	含义	解读方式
gene_id	基因 ID	基因标识符
baseMean	所有样本中的平均表达量	过滤低表达基因用
log2FoldChange	表达量变化的 log2 倍数	正值 = 上调；负值 = 下调；绝对值越大变化越大
pvalue	原始 P 值	衡量该基因差异是否显著，但不考虑多重检验
padj (FDR)	校正后的 P 值	经过 BH 校正，**这是判断显著性的标准**

log2FoldChange 解读

```
log2FC = log2(处理组表达量 / 对照组表达量)

log2FC = +1  →  处理组是对照组的 2^1 = 2 倍  →  上调 2 倍
log2FC = +2  →  处理组是对照组的 2^2 = 4 倍  →  上调 4 倍
log2FC = -1  →  处理组是对照组的 2^(-1) = 0.5 倍  →  下调 2 倍
log2FC =  0  →  两组表达量相同  →  无变化
```

差异基因筛选标准（常见阈值）

- **|log2FC| ≥ 1**（变化至少 2 倍）
- **padj < 0.05**（FDR 校正后仍显著）
- 同时满足以上两个条件的基因，通常被定义为差异表达基因（DEG）

—— 第 3 章结束 ——

第 4 章　统计学基础

4.1 P 值："证据够不够强"

P 值 是生信统计分析中最核心的概念之一。它回答的问题是：**"如果原假设成立（即两组没有差异），观察到当前数据或更极端情况的概率有多大？"**

生活比喻

你怀疑一枚硬币被做了手脚（正面朝上概率 ≠ 50%）。抛了 10 次，9 次正面。

原假设：硬币是公平的。

P 值 = "如果硬币公平，出现 9 次或更多正面的概率" ≈ 1.1%。

P 值很小 → 不像巧合 → **有理由拒绝原假设，认为硬币不公平**。

P 值范围	含义	结论
P < 0.05	如果没差异，出现这种结果概率 < 5%	通常认为**显著**（但需校正）
P < 0.01	概率 < 1%	更**显著**
P ≥ 0.05	概率 ⩾ 5%，不能排除巧合	**不显著**，不能说有差异

P 值的常见误解

- P 值 **不是** "原假设为真的概率"
- P 值 **不是** "差异的大小"（差异大小看效应量，如 log2FC）
- P 值 **只是** "如果没差异，数据有多极端"的概率
- P < 0.05 不等于"有差异"，只是"有理由怀疑有差异"

4.2 多重假设检验与 FDR 校正

问题：为什么要校正？

RNA-seq 差异表达分析通常同时检验**上万个基因**。如果每个基因用 P < 0.05 的标准，即使所有基因都没有真实差异，纯靠运气也会有约 5% 被误判为显著。

```
# 假设检验 20,000 个基因，真实差异为 0
# 每个基因 P < 0.05 的误报概率 = 5%
# 期望误报基因数 = 20,000 × 0.05 = 1,000 个!

# 也就是说，不做校正的话你会得到 ~1,000 个"假阳性"差异基因
```

FDR（False Discovery Rate）校正

FDR（错误发现率）控制的是：在被判定为显著的结果中，假阳性的比例。

最常用的方法是 **Benjamini-Hochberg（BH）校正**，校正后的 P 值通常标记为 `padj` 或 `q-value`。

```
# BH 校正简化逻辑：
1. 把所有 P 值从小到大排列：p(1) ≤ p(2) ≤ ... ≤ p(m)
2. 对第 i 小的 P 值：padj(i) = p(i) × m / i
3. 取单调性保证（从大到小取较小值）
4. 最终用 padj < 0.05 作为显著性阈值
```

概念	控制的是什么	常用阈值
原始 P 值	单个检验的假阳性率	P < 0.05
FDR (padj)	所有显著结果中假阳性的**比例**	padj < 0.05（即假阳性比例 < 5%）

> **实践建议**
>
> 生信分析中**始终使用校正后的 P 值（padj/FDR）**作为显著性判断标准。DESeq2、edgeR 等工具默认输出 padj 列。

4.3 log2FoldChange 解读

差异表达分析中，基因表达变化的幅度用 **log2FoldChange（log2FC）**衡量，已在第 3 章详细说明。这里从统计学角度补充几个要点：

- **为什么用 log2 而不是原始倍数？**原始倍数变化不对称（上调 4 倍 = 4，下调 4 倍 = 0.25），取 log2 后对称（+2 和 −2），便于可视化和统计
- **log2FC 的标准误差**：DESeq2 会同时输出 `lfcSE`（log2FC 的标准误差），SE 越大说明估计越不可靠
- **效应量与显著性**的关系：P 值衡量"有没有差异"，log2FC 衡量"差异有多大"。两者都重要——P 值小但 log2FC 接近 0 的基因可能只是因为样本量大而检测到了微小差异

4.4 PCA 与聚类的基本思想

4.4.1 PCA（主成分分析）

PCA（Principal Component Analysis）是一种降维方法，把高维数据（如几万个基因的表达量）压缩到 2D 或 3D 空间中可视化。

> **生活比喻**
>
> 想象你要用一张照片展示一个 3D 雕像。你会找一个**最能体现雕像特征的角度**来拍。PCA 做的就是这件事——从几万个维度中找到"最能区分样本"的 2 个方向，画在二维图上。

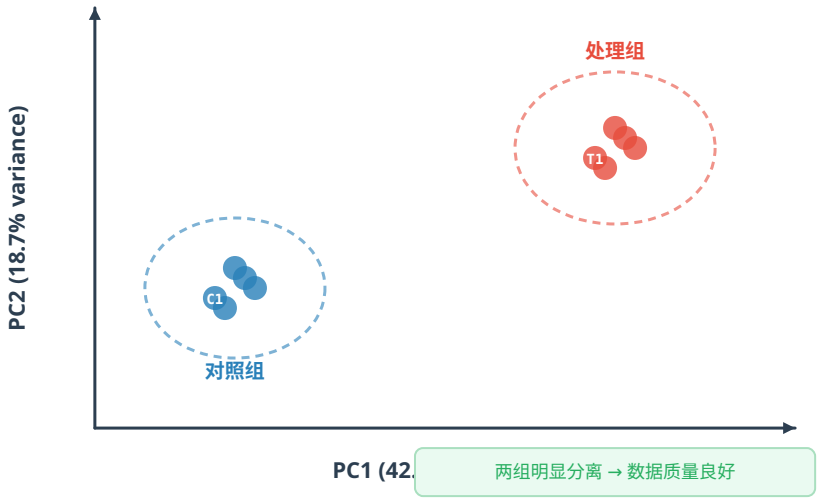

图 4-1　*PCA* 散点图示意：对照组（蓝）与处理组（红）在 *PC1/PC2* 空间中明显分离

PCA 图的解读：

- 每个点代表一个样本
- 距离越近的样本，表达谱越相似
- 如果生物学重复聚在一起、不同组分开，说明数据质量好
- PC1（第一主成分）通常解释最大的变异来源

> **PCA 在生信中的用途**
>
> - **质控**：检查样本是否按实验分组聚集，有无离群样本
> - **发现批次效应**：如果 PCA 图上样本按测序日期而非处理组分聚，说明存在批次效应
> - **可视化**：在论文中展示组间差异的总体趋势

4.4.2 聚类分析（Clustering）

聚类是将相似的样本/基因自动分组的无监督方法。生信中最常用的是**层次聚类（Hierarchical Clustering）**：

- **样本聚类**：表达谱相似的样本聚在一起 → 检查样本重复性
- **基因聚类**：表达模式相似的基因聚在一起 → 找共调控基因模块
- 常配合**热图（Heatmap）**可视化：行 = 基因，列 = 样本，颜色 = 表达量高低

> **PCA vs 聚类**
>
> - **PCA**：降维可视化，看全局趋势，适合 2D 散点图
> - **聚类**：自动分组，适合热图和树状图
> - 两者常配合使用：先 PCA 看总体分布，再聚类细分模块

—— 第 4 章结束 ——

第 5 章　常见数据库

5.1 NCBI（GenBank / GEO / SRA）

NCBI（National Center for Biotechnology Information，美国国家生物技术信息中心）是全球最大的生物信息数据库，旗下包含多个子库：

子库	全称	存储内容	生信用途
GenBank	Genetic Sequence Database	所有已公开的核酸序列	序列检索、BLAST 搜索、获取参考序列
GEO	Gene Expression Omnibus	基因表达数据（RNA-seq、微阵列、ChIP-seq 等）	下载公开表达数据、复现分析、数据挖掘
SRA	Sequence Read Archive	原始测序数据（FASTQ）	下载原始 reads 重新分析
PubMed	—	生物医学文献	查找文献、获取方法参考

生信常用操作

- **从 GEO 找数据**：搜索疾病/组织名 → 找到 GSE 系列 → 查看实验设计 → 下载表达矩阵或原始数据
- **从 SRA 下载 FASTQ**：用 `sratoolkit` 的 `prefetch` + `fasterq-dump` 命令
- **BLAST 搜序列**：输入查询序列 → 选择数据库 → 运行 → 查看相似序列

5.2 Ensembl

Ensembl 是欧洲生物信息研究所（EBI）维护的基因组注释数据库，与 NCBI 并列为两大参考基因组来源。

特点	说明
基因组注释	提供多物种的参考基因组和高质量基因注释（GTF/GFF3）
BioMart	批量获取基因 ID 转换、序列、注释信息的强大工具
ID 系统	ENSG（基因）、ENST（转录本）、ENSP（蛋白质），稳定且唯一
比较基因组	提供直系同源（ortholog）和旁系同源（paralog）信息
变异数据	整合 dbSNP、ClinVar 等变异数据

5.3 UCSC Genome Browser

UCSC Genome Browser 是加州大学圣克鲁兹分校维护的基因组浏览器，是**可视化基因组区域**的首选工具。

- **基因组浏览**：输入染色体坐标或基因名 → 在基因组上可视化该区域的所有注释 track
- **多 track 叠加**：可以同时查看基因模型、保守性、变异、表达、调控元件等几十种 track
- **序列提取**：提取指定区域的 DNA/蛋白质序列
- **Table Browser**：批量导出指定区域的注释数据（BED/GTF 格式）
- **BLAT**：快速序列比对工具，比 BLAST 更适合找基因组中的精确位置

5.4 GO 与 KEGG 通路数据库

5.4.1 GO（Gene Ontology）

GO 是一套标准化的基因功能描述体系，分三大类：

类别	描述	示例
Biological Process（BP）	生物学过程	细胞凋亡、DNA 修复、代谢通路
Molecular Function（MF）	分子功能	激酶活性、DNA 结合、催化活性
Cellular Component（CC）	细胞组分	细胞核、线粒体、核糖体

GO 富集分析：给定一组差异基因，判断它们是否在某些 GO term 中"过度出现"，从而推断这组基因可能参与的生物学功能。

5.4.2 KEGG

KEGG（Kyoto Encyclopedia of Genes and Genomes）是**通路（Pathway）数据库**，以图形化方式展示代谢和信号通路。

- 每个通路是一张**可视化图谱**，标注了基因/酶在通路中的位置
- **KEGG 富集分析**：判断差异基因是否在某些通路中显著富集 → 推断受影响的生物学通路
- 常与 GO 富集分析配合使用：GO 看功能分类，KEGG 看通路层面的系统性影响

5.5 Bioconductor 在生信生态中的位置

Bioconductor 是基于 R 语言的生物信息分析包仓库，是 R 语言生信生态的核心。

方面	说明
定位	类似 CRAN，但专注于高通量基因组学数据分析
包数量	2000+ 个包，覆盖从数据导入到可视化全流程
核心包	`DESeq2`（差异表达）、`edgeR`（差异表达）、`limma`（线性模型）、`clusterProfiler`（富集分析）
数据结构	`SummarizedExperiment`、`GRanges` 等标准化数据容器
发布周期	半年一次（4 月和 10 月），与 R 版本对齐

```
# 安装 Bioconductor 的标准方式
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("DESeq2")
BiocManager::install("clusterProfiler")
```

—— 第 5 章结束 ——

附录 A　术语速查表 · 中心法则示意图 · FASTQ 格式图解

A.1 术语速查表

分子生物学与生信术语速查表

术语	英文	简要说明
中心法则	Central Dogma	遗传信息流向：DNA → RNA → 蛋白质
基因	Gene	DNA 上有功能的片段，遗传信息基本单位
转录本	Transcript	一个基因通过可变剪接产生的 mRNA 变体
外显子	Exon	基因中保留在成熟 mRNA 中的序列
内含子	Intron	基因中在剪接时被去除的序列
启动子	Promoter	基因上游调控转录起始的 DNA 区域
UTR	Untranslated Region	mRNA 上不被翻译的 5' / 3' 端区域
CDS	Coding Sequence	编码蛋白质的序列（ATG 到 Stop 之间）
密码子	Codon	mRNA 上 3 个相邻核苷酸，对应一个氨基酸
读长	Read Length	单条测序读取的碱基长度
覆盖度	Coverage / Depth	每个碱基平均被测到的次数
FASTA	—	只含序列信息的文本格式（> 开头）
FASTQ	—	含序列 + 质量值的格式，每条 read 4 行
Phred 值	Phred Score	碱基质量分值，Q30 = 99.9% 准确
GTF / GFF	—	基因组注释格式，9 列，1-based 闭区间
BED	—	基因组区间格式，最少 3 列，0-based 半开
计数矩阵	Count Matrix	基因 × 样本的原始 read 计数表
TPM	Transcripts Per Million	长度+深度校正后的表达量，推荐使用
FPKM / RPKM	—	旧版标准化方式，不建议做差异分析
log2FC		表达量变化的 log2 倍数，正值上调、负值下调

术语	英文	简要说明
	log2 Fold Change	
P 值	P-value	原假设成立时观察到当前数据的概率
FDR	False Discovery Rate	校正后的 P 值，控制假阳性比例
PCA	Principal Component Analysis	降维可视化方法，看样本总体分布
聚类	Clustering	将相似样本/基因自动分组
GO	Gene Ontology	基因功能标准化分类体系（BP/MF/CC）
KEGG	—	通路数据库，可视化代谢/信号通路
Bioconductor	—	R 语言生信包仓库，核心分析工具来源

A.2 中心法则示意图

图 *A-1*　中心法则完整示意图

DNA 储存遗传信息 → 转录为 mRNA 携带信息 → 翻译为蛋白质执行功能
原版图纸 → 工作副本 → 最终产品

A.3 FASTQ 格式图解

```
@SRR1234567.1 HWUSI-AS1234:1:1101:1000:2000 length=150
← 第 1 行：@ + read ID + 测序仪元数据
ATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGATCGAT
← 第 2 行：碱基序列（A/T/G/C/N）
+
← 第 3 行：分隔符（+）
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
← 第 4 行：质量值（Phred+33 编码）
```

图 *A-2*　*FASTQ* 格式完整图解

质量值字符速查

字符	'+'	'5'	'?'	'I'
Phred 值	Q10	Q20	Q30	Q40
准确率	90%	99%	99.9%	99.99%

—— 附录 A 结束 ——

PART TWO

Linux 基础入门

第 6 章　为什么学 Linux

如果你问一个生信工程师："日常工作中最常用的工具是什么？"答案不是某个 R 包，也不是某个数据库——而是 **Linux 命令行**。

6.1 生信离不开 Linux 的三大原因

#	原因	详细说明
1	**服务器 / 超算 / 云平台均运行 Linux**	生信数据量动辄几十到上百 GB，个人电脑跑不动。几乎所有生信分析都在 Linux 服务器或超算上完成。你必须通过命令行远程操作这些机器
2	**生信软件原生支持 Linux**	STAR、BWA、samtools、DESeq2 的底层依赖……绝大多数生信工具优先甚至**仅**支持 Linux。Windows 上需要 WSL 虚拟环境，macOS 兼容性更好但仍非原生
3	**公共数据下载和批量处理需要命令行**	从 SRA 下载数据用 `prefetch`，批量处理几十个样本用 shell 脚本循环，这些在图形界面下根本无法高效完成

真实场景

你拿到一个 RNA-seq 项目，30 个样本，每个样本的 FASTQ 约 5 GB。
要做的事情：下载 → 质控 → 比对 → 定量 → 差异分析 → 富集分析。
这整个流程在 Linux 上用一个 shell 脚本就能批量跑完，而在 Windows 图形界面下几乎不可能高效完成。

6.2 学习路线建议

零基础学习路线

1. **先过一遍基础概念**：文件系统结构、路径、权限（第 7 章）
2. **掌握 20 个高频命令**：覆盖 90% 日常操作（第 8 章）
3. **理解管道与重定向**：组合命令的核心思维（第 9 章）
4. **学会 conda 管理环境**：装软件不再痛苦（第 10 章）
5. **会用 SSH 连服务器**：实际工作场景（第 11 章）
6. **在真实项目中练习**：跑一遍 RNA-seq 流程，自然就熟了

—— 第 6 章结束 ——

第 7 章　基础概念

7.1 文件系统树状结构

Linux 的文件系统是一棵**倒置的树**，所有文件和目录都从**根目录** `/` 开始。

图 7-1　*Linux* 文件系统树状结构（不同颜色区分不同用途的目录）

目录	用途	生信场景
`/`	根目录，一切起点	—
`/home/用户名`	用户主目录	你的分析脚本、结果通常放这里
`/data` 或 `/mnt/data`	数据存储目录	大文件（FASTQ、BAM）常放此处的共享存储
`/tmp`	临时文件	中间文件可放这里，但注意空间限制
`/opt`	第三方软件	手动安装的生信软件

7.2 绝对路径 vs 相对路径

类型	定义	示例	特点
绝对路径	从根目录 `/` 开始的完整路径	`/home/alice/data/fastq/sample1.fastq`	不管你在哪，都指向同一位置
相对路径	从**当前所在目录**出发的路径	`data/fastq/sample1.fastq`	取决于当前目录，更简洁但可能混淆

特殊目录符号

符号	含义	示例
`.`	当前目录	`./script.sh` = 执行当前目录下的脚本
`..`	上一级目录	`cd ..` = 返回上级
`~`	用户主目录（home）	`cd ~` = 回到 /home/用户名
`-`	上一次所在目录	`cd -` = 在两个目录间切换

7.3 文件权限（读 / 写 / 执行）

Linux 中每个文件和目录都有权限属性，控制谁能读、写、执行。

```
$ ls -l
-rwxr-xr-- 1 alice bioinfo 2.3M Aug 16 10:30 align.sh
 |┆┆┆┆┆┆┆┆┆┆
 |┆┆┆┆┆┆┆┆└── others（其他用户）权限：r--（只读）
 |┆┆┆┆┆┆┆└─── group（同组用户）权限：r-x（读+执行）
 |┆┆┆┆┆┆└──── owner（文件所有者）权限：rwx（读+写+执行）
 |
 └─── 文件类型：- 普通文件，d 目录，l 符号链接
```

权限	字母	数字	对文件	对目录
读	r	4	查看文件内容	列出目录内容（ls）
写	w	2	修改文件内容	在目录中创建/删除文件
执行	x	1	运行脚本/程序	进入目录（cd）

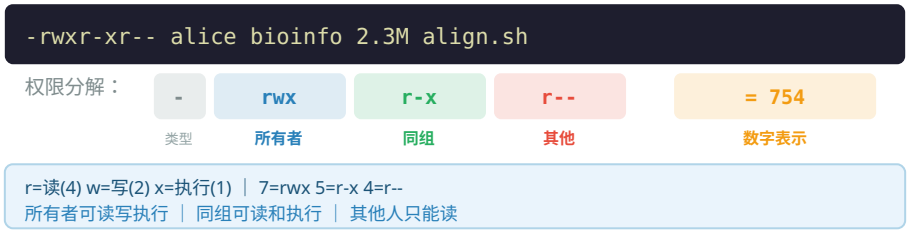

图 7-2　文件权限可视化：*rwx* 三组权限对应所有者、同组、其他用户

```
# 修改权限的常见方式

# 方式一：数字表示法（owner=7, group=5, others=4）
chmod 754 align.sh
# 7 = 4+2+1 = rwx   （所有者：读+写+执行）
# 5 = 4+0+1 = r-x   （同组：读+执行）
# 4 = 4+0+0 = r--   （其他：只读）

# 方式二：字母表示法
chmod +x align.sh      # 给所有人加执行权限
chmod u+x align.sh     # 仅给 owner 加执行权限
chmod g-w align.sh     # 去掉 group 的写权限
```

生信常见场景

下载了一个 shell 脚本或二进制程序，直接运行会报"Permission denied"。这是因为文件没有执行权限。执行 `chmod +x 文件名` 即可。

—— 第 7 章结束 ——

第 8 章　高频命令精讲

本章覆盖生信日常工作中最常用的命令，每条命令都附带**实际生信场景示例**。

8.1 文件与目录操作

ls — 列出目录内容

```
# 基本用法
ls                          # 列出当前目录文件
ls -l                       # 详细信息（权限、大小、日期）
ls -lh                      # -h：人类可读大小（K/M/G）
ls -la                      # -a：显示隐藏文件（以 . 开头）
ls -lt                      # 按修改时间排序（新的在前）
ls -lS                      # 按文件大小排序（大的在前）

# 生信示例
ls -lh /data/fastq/                     # 查看 FASTQ 文件大小
ls -lh *.fastq.gz                       # 只列出 fastq.gz 文件
ls -lt /data/results/ | head -10        # 查看最新生成的结果文件
```

cd — 切换目录

```
cd /data/fastq                # 切换到指定目录（绝对路径）
cd results/                   # 切换到子目录（相对路径）
cd ..                         # 返回上一级
cd ../..                      # 返回上两级
cd ~                          # 回到主目录
cd -                          # 回到上次所在目录

# 生信示例
cd /data/projects/rnaseq/     # 进入 RNA-seq 项目目录
cd ~/analysis/                 # 进入主目录下的 analysis 文件夹
```

pwd — 显示当前目录

```
pwd                              # 输出当前绝对路径

# 生信示例
# 在服务器上不知道自己在哪时，先 pwd 确认
pwd
# 输出：/home/alice/projects/rnaseq/results
```

mkdir — 创建目录

```
mkdir fastq                      # 创建单个目录
mkdir -p results/figures/heatmap    # -p：递归创建（父目录不存在时自动建）

# 生信示例
mkdir -p /data/projects/rnaseq/{fastq,aligned,counts,de,figures}
# 一次创建 5 个子目录（花括号展开）
```

cp — 复制文件 / 目录

```
cp file.txt backup.txt               # 复制文件
cp -r results/ results_backup/       # -r：递归复制目录
cp -i file.txt existing.txt          # -i：覆盖前确认

# 生信示例
cp /data/reference/hg38.fa ~/analysis/ref/     # 复制参考基因组
cp *.fastq.gz /backup/fastq/                    # 复制所有 fastq.gz 到备份目录
```

mv — 移动 / 重命名

```
mv old_name.txt new_name.txt         # 重命名
mv file.txt /target/dir/             # 移动文件到目录

# 生信示例
mv sample1.fastq.gz sample1_R1.fastq.gz      # 重命名测序文件
mv /tmp/*.bam /data/results/bam/             # 把临时目录的 BAM 移到结果目录
```

rm — 删除

```
rm file.txt                          # 删除文件（不进回收站！）
rm -r results/                       # -r：递归删除目录
rm -i important.txt                  # -i：删除前确认
rm -f temp_*.txt                     # -f：强制删除，不报错

# 生信示例
rm /tmp/*.sam                        # 清理中间文件（SAM 很大）
# ⚠ 警告：rm -rf / 会删除整个系统，永远不要这样用！
```

rm 不可逆

Linux 没有"回收站"。`rm` 删除的文件**无法恢复**。建议：

1. 删除前先 `ls` 确认要删的文件
2. 重要操作加 `-i` 参数
3. 永远不要在 root 目录下执行 `rm -rf *`

8.2 文件内容查看与搜索

head / tail — 查看文件开头 / 结尾

```
head file.txt                        # 默认前 10 行
head -n 20 file.txt                  # 前 20 行（-n 20 或 -20 均可）
tail file.txt                        # 后 10 行
tail -n 50 file.txt                  # 后 50 行
tail -f logfile.log                  # -f：实时跟踪新内容（看日志神器）

# 生信示例
head -4 sample.fastq                 # 查看 FASTQ 前 4 行（即第一条 read）
head -1 *.fastq.gz                   # 检查每个 fastq 文件的 ID 格式
tail -f star_align.log               # 实时看 STAR 比对进度
head -100 counts.txt | tail -20      # 看第 81-100 行
```

less — 分页查看大文件

```
less large_file.txt                # 分页查看（按 q 退出）
less -S large_file.txt             # -S：长行不换行（左右滚动）

# 常用快捷键（less 内部）：
# 空格/f    向下翻页
# b         向上翻页
# /关键词   向下搜索
# ?关键词   向上搜索
# n         下一个匹配
# N         上一个匹配
# q         退出

# 生信示例
less -S genes.gtf                  # 查看 GTF 注释文件（行很长）
less counts_matrix.tsv             # 查看表达矩阵
```

grep — 文本搜索

```
grep "pattern" file.txt            # 搜索包含 pattern 的行
grep -i "pattern" file.txt         # -i：忽略大小写
grep -v "pattern" file.txt         # -v：反向匹配（不包含的行）
grep -c "pattern" file.txt         # -c：只输出匹配行数
grep -n "pattern" file.txt         # -n：显示行号
grep -w "BRCA1" genes.txt          # -w：全词匹配（避免匹配 BRCA1-AS）
grep -A 3 "pattern" file.txt       # -A 3：匹配行后再显示 3 行
grep -B 2 "pattern" file.txt       # -B 2：匹配行前显示 2 行
grep -E "pat1|pat2" file.txt       # -E：扩展正则（或匹配）

# 生信示例
grep -c "^>" reference.fa          # 统计 FASTA 中序列条数（>开头的行数）
grep "BRCA1" genes.gtf             # 找 GTF 中 BRCA1 的注释行
grep -v "^#" file.vcf              # 去掉 VCF 文件的注释行（#开头）
grep -E "exon|CDS" genes.gtf       # 提取 GTF 中 exon 或 CDS 行
grep -w "chr1" regions.bed         # 在 BED 中找 chr1 的行（全词匹配防止 chr11/chr1* 被
误匹配）
```

wc — 统计行数 / 词数 / 字节数

```
wc -l file.txt                        # -l：统计行数
wc -w file.txt                        # -w：统计词数
wc -c file.txt                        # -c：统计字节数

# 生信示例
wc -l sample.fastq                    # FASTQ 行数 ÷ 4 = read 数
# 例如输出 12000000 → 3,000,000 条 reads
echo $(( $(wc -l < sample.fastq) / 4 ))   # 直接算出 read 数
```

8.3 文本处理三剑客

sort — 排序

```
sort file.txt                         # 按字母排序
sort -n numbers.txt                   # -n：按数值排序
sort -r file.txt                      # -r：逆序
sort -k2 file.txt                     # -k2：按第 2 列排序
sort -k2,2n file.txt                  # 按第 2 列数值排序
sort -u file.txt                      # -u：排序并去重

# 生信示例
sort -k1,1 -k2,2n peaks.bed           # 按 chr 排序，chr 内按 start 坐标排序
sort -k5,5nr gene_counts.txt          # 按第 5 列（count）数值降序排列
```

uniq — 去重（通常配合 sort 使用）

```
sort file.txt | uniq                  # 排序后去重
sort file.txt | uniq -c               # -c：统计每行出现次数
sort file.txt | uniq -d               # -d：只显示重复的行

# 生信示例
cut -f1 genes.gtf | sort | uniq -c  # 统计 GTF 每种特征类型（gene/exon/CDS）的数量
# 输出示例:
#  50000 exon
#  20000 CDS
#  10000 gene
```

cut — 按列提取

```
cut -f1 file.txt                    # 提取第 1 列（默认 Tab 分隔）
cut -f1,3 file.txt                  # 提取第 1 和第 3 列
cut -d"," -f2 file.csv              # -d：指定分隔符（逗号），提取第 2 列
cut -c1-10 file.txt                 # -c：提取第 1-10 个字符

# 生信示例
cut -f1,4,5 genes.gtf               # 提取 GTF 的 chr、start、end 列
cut -f1 counts.txt | head           # 看计数矩阵的基因 ID 列
cut -d" " -f1-3 file.sam            # 提取 SAM 文件前三列
```

8.4 下载与压缩

wget — 文件下载

```
wget https://example.com/file.gz     # 下载文件
wget -c https://example.com/file.gz  # -c：断点续传
wget -O newname.gz https://url/old.gz  # -O：指定保存文件名
wget -q https://example.com/file.gz # -q：静默模式（不输出进度）

# 生信示例
wget https://ftp.ensembl.org/pub/release-110/fasta/homo_sapiens/dna/
Homo_sapiens.GRCh38.dna.toplevel.fa.gz
# 下载 Ensembl 人类参考基因组
```

curl — 数据传输（更灵活）

```
curl -O https://example.com/file.gz  # -O：用原文件名保存
curl -o newname.gz https://url/file.gz  # -o：指定文件名
curl -L https://short.url/abc        # -L：跟随重定向
curl -s https://api.example.com/data # -s：静默模式

# 生信示例
curl -L -o miniconda.sh https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-
x86_64.sh
# 下载 Miniconda 安装脚本
```

tar — 压缩 / 解压

```
# 解压 .tar.gz
tar -zxvf file.tar.gz                # -z：gzip，-x：解压，-v：显示过程，-f：指定文件
tar -zxvf file.tar.gz -C /target/    # -C：解压到指定目录

# 压缩为 .tar.gz
tar -zcvf archive.tar.gz dir/        # -c：创建压缩包

# 解压 .tar.bz2
tar -jxvf file.tar.bz2               # -j：bzip2

# 生信示例
tar -zxvf fastq_files.tar.gz         # 解压测序数据包
tar -zcvf results_backup.tar.gz results/  # 打包备份分析结果
tar -zxvf tools.tar.gz -C /opt/      # 解压软件到 /opt 目录
```

gzip / gunzip — 单文件压缩 / 解压

```
gzip file.txt                       # 压缩（原文件消失，变为 file.txt.gz）
gzip -k file.txt                     # -k：保留原文件
gunzip file.txt.gz                   # 解压
zcat file.txt.gz                     # 不解压直接查看内容（不占额外空间）

# 生信示例
zcat sample.fastq.gz | head -4       # 查看压缩 FASTQ 的前 4 行
zgrep "BRCA1" genes.gtf.gz           # 在压缩 GTF 中搜索
zless sample.fastq.gz                # 分页查看压缩文件
```

8.5 进程与会话管理

top / htop — 查看系统资源

```
top                                  # 实时查看 CPU/内存/进程
htop                                 # 更友好的版本（需安装）

# top 内常用按键：
# P     按 CPU 使用率排序
# M     按内存使用率排序
# q     退出

# 生信示例
top -u alice                         # 只看自己的进程
# 如果 STAR 占了 32G 内存，说明它在正常运行
# 如果进程已消失但结果没出来，说明可能报错了
```

screen — 终端会话管理

```
# 创建新会话
screen -S rnaseq_align               # -S：给会话命名

# 在会话中运行命令后，按 Ctrl+A 然后按 D 脱离（detach）
# 命令继续在后台运行！

# 查看已有会话
screen -ls

# 重新连接会话
screen -r rnaseq_align               # -r：恢复会话

# 生信示例
screen -S star_align
STAR --runThreadN 8 --genomeDir ~/ref/star_index \
  --readFilesIn sample_R1.fastq.gz sample_R2.fastq.gz \
  --readFilesCommand zcat \
  --outSAMtype BAM SortedByCoordinate
# 按 Ctrl+A D 脱离，关掉电脑回家，命令继续跑
# 第二天回来：screen -r star_align
```

tmux — 更强大的终端复用器

```
# 新建会话
tmux new -s rnaseq

# 脱离：Ctrl+B 然后按 D

# 列出会话
tmux ls

# 恢复会话
tmux attach -t rnaseq

# 生信示例：在 tmux 中开多个窗口并行跑不同样本
# Ctrl+B C    创建新窗口
# Ctrl+B 0/1  切换窗口
# 每个窗口跑一个样本的比对
```

nohup — 后台运行（断开连接也不中断）

```
nohup command &                       # 后台运行，输出重定向到 nohup.out
nohup command > output.log 2>&1 &     # 自定义输出文件，2>&1 合并标准错误到标准输出

# 查看后台任务
jobs                                  # 查看当前 shell 的后台任务
jobs -l                               # 包含 PID

# 生信示例
nohup STAR --runThreadN 16 --genomeDir ref/ \
  --readFilesIn sample.fq.gz \
  --readFilesCommand zcat \
  > star.log 2>&1 &

# 用 tail -f star.log 实时看进度
# 关掉 SSH 连接，任务继续跑
```

chmod — 修改权限

```
chmod +x script.sh                  # 加执行权限
chmod 755 program                   # rwxr-xr-x
chmod 644 data.txt                  # rw-r--r--

# 生信示例
chmod +x run_pipeline.sh            # 让脚本可执行
./run_pipeline.sh                   # 然后才能直接运行
```

—— 第 8 章结束 ——

第 9 章　管道与重定向

管道和重定向是 Linux 命令行的**核心思维方式**——把简单命令组合成强大的数据处理流水线。

9.1 管道 |

管道（|）将前一个命令的标准输出作为后一个命令的标准输入。可以串联多个命令，形成"流水线"。

```
命令A | 命令B | 命令C      # A 的输出 → B 的输入 → B 的输出 → C 的输入
```

生活比喻

就像工厂流水线：第一道工序的半成品直接送到第二道工序，不需要中间仓库暂存。管道让数据在命令间"无缝传递"，无需创建中间文件。


```
zcat sample.fq.gz | grep "ATCG" | wc -l
```

图 9-1　管道流水线示意：解压 → 搜索 → 计数，数据在命令间无缝传递

生信中的管道实战

```
# 1. 统计 FASTQ 的 read 数（不解压）
zcat sample.fastq.gz | wc -l        # 总行数
echo $(( $(zcat sample.fastq.gz | wc -l) / 4 ))  # read 数

# 2. 统计 GTF 中各特征类型数量
cut -f3 genes.gtf | sort | uniq -c
# 输出：
#   50000 exon
#   20000 CDS
#   10000 transcript
#    5000 gene

# 3. 找出表达量最高的前 10 个基因
sort -k7,7nr gene_counts.txt | head -10

# 4. 从 VCF 中提取 chr1 的变异并计数
grep "^chr1" variants.vcf | wc -l

# 5. 从 BED 文件提取 chr1 的区域并排序
grep -w "chr1" regions.bed | sort -k2,2n > chr1_sorted.bed

# 6. 多步处理：提取 GTF 中 chr1 的 exon，排序，统计数量
grep -w "chr1" genes.gtf | grep "exon" | cut -f4,5 | sort -k1,1n | wc -l
```

9.2 重定向 > 和 >>

重定向将命令输出写入文件而非屏幕。

符号	含义	行为
`>`	输出重定向	**覆盖**写入文件（文件不存在则创建）
`>>`	追加重定向	**追加**到文件末尾（不覆盖原内容）
`2>`	错误重定向	将**标准错误**输出到文件
`&>`	全部重定向	将标准输出**和**标准错误都输出到文件
`<`	输入重定向	从文件读取输入（而非键盘）

```
# 基本用法
echo "Hello" > output.txt           # 覆盖写入
echo "World" >> output.txt          # 追加写入
ls /nonexistent 2> error.log       # 只把错误信息写入文件
ls /data/ &> all_output.log        # 正常输出和错误都写入

# 标准输出和标准错误分别写入不同文件
command > output.log 2> error.log

# 丢弃输出（/dev/null 是"黑洞"）
command > /dev/null 2>&1           # 什么都不显示

# 生信示例
grep "BRCA1" genes.gtf > brca1_annotations.txt  # 提取结果写入文件
sort -k1,1 -k2,2n peaks.bed >> all_sorted.bed   # 追加排序结果
STAR ... > star.log 2>&1                        # 比对日志全部写入文件
samtools view -h aligned.bam chr1 2>/dev/null | samtools sort -o chr1.bam  # 忽略警告
```

9.3 管道 + 重定向组合

```
# 管道用于中间处理，重定向用于最终保存

# 提取 GTF 中 chr1 外显子 → 排序 → 保存为 BED
grep -w "chr1" genes.gtf | grep "exon" | \
  cut -f1,4,5 | sort -k2,2n > chr1_exons.bed

# 统计每个染色体上的基因数 → 保存
cut -f1 genes.gtf | sort | uniq -c | sort -rn > gene_count_per_chr.txt

# FASTQ 质控：解压 → fastp 质控 → 压缩保存
zcat sample.fastq.gz | fastp --stdin --stdout -o clean.fastq.gz
```

—— 第 9 章结束 ——

第 10 章　conda / mamba 实战

conda 是生信工作者最重要的软件管理工具——没有之一。它能帮你解决"装软件地狱"的问题。

10.1 为什么需要 conda

问题	没有 conda 时	有 conda 后
软件依赖冲突	手动编译、版本冲突、系统库不兼容	每个环境独立，互不影响
安装生信软件	下载源码 → ./configure → make → 报错 → Google → 重试	`conda install -c bioconda samtools` 一行搞定
管理多版本	手动切换路径、设置环境变量	不同环境装不同版本，自由切换
团队协作	"我这能跑你那不能跑"的环境地狱	导出 environment.yml，他人一键复现

10.2 安装 Miniconda

```
# 1. 下载 Miniconda 安装脚本
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

# 2. 运行安装
bash Miniconda3-latest-Linux-x86_64.sh

# 3. 按提示操作（一路回车 + yes），安装完成后重新打开终端

# 4. 验证安装
conda --version
# 输出：conda 24.x.x
```

10.3 配置 bioconda 频道

conda 的软件来自不同的"频道（channel）"。生信软件主要集中在 **bioconda** 频道。配置时必须按以下顺序添加：

```
# 设置频道（顺序很重要！）
conda config --add channels defaults
conda config --add channels conda-forge
conda config --add channels bioconda

# 设置默认使用 conda-forge
conda config --set channel_priority strict

# 验证配置
conda config --show channels
# 应输出：
#   - bioconda
#   - conda-forge
#   - defaults
```

频道顺序很重要

必须是 **bioconda > conda-forge > defaults**。顺序错误会导致依赖冲突。

`channel_priority strict` 确保从优先频道取包。

10.4 创建与管理环境

```
# 创建新环境
conda create -n rnaseq python=3.10

# 激活环境
conda activate rnaseq
# 激活后，命令行提示符前会出现 (rnaseq)

# 退出环境
conda deactivate

# 列出所有环境
conda env list

# 删除环境
conda env remove -n rnaseq

# 生信示例：创建 RNA-seq 分析环境
conda create -n rnaseq python=3.10
conda activate rnaseq
conda install -c bioconda star samtools fastp
conda install -c bioconda subread  # featureCounts
```

10.5 安装生信软件

```
# 安装单个软件
conda install -c bioconda samtools
conda install -c bioconda bwa
conda install -c bioconda star

# 安装指定版本
conda install -c bioconda samtools=1.18

# 搜索可用软件
conda search -c bioconda fastp

# 查看已安装的软件
conda list

# 更新软件
conda update samtools

# 生信常用软件一键安装
conda install -c bioconda fastp star samtools subread multiqc
```

10.6 环境导出与复现

```
# 导出环境配置文件
conda env export > rnaseq_env.yml

# 在另一台机器上复现
conda env create -f rnaseq_env.yml

# 更轻量的导出（只包含手动指定的包）
conda env export --from-history > rnaseq_env.yml
```

10.7 mamba：更快的 conda

mamba 是 conda 的 C++ 重写版，解决依赖的速度比 conda 快 10–100 倍，强烈推荐。

```
# 安装 mamba
conda install -c conda-forge mamba

# 之后把所有 conda 命令替换为 mamba 即可
mamba install -c bioconda star
mamba create -n rnaseq python=3.10 star samtools fastp
```

生信环境管理最佳实践

- 每个项目/流程创建独立环境，避免冲突
- 用 `mamba` 代替 `conda` 安装软件，速度快很多
- 导出 environment.yml 保存环境配置，方便复现
- 不要往 base 环境装东西，base 只用来管理 conda 本身

—— 第 10 章结束 ——

第 11 章　SSH 远程连接

生信分析几乎都在远程服务器上进行，**SSH（Secure Shell）**是你连接服务器的标准方式。

11.1 基本连接

```
# 基本格式
ssh 用户名@服务器地址

# 生信示例
ssh alice@10.0.1.100                    # 连接内网服务器
ssh alice@bioinfo.server.edu.cn         # 连接域名服务器
ssh -p 22222 alice@server.com           # -p：指定端口（默认 22）

# 首次连接会提示确认指纹，输入 yes 即可
```

11.2 免密登录（SSH 密钥）

```
# 1. 在本地生成密钥对（一路回车即可）
ssh-keygen -t ed25519 -C "alice@laptop"

# 2. 将公钥复制到服务器
ssh-copy-id alice@server.com

# 3. 之后直接 ssh 即可，无需输入密码
ssh alice@server.com
```

11.3 scp — 远程文件复制

```
# 从本地复制到服务器
scp local_file.txt alice@server:/data/fastq/

# 从服务器复制到本地
scp alice@server:/data/results/counts.txt ./

# 复制整个目录（-r）
scp -r results/ alice@server:/data/projects/

# 指定端口
scp -P 22222 file.txt alice@server:/data/
```

11.4 rsync — 增量同步（更高效）

```
# 基本同步
rsync -avz local_dir/ alice@server:/data/remote_dir/

# 参数说明：
# -a  归档模式（保留权限、时间等）
# -v  显示详细信息
# -z  压缩传输
# --progress  显示进度
# -P  支持断点续传

# 生信示例：同步大量 FASTQ 文件
rsync -avP /data/fastq/ alice@server:/data/fastq/

# rsync 的优势：只传输变化的文件，速度远超 scp
```

11.5 ~/.ssh/config — SSH 配置文件

在本地 `~/.ssh/config` 文件中配置服务器别名，省去每次输入长地址的麻烦：

```
# 编辑 ~/.ssh/config（没有就创建）
Host bioinfo
    HostName     10.0.1.100
    User         alice
    Port         22
    IdentityFile ~/.ssh/id_ed25519
    ServerAliveInterval 60

Host gpu-cluster
    HostName     gpu.server.edu.cn
    User         alice
    Port         22222

# 配置后直接用别名连接
ssh bioinfo          # = ssh -p 22 alice@10.0.1.100
ssh gpu-cluster      # = ssh -p 22222 alice@gpu.server.edu.cn

# scp 也可以用别名
scp file.txt bioinfo:/data/
rsync -avP results/ bioinfo:/data/results/
```

生信工作流推荐配置

1. 配置 SSH 密钥免密登录
2. 写好 `~/.ssh/config`，给服务器起短名
3. 设置 `ServerAliveInterval 60` 防止连接超时断开
4. 大文件传输用 `rsync -avP` 而非 scp
5. 长时间任务配合 `screen` / `tmux` 使用

—— 第 11 章结束 ——

第 12 章　实用小技巧

12.1 通配符 *

通配符是命令行中最重要的"快捷键"之一，让你批量操作文件而无需逐个输入文件名。

```
# * 匹配任意数量的任意字符
ls *.fastq              # 所有 .fastq 文件
ls *.fastq.gz           # 所有 .fastq.gz 文件
ls sample_*             # 所有以 sample_ 开头的文件

# ? 匹配单个字符
ls sample_?.fastq      # sample_1.fastq, sample_2.fastq（不匹配 sample_10）

# [] 匹配括号内的任意一个字符
ls sample_[1-3].fastq  # sample_1.fastq, sample_2.fastq, sample_3.fastq

# {} 展开
cp file.txt{,.bak}     # = cp file.txt file.txt.bak
mkdir -p dir/{a,b,c}   # 创建 dir/a dir/b dir/c

# 生信示例
for f in *.fastq.gz; do
    echo "Processing $f"
    fastp -i "$f" -o "clean_${f}"
done
```

12.2 任务控制：Ctrl+C / Ctrl+Z / jobs / fg / bg

快捷键/命令	功能	生信场景
`Ctrl+C`	**终止**当前前台命令	命令跑错了想立即停掉
`Ctrl+Z`	**暂停**当前命令，放入后台	正在跑的程序想先做别的事
`jobs`	列出当前所有后台/暂停的任务	查看有哪些暂停的任务
`fg %1`	将任务 1 调回**前台**	恢复查看暂停的程序输出
`bg %1`	让任务 1 在**后台**继续运行	暂停的程序放到后台继续跑

```
# 实战演示

# 场景：正在跑一个比对，但想先看看别的文件
STAR --runThreadN 8 ...
# 按 Ctrl+Z 暂停
# 输出：[1]+  Stopped   STAR --runThreadN 8 ...

jobs
# 输出：[1]+  Stopped   STAR --runThreadN 8 ...

bg %1                    # 让 STAR 在后台继续跑
# 输出：[1]+ STAR --runThreadN 8 ... &

# 现在可以继续在终端做其他事
ls /data/results/

# 想看 STAR 的输出？调回前台
fg %1

# 想终止？调回前台后按 Ctrl+C
```

12.3 其他高频小技巧

```
# Tab 键自动补全
# 输入 cd /da 然后按 Tab → 自动补全为 cd /data/
# 输入文件名前几个字母按 Tab → 自动补全文件名

# 上下箭头：浏览命令历史
# 也可以用 history 命令查看
history | tail -20       # 看最近 20 条命令

# Ctrl+R：反向搜索历史命令
# 按 Ctrl+R 然后输入关键词，快速找到之前用过的命令

# !!：重复上一条命令
sudo !!                  # 用 sudo 重新运行上一条命令

# !$：上一条命令的最后一个参数
vim /data/genes.gtf      # 编辑文件
less !$                  # = less /data/genes.gtf

# 命令别名
alias ll='ls -lh'        # 之后用 ll 就等于 ls -lh
alias la='ls -lah'

# 查看磁盘空间
df -h                    # 查看各分区使用情况
du -sh /data/*           # 查看 /data 下各目录大小
du -sh * | sort -rh | head -10  # 找出最大的 10 个目录/文件
```

—— 第 12 章结束 ——

附录 B　命令速查卡 · 实战练习任务

B.1 Linux 命令速查卡（可打印）

生信 Linux 命令速查卡

文件与目录操作

`ls -lh [目录]`	列出文件（含大小、权限、日期）
`cd [路径]`	切换目录；cd ~ 回主目录；cd - 回上次目录
`pwd`	显示当前绝对路径
`mkdir -p [目录]`	创建目录（-p 递归创建）
`cp -r [源] [目标]`	复制文件/目录（-r 递归）
`mv [源] [目标]`	移动/重命名
`rm -rf [文件/目录]`	强制删除（不可恢复！谨慎使用）
`chmod +x [文件]`	添加执行权限

文件查看与搜索

`head -n [N] [文件]`	查看前 N 行
`tail -f [文件]`	实时查看文件末尾（看日志）
`less -S [文件]`	分页查看（-S 不换行）；q 退出
`grep "pattern" [文件]`	搜索文本；-c 计数；-v 反向；-w 全词
`wc -l [文件]`	统计行数
`zcat [文件.gz]`	查看 gzip 压缩文件内容

文本处理

`sort -k[N],[N]n [文件]`	按第 N 列数值排序
`uniq -c`	去重并计数（需先 sort）
`cut -f[N] [文件]`	提取第 N 列（Tab 分隔）

下载与压缩

`wget -c [URL]`	下载文件（-c 断点续传）
`curl -L -o [文件] [URL]`	下载文件（-L 跟随重定向）
`tar -zxvf [文件.tar.gz]`	解压 .tar.gz
`tar -zcvf [输出.tar.gz] [目录]`	压缩为 .tar.gz

进程与会话

`top / htop`	查看系统资源与进程
`screen -S [名称]`	创建会话；Ctrl+A D 脱离；-r 恢复
`nohup [命令] >log 2>&1 &`	后台运行，断开连接不中断

`jobs / fg / bg`	查看/前台/后台任务

conda / mamba

`conda create -n [名] python=3.10`	创建环境
`conda activate [名]`	激活环境
`conda install -c bioconda [包]`	安装生信软件
`conda env export > env.yml`	导出环境配置

SSH 与传输

`ssh [用户]@[地址]`	远程连接服务器
`scp -r [源] [目标]`	远程复制文件/目录
`rsync -avP [源] [目标]`	增量同步（支持断点续传）

管道与重定向

`命令A | 命令B`	管道：A 的输出作为 B 的输入
`> [文件]`	覆盖写入文件
`>> [文件]`	追加写入文件
`2>&1`	标准错误合并到标准输出

B.2 五个实战练习任务

练习 1

FASTQ 文件探索

任务：假设你有一个 FASTQ 文件 `sample.fastq.gz`（压缩状态），请完成以下操作：

1. 不解压，查看前 8 行（即前 2 条 read）
2. 统计该文件有多少条 read（提示：行数 ÷ 4）
3. 查看第一条 read 的 ID 和质量值首字母

参考答案：

```
# 1. 查看前 8 行
zcat sample.fastq.gz | head -8

# 2. 统计 read 数
echo $(( $(zcat sample.fastq.gz | wc -l) / 4 ))

# 3. 第一条 read 的 ID（第 1 行）和质量值首字母（第 4 行第 1 个字符）
zcat sample.fastq.gz | head -1           # read ID
zcat sample.fastq.gz | sed -n '4p' | cut -c1  # 质量值首字母
```

练习 2

GTF 注释文件分析

任务：给定一个 GTF 文件 `genes.gtf`，请完成：

1. 统计该文件中有多少种特征类型（第 3 列），各有多少个
2. 提取 chr1 上所有 exon 的坐标信息（第 1、4、5 列），保存为 BED 格式
3. 找出 BRCA1 基因的所有转录本 ID

参考答案：

```
# 1. 统计特征类型
cut -f3 genes.gtf | sort | uniq -c

# 2. 提取 chr1 exon → BED 格式（注意坐标转换：GTF 1-based → BED 0-based）
grep -w "chr1" genes.gtf | grep -w "exon" | \
  awk -F'\t' '{print $1"\t"$4-1"\t"$5}' > chr1_exons.bed

# 3. 找 BRCA1 的转录本 ID
grep "BRCA1" genes.gtf | grep "transcript" | \
  grep -o 'transcript_id "[^"]*"' | sort -u
```

练习 3

conda 环境搭建

任务：创建一个 RNA-seq 分析环境并安装核心工具：

1. 创建名为 `rnaseq` 的 conda 环境，Python 版本 3.10
2. 安装以下软件：fastp（质控）、STAR（比对）、samtools（BAM 处理）、subread（定量）
3. 验证安装：运行 `STAR --version` 和 `samtools --version`
4. 导出环境配置为 `rnaseq.yml`

参考答案：

```
# 1. 创建环境
conda create -n rnaseq python=3.10 -y

# 2. 激活并安装软件
conda activate rnaseq
mamba install -c bioconda fastp star samtools subread -y

# 3. 验证安装
STAR --version
samtools --version

# 4. 导出环境
conda env export > rnaseq.yml
```

练习 4

差异表达结果筛选

任务：给定一个差异表达结果表 `DE_results.tsv`（Tab 分隔，含表头），格式为：

```
gene_id    baseMean    log2FoldChange    pvalue    padj
```

请完成以下筛选：

1. 筛选 padj < 0.05 且 |log2FC| ≥ 1 的差异基因，保存为 `DEG.tsv`
2. 分别统计上调（log2FC > 0）和下调（log2FC < 0）的基因数
3. 找出变化幅度最大的前 10 个基因（按 |log2FC| 排序）

参考答案：

```
# 1. 筛选差异基因
awk -F'\t' 'NR>1 && $5 < 0.05 && ($3 >= 1 || $3 <= -1)' DE_results.tsv >
DEG.tsv

# 2. 统计上调/下调基因数
echo "上调基因数: $(awk -F'\t' '$3 > 0' DEG.tsv | wc -l)"
echo "下调基因数: $(awk -F'\t' '$3 < 0' DEG.tsv | wc -l)"

# 3. 变化幅度最大的前 10 个基因
awk -F'\t' '{print $1"\t"$3"\t"($3>=0?$3:-$3)}' DEG.tsv | \
  sort -k3,3nr | head -10 | cut -f1,2
```

练习 5

批量处理与后台运行

任务：你有 10 个 FASTQ 文件（`sample_1.fq.gz` 到 `sample_10.fq.gz`），需要对每个文件运行 fastp 质控。

1. 写一个 shell 脚本批量处理所有文件
2. 用 `nohup` 在后台运行该脚本
3. 实时查看运行日志

参考答案：

```
# 1. 创建脚本 run_fastp.sh
cat > run_fastp.sh << 'EOF'
#!/bin/bash
for i in $(seq 1 10); do
    echo "=== Processing sample_$i ==="
    fastp -i sample_${i}.fq.gz \
          -o clean_sample_${i}.fq.gz \
          --html sample_${i}.html \
          --json sample_${i}.json
    echo "=== Done sample_$i ==="
done
echo "All samples processed!"
EOF

# 2. 赋予执行权限并后台运行
chmod +x run_fastp.sh
nohup ./run_fastp.sh > fastp.log 2>&1 &

# 3. 实时查看日志
tail -f fastp.log

# 检查进程是否还在运行
jobs -l
# 或
ps aux | grep run_fastp
```

生物信息学入门教程

从分子生物学到 Linux 实战 · 面向零基础学习者

—— 全书完 ——