BIOF3 组学数据分析

03 质量控制、聚类与细胞类型注释

导出日期：2026年5月12日

03 质量控制、聚类与细胞类型注释

拿到 Cell Ranger（或同类工具）输出的表达矩阵，接下来要做的事情大致是一条固定的流水线：过滤掉质量差的细胞和基因、把深度差异归一化掉、挑选有信息量的基因、降维、聚类、给每个 cluster 贴上细胞类型标签、找差异基因。这条流水线是所有下游分析（整合、轨迹、通讯等）的起点，走通一次之后，后面所有模块都基于它继续。

本节以 Seurat 为主线走一遍这条流程，同时给出 Scanpy 的等价写法。数据用 01 章里介绍的 PBMC 3k 公开数据，跑完之后你会得到一张 UMAP 图，上面的每个细胞都被标了 CD4 T、CD8 T、B、NK、monocyte 等注释。

流程概览

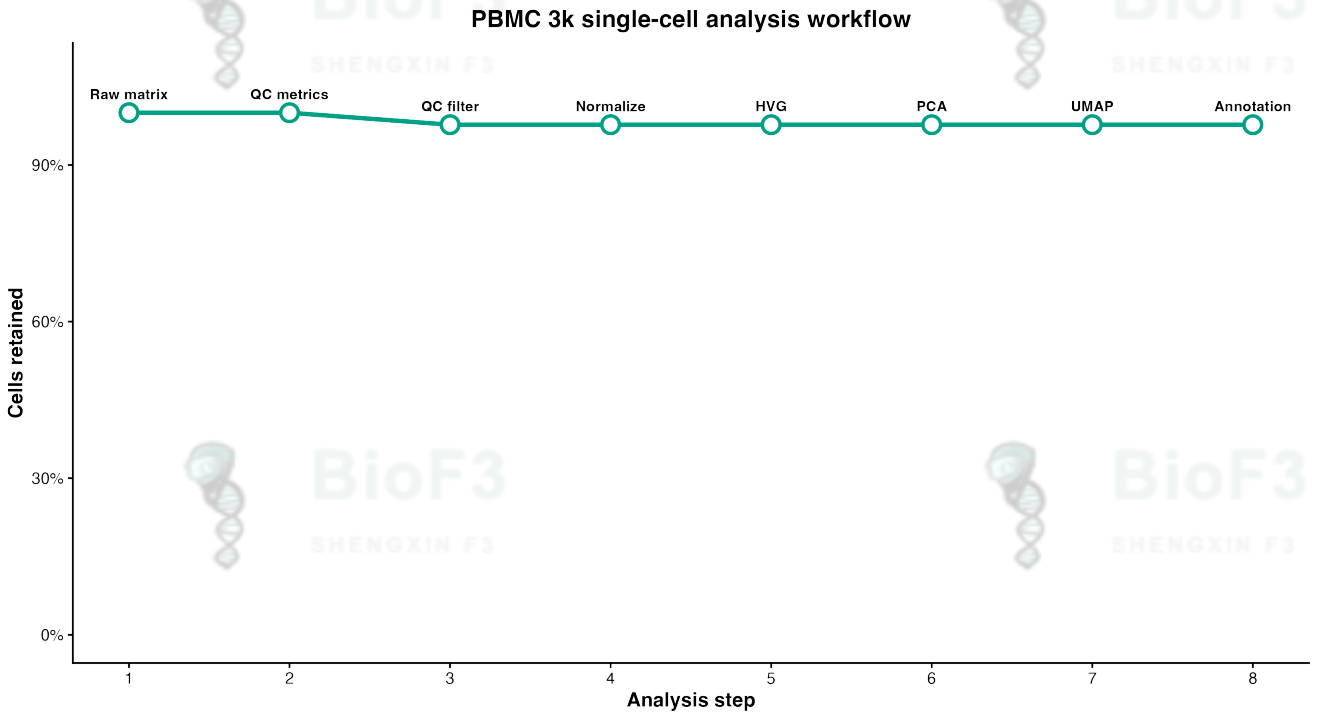

图 1：单细胞分析完整流程。展示了从原始数据到细胞类型注释的 8 个主要步骤及细胞保留率。

```
表达矩阵
  ↓ 质控：按基因数、UMI 数、线粒体比例过滤细胞
  ↓ 标准化：消除测序深度差异（LogNormalize 或 SCTransform）
  ↓ 高变基因：选最有信息量的 ~2000 个基因
  ↓ 缩放：把这些基因归一化到均值 0 方差 1
  ↓ PCA：把 2000 维降到 ~30 维
  ↓ 邻居图 + 聚类：找到 cluster
  ↓ UMAP：把 ~30 维压成 2D 可视化
  ↓ 差异基因：每个 cluster vs 其他的 marker
  ↓ 细胞类型注释：marker 对照或 SingleR 自动
```

环境准备

只需要跑一次：

```r
install.packages(c("Seurat", "dplyr", "ggplot2"))

if (!require("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install(c("SingleR", "celldex", "clusterProfiler"))
```

Python 端：

```bash
pip install scanpy python-igraph leidenalg
```

读入数据

假设 `filtered_feature_bc_matrix/` 已经存在（01 章和 02 章的产物）。

```r
library(Seurat)
library(dplyr)
library(ggplot2)

data <- Read10X(data.dir = "path/to/filtered_feature_bc_matrix/")

pbmc <- CreateSeuratObject(
  counts       = data,
  project      = "PBMC3k",
  min.cells    = 3,     # 基因至少在 3 个细胞里表达
  min.features = 200    # 细胞至少检测到 200 个基因
)
pbmc
```

Scanpy 的等价写法：

```python
import scanpy as sc
adata = sc.read_10x_mtx("path/to/filtered_feature_bc_matrix/", var_names="gene_symbols")
sc.pp.filter_cells(adata, min_genes=200)
sc.pp.filter_genes(adata, min_cells=3)
```

Seurat 的 `min.cells` 和 `min.features` 是"创建对象时就先丢一批明显垃圾的 barcode 和 gene"，不用来做最终的 QC 过滤。

质量控制

最常看的三项 QC 指标：

指标	Seurat 字段	Scanpy 字段	合理范围	异常提示
每细胞基因数	nFeature_RNA	n_genes_by_counts	200 – 6,000	过低 → 空液滴/破损；过高 → 双细胞
每细胞 UMI 数	nCount_RNA	total_counts	500 – 50,000	过低 → 测序不足；过高 → 双细胞
线粒体基因比例	percent.mt	pct_counts_mt	< 5 – 10%	过高 → 细胞破损 / 应激

具体阈值要看这份数据自己的分布，不是死背上面的数字。一般做法是先算指标 → 画小提琴图看分布 → 按分布定阈值。

计算指标

```r
pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")
# 可选：核糖体蛋白基因也常用
pbmc[["percent.rb"]] <- PercentageFeatureSet(pbmc, pattern = "^RP[SL]")

head(pbmc@meta.data)
```

```python
adata.var["mt"] = adata.var_names.str.startswith("MT-")
sc.pp.calculate_qc_metrics(
    adata, qc_vars=["mt"], percent_top=None, log1p=False, inplace=True,
)
```

看分布

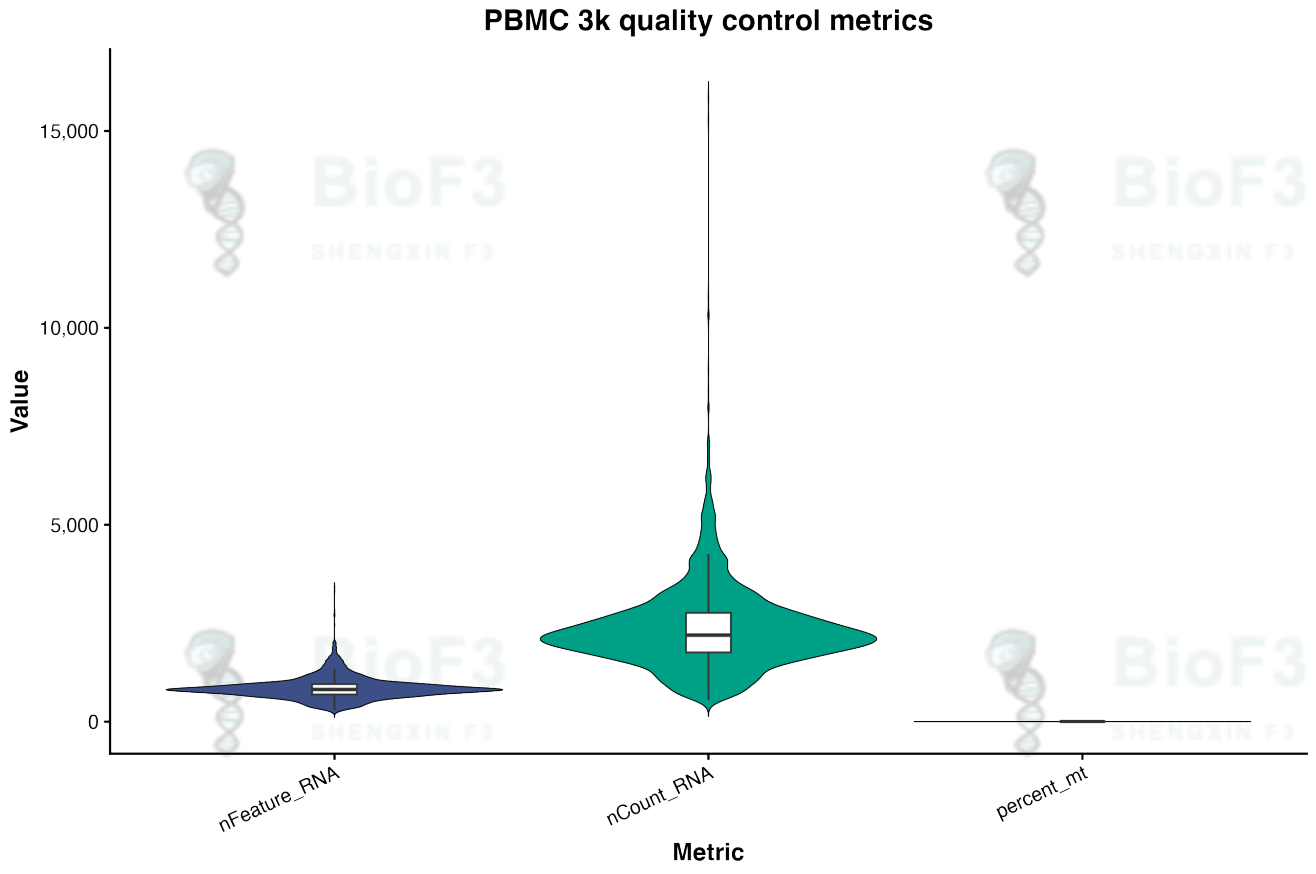

图 2：质量控制指标小提琴图。展示了基因数、UMI 数和线粒体基因比例的分布情况。

```r
VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

# 两两散点：UMI vs 线粒体、UMI vs 基因数
FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "percent.mt") +
  FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```

散点图比小提琴更有用：两个尖峰不代表双模态，但右上角偏离主分布的点通常是双细胞。

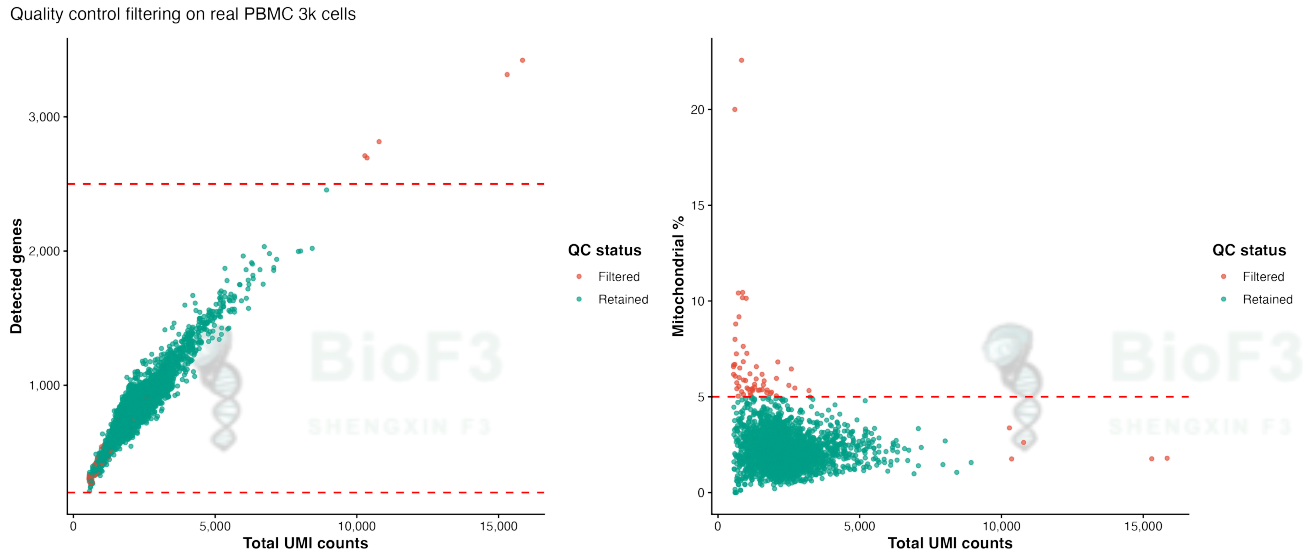

图 3：质量控制过滤散点图。左图展示 UMI 数与基因数的关系，右图展示 UMI 数与线粒体基因比例的关系。红色虚线表示过滤阈值，绿色点为保留的细胞，红色点为过滤的细胞。

过滤

```r
pbmc <- subset(pbmc,
  subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5
)
pbmc
```

```python
sc.pp.filter_cells(adata, max_genes=2500)
adata = adata[adata.obs.pct_counts_mt < 5, :]
```

标准化

不同细胞的测序深度差距能有十倍以上。直接比较 raw counts 会把"深度不同"当成"生物学差异"。标准化的目标是让每个细胞的表达量在同一刻度上。

最常见有两种选择：

- **LogNormalize**：每个细胞归一化到相同总 UMI（默认 10000），再 `log1p` 。快、适合教学。
- **SCTransform**：负二项回归建模，同时归一化和去除技术协变量。效果更好，但慢。

PBMC 教学数据用 LogNormalize 就够；真实项目里 SCTransform 更稳。

```r
# LogNormalize 路线
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)

# 或 SCTransform 路线（可选替代，如果用这个就跳过后面的 ScaleData）
# pbmc <- SCTransform(pbmc, vars.to.regress = "percent.mt", verbose = FALSE)
```

```python
sc.pp.normalize_total(adata, target_sum=1e4)
sc.pp.log1p(adata)
adata.raw = adata   # 后面做 FindMarkers 时用得到原始归一化数据
```

特征选择：找高变基因

一个矩阵有两万多个基因，但大多数基因在所有细胞里都差不多。聚类前先筛出 ~2000 个"在细胞间差异最大"的基因（highly variable genes），既减少计算量又降低技术噪声影响。

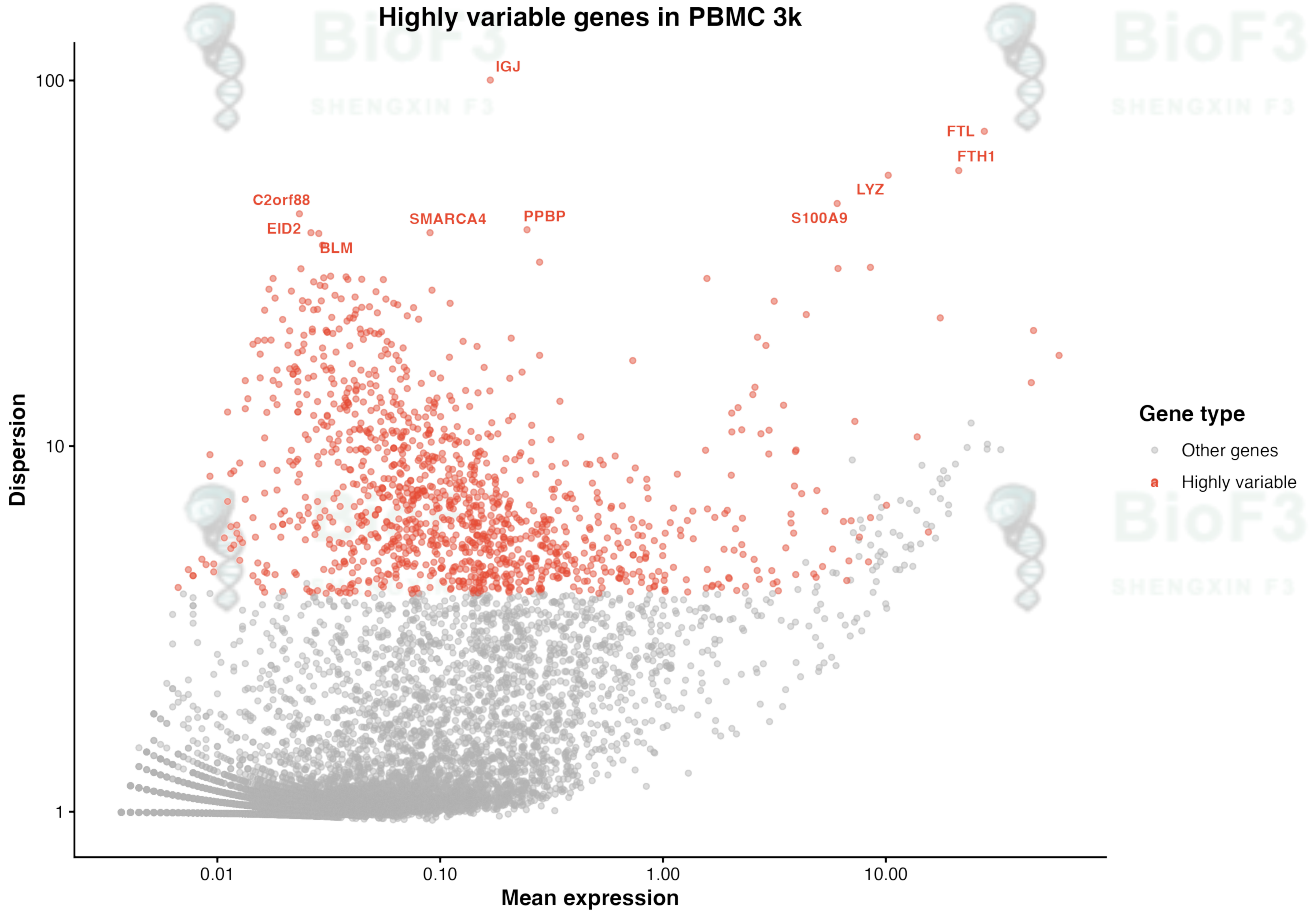

图 4：高变异基因选择。展示了基因的平均表达量与离散度的关系，红色点为高变异基因，标注了前 10 个高变异基因。

```r
pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)

top10 <- head(VariableFeatures(pbmc), 10)
LabelPoints(plot = VariableFeaturePlot(pbmc), points = top10, repel = TRUE)
```

```
sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3, min_disp=0.5)
sc.pl.highly_variable_genes(adata)
adata = adata[:, adata.var.highly_variable]
```

缩放

PCA 前把每个基因归一化到均值 0 方差 1，避免高表达基因主导主成分。`vars.to.regress` 可以在这一步同时把线粒体比例、UMI 数等协变量 regress out：

```
pbmc <- ScaleData(pbmc)   # 默认只缩放高变基因
```

```
sc.pp.regress_out(adata, ["total_counts", "pct_counts_mt"])
sc.pp.scale(adata, max_value=10)
```

PCA 降维

```
pbmc <- RunPCA(pbmc, features = VariableFeatures(pbmc), npcs = 50)

# 打印前几个 PC 上 loading 最高的基因
print(pbmc[["pca"]], dims = 1:5, nfeatures = 5)
```

```
sc.tl.pca(adata, svd_solver="arpack")
```

选多少个 PC

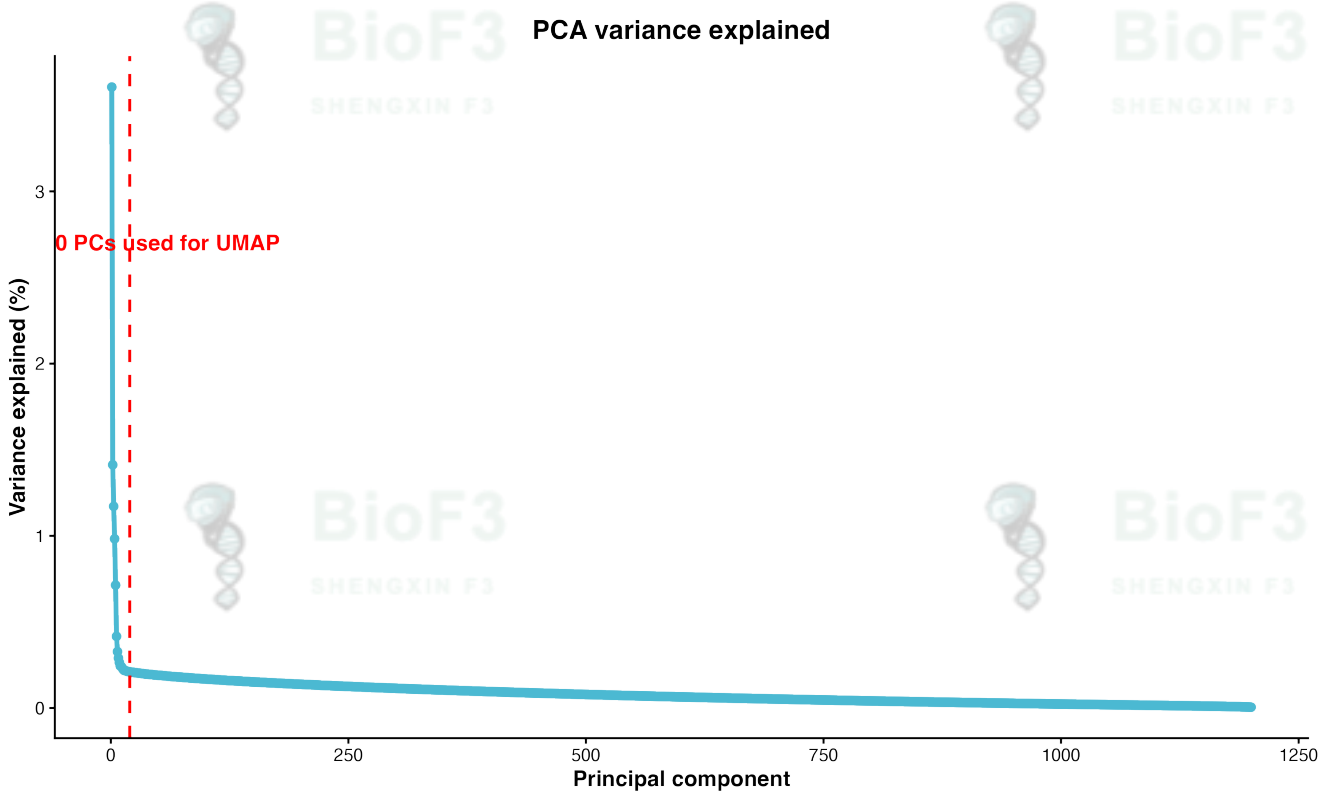

图 5：PCA 方差解释图（Elbow Plot）。展示了每个主成分解释的方差比例，红色虚线标注了选择的 PC 数量（20 个）。

看 elbow plot：方差解释率下降突然变平的位置，就是合理的 PC 数。PBMC 通常选 10-30，不用纠结具体选 15 还是 20，结果差别不大。

```
ElbowPlot(pbmc, ndims = 50)
```

```
sc.pl.pca_variance_ratio(adata, n_pcs=50)
```

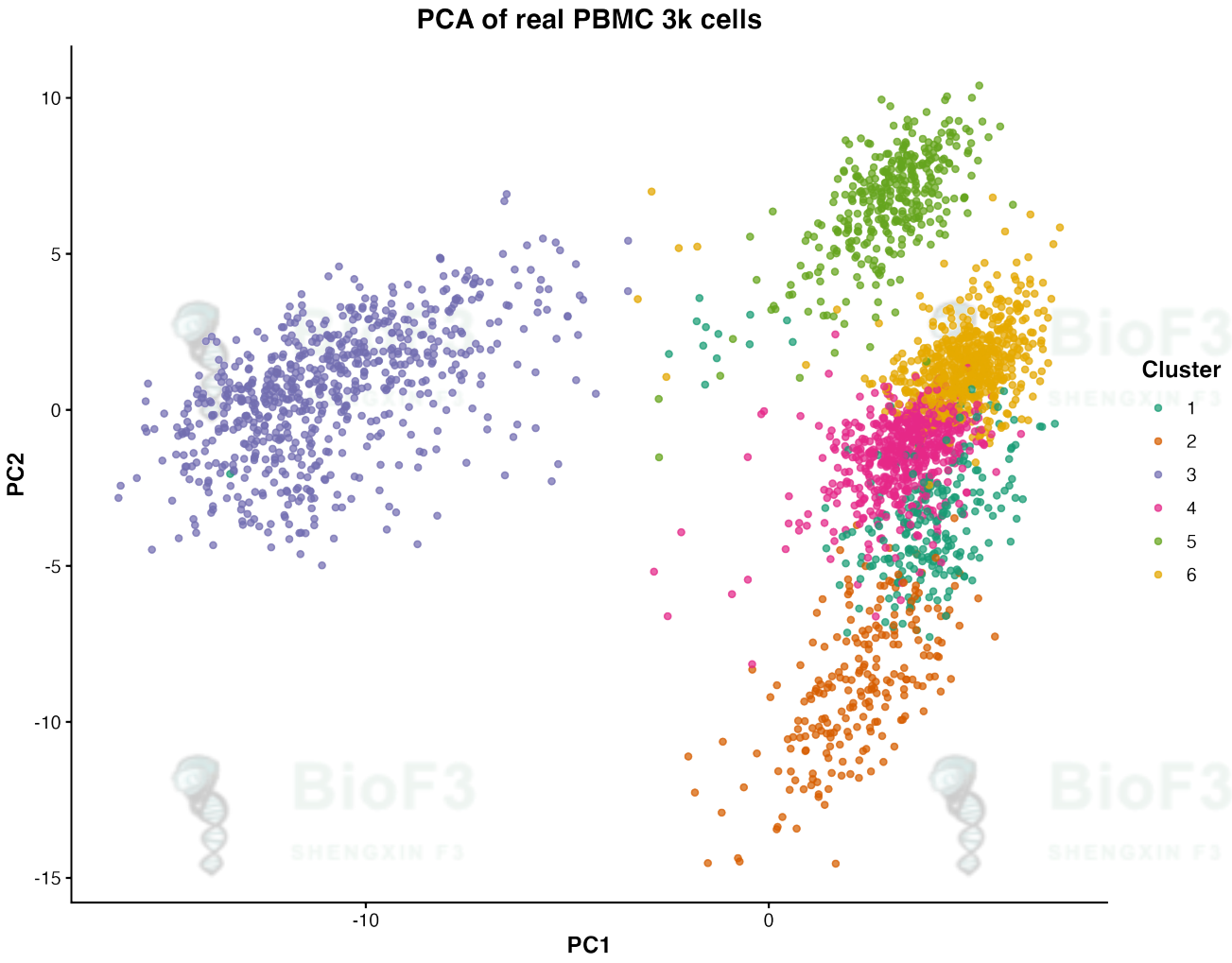

图 6：PCA 散点图。展示了前两个主成分（PC1 和 PC2）的细胞分布，不同颜色代表不同的聚类。

聚类

用选好的 PC 构建 KNN 图，再在图上做 Louvain/Leiden 聚类。`resolution` 是唯一要调的参数：值越大 cluster 越多。

```
pbmc <- FindNeighbors(pbmc, dims = 1:20)
pbmc <- FindClusters(pbmc, resolution = 0.5)
```

```
sc.pp.neighbors(adata, n_neighbors=10, n_pcs=40)
sc.tl.leiden(adata, resolution=0.5)
```

`resolution` 没有绝对正确值。一般从 0.5 开始，结合后面 marker gene 的合理性和生物学预期微调。PBMC 3k 在 resolution=0.5 下通常分出 8-9 个 cluster，对应主要免疫细胞类型。

UMAP 可视化

UMAP 本身不影响聚类结果（聚类是在 PC 空间做的），只是把高维结构投影到 2D 方便人看。

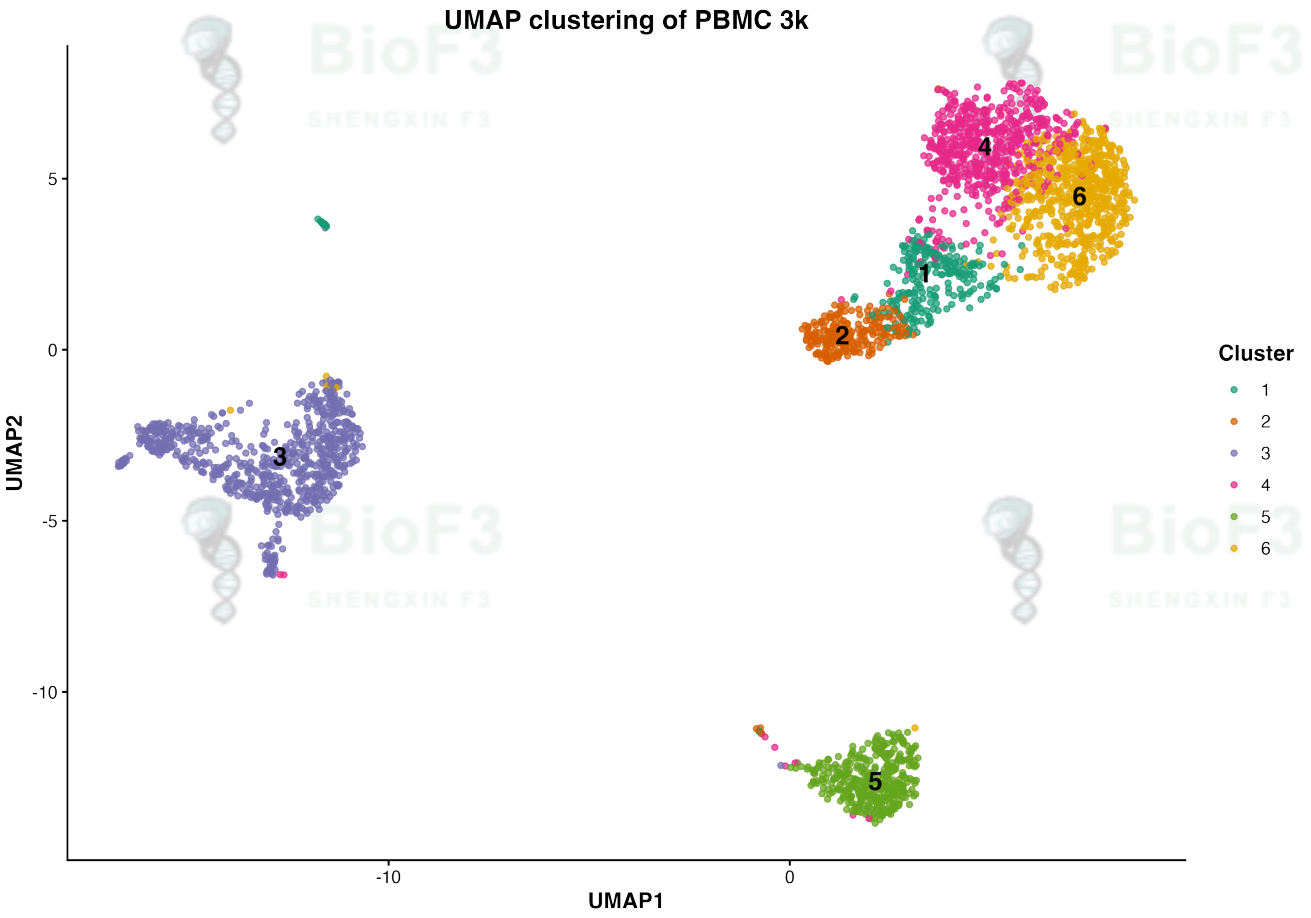

图 7：UMAP 聚类可视化。展示了 6 个细胞簇在 UMAP 空间中的分布，数字标注了簇的编号。

```r
pbmc <- RunUMAP(pbmc, dims = 1:20)
DimPlot(pbmc, reduction = "umap", label = TRUE)

# 用 marker 基因上色验证聚类
FeaturePlot(pbmc, features = c("MS4A1", "CD79A", "CD3D", "CD8A"))
```

```python
sc.tl.umap(adata)
sc.pl.umap(adata, color=["leiden", "CST3", "NKG7"])
```

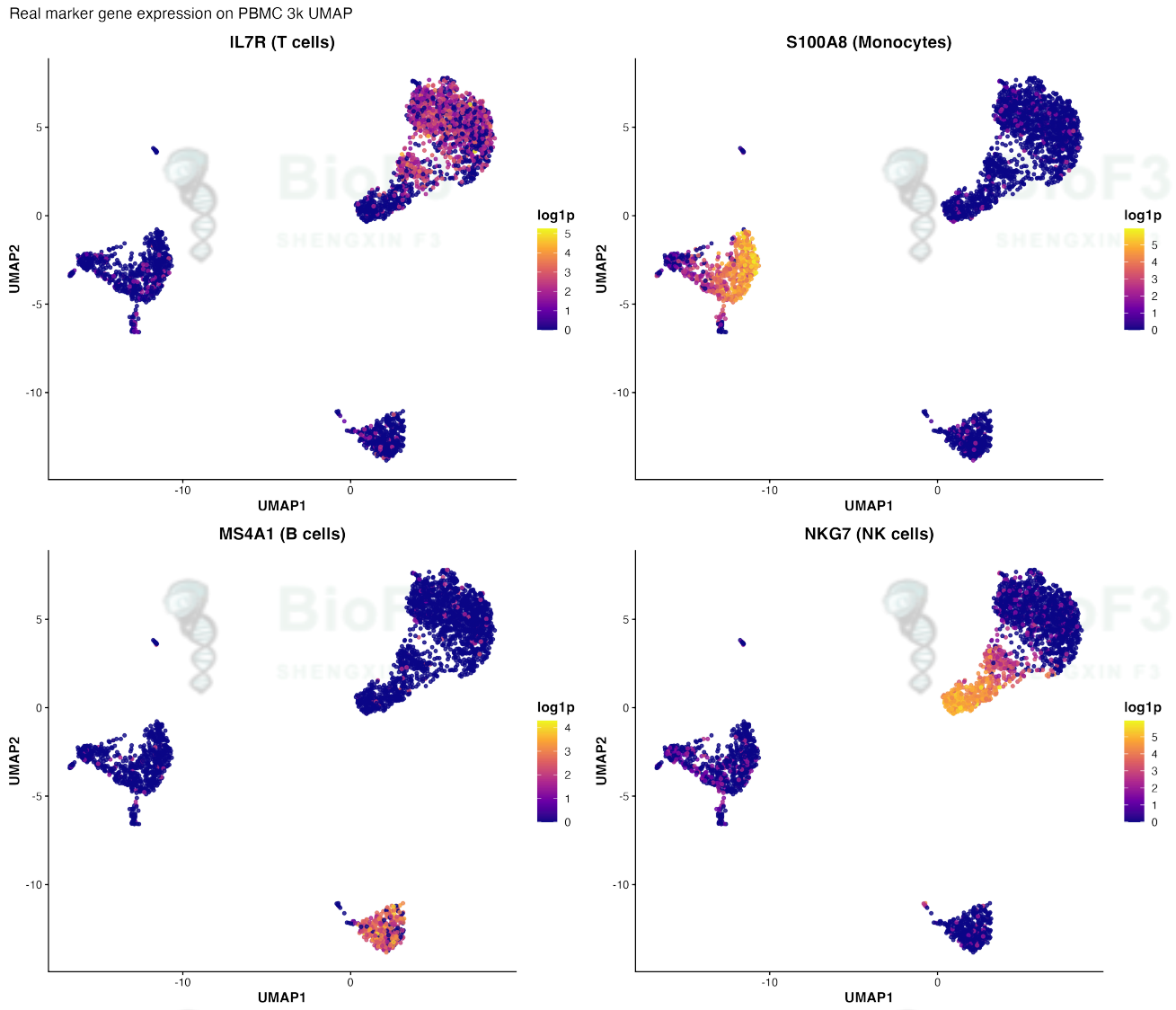

图 8：UMAP 上的标志基因表达。展示了 CD3D（T 细胞）、CD14（单核细胞）、MS4A1（B 细胞）和 NKG7（NK 细胞）的表达模式。

把 UMAP 染上聚类编号，再把几个已知 marker 画在同样的坐标上——如果 CD3D 的高表达区域恰好覆盖某个 cluster，这就是该 cluster 是 T 细胞的证据。

找 marker 基因

每个 cluster 和"其他 cluster 总和"做差异表达，就能得到这个 cluster 的 marker：

```
pbmc.markers <- FindAllMarkers(
  pbmc,
  only.pos        = TRUE,
  min.pct         = 0.25,
  logfc.threshold = 0.25
)

# 每个 cluster 前 5 个 marker
pbmc.markers %>% group_by(cluster) %>% slice_max(n = 5, order_by = avg_log2FC)
```

```
sc.tl.rank_genes_groups(adata, "leiden", method="wilcoxon")
sc.pl.rank_genes_groups(adata, n_genes=25, sharey=False)
```

热图把 top marker 画在一起看：

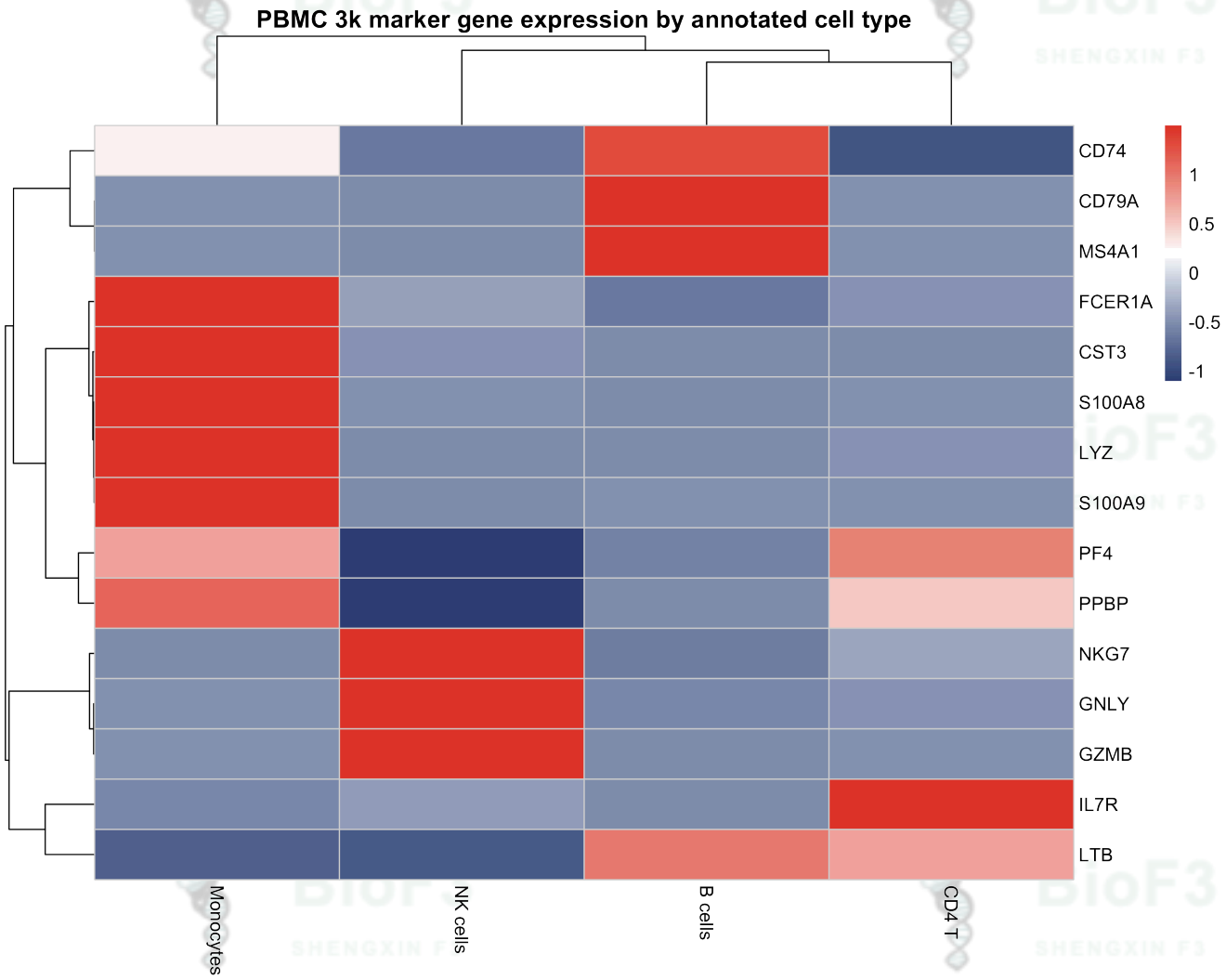

图 9：标志基因表达热图。展示了不同细胞类型的特征标志基因表达模式，每列代表一个细胞簇。

```
top10 <- pbmc.markers %>% group_by(cluster) %>% top_n(10, avg_log2FC)
DoHeatmap(pbmc, features = top10$gene) + NoLegend()
```

细胞类型注释

手动：按经典 marker 对照

PBMC 里常用的一组 marker：

细胞类型	经典 marker
CD4+ T	IL7R, CD4, CCR7（naive）/S100A4（memory）
CD8+ T	CD8A, CD8B
B	MS4A1 (CD20), CD79A
NK	GNLY, NKG7
单核 CD14	CD14, LYZ
单核 FCGR3A	FCGR3A, MS4A7
树突	FCER1A, CST3
巨核 / 血小板	PPBP

对照 cluster 的 top marker 决定怎么命名：

```
new_ids <- c(
  "Naive CD4 T", "CD14+ Mono", "Memory CD4 T", "B",
  "CD8 T", "FCGR3A+ Mono", "NK", "DC", "Platelet"
)
names(new_ids) <- levels(pbmc)
pbmc <- RenameIdents(pbmc, new_ids)

DimPlot(pbmc, reduction = "umap", label = TRUE, pt.size = 0.5) + NoLegend()
```

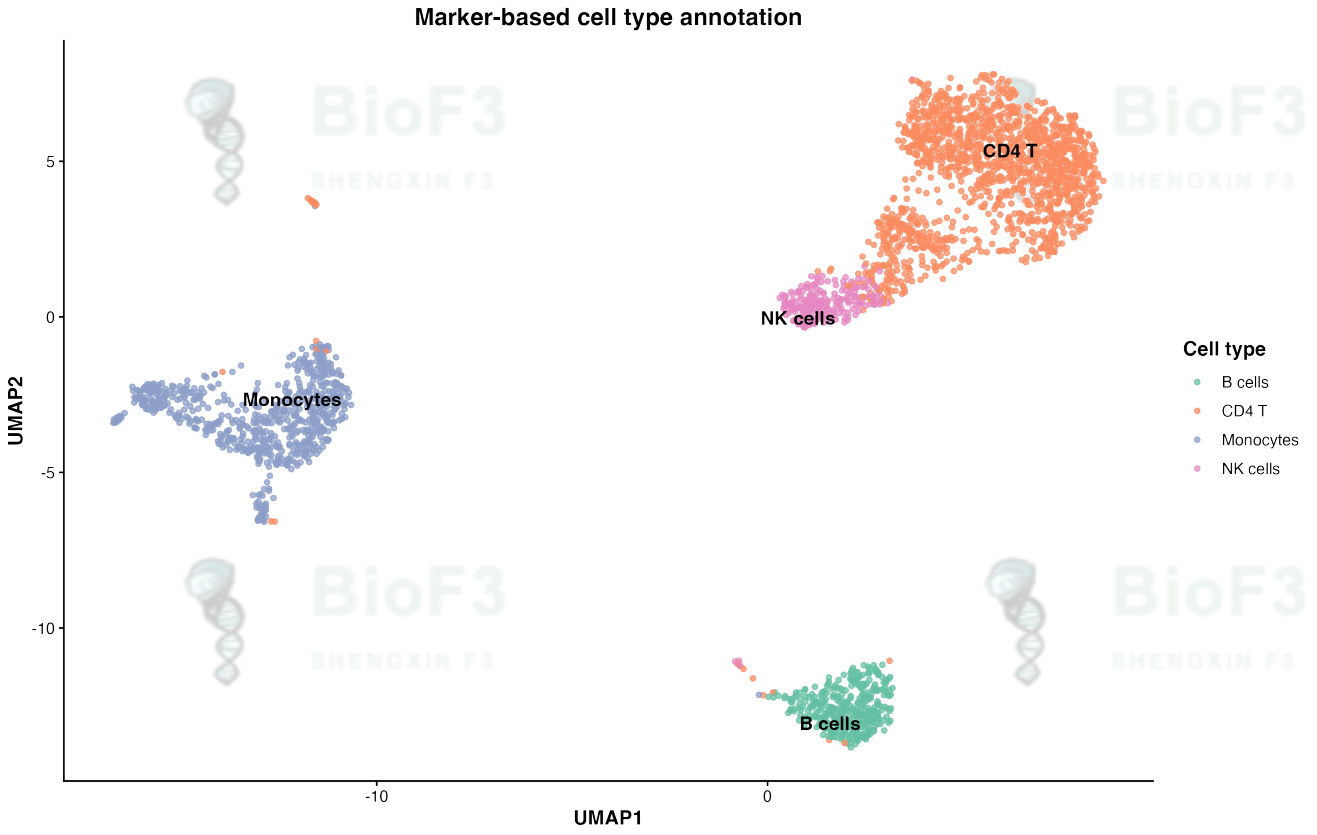

图 10：细胞类型注释 UMAP 图。展示了 6 种主要细胞类型在 UMAP 空间中的分布。

自动：SingleR

`SingleR` 拿一份参考数据集做最近邻注释，适合快速拿个初稿：

```
library(SingleR)
library(celldex)

ref  <- celldex::HumanPrimaryCellAtlasData()
sce  <- as.SingleCellExperiment(pbmc)
pred <- SingleR(test = sce, ref = ref, labels = ref$label.main)

pbmc$singler <- pred$labels
DimPlot(pbmc, reduction = "umap", group.by = "singler")
```

自动注释适合做第一轮筛查，但最终 cluster 命名还是得回到 marker gene + 人工判断。

差异表达：两组比较

注释之后，常见需求是对两群细胞做差异分析（比如 CD4 T vs CD8 T）：

```
cd4_vs_cd8 <- FindMarkers(pbmc, ident.1 = "CD4 T", ident.2 = "CD8 T")
head(cd4_vs_cd8)
```

```
sc.tl.rank_genes_groups(
    adata, "cell_type", groups=["CD4 T"], reference="CD8 T", method="wilcoxon"
)
sc.pl.rank_genes_groups(adata)
```

功能富集

把某个 cluster 的 marker 做 GO/KEGG 富集，看它主要落在什么通路：

```r
library(clusterProfiler)
library(org.Hs.eg.db)

cluster0_genes <- pbmc.markers %>%
  filter(cluster == 0, p_val_adj < 0.05) %>% pull(gene)

gene_ids <- bitr(cluster0_genes,
  fromType = "SYMBOL", toType = "ENTREZID", OrgDb = org.Hs.eg.db
)

ego <- enrichGO(
  gene     = gene_ids$ENTREZID,
  OrgDb    = org.Hs.eg.db,
  ont      = "BP",
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05
)
dotplot(ego, showCategory = 10)

kk <- enrichKEGG(gene = gene_ids$ENTREZID, organism = "hsa")
dotplot(kk, showCategory = 10)
```

保存结果

```r
saveRDS(pbmc, "pbmc_analyzed.rds")
write.csv(pbmc@meta.data,                "metadata.csv")
write.csv(pbmc.markers,                   "marker_genes.csv")
```

```python
adata.write("pbmc_analyzed.h5ad")
```

分析脚本和输出一起提交到版本控制，下次复现或多样本整合时能直接用。

下载资源

module04_complete_sci.R
16 KB

下载图表生成脚本 ↗

下一步

继续学习：04 多样本数据整合

参考资源

- Seurat PBMC 3k 教程
- Scanpy 教程
- 单细胞最佳实践

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。