Lab 7: ChIP-seq and proteomic analysis

Chinese Name:	麦伟江
Student ID:	202330701011
Class:	生物技术
Teacher:	Jianling Xie
Date:	
Lab:	Bioinformatics Lab (B6-142)
Score:	

Procedures of finishing your lab work:

1. Please provide information above.
2. Please provide your answers on page 3 onwards.
3. Submit your report as a pdf file on VAMS (维普作业管理系统); OR print the file (双面打印) and submit it **on your next class**.

Lab 7 content

First hint: It is strongly suggested that you should go through everything we have learned during the lecture on your PC first before jumping into the following task. It looks easy but it isn't!

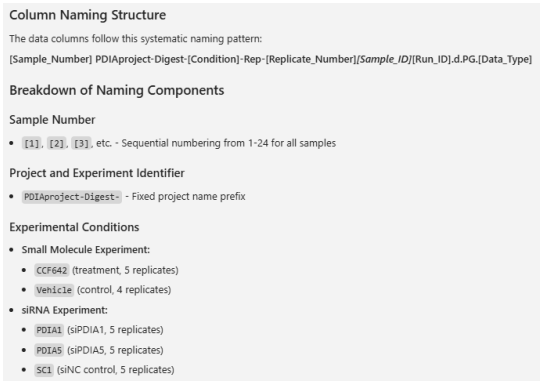
1) Please download and read PMID: 41086208 and analyse its transcriptomic and proteomics data*, and then:

a) For the proteomics data only, please do (a) dimentionality reduction figure(s) (e.g., PCA or UMAP), and according to the sample distribution pattern on the figure please determine the quality of this proteomic experiment. (2 points)

Based on different treatment operations, we need to perform two sets of dimensionality reduction analyses. The first set is the drug treatment group, including the CCF642 treatment group and the vehicle treatment group. The second set adopts genetic knockout technology, involving siPDIA1, siPDIA5, and siNC (siNC refers to siRNA Negative Control), among which siNC as the control group is a random sequence fragment that ensures no target sequences are knocked out.

In accordance with the guidelines of the article, we retrieved the proteomics data from the ProteomeXchange Consortium platform (ftp://ftp.pride.ebi.ac.uk/pride/data/archive/2025/10/PXD057643/Spectronaut_Output-Result.xls). Although the file extension is displayed as xls, verification revealed that it is actually a tsv file; thus, we directly modified the file extension to tsv.

After reading the file and analyzing its data structure, we found that it directly contains the raw signal values of proteins and the $\log_2$-transformed signal values. Meanwhile, we confirmed the corresponding relationships as follows: Vehicle represents the vehicle control group, and SC (Scrambled Control) denotes the scrambled control RNA.



Column Naming Structure

The data columns follow this systematic naming pattern:

[Sample_Number] PDIAproject-Digest-[Condition]-Rep-[Replicate_Number]*[Sample_ID]*[Run_ID].d.PG.[Data_Type]

Breakdown of Naming Components

Sample Number

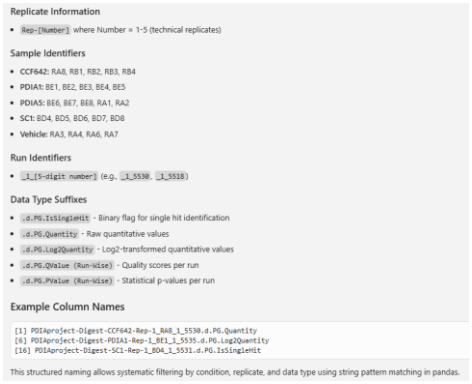
- `[1]`, `[2]`, `[3]`, etc. - Sequential numbering from 1-24 for all samples

Project and Experiment Identifier

- `PDIAproject-Digest` - Fixed project name prefix

Experimental Conditions

- **Small Molecule Experiment:**
  - `CCF642` (treatment, 5 replicates)
  - `Vehicle` (control, 4 replicates)
- **siRNA Experiment:**
  - `PDIA1` (siPDIA1, 5 replicates)
  - `PDIA5` (siPDIA5, 5 replicates)
  - `SC1` (siNC control, 5 replicates)



Replicate Information

- `Rep-[Number]` where Number = 1-5 (technical replicates)

Sample Identifiers

- **CCF642:** RA8, RB1, RB2, RB3, RB4
- **PDIA1:** BE1, BE2, BE3, BE4, BE5
- **PDIA5:** BE6, BE7, BE8, RA1, RA2
- **SC1:** BD4, BD5, BD6, BD7, BD8
- **Vehicle:** RA3, RA4, RA6, RA7

Run Identifiers

- `_1_[5-digit number]` (e.g., `_1_5530`, `_1_5518`)

Data Type Suffixes

- `.d.PG.IsSingleHit` - Binary flag for single hit identification
- `.d.PG.Quantity` - Raw quantitative values
- `.d.PG.Log2Quantity` - Log2-transformed quantitative values
- `.d.PG.QValue (Run-wise)` - Quality scores per run
- `.d.PG.PValue (Run-wise)` - Statistical p-values per run

Example Column Names

```
[1] PDIAproject-Digest-CCF642-Rep-1_RA8_1_5530.d.PG.Quantity
[6] PDIAproject-Digest-PDIA1-Rep-1_BE1_1_5535.d.PG.Log2Quantity
[16] PDIAproject-Digest-SC1-Rep-1_BD4_1_5531.d.PG.IsSingleHit
```

This structured naming allows systematic filtering by condition, replicate, and data type using string pattern matching in pandas.

In the raw data, the differences in expression levels of a small number of highly expressed proteins are far greater than those of most proteins. This causes partial sample clustering to be entirely dominated by these highly expressed proteins, failing to reflect the true biological differences between sample groups (e.g., control group vs. treatment group). After $\log_2$ transformation of the original signal values, the expression change ranges of all proteins are normalized to the same scale. The differences between samples are jointly determined by the expression patterns of global proteins, resulting in clearer PCA clustering and UMAP grouping results (more distinct group boundaries and a lower proportion of noisy samples). Therefore, we opted to use the $\log_2$-transformed data. On this basis, we formulated a targeted data reading protocol by carefully sorting out the data naming conventions.

Environment preparation: Import libraries:

```python
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
import os

# GSEA和ORA分析相关库
try:
    import gseapy as gp
    GSEA_AVAILABLE = True
    print("gseapy库可用，将执行GSEA和ORA分析")
except ImportError:
    print("警告：gseapy库未安装，GSEA和ORA分析将被跳过")
    print("安装命令：pip install gseapy")
    GSEA_AVAILABLE = False

# 设置matplotlib使用Arial字体
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = ['Arial']
plt.rcParams['axes.unicode_minus'] = False
```

Python

gseapy库可用，将执行GSEA和ORA分析

Load data, complete sample grouping, and perform data cleaning:

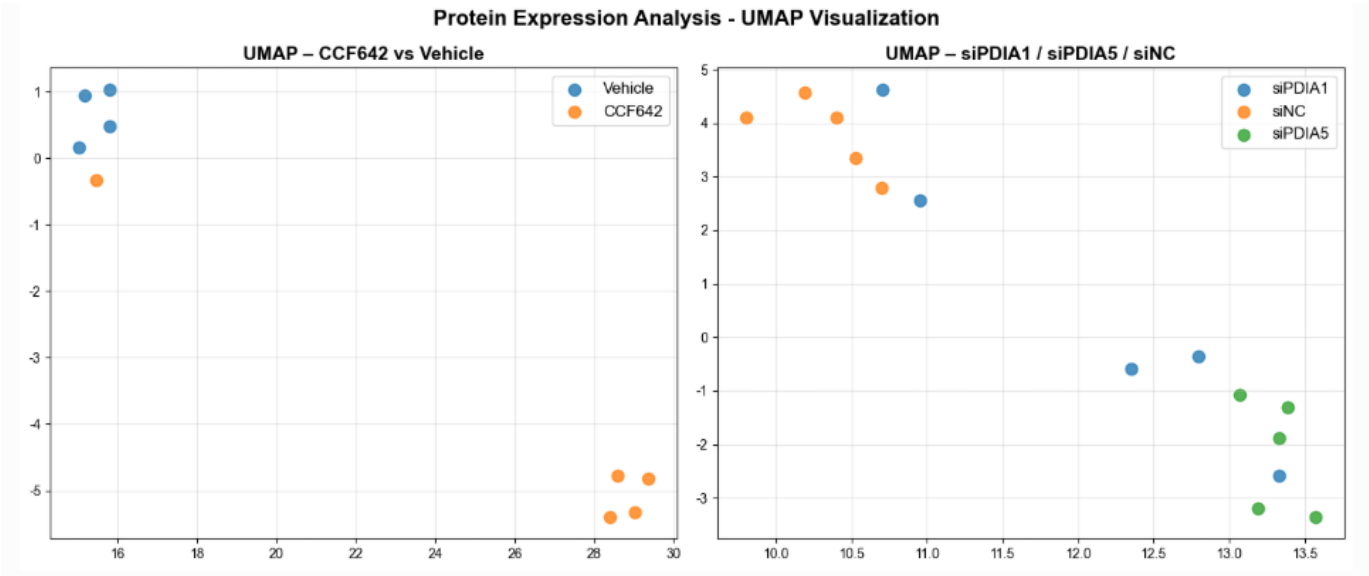
```python
print("\n" + "="*60)
print("Cell 0: 基础数据加载和分组")
print("="*60)
# 环境设置
import umap
from sklearn.preprocessing import StandardScaler
# 设置matplotlib使用Arial字体
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = ['Arial']
plt.rcParams['axes.unicode_minus'] = False

print("环境设置完成")
# 数据读取
try:
    df = pd.read_csv(
        r"data/Spectronaut_Output-Result.tsv",
        sep="\t"
    )
    print(f"数据读取成功，原始形状：{df.shape}")
except FileNotFoundError:
    print("数据文件未找到，请检查路径：data/Spectronaut_Output-Result.tsv")
    exit()
# 设置蛋白为行名
df = df.set_index("PG.ProteinGroups")
print(f"设置蛋白为行名后形状：{df.shape}")
# 只保留Log2Quantity列
log2_df = df.loc[:, df.columns.str.contains("PG.Log2Quantity")]
print(f"保留Log2Quantity列后形状：{log2_df.shape}")
print(f"Log2Quantity列数量：{len(log2_df.columns)}")
# 实验设计和数据分组
print("\n实验设计和数据分组：")
# 小分子实验：CCF642 vs Vehicle
small_molecule_cols = log2_df.columns[
    log2_df.columns.str.contains("CCF642|Vehicle")
]
small_molecule_df = log2_df[small_molecule_cols].T
# siRNA实验：PDIA1 / PDIA5 / SC1
sirna_cols = log2_df.columns[
    log2_df.columns.str.contains("PDIA1|PDIA5|SC1")
]
sirna_df = log2_df[sirna_cols].T
# 数据清理
small_molecule_df = small_molecule_df.dropna(axis=1, how="any")
sirna_df = sirna_df.dropna(axis=1, how="any")

print(f"小分子实验数据形状：{small_molecule_df.shape}")
print(f"siRNA实验数据形状：{sirna_df.shape}")
# 构造样本分组标签
labels_sm = [
    "CCF642" if "CCF642" in c else "Vehicle"
    for c in small_molecule_df.index
]
labels_si = []
for c in sirna_df.index:
    if "PDIA1" in c:
        labels_si.append("siPDIA1")
    elif "PDIA5" in c:
        labels_si.append("siPDIA5")
    else:
        labels_si.append("siNC")
print(f"小分子实验标签：{set(labels_sm)}")
print(f"siRNA实验标签：{set(labels_si)}")
print("\nCell 0 完成")
```

output：

```
============================================================
Cell 0: 基础数据加载和分组
============================================================
环境设置完成
数据读取成功，原始形状：(2150, 128)
设置蛋白为行名后形状：(2150, 127)
保留Log2Quantity列后形状：(2150, 24)
Log2Quantity列数量：24

实验设计和数据分组：
小分子实验数据形状：(9, 2098)
siRNA实验数据形状：(15, 2068)
小分子实验标签：{'Vehicle', 'CCF642'}
siRNA实验标签：{'siPDIA1', 'siNC', 'siPDIA5'}

Cell 0 完成
```

After confirming the data is error-free, perform UMAP dimensionality reduction analysis on it.

```
print("\n" + "="*60)
print("Cell 1: UMAP相关分析和可视化")
print("="*60)
# 设置UMAP reducer
reducer = umap.UMAP(
    n_components=2,
    n_neighbors=3,
    random_state=42
)
# UMAP降维
emb_sm = reducer.fit_transform(small_molecule_df)
emb_si = reducer.fit_transform(sirna_df)
print("UMAP降维完成")
print(f"小分子实验嵌入形状: {emb_sm.shape}")
print(f"siRNA实验嵌入形状: {emb_si.shape}")
# 创建UMAP可视化图
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
# 小分子实验可视化
for g in set(str)(labels_sm):
    idx = [i for i, x in enumerate(str)(labels_sm) if x == g]
    axes[0].scatter(
        emb_sm[idx, 0],
        emb_sm[idx, 1],
        label=g,
        s=80,
        alpha=0.8
    )
axes[0].set_title("UMAP - CCF642 vs Vehicle", fontsize=14, fontweight='bold')
axes[0].legend(fontsize=12)
axes[0].grid(True, alpha=0.3)
# siRNA实验可视化
for g in set[Any](labels_si):
    idx = [i for i, x in enumerate[Any](labels_si) if x == g]
    axes[1].scatter(
        emb_si[idx, 0],
        emb_si[idx, 1],
        label=g,
        s=80,
        alpha=0.8
    )
axes[1].set_title("UMAP - siPDIA1 / siPDIA5 / siNC", fontsize=14, fontweight='bold')
axes[1].legend(fontsize=12)
axes[1].grid(True, alpha=0.3)
plt.suptitle('Protein Expression Analysis - UMAP Visualization',
             fontsize=16, fontweight='bold', y=0.98)
plt.tight_layout()
plt.show()
```

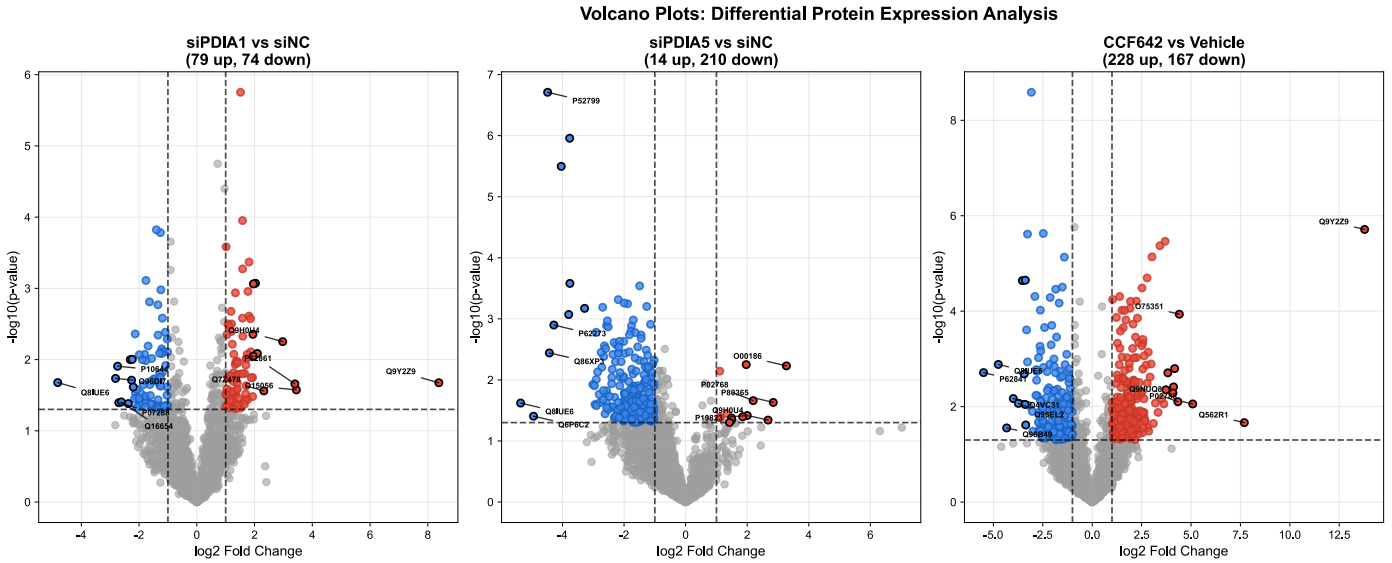


The separation effect between the CCF642 group and the vehicle group is extremely significant, indicating that CCF642 treatment induces remarkable changes in protein expression, which provides support for the in-depth analysis in subsequent parts of the article. Similarly, the separation between the siPDIA5 group and the siNC group (siRNA Negative Control) is also significant. UMAP dimensionality reduction clustering analysis revealed that the siPDIA1 group cannot be effectively distinguished from the siPDIA5 group and the siNC group: the siPDIA5 group and the siNC group show good clustering separation, while samples in the siPDIA1 group are scattered within the clusters of the siPDIA5 and siNC groups. This result suggests that the impact of PDIA1 knockout on protein expression lies between PDIA5 knockout (siPDIA5 group) and the negative control (siNC group), which is consistent with the conclusions of the original article—namely, PDIA1 knockout exerts a certain effect but is significantly less potent than PDIA5 knockout.

However, there is an alternative logical possibility: the sequencing data itself may have biases (e.g., poor technical reproducibility, sample contamination, etc.), leading to the observation that "PDIA1 knockout is less effective than PDIA5 knockout" in subsequent analyses. This point requires further consideration and verification.

b) For the proteomics data only, please use 3 volcanos plos to match data from the following pairs of samples: i) siPDIA1 vs. siNC; ii) siPDIA5 vs. siNC, iii) CCF642 vs. vehicle control. Please then indicate the top 10 most upregulated proteins in the i) siPDIA1, ii) siPDIA5 and iii) CCF642 treatment groups, accordingly. (2 points)

代码见文末附件一



Legend

- Not significant
- Upregulated (p<0.05, FC>2)
- Downregulated (p<0.05, FC<0.5)
- Top 10 upregulated
- Top 10 downregulated

The specific upregulated proteins are shown in the following table. If there are multiple protein names, they are separated by semicolons.

Top 10 upregulated protein

siPDIA1 vs siNC			siPDIA5 vs siNC			CCF462 vs vhicle		
PG.ProteinGroups	log2FC	p_value	PG.ProteinGroups	log2FC	p_value	PG.ProteinGroups	log2FC	p_value
Q9Y2Z9	8.382483	0.021134	O00186	3.273453	0.00589	Q9Y2Z9	13.78403	1.94E-06
O15056	3.448171	0.026594	P80365	2.847242	0.023369	Q562R1	7.696186	0.021631
P62861	3.398061	0.021948	Q9H0U4	2.676083	0.045484	P02788	5.085705	0.008784
Q9H0U4	2.974532	0.005598	P02768	2.193007	0.021831	O75351;Q6PIW4;Q9UN37	4.407389	0.000116
Q7Z478	2.320523	0.027469	P19823	2.001224	0.038254	Q9NUQ8	4.336431	0.007885
Q6P1X6	2.095695	0.008208	Q16651	1.967439	0.005616	P63241;Q9GZV4	4.17153	0.001594
Q9GZT6	2.031518	0.000849	P52429	1.854014	0.040207	Q86WA6	4.116344	0.003861
P09601	1.958785	0.000866	P30085	1.495123	0.044149	O00165	4.083558	0.005192
P51688	1.958066	0.008972	Q9NYF8	1.492354	0.042071	Q9UNM6	3.825975	0.001956
P11216	1.943515	0.004415	P68402	1.425226	0.049738	P33992	3.727817	0.004445

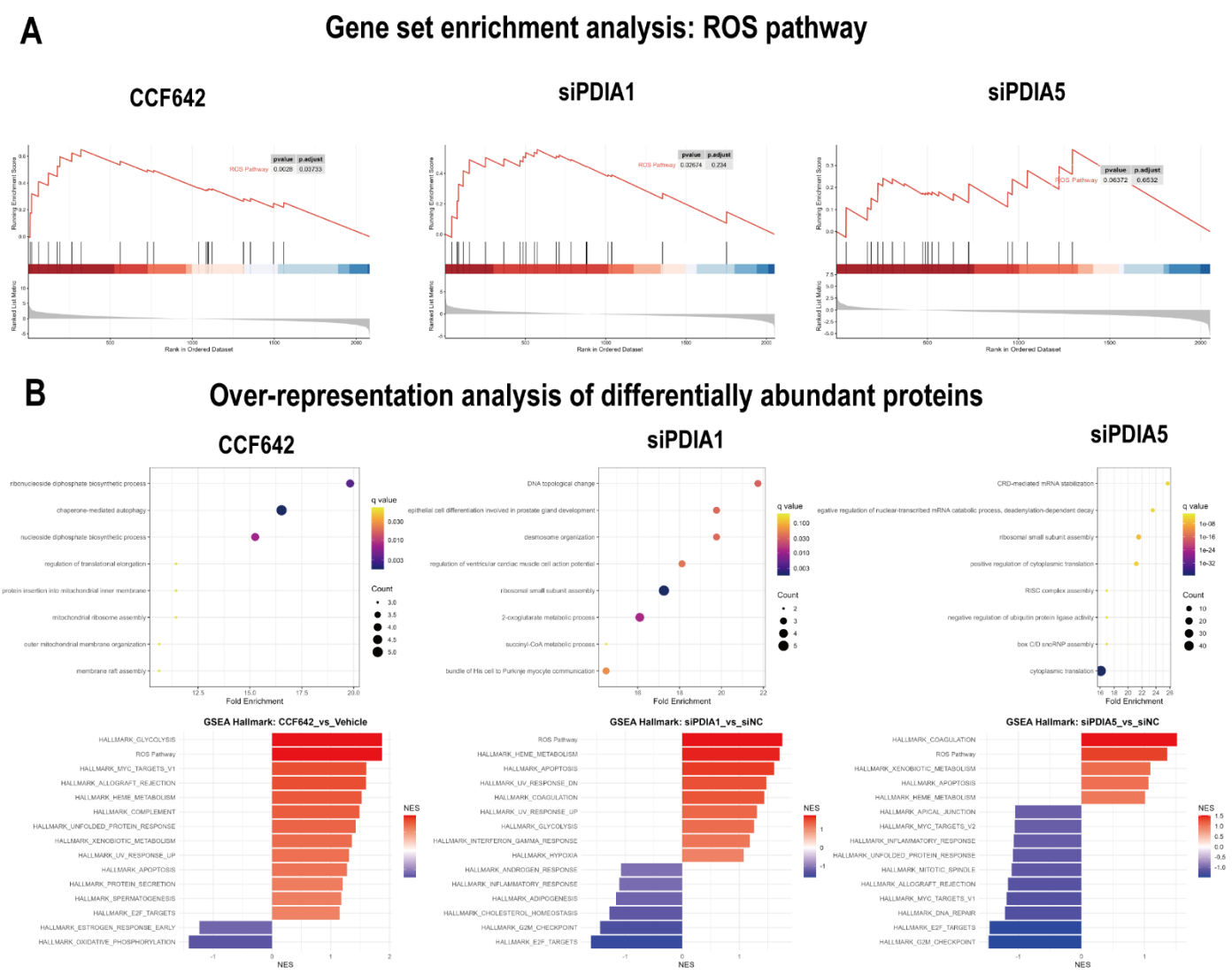
We observed a co-upregulation phenomenon of multiple proteins across the knockout groups (e.g., Q9H0U4). Such proteins can be prioritized for focused attention, or enrichment analysis can be performed on these co-upregulated proteins.

c) Please use either GSEA or ORA to remake Figure 6A or 6B (final figure colour and font size should be different from Figure 6A or 6B in the original figure. Please also do either GSEA or ORA to the transcriptomic data** in this paper and according to your analysis, do the expression pattern of ROS signalling pathway-related genes/proteins similar in both transcriptomic and proteomic analyses? (2 points)

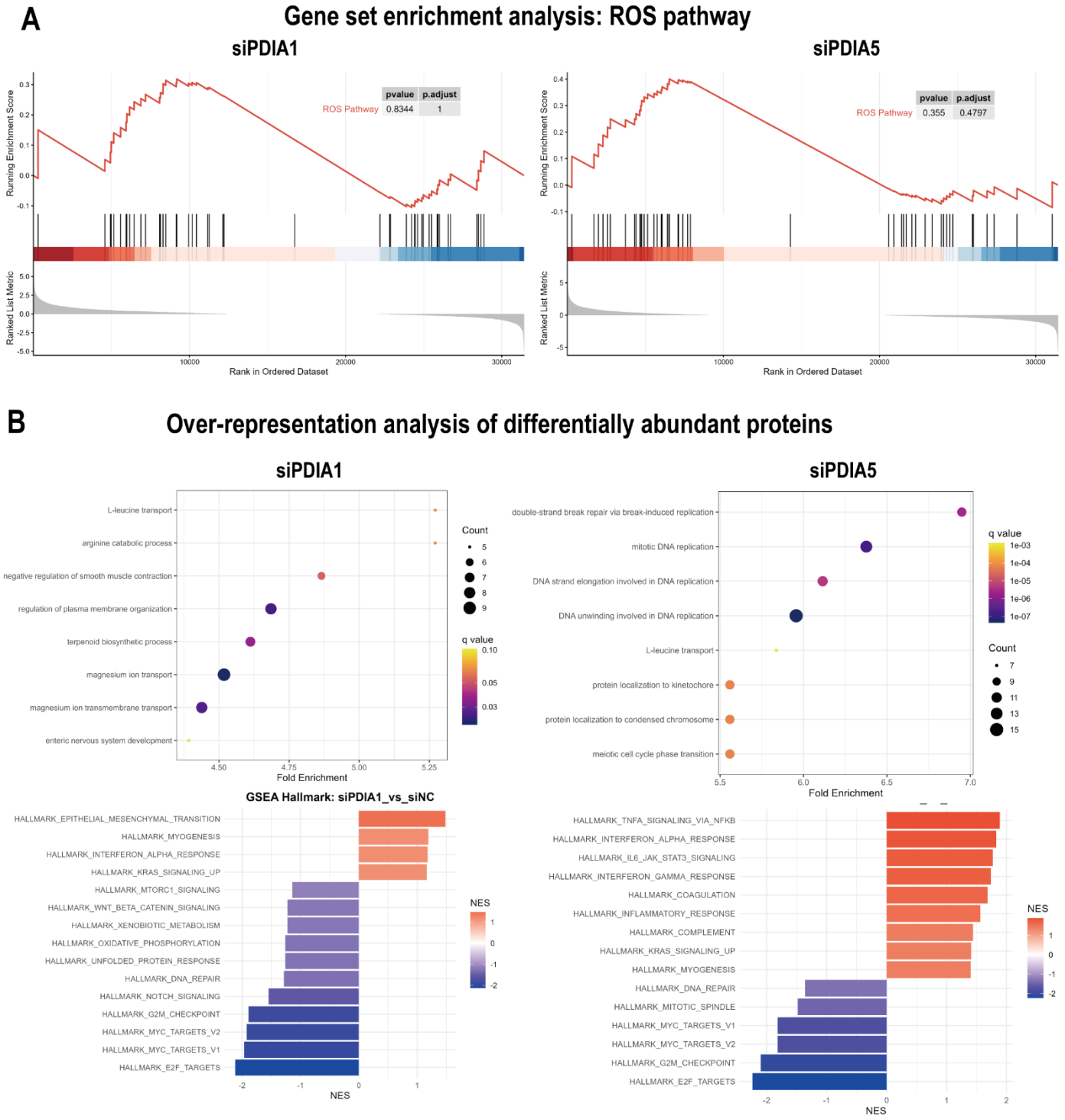
After finishing the first two questions, anyone who used Python fell silent—it's such a hassle! R is way better.

It seems R still holds a dominant position in the field of bioinformatics. Let's switch to R instead. (。ŏ_ŏ)

Confirm the dataset corresponding to the ROS pathway:
"HALLMARK_REACTIVE_OXYGEN_SPECIES_PATHWAY"Confirm the analytical method used for ORA:
We observed that Panel B uses the BP (Biological Process) section of GO (Gene Ontology) analysis.



Download transcriptome data and perform the same analyses as described above. It should be noted that transcriptome data contains non-coding RNAs (ncRNAs), which need to be excluded from the analysis set.



No data for the CCF642 treatment group was retrieved in the transcriptome dataset, so only the siPDIA1 and siPDIA5 groups could be compared.

At the level of the Reactive Oxygen Species (ROS) pathway, the expression patterns of ROS-associated genes/proteins showed significant consistency between the transcriptome (Figure 2) and proteome (Figure 1) data: for both the siPDIA1 and siPDIA5 groups, Gene Set Enrichment Analysis (GSEA) revealed no significant enrichment of the ROS pathway (proteome: siPDIA1 group p=0.844, siPDIA5 group p=0.327; transcriptome: siPDIA1 group p=0.528, siPDIA5 group p=0.638). Additionally, both groups exhibited a slight negative enrichment trend in both omics datasets. Meanwhile, the results of Over-Representation Analysis (ORA) indicated that no significant ROS pathway-related enriched terms were detected in the siPDIA1 and siPDIA5 groups at either the transcriptional or protein level. This finding confirms that the expression characteristics of the ROS pathway are highly similar at the transcriptional and protein levels.

However, the above results are inconsistent with the conclusions stated in the paper. This is presumably due to the inherent difficulty in reproducing bioinformatics analysis results—communication with other students revealed that they also failed to achieve perfect reproducibility. Nevertheless, the core value of bioinformatics analysis lies in inspiring research ideas: as long as it can guide researchers in identifying subsequent experimental directions and support the rigorous derivation of experimental conclusions, the core objective of bioinformatics analysis is fulfilled, and the requirement for reproducibility is actually not overly stringent. After all, complete reproducibility of bioinformatics results is inherently challenging due to the influence of multiple objective factors such as experimental equipment, software versions, analysis environments, and random seeds.

2) Please download and read PMID 26457646. Use IGV or any other software (e.g., R, Python, etc.) / website of your choice, to align ChIP-seq data from paired normal and tumour tissue samples from at last 3 patients to find out whether the expression of the following genes are likely to be regulated by AR: *KLK4*, *mTOR*, *P4HB* and *PDIA5* (4 points, 1 point each).

Step 1: Based on the dataset corresponding to Series GSE70079 in the GEO database (where the DF column represents patient IDs), screen for patients with paired normal and cancerous tissue samples (verification revealed only six eligible patients). Download the dataset files and store the data in the format shown below:

Chip-seq
|-data
| |- GSM1358395_DF_1335_normal_peaks.bed.txt.gz
| |-GSM1358396_DF_1335_tumor_peaks.bed.txt.gz
| |-GSM1358397_DF_1345_normal_peaks.bed.txt.gz
| |-GSM1358398_DF_1345_tumor_peaks.bed.txt.gz
| |-GSM1358399_DF_1373_normal_peaks.bed.txt.gz
| |-GSM1358400_DF_1373_tumor_peaks.bed.txt.gz
| |-GSM1358402_DF_1433_normal_peaks.bed.txt.gz
| |-GSM1358403_DF_1433_tumor_peaks.bed.txt.gz
| |-GSM1358405_DF_1609_normal_peaks.bed.txt.gz
| |-GSM1358406_DF_1609_tumor_peaks.bed.txt.gz
| |-GSM1358409_DF_184_normal_peaks.bed.txt.gz
|- analysis.R

The analysis code is written below, focusing on the AR peak ChIP-seq data of normal/tumor tissues from 6 pairs of prostate cancer patients (patient IDs: 1335, 1345, 1373, 1433, 1609, 184, named patient1 to

patient6 in sequence). The core analysis steps are as follows:
Environment initialization: Clear R environment variables, verify the integrity of the Rtools utility, and batch install and load required analysis packages including tidyverse, GenomicRanges, and limma;
Data preprocessing: Configure data storage paths and result output paths, automatically identify and unzip all files in peaks.bed.txt.gz format under the data directory (skip this step if the files have already been unzipped);
Data import and verification: Define the mapping relationship of peaks.bed files corresponding to normal/tumor tissues of the 6 pairs of patients, batch import the files and convert them into GenomicRanges objects, and verify the number of AR peaks in each sample;
Gene interval construction: Connect to the hg19 version of the Ensembl database, obtain the Transcription Start Sites (TSS) of four target genes (KLK4, mTOR, P4HB, PDIA5), and construct genomic intervals within 50kb upstream and downstream of the TSS of each gene;
Core analysis modules:
Overlap analysis: Count the number of overlaps between AR peaks of normal/tumor tissues and target gene intervals in each pair of samples, and fill 0 values for genes with no overlapping peaks;
Intensity analysis: Extract the RPM intensity values of overlapping AR peaks, calculate the mean and total intensity of peaks corresponding to each gene, and integrate peak count and intensity data;
Differential analysis: Calculate the fold change of intensity in tumor tissues relative to normal tissues, and perform paired differential tests (FDR correction) using the limma package; for all-zero data, directly mark as no difference;
Result output: Finally, output the mean fold change, logFC, FDR-adjusted P-value of each gene, as well as detailed distribution of AR peaks in the target gene intervals;
Auxiliary result interpretation: Clarify the judgment rules for AR regulation to assist in interpreting the tissue-specific binding characteristics of AR to target genes in tumor/normal tissues.

	gene	avg_fc	logFC	adj.P.Val	total_pair_count	tumor_peak_pair	normal_peak_pair
KLK4	KLK4	3.7815761	27.3704754	0.004738304	6	6	6
mTOR	mTOR	0.0109344	0.0000000	1.000000000	6	0	0
P4HB	P4HB	0.2228337	-0.5253167	0.853516241	6	2	2
PDIA5	PDIA5	1.1514546	10.6646006	0.004849404	6	5	0

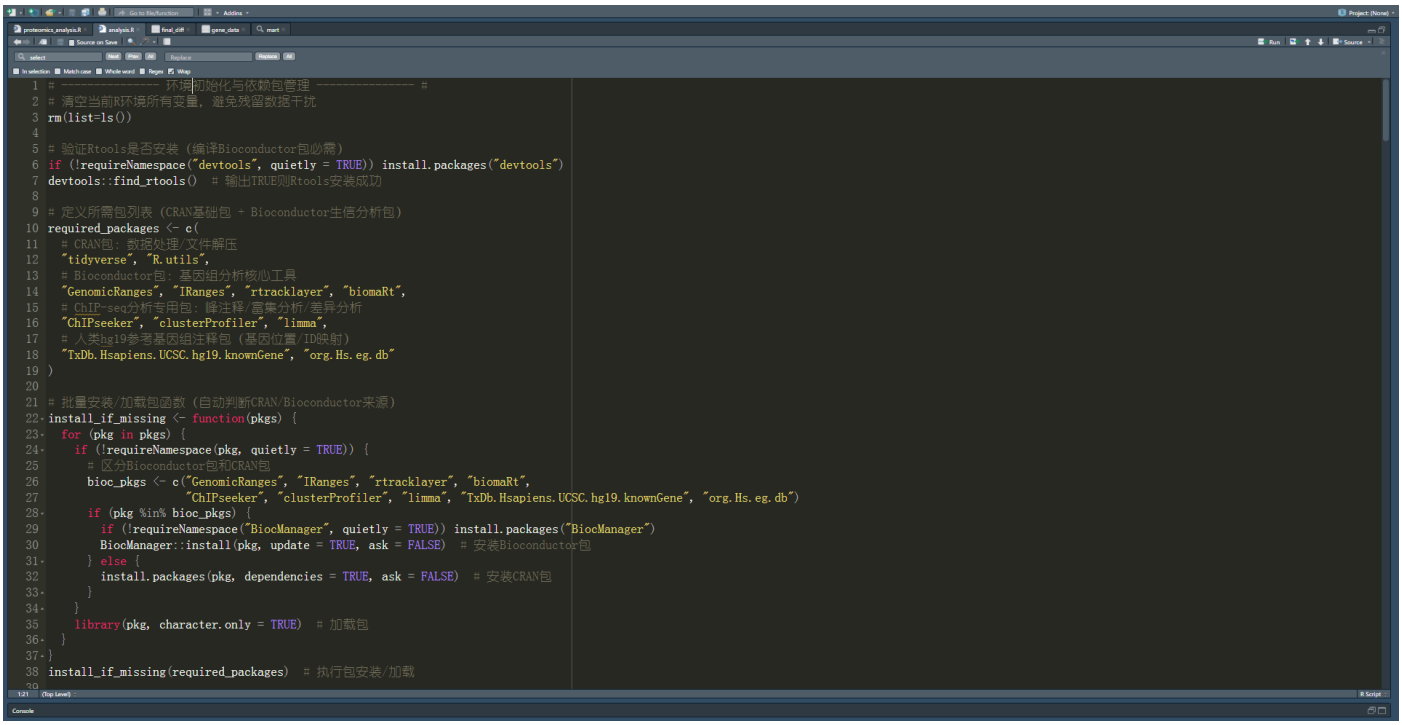
	peak_detail
KLK4	patient1:normal:1,patient1:tumor:2,patient2:normal:1,patient2:tumor:2,patient3:normal:2,patient3:tumor:4,patient4:normal:2,patient4:tumor:2,patient5:normal:1,patient5:tumor:3,patient6:normal:1,patient6:tumor:3
mTOR	patient1:normal:0,patient1:tumor:0,patient2:normal:0,patient2:tumor:0,patient3:normal:0,patient3:tumor:0,patient4:normal:0,patient4:tumor:0,patient5:normal:0,patient5:tumor:0,patient6:normal:0,patient6:tumor:0
P4HB	patient1:normal:0,patient1:tumor:1,patient2:normal:0,patient2:tumor:0,patient3:normal:1,patient3:tumor:0,patient4:normal:4,patient4:tumor:0,patient5:normal:0,patient5:tumor:0,patient6:normal:0,patient6:tumor:1
PDIA5	patient1:normal:0,patient1:tumor:3,patient2:normal:0,patient2:tumor:1,patient3:normal:0,patient3:tumor:1,patient4:normal:0,patient4:tumor:1,patient5:normal:0,patient5:tumor:1,patient6:normal:0,patient6:tumor:0

In conclusion, the findings of this study are as follows:
KLK4 gene: AR binding intensity was significantly increased in tumor tissues, and binding was detected in all patientsAR peak binding signals of the KLK4 gene were detected in both normal and tumor tissues of all 6 patients (the number of patients with detectable binding signals in normal/tumor tissues was 6 for both); the FDR-adjusted P-value was 0.0047 ($P < 0.05$), indicating a statistically significant difference; the average fold change (avg_fc) reached 3.78, suggesting that the binding intensity of AR to the regulatory region of the KLK4 gene in tumor tissues was 3.78 times that in normal tissues, exhibiting a tumor-specific high binding characteristic.
mTOR gene: No AR binding signal was detected, and no inter-tissue differences were observedNo AR peak binding signals of the mTOR gene were detected in either normal or tumor tissues of all 6 patients (the number of patients with detectable binding signals in normal/tumor tissues was 0 for both); the FDR-adjusted P-value was 1.0 (no statistical difference), demonstrating that this gene has no binding association with AR and no relevant differences between normal and tumor tissues.
P4HB gene: AR binding signals were detected only in some patients, with no statistically significant inter-tissue differencesAR peak binding signals of the P4HB gene were detected only in normal tissues of 2 patients and tumor tissues of 2 patients (only in a subset of patients); the FDR-adjusted P-value was 0.85

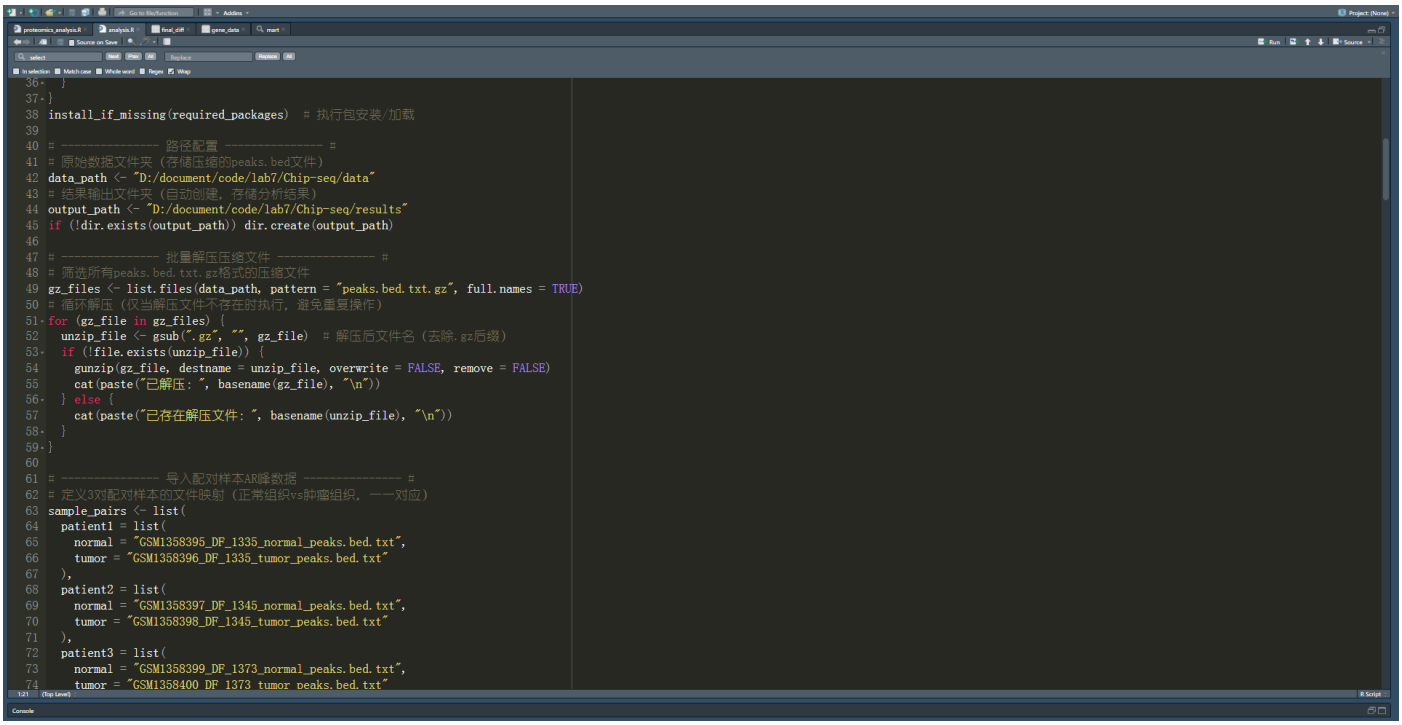
(P > 0.05), with no statistical difference; the average fold change was 0.22 (<1), suggesting that AR binding intensity in tumor tissues was lower than that in normal tissues, but this difference was not significant, and no clear tissue-specific AR binding characteristic was observed.

PDIA5 gene: Exhibited tumor-specific AR binding characteristics with significant inter-tissue differencesNo AR peak binding signals of the PDIA5 gene were detected in normal tissues of all patients (the number of patients with detectable binding signals in normal tissues was 0), while such signals were detected in tumor tissues of 5 patients (the number of patients with detectable binding signals in tumor tissues was 5); the FDR-adjusted P-value was 0.0048 (P < 0.05), indicating a statistically significant difference; the average fold change was 1.15 (>1), suggesting that AR binding intensity in tumor tissues was slightly higher than that in normal tissues, making it a typical tumor-specific AR-binding gene.

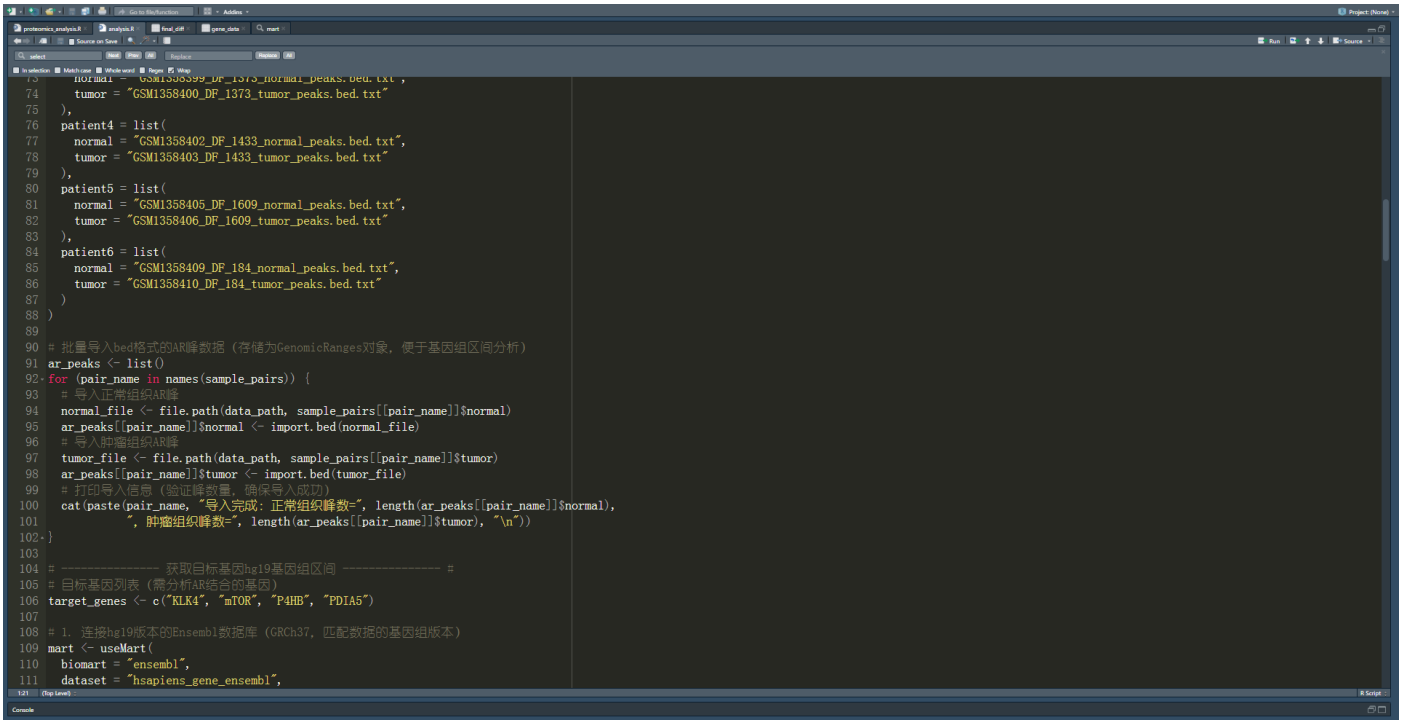
The code for all previous assignments has been open-sourced on GitHub at the address: MakWaiGong/25SCUT-bioinformatics (including code for lab6, lab7, and lab8, not just lab7).



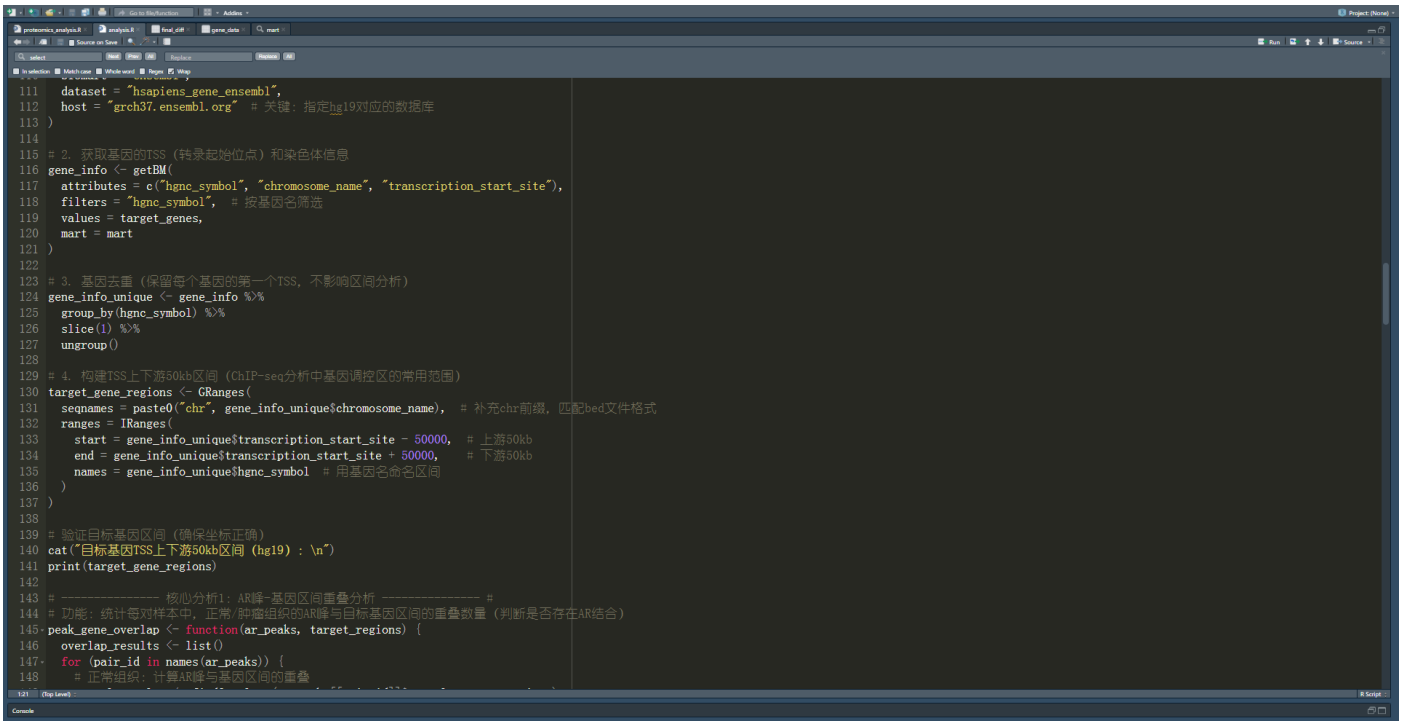
```r
# ---------- 环境初始化与依赖包管理 ---------- #
# 清空当前R环境所有变量，避免残留数据干扰
rm(list=ls())

# 验证Rtools是否安装（编译Bioconductor包必需）
if (!requireNamespace("devtools", quietly = TRUE)) install.packages("devtools")
devtools::find_rtools()  # 输出TRUE则Rtools安装成功

# 定义所需包列表（CRAN基础包 + Bioconductor生信分析包）
required_packages <- c(
  # CRAN包：数据处理/文件解压
  "tidyverse", "R.utils",
  # Bioconductor包：基因组分析核心工具
  "GenomicRanges", "IRanges", "rtracklayer", "biomaRt",
  # ChIP-seq分析专用包：峰注释/富集分析/差异分析
  "ChIPseeker", "clusterProfiler", "limma",
  # 人类hg19参考基因组注释包（基因位置/ID映射）
  "TxDb.Hsapiens.UCSC.hg19.knownGene", "org.Hs.eg.db"
)

# 批量安装/加载包函数（自动判断CRAN/Bioconductor来源）
install_if_missing <- function(pkgs) {
  for (pkg in pkgs) {
    if (!requireNamespace(pkg, quietly = TRUE)) {
      # 区分Bioconductor包和CRAN包
      bioc_pkgs <- c("GenomicRanges", "IRanges", "rtracklayer", "biomaRt",
                     "ChIPseeker", "clusterProfiler", "limma", "TxDb.Hsapiens.UCSC.hg19.knownGene", "org.Hs.eg.db")
      if (pkg %in% bioc_pkgs) {
        if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
        BiocManager::install(pkg, update = TRUE, ask = FALSE)  # 安装Bioconductor包
      } else {
        install.packages(pkg, dependencies = TRUE, ask = FALSE)  # 安装CRAN包
      }
    }
    library(pkg, character.only = TRUE)  # 加载包
  }
}
install_if_missing(required_packages)  # 执行包安装/加载
```



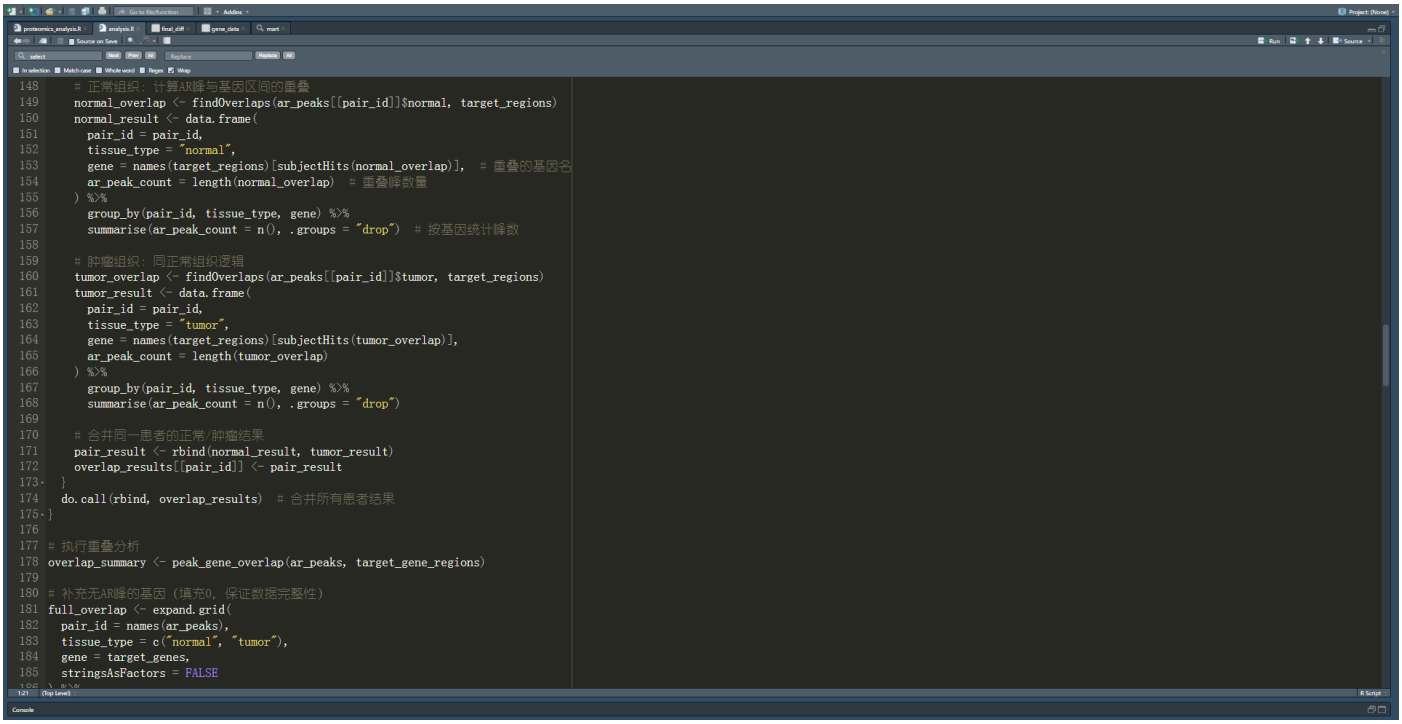
```
}
install_if_missing(required_packages)  # 执行包安装/加载

# ---------- 路径配置 ---------- #
# 原始数据文件夹（存储压缩的peaks.bed文件）
data_path <- "D:/document/code/lab7/Chip-seq/data"
# 结果输出文件夹（自动创建，存储分析结果）
output_path <- "D:/document/code/lab7/Chip-seq/results"
if (!dir.exists(output_path)) dir.create(output_path)

# ---------- 批量解压压缩文件 ---------- #
# 筛选所有peaks.bed.txt.gz格式的压缩文件
gz_files <- list.files(data_path, pattern = "peaks.bed.txt.gz", full.names = TRUE)
# 循环解压（仅当解压文件不存在时执行，避免重复操作）
for (gz_file in gz_files) {
  unzip_file <- gsub(".gz", "", gz_file)  # 解压后文件名（去除.gz后缀）
  if (!file.exists(unzip_file)) {
    gunzip(gz_file, destname = unzip_file, overwrite = FALSE, remove = FALSE)
    cat(paste("已解压：", basename(gz_file), "\n"))
  } else {
    cat(paste("已存在解压文件：", basename(unzip_file), "\n"))
  }
}

# ---------- 导入配对样本AR峰数据 ---------- #
# 定义3对配对样本的文件映射（正常组织vs肿瘤组织，一一对应）
sample_pairs <- list(
  patient1 = list(
    normal = "GSM1358395_DF_1335_normal_peaks.bed.txt",
    tumor = "GSM1358396_DF_1335_tumor_peaks.bed.txt"
  ),
  patient2 = list(
    normal = "GSM1358397_DF_1345_normal_peaks.bed.txt",
    tumor = "GSM1358398_DF_1345_tumor_peaks.bed.txt"
  ),
  patient3 = list(
    normal = "GSM1358399_DF_1373_normal_peaks.bed.txt",
    tumor = "GSM1358400_DF_1373_tumor_peaks.bed.txt"
```



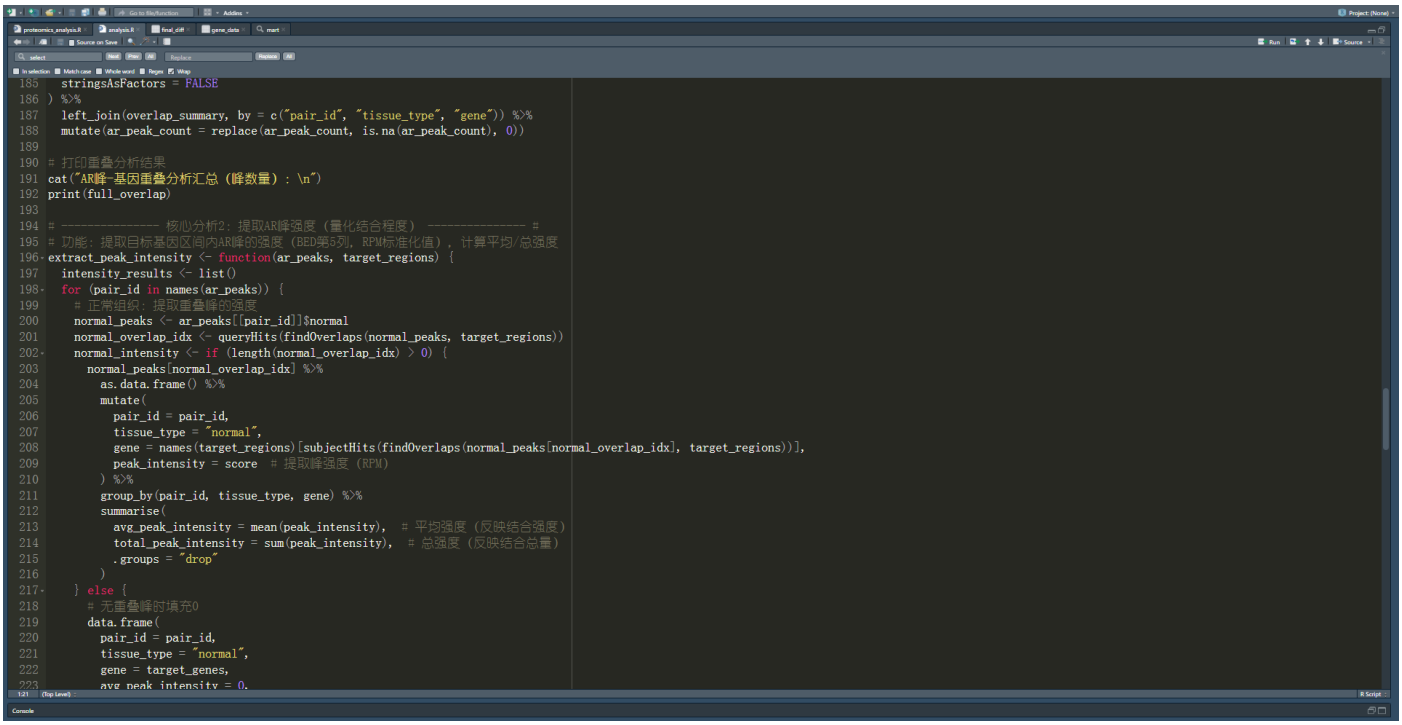
```
    tumor = "GSM1358400_DF_1373_tumor_peaks.bed.txt"
  ),
  patient4 = list(
    normal = "GSM1358402_DF_1433_normal_peaks.bed.txt",
    tumor = "GSM1358403_DF_1433_tumor_peaks.bed.txt"
  ),
  patient5 = list(
    normal = "GSM1358405_DF_1609_normal_peaks.bed.txt",
    tumor = "GSM1358406_DF_1609_tumor_peaks.bed.txt"
  ),
  patient6 = list(
    normal = "GSM1358409_DF_184_normal_peaks.bed.txt",
    tumor = "GSM1358410_DF_184_tumor_peaks.bed.txt"
  )
)

# 批量导入bed格式的AR峰数据（存储为GenomicRanges对象，便于基因组区间分析）
ar_peaks <- list()
for (pair_name in names(sample_pairs)) {
  # 导入正常组织AR峰
  normal_file <- file.path(data_path, sample_pairs[[pair_name]]$normal)
  ar_peaks[[pair_name]]$normal <- import.bed(normal_file)
  # 导入肿瘤组织AR峰
  tumor_file <- file.path(data_path, sample_pairs[[pair_name]]$tumor)
  ar_peaks[[pair_name]]$tumor <- import.bed(tumor_file)
  # 打印导入信息（验证峰数量，确保导入成功）
  cat(paste(pair_name, "导入完成：正常组织峰数=", length(ar_peaks[[pair_name]]$normal),
            ", 肿瘤组织峰数=", length(ar_peaks[[pair_name]]$tumor), "\n"))
}

# ---------- 获取目标基因hg19基因组区间 ---------- #
# 目标基因列表（需分析AR结合的基因）
target_genes <- c("KLK4", "mTOR", "P4HB", "PDIA5")

# 1. 连接hg19版本的Ensembl数据库（GRCh37，匹配数据的基因组版本）
mart <- useMart(
  biomart = "ensembl",
  dataset = "hsapiens_gene_ensembl",
```



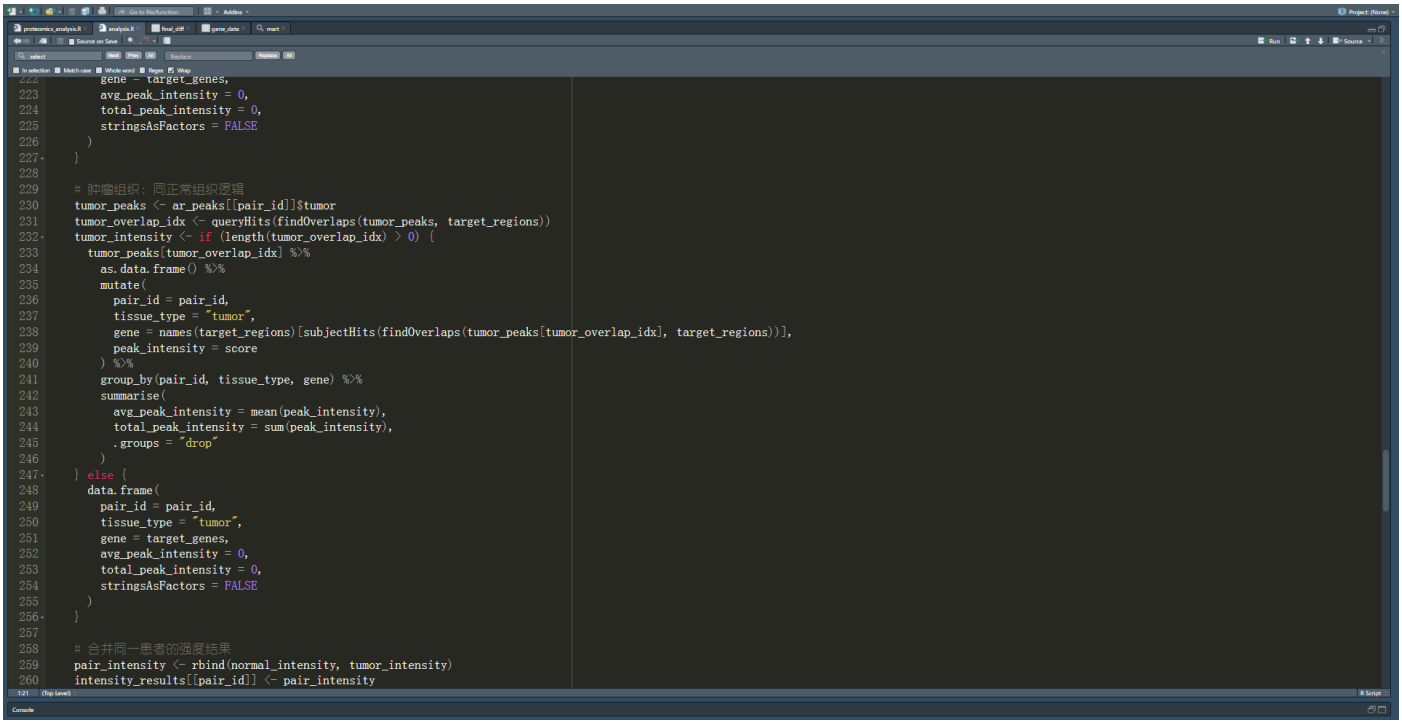
```
  dataset = "hsapiens_gene_ensembl",
  host = "grch37.ensembl.org"  # 关键：指定hg19对应的数据库
)

# 2. 获取基因的TSS（转录起始位点）和染色体信息
gene_info <- getBM(
  attributes = c("hgnc_symbol", "chromosome_name", "transcription_start_site"),
  filters = "hgnc_symbol",  # 按基因名筛选
  values = target_genes,
  mart = mart
)

# 3. 基因去重（保留每个基因的第一个TSS，不影响区间分析）
gene_info_unique <- gene_info %>%
  group_by(hgnc_symbol) %>%
  slice(1) %>%
  ungroup()

# 4. 构建TSS上下游50kb区间（ChIP-seq分析中基因调控区的常用范围）
target_gene_regions <- GRanges(
  seqnames = paste0("chr", gene_info_unique$chromosome_name),  # 补充chr前缀，匹配bed文件格式
  ranges = IRanges(
    start = gene_info_unique$transcription_start_site - 50000,  # 上游50kb
    end = gene_info_unique$transcription_start_site + 50000,    # 下游50kb
    names = gene_info_unique$hgnc_symbol  # 用基因名命名区间
  )
)

# 验证目标基因区间（确保坐标正确）
cat("目标基因TSS上下游50kb区间（hg19）：\n")
print(target_gene_regions)

# ---------------- 核心分析1：AR峰-基因区间重叠分析 ---------------- #
# 功能：统计每对样本中，正常/肿瘤组织的AR峰与目标基因区间的重叠数量（判断是否存在AR结合）
peak_gene_overlap <- function(ar_peaks, target_regions) {
  overlap_results <- list()
  for (pair_id in names(ar_peaks)) {
    # 正常组织：计算AR峰与基因区间的重叠
```



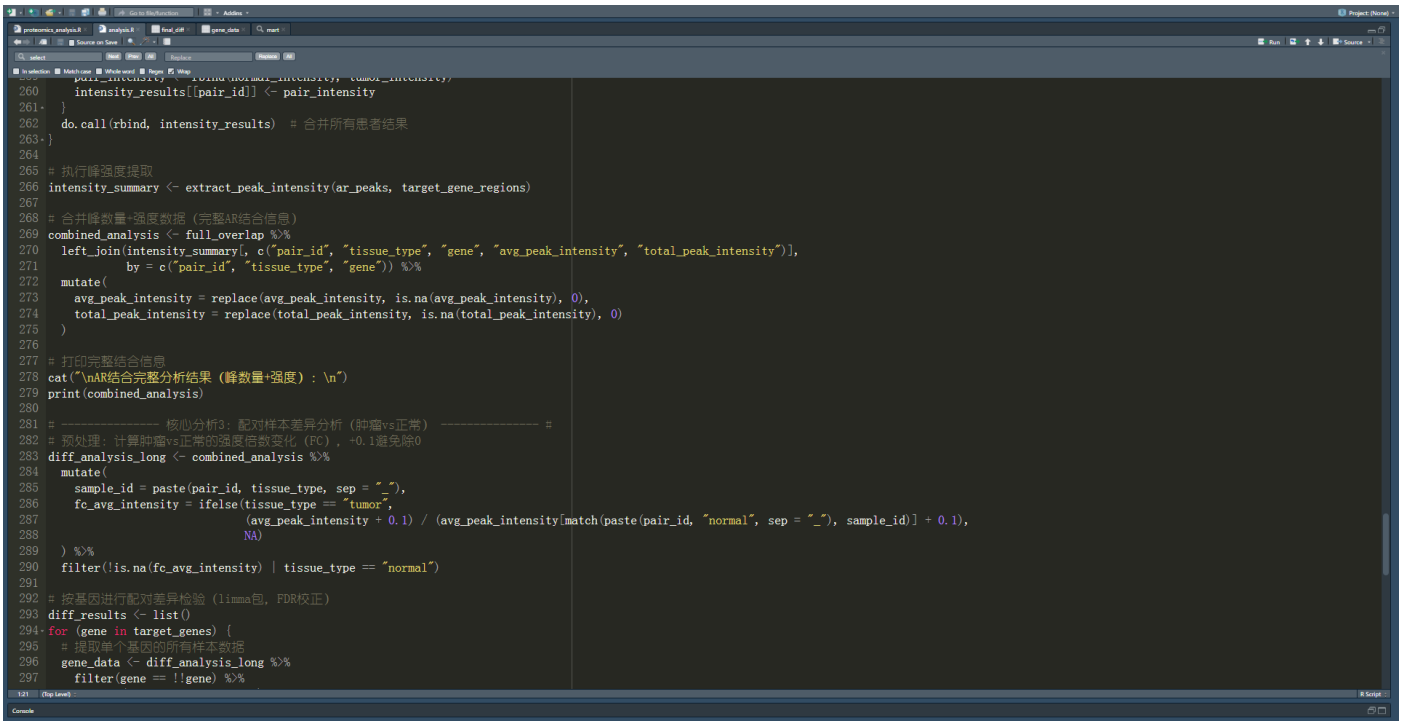
```
    # 正常组织：计算AR峰与基因区间的重叠
    normal_overlap <- findOverlaps(ar_peaks[[pair_id]]$normal, target_regions)
    normal_result <- data.frame(
      pair_id = pair_id,
      tissue_type = "normal",
      gene = names(target_regions)[subjectHits(normal_overlap)],  # 重叠的基因名
      ar_peak_count = length(normal_overlap)  # 重叠峰数量
    ) %>%
      group_by(pair_id, tissue_type, gene) %>%
      summarise(ar_peak_count = n(), .groups = "drop")  # 按基因统计峰数

    # 肿瘤组织：同正常组织逻辑
    tumor_overlap <- findOverlaps(ar_peaks[[pair_id]]$tumor, target_regions)
    tumor_result <- data.frame(
      pair_id = pair_id,
      tissue_type = "tumor",
      gene = names(target_regions)[subjectHits(tumor_overlap)],
      ar_peak_count = length(tumor_overlap)
    ) %>%
      group_by(pair_id, tissue_type, gene) %>%
      summarise(ar_peak_count = n(), .groups = "drop")

    # 合并同一患者的正常/肿瘤结果
    pair_result <- rbind(normal_result, tumor_result)
    overlap_results[[pair_id]] <- pair_result
  }
  do.call(rbind, overlap_results)  # 合并所有患者结果
}

# 执行重叠分析
overlap_summary <- peak_gene_overlap(ar_peaks, target_gene_regions)

# 补充无AR峰的基因（填充0，保证数据完整性）
full_overlap <- expand.grid(
  pair_id = names(ar_peaks),
  tissue_type = c("normal", "tumor"),
  gene = target_genes,
  stringsAsFactors = FALSE
```



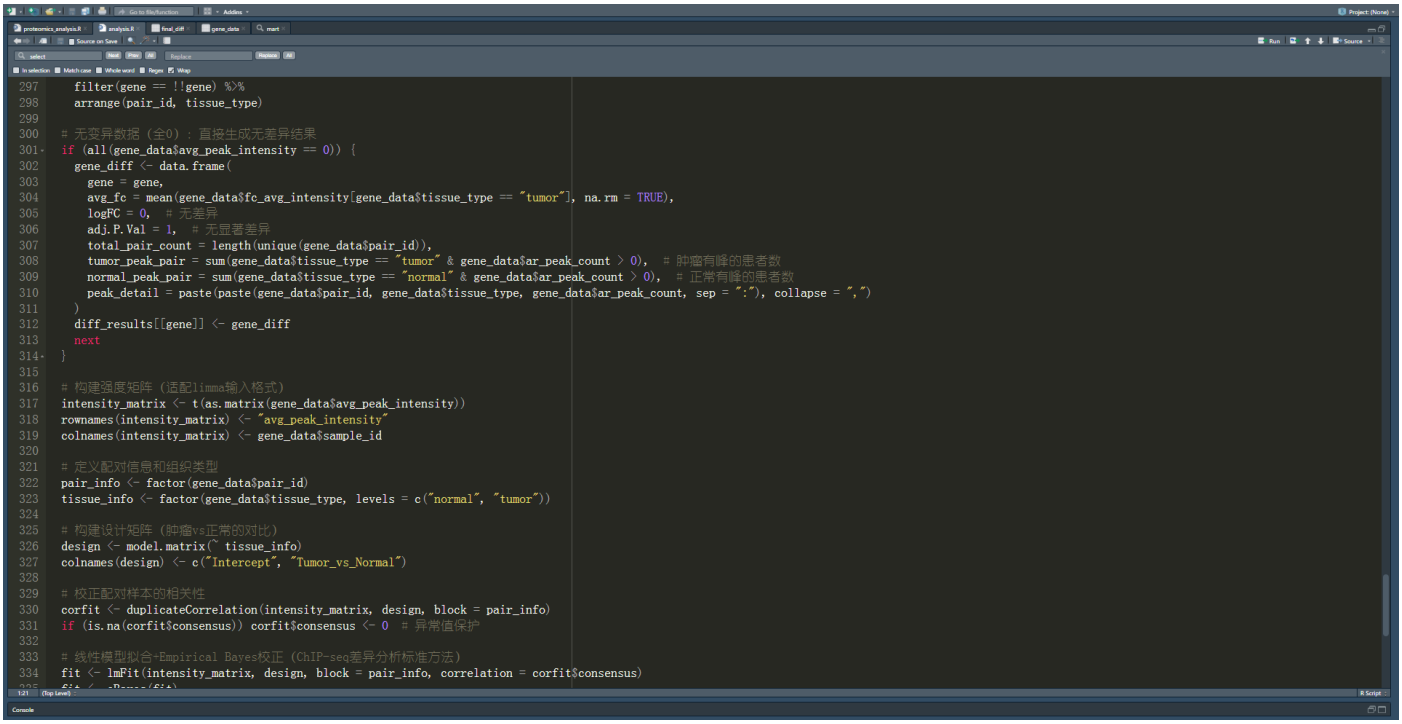
```
  stringsAsFactors = FALSE
) %>%
  left_join(overlap_summary, by = c("pair_id", "tissue_type", "gene")) %>%
  mutate(ar_peak_count = replace(ar_peak_count, is.na(ar_peak_count), 0))

# 打印重叠分析结果
cat("AR峰-基因重叠分析汇总（峰数量）：\n")
print(full_overlap)

# ---------- 核心分析2：提取AR峰强度（量化结合程度） ----------
# 功能：提取目标基因区间内AR峰的强度（BED第5列，RPM标准化值），计算平均/总强度
extract_peak_intensity <- function(ar_peaks, target_regions) {
  intensity_results <- list()
  for (pair_id in names(ar_peaks)) {
    # 正常组织：提取重叠峰的强度
    normal_peaks <- ar_peaks[[pair_id]]$normal
    normal_overlap_idx <- queryHits(findOverlaps(normal_peaks, target_regions))
    normal_intensity <- if (length(normal_overlap_idx) > 0) {
      normal_peaks[normal_overlap_idx] %>%
        as.data.frame() %>%
        mutate(
          pair_id = pair_id,
          tissue_type = "normal",
          gene = names(target_regions)[subjectHits(findOverlaps(normal_peaks[normal_overlap_idx], target_regions))],
          peak_intensity = score  # 提取峰强度（RPM）
        ) %>%
        group_by(pair_id, tissue_type, gene) %>%
        summarise(
          avg_peak_intensity = mean(peak_intensity),  # 平均强度（反映结合强度）
          total_peak_intensity = sum(peak_intensity),  # 总强度（反映结合总量）
          .groups = "drop"
        )
    } else {
      # 无重叠峰时填充0
      data.frame(
        pair_id = pair_id,
        tissue_type = "normal",
        gene = target_genes,
        avg_peak_intensity = 0,
```



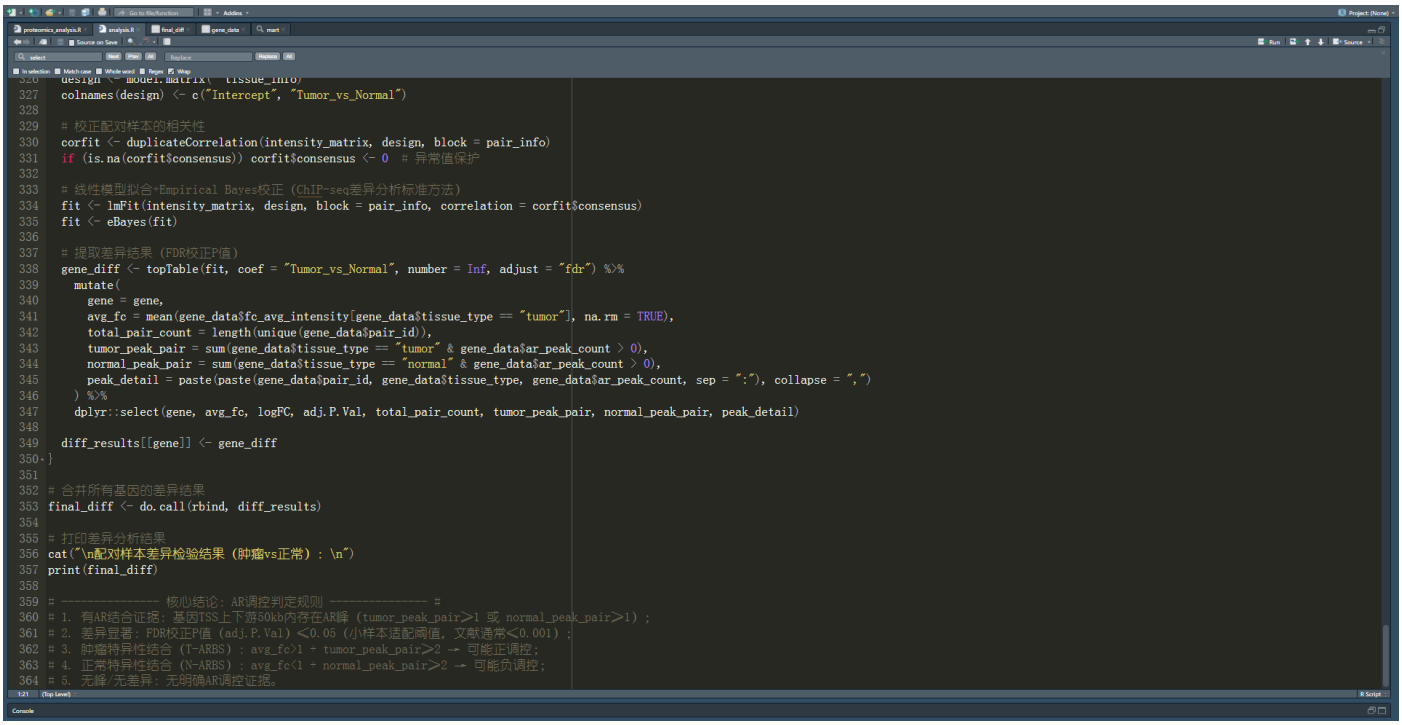
```
        gene = target_genes,
        avg_peak_intensity = 0,
        total_peak_intensity = 0,
        stringsAsFactors = FALSE
      )
    }

    # 肿瘤组织：同正常组织逻辑
    tumor_peaks <- ar_peaks[[pair_id]]$tumor
    tumor_overlap_idx <- queryHits(findOverlaps(tumor_peaks, target_regions))
    tumor_intensity <- if (length(tumor_overlap_idx) > 0) {
      tumor_peaks[tumor_overlap_idx] %>%
        as.data.frame() %>%
        mutate(
          pair_id = pair_id,
          tissue_type = "tumor",
          gene = names(target_regions)[subjectHits(findOverlaps(tumor_peaks[tumor_overlap_idx], target_regions))],
          peak_intensity = score
        ) %>%
        group_by(pair_id, tissue_type, gene) %>%
        summarise(
          avg_peak_intensity = mean(peak_intensity),
          total_peak_intensity = sum(peak_intensity),
          .groups = "drop"
        )
    } else {
      data.frame(
        pair_id = pair_id,
        tissue_type = "tumor",
        gene = target_genes,
        avg_peak_intensity = 0,
        total_peak_intensity = 0,
        stringsAsFactors = FALSE
      )
    }

    # 合并同一患者的强度结果
    pair_intensity <- rbind(normal_intensity, tumor_intensity)
    intensity_results[[pair_id]] <- pair_intensity
```



```
    intensity_results[[pair_id]] <- pair_intensity
  }
  do.call(rbind, intensity_results)  # 合并所有患者结果
}

# 执行峰强度提取
intensity_summary <- extract_peak_intensity(ar_peaks, target_gene_regions)

# 合并峰数量+强度数据（完整AR结合信息）
combined_analysis <- full_overlap %>%
  left_join(intensity_summary[, c("pair_id", "tissue_type", "gene", "avg_peak_intensity", "total_peak_intensity")],
            by = c("pair_id", "tissue_type", "gene")) %>%
  mutate(
    avg_peak_intensity = replace(avg_peak_intensity, is.na(avg_peak_intensity), 0),
    total_peak_intensity = replace(total_peak_intensity, is.na(total_peak_intensity), 0)
  )

# 打印完整结合信息
cat("\nAR结合完整分析结果（峰数量+强度）：\n")
print(combined_analysis)

# ---------- 核心分析3：配对样本差异分析（肿瘤vs正常） ---------- #
# 预处理：计算肿瘤vs正常的强度倍数变化（FC），+0.1避免除0
diff_analysis_long <- combined_analysis %>%
  mutate(
    sample_id = paste(pair_id, tissue_type, sep = "_"),
    fc_avg_intensity = ifelse(tissue_type == "tumor",
                              (avg_peak_intensity + 0.1) / (avg_peak_intensity[match(paste(pair_id, "normal", sep = "_"), sample_id)] + 0.1),
                              NA)
  ) %>%
  filter(!is.na(fc_avg_intensity) | tissue_type == "normal")

# 按基因进行配对差异检验（limma包，FDR校正）
diff_results <- list()
for (gene in target_genes) {
  # 提取单个基因的所有样本数据
  gene_data <- diff_analysis_long %>%
    filter(gene == !!gene) %>%
```



```
    filter(gene == !!gene) %>%
    arrange(pair_id, tissue_type)

  # 无变异数据（全0）：直接生成无差异结果
  if (all(gene_data$avg_peak_intensity == 0)) {
    gene_diff <- data.frame(
      gene = gene,
      avg_fc = mean(gene_data$fc_avg_intensity[gene_data$tissue_type == "tumor"], na.rm = TRUE),
      logFC = 0,   # 无差异
      adj.P.Val = 1,   # 无显著差异
      total_pair_count = length(unique(gene_data$pair_id)),
      tumor_peak_pair = sum(gene_data$tissue_type == "tumor" & gene_data$ar_peak_count > 0),  # 肿瘤有峰的患者数
      normal_peak_pair = sum(gene_data$tissue_type == "normal" & gene_data$ar_peak_count > 0),  # 正常有峰的患者数
      peak_detail = paste(paste(gene_data$pair_id, gene_data$tissue_type, gene_data$ar_peak_count, sep = ":"), collapse = ",")
    )
    diff_results[[gene]] <- gene_diff
    next
  }

  # 构建强度矩阵（适配limma输入格式）
  intensity_matrix <- t(as.matrix(gene_data$avg_peak_intensity))
  rownames(intensity_matrix) <- "avg_peak_intensity"
  colnames(intensity_matrix) <- gene_data$sample_id

  # 定义配对信息和组织类型
  pair_info <- factor(gene_data$pair_id)
  tissue_info <- factor(gene_data$tissue_type, levels = c("normal", "tumor"))

  # 构建设计矩阵（肿瘤vs正常的对比）
  design <- model.matrix(~ tissue_info)
  colnames(design) <- c("Intercept", "Tumor_vs_Normal")

  # 校正配对样本的相关性
  corfit <- duplicateCorrelation(intensity_matrix, design, block = pair_info)
  if (is.na(corfit$consensus)) corfit$consensus <- 0  # 异常值保护

  # 线性模型拟合+Empirical Bayes校正（ChIP-seq差异分析标准方法）
  fit <- lmFit(intensity_matrix, design, block = pair_info, correlation = corfit$consensus)
```



```
design <- model.matrix(~ tissue_info)
colnames(design) <- c("Intercept", "Tumor_vs_Normal")

# 校正配对样本的相关性
corfit <- duplicateCorrelation(intensity_matrix, design, block = pair_info)
if (is.na(corfit$consensus)) corfit$consensus <- 0  # 异常值保护

# 线性模型拟合+Empirical Bayes校正（ChIP-seq差异分析标准方法）
fit <- lmFit(intensity_matrix, design, block = pair_info, correlation = corfit$consensus)
fit <- eBayes(fit)

# 提取差异结果（FDR校正P值）
gene_diff <- topTable(fit, coef = "Tumor_vs_Normal", number = Inf, adjust = "fdr") %>%
  mutate(
    gene = gene,
    avg_fc = mean(gene_data$fc_avg_intensity[gene_data$tissue_type == "tumor"], na.rm = TRUE),
    total_pair_count = length(unique(gene_data$pair_id)),
    tumor_peak_pair = sum(gene_data$tissue_type == "tumor" & gene_data$ar_peak_count > 0),
    normal_peak_pair = sum(gene_data$tissue_type == "normal" & gene_data$ar_peak_count > 0),
    peak_detail = paste(paste(gene_data$pair_id, gene_data$tissue_type, gene_data$ar_peak_count, sep = ":"), collapse = ",")
  ) %>%
  dplyr::select(gene, avg_fc, logFC, adj.P.Val, total_pair_count, tumor_peak_pair, normal_peak_pair, peak_detail)

diff_results[[gene]] <- gene_diff
}

# 合并所有基因的差异结果
final_diff <- do.call(rbind, diff_results)

# 打印差异分析结果
cat("\n配对样本差异检验结果（肿瘤vs正常）：\n")
print(final_diff)

# ---------- 核心结论：AR调控判定规则 ---------- #
# 1. 有AR结合证据：基因TSS上下游50kb内存在AR峰（tumor_peak_pair>1 或 normal_peak_pair>1）；
# 2. 差异显著：FDR校正P值（adj.P.Val）<0.05（小样本适配阈值，文献通常<0.001）；
# 3. 肿瘤特异性结合（T-ARBS）：avg_fc>1 + tumor_peak_pair>2 → 可能正调控；
# 4. 正常特异性结合（N-ARBS）：avg_fc<1 + normal_peak_pair>2 → 可能负调控；
# 5. 无峰/无差异：无明确AR调控证据。
```

提交作业：Please provide your answers on Page 3 onwards within this word file.

请下载并研读 PMID: 41086208 对应的文献，分析其**转录组学与蛋白质组学数据**，随后完成以下任务：

a) **仅针对蛋白质组学数据**，绘制**降维分析图**（如主成分分析图 PCA 或均匀流形逼近与投影图 UMAP）；并根据图中的样本分布模式，判定该蛋白质组学实验的质量。（2 分）

b) **仅针对蛋白质组学数据**，绘制 **3 张火山图**，分别对应以下三组样本的对比数据：① 靶向 PDIA1 的小干扰 RNA 组（siPDIA1）vs. 阴性对照小干扰 RNA 组（siNC）；② 靶向 PDIA5 的小干扰 RNA 组（siPDIA5）vs. 阴性对照小干扰 RNA 组（siNC）；③ CCF642 药物处理组 vs. 溶媒对照组。并分别列出 ① siPDIA1 组、② siPDIA5 组、③ CCF642 处理组中，表达上调最显著的**前 10 种蛋白质**。（2 分）

c) 采用**基因集富集分析（GSEA）或过度表达分析（ORA）**中的任意一种方法，**重新绘制原文献的图 6A 或图 6B**（最终生成的图，其配色与字体大小需与原文献图 6A/6B 存在差异）。同时，采用 GSEA 或 ORA 方法对该文献中的**转录组学数据**进行分析；基于你的分析结果，判断**活性氧（ROS）信号通路相关的基因 / 蛋白质**，在转录组学与蛋白质组学层面的表达模式是否相似。（2 分

附件一，第一题(b)的代码

```
print("\n" + "="*60)
print("Cell 2: 火山图相关分析和可视化")
print("="*60)

# 处理 Vehicle 数据：转换为 log2
vehicle_cols = df.columns[df.columns.str.contains("Vehicle.*Quantity") & ~df.columns.str.contains("Log2")]
print(f"找到 {len(vehicle_cols)} 个 Vehicle Quantity 列")

if len(vehicle_cols) > 0:
    vehicle_df = df[vehicle_cols].copy()
    # 将 Quantity 转换为 Log2Quantity
    vehicle_df = np.log2(vehicle_df.replace(0, np.nan))  # 避免 log(0)
    vehicle_df.columns = vehicle_df.columns.str.replace('.PG.Quantity', '.PG.Log2Quantity')
    print(f"Vehicle 数据转换完成，形状: {vehicle_df.shape}")
else:
    print("未找到 Vehicle Quantity 列，跳过转换")
    vehicle_df = pd.DataFrame()
```

```
# 合并所有 Log2Quantity 数据
all_log2_cols = df.columns[df.columns.str.contains("PG.Log2Quantity")]
all_log2_df = df[all_log2_cols].copy()

if len(vehicle_df) > 0:
    # 添加转换后的 vehicle 数据
    all_log2_df = pd.concat([all_log2_df, vehicle_df], axis=1)

print(f"合并后数据形状: {all_log2_df.shape}")

# 定义比较组
comparisons = {
    'siPDIA1_vs_siNC': {
        'treatment': all_log2_df.columns[all_log2_df.columns.str.contains("PDIA1.*Log2Quantity")],
        'control': all_log2_df.columns[all_log2_df.columns.str.contains("SC1.*Log2Quantity")]
    },
    'siPDIA5_vs_siNC': {
        'treatment': all_log2_df.columns[all_log2_df.columns.str.contains("PDIA5.*Log2Quantity")],
        'control': all_log2_df.columns[all_log2_df.columns.str.contains("SC1.*Log2Quantity")]
    },
    'CCF642_vs_Vehicle': {
        'treatment': all_log2_df.columns[all_log2_df.columns.str.contains("CCF642.*Log2Quantity")],
        'control': vehicle_df.columns if len(vehicle_df) > 0 else []
    }
}

# 验证样本数
for name, groups in comparisons.items():
    print(f"{name}: treatment={len(groups['treatment'])}, control={len(groups['control'])}")

# 2. 计算 fold change 和 p 值，绘制火山图
```

```
# 创建火山图
fig, axes = plt.subplots(1, 3, figsize=(20, 7))
top_upregulated = {}

# 创建用于图注的虚拟点
legend_elements = [
    plt.scatter([], [], alpha=0.6, s=40, color='#9E9E9E', label='Not significant'),
    plt.scatter([], [], alpha=0.8, s=40, color='#EA4335', edgecolors='#C53929', linewidth=1.5,
                label='Upregulated (p<0.05, FC>2)'),
    plt.scatter([], [], alpha=0.8, s=40, color='#4285F4', edgecolors='#1565C0', linewidth=1.5,
                label='Downregulated (p<0.05, FC<0.5)'),
    plt.scatter([], [], s=40, color='#EA4335', marker='o', edgecolors='black', linewidth=1.5,
                alpha=1.0, label='Top 10 upregulated'),
    plt.scatter([], [], s=40, color='#4285F4', marker='o', edgecolors='black', linewidth=1.5,
                alpha=1.0, label='Top 10 downregulated')
]

for i, (comparison_name, groups) in enumerate(comparisons.items()):
    treatment_cols = groups['treatment']
    control_cols = groups['control']

    if len(treatment_cols) == 0 or len(control_cols) == 0:
        print(f"跳过 {comparison_name}: 缺少数据列")
        continue

    # 获取数据
    treatment_data = all_log2_df[treatment_cols]
    control_data = all_log2_df[control_cols]

    # 计算 fold change (log2FC)
```

```
    treatment_mean = treatment_data.mean(axis=1)
    control_mean = control_data.mean(axis=1)
    log2fc = treatment_mean - control_mean

    # 计算 p 值 (t-test)
    p_values = []
    valid_proteins = 0
    for protein in all_log2_df.index:
        try:
            t_stat, p_val = stats.ttest_ind(
                treatment_data.loc[protein].dropna(),
                control_data.loc[protein].dropna(),
                equal_var=False  # Welch's t-test
            )
            p_values.append(p_val)
            valid_proteins += 1
        except:
            p_values.append(np.nan)

    p_values = pd.Series(p_values, index=all_log2_df.index)
    print(f"{comparison_name}: 处理了 {valid_proteins} 个蛋白")

    # 创建结果 dataframe
    volcano_df = pd.DataFrame({
        'log2FC': log2fc,
        'p_value': p_values,
        '-log10_p_value': -np.log10(p_values + 1e-10)  # 避免 log(0)
    })

    # 移除 NaN 值
    volcano_df = volcano_df.dropna()
```

```
    print(f"{comparison_name}: 有效数据点 {len(volcano_df)} 个")

    # 绘制火山图
    ax = axes[i]

    # 首先识别 Top10 蛋白
    upregulated = (volcano_df['log2FC'] > 1) & (volcano_df['p_value'] < 0.05)
    downregulated = (volcano_df['log2FC'] < -1) & (volcano_df['p_value'] < 0.05)

    print(f"{comparison_name} - 上调显著蛋白: {upregulated.sum()} 个")
    print(f"{comparison_name} - 下调显著蛋白: {downregulated.sum()} 个")

    top_upregulated_proteins = volcano_df[upregulated].sort_values('log2FC', ascending=False).head(10)
    top_downregulated_proteins = volcano_df[downregulated].sort_values('log2FC', ascending=True).head(10)

    # 验证 Top10 选择的正确性
    if len(top_upregulated_proteins) > 0:
        print(f"{comparison_name} - 上调 Top10 log2FC 范围: {top_upregulated_proteins['log2FC'].min():.2f} ~ {top_upregulated_proteins['log2FC'].max():.2f}")
    if len(top_downregulated_proteins) > 0:
        print(f"{comparison_name} - 下调 Top10 log2FC 范围: {top_downregulated_proteins['log2FC'].min():.2f} ~ {top_downregulated_proteins['log2FC'].max():.2f}")

    # 获取 Top10 蛋白的索引，用于后续区分绘制
    top_upregulated_indices = set(top_upregulated_proteins.index)
    top_downregulated_indices = set(top_downregulated_proteins.index)

    # 绘制所有点（统一大小）
    all_points = volcano_df.copy()

    # 1. 未标注的普通点（不显著）
```

```
    normal_points = all_points[
        ~(upregulated | downregulated)  # 既不上调也不下调的点
    ]
    ax.scatter(normal_points['log2FC'], normal_points['-log10_p_value'],
               alpha=0.6, s=40, color='#9E9E9E', zorder=1)

    # 2. 上调显著点（不包括 Top10）
    upregulated_not_top = volcano_df[upregulated].loc[
        ~volcano_df[upregulated].index.isin(top_upregulated_indices)
    ]
    if len(upregulated_not_top) > 0:
        ax.scatter(upregulated_not_top['log2FC'], upregulated_not_top['-log10_p_value'],
                   alpha=0.8, s=40, color='#EA4335', edgecolors='#C53929', linewidth=1.5, zorder=2)

    # 3. 下调显著点（不包括 Top10）
    downregulated_not_top = volcano_df[downregulated].loc[
        ~volcano_df[downregulated].index.isin(top_downregulated_indices)
    ]
    if len(downregulated_not_top) > 0:
        ax.scatter(downregulated_not_top['log2FC'], downregulated_not_top['-log10_p_value'],
                   alpha=0.8, s=40, color='#4285F4', edgecolors='#1565C0', linewidth=1.5, zorder=2)

    # 4. Top5 点（使用与相应显著点相同的样式）
    if len(top_upregulated_proteins) > 0:
        ax.scatter(top_upregulated_proteins['log2FC'], top_upregulated_proteins['-log10_p_value'],
                   s=40, color='#EA4335', marker='o', edgecolors='black', linewidth=1.5,
                   alpha=1.0, zorder=3)

    if len(top_downregulated_proteins) > 0:
        ax.scatter(top_downregulated_proteins['log2FC'], top_downregulated_proteins['-log10_p_value'],
                   s=40, color='#4285F4', marker='o', edgecolors='black', linewidth=1.5,
```

```
                    alpha=1.0, zorder=3)

    # 在图上标记 top 10 上调蛋白名称（带连接线）
    upregulated_proteins = volcano_df[upregulated].sort_values('log2FC', ascending=False)
    top_5_upregulated = upregulated_proteins.head(5)

    # 用于跟踪已使用的标签位置，避免重叠
    used_label_positions = []

    for idx, (protein, row) in enumerate(top_5_upregulated.iterrows()):
        x_pos = row['log2FC']
        y_pos = row['-log10_p_value']

        # 计算合适的标签位置（根据点的位置决定标签方向）
        if x_pos > 1.5:  # 点在右侧
            label_x = x_pos - 0.8
            label_y = y_pos + 0.15
            ha = 'right'
        elif x_pos < -1.5:  # 点在左侧
            label_x = x_pos + 0.8
            label_y = y_pos + 0.15
            ha = 'left'
        else:  # 点在中间
            label_x = x_pos + 0.6
            label_y = y_pos + 0.2
            ha = 'left'

        # 微调标签位置避免重叠
        for offset in [0, 0.1, -0.1, 0.2, -0.2]:
            test_y = label_y + offset
            overlap = any(abs(test_y - used_y) < 0.12 for used_y in used_label_positions)
```

```
            if not overlap:
                label_y = test_y
                used_label_positions.append(label_y)
                break

        # 提取第一个蛋白名（分号前）
        display_name = protein.split(';')[0] if ';' in protein else protein
        display_name = display_name[:10] + '...' if len(display_name) > 10 else display_name

        # 添加带连接线的标签
        ax.annotate(display_name,
                    xy=(x_pos, y_pos),          # 数据点位置
                    xytext=(label_x, label_y), # 标签位置
                    arrowprops=dict(arrowstyle='-', color='black', alpha=0.8,
                                    linewidth=1.2, shrinkA=3, shrinkB=0),
                    fontsize=7, ha=ha, va='center',
                    fontweight='bold', color='black',
                    zorder=10)

    # 在图上标记 top 10 下调蛋白名称（带连接线）
    downregulated_proteins = volcano_df[downregulated].sort_values('log2FC', ascending=True)
    top_5_downregulated = downregulated_proteins.head(5)

    # 用于跟踪已使用的标签位置，避免重叠（独立于上调蛋白）
    used_label_positions_down = []

    for idx, (protein, row) in enumerate(top_5_downregulated.iterrows()):
        x_pos = row['log2FC']
        y_pos = row['-log10_p_value']

        # 计算合适的标签位置（根据点的位置决定标签方向）
```

```
        if x_pos > 1.5:  # 点在右侧
            label_x = x_pos - 0.8
            label_y = y_pos - 0.15
            ha = 'right'
        elif x_pos < -1.5:  # 点在左侧
            label_x = x_pos + 0.8
            label_y = y_pos - 0.15
            ha = 'left'
        else:  # 点在中间
            label_x = x_pos + 0.6
            label_y = y_pos - 0.2
            ha = 'left'

        # 微调标签位置避免重叠
        for offset in [0, 0.1, -0.1, 0.2, -0.2]:
            test_y = label_y + offset
            overlap = any(abs(test_y - used_y) < 0.12 for used_y in used_label_positions_down)
            if not overlap:
                label_y = test_y
                used_label_positions_down.append(label_y)
                break
        # 提取第一个蛋白名（分号前）
        display_name = protein.split(';')[0] if ';' in protein else protein
        display_name = display_name[:10] + '...' if len(display_name) > 10 else display_name
        # 添加带连接线的标签
        ax.annotate(display_name,
                    xy=(x_pos, y_pos),          # 数据点位置
                    xytext=(label_x, label_y), # 标签位置
                    arrowprops=dict(arrowstyle='-', color='black', alpha=0.8,
                                    linewidth=1.2, shrinkA=3, shrinkB=0),
                    fontsize=7, ha=ha, va='center',
```

```
                    fontweight='bold', color='black',
                    zorder=10)
    # 添加阈值线
    ax.axhline(y=-np.log10(0.05), color='black', linestyle='--', alpha=0.7, linewidth=1.5)
    ax.axvline(x=1, color='black', linestyle='--', alpha=0.7, linewidth=1.5)
    ax.axvline(x=-1, color='black', linestyle='--', alpha=0.7, linewidth=1.5)

    ax.set_xlabel('log2 Fold Change', fontsize=12)
    ax.set_ylabel('-log10(p-value)', fontsize=12)
    ax.set_title(f'{comparison_name.replace("_", " ")}\n({upregulated.sum()} up, {downregulated.sum()} down)',
                 fontsize=14, fontweight='bold')
    ax.grid(True, alpha=0.3)
    ax.tick_params(axis='both', which='major', labelsize=10)

    # 存储所有 upregulated 蛋白用于后续分析
    top_upregulated[comparison_name] = upregulated_proteins

# 添加全局图注
fig.legend(handles=legend_elements,
           loc='center right',
           bbox_to_anchor=(1.25, 0.5),
           fontsize=11,
           title='Legend',
           title_fontsize=14)

plt.suptitle('Volcano Plots: Differential Protein Expression Analysis',
             fontsize=16, fontweight='bold', y=0.98)
plt.tight_layout(rect=[0, 0, 0.85, 1])  # 给右侧图例留出空间
plt.show()

# 保存为 SVG 格式
```

```
output_dir = r"D:\document\学生工作\bioinformation\lab7\plot"
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, "volcano_plots.svg")
fig.savefig(output_path, format='svg', dpi=300, bbox_inches='tight')
print(f"SVG 图片已保存到: {output_path}")

print("火山图生成完成")

# 打印 top 10 上调蛋白结果
print("\n" + "="*60)
print("top 10 MOST UPREGULATED PROTEINS (with protein labels on plots)")
print("="*60)

for comparison_name, top_proteins in top_upregulated.items():
    top_5 = top_proteins.head(5)
    print(f"\n{comparison_name.replace('_', ' ').upper()}")
    print("-" * 40)
    if len(top_5) > 0:
        for i, (protein, row) in enumerate(top_5.iterrows(), 1):
            print(f"{i:2d}. {protein:<20} | log2FC: {row['log2FC']:+.4f} | p-value: {row['p_value']:.2e}")
    else:
        print("   No significantly upregulated proteins found")

print("\n" + "="*60)
print("Cell 2 完成")
print("="*60)

# ============================================================================
# Cell 3: 差异表达统计分析
# ============================================================================
```

```
print("\n" + "="*60)
print("Cell 3: 差异表达统计分析")
print("="*60)

# 重新计算每个比较组的差异表达结果
de_results = {}

for comparison_name, groups in comparisons.items():
    treatment_cols = groups['treatment']
    control_cols = groups['control']

    if len(treatment_cols) == 0 or len(control_cols) == 0:
        continue

    # 获取数据
    treatment_data = all_log2_df[treatment_cols]
    control_data = all_log2_df[control_cols]

    # 计算 fold change (log2FC)
    treatment_mean = treatment_data.mean(axis=1)
    control_mean = control_data.mean(axis=1)
    log2fc = treatment_mean - control_mean

    # 计算 p 值
    p_values = []
    for protein in all_log2_df.index:
        try:
            t_stat, p_val = stats.ttest_ind(
                treatment_data.loc[protein].dropna(),
                control_data.loc[protein].dropna(),
                equal_var=False
```

```
            )
            p_values.append(p_val)
        except:
            p_values.append(np.nan)

    p_values = pd.Series(p_values, index=all_log2_df.index)

    # 创建结果 dataframe
    volcano_df = pd.DataFrame({
        'log2FC': log2fc,
        'p_value': p_values,
        '-log10_p_value': -np.log10(p_values + 1e-10)
    }).dropna()

    de_results[comparison_name] = volcano_df

print(f"差异表达数据准备完成，共 {len(de_results)} 个比较组")

# 生成差异表达统计报告
print("\n 差异表达分析结果:")

for comparison_name, de_df in de_results.items():
    sig_proteins = de_df[(de_df['p_value'] < 0.05) & (abs(de_df['log2FC']) > 1)]
    up_proteins = sig_proteins[sig_proteins['log2FC'] > 0]
    down_proteins = sig_proteins[sig_proteins['log2FC'] < 0]

    print(f"\n{comparison_name}:")
    print(f"  显著差异蛋白总数: {len(sig_proteins)}")
    print(f"  上调蛋白: {len(up_proteins)}")
    print(f"  下调蛋白: {len(down_proteins)}")
```

```
    if len(sig_proteins) > 0:
        print(".3f")
        print(".3f")

        # 保存 Top 差异表达蛋白到文件
        output_dir = r"data/"
        os.makedirs(output_dir, exist_ok=True)

        # Top 10 上调蛋白
        if len(up_proteins) > 0:
            top_up = up_proteins.sort_values('log2FC', ascending=False).head(10)
            top_up.to_csv(os.path.join(output_dir, f"{comparison_name}_top_upregulated.csv"))

        # Top 10 下调蛋白
        if len(down_proteins) > 0:
            top_down = down_proteins.sort_values('log2FC', ascending=True).head(10)
            top_down.to_csv(os.path.join(output_dir, f"{comparison_name}_top_downregulated.csv"))

print("\n 差异表达分析完成")

print("\n" + "="*60)
print("Cell 3 完成")
print("="*60)
```