

生信初学者教程（癌症转录组学）

生信学习者

2024 年 7 月 30 日

目录

图索引	viii	第一部分 基础知识	14
表索引	x	第二章 软件	15
欢迎	1	2.1 R	15
编译方法	1	2.2 Rstudio	15
R 包版本	1	2.3 Linux 系统	15
安装包	3	2.4 其他软件	15
版权	5		
售价	5	第三章 R 语言基础	16
答疑	5	3.1 数据类型	16
前言	7	3.1.1 整型	16
为什么撰写本教程?	7	3.1.2 逻辑型	17
该教程适合什么类型人群?	7	3.1.3 字符型	17
该怎么学习该教程?	8	3.1.4 日期型	17
学习完该教程会获得什么?	8	3.1.5 数值型	17
第一章 介绍	9	3.1.6 复杂数	17
1.1 参考文章	9	3.2 数据结构	17
1.2 HCC 项目流程	9	3.2.1 向量	17
1.3 数据地址	9	3.2.2 矩阵	18
1.4 基础知识	11	3.2.3 数组	18
1.5 数据准备	11	3.2.4 列表	18
1.6 差异分析	11	3.2.5 因子	18
1.7 功能分析	12	3.2.6 数据框	18
1.8 浸润分析	12	3.2.7 ts	18
1.9 标记筛选	12	3.3 特殊值	20
1.10 关联分析	12	3.3.1 缺失值 (NA)	20
1.11 单细胞分析	12	3.3.2 无穷大 (Inf)	20
1.12 撰写文章	13	3.3.3 非数字 (NaN)	20
		3.4 安装 R 包	20
		第四章 数学基础	21
		4.1 描述性统计	21

4.2	推断性统计	21	8.2.2	导入数据	45
4.2.1	假设检验	21	8.2.3	生成 ExpressionSet	45
4.2.2	检验分布	22	8.2.4	输出结果	47
4.2.3	线性回归	22	8.3	GSE14520	48
4.3	机器学习	22	8.3.1	加载 R 包	48
第五章	数据库	23	8.3.2	导入数据	48
5.1	Gene Expression Omnibus	23	8.3.3	生成 ExpressionSet	48
5.2	ICGC Data Portal	23	8.3.4	输出结果	50
5.3	Genomic Data Commons Data Porta	23	8.4	重叠基因	50
			8.4.1	加载 R 包	50
			8.4.2	导入数据	51
			8.4.3	共有基因	51
			8.4.4	生成数据对象	51
			8.4.5	输出结果	53
第二部分	数据准备	24	8.5	总结	53
第六章	数据收集	25	第九章	数据校正	55
6.1	数据分布	25	9.1	加载 R 包	55
6.2	表达谱数据	25	9.2	导入数据	56
6.3	最终数据分布	25	9.3	准备数据	56
6.4	自动下载 GSE14520	26	9.4	ComBat	56
6.5	下载 GSE149614	28	9.5	removeBatchEffect	57
6.6	下载其它数据	28	9.6	Voom SNM	57
第七章	数据预处理	30	9.7	批次效应校正结果比较	58
7.1	LIRI-JP	30	9.8	校正后的结果	61
7.1.1	LIRI-JP 临床表型	30	9.9	输出校正后的结果	64
7.1.2	LIRI-JP 转录组	32	9.10	总结	64
7.2	TCGA-LIHC	33	第十章	数据汇总	69
7.2.1	TCGA-LIHC 临床表型	35	10.1	导入数据	69
7.2.2	TCGA-LIHC 转录组	36	10.2	汇总表格	70
7.3	GSE14520	38	10.3	输出结果	70
7.3.1	GSE14520 临床表型	38	10.4	总结	70
7.3.2	GSE14520 转录组	39			
7.4	总结	40	第三部分	差异分析	72
第八章	数据对象	42	第十一章	差异分析之 limma	73
8.1	LIRI-JP	42	11.1	加载 R 包	73
8.1.1	加载 R 包	42	11.2	读取函数	73
8.1.2	导入数据	42	11.3	导入数据	82
8.1.3	生成 ExpressionSet	43			
8.1.4	输出结果	44			
8.2	TCGA-LIHC	45			
8.2.1	加载 R 包	45			

11.4 差异分析函数	83	16.1 加载 R 包	118
11.5 运行差异分析	85	16.2 导入数据	118
11.6 输出结果	86	16.3 所需函数	119
11.7 总结	87	16.4 运行	121
第十二章 差异结果的火山图	89	16.5 输出结果	122
12.1 加载 R 包	89	16.6 总结	122
12.2 导入数据	89		
12.3 画图函数	90	第五部分 浸润分析	126
12.4 火山图	90		
12.5 输出结果	91	第十七章 免疫浸润细胞	127
12.6 总结	92	17.1 加载 R 包	129
第十三章 差异结果的热图	96	17.2 导入数据	129
13.1 加载 R 包	96	17.3 所需函数	129
13.2 导入数据	96	17.4 运行 ImmuCellAI	130
13.3 画图函数	97	17.5 其他免疫浸润方法	131
13.4 热图	98	17.6 输出结果	132
13.5 输出结果	99	17.7 总结	132
13.6 总结	99		
第四部分 功能分析	101	第十八章 免疫浸润分析	135
第十四章 GO 富集分析	102	18.1 加载 R 包	135
14.1 加载 R 包	102	18.2 导入数据	135
14.2 导入数据	103	18.3 所需函数	136
14.3 所需函数	103	18.4 堆积图	141
14.4 运行	105	18.5 箱线图	142
14.5 输出结果	105	18.6 热图	143
14.6 总结	106	18.7 相关性矩阵图	143
第十五章 KEGG 通路富集分析	108	18.8 输出结果	144
15.1 加载 R 包	108	18.9 总结	144
15.2 导入数据	109		
15.3 所需函数	109	第六部分 标记筛选	146
15.4 运行	112		
15.5 其他可视化方法	113	第十九章 LASSO LR	147
15.6 输出结果	114	19.1 加载 R 包	147
15.7 总结	114	19.2 导入数据	147
第十六章 ssGSEA 富集分析	118	19.3 准备数据	148
		19.4 机器学习特征筛选	148
		19.4.1 数据分割	149
		19.4.2 调参	150
		19.4.3 测试集验证	150
		19.4.4 最终分类模型	151

19.5 标记基因	156	22.6 总结	193
19.6 输出结果	157		
19.7 总结	158		
第二十章 Boruta RF	160	第二十三章 验证特征	196
20.1 加载 R 包	160	23.1 加载 R 包	196
20.2 导入数据	160	23.2 导入数据	197
20.3 准备数据	161	23.3 函数	197
20.4 机器学习特征筛选	162	23.4 重要特征的表达	200
20.4.1 数据分割	163	23.5 ROC 分析	202
20.4.2 基础模型	164	23.6 汇总筛选结果	203
20.4.3 Boruta 特征筛选	164	23.7 输出结果	204
20.4.4 调参	165	23.8 总结	206
20.4.5 最终分类模型	167		
20.4.6 测试集验证	167		
20.5 标记基因	171	第七部分 关联分析	208
20.6 输出结果	172	第二十四章 关联分析	209
20.7 总结	173	24.1 加载 R 包	209
第二十一章 REF SVM	176	24.2 导入数据	209
21.1 加载 R 包	176	24.3 函数	210
21.2 导入数据	177	24.4 重要特征与免疫细胞的相关热图	213
21.3 准备数据	177	24.5 SLC6A8 关联图	214
21.4 机器学习特征筛选	179	24.6 SLC6A8 与特定免疫细胞	214
21.4.1 数据分割	180	24.7 SLC6A8 与其他免疫细胞	215
21.4.2 基础模型	180	24.8 输出结果	216
21.4.3 Recursive Feature Elimination 特征筛选	181	24.9 总结	216
21.4.4 调参	181		
21.4.5 最终分类模型	182	第八部分 单细胞分析	219
21.4.6 测试集验证	183	第二十五章 单细胞数据处理	220
21.5 标记基因	187	25.1 加载 R 包	220
21.6 输出结果	188	25.2 导入数据	220
21.7 总结	188	25.3 原始数据转换成 Seurat 对象	221
第二十二章 交集特征	191	25.4 数据过滤	222
22.1 加载 R 包	191	25.4.1 过滤指标	222
22.2 导入数据	191	25.4.2 过滤处理	224
22.3 重叠的重要特征	192	25.5 输出结果	225
22.4 重要特征的韦恩图	192	25.6 总结	225
22.5 输出结果	193	第二十六章 单细胞数据标准化	227
		26.1 加载 R 包	227
		26.2 导入数据	227

26.3 消除测序深度影响	228	29.8 输出结果	253
26.4 评估细胞周期的影响	228	29.9 总结	254
第二十七章 单细胞聚类分析	232	第九部分 撰写文章	256
27.1 加载 R 包	232	第三十章 框架	257
27.2 导入数据	232	30.1 标题 (title)	257
27.3 确定 PC 纬度	233	30.2 摘要 (abstract)	257
27.4 寻找邻居	233	30.3 介绍 (introduction)	257
27.5 寻找聚类	233	30.4 方法和材料 (methods and materials)	257
27.6 可视化聚类	234	30.5 结果 (results)	258
27.7 输出结果	235	30.6 讨论 (discussion)	258
27.8 总结	235	30.7 结论 (conclusion)	258
第二十八章 单细胞细胞识别	238	第三十一章 结果	259
28.1 加载 R 包	238	31.1 Supplemental Figure1	260
28.2 导入数据	239	31.2 Figure2	260
28.3 细胞簇的标记基因	239	31.3 Figure3	260
28.4 ATC 细胞注释	240	31.4 Supplemental Figure2	263
28.5 scCATCH 细胞注释	240	31.5 Figure4	263
28.6 SingleR 细胞注释	241	31.6 Supplemental Figure3	263
28.7 内部细胞注释	242	31.7 Figure5	263
28.8 重命名簇	243	31.8 Supplemental Figure4	263
28.9 展示分组基因表达	244	31.9 Figure6	263
28.10 美化气泡图	245	31.10 Figure7	263
28.11 单细胞评分	246	31.11 撰写结果	263
28.12 输出结果	246	第三十二章 方法	270
28.13 总结	247	32.1 数据收集和整理	270
第二十九章 核心特征单细胞表达	251	32.2 数据整合	271
29.1 加载 R 包	251	32.3 差异基因分析	271
29.2 导入数据	252	32.4 功能富集分析	271
29.3 细胞类群的 tSNE	252	32.5 免疫浸润分析	271
29.4 核心特征表达分布	252	32.6 候选标记物识别	272
29.5 核心特征点图	253	32.7 诊断 ROC 曲线	272
29.6 SLC6A8 表达分布	253	32.8 单细胞分析	272
29.7 SLC6A8 差异结果	253	32.9 统计方法	272
		第三十三章 讨论	273
		33.1 段落逻辑	273

第三十四章 介绍	275	Functional analysis of DEGs by GO and KEGG enrichment analysis 280
第三十五章 结论和摘要	276	Significant changes between two stages in immune cells by ImmuneCellAI 280
35.1 结论	276	Potential gene biomarkers identified by multiple machine learning approaches 280
35.2 摘要	276	Correlation analysis between SLC6A8 and immune cells . . 280
初稿	277	Expression level of SLC6A8 in single-cell transcriptomic data 280
Title	277	Discussion 280
Abstract	279	Conclusion 280
Introduction	279	Limitation of the study 280
Materials and methods	279	Supplemental Figures 280
Data collection and download	279	
Data Integration	279	
Differential Expression Genes	279	
Functional enrichment analysis	279	
Immune cell infiltration	279	
Potential biomarkers selection	279	
ROC of diagnostic biomarker	279	
Single cell transcriptome data processing and analyzing	279	
Statistical analysis	279	
Results	279	
Identification of DEGs in the HCC early stage and late stage	279	
		References 286
		附录 289
		附录 A 补充知识 289
		A.1 数据库 289
		A.2 R 学习材料 289
		A.3 R 与统计 289

图索引

1.1 HCC 项目流程 (Fig1)	10
4.1 可变性统计变量在数据中的展示	22
12.1 差异基因的火山图 (Fig2-A)	91
31.1 Figure 1: Schematic workflow of this study	259
31.2 Supplemental Figure 1: PCA of multiple types of gene expression matrix for batch removal on merge data cohort (LIHC-US and LIRI-JP).	261
31.3 Figure 2: DEGs between the early-stage group and the late-stage group in HCC merge dataset (LIHC-US and LIRI-JP).	261
a	261
b	261
31.4 Figure 3: Functional analysis of DEGs between the early-stage group and the late-stage group in HCC merge cohort.	262
a	262
b	262
c	262
d	262
31.5 Supplemental Figure 2: GSVA enrichment analysis of gene expression matrix between the HCC early stage and late stage in merge data.	263
31.6 Figure 4: Evaluation and visualization of immune cell infiltration between the HCC early stage and late stage in merge cohort.	264
a	264
b	264
c	264
d	264
31.7 Supplemental Figure 3: Machine learning algorithms to screen potential diagnostic markers from DEGs between the HCC early stage and late stage in merge data.	265
a	265
b	265
c	265
d	265
e	265
f	265
g	265
h	265
31.8 Figure 5: Validation and ROC of SLC6A8 diagnostic marker identified from the merge datasets by multiple machine learning algorithms.	266
a A1	266
b A2	266
c B	266
d D	266

e	C	266
f	E	266
31.9	Supplemental Figure 4: 10 overlapping genes screened from machine learning algorithms were validated in the validation cohort.	267
a		267
b		267
31.10	Figure 6: Correlation between SLC6A8 and 21 immune cells.	267
a		267
b		267
c		267
d		267
31.11	Figure 7: SLC6A8 expression in single-cell GSE149614 transcriptomic dataset.	268
a		268
b		268
c		268
d		268
35.1	Figure 1	279
a	Figure 2A	280
b	Figure 2B	280
35.3	Figure 3A	281
35.4	Figure 3C	281
35.5	Figure 3B	281
35.6	Figure 3D	281
35.7	Figure 4A	282
35.8	Figure 4B	282
35.9	Figure 4C	282
35.10	Figure 4D	282
35.11	Figure 5A1	283
35.12	Figure 5A2	283
35.13	Figure 5B	283
35.14	Figure 5D	283
35.15	Figure 5C	283
35.16	Figure 5E	283
35.17	Figure 6A	284
35.18	Figure 6B	284
35.19	Figure 6C	284
35.20	Figure 6D	284
35.21	Figure 7A	285
35.22	Figure 7B	285
35.23	Figure 7C	285
35.24	Figure 7D	285

表索引

11.1 差异分析 limma 的结果说明	85	17.1 免疫浸润算法比较	128
---------------------------------	----	-------------------------	-----

欢迎

在生物信息学（生信）领域，随着高通量测序技术的不断发展，大量数据涌现，为科研工作者提供了丰富的资源。然而，对于初学者而言，如何从海量的数据中挖掘有价值的信息，并开展一个完整的生信项目，仍然是一个挑战。目前，市面上针对初学者的系统性、以项目为框架的生信教程相对较少，这无疑增加了初学者的学习难度。

鉴于此，结合先前的项目经验，笔者特别为初学者设计了一套基于癌症转录组数据的纯生信文章项目教程。该教程涵盖了项目设计、数据下载、数据处理、数据分析以及结果整理和文章撰写等关键步骤，旨在帮助初学者快速掌握生信项目的全流程操作，提升科研能力。

笔者认为，这套教程对于初学者而言，具有较高的实用价值和指导意义。通过系统学习，您可以更加清晰地了解生信项目的各个环节，掌握必备的技能和知识，为后续的研究工作打下坚实的基础。同时，笔者也相信，通过付费学习，您可以更加珍惜这套教程，更加投入地学习，取得更好的学习效果。

编译方法

本教程是在 RStudio IDE 内使用 [Quarto](#) 编辑的，Quarto 是继[R Markdown](#)之后，一个新的开源的科学和技术发布系统，它基于 [Pandoc](#)支持输出多种格式的书稿，比如 HTML 网页、EPUB 电子书、DOCX 文档和 PDF 便携式文档等。Quarto 吸收了过去 10 年 R Markdown 取得的经验和教训，在书籍写作、创建博客、制作简历和幻灯片等系列场景中支持更加统一的使用语法，一份源文档输出多种格式，使文档内容在不同场景中的迁移成本更低。了解更多 Quarto 特性，请访问 <https://quarto.org/>。

书中 R 包名以粗体表示，如 **knitr** 包，函数名以等宽体表示，如 **plot()**，函数的参数名同理。代码块内注释用 `#` 表示，运行结果每一行开头以 `#>` 标记。本书写作过程中，依赖 [knitr](#) ([Xie 2015](#))、[ggplot2](#) ([Wickham 2016](#)) 和 [lattice](#) ([Sarkar 2008](#)) 等众多 R 包。考虑到要同时支持 DOCX、EPUB、PDF 和 HTML 四种书籍格式，书中使用 **knitr** 包和 **gt** 包制作静态的表格。

R 包版本

在编写和生成这些书籍格式的过程中，使用了多个 R 语言包，并特别关注了这些包的版本信息以确保结果的准确性和兼容性。以下是涉及到的 R 包及其相应版本的详细说明：

©Hua

```

xfun::session_info(packages = c(
  "dplyr", "tidyverse", "showtext", "data.table",
  "Biobase", "sva", "limma", "MicrobiomeAnalysis",
  "SummarizedExperiment", "snm", "ggpubr", "cowplot",
  "gtsummary", "xlsx", "ComplexHeatmap", "circlize",
  "ggplotify", "massdatabase", "clusterProfiler",
  "org.Hs.eg.db", "enrichplot", "GSVA", "corrplot",
  "RColorBrewer", "glmnet", "caret", "ROCR",
  "pROC", "randomForest", "mlbench", "Boruta",
  "Hmisc", "e1071", "ggvenn", "UpSetR",
  "multipleROC", "ggdist", "gghalves",
  "ggpmisc", "ggExtra", "MLmetrics",
  "Seurat", "GseaVis", "ggh4x",
  "ggunchull", "scCATCH", "SingleR",
  "UCell", "rstatix", "viridis",
  "immunedeconv", "ImmuCellAI"),
  dependencies = FALSE)

#> R version 4.3.3 (2024-02-29)
#> Platform: aarch64-apple-darwin20 (64-bit)
#> Running under: macOS Sonoma 14.2
#>
#> Locale: en_US.UTF-8 / en_US.UTF-8 / en_US.UTF-8 / C / en_US.UTF-8 / en_US.UTF-8
#>
#> Package version:
#>   Biobase_2.62.0           Boruta_8.0.0
#>   caret_6.0.94              circlize_0.4.16
#>   clusterProfiler_4.10.1    ComplexHeatmap_2.18.0
#>   corrplot_0.92             cowplot_1.1.3
#>   data.table_1.15.4         dplyr_1.1.4
#>   e1071_1.7.14              enrichplot_1.22.0
#>   ggdist_3.3.2              ggExtra_0.10.1
#>   ggh4x_0.2.8               gghalves_0.1.4
#>   ggplotify_0.1.2           ggpmisc_0.5.6
#>   ggpubr_0.6.0               ggunchull_1.0.1
#>   ggvenn_0.1.10              glmnet_4.1.8
#>   GseaVis_0.1.0              GSVA_1.50.5
#>   gtsummary_1.7.2             Hmisc_5.1.2
#>   ImmuCellAI_0.1.0           immunedeconv_2.1.0
#>   limma_3.58.1               massdatabase_1.0.10

```

```
#> MicrobiomeAnalysis_1.0.3      mlbench_2.1.5
#> MLmetrics_1.1.3                multipleROC_0.1.1
#> org.Hs.eg.db_3.18.0            pROC_1.18.5
#> randomForest_4.7.1.1          RColorBrewer_1.1.3
#> ROCR_1.0.11                   rstatix_0.7.2
#> scCATCH_3.2.2                 Seurat_5.0.3
#> showtext_0.9.7                 SingleR_2.4.1
#> snm_1.50.0                    SummarizedExperiment_1.32.0
#> sva_3.50.0                   tidyverse_2.0.0
#> UCell_2.6.2                   UpSetR_1.4.0
#> viridis_0.6.5                 xlsx_0.6.5
```

这些 R 包的选择和版本信息均经过仔细考量，旨在为用户提供稳定、高效且易于理解的书籍生成体验。请注意，由于 R 语言的生态系统持续发展，这些包的版本信息可能会有所更新。建议用户在实际使用中检查并更新到最新版本的 R 包，以获取最佳性能和最新的功能。

安装包

🔥 安装包

安装 R 包过程可能出现依赖包缺失的情况，需要耐心安装 ~

- 通过 CRAN 安装的 R 包

```
cran_packages <- c("Boruta", "caret", "corrplot",
                  "cowplot", "data.table", "dplyr",
                  "ggplotify", "ggpubr", "glmnet",
                  "gtsummary", "Hmisc", "mlbench",
                  "pROC", "showtext", "RColorBrewer",
                  "ROCR", "randomForest", "xlsx",
                  "devtools", "remotes", "ggExtra",
                  "e1071", "gddist", "ggh4x",
                  "ggpmisc", "ggvenn", "MLmetrics",
                  "rstatix", "scCATCH", "Seurat",
                  "UpSetR", "viridis")

for (i in cran_packages) {

  p_in <- installed.packages()
  p_names <- rownames(p_in)
```

©Hua-

```

if (!i %in% p_names) {
  install.packages(i, dependencies = T)
}
}
```

- 通过 bioconductor 安装的 R 包

```

bio_packages <- c("Biobase", "circlize", "clusterProfiler",
                 "enrichplot", "GSVA", "limma",
                 "org.Hs.eg.db", "snm", "sva",
                 "tidyverse", "UCell")

for (j in bio_packages) {

  p_in <- installed.packages()
  p_names <- rownames(p_in)

  if (!j %in% p_names) {
    BiocManager::install(j)
  }
}
```

- 可能会用到的开发版 R 包

```

remotes::install_github("tidymass/massdatabase")
devtools::install_github("bryandmartin/corncob")
devtools::install_github("HuaZou/MicrobiomeAnalysis",
                       dependencies = c("Depends", "Imports", "LinkingTo"),
                       repos = c("https://cloud.r-project.org/",
                                BiocManager::repositories()))
devtools::install_github('erocoar/gghalves')
devtools::install_github("sajuukLyu/ggunchull", type = "source")
devtools::install_github("junjunlab/GseaVis")
remotes::install_github("cardiomoon/multipleROC")
devtools::install_github('dviraran/SingleR')
remotes::install_github("omnideconv/immunedeconv")
```

- 可能会用到的需要手动下载的 R 包

```
install.packages("ImmucellAI_0.1.0.tar.gz", repos = NULL, type = "source")
```

版权

首先，我衷心感谢您对本书的关注与支持。本书是我倾注心血与智慧的结晶，每一页都承载着我对知识的尊重与热爱。

禁止任何形式的盗版行为。盗版不仅侵犯了我的合法权益，更是对作者创作成果的不尊重。我呼吁广大读者自觉抵制盗版，选择正规渠道购买正版书籍，共同维护一个健康、有序的版权环境。

购买后的书籍属于您的个人财产，但您不得将其进行二次销售或用于商业用途。请在使用本书内容时，遵守相关的版权法律法规，不得擅自复制、传播、修改或用于其他未经授权的用途。

知识产权是创新的重要基石，也是推动社会进步的重要力量。让我们携手共同尊重和保护知识产权，为构建一个更加美好的未来贡献力量。

再次感谢您的关注与支持，祝您生活愉快！

售价

本教程整体售价 **998** 元。可通过微信交易。本产品一经售出，不接受任何形式的退款申请。请大家在购买前仔细考虑，确保您已充分了解教程的内容并认同交易规则。购买成功后，您将获得一个专属下载链接，链接中包含以下丰富的学习材料：

- 教程说明：**HTML/PDF 格式的详细、系统的教程说明，旨在引导您逐步掌握相关知识，并帮助您解决在学习过程中可能遇到的问题。
- 配套代码：**与教程说明一一对应，确保您在学习理论知识的同时，能够通过实践加深对知识点的理解。代码质量高，注释清晰，易于理解和使用。
- 配套数据：**本教程所需要用到的所有数据，需注意数据所处的目录位置。

关注微信公众号**生信学习者**或添加 **Lavender_George**，在菜单栏找到购买教程链接，付费后即可获得所有材料的下载链接。

最后温馨提示。请在购买前确认您的支付账户余额充足，以确保交易顺利完成。购买成功后，请务必妥善保管您的下载链接（**有效期 7 天，在有效期内若链接有问题，请随时和我联系**），以便随时访问和下载所需材料。

答疑

感谢您对本书的关注与购买。在此，我需要明确说明关于本书购买后的答疑服务相关事宜。

本次出售的教程仅包含书籍本身的内容，即书籍中的文字、图片等所呈现的知识与信息，不包括提供专门的答疑解惑服务。尽管我未提供专门的答疑解惑服务，但考虑到读者的学习需求，对于相对简单且不占据我过多时间的问题，我将酌情提供免费答疑。这类问题通常指那些不涉及复杂分析、计算或深入讨

论的问题，且能够迅速得到解答。如有需要，您可以通过[我的微信号（Lavender_George）](#) 提出您的问题。我会尽快查看并回复您的问题，但请注意，由于时间和精力的限制，我无法对所有问题都提供即时的详细解答。

请读者朋友们在购买前仔细阅读以上说明，确保您已了解购买服务的具体内容。鼓励您通过自主学习和查阅相关资料等方式解决学习中遇到的问题。如有需要，您还可以考虑购买我提供的其他增值服务，以获得更专业的指导和帮助。

前言

为什么撰写本教程？

关于生物信息学入门的难易程度，这是一个常被问及的问题。从我个人的经历和观察来看，生物信息学领域确实具有一定的挑战性。这一领域不仅融合了生物学、计算机科学、统计学等多个学科的知识，而且要求对这些知识有深入的理解和熟练的应用。然而，值得注意的是，尽管生物信息学整体上可能显得复杂，但只要能够专注于自己的特定领域，并具备扎实的基础数据分析能力，那么它也可以变得相对简单。通过系统地学习相关知识和技能，逐步积累实践经验，可以逐步掌握生物信息学的核心方法和技术，从而更好地应对各种挑战。因此，对于初学者而言，不必过分担心生物信息学的复杂性。只要保持耐心和恒心，专注于自己的领域，并不断提升自己的数据分析能力，就能够逐步掌握这一领域的知识和技能，实现自我提升和成长。

生物信息学是一门跨学科的科学领域，它结合了生物学、计算机科学、数学和统计学等多个学科的知识和技术，旨在理解生物学系统中的生物学现象和问题。生物信息学的主要任务包括收集、存储、管理、分析和解释生物学数据，以及开发和应用相关的计算工具和方法。这些数据可以包括基因组序列、蛋白质结构、基因表达数据、代谢组数据等。生物信息学在基础研究、生物医学研究、药物研发、农业和环境保护等领域都发挥着重要作用。

该教程适合什么类型人群？

本教程是专为那些已经具备 R 语言和统计基础的初学者精心编写的，它以一个简单而完整的项目为框架，为学习者提供了一份深入浅出的生物信息学入门材料。教程内容广泛而全面，涵盖了生物信息学领域的众多核心知识点，旨在帮助初学者建立起扎实的理论基础和实践技能。然而，鉴于生物信息学领域的复杂性和跨学科性，本教程所涉及的知识领域较为广泛，初学者可能需要在学习的过程中展现出一定的自学能力。通过深入探索教程中提及的知识点，并主动查找相关资料，初学者可以更加全面地理解和掌握这些知识背后的原理和应用，从而不断提升自己的生物信息学技能。因此，鼓励学习者保持积极的学习态度，勇于探索和挑战，通过不断的学习和实践，逐步成为生物信息学领域的专业人士。

该怎么学习该教程？

本教程以项目数据分析和文章撰写的视角出发，旨在详细解析数据分析的目的以及如何科学合理地解释所得结果。教程将按照逻辑顺序，逐步引导读者深入理解数据分析的各个环节，并阐述每一步结果背后的生物学意义。同时，为了帮助读者更好地理解统计方法的应用，教程还简要介绍了相关统计方法的基本原理。对于初学者而言，本教程提供了清晰的章节结构，可以按照章节顺序逐步学习。每一章节都围绕一个核心主题展开，确保读者能够系统地掌握相关知识。当然，如果读者对某个章节的内容已经有所了解，也可以自由选择跳过，直接进入下一个章节的学习。在学习过程中，如果遇到不理解的问题或难点，读者不必过于担忧。现代网络技术的发展为提供了丰富的资源，可以借助网络上的 AI 工具、在线论坛、教程视频等多种方式进行求助。这些资源能够帮助快速解决疑惑，确保学习过程的顺畅进行。

学习完该教程会获得什么？

本教程深入解析了文章发表的完整流程，从项目的构思、数据的分析到结果的解读，再到最终文章的撰写，每一步都进行了详尽的阐述。它不仅为初学者提供了一次纯生物信息学文章创作过程的实践机会，还帮助他们初步掌握了这一领域的研究方法和技巧。通过学习本教程，读者能够全面了解文章发表的各个环节，并能够在未来的研究中灵活应用所学的知识和方法。此外，教程中使用的数据和代码也具有一定的通用性，读者可以将其应用于其他类似的项目中，进一步拓展自己的研究领域。

第一章 介绍

1.1 参考文章

在生物信息学的助力下，科研人员不断探索着癌症转录组学的奥秘，以期发现能够早期诊断、指导治疗和预测疾病进程的生物标志物。本教程参考已发表的文章 ([Yang 等 2023](#)) 的研究方法，该文章深入探讨了慢性阻塞性肺病 (COPD) 晚期的潜在诊断基因生物标志物，为科研人员提供了宝贵的经验和启示。

1.2 HCC 项目流程

⚠ Pay Attention!!!

该文章已发表了，请勿使用该初稿投递任何期刊，它仅作为学习的参考材料。

本书围绕癌症生信实战工作流分为以下几个部分：

- (1) 基础知识。
- (2) 数据准备。
- (3) 差异分析。
- (4) 功能分析。
- (5) 浸润分析。
- (6) 标记筛选。
- (7) 关联分析。
- (8) 单细胞分析。
- (9) 撰写文章。

1.3 数据地址

为了方便用户进行本教程中的生物信息学分析，已将所需的所有输入数据和预期的输出数据整理并上传至百度网盘。用户可以通过访问指定的百度网盘链接，下载所需的数据集，以便在本地环境中进行后续的分析和验证。请确保在下载和使用数据时，遵循相关的版权和引用规定。

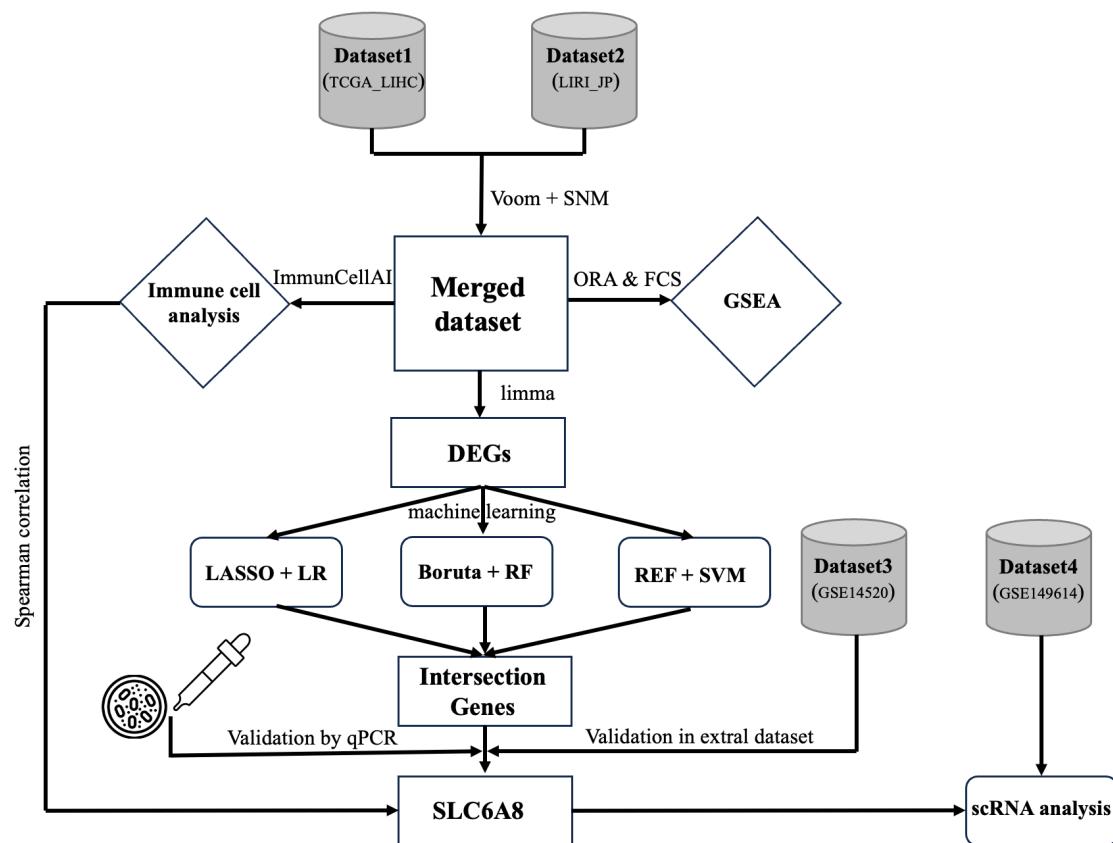


图 1.1: HCC 项目流程 (Fig1)

⚠ Pay Attention!!!

购买了该教程的用户，自动获得本教程所需要的所有数据。

1.4 基础知识

该部分旨在为初学者提供生物信息学分析所需的基本工具、语言、数学及数据库知识的概览。以下是各部分的简要介绍：

1. 软件工具

本教程推荐使用 R 和 RStudio 作为主要的生物信息学分析工具。

2. R 语言基础

在进行生物信息学分析之前，掌握 R 语言的基础知识是至关重要的。

3. 数学基础

数学是生物信息学分析的核心。

4. 数据库基础

生物信息学分析离不开数据的支持。

1.5 数据准备

该部分详细阐述了一系列数据处理流程，旨在确保数据质量，为后续的深入分析奠定坚实基础。以下是对各个步骤的详细书面化处理：

1. 数据收集

2. 数据预处理

3. 数据对象转换

4. 数据校正

5. 数据汇总

1.6 差异分析

1. 差异分析 limma

2. 火山图

3. 热图

1.7 功能分析

1. GO term 富集分析
2. KEGG pathway 富集分析
3. GSVA 富集分析

1.8 浸润分析

1. 免疫浸润分析
2. 免疫细胞数据分析

1.9 标记筛选

在该部分中，我们采用了三种不同的特征筛选与机器学习算法模式来识别关键的标记基因。

1. LASSO+LR
2. Boruta+RF
3. REF+SVM
4. 交集特征
5. 验证特征

1.10 关联分析

通过关联分析核心特征与免疫浸润细胞，我们可以进一步理解这些特征在肿瘤微环境中的功能和作用，以及它们如何影响免疫浸润细胞的数量和功能。

1.11 单细胞分析

在该部分中，我们对原始单细胞转录组数据进行了详尽的处理和分析。

1. 单细胞数据处理
2. 单细胞数据标准
3. 单细胞聚类分析
4. 单细胞细胞识别

5. 核心特征单细胞表达

1.12 撰写文章

第一部分

基础知识

第二章 软件

本书是使用 R 语言编写的教程，用户需要下载 R 和 RStudio 软件用于进行分析。

2.1 R

R 语言是一种免费的统计计算和图形化编程语言，是一种用于数据分析和统计建模的强大工具。它具有丰富的统计分析函数库和数据处理功能，可以进行数据清洗、数据可视化、统计建模等多种操作。

2.2 Rstudio

RStudio 是一个集成开发环境（IDE），专门用于编写和运行 R 语言代码。它提供了丰富的功能，包括代码编辑器、数据查看器、图形输出窗口、调试器等，使得 R 语言的使用更加方便和高效。

2.3 Linux 系统

Linux 是一个开源的类 Unix 操作系统，具有多用户、多任务、稳定、安全等特点。它由 Linux 内核及其他开源软件组成。Linux 广泛应用于服务器、桌面和嵌入式系统中。

2.4 其他软件

- 精修图片：Adobe Illustrator。

第三章 R 语言基础

R 语言是一种用于统计计算和图形展示的编程语言和软件环境，广泛应用于数据分析、统计建模和数据可视化。1991 年：R 语言的最初构想源自新西兰奥克兰大学的统计学家 Ross Ihaka 和 Robert Gentleman。他们受到 S 语言（由 John Chambers 等人在贝尔实验室开发的统计编程语言）的启发，开始开发一个自由、开源的统计编程环境。1995 年：R 语言首次发布。Ross Ihaka 和 Robert Gentleman 将 R 的代码发布到 statlib（一个用于统计软件和数据的在线服务）上。1997 年：R 的代码以 GNU 通用公共许可证 (GPL) 发布，正式成为开源项目。这一决定吸引了许多开发者和统计学家的兴趣和贡献。1997 年：为了更好地管理和推动 R 的发展，R 核心开发团队 (R Core Team) 成立。这个团队负责 R 语言的维护和新功能的开发。1997 年：The Comprehensive R Archive Network (CRAN) 成立，成为 R 软件包和文档的主要分发平台。CRAN 的建立大大促进了 R 语言的传播和使用。2000 年代：R 语言在学术界和研究机构中迅速普及，成为数据分析和统计研究的标准工具之一。CRAN 上的软件包数量快速增长，涵盖了广泛的统计方法和数据分析领域。2010 年代：R 语言的影响力进一步扩展到工业界、金融业、生物信息学等多个领域。随着大数据和数据科学的兴起，R 成为数据科学家和分析师的重要工具。2013 年：RStudio 公司成立，开发了 RStudio 集成开发环境 (IDE)，极大地提升了 R 语言的开发体验和用户友好性。2015 年：R Consortium 成立，这是一个由企业和研究机构组成的非营利组织，旨在支持 R 语言的发展和推广。R 语言仍在不断发展，社区的活跃和丰富的软件包生态系统确保了其在数据科学和统计计算领域的持续重要性。现代 R 的发展还包括对大数据处理、高性能计算和机器学习等领域的支持。

数据类型有整型、逻辑型、字符型、日期型、数值型（单、双精度浮点型）等。

数据结构有向量、矩阵、数组、列表和数据框等。

3.1 数据类型

数据类型是指数据的分类和属性，决定了数据在内存中的存储方式以及可以对数据执行的操作

3.1.1 整型

整数 (Integer)：整数类型的数据，可以用 L 后缀来标识。例如：42L。

`c(1L, 2L)`

```
[1] 1 2
```

3.1.2 逻辑型

用于表示布尔值，只有两个可能的取值：TRUE 或 FALSE。可以用于条件判断和逻辑运算。

```
c(TRUE, FALSE)
```

```
[1] TRUE FALSE
```

3.1.3 字符型

用于表示文本数据，字符串用双引号或单引号括起来。例如：“Hello”，‘World’。

```
c("A", "B")
```

```
[1] "A" "B"
```

3.1.4 日期型

```
c(as.Date("2022-01-01"), as.Date("2022-01-02"))
```

```
[1] "2022-01-01" "2022-01-02"
```

3.1.5 数值型

双精度浮点数 (Double)：默认的数值类型，用于表示实数。例如：3.14, 42.0。

```
c(1, 1.2)
```

```
[1] 1.0 1.2
```

3.1.6 复杂数

用于表示复数，形式为实部加上虚部，例如：2 + 3i。

3.2 数据结构

3.2.1 向量

R 中最基本的数据结构，可以包含相同类型的元素。向量可以是数值、字符、逻辑或复数。例如：c(1, 2, 3), c("a", "b", "c")。

3.2.2 矩阵

矩阵是二维的数组，只能包含相同类型的数据，通常用于线性代数计算。例如：`matrix(1:6, nrow = 2, ncol = 3)`。

3.2.3 数组

数组是多维的矩阵，可以包含相同类型的数据。例如：`array(1:8, dim = c(2, 2, 2))`。

3.2.4 列表

列表是一种通用的数据类型，可以包含不同类型的数据。列表中的元素可以是向量、矩阵、数据框，甚至是另一个列表。例如：`list(1, "a", TRUE, c(1, 2, 3))`。

3.2.5 因子

用于表示分类数据，存储的是整数向量以及对应的分类水平（labels）。因子常用于统计建模中。例如：`factor(c("male", "female", "female", "male"))`。

3.2.6 数据框

数据框是二维的表格数据结构，每列可以包含不同类型的数据，类似于数据库表或 Excel 工作表。数据框是数据分析中常用的结构。例如：`data.frame(Name = c("Alice", "Bob"), Age = c(25, 30))`。

3.2.7 ts

`ts` 类型用于表示时间序列数据，是继承自数组类型的。给定数据、采样初始时间、采样频率的情况下，利用内置的函数 `ts()` 构造一个 `ts` 类型的分钟级的时间序列对象。

```
x <- ts(
  data = rnorm(100),
  start = c(2017, 1),
  frequency = 365.25 * 24 * 60,
  class = "ts", names = "Time_Series"
)
```

`ts()` 函数的 `start` 和 `frequency` 参数很关键，前者指定了时间单位是天，后者指定每个时间单位下的数据点的数量。其中 365.25 是因为每隔 4 年有 366 天，平均下来，每年算 365.25 天。每隔 $1 / (24 * 60)$ 天（即 1 分钟）采样一个点。如果初始时间不是从一年的第 1 分钟开始，而是从此时此刻 2023-01-31

10:43:30 CST 开始，则可以换算成今年的第 $30 * 24 * 60 + 9 * 60 + 43 = 43783$ 分钟，则 `Start = c(2023, 43783)`。

以数据集 `x` 为例，它是一个 `ts` 类型的时间序列数据对象。时间序列对象有很多方法，如函数 `class()`、`mode()` 和 `str()` 分别可以查看其数据类型、存储类型和数据结构。

```
# 数据类型
class(x)

[1] "ts"

# 存储类型
mode(x)

[1] "numeric"

# 数据结构
str(x)

Time-Series [1:100] from 2017 to 2017: -0.3285 -1.3647 -0.8494 -1.1065 0.0648 ...
```

函数 `start()` 和 `end()` 查看开始和结束的时间点。

```
c(start(x), end(x))

[1] 2017     1 2017   100
```

函数 `time()` 可以查看在以上时间区间的划分。

```
time(x)

Time Series:
Start = c(2017, 1)
End = c(2017, 100)
Frequency = 525960
[1] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
[16] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
[31] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
[46] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
[61] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
[76] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
[91] 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017
```

函数 `tsp()` 可以查看其期初、期末和周期。

```
tsp(x)

[1] 2017 2017 525960
```

3.3 特殊值

3.3.1 缺失值 (NA)

表示缺失数据，在数据处理中需要特别注意。例如: `c(1, 2, NA, 4)`。

3.3.2 无穷大 (Inf)

用于表示正无穷大和负无穷大。例如: $1/0$ 会返回 `Inf`, $-1/0$ 会返回`-Inf`。

3.3.3 非数字 (NaN)

表示不是一个数字 (Not a Number), 通常在数值运算未定义时返回。例如: $0/0$ 会返回 `NaN`。

3.4 安装 R 包

- 从特定 CRAN 镜像安装: 清华大学的镜像网站安装指定的扩展包:

```
options(repos=c(CRAN="https://mirror.tuna.tsinghua.edu.cn/CRAN/"))
install.packages("devtools")
```

- 从 Github 安装扩展包: 扩展包没有在 CRAN 系统提供

```
if (!requireNamespace(c("remotes", "devtools"), quietly=TRUE)) {
  install.packages(c("devtools", "remotes"))

}

devtools::install_github("HuaZou/MicrobiomeAnalysis",
  dependencies = c("Depends", "Imports", "LinkingTo"),
  repos = c("https://cloud.r-project.org/",
  BiocManager::repositories()))
```

- 从 BioConductor 安装扩展包: 扩展包没有在 CRAN 系统提供

```
options(repos=c(CRAN="https://mirror.tuna.tsinghua.edu.cn/CRAN/"))
if (!requireNamespace("BiocManager", quietly = TRUE)) {
  install.packages("BiocManager")
}

options(BioC_mirror="https://mirrors.tuna.tsinghua.edu.cn/bioconductor")
BiocManager::install(c("Biostrings"))
```

第四章 数学基础

本教程的核心在于数据分析，因此初学者需要具备一定的数学基础。在这里，简要介绍一些基础数学概念，包括描述性统计、推断性统计以及机器学习等内容。特别地，对于机器学习基础的学习至关重要，例如如何评估分类器的性能优劣等方面。

4.1 描述性统计

描述统计是统计学的一个分支，它涉及有意义和简洁地总结、组织和呈现数据。它侧重于描述数据集的主要特征，而不会对其进行任何概括或推断。

描述性统计的主要目标是提供清晰而简洁的数据摘要，使研究人员或分析人员能够深入了解数据集中的模式、趋势和分布。这种总结通常包括集中趋势（例如，平均值、中位数、众数）、离散度（例如，范围、方差、标准差）和分布形状（例如，偏度、峰度）等度量。

4.2 推断性统计

推断统计是统计学的一个分支，旨在通过对样本数据的分析，推断出关于总体的一般性结论。与描述统计不同，描述统计主要集中于对已有数据的总结和描述（如计算平均值、中位数、标准差等），而推断统计则试图通过样本数据来推测和预测总体的特性。

4.2.1 假设检验

在推断统计中，假设检验是一个重要的工具，用于判断一个关于总体参数的假设是否成立。以下是关于假设检验的详细解释，包括其一般步骤和显著性 p 值的含义

- **假设检验的定义：**

假设检验是关于总体参数的一个陈述。在一个假设检验问题中，通常有两个互补的假设，称为原假设 (H_0) 和备择假设 (H_1)。原假设是要检验的假设，备择假设则是与原假设相反的假设。假设检验的目的是基于样本数据来决定是否拒绝原假设。

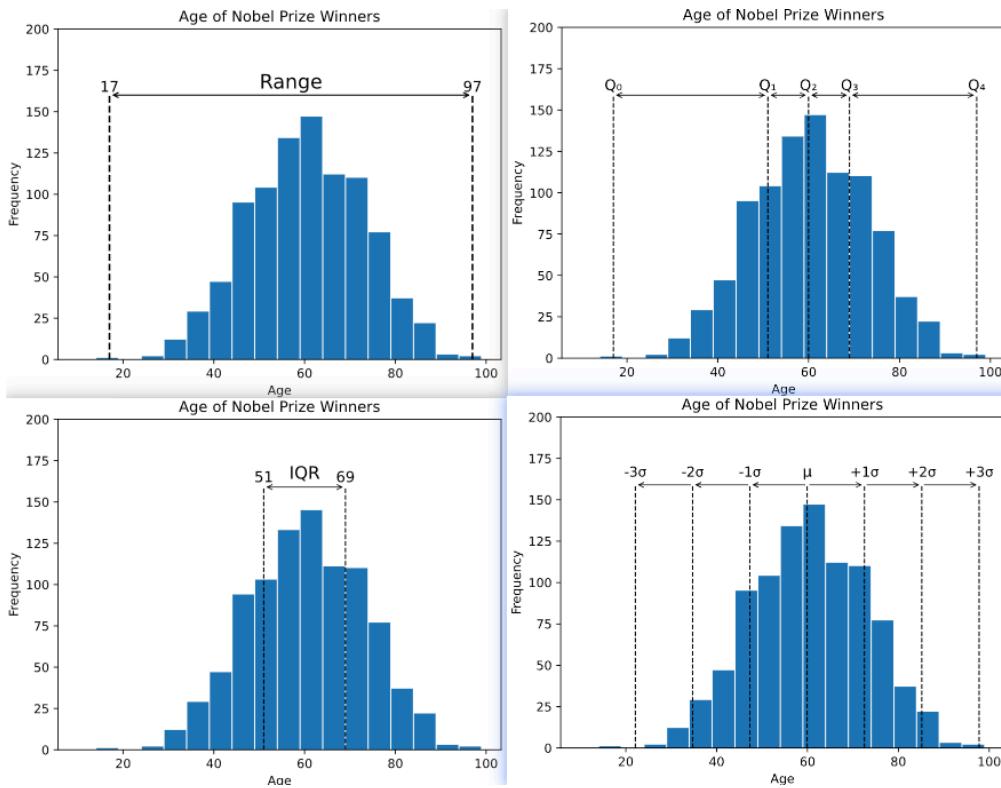


图 4.1: 可变性统计变量在数据中的展示

4.2.2 检验分布

假设检验的几大分布是指 Z 检验分布、T 检验分布以及卡方检验分布和 F 分布，它们在统计学中各自有其特定的应用场景和使用方法。

4.2.3 线性回归

线性回归是一种统计学上分析的方法，用于确定两种或两种以上变量间相互依赖的定量关系。它通常用于预测一个或多个自变量（特征值）的变化如何影响因变量（目标值）。线性回归模型假设因变量（Y）是一个或多个自变量（X）的线性组合，并加上一个误差项。

4.3 机器学习

机器学习是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。它专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。机器学习是人工智能的核心，是使计算机具有智能的根本途径。

第五章 数据库

5.1 Gene Expression Omnibus

Gene Expression Omnibus (GEO) [数据库](#)是一个公共基因表达数据存储库，由美国国家生物技术信息中心 (NCBI) 维护。

5.2 ICGC Data Portal

ICGC (International Cancer Genome Consortium) [Data Portal](#) 是一个旨在为癌症基因组学研究提供数据和资源的在线平台。

5.3 Genomic Data Commons Data Porta

Genomic Data Commons (GDC) [Data Portal](#) 是由美国国家癌症研究所 (NCI) 维护的一个数据平台，旨在为癌症基因组学研究提供数据和工具。

第二部分 数据准备

第六章 数据收集

在确定研究疾病为肝细胞癌 (**Liver Hepatocellular Carcinoma: HCC**) 后，系统地进行了文献回顾，专注于搜索与 HCC 相关的荟萃分析文章，以获取该领域的研究动态和已有成果。为了支持的研究，通过在线资源检索并下载了必要的数据集。具体而言，利用了国际癌症基因组协作组 (ICGC) 数据门户[ICGC Data Portal](#)（参考小节 5.2），美国国家癌症研究所的基因组数据公共库[GenomicDataCommonsDataPorta](#)（参考小节 5.3），以及基因表达综合数据库[GeneExpressionOmnibus](#)（参考小节 5.1）这三个主要的数据仓库。

6.1 数据分布

四个 HCC 转录组数据集分别是 LICA-FR, LIRI-JP, LIHC-US/TCGA-LIHC 和 GSE14520，一个 HCC 单细胞数据集是 GSE149614。

6.2 表达谱数据

针对上述 HCC 数据集进行表达谱的下载，分别获得了它们的不同类型的表达谱数据：

- LICA-FR: France (Tumor TNM stage)
- LIRI-JP: Japan (Tumor TNM stage)
- LIHC-US/TCGA-LIHC: TCGA (Tumor TNM stage)
- GSE14520: GEO (Tumor TNM stage)
- GSE149614: GEO (Tumor TNM stage)

6.3 最终数据分布

在仔细过滤和整理收集到的表型数据后，

- LICA-FR (drop) : France (TPM normalization)
- LIRI-JP: Japan (TPM normalization)

- LIHC-US/TCGA-LIHC: TCGA (TPM normalization)
- GSE14520: GEO (The chip data were standardized based on a robust multichip average method)

6.4 自动下载 GSE14520

为了简化数据获取流程并避免重复下载相似的数据，开发了一个 R 脚本，用于自动从 *Gene Expression Omnibus (GEO)* 下载数据集。

```
# 加载R包
library(GEOquery)
library(tidyverse)
library(stringr)
library(optparse)
library(convert)
library(idmap1)

# 设置参数
GEO_name <- "GSE14520"
GPL_number <- "GPL571"
GPL_number2 <- "GPL3921"
Array_type <- "array"
dir <- "./"

# clinical and expression profile
gset <- getGEO(GEO = GEO_name,
                destdir = dir,
                AnnotGPL = F,
                getGPL = F)
phen <- pData(gset[[1]])
prof <- exprs(gset[[1]])

# output
outdir <- paste0(dir, "/", GEO_name, "_process/")
if (!dir.exists(outdir)) {
  dir.create(outdir)
}

phen_origin <- paste0(outdir, GEO_name, "_clinical_origin.csv")
phen_process <- paste0(outdir, GEO_name, "_clinical_post.csv")
```

```
write.csv(phen, file = phen_origin, row.names = F)
write.csv(phen_post, file = phen_process, row.names = F)

prof_origin <- paste0(outdir, GEO_name, "_profile_origin.tsv")
prof_process <- paste0(outdir, GEO_name, "_profile_post.tsv")
write.table(data.frame(prof) %>% rownames_to_column("GeneID"),
            file = prof_origin, row.names = F, quote = F, sep = "\t")
write.table(prof_post %>% rownames_to_column("GeneID"),
            file = prof_process, row.names = F, quote = F, sep = "\t")

probe2gene_name <- paste0(outdir, GPL_number, "_probe2gene_table.tsv")
write.table(probe2gene, file = probe2gene_name,
            row.names = F, quote = F, sep = "\t")

ExprSet_name <- paste0(outdir, GEO_name, "_GeneExprSet.RDS")
saveRDS(ExprSet_object, file = ExprSet_name)

message("Congrats, Program Ended without problems")
```

上述代码由以下几部分组成：

- 加载 R 包；
- 设置 GSE 和 GPL 编号，其中 GPL 是 GSE 对应的平台号，通常适用于芯片探针数据；
- 进行基因 ID 转换和表型数据筛选；
- 将数据转换为 ExpressionSet 数据对象，该类型数据包含表型数据和表达谱数据，便于后续的下游分析。

最后获得了以下数据：

```
GSE14520/
├── GPL3921.soft.gz
├── GPL571.soft.gz
├── GSE14520-GPL3921_series_matrix.txt.gz
├── GSE14520-GPL571_series_matrix.txt.gz
├── GSE14520_Extra_Supplement.txt
└── GSE14520_process
    ├── GPL571_probe2gene_table.tsv
    ├── GSE14520_GeneExprSet.RDS
    ├── GSE14520_clinical_origin.csv
    ├── GSE14520_clinical_post.csv
    └── GSE14520_profile_origin.tsv
```

```

└── GSE14520_profile_post.tsv
├── download.R
└── work.sh

```

6.5 下载 GSE149614

GSE149614 数据集是单细胞数据集

```

GSE149614_scRNA/
├── 41467_2022_32283_MOESM1_ESM.pdf
├── 41467_2022_32283_MOESM3_ESM.zip
├── A single-cell atlas of the multicellular ecosystem of primary and metastatic hepatocellular carcinom
├── GSE149614_HCC.metadata.updated.txt
├── GSE149614_HCC.scRNAseq.S71915.count.txt
├── GSE149614_HCC.scRNAseq.S71915.count.txt.gz
├── GSE149614_HCC.scRNAseq.S71915.normalized.txt.gz
└── SupplementaryData
    ├── Supplementary Data 1.xlsx
    ├── Supplementary Data 10.xlsx
    ├── Supplementary Data 11.xlsx
    ├── Supplementary Data 2.xlsx
    ├── Supplementary Data 3.xlsx
    ├── Supplementary Data 4.xlsx
    ├── Supplementary Data 5.xlsx
    ├── Supplementary Data 6.xlsx
    ├── Supplementary Data 7.xlsx
    ├── Supplementary Data 8.xlsx
    └── Supplementary Data 9.xlsx

```

6.6 下载其它数据

最后获得了以下数据:

```

LIRI-JP/
├── donor.LIRI-JP.tsv
├── exp_seq.LIRI-JP.tsv
├── sample.LIRI-JP.tsv
└── specimen.LIRI-JP.tsv

```

TCGA_LIHC/
└── TCGA-LIHC.htseq_counts.tsv
└── TCGA-LIHC.htseq_fpkm.tsv
└── TCGA-LIHC_clinical_origin.csv

第七章 数据预处理

在对上述数据集进行数据预处理时，主要进行了两个关键步骤的清洗工作：

- 临床表型数据清洗：对临床表型数据进行了全面的检查和清理。首先，检查了数据的完整性，确保没有缺失值或异常值对后续分析造成影响。其次，核对了数据的准确性。最后，对临床表型数据进行了必要的标准化处理，以确保不同数据集之间的数据可以相互比较和整合。
- 表达谱数据清洗：在表达谱数据的清洗过程中，同样注重数据的完整性和准确性。首先，检查了基因表达数据是否存在缺失值或异常值。其次，对表达谱数据进行了质量控制分析。

7.1 LIRI-JP

下载到的数据有如下：

- 病人的数据（临床数据）：donor.LIRI-JP.tsv.gz
- 表达数据（测序数据）：exp_seq.LIRI-JP.tsv.gz
- 样品信息：sample.LIRI-JP.tsv.gz
- 突变数据：simple_somatic_mutation.open.LIRI-JP.tsv.gz
- 实验处理数据：specimen.LIRI-JP.tsv.gz
- 结构变异数据：structural_somatic_mutation.LIRI-JP.tsv.gz

7.1.1 LIRI-JP 临床表型

本节将生成临床患者临床数据以及样本对应的表型数据。

7.1.1.1 加载 R 包

使用 `rm(list = ls())` 清空环境内变量

```
library(tidyverse)
```

```
rm(list = ls())
options(stringsAsFactors = F)
```

©Hua

```
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#1F78B4", "#EE2B2B")
```

7.1.1.2 导入数据

```
LIRIJP_metadata <- read.csv("./data/LIRI-JP/donor.LIRI-JP.tsv", sep = "\t")
LIRIJP_sample <- read.csv("./data/LIRI-JP/sample.LIRI-JP.tsv", sep = "\t")
LIRIJP_specimen <- read.csv("./data/LIRI-JP/specimen.LIRI-JP.tsv", sep = "\t")
```

7.1.1.3 清洗临床数据

清洗临床数据

```
LIRIJP_clin_columns <- c("icgc_donor_id", "project_code", "submitted_donor_id",
                           "donor_sex", "donor_vital_status", "donor_age_at_diagnosis",
                           "donor_tumour_stage_at_diagnosis", "donor_survival_time")

LIRIJP_clinic_data <- LIRIJP_metadata

head(LIRIJP_clinic_data)
```

7.1.1.4 清洗实验处理数据

```
LIRIJP_spe_columns <- c("icgc_specimen_id", "project_code", "submitted_specimen_id",
                           "icgc_donor_id", "submitted_donor_id", "specimen_type")

LIRIJP_specimen_data <- LIRIJP_specimen

head(LIRIJP_specimen_data)
```

7.1.1.5 清洗样品信息数据

- 数据清洗: 样品信息

```
LIRIJP_sam_columns <- c("icgc_sample_id", "project_code", "submitted_sample_id",
                           "icgc_specimen_id", "submitted_specimen_id", "icgc_donor_id",
                           "submitted_donor_id")

LIRIJP_sample_data <- LIRIJP_sample
```

```
head(LIRIJP_sample_data)
```

7.1.1.6 输出结果

输出结果: 将上述结果以 csv 格式输出到结果目录

```
if (!dir.exists("./data/result/metadata/")) {
  dir.create("./data/result/metadata/", recursive = TRUE)
}

write.csv(LIRIJP_clinic_data, "./data/result/metadata/LIRI-JP-clinical.csv", row.names = F)
write.csv(LIRIJP_sample_data, "./data/result/metadata/LIRI-JP-sample.csv", row.names = F)
write.csv(LIRIJP_specimen_data, "./data/result/metadata/LIRI-JP-specimen.csv", row.names = F)
```

7.1.2 LIRI-JP 转录组

基因表达数据需要进行基因 ID 转换以及冗余基因的过滤，最后生成符合要求的输出结果。

7.1.2.1 加载 R 包

```
library(tidyverse)
library(data.table)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)
```

7.1.2.2 导入数据

```
LIRIJP_genexp <- fread("./data/LIRI-JP/exp_seq.LIRI-JP.tsv")
LIRIJP_geneID <- fread("./data/GeneIDMap/human_gene_all.tsv")

head(LIRIJP_geneID)
```

7.1.2.3 数据清洗

数据清洗

```
LIRIJP_gene_data <- LIRIJP_genexp %>%
  dplyr::select(icgc_specimen_id, gene_id,
                normalized_read_count, raw_read_count) %>%
  dplyr::mutate(gene_id = gsub("\\.\\d+", "", gene_id))

head(LIRIJP_relative_df_final[, 1:6])
```

7.1.2.4 过滤基因

```
get_trimGene_LIRIJP <- function(input, occurrence = 0.1) {
  input_df <- data.frame(input)

  return(prof_res_v2)
}

LIRIJP_counts_df_trim <- get_trimGene_LIRIJP(input = LIRIJP_counts_df_final)
LIRIJP_relative_df_trim <- get_trimGene_LIRIJP(input = LIRIJP_relative_df_final)

dim(LIRIJP_counts_df_trim)
```

7.1.2.5 输出结果

输出结果: 将上述结果以 tsv 格式输出到结果目录

```
if (!dir.exists("./data/result/profile")) {
  dir.create("./data/result/profile/", recursive = TRUE)
}

write.table(LIRIJP_counts_df_trim, "./data/result/profile/LIRI-JP_gene_counts.tsv",
            sep = "\t", quote = F, row.names = T)
write.table(LIRIJP_relative_df_trim, "./data/result/profile/LIRI-JP_gene_relative.tsv",
            sep = "\t", quote = F, row.names = T)
```

7.2 TCGA-LIHC

```
options(warn = -1)
library(TCGAbiolinks)
```

```

library(dplyr)
library(SummarizedExperiment)
library(optparse)
options(warn = 0)

option_list <- list(
  make_option(c("-t", "--tumor"), type="character", default="TCGA-KIRC",
             help="tumor type from TCGA", metavar="character"),
  make_option(c("-o", "--out"), type="character",
             help="output directory", metavar="character")
);
opt_parser <- OptionParser(option_list = option_list)
opt <- parse_args(opt_parser)

current_dir <- getwd()

# input parameters
tumor_type <- opt$tumor
data_type <- opt$out

# clinical information output
if(!dir.exists(paste0(current_dir, "/Clinical/"))){
  dir.create(paste0(current_dir, "/Clinical/"))
}
origin_clinical_name <- paste0(current_dir, "/Clinical/", tumor_type, "_clinical_origin.csv")
if(!file.exists(origin_clinical_name)){
  clinical <- GDCquery_clinic(project = tumor_type, type = "clinical")
  write.csv(clinical, origin_clinical_name, row.names = F)
}
message("Clinical information has been successfully downloaded")

#####
#   omics-data layer: SummarizedExperiment Object  #
#####

OmicsData <- get_OmicsData(project = tumor_type, Outdir = data_type)
if(!dir.exists(paste0(current_dir, "/Omics/"))){
  dir.create(paste0(current_dir, "/Omics/"))
}

```

```

    }
OmicsData_filename <- paste0(current_dir, "/Omics/", tumor_type, "_", data_type, ".RDS")
saveRDS(OmicsData, file = OmicsData_filename)
message("Omics data also has been successfully downloaded")

```

最终获得了临床表型和表达谱数据：

- 病人的数据（临床数据）：TCGA-LIHC_clinical_origin.csv
- 表达数据（测序数据）：TCGA-LIHC.htseq_counts.tsv/TCGA-LIHC.htseq_fpkm.tsv

7.2.1 TCGA-LIHC 临床表型

本节将生成临床患者临床数据以及样本对应的表型数据。

7.2.1.1 加载 R 包

使用 `rm(list = ls())` 清空环境内变量

```

library(tidyverse)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#1F78B4", "#EE2B2B")

```

7.2.1.2 导入数据

```
TCGALIHC_metadata <- read.csv("./data/TCGA_LIHC/TCGA-LIHC_clinical_origin.csv")
```

7.2.1.3 数据清洗

重命名列名

```

TCGALIHC_clin_columns <- c("submitter_id", "race",
                            "gender", "vital_status", "age_at_index",
                            "ajcc_pathologic_stage", "days_to_last_follow_up",

head(TCGALIHC_clinic_data)

```

7.2.1.4 输出结果

将上述结果以 csv 格式输出到结果目录

```
if (!dir.exists("./data/result/metadata/")) {
  dir.create("./data/result/metadata/", recursive = TRUE)
}

write.csv(TCGALIHC_clinic_data, "./data/result/metadata/TCGA-LIHC-clinical.csv", row.names = F)
```

7.2.2 TCGA-LIHC 转录组

基因表达数据需要转换基因 ID 以及过滤冗余基因

7.2.2.1 加载 R 包

```
library(tidyverse)
library(data.table)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)
```

7.2.2.2 导入数据

```
TCGALIHC_genexp <- fread("./data/TCGA_LIHC/TCGA-LIHC.htseq_counts.tsv")
TCGALIHC_genexp_fpkm <- fread("./data/TCGA_LIHC/TCGA-LIHC.htseq_fpkm.tsv")
TCGALIHC_geneID <- fread("./data/GeneIDMap/human_gene_all.tsv")
```

7.2.2.3 数据清洗

```
colnames(TCGALIHC_genexp) <- gsub("-", "_", colnames(TCGALIHC_genexp))
colnames(TCGALIHC_genexp) <- gsub("_01A", "", colnames(TCGALIHC_genexp))
TCGALIHC_gene_data <- TCGALIHC_genexp %>%
  dplyr::select(-all_of(grep("11A", colnames(TCGALIHC_genexp), value = T))) %>%
  dplyr::rename(gene_id = Ensembl_ID) %>%
  dplyr::mutate(gene_id = gsub("\\.\\d+", "", gene_id))

TCGALIHC_fpkm_df_final <- TCGALIHC_gene_fpkm_data %>%
```

©Hua

```
dplyr::left_join(TCGALIHC_geneID %>%
  dplyr::select(ensembl_gene_id, external_gene_name) %>%
  dplyr::distinct(),
  by = c("gene_id" = "ensembl_gene_id")) %>%
dplyr::select(-gene_id) %>%
dplyr::rename(GeneName = external_gene_name) %>%
dplyr::select(GeneName, everything()) %>%
dplyr::filter(!is.na(GeneName)) %>%
dplyr::distinct()
head(TCGALIHC_fpkm_df_final[, 1:6])
```

7.2.2.4 过滤基因

```
TCGALIHC_counts_df_trim <- get_trimGene_TCGALIHC(input = TCGALIHC_counts_df_final)
TCGALIHC_fpkm_df_trim <- get_trimGene_TCGALIHC(input = TCGALIHC_fpkm_df_final)

dim(TCGALIHC_fpkm_df_trim)
```

7.2.2.5 表达值转换成 count abundance

```
#| warning: false
#| message: false

TCGALIHC_counts_df_trim_final <- apply(TCGALIHC_counts_df_trim, 2, function(x) {
  return(2^x - 1)
})

head(TCGALIHC_counts_df_trim_final[, 1:5])
```

7.2.2.6 输出结

输出结果: 将上述结果以 tsv 格式输出到结果目录

```
if (!dir.exists("./data/result/profile/")) {
  dir.create("./data/result/profile/", recursive = TRUE)
}

write.table(TCGALIHC_counts_df_trim, "./data/result/profile/TCGA-LIHC_gene_counts.tsv",
            sep = "\t", quote = F, row.names = T)
write.table(TCGALIHC_counts_df_trim_final, "./data/result/profile/TCGA-LIHC_gene_Truecounts.tsv",
```

```

        sep = "\t", quote = F, row.names = T)
write.table(TCGALIHC_fpkm_df_trim, "./data/result/profile/TCGA-LIHC_gene_fpkm.tsv",
            sep = "\t", quote = F, row.names = T)

```

7.3 GSE14520

最终获得了临床表型和表达谱数据：

- 病人的数据（临床数据）：GSE14520_clinical_post.csv/GSE14520_Extra_Supplement.txt
- 表达数据（测序数据）：GSE14520_profile_post.tsv

7.3.1 GSE14520 临床表型

本节将生成临床患者临床数据以及样本对应的表型数据。

7.3.1.1 加载 R 包

加载 R 包：使用 `rm(list = ls())` 清空环境内变量

```

library(tidyverse)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#1F78B4", "#EE2B2B")

```

7.3.1.2 导入数据

```

GSE14520_metadata <- read.csv("./data/GSE14520/GSE14520_process/GSE14520_clinical_post.csv")
GSE14520_sup_info <- read.table("./data/GSE14520/GSE14520_Extra_Supplement.txt", sep = "\t", header = T)

```

7.3.1.3 数据清洗

数据清洗

```

GSE14520_clin_columns <- c("Barcode", "Disease.state", "Individual",
                           "Tissue", "Gender", "Age", "TNM.staging",
                           "Survival.status", "Survival.months")

```

```
head(GSE14520_clinic_data)
```

7.3.1.4 输出结果

- 输出结果: 将上述结果以 csv 格式输出到结果目录

```
if (!dir.exists("./data/result/metadata/")) {
  dir.create("./data/result/metadata/", recursive = TRUE)
}

write.csv(GSE14520_clinic_data, "./data/result/metadata/GSE14520-clinical.csv", row.names = F)
```

7.3.2 GSE14520 转录组

基因表达数据需要转换基因 ID 以及过滤冗余基因

7.3.2.1 加载 R 包

加载 R 包

```
library(tidyverse)
library(data.table)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)
```

7.3.2.2 导入数据

```
GSE14520_genexp <- fread("./data/GSE14520/GSE14520_process/GSE14520_profile_post.tsv")
```

7.3.2.3 数据清洗

数据清洗

```
GSE14520_counts_df_final <- GSE14520_genexp %>%
  dplyr::rename(GeneName = GeneID) %>%
  dplyr::select(GeneName, everything()) %>%
  dplyr::filter(!is.na(GeneName)) %>%
  dplyr::distinct()
```

```
head(GSE14520_counts_df_final[, 1:6])
```

7.3.2.4 过滤基因

```
GSE14520_counts_df_trim <- get_trimGene_GSE14520(input = GSE14520_counts_df_final)

dim(GSE14520_counts_df_trim)
```

7.3.2.5 输出结果

输出结果: 将上述结果以 tsv 格式输出到结果目录

```
if (!dir.exists("./data/result/profile/")) {
  dir.create("./data/result/profile/", recursive = TRUE)
}

write.table(GSE14520_counts_df_trim, "./data/result/profile/GSE14520_gene_counts.tsv",
            sep = "\t", quote = F, row.names = T)
```

7.4 总结

针对 LIRI-JP, LIHC-US/TCGA-LIHC 和 GSE14520 这三个数据集的临床表型和表达谱数据, 进行了详尽的数据预处理工作。经过清洗、整合和筛选, 成功去除了冗余信息、修正了潜在错误, 并确保了数据的完整性和准确性。最终, 得到了符合后续分析要求的高质量输入文件。

为了更高效地利用这些数据, 决定将它们存储为常见的 *ExpressionSet* 数据对象 (见章节 八)。 *ExpressionSet* 是生物信息学分析中常用的数据结构, 它能够同时容纳表达谱数据和与之相关的临床表型信息。通过将数据转换为 *ExpressionSet* 格式, 可以方便地访问和操作数据, 为后续的统计分析、可视化以及模型构建等步骤提供便利。这一步骤的完成, 为后续的数据分析奠定了坚实的基础。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
```

LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v

locale:

[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:

[1] stats graphics grDevices datasets utils methods base

other attached packages:

[1] data.table_1.15.4 lubridate_1.9.3forcats_1.0.0 stringr_1.5.1
[5] dplyr_1.1.4 purrrr_1.0.2readr_2.1.5 tidyverse_1.3.1
[9] tibble_3.2.1 ggplot2_3.5.1 tidyverse_2.0.0

loaded via a namespace (and not attached):

[1] gtable_0.3.5 jsonlite_1.8.8 compiler_4.3.3
[4] BiocManager_1.30.23 renv_1.0.0 tidyselect_1.2.1
[7] scales_1.3.0 yaml_2.3.8 fastmap_1.1.1
[10] R6_2.5.1 generics_0.1.3 knitr_1.46
[13] munsell_0.5.1 pillar_1.9.0 tzdb_0.4.0
[16] rlang_1.1.3 utf8_1.2.4 stringi_1.8.4
[19] xfun_0.43 timechange_0.3.0 cli_3.6.2
[22] withr_3.0.0 magrittr_2.0.3 digest_0.6.35
[25] grid_4.3.3 rstudioapi_0.16.0 hms_1.1.3
[28] lifecycle_1.0.4 vctrs_0.6.5 evaluate_0.23
[31] glue_1.7.0 fansi_1.0.6 colorspace_2.1-0
[34] rmarkdown_2.26 tools_4.3.3 pkgconfig_2.0.3
[37] htmltools_0.5.8.1

第八章 数据对象

ExpressionSet 是 **Biobase**(Huber 等 2015) 包提供的一种数据对象，专为存储生物学实验数据而设计。它允许用户将表型数据、表达谱以及特征数据（如基因或蛋白质标识符）集成在一个统一的数据结构中。在本章节中，将利用这种数据对象来组织并处理获得的三个数据集：LIRI-JP，LIHC-US/TCGA-LIHC 和 GSE14520。

8.1 LIRI-JP

要生成 LIRI-JP 的 *ExpressionSet* 对象，首先需要加载必要的 R 包，然后导入数据和元数据，最后将这些数据组合成一个 *ExpressionSet* 对象。

8.1.1 加载 R 包

加载 R 包：使用 `rm(list = ls())` 清空环境内变量

```
library(tidyverse)
library(data.table)
library(Biobase)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#1F78B4", "#EE2B2B")
```

8.1.2 导入数据

输入数据来自于章节 七

```
LIRIJP_count_data <- fread("./data/result/profile/LIRI-JP_gene_counts.tsv")
LIRIJP_relative_data <- fread("./data/result/profile/LIRI-JP_gene_relative.tsv")
LIRIJP_phenotype <- read.csv("./data/result/metadata/LIRI-JP-specimen.csv")
```

8.1.3 生成 ExpressionSet

生成 *ExpressionSet-class* 数据对象：过滤掉低出现率的基因，默认出现率阈值是 20%

```
get_ExprSet_LIRIJP <- function(
  profile,
  metadata,
  occurrence = 0.2) {

  profile_new <- profile %>%
    tibble::column_to_rrownames("V1")
  metadata_new <- metadata %>%
    dplyr::filter(Tumour_Stage != "") %>%
    dplyr::mutate(Group = case_when(
      Tumour_Stage %in% c("T1", "T2") ~ "Early Stage",
      Tumour_Stage %in% c("T3", "T4") ~ "Late Stage"))

  # metadata Description
  sid <- dplyr::intersect(colnames(profile_new), metadata_new$SpecimenID)
  phen <- metadata_new[pmatch(sid, metadata_new$SpecimenID), , ]
  phen <- phen[pmatch(unique(phen$PatientID), phen$PatientID), , ]
  prof <- profile_new %>%
    dplyr::select(dplyr::all_of(phen$SpecimenID))

  if (all(colnames(prof) == phen$SpecimenID)) {
    colnames(prof) <- phen$PatientID
  }

  # filter features according gene symbol
  prof_cln <- prof %>%
    tibble::rownames_to_column("Gene") %>%
    data.frame()
  idx <- grep("Gene", colnames(prof_cln))
  prof_cln$median <- apply(prof_cln[, -idx], 1, median)
  prof_cln <- with(prof_cln, prof_cln[order(Gene, median, decreasing = T), ])
  prf_deduplicated <- prof_cln[!duplicated(prof_cln$Gene), ] %>%
```

```

dplyr::select(-median)

# return result: gene symbol
rownames(prf_deduplicated) <- NULL
prof_res <- prf_deduplicated %>%
  dplyr::select(Gene, everything()) %>%
  tibble::column_to_rownames("Gene")

# determine the right order between profile and phenotype
for (i in 1:ncol(prof_trim)) {
  if (!(colnames(prof_trim)[i] == rownames(phen)[i])) {
    stop(paste0(i, " Wrong"))
  }
}

expressionSet <- new("ExpressionSet",
  exprs = exprs,
  phenoData = adf,
  experimentData = experimentData)

return(expressionSet)
}

LIRIJP_ExprSet_count <- get_ExprSet_LIRIJP(
  profile = LIRIJP_count_data,
  metadata = LIRIJP_phenotype,
  occurrence = 0.2)

LIRIJP_ExprSet_rela <- get_ExprSet_LIRIJP(
  profile = LIRIJP_relative_data,
  metadata = LIRIJP_phenotype,
  occurrence = 0.2)

LIRIJP_ExprSet_rela

```

8.1.4 输出结果

输出结果: 将上述结果以 RDS 格式输出到结果目录

```

if (!dir.exists("./data/result/ExpSetObject/")) {
  dir.create("./data/result/ExpSetObject/", recursive = TRUE)
}

```

```

}

saveRDS(LIRIJP_ExprSet_count, "./data/result/ExpSetObject/LIRI-JP_ExpSet_counts.RDS", compress = TRUE)
saveRDS(LIRIJP_ExprSet_rela, "./data/result/ExpSetObject/LIRI-JP_ExpSet_relative.RDS", compress = TRUE)

```

8.2 TCGA-LIHC

要生成 TCGA-LIHC 的 *ExpressionSet* 对象，首先需要加载必要的 R 包，然后导入数据和元数据，最后将这些数据组合成一个 *ExpressionSet* 对象。

8.2.1 加载 R 包

加载 R 包：使用 `rm(list = ls())` 清空环境内变量

```

library(tidyverse)
library(data.table)
library(BioBase)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#1F78B4", "#EE2B2B")

```

8.2.2 导入数据

```

TCGALIHC_count_data <- fread("./data/result/profile/TCGA-LIHC_gene_counts.tsv")
TCGALIHC_TrueCount_data <- fread("./data/result/profile/TCGA-LIHC_gene_Truecounts.tsv")
TCGALIHC_fpkm_data <- fread("./data/result/profile/TCGA-LIHC_gene_fpkm.tsv")
TCGALIHC_phenotype <- read.csv("./data/result/metadata/TCGA-LIHC-clinical.csv")

```

8.2.3 生成 ExpressionSet

生成 *ExpressionSet-class* 数据对象：过滤掉低出现率的基因，默认出现率阈值是 **20%**

```

get_ExprSet_TCGALIHC <- function(
  profile,

```

```

metadata,
occurrence = 0.2) {

profile_new <- profile %>%
  tibble::column_to_rownames("V1")
metadata_new <- metadata %>%
  dplyr::filter(Tumour_Stage != "") %>%
  dplyr::mutate(Group = case_when(
    Tumour_Stage %in% c("T1", "T2") ~ "Early Stage",
    Tumour_Stage %in% c("T3", "T4") ~ "Late Stage")) %>%
  dplyr::mutate(Time = ifelse(is.na(death_Time), follow_up_Time, death_Time))

# metadata Description
sid <- dplyr::intersect(colnames(profile_new), metadata_new$PatientID)
phen <- metadata_new[pmatch(sid, metadata_new$PatientID), , ]
prof <- profile_new %>%
  dplyr::select(dplyr::all_of(phen$PatientID))

# filter features according gene symbol
prof_cln <- prof %>%
  tibble::rownames_to_column("Gene") %>%
  data.frame()
idx <- grep("Gene", colnames(prof_cln))
prof_cln$median <- apply(prof_cln[, -idx], 1, median)
prof_cln <- with(prof_cln, prof_cln[order(Gene, median, decreasing = T), ])
prf_deduplicated <- prof_cln[!duplicated(prof_cln$Gene), ] %>%
  dplyr::select(-median)

# return result: gene symbol
rownames(prf_deduplicated) <- NULL
prof_res <- prf_deduplicated %>%
  dplyr::select(Gene, everything()) %>%
  tibble::column_to_rownames("Gene")

# determine the right order between profile and phenotype
for (i in 1:ncol(prof_res)) {
  if (!(colnames(prof_res)[i] == rownames(phen)[i])) {
    stop(paste0(i, " Wrong"))
  }
}

```

```

expressionSet <- new("ExpressionSet",
                      exprs = exprs,
                      phenoData = adf,
                      experimentData = experimentData)

return(expressionSet)
}

TCGALIHC_ExprSet_count <- get_ExprSet_TCGALIHC(
  profile = TCGALIHC_count_data,
  metadata = TCGALIHC_phenotype,
  occurrence = 0.2)
# TCGALIHC_ExprSet_count

TCGALIHC_ExprSet_TrueCount <- get_ExprSet_TCGALIHC(
  profile = TCGALIHC_TrueCount_data,
  metadata = TCGALIHC_phenotype,
  occurrence = 0.2)
# TCGALIHC_ExprSet_TrueCount

TCGALIHC_ExprSet_fpkm <- get_ExprSet_TCGALIHC(
  profile = TCGALIHC_fpkm_data,
  metadata = TCGALIHC_phenotype,
  occurrence = 0.2)
TCGALIHC_ExprSet_fpkm

```

8.2.4 输出结果

输出结果: 将上述结果以 RDS 格式输出到结果目录

```

if (!dir.exists("./data/result/ExpSetObject/")) {
  dir.create("./data/result/ExpSetObject/", recursive = TRUE)
}

saveRDS(TCGALIHC_ExprSet_count, "./data/result/ExpSetObject/TCGA-LIHC_ExpSet_counts.RDS", compress = TRUE)
saveRDS(TCGALIHC_ExprSet_TrueCount, "./data/result/ExpSetObject/TCGA-LIHC_ExpSet_TrueCount.RDS", compress = TRUE)
saveRDS(TCGALIHC_ExprSet_fpkm, "./data/result/ExpSetObject/TCGA-LIHC_ExpSet_fpkm.RDS", compress = TRUE)

```

8.3 GSE14520

要生成 GSE14520 的 *ExpressionSet* 对象，首先需要加载必要的 R 包，然后导入数据和元数据，最后将这些数据组合成一个 *ExpressionSet* 对象。

8.3.1 加载 R 包

加载 R 包：使用 `rm(list = ls())` 清空环境内变量

```
library(tidyverse)
library(data.table)
library(Biostrings)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#1F78B4", "#EE2B2B")
```

8.3.2 导入数据

导入数据

```
GSE14520_count_data <- fread("./data/result/profile/GSE14520_gene_counts.tsv")
GSE14520_phenotype <- read.csv("./data/result/metadata/GSE14520-clinical.csv")
```

8.3.3 生成 ExpressionSet

生成 *ExpressionSet-class* 数据对象：过滤掉低出现率的基因，默认出现率阈值是 20%

```
get_ExprSet_GSE14520 <- function(
  profile,
  metadata,
  occurrence = 0.2) {

  profile_new <- profile %>%
    tibble::column_to_rownames("V1")
  metadata_new <- metadata %>%
```

©Hua

```

dplyr::filter(Tumour_Stage != "") %>%
dplyr::mutate(Group = case_when(
  Tumour_Stage %in% c("T1", "T2") ~ "Early Stage",
  Tumour_Stage %in% c("T3", "T4") ~ "Late Stage"))

# metadata Description
sid <- dplyr::intersect(colnames(profile_new), metadata_new$SampleID)
phen <- metadata_new[pmatch(sid, metadata_new$SampleID), , ]
prof <- profile_new %>%
  dplyr::select(dplyr::all_of(phen$SampleID))

# return result: gene symbol
rownames(prf_deduplicated) <- NULL
prof_res <- prf_deduplicated %>%
  dplyr::select(Gene, everything()) %>%
  tibble::column_to_rownames("Gene")

# determine the right order between profile and phenotype
for (i in 1:ncol(prof_trim)) {
  if (!(colnames(prof_trim)[i] == rownames(phen)[i])) {
    stop(paste0(i, " Wrong"))
  }
}

expressionSet <- new("ExpressionSet",
  exprs = exprs,
  phenoData = adf,
  experimentData = experimentData)

return(expressionSet)
}

GSE14520_ExprSet_count <- get_ExprSet_GSE14520(
  profile = GSE14520_count_data,
  metadata = GSE14520_phenotype,
  occurrence = 0.2)
GSE14520_ExprSet_count

```

8.3.4 输出结果

输出结果: 将上述结果以 RDS 格式输出到结果目录

```
if (!dir.exists("./data/result/ExpSetObject/")) {
  dir.create("./data/result/ExpSetObject/", recursive = TRUE)
}

saveRDS(GSE14520_ExprSet_count, "./data/result/ExpSetObject/GSE14520_ExpSet_counts.RDS", compress = TRUE)
```

8.4 重叠基因

为了确保在发现数据集中识别的标记基因能够在验证数据集中进行准确的验证，首先加载了基于 *count abundance* 的 RDS，这些 RDS 分别包含 LIRI-JP，LIHC-US/TCGA-LIHC 和 GSE14520 数据集的基因表达信息。

步骤：

1. 加载 RDS 文件：首先，将分别加载 LIRI-JP，LIHC-US/TCGA-LIHC 和 GSE14520 的 *count abundance* 的 RDS 文件。
2. 提取基因标识符：从每个数据集中提取基因标识符列表。
3. 获取基因交集：比较这三个基因标识符列表，找出它们之间的交集。这个交集将包含所有在三个数据集中都存在的基因。
4. 更新数据集：使用交集中的基因标识符来过滤和更新每个数据集，以确保后续分析中的基因一致性。

8.4.1 加载 R 包

加载 R 包

```
library(tidyverse)
library(data.table)
library(BioBase)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)
```

```
grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#1F78B4", "#EE2B2B")
```

8.4.2 导入数据

导入数据

```
ExprSet_LIRI_JP <- readRDS("./data/result/ExpSetObject/LIRI-JP_ExpSet_counts.RDS")
ExprSet_TCGA_LIHC <- readRDS("./data/result/ExpSetObject/TCGA-LIHC_ExpSet_TrueCount.RDS")
ExprSet_GSE14520 <- readRDS("./data/result/ExpSetObject/GSE14520_ExpSet_counts.RDS")
```

8.4.3 共有基因

共有基因

```
length(overlap_genes)
```

8.4.4 生成数据对象

生成数据对象

```
get_ExprSet <- function(
  dat,
  gene_list = overlap_genes,
  occurrence = 0.2) {

  profile <- exprs(dat) %>%
    as.data.frame() %>%
    t() %>%
    as.data.frame() %>%
    dplyr::select(all_of(overlap_genes)) %>%
    t() %>%
    as.data.frame()

  metadata <- pData(dat) %>%
    as.data.frame()

  # metadata Description
  sid <- dplyr::intersect(colnames(profile), rownames(metadata))
  phen <- metadata[pmatch(sid, rownames(metadata)), , ]
```

```

prof <- profile %>%
  dplyr::select(dplyr::all_of(rownames(pheno)))

# filter features according gene symbol
prof_cln <- prof %>%
  tibble::rownames_to_column("Gene") %>%
  data.frame()
idx <- grep("Gene", colnames(prof_cln))
prof_cln$median <- apply(prof_cln[, -idx], 1, median)
prof_cln <- with(prof_cln, prof_cln[order(Gene, median, decreasing = T), ])
prof_deduplicated <- prof_cln[!duplicated(prof_cln$Gene), ] %>%
  dplyr::select(-median)

# determine the right order between profile and phenotype
for (i in 1:ncol(prof_trim)) {
  if (!(colnames(prof_trim)[i] == rownames(pheno)[i])) {
    stop(paste0(i, " Wrong"))
  }
}
expressionSet <- new("ExpressionSet",
  exprs = exprs,
  phenoData = adf,
  experimentData = experimentData)

return(expressionSet)
}

ExprSet_LIRI_JP_new <- get_ExprSet(
  dat = ExprSet_LIRI_JP,
  gene_list = overlap_genes,
  occurrence = 0.2)
ExprSet_LIRI_JP_new

ExprSet_TCGA_LIHC_new <- get_ExprSet(
  dat = ExprSet_TCGA_LIHC,
  gene_list = overlap_genes,
  occurrence = 0.2)
ExprSet_TCGA_LIHC_new

```

```
ExprSet_GSE14520_new <- get_ExprSet(
  dat = ExprSet_GSE14520,
  gene_list = overlap_genes,
  occurrence = 0.2)
ExprSet_GSE14520_new
```

8.4.5 输出结果

输出结果: 将上述结果以 RDS 格式输出到结果目录

```
if (!dir.exists("./data/result/ExpSetObject/")) {
  dir.create("./data/result/ExpSetObject/", recursive = TRUE)
}

saveRDS(ExprSet_LIRI_JP_new, "./data/result/ExpSetObject/Final_ExpSet_LIRI_JP_TrueCounts.RDS", compress = TRUE)
saveRDS(ExprSet_TCGA_LIHC_new, "./data/result/ExpSetObject/Final_ExpSet_TCGA_LIHC_TrueCounts.RDS", compress = TRUE)
saveRDS(ExprSet_GSE14520_new, "./data/result/ExpSetObject/Final_ExpSet_GSE14520_TrueCounts.RDS", compress = TRUE)
```

8.5 总结

在完成了对三个肝细胞癌 (HCC) 群体的转录组和临床数据的预处理后，分别将这些数据转换为了 *ExpressionSet* 数据对象，并保存为 RDS 文件格式，以便于后续的数据管理和分析。为了确保后续分析的一致性和可靠性，进一步从这些 RDS 文件中提取了三个数据集中共有的基因。这些共有的基因是后续分析的基础，因为它们代表了三个群体中均存在并可能参与 HCC 发展的生物学过程。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v3.10.1

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

©Hua

```
time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets   utils      methods    base

other attached packages:
[1] Biobase_2.62.0      BiocGenerics_0.48.1  data.table_1.15.4
[4] lubridate_1.9.3     forcats_1.0.0       stringr_1.5.1
[7] dplyr_1.1.4         purrr_1.0.2        readr_2.1.5
[10] tidyr_1.3.1        tibble_3.2.1       ggplot2_3.5.1
[13] tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] gtable_0.3.5        jsonlite_1.8.8      compiler_4.3.3
[4] BiocManager_1.30.23  renv_1.0.0        tidyselect_1.2.1
[7] scales_1.3.0        yaml_2.3.8        fastmap_1.1.1
[10] R6_2.5.1           generics_0.1.3     knitr_1.46
[13] munsell_0.5.1      pillar_1.9.0       tzdb_0.4.0
[16] rlang_1.1.3         utf8_1.2.4        stringi_1.8.4
[19] xfun_0.43          timechange_0.3.0   cli_3.6.2
[22] withr_3.0.0         magrittr_2.0.3     digest_0.6.35
[25] grid_4.3.3          rstudioapi_0.16.0   hms_1.1.3
[28] lifecycle_1.0.4     vctrs_0.6.5       evaluate_0.23
[31] glue_1.7.0          fansi_1.0.6       colorspace_2.1-0
[34] rmarkdown_2.26       tools_4.3.3       pkgconfig_2.0.3
[37] htmltools_0.5.8.1
```

第九章 数据校正

批次效应在生物学数据分析中是一个普遍存在的问题，它指的是由于实验过程中非生物学因素（如样本处理时间、实验条件、测序平台等）的差异，导致实验结果中混入与研究目标不相关的变异。在比较对照组和实验组时，这些非生物学因素可能引入额外的噪声，影响对生物学问题真实效应的判断。

因此，在本章节中，采用了不同的降低批次效应的方法：

- `sva::ComBat`: 来自 ([Leek 和 Storey 2007](#));
- `limma::removeBatchEffect`: 来自 ([Ritchie 等 2015](#));
- `limma::voom & snm::snm`: 来自 ([Ritchie 等 2015](#)) 和 ([Mecham, Nelson, 和 Storey 2010](#))，参考文章 ([Poore 等 2020](#));

9.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(Biobase)
library(sva)
library(limma)
library(MicrobiomeAnalysis)
library(SummarizedExperiment)
library(snm)
library(ggpubr)
library(cowplot)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
```

```
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```



9.2 导入数据

输入数据来自于章节 八

```
ExprSet_LIRI_JP <- readRDS("./data/result/ExpSetObject/Final_ExpSet_LIRI_JP_TrueCounts.RDS")
ExprSet_TCGA_LIHC <- readRDS("./data/result/ExpSetObject/Final_ExpSet_TCGA_LIHC_TrueCounts.RDS")
```

9.3 准备数据

- 准备三种校正批次效应的方法所需的输入数据。

```
metadata <- rbind(LIRI_JP_meta, TCGA_LIHC_meta)
profile <- LIRI_JP_eprs %>%
  tibble::rownames_to_column("GeneID") %>%
  dplyr::inner_join(TCGA_LIHC_eprs %>%
    tibble::rownames_to_column("GeneID"),
    by = "GeneID") %>%
  tibble::column_to_rownames("GeneID")

head(as.matrix(profile[, 1:5]))
```

- 准备批次校正的因素及其矩阵

```
dat_raw <- profile
dat_meta <- metadata %>%
  dplyr::select(ProjectID, Group)
```

9.4 ComBat

sva::ComBat(Leek 和 Storey 2007) 是一种常用的批次校正算法，用于调整高通量数据中的批次效应。

```
head(ComBat_data[, 1:5])
```

9.5 removeBatchEffect

`limma:::removeBatchEffect`(Ritchie 等 2015) 是另一种常用的批次校正算法，它的原理基于线性模型和对批次效应的建模。

```
head(RME_data[, 1:5])
```

9.6 Voom SNM

集合 `limma:::voom` (Ritchie 等 2015) & `snm:::snm` (Mecham, Nelson, 和 Storey 2010)

```
get_VoomSNM <- function(
  qcMetadata,
  qcProfile,
  batchVar,
  groupVar){

  # qcMetadata = dat_meta
  # qcProfile = dat_raw
  # batchVar = "ProjectID"
  # groupVar = "Group"

  colnames(qcMetadata)[which(colnames(qcMetadata) == batchVar)] <- "Batch"
  colnames(qcMetadata)[which(colnames(qcMetadata) == groupVar)] <- "Group"

  # Set up counts matrix
  counts <- qcProfile # DGEList object from a table of counts (rows=features, columns=samples)

  snmData <- snmDataObjOnly$norm.dat

  return(snmData)
}

VoomSNM_data <- get_VoomSNM(
  qcMetadata = dat_meta,
  qcProfile = dat_raw,
  batchVar = "ProjectID",
  groupVar = "Group")
```

```
head(VoomSNM_data[, 1:5])
```

9.7 批次效应校正结果比较

MicrobiomeAnalysis(Zou 2022)

```
get_plot <- function(
  datMeta,
  datProf,
  group = "Group",
  group_names = grp_names,
  group_colors = grp_colors,
  method = "PCA",
  shape_var = "ProjectID",
  shape_values = grp_shapes) {

  datProf <- as.data.frame(datProf)

  sid <- intersect(rownames(datMeta), colnames(datProf))
  phen <- datMeta[rownames(datMeta) %in% sid, , ]
  counts <- datProf %>%
    dplyr::select(all_of(rownames(phen))) %>%
    as.matrix()

  if (!all(rownames(phen) == colnames(counts))) {
    stop("wrong order of samples")
  }

  se <- SummarizedExperiment(
    assays = SimpleList(counts = counts),
    colData = phen
  )

  return(pl)
}
```

- 合并数据集的原始数据 PCA 结果

```
# 计算消耗时间太久，设置该代码避免重复运算
if (file.exists("./data/result/Figure/Data_RawData_pl.RDS")) {
  RawData_pl <- readRDS("./data/result/Figure/Data_RawData_pl.RDS")
} else {
  RawData_pl <- get_plot(
    datMeta = metadata,
    datProf = profile,
    group = "Group",
    method = "PCA",
    shape_var = "ProjectID")

  saveRDS(RawData_pl, "./data/result/Figure/Data_RawData_pl.RDS", compress = TRUE)
}
```

RawData_pl

- 合并数据集的 `sva::ComBat` 数据 PCA 结果

```
# 计算消耗时间太久，设置该代码避免重复运算
if (file.exists("./data/result/Figure/Data_ComBat_pl.RDS")) {
  ComBat_pl <- readRDS("./data/result/Figure/Data_ComBat_pl.RDS")
} else {
  ComBat_pl <- get_plot(
    datMeta = metadata,
    datProf = ComBat_data,
    group = "Group",
    method = "PCA",
    shape_var = "ProjectID")

  saveRDS(ComBat_pl, "./data/result/Figure/Data_ComBat_pl.RDS", compress = TRUE)
}
```

ComBat_pl

- 合并数据集的 `limma::removeBatchEffect` 数据 PCA 结果

```
# 计算消耗时间太久，设置该代码避免重复运算
if (file.exists("./data/result/Figure/Data_RME_pl.RDS")) {
  RME_pl <- readRDS("./data/result/Figure/Data_RME_pl.RDS")
```

©Hua

```

} else {
  RME_pl <- get_plot(
    datMeta = metadata,
    datProf = RME_data,
    group = "Group",
    method = "PCA",
    shape_var = "ProjectID")

  saveRDS(RME_pl, "./data/result/Figure/Data_RME_pl.RDS", compress = TRUE)
}

```

RME_pl

- 合并数据集的 Voom+SNM 数据 PCA 结果

```

# 计算消耗时间太久，设置该代码避免重复运算
if (file.exists("./data/result/Figure/Data_VoomSNM_pl.RDS")) {
  VoomSNM_pl <- readRDS("./data/result/Figure/Data_VoomSNM_pl.RDS")
} else {
  VoomSNM_pl <- get_plot(
    datMeta = metadata,
    datProf = VoomSNM_data,
    group = "Group",
    method = "PCA",
    shape_var = "ProjectID")

  saveRDS(VoomSNM_pl, "./data/result/Figure/Data_VoomSNM_pl.RDS", compress = TRUE)
}

```

VoomSNM_pl

- 合并所有的图

为了便于观察不同批次校正方法的结果，需要将上述生成的图形进行合并。

```

adjusted_pl <- cowplot::plot_grid(
  RawData_pl, ComBat_pl, RME_pl, VoomSNM_pl,
  ncol = 2, align = "hv",
  labels = LETTERS[1:4])

```

RawData_pl

ComBat_pl
RME_pl
VoomSNM_pl

结果：

- 根据观察，经过 Voom+SNM 方法校正后的基因表达谱数据分布并不倾向于按项目聚类，相较于其他两种方法，效果更佳。

9.8 校正后的结果

对上述三种方法校正后表达谱数据存成 ExpressionSet，以便后续分析。另外，采用生态学常用评估整体变异和组间关系的 PERMANOVA([Anderson 2014](#)) 检验方法评价校正前后的结果。

```
get_ExprSet <- function(
  x,
  y,
  occurrence = 0.5) {

  # metadata Description
  sid <- dplyr::intersect(colnames(x), y$SampleID)
  phen <- y[pmatch(sid, y$SampleID), , ]
  prof <- x %>%
    as.data.frame() %>%
    dplyr::select(dplyr::all_of(phen$SampleID))

  # determine the right order between profile and phenotype
  for (i in 1:ncol(prof)) {
    if (!(colnames(prof)[i] == rownames(phen)[i])) {
      stop(paste0(i, " Wrong"))
    }
  }

  expressionSet <- new("ExpressionSet",
    exprs = exprs,
    phenoData = adf,
    experimentData = experimentData)

  return(expressionSet)
}
```

- RawData: R^2 结果

```

if (!dir.exists("./data/result/BatchEffect/")) {
  dir.create("./data/result/BatchEffect/", recursive = TRUE)
}

if (file.exists("./data/result/BatchEffect/Raw_PERMANOVA.RDS")) {
  Raw_PERMANOVA <- readRDS("./data/result/BatchEffect/Raw_PERMANOVA.RDS")
} else {
  saveRDS(Raw_PERMANOVA, "./data/result/BatchEffect/Raw_PERMANOVA.RDS", compress = TRUE)
}

```

Raw_PERMANOVA

结果：基于 PERMANOVA 分析结果， $\text{Pr}(>F) < 0.05$ 表明 Group 和 ProjectID 均与整体转录组结构存在显著差异。同时，ProjectID 解释了整体转录组结构变异的 **69.3%**，这表明批次效应在整体转录组结构中占据了相当大的比例。因此，在后续关于 Group 的差异研究中，必须进行批次校正，以确保研究结果的准确性和可靠性。

- sva::ComBat: 校正后的 R2 结果

```
#| warning: false
#| message: false
```

```
ComBat_ExprSet <- get_ExprSet(
  x = ComBat_data,
  y = metadata,
  occurrence = 0.5)
```

```

if (file.exists("./data/result/BatchEffect/ComBat_PERMANOVA.RDS")) {
  ComBat_PERMANOVA <- readRDS("./data/result/BatchEffect/ComBat_PERMANOVA.RDS")
} else {
  saveRDS(ComBat_PERMANOVA, "./data/result/BatchEffect/ComBat_PERMANOVA.RDS", compress = TRUE)
}

```

ComBat_PERMANOVA

结果：sva::ComBat 方法均降低了 Group 和 ProjectID 的解释的变异比例。ProjectID 的整体转录组结构变异从 **69.3%** 降低至 **22.8%**，而 Group 则从 **1.6%** 降低至 **1.3%**，这表明该方法同时对批次效应和生物学效应产生了影响。

• limma::removeBatchEffect

```
#| warning: false
#| message: false

RME_ExprSet <- get_ExprSet(
  x = RME_data,
  y = metadata,
  occurrence = 0.5)

if (file.exists("./data/result/BatchEffect/RME_PERMANOVA.RDS")) {
  RME_PERMANOVA <- readRDS("./data/result/BatchEffect/RME_PERMANOVA.RDS")
} else {
  saveRDS(RME_PERMANOVA, "./data/result/BatchEffect/RME_PERMANOVA.RDS", compress = TRUE)
}
```

RME_PERMANOVA

结果：limma::removeBatchEffect 方法也均降低了 Group 和 ProjectID 的解释的变异比例。ProjectID 的整体转录组结构变异从 69.3% 降低至 9.7%，而 Group 则从 1.6% 降低至 1%，这表明该方法同时对批次效应和生物学效应产生了影响。该方法在降低批次效应要好于 sva::ComBat。

- Voom+SNM

```
VoomSNM_ExprSet <- get_ExprSet(
  x = VoomSNM_data,
  y = metadata,
  occurrence = 0.5)

if (file.exists("./data/result/BatchEffect/VoomSNM_PERMANOVA.RDS")) {
  VoomSNM_PERMANOVA <- readRDS("./data/result/BatchEffect/VoomSNM_PERMANOVA.RDS")
} else {
  saveRDS(VoomSNM_PERMANOVA, "./data/result/BatchEffect/VoomSNM_PERMANOVA.RDS", compress = TRUE)
}
```

VoomSNM_PERMANOVA

结果：Voom+SNM 方法也均降低了 Group 和 ProjectID 的解释的变异比例。ProjectID 的整体转录组结构变异从 69.3% 降低至 0.3%，而 Group 则从 1.6% 降低至 1.3%，这表明该方法同时对批次效应和生物学效应产生了影响。相比其他两种方法 (sva::ComBat 和 limma::removeBatchEffect)，Voom+SNM

不仅显著降低批次效应并且也最大程度保留生物学效应。

9.9 输出校正后的结果

```
#| warning: false
#| message: false

if (!dir.exists("./data/result/ExpSetObject")) {
  dir.create("./data/result/ExpSetObject", recursive = TRUE)
}

# saveRDS(ComBat_ExprSet, "./data/result/ExpSetObject/MergeExpSet_ComBat_LIRI-JP_TCGA-LIHC.RDS", compress = TRUE)
# saveRDS(RME_ExprSet, "./data/result/ExpSetObject/MergeExpSet_RME_LIRI-JP_TCGA-LIHC.RDS", compress = TRUE)
# saveRDS(MMUPHin_ExprSet, "./data/result/ExpSetObject/MergeExpSet_MMUPHin_LIRI-JP_TCGA-LIHC.RDS", compress = TRUE)
# saveRDS(ComBatSeq_ExprSet, "./data/result/ExpSetObject/MergeExpSet_ComBatSeq_LIRI-JP_TCGA-LIHC.RDS", compress = TRUE)
saveRDS(VoomSNM_ExprSet, "./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS", compress = TRUE)

if (!dir.exists("./data/result/Figure")) {
  dir.create("./data/result/Figure", recursive = TRUE)
}

ggsave("./data/result/Figure/SFig1.pdf", adjusted_pl, width = 10, height = 7, dpi = 600)
```

9.10 总结

在评估了不同批次效应校正方法后，发现 Voom + SNM 的组合在降低批次效应方面表现最佳，同时能够有效地保留生物学效应。这一组合不仅显著减少了由于非生物学因素导致的变异，还确保了数据中生物学信号的完整性和准确性。因此，决定使用 Voom + SNM 校正后的表达谱数据，并将其保存为 RDS 文件格式，以便直接用于后续的下游分析。这样，能够基于准确且可靠的数据集，进行更深入的生物学研究。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2
```

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v

©Hua

locale:

[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:

[1] stats4 stats graphics grDevices datasets utils methods
[8] base

other attached packages:

[1] cowplot_1.1.3	ggnetwork_0.6.0
[3] snm_1.50.0	SummarizedExperiment_1.32.0
[5] GenomicRanges_1.54.1	GenomeInfoDb_1.38.8
[7] IRanges_2.36.0	S4Vectors_0.40.2
[9] MatrixGenerics_1.14.0	matrixStats_1.3.0
[11] MicrobiomeAnalysis_1.0.3	limma_3.58.1
[13] sva_3.50.0	BiocParallel_1.36.0
[15] genefilter_1.84.0	mgcv_1.9-1
[17] nlme_3.1-164	Biobase_2.62.0
[19] BiocGenerics_0.48.1	data.table_1.15.4
[21] lubridate_1.9.3	forcats_1.0.0
[23] stringr_1.5.1	dplyr_1.1.4
[25] purrr_1.0.2	readr_2.1.5
[27] tidyverse_1.3.1	tibble_3.2.1
[29] ggplot2_3.5.1	tidyverse_2.0.0

loaded via a namespace (and not attached):

[1] fs_1.6.4	bitops_1.0-7
[3] DirichletMultinomial_1.44.0	httr_1.4.7
[5] RColorBrewer_1.1-3	doParallel_1.0.17
[7] numDeriv_2016.8-1.1	tools_4.3.3
[9] doRNG_1.8.6	backports_1.4.1
[11] utf8_1.2.4	R6_2.5.1
[13] vegan_2.6-4	lazyeval_0.2.2
[15] rhdf5filters_1.14.1	permute_0.9-7

```
[17] withr_3.0.0                  gridExtra_2.3
[19] cli_3.6.2                   sandwich_3.1-0
[21] mvtnorm_1.2-4               proxy_0.4-27
[23] yulab.utils_0.1.4           foreign_0.8-86
[25] scater_1.30.1              decontam_1.22.0
[27] readxl_1.4.3                rstudioapi_0.16.0
[29] RSQLite_2.3.6                generics_0.1.3
[31] shape_1.4.6.1               gtools_3.9.5
[33] car_3.1-2                  Matrix_1.6-5
[35] biomformat_1.30.0            ggbeeswarm_0.7.2
[37] fansi_1.0.6                 DescTools_0.99.54
[39] DECIPHER_2.30.0              abind_1.4-5
[41] lifecycle_1.0.4              multcomp_1.4-25
[43] yaml_2.3.8                  edgeR_4.0.16
[45] carData_3.0-5               gplots_3.1.3.1
[47] rhdf5_2.46.1                SparseArray_1.2.4
[49] grid_4.3.3                  blob_1.2.4
[51] crayon_1.5.2                lattice_0.22-6
[53] beachmat_2.18.1              annotate_1.80.0
[55] KEGGREST_1.42.0              pillar_1.9.0
[57] knitr_1.46                  boot_1.3-30
[59] gld_2.6.6                   corpcor_1.6.10
[61] codetools_0.2-19             glue_1.7.0
[63] MultiAssayExperiment_1.28.0  vctrs_0.6.5
[65] png_0.1-8                   treeio_1.26.0
[67] Rdpack_2.6                  cellranger_1.1.0
[69] gtable_0.3.5                cachem_1.0.8
[71] xfun_0.43                  rbibutils_2.2.16
[73] S4Arrays_1.2.1              metagenomeSeq_1.43.0
[75] survival_3.7-0              SingleCellExperiment_1.24.0
[77] iterators_1.0.14             statmod_1.5.0
[79] bluster_1.12.0              gmp_0.7-4
[81] TH.data_1.1-2               ANCOMBC_2.4.0
[83] phyloseq_1.46.0              bit64_4.0.5
[85] irlba_2.3.5.1              KernSmooth_2.23-22
[87] vipor_0.4.7                 rpart_4.1.23
[89] colorspace_2.1-0             DBI_1.2.2
[91] Hmisc_5.1-2                 nnet_7.3-19
[93] ade4_1.7-22                 Exact_3.2
[95] DESeq2_1.42.1               tidyselect_1.2.1
```

[97] bit_4.0.5 compiler_4.3.3
[99] glmnet_4.1-8 htmlTable_2.4.2
[101] BiocNeighbors_1.20.2 expm_0.999-9
[103] DelayedArray_0.28.0 caTools_1.18.2
[105] checkmate_2.3.1 scales_1.3.0
[107] digest_0.6.35 minqa_1.2.6
[109] rmarkdown_2.26 XVector_0.42.0
[111] htmltools_0.5.8.1 pkgconfig_2.0.3
[113] base64enc_0.1-3 lme4_1.1-35.3
[115] sparseMatrixStats_1.14.0 fastmap_1.1.1
[117] rlang_1.1.3 htmlwidgets_1.6.4
[119] DelayedMatrixStats_1.24.0 zoo_1.8-12
[121] jsonlite_1.8.8 energy_1.7-11
[123] BiocSingular_1.18.0 RCurl_1.98-1.14
[125] magrittr_2.0.3 Formula_1.2-5
[127] scuttle_1.12.0 GenomeInfoDbData_1.2.11
[129] Rhdf5lib_1.24.2 munsell_0.5.1
[131] Rcpp_1.0.12 ape_5.8
[133] viridis_0.6.5 CVXR_1.0-12
[135] stringi_1.8.4 rootSolve_1.8.2.4
[137] zlibbioc_1.48.2 MASS_7.3-60.0.1
[139] plyr_1.8.9 parallel_4.3.3
[141] ggrepel_0.9.5 lmom_3.0
[143] Biostrings_2.70.3 splines_4.3.3
[145] multtest_2.58.0 hms_1.1.3
[147] locfit_1.5-9.9 igraph_2.0.3
[149] ggsignif_0.6.4 Wrench_1.20.0
[151] rngtools_1.5.2 reshape2_1.4.4
[153] ScaledMatrix_1.10.0 XML_3.99-0.16.1
[155] evaluate_0.23 renv_1.0.0
[157] BiocManager_1.30.23 nloptr_2.0.3
[159] tzdb_0.4.0 foreach_1.5.2
[161] rsvd_1.0.5 broom_1.0.5
[163] xtable_1.8-4 Rmpfr_0.9-5
[165] e1071_1.7-14 tidytree_0.4.6
[167] rstatix_0.7.2 viridisLite_0.4.2
[169] class_7.3-22 gsl_2.1-8
[171] lmerTest_3.1-3 memoise_2.0.1
[173] beeswarm_0.4.0 AnnotationDbi_1.66.0
[175] cluster_2.1.6 TreeSummarizedExperiment_2.10.0

[177] timechange_0.3.0 mia_1.10.0

第十章 数据汇总

在本教程中，汇总了三个肝细胞癌（HCC）的转录组数据集，分别是 LIRI-JP，LIHC-US/TCGA-LIHC 和 GSE14520，以及一个 HCC 的单细胞数据集 GSE149614 的临床表型信息。这些数据集为科研人员提供了丰富的基因表达数据和相关的临床信息，有助于科研人员更深入地理解 HCC 的分子机制和临床特征。

加载 R 包 {#sec-gtsummary-packages}

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(Biobase)
library(gtsummary)
library(xlsx)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

10.1 导入数据

- `ExpressionSet` 来自于章节 九；
- `SupplementaryData` 来自于小节 6.5。

```
ExprSet_LIRI_JP <- readRDS("./data/result/ExpSetObject/Final_ExpSet_LIRI_JP_TrueCounts.RDS")
ExprSet_TCGA_LIHC <- readRDS("./data/result/ExpSetObject/Final_ExpSet_TCGA_LIHC_TrueCounts.RDS")

ExprSet_dis <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RD
ExprSet_val <- readRDS("./data/result/ExpSetObject/GSE14520_ExpSet_counts.RDS")
```

```
meatdata_sc <- readxl::read_xlsx("./data/GSE149614_scRNA/SupplementaryData/Supplementary Data 1.xlsx", s
```

10.2 汇总表格

```
meta_dis <- pData(ExprSet_dis)
meta_val <- pData(ExprSet_val)
meta_sc <- meatdata_sc %>%
  na.omit()
colnames(meta_sc) <- c(as.character(meatdata_sc[2, ]))

meta_tbl <- metadata %>%
  dplyr::select(-all_of(c("SampleID", "Status", "Time")))) %>%
 tbl_summary(by = "ProjectID") %>%
  add_p() %>%
  bold_labels()

meta_tbl
```

结果：本教程涉及到的肝细胞癌（HCC）患者的临床病理特征的数据分布情况

10.3 输出结果

```
if (!dir.exists("./data/result/metadata/")) {
  dir.create("./data/result/ExpSetObject", recursive = TRUE)
}

filename <- paste0("./data/result/metadata", "/HCC_summary_metadata", ".xlsx")
```

10.4 总结

总计位患者用于本教程。

系统信息

```
sessionInfo()
```

```
#> R version 4.3.3 (2024-02-29)
#> Platform: aarch64-apple-darwin20 (64-bit)
#> Running under: macOS Sonoma 14.2
#>
#> Matrix products: default
#> BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPAC
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> time zone: Asia/Shanghai
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats      graphics   grDevices datasets  utils      methods    base
#>
#> other attached packages:
#> [1] xlsx_0.6.5       gtsummary_1.7.2     Biobase_2.62.0
#> [4] BiocGenerics_0.48.1 lubridate_1.9.3    forcats_1.0.0
#> [7] stringr_1.5.1     dplyr_1.1.4       purrr_1.0.2
#> [10] readr_2.1.5      tidyverse_2.0.0    tibble_3.2.1
#> [13] ggplot2_3.5.1    tidyverse_2.0.0
#>
#> loaded via a namespace (and not attached):
#> [1] gt_0.10.1        xlsxjars_0.6.1    utf8_1.2.4
#> [4] generics_0.1.3    renv_1.0.0        xml2_1.3.6
#> [7] stringi_1.8.4    hms_1.1.3        digest_0.6.35
#> [10] magrittr_2.0.3   evaluate_0.23   grid_4.3.3
#> [13] timechange_0.3.0 sysfonts_0.8.9  fastmap_1.1.1
#> [16] broom.helpers_1.15.0 jsonlite_1.8.8 BiocManager_1.30.23
#> [19] fansi_1.0.6      scales_1.3.0     cli_3.6.2
#> [22] rlang_1.1.3     munsell_0.5.1    withr_3.0.0
#> [25] yaml_2.3.8      tools_4.3.3     tzdb_0.4.0
#> [28] colorspace_2.1-0 vctrs_0.6.5     R6_2.5.1
#> [31] lifecycle_1.0.4   pkgconfig_2.0.3   rJava_1.0-11
#> [34] pillar_1.9.0     gtable_0.3.5    glue_1.7.0
#> [37] xfun_0.43       tidyselect_1.2.1  rstudioapi_0.16.0
#> [40] knitr_1.46       htmltools_0.5.8.1 rmarkdown_2.26
#> [43] compiler_4.3.3
```

第三部分

差异分析

第十一章 差异分析之 limma

基因表达差异分析是一种用来比较不同条件下基因表达水平的方法，其目的是发现在不同条件下表达水平有显著差异的基因。这种分析通常在生物学研究中用于识别与特定生物过程、疾病或环境因素相关的基因。

11.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(Biostrings)
library(limma)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

11.2 读取函数

```
determine_tables <- function(
  object = NULL,
  data_counts = NULL,
  data_metadata = NULL,
  group,
  group_names = NULL,
```

©Hua

```

p_adjust = c("none", "fdr", "bonferroni", "holm",
            "hochberg", "hommel", "BH", "BY"),
da_method = NULL) {

# p_adjust
p_adjust <- match.arg(p_adjust,
                      c("none", "fdr", "bonferroni", "holm",
                        "hochberg", "hommel", "BH", "BY"))

}

# determine to use the ExpressionSet-class object
if (all(!is.null(object), !is.null(data_counts), !is.null(data_metadata))) {
  message("Either ExpressionSet object or counts and metadata as inputdata")
  message("Use the ExpressionSet object as default in the following processes")
}

# whether to extract tables from ExpressionSet
if (!is.null(object)) {
  counts <- Biobase:::exprs(object) %>%
    data.frame()
  metadata <- Biobase:::pData(object) %>%
    data.frame()
} else if (!is.null(data_counts)) {
  counts <- data_counts %>%
    data.frame()
  if (!is.null(data_metadata)) {
    metadata <- data_metadata %>%
      data.frame()
  } else {
    stop("Please load sample data")
  }
} else {
  stop("Please load ExpressionSet and counts")
}

# check whether group is valid
meta_nms <- names(metadata)
if (!group %in% meta_nms) {
  stop(
    group, " are not contained in the `pData` of `ExpressionSet`",

```

```

    call. = FALSE
  )
}

# groups split
if (!is.null(group_names)) {
  if (length(grep(":", group_names)) == 0) {
    group_names <- group_names
  } else {
    group_names <- unlist(strsplit(group_names, ":"), fixed = TRUE))
  }
}

# metadata table: row -> SampleID; column -> phenotypic index
sample_data_df <- metadata %>%
  data.frame()
colnames(sample_data_df)[which(colnames(sample_data_df) == group)] <- "Compvar"
if (!is.null(group_names)) {
  sam_tab <- sample_data_df %>%
    tibble::rownames_to_column("TempRowNames") %>%
    dplyr::filter(Compvar %in% group_names) %>%
    tibble::column_to_rownames("TempRowNames")
  sam_tab$Compvar <- factor(sam_tab$Compvar, levels = group_names)
} else {
  sam_tab <- sample_data_df
  group_names <- as.character(unique(sam_tab$Compvar))
  sam_tab$Compvar <- factor(sam_tab$Compvar, levels = group_names)
}

# check the levels of group are 2
if (length(unique(sam_tab$Compvar)) != 2) {
  stop(
    group, " contains more than 2 levels in `ps`, please input the names of comparison",
    call. = FALSE
  )
}
count_tab <- counts[, match(rownames(sam_tab), colnames(counts))]

# remove gene with constant in groups
## Judge whether the data are not 0 per groups

```

```

mdat <- dplyr::inner_join(sam_tab %>%
                           tibble::rownames_to_column("TempRowNames") %>%
                           dplyr::select(dplyr::all_of(c("TempRowNames", "Compvar"))),
                           count_tab %>%
                           t() %>% data.frame() %>%
                           tibble::rownames_to_column("TempRowNames"),
                           by = "TempRowNames") %>%
                           tibble::column_to_rownames("TempRowNames")

occ_fun <- function(x){
  length(x[c(which(!is.na(x) & x != 0))])/length(x)
}

mdat_zero_occ <- mdat %>% dplyr::group_by(Compvar) %>%
  dplyr::summarise(across(everything(), ~ occ_fun(.))) %>%
  tibble::column_to_rownames("Compvar") %>%
  t() %>% data.frame()

# change & (and) into | (or): 7/19/2022
mdat_zero_occ$KEEP <- ifelse(mdat_zero_occ[, 1] > 0.1 | mdat_zero_occ[, 2] > 0.1, TRUE, FALSE)
nfeature <- rownames(mdat_zero_occ)[mdat_zero_occ$KEEP]
mdat_remain <- mdat[, colnames(mdat) %in% c("Compvar", nfeature)]


## Judge whether the data are identical (such as all equal=1)
idential_res <- apply(mdat_remain[, -1], 2, function(x) {
  length(unique(x)[unique(x) != 0]) != 1
}) %>%
  data.frame()
nfeature_idential <- rownames(idential_res)[idential_res$.]
mdat_remain_v2 <- mdat_remain[, colnames(mdat_remain) %in% c("Compvar", nfeature_idential)]


## remove value appears to be constant with groups
## but the strict filtered criterion would remove too many taxa if the cutoff is 3
constant_res <- mdat_remain_v2 %>% dplyr::group_by(Compvar) %>%
  dplyr::summarise(across(.cols = everything(), .fns = function(x){length(unique(x)) > 0})) %>%
  tibble::column_to_rownames("Compvar") %>%
  t() %>% data.frame()

nfeature_no_constant <- data.frame()
for (i in 1:nrow(constant_res)) {
  nfeature_no_constant <- rbind(nfeature_no_constant,
                                 data.frame(FeatureID=rownames(constant_res)[i],
                                            KEEP=all(constant_res[i, 1], constant_res[i, 2])))
}

```

```

}

nfeature_no_constant_final <- rownames(idential_res)[nfeature_no_constant$KEEP]
mdat_remain_final <- mdat_remain[, colnames(mdat_remain) %in%
                                         c("Compvar", nfeature_no_constant_final)]


## final tables
otu_tab_final <- mdat_remain_final %>%
  dplyr::select(-Compvar) %>%
  t() %>% data.frame()
sam_tab_final <- sam_tab[pmatch(colnames(otu_tab_final), rownames(sam_tab)), , F]

# return results
res <- list(count = otu_tab_final,
            phen = sam_tab_final,
            nf = NULL,
            lib_size = NULL,
            group_names = group_names)

return(res)
}

calculate_occurrence <- function(profile, metadata, groups) {

  if (!all(rownames(metadata) == colnames(profile))) {
    stop("Order of sampleID between colData and proData is wrong please check your inputdata")
  }

  res <- apply(profile, 1, function(x, y) {
    dat <- data.frame(value = as.numeric(x), group = y)
    occ <- tapply(dat$value, dat$group, function(x){
      length(x[c(which(!is.na(x) & x != 0))]) / length(x)
    }) %>%
      data.frame() %>% stats::setNames("occ") %>%
      tibble::rownames_to_column("Group")
    occ1 <- occ[occ$Group %in% groups[1], "occ"]
    occ2 <- occ[occ$Group %in% groups[2], "occ"]
    occall <- length(dat$value[c(which(!is.na(dat$value) & dat$value != 0))]) / length(dat$value)

    # 100%
    occ1_percentage <- round(occ1, 4) * 100
  })
}

```

```

occ2_percentage <- round(occ2, 4) * 100
occall_percentage <- round(occall, 4) * 100

res <- c(occall_percentage, occ1_percentage, occ2_percentage)
return(res)
}, metadata$Compvar) %>%
t() %>% data.frame() %>%
tibble::rownames_to_column("FeatureID")

colnames(res) <- c("FeatureID",
"Occurrence (100%)\n(All)",
paste0("Occurrence (100%)\n", groups))

return(res)
}

run_OddRatio <- function(datx, daty, GroupName) {

# glm result for odd ratios 95%CI
mdat <- dplyr::inner_join(datx %>% tibble::rownames_to_column("TempRowNames") %>%
dplyr::select(dplyr::all_of(c("TempRowNames", "Compvar"))),
daty %>% t() %>% data.frame() %>%
tibble::rownames_to_column("TempRowNames"),
by = "TempRowNames") %>%
tibble::column_to_rownames("TempRowNames")

# Judge whether the data are not 0 per groups
occ_fun <- function(x){
length(x[c(which(!is.na(x) & x != 0))]) / length(x)
}
mdat_zero_occ <- mdat %>%
dplyr::group_by(Compvar) %>%
dplyr::summarise(across(everything(), ~ occ_fun(.))) %>%
tibble::column_to_rownames("Compvar") %>%
t() %>% data.frame()
mdat_zero_occ$KEEP <- ifelse(mdat_zero_occ[, 1] > 0 & mdat_zero_occ[, 2] > 0, TRUE, FALSE)
nfeature <- rownames(mdat_zero_occ)[mdat_zero_occ$KEEP]
mdat_remain <- mdat[, colnames(mdat) %in% c("Compvar", nfeature)]

# Judge whether the data are identical (such as all equal=1)

```

```

idential_res <- apply(mdat_remain[, -1], 2, function(x) {
  length(unique(x)[unique(x) != 0]) != 1
}) %>%
  data.frame()
nfeature_idential <- rownames(idential_res)[idential_res$.]
mdat_remain_final <- mdat_remain[, colnames(mdat_remain) %in% c("Compvar", nfeature_idential)]
```

```

dat_phe <- mdat_remain_final %>%
  dplyr::select(Compvar) %>%
  dplyr::mutate(Compvar = ifelse(Compvar == GroupName[2], 1, 0))
dat_prf <- mdat_remain_final %>% dplyr::select(-Compvar)
```

```

glmFun <- function(GroupN, MarkerN) {

  MarkerN[MarkerN == 0] <- min(MarkerN[MarkerN != 0])
  dat_glm <- data.frame(group = GroupN,
                         marker = scale(MarkerN, center = TRUE, scale = TRUE)) %>%
    na.omit()
  model <- summary(stats::glm(group ~ marker, data = dat_glm,
                                family = binomial(link = "logit")))
  res <- signif(exp(model$coefficients["marker", 1]) +
    qnorm(c(0.025, 0.5, 0.975)) * model$coefficients["marker", 1], 2)

  return(res)
}
```

```

glm_res <- t(apply(dat_prf, 2, function(x, group) {
  res <- glmFun(group, as.numeric(x))
  return(res)
}), dat_phe$Compvar))

Odd <- glm_res %>% data.frame() %>%
  setNames(c("upper", "expected", "lower")) %>%
  dplyr::mutate("Odds Ratio (95% CI)" = paste0(expected, " (", lower, ";", upper, ")"))
Odd$FeatureID <- rownames(glm_res)

res_ratain <- Odd[, c(5, 4)]

# drop features
drop_features <- dplyr::setdiff(rownames(daty), nfeature_idential)

```

```

if (length(drop_features) > 0) {
  res_drop <- data.frame(FeatureID = drop_features,
                         Value = NA) %>%
    stats::setNames(c("FeatureID", "Odds Ratio (95% CI)"))

  res <- rbind(res_ratain, res_drop)
} else {
  res <- res_ratain
}

return(res)
}

calculate_median_abundance <- function(profile, metadata, groups) {

  if (!all(rownames(metadata) == colnames(profile))) {
    stop("Order of sampleID between colData and proData is wrong please check your inputdata")
  }

  res <- apply(profile, 1, function(x, y) {
    dat <- data.frame(value = as.numeric(x), group = y)
    mn <- tapply(dat$value, dat$group, median) %>%
      data.frame() %>% setNames("value") %>%
      tibble::rownames_to_column("Group")
    mn1 <- with(mn, mn[Group %in% groups[1], "value"])
    mn2 <- with(mn, mn[Group %in% groups[2], "value"])
    mnall <- median(dat$value)

    if (all(mn1 != 0, mn2 != 0)) {
      Log2median <- log2(mn1 / mn2)
    } else {
      Log2median <- NA
    }

    res <- c(Log2median, mnall, mn1, mn2)
  })
  return(res)
}, metadata$Compvar) %>%
  t() %>% data.frame() %>%
  tibble::rownames_to_column("FeatureID")

```

```
colnames(res) <- c("FeatureID",
                     paste0("Log2FoldChange (Median)\n", paste(groups, collapse = "_vs_")),
                     "Median Abundance\n(All)",
                     paste0("Median Abundance\n", groups))

return(res)
}

calculate_mean_abundance <- function(profile, metadata, groups) {

  if (!all(rownames(metadata) == colnames(profile))) {
    stop("Order of sampleID between colData and proData is wrong please check your inputdata")
  }

  res <- apply(profile, 1, function(x, y) {
    dat <- data.frame(value = as.numeric(x), group = y)
    mn <- tapply(dat$value, dat$group, mean) %>%
      data.frame() %>% stats::setNames("value") %>%
      tibble::rownames_to_column("Group")
    mn1 <- with(mn, mn[Group %in% groups[1], "value"])
    mn2 <- with(mn, mn[Group %in% groups[2], "value"])
    mnall <- mean(dat$value)

    if (all(mn1 != 0, mn2 != 0)) {
      Log2mean <- log2(mn1 / mn2)
    } else {
      Log2mean <- NA
    }
  }

  res <- c(Log2mean, mnall, mn1, mn2)
  return(res)
}, metadata$Compvar) %>%
  t() %>% data.frame() %>%
  tibble::rownames_to_column("FeatureID")

colnames(res) <- c("FeatureID",
                  paste0("Log2FoldChange (Mean)\n", paste(groups, collapse = "_vs_")),
                  "Mean Abundance\n(All)",
                  paste0("Mean Abundance\n", groups))
```

©Hua

```

    return(res)
}

create_contrast <- function(groups, contrast = NULL) {
  if (!is.factor(groups)) {
    groups <- factor(groups)
  }
  lvl <- levels(groups)
  n_lvl <- length(lvl)
  if (n_lvl < 2) {
    stop("Differential analysis requires at least two groups.")
  }

  if (n_lvl == 2) { # two groups
    if (!is.null(contrast)) {
      warning(
        "`contrast` is ignored, you do not need to set it",
        call. = FALSE
      )
    }
    design <- rep(0, n_lvl)
    design[1] <- -1
    design[2] <- 1
  }

  return(design)
}

```

上述函数可以存到本地 `utilities.R` 脚本，可以通过 `source("utilities.R")` 加载函数。

11.3 导入数据

- `ExpressionSet` 来自于章节 九

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

11.4 差异分析函数

```
#| warning: false
#| message: false

run_limma_voom <- function(
  object = NULL,
  data_counts = NULL,
  data_metadata = NULL,
  group,
  group_names = NULL,
  voom_span = 0.5,
  p_adjust = "BH",
  qvalue_cutoff = 0.05) {

  # extract tables
  dat_tables <- determine_tables(
    object = object,
    data_counts = data_counts,
    data_metadata = data_metadata,
    group = group,
    group_names = group_names,
    p_adjust = p_adjust)

  otu_tab <- dat_tables$count
  sam_tab <- dat_tables$phen
  nf <- dat_tables$nf
  lib_size <- colSums(dat_tables$count)
  group_names <- dat_tables$group_names

  # calculate occurrence for group each taxa
  occurrence_taxa <- calculate_occurrence(otu_tab, sam_tab, group_names)
  # calculate median abundance per group for enriched directory
  median_abundance <- calculate_median_abundance(otu_tab, sam_tab, group_names)
  # calculate mean abundance per group
  mean_abundance <- calculate_mean_abundance(otu_tab, sam_tab, group_names)
  # 95% CI Odd Ratio
  odd_Ratio <- run_OddRatio(sam_tab, otu_tab, group_names)

  # building contrast object
```

©Hua

```

lvl <- levels(sam_tab$Compvar)
n_lvl <- length(lvl)
groups <- sam_tab$Compvar
contrast <- create_contrast(groups, group_names)

# design matrix
design <- model.matrix(~ 0 + groups)

# DA analysis:
res_temp <- data.frame(
  FeatureID = rownames(test_df),
  Enrichment = enrich_group,
  EffectSize = ef,
  logFC = test_df$logFC,
  Pvalue = test_df$P.Value,
  AdjustedPvalue = test_df$adj.P.Val) %>%
  dplyr::inner_join(median_abundance, by = "FeatureID") %>%
  dplyr::inner_join(mean_abundance, by = "FeatureID") %>%
  dplyr::inner_join(occurrence_taxa, by = "FeatureID")

res_temp$Enrichment <- ifelse(res_temp$AdjustedPvalue < qvalue_cutoff,
                                res_temp$Enrichment, "Nonsignif")

# Number of Group
dat_status <- table(sam_tab$Compvar)
dat_status_number <- as.numeric(dat_status)
dat_status_name <- names(dat_status)
res_temp$Block <- paste(paste(dat_status_number[1], dat_status_name[1], sep = "_"),
                        "vs",
                        paste(dat_status_number[2], dat_status_name[2], sep = "_"))

if (nrow(odd_Ratio) != 0) {
  res_final <- res_temp[, c(1, 18, 2:17)] %>%
    dplyr::inner_join(odd_Ratio, by = "FeatureID")
} else {
  res_final <- res_temp[, c(1, 18, 2:17)]
}

return(res_final)
}

```

11.5 运行差异分析

- 设置对应参数，运行 `run_limma_voom`

```

if (!dir.exists("./data/result/DA/")) {
  dir.create("./data/result/DA/", recursive = TRUE)
}

if (file.exists("./data/result/DA/HCC_Early_vs_Late_limma.csv")) {
  Early_vs_Late <- readr::read_csv("./data/result/DA/HCC_Early_vs_Late_limma.csv")
} else {
  Early_vs_Late <- run_limma_voom(
    object = ExprSet,
    group = "Group",
    group_names = c("Early Stage", "Late Stage"))
  write.csv(Early_vs_Late, "./data/result/DA/HCC_Early_vs_Late_limma.csv", row.names = F)
}

head(Early_vs_Late)

```

结果：输出结果包含了 19 列。它们分别是：

表 11.1: 差异分析 limma 的结果说明

列名	中文名称	含义
FeatureID	特征名称	基因名
Block	区块	分组检验数目
Enrichment	富集方向	简单的富集方向
EffectSize	组间效应值	两组均值差异比值
logFC	组间 log 倍数	limma 的 logFC
Pvalue	检验 P 值	limma 的 pvalue
AdjustedPvalue	检验 BH 值	limma 的多重检验校正后的 pvalue
Log2FoldChange (Median) Early	中位数的 log2 倍数	
Stage_vs_Late Stage		
Median Abundance (All)	中位数丰度	
Median Abundance Early Stage	中位数 Early 组丰度	
Median Abundance Late Stage	中位数 Late 组丰度	
Log2FoldChange (Mean) Early	组间 log 倍数	
Stage_vs_Late Stage		

列名	中文名称	含义
Mean Abundance (All)	平均值丰度	
Mean Abundance Early Stage	平均值 Early 组丰度	
Mean Abundance Late Stage	平均值 Late 组丰度	
Occurrence (100%) (All)	出现率	
Occurrence (100%) Early Stage	Early 组出现率	
Occurrence (100%) Late Stage	Late 组出现率	
Odds Ratio (95% CI)	比值比	95% 置信区间比值比

- 对小节 11.5 上述结果进行过滤，挑选显著差异的特征
 - 1) qvalue < 0.05;
 - 2) |log2foldchange| >= 0.5;
 - 3) |log2foldchange| <= 50;
 - 4) prevalence > 50

```

qval_cutoff <- 0.05
lgfc_cutoff <- 0.5
lgfc_max <- 50
prev_cutoff <- 50

Early_vs_Late_signif <- Early_vs_Late %>%
  dplyr::filter(AdjustedPvalue < qval_cutoff) %>%
  dplyr::filter(abs(logFC) >= lgfc_cutoff) %>%
  dplyr::filter(abs(logFC) <= lgfc_max) %>%
  dplyr::filter(`Occurrence (100%)\\nEarly Stage` > prev_cutoff) %>%
  dplyr::filter(`Occurrence (100%)\\nLate Stage` > prev_cutoff)
# dplyr::filter(`Occurrence..100...Early.Stage` > prev_cutoff) %>%
# dplyr::filter(`Occurrence..100...Late.Stage` > prev_cutoff)

table(Early_vs_Late_signif$Enrichment)

```

结果：分别获得富集在 Early 和 Late 分组的差异基因，这些差异基因将用于后续特征选择。

11.6 输出结果

```
if (!dir.exists("./data/result/DA")) {
```

```

dir.create("./data/result/DA/", recursive = TRUE)
}

write.csv(Early_vs_Late_signif, "./data/result/DA/HCC_Early_vs_Late_limma_select.csv", row.names =

```

11.7 总结

在基因标记的探索和识别过程中，确保选取合适的差异分析方法和评估差异基因的指标是至关重要的步骤。为了有效识别出具有统计学显著性的基因差异，本节研究采用了广泛认可和应用的 `limma` 差异分析包。

系统信息

```

sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets   utils      methods    base

other attached packages:
[1] limma_3.58.1        Biobase_2.62.0       BiocGenerics_0.48.1
[4] data.table_1.15.4   lubridate_1.9.3    forcats_1.0.0
[7] stringr_1.5.1       dplyr_1.1.4         purrr_1.0.2
[10] readr_2.1.5        tidyverse_2.0.0
[13] ggplot2_3.5.1

loaded via a namespace (and not attached):
[1] gtable_0.3.5        jsonlite_1.8.8       compiler_4.3.3

```

[4] BiocManager_1.30.23 renv_1.0.0 tidyselect_1.2.1
[7] scales_1.3.0 statmod_1.5.0 yaml_2.3.8
[10] fastmap_1.1.1 R6_2.5.1 generics_0.1.3
[13] knitr_1.46 munsell_0.5.1 pillar_1.9.0
[16] tzdb_0.4.0 rlang_1.1.3 utf8_1.2.4
[19] stringi_1.8.4 xfun_0.43 timechange_0.3.0
[22] cli_3.6.2 withr_3.0.0 magrittr_2.0.3
[25] digest_0.6.35 grid_4.3.3 rstudioapi_0.16.0
[28] hms_1.1.3 lifecycle_1.0.4 vctrs_0.6.5
[31] evaluate_0.23 glue_1.7.0 fansi_1.0.6
[34] colorspace_2.1-0 rmarkdown_2.26 tools_4.3.3
[37] pkgconfig_2.0.3 htmltools_0.5.8.1

第十二章 差异结果的火山图

火山图（Volcano Plot）是一种用于展示基因差异表达分析结果的二维散点图。它通过同时展示统计显著性和变化幅度，帮助研究者识别出在不同条件下显著差异表达的基因。火山图的横轴通常表示基因表达变化的倍数对数 (log₂ fold change)，纵轴表示统计显著性的负对数值 (-log₁₀ p-value)。这种图表因其形状类似火山而得名。

12.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(Biobase)
library(MicrobiomeAnalysis)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

12.2 导入数据

- 差异结果来自于章节 [十一](#)；
- `ExpressionSet` 来自于章节 [九](#)。

```
da_res_all <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma.csv")
da_res_signif <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma_select.csv")
```

©HuaJ

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

12.3 画图函数

12.4 火山图

- 设置对应参数，运行 `get_volcano`

```
DEG_vol <- get_volcano(  
  datsignif = da_res_all,  
  group_names = grp_names,  
  x_name = "logFC",  
  x_name_cutoff = 0.5,  
  y_name = "AdjustedPvalue",  
  y_name_cutoff = 0.05,  
  group_colors = c(grp_colors[1], "grey", grp_colors[2]),  
  topN = 10,  
  add_enrich_arrow = TRUE)  
  
DEG_vol
```

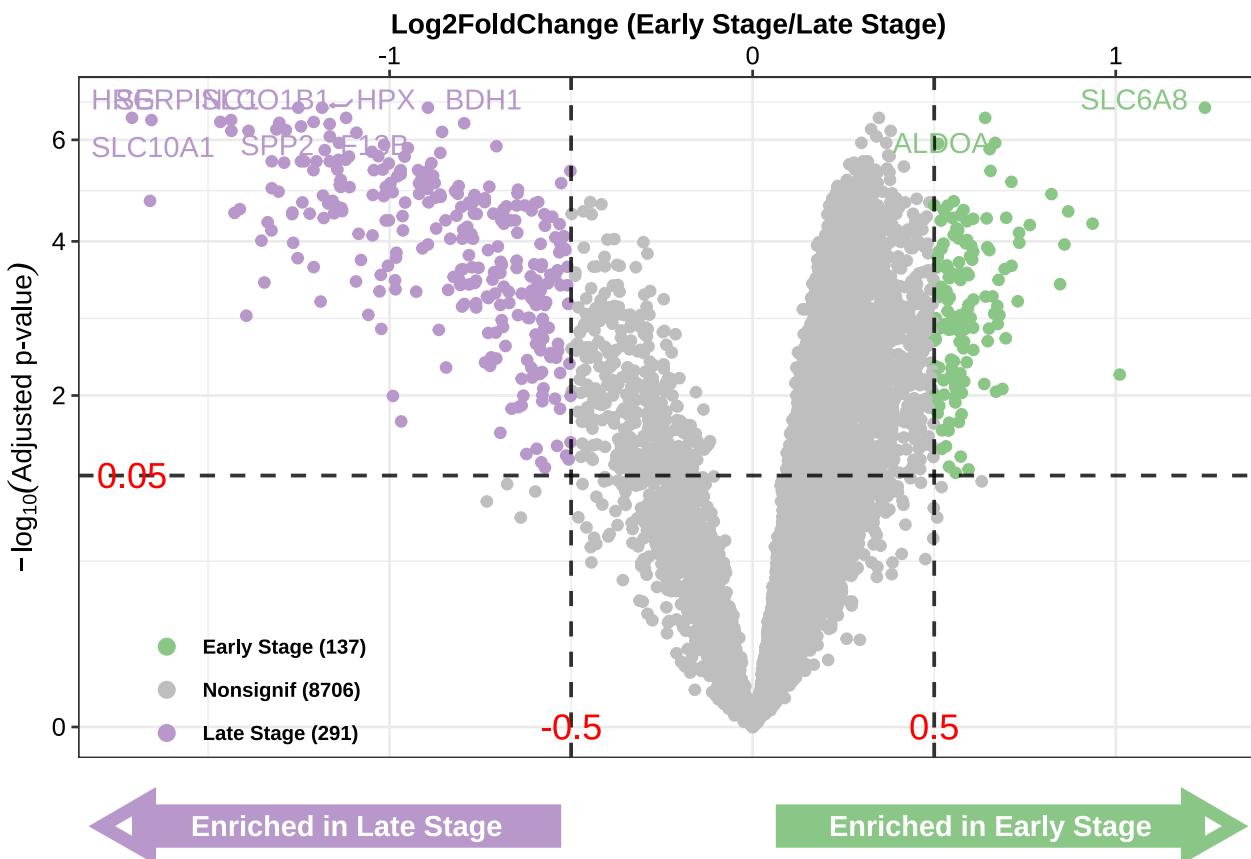


图 12.1: 差异基因的火山图 (Fig2-A)

结果：火山图展示了富集在不同分组的基因情况

- X 轴是 limma 的 logfc 的结果评估基因富集方向和大小；
- Y 轴是 limma 的 AdjustedPvalue 的结果判断基因显著性；
- 散点图的颜色表示基因的属性，下方箭头表示富集方向；
- 从图中可以得知，总计 428 个基因在特定阈值下 ($\text{abs}(\text{logFC}) > 0.5$, $\text{AdjustedPvalue} < 0.05$) 被筛选出来；
- 图中展示的基因 (*SPP2*, *SLC6A8*) 是分别在两组高表达的差异基因。

12.5 输出结果

```
if (!dir.exists("./data/result/Figure")) {
  dir.create("./data/result/Figure", recursive = TRUE)
}
```

```
ggsave("./data/result/Figure/Fig2-A.pdf", DEG_vol, width = 10, height = 7, dpi = 600)
```

12.6 总结

在成功获取差异基因的分析结果后，采用了火山图（Volcano Plot）这一强大的可视化工具来展示这些结果。

系统信息

```
sessionInfo()
```

```
R version 4.3.3 (2024-02-29)
```

```
Platform: aarch64-apple-darwin20 (64-bit)
```

```
Running under: macOS Sonoma 14.2
```

```
Matrix products: default
```

```
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
```

```
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version
```

```
locale:
```

```
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: Asia/Shanghai
```

```
tzcode source: internal
```

```
attached base packages:
```

```
[1] stats      graphics   grDevices datasets   utils      methods    base
```

```
other attached packages:
```

[1]	MicrobiomeAnalysis_1.0.3	Biobase_2.62.0	BiocGenerics_0.48.1
[4]	data.table_1.15.4	lubridate_1.9.3	forcats_1.0.0
[7]	stringr_1.5.1	dplyr_1.1.4	purrr_1.0.2
[10]	readr_2.1.5	tidyverse_2.0.0	tibble_3.2.1
[13]	ggplot2_3.5.1		

```
loaded via a namespace (and not attached):
```

[1]	fs_1.6.4	matrixStats_1.3.0
[3]	bitops_1.0-7	DirichletMultinomial_1.44.0
[5]	httr_1.4.7	RColorBrewer_1.1-3
[7]	doParallel_1.0.17	numDeriv_2016.8-1.1

```
[9] tools_4.3.3                         doRNG_1.8.6
[11] backports_1.4.1                      utf8_1.2.4
[13] R6_2.5.1                            vegan_2.6-4
[15] lazyeval_0.2.2                       mgcv_1.9-1
[17] rhdf5filters_1.14.1                  permute_0.9-7
[19] withr_3.0.0                          gridExtra_2.3
[21] cli_3.6.2                            sandwich_3.1-0
[23] labeling_0.4.3                        mvtnorm_1.2-4
[25] proxy_0.4-27                         yulab.utils_0.1.4
[27] foreign_0.8-86                        scater_1.30.1
[29] showtext_0.9-7                        decontam_1.22.0
[31] limma_3.58.1                          readxl_1.4.3
[33] rstudioapi_0.16.0                     sysfonts_0.8.9
[35] RSQLite_2.3.6                          generics_0.1.3
[37] shape_1.4.6.1                         gtools_3.9.5
[39] Matrix_1.6-5                           biomformat_1.30.0
[41] ggbeeswarm_0.7.2                      fansi_1.0.6
[43] DescTools_0.99.54                     S4Vectors_0.40.2
[45] DECIPHER_2.30.0                        abind_1.4-5
[47] lifecycle_1.0.4                        multcomp_1.4-25
[49] yaml_2.3.8                            SummarizedExperiment_1.32.0
[51] gplots_3.1.3.1                         rhdf5_2.46.1
[53] SparseArray_1.2.4                      grid_4.3.3
[55] blob_1.2.4                            crayon_1.5.2
[57] lattice_0.22-6                        beachmat_2.18.1
[59] cowplot_1.1.3                         pillar_1.9.0
[61] knitr_1.46                            GenomicRanges_1.54.1
[63] boot_1.3-30                           gld_2.6.6
[65] codetools_0.2-19                      glue_1.7.0
[67] MultiAssayExperiment_1.28.0          vctrs_0.6.5
[69] treeio_1.26.0                          Rdpack_2.6
[71] cellranger_1.1.0                      gtable_0.3.5
[73] cachem_1.0.8                          xfun_0.43
[75] rbibutils_2.2.16                      S4Arrays_1.2.1
[77] metagenomeSeq_1.43.0                  survival_3.7-0
[79] SingleCellExperiment_1.24.0           iterators_1.0.14
[81] tinytex_0.51                           showtextdb_3.0
[83] statmod_1.5.0                          bluster_1.12.0
[85] gmp_0.7-4                            TH.data_1.1-2
[87] nlme_3.1-164                          ANCOMBC_2.4.0
```

```
[89] phyloseq_1.46.0           bit64_4.0.5
[91] GenomeInfoDb_1.38.8        irlba_2.3.5.1
[93] vipor_0.4.7              KernSmooth_2.23-22
[95] rpart_4.1.23             colorspace_2.1-0
[97] DBI_1.2.2                Hmisc_5.1-2
[99] nnet_7.3-19              ade4_1.7-22
[101] Exact_3.2               DESeq2_1.42.1
[103] tidyselect_1.2.1         bit_4.0.5
[105] compiler_4.3.3          glmnet_4.1-8
[107] htmlTable_2.4.2          BiocNeighbors_1.20.2
[109] expm_0.999-9            DelayedArray_0.28.0
[111] checkmate_2.3.1         scales_1.3.0
[113] caTools_1.18.2          digest_0.6.35
[115] minqa_1.2.6             rmarkdown_2.26
[117] XVector_0.42.0           htmltools_0.5.8.1
[119] pkgconfig_2.0.3          base64enc_0.1-3
[121] lme4_1.1-35.3           sparseMatrixStats_1.14.0
[123] MatrixGenerics_1.14.0    fastmap_1.1.1
[125] rlang_1.1.3              htmlwidgets_1.6.4
[127] DelayedMatrixStats_1.24.0 farver_2.1.1
[129] zoo_1.8-12               jsonlite_1.8.8
[131] energy_1.7-11            BiocParallel_1.36.0
[133] BiocSingular_1.18.0      RCurl_1.98-1.14
[135] magrittr_2.0.3            Formula_1.2-5
[137] scuttle_1.12.0           GenomeInfoDbData_1.2.11
[139] Rhdf5lib_1.24.2          munsell_0.5.1
[141] Rcpp_1.0.12               ape_5.8
[143] viridis_0.6.5            CVXR_1.0-12
[145] stringi_1.8.4            rootSolve_1.8.2.4
[147] zlibbioc_1.48.2          MASS_7.3-60.0.1
[149] plyr_1.8.9               parallel_4.3.3
[151] ggrepel_0.9.5            lmom_3.0
[153] Biostrings_2.70.3         splines_4.3.3
[155] multtest_2.58.0           hms_1.1.3
[157] locfit_1.5-9.9            igraph_2.0.3
[159] Wrench_1.20.0             rngtools_1.5.2
[161] reshape2_1.4.4            stats4_4.3.3
[163] ScaledMatrix_1.10.0       evaluate_0.23
[165] renv_1.0.0                BiocManager_1.30.23
[167] nloptr_2.0.3              tzdb_0.4.0
```

[169] foreach_1.5.2
[171] Rmpfr_0.9-5
[173] tidytree_0.4.6
[175] class_7.3-22
[177] lmerTest_3.1-3
[179] beeswarm_0.4.0
[181] cluster_2.1.6
[183] timechange_0.3.0

rsvd_1.0.5
e1071_1.7-14
viridisLite_0.4.2
gsl_2.1-8
memoise_2.0.1
IRanges_2.36.0
TreeSummarizedExperiment_2.10.0
mia_1.10.0

第十三章 差异结果的热图

热图是一种数据可视化工具，用于展示矩阵数据中的数值大小，通过颜色的深浅或色调变化来表示不同的数值。通常用于生物信息学、统计学和数据分析等领域，以便直观地比较和分析大量数据。

13.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(BioBase)
library(ComplexHeatmap)
library(circlize)
library(ggplotify)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

13.2 导入数据

- 差异结果来自于章节 [十一](#)；
- `ExpressionSet` 来自于章节 [九](#)。

```
da_res_signif <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma_select.csv")
```

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

13.3 画图函数

调用 `ComplexHeatmap::Heatmap` 函数 (Gu, Eils, 和 Schlesner 2016) 对差异分析的结果即差异基因进行绘图, 采用常见的热图。`ComplexHeatmap::Heatmap` 的详细用法可以直接查看函数帮助文档。除此之外, 还使用 `circlize::colorRamp2` 函数 (Gu 等 2014) 调用颜色参数。

```
get_heatmap <- function(
  object,
  datsignif,
  group,
  group_names = grp_names,
  group_colors = grp_colors) {

  metadata <- pData(object) %>%
    as.data.frame()
  profile <- exprs(object) %>%
    as.data.frame()

  colnames(metadata)[which(colnames(metadata) == group)] <- "GroupCompvar"

  phen <- metadata %>%
    dplyr::arrange(GroupCompvar, Tumour_Stage, ProjectID)
  phen$GroupCompvar <- factor(phen$GroupCompvar, levels = group_names)
  prof <- profile[rownames(profile) %in% datsignif$FeatureID,
    pmatch(phen$SampleID, colnames(profile)), ] %>%
    as.matrix()

  project_id <- unique(phen$ProjectID)
  project_colors <- c("#E69F00", "#56B4E9")
  names(project_colors) <- project_id

  stage_id <- unique(phen$Tumour_Stage)
  stage_colors <- c("#D51F26", "#272E6A", "#208A42", "#89288F")
  names(stage_colors) <- stage_id

  # scale prof
```

```

prof_scale <- scale(prof, center = T, scale = T)

pl <- ComplexHeatmap::Heatmap(
  mat = prof_scale,
  col = circlize::colorRamp2(c(-3, 0, 3), c("blue", "white", "red")),
  top_annotation = top_anno,
  cluster_columns = FALSE,
  cluster_rows = TRUE,
  show_row_names = FALSE,
  show_column_names = FALSE,
  column_split = phen$GroupCompvar,
  heatmap_legend_param = list(
    legend_direction = "vertical",
    legend_width = unit(6, "cm"),
    title = "Relative Abundance"))

return(pl)
}

```

13.4 热图

- 设置对应参数，运行 `get_heatmap`

```

DEG_heat <- get_heatmap(
  object = ExprSet,
  datsignif = da_res_signif,
  group = "Group",
  group_names = grp_names,
  group_colors = grp_colors)

```

`DEG_heat`

结果：热图展示了差异基因的表达状态

- X 轴是样本 ID；
- Y 轴是基因 ID；

13.5 输出结果

使用 `ggplotify::as.ggplot` 函数 (Yu 2014) 转换 `ComplexHeatmap::Heatmap` 函数 (Gu, Eils, 和 Schlesner 2016) 获得的热图。

```
if (!dir.exists("./data/result/Figure")) {  
  dir.create("./data/result/Figure", recursive = TRUE)  
}  
  
ggsave("./data/result/Figure/Fig2-B.pdf", ggplotify::as.ggplot(DEG_heat), width = 12, height = 7, c
```

13.6 总结

在成功获得差异基因的分析结果后，进一步提取了这些差异基因在每个样本中的表达值。

系统信息

```
sessionInfo()  
  
R version 4.3.3 (2024-02-29)  
Platform: aarch64-apple-darwin20 (64-bit)  
Running under: macOS Sonoma 14.2  
  
Matrix products: default  
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib  
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v  
  
locale:  
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8  
  
time zone: Asia/Shanghai  
tzcode source: internal  
  
attached base packages:  
[1] grid      stats     graphics   grDevices datasets  utils      methods  
[8] base  
  
other attached packages:  
[1] ggplotify_0.1.2      circlize_0.4.16      ComplexHeatmap_2.18.0  
[4] Biobase_2.62.0       BiocGenerics_0.48.1    data.table_1.15.4
```

©Hua_

```
[7] lubridate_1.9.3     forcats_1.0.0      stringr_1.5.1
[10] dplyr_1.1.4        purrr_1.0.2       readr_2.1.5
[13] tidyverse_2.0.0     tibble_3.2.1      ggplot2_3.5.1
```

```
loaded via a namespace (and not attached):
[1] gtable_0.3.5      shape_1.4.6.1     rjson_0.2.21
[4] xfun_0.43         GlobalOptions_0.1.2 tzdb_0.4.0
[7] yulab.utils_0.1.4 vctrs_0.6.5      tools_4.3.3
[10] generics_0.1.3   stats4_4.3.3     parallel_4.3.3
[13] fansi_1.0.6      cluster_2.1.6    pkgconfig_2.0.3
[16] RColorBrewer_1.1-3 S4Vectors_0.40.2 lifecycle_1.0.4
[19] compiler_4.3.3   munsell_0.5.1     codetools_0.2-19
[22] clue_0.3-65      htmltools_0.5.8.1  yaml_2.3.8
[25] pillar_1.9.0     crayon_1.5.2     cachem_1.0.8
[28] iterators_1.0.14 foreach_1.5.2     tidyselect_1.2.1
[31] digest_0.6.35    stringi_1.8.4    fastmap_1.1.1
[34] colorspace_2.1-0 cli_3.6.2       magrittr_2.0.3
[37] utf8_1.2.4       withr_3.0.0      scales_1.3.0
[40] timechange_0.3.0 rmarkdown_2.26   matrixStats_1.3.0
[43] png_0.1-8        GetoptLong_1.0.5   hms_1.1.3
[46] memoise_2.0.1    evaluate_0.23   knitr_1.46
[49] IRanges_2.36.0   doParallel_1.0.17 gridGraphics_0.5-1
[52] rlang_1.1.3      glue_1.7.0       BiocManager_1.30.23
[55] renv_1.0.0       rstudioapi_0.16.0 jsonlite_1.8.8
[58] R6_2.5.1         fs_1.6.4
```

第四部分

功能分析

第十四章 GO 富集分析

GO (Gene Ontology) 是一个在生物信息学中广泛使用的概念，用于描述基因和基因产物的功能、它们所处的细胞位置以及它们参与的生物过程。GO 项目是一个协作性的国际努力，旨在建立和维护一个适用于各种物种的、结构化的、可控制的词汇表（本体论），用以描述生物学的各个方面。具体来说，GO 分为三个主要方面：

1. 分子功能 (Molecular Function, MF): 描述基因产物在分子水平上的活动，如催化活性或结合能力。
2. 生物过程 (Biological Process, BP): 描述生物学上发生的一系列事件，如细胞生长、代谢过程或信号传导。
3. 细胞组件 (Cellular Component, CC): 描述基因产物在细胞内的位置，如细胞核、线粒体或细胞膜。

14.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

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

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

14.2 导入数据

- 差异结果来自于章节 十一；
- ExpressionSet 来自于章节 九。

```
da_res <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma.csv")
```

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

14.3 所需函数

- get_DEGs 获得不同类型的差异基因列表；
- get_ORA 基于 *Over-Representative analysis (ORA)* 方法功能富集分析；
- 使用 clusterProfiler::enrichGO(Yu 等 2012) 做 GO 分析。

```
get_DEGs <- function(
  dat,
  lg2fc_cutoff = 0.5,
  pval_cutoff = 0.05,
  qval_cutoff = 0.05) {

  # group_names
  group_names <- gsub("\\d+", "", unlist(strsplit(dat$Block[1], " vs ")))
  colnames(dat)[5] <- "lg2fc"
  colnames(dat)[6] <- "pval"
  colnames(dat)[7] <- "qval"

  # enrichment by beta and Pvalue AdjustedPvalue
  dat[which(dat$lg2fc > lg2fc_cutoff &
             dat$pval < pval_cutoff &
             dat$qval < qval_cutoff),
      "EnrichedDir"] <- group_names[2]
  dat[which(dat$lg2fc < -lg2fc_cutoff &
             dat$pval < pval_cutoff &
             dat$qval < qval_cutoff),
      "EnrichedDir"] <- group_names[1]
  dat[which(abs(dat$lg2fc) <= lg2fc_cutoff |
```

```

        dat$pval >= pval_cutoff |  

        dat$qval >= qval_cutoff),  

    "EnrichedDir"] <- "Nonsignif"  
  

# dat status  

dat$EnrichedDir <- factor(dat$EnrichedDir,  

                           levels = c(group_names[2], "Nonsignif", group_names[1]))  

df_status <- table(dat$EnrichedDir) %>% data.frame() %>%  

  stats::setNames(c("Group", "Number"))  

grp1_number <- with(df_status, df_status[Group %in% group_names[1], "Number"])  

grp2_number <- with(df_status, df_status[Group %in% group_names[2], "Number"])  

nsf_number <- with(df_status, df_status[Group %in% "Nonsignif", "Number"])  

legend_label <- c(paste0(group_names[1], " (", grp1_number, ")"),  

                  paste0("Nonsignif", " (", nsf_number, ")"),  

                  paste0(group_names[2], " (", grp2_number, ")"))  
  

res_up <- dat_signif %>% # enriched in 1st group  

  dplyr::filter(EnrichedDir == group_names[1]) %>%  

  dplyr::mutate(Status = "Up_regulated")  
  

res_down <- dat_signif %>% # enriched in 2st group  

  dplyr::filter(EnrichedDir == group_names[2]) %>%  

  dplyr::mutate(Status = "Down_regulated")  
  

res <- list(all = dat_signif,  

           up = res_up,  

           down = res_down)  
  

return(res)
}  
  

get_ORA <- function(  

  genelist,  

  genotype = c("all", "up", "down"),  

  ORAtype = c("GO", "KEGG"),  

  showcase = 10,  

  group_names = grp_names) {  
  

  res <- list(fit = fit,
              result = fit_result,

```

```

    pl = pl)

return(res)
}

```

14.4 运行

- 设置对应参数，运行 `get_DEGs`

```

df_DEGs <- get_DEGs(dat = da_res)

names(df_DEGs)

```

- 使用 ORA 方法计算所有差异基因的富集 GO term 结果

```

All_ORA_GO <- get_ORA(
  genelist = df_DEGs$all,
  genetype = "all",
  ORAtype = "GO",
  showcase = 10)

```

All_ORA_GO\$pl

结果：该图片展示了基于基因本体论（Gene Ontology, GO）的生物信息学分析结果，主要关注细胞过程、分子功能和细胞组件的多个方面。分析结果显示，在肝细胞癌（HCC）相关的基因中，有几个关键的生物过程和分子功能显著富集。

14.5 输出结果

```

if (!dir.exists("./data/result/Figure")) {
  dir.create("./data/result/Figure", recursive = TRUE)
}

ggsave("./data/result/Figure/Fig3-A.pdf", All_ORA_GO$pl, width = 8, height = 6, dpi = 600)

```

14.6 总结

通过进行 *GO* (*Gene Ontology*) 富集分析，能够深入理解差异基因在细胞内的功能角色，这一分析为后续关于肝细胞癌 (HCC) 的研究奠定了坚实的生物学基础。

系统信息

```
sessionInfo()
```

```
R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2
```

```
Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version
```

```
locale:
```

```
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: Asia/Shanghai
tzcode source: internal
```

```
attached base packages:
```

```
[1] stats4      stats       graphics   grDevices datasets   utils       methods
[8] base
```

```
other attached packages:
```

```
[1] org.Hs.eg.db_3.18.0    AnnotationDbi_1.66.0    IRanges_2.36.0
[4] S4Vectors_0.40.2       Biobase_2.62.0        BiocGenerics_0.48.1
[7] clusterProfiler_4.10.1  lubridate_1.9.3     forcats_1.0.0
[10] stringr_1.5.1         dplyr_1.1.4          purrrr_1.0.2
[13] readr_2.1.5           tidyverse_2.0.0       tibble_3.2.1
[16] ggplot2_3.5.1         tidyverse_2.0.0
```

```
loaded via a namespace (and not attached):
```

```
[1] DBI_1.2.2            bitops_1.0-7        gson_0.1.0
[4] shadowtext_0.1.3     gridExtra_2.3       rlang_1.1.3
[7] magrittr_2.0.3       DOSE_3.28.2        compiler_4.3.3
[10] RSQLite_2.3.6        png_0.1-8          vctrs_0.6.5
[13] reshape2_1.4.4       pkgconfig_2.0.3     crayon_1.5.2
```

[16] fastmap_1.1.1 XVector_0.42.0 ggraph_2.2.1
[19] utf8_1.2.4 HDO.db_0.99.1 rmarkdown_2.26
[22] tzdb_0.4.0 enrichplot_1.22.0 bit_4.0.5
[25] xfun_0.43 zlibbioc_1.48.2 cachem_1.0.8
[28] aplot_0.2.2 GenomeInfoDb_1.38.8 jsonlite_1.8.8
[31] blob_1.2.4 BiocParallel_1.36.0 tweenr_2.0.3
[34] parallel_4.3.3 R6_2.5.1 RColorBrewer_1.1-3
[37] stringi_1.8.4 GOSemSim_2.28.1 Rcpp_1.0.12
[40] knitr_1.46 Matrix_1.6-5 splines_4.3.3
[43] igraph_2.0.3 timechange_0.3.0 tidyselect_1.2.1
[46] qvalue_2.34.0 rstudioapi_0.16.0 yaml_2.3.8
[49] viridis_0.6.5 codetools_0.2-19 lattice_0.22-6
[52] plyr_1.8.9 treeio_1.26.0 withr_3.0.0
[55] KEGGREST_1.42.0 evaluate_0.23 gridGraphics_0.5-1
[58] scatterpie_0.2.2 polyclip_1.10-6 Biostrings_2.70.3
[61] ggtree_3.10.1 pillar_1.9.0 BiocManager_1.30.23
[64] renv_1.0.0 ggfunk_0.1.4 generics_0.1.3
[67] RCurl_1.98-1.14 hms_1.1.3 tidytree_0.4.6
[70] munsell_0.5.1 scales_1.3.0 glue_1.7.0
[73] lazyeval_0.2.2 tools_4.3.3 data.table_1.15.4
[76] fgsea_1.28.0 fs_1.6.4 graphlayouts_1.1.1
[79] fastmatch_1.1-4 tidygraph_1.3.1 cowplot_1.1.3
[82] grid_4.3.3 ape_5.8 colorspace_2.1-0
[85] nlme_3.1-164 patchwork_1.2.0 GenomeInfoDbData_1.2.11
[88] ggforce_0.4.2 cli_3.6.2 fansi_1.0.6
[91] viridisLite_0.4.2 gtable_0.3.5 yulab.utils_0.1.4
[94] digest_0.6.35 ggplotify_0.1.2 ggrepel_0.9.5
[97] farver_2.1.1 memoise_2.0.1 htmltools_0.5.8.1
[100] lifecycle_1.0.4 httr_1.4.7 G0.db_3.19.1
[103] bit64_4.0.5 MASS_7.3-60.0.1

第十五章 KEGG 通路富集分析

KEGG pathway 是京都基因与基因组百科全书（Kyoto Encyclopedia of Genes and Genomes，简称 KEGG）中的一个子数据库，它包括了代谢、调控、通路、生化、疾病、药物等相关的分子相互作用和关系网络。

KEGG 通路富集分析是一种用于分析高通量实验数据的方法，其目标是确定在实验中观察到的基因或其他实体是否集中在特定的 KEGG 通路中，以便推断这些通路在实验条件下是否显著富集。在生物学研究中，通过通路富集分析可以了解一组基因是否在某些已知的代谢通路、信号通路或细胞过程中显著富集，从而揭示这些基因在生物体内可能的功能和相互作用。这种分析方法通常利用 KEGG 等数据库提供的已知生物通路信息，结合超几何分布等方法计算给定基因集在某个通路上的 P 值，以判断该基因集在该通路上的富集程度是否显著。

15.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(clusterProfiler)
library(org.Hs.eg.db)
library(msigdbr)
library(massdatabase)
library(enrichplot)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

15.2 导入数据

- 差异结果来自于章节 [十一](#);
- ExpressionSet 来自于章节 [九](#)。

```
da_res <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma.csv")
```

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

15.3 所需函数

- `get_KEGG_category` 获得 KEGG pathway 的不同层级关系表;
- `get_GSEA` 基于 *GSEA* 方法功能富集分析;
- 使用 `clusterProfiler:::GSEA`([Yu 等 2012](#)) 做 GSEA 分析;
- 使用 `msigdbr:::msigdbr`([Dolgalev 2020](#)) 拿到基因和通路关系对应表。
- 下载 KEGG 关系表的函数，它们均来自于 `massdatabase` 包 ([Shen 等 2022](#))。

```
get_KEGG_category <- function(
  pathdir,
  sleeptime = 2,
  organ = "hsa") {

  KEGG_pathway_database <- massdatabase::convert_kegg2metpath(
    data = KEGG_data,
    path = pathdir)

  res <- KEGG_relation %>%
    dplyr::group_by(pathID) %>%
    dplyr::mutate(category = unlist(strsplit(pathClass, ";\\s+"))[1],
                  subcategory = unlist(strsplit(pathClass, ";\\s+"))[2]) %>%
    dplyr::ungroup() %>%
    dplyr::distinct() %>%
    dplyr::select(pathID, category, subcategory, pathName, Describtion)

  return(res)
}
```

```
}  
  
get_GSEA <- function(  
  genelist,  
  showcase = 10,  
  species_type = c("Mus musculus", "Homo sapiens"),  
  pathway_type = c("C2", "CP:KEGG"),  
  group_names = grp_names) {  
  
  # KEGG table  
  if (!dir.exists("./data/result/Function")) {  
    dir.create("./data/result/Function", recursive = TRUE)  
  }  
  
  hsa_kegg_tab <- get_KEGG_category(  
    pathdir = "./data/result/Function/KEGG/hsa_KEGG",  
    sleeptime = 2,  
    organ = "hsa") %>%  
    dplyr::filter(!subcategory %in%  
      c("Drug resistance: antineoplastic", "Cardiovascular disease",  
        "Infectious disease: parasitic", "Neurodegenerative disease",  
        "Aging", "Cellular community - eukaryotes", "Infectious disease: bacterial",  
        "Cancer: specific types"))  
  
  genelist_new <- genelist %>%  
    dplyr::arrange(desc(logFC))  
  
  inputgenes <- genelist_new$logFC  
  names(inputgenes) <- genelist_new$FeatureID  
  
  # download background genes  
  kegg_df_cln <- kegg_df %>%  
    dplyr::inner_join(hsa_kegg_tab %>%  
      dplyr::select(category, subcategory, pathName),  
      by = c("gs_description" = "pathName"))  
  
  # KEGG pathway (geneID)  
  fit <- clusterProfiler::GSEA(  
    geneList = inputgenes,  
    pvalueCutoff = 0.05,  
    pAdjustMethod = "BH",
```

```

minGSSize = 1,
maxGSSize = 500,
TERM2GENE = dplyr::select(
    kegg_df_cln,
    gs_description,
    gene_symbol),
#by = "DOSE",
#nPerm = 1000
by = "fgsea")

fit_result <- fit@result %>%
  as.data.frame() %>%
  dplyr::mutate(Order = 1:nrow(fit@result)) %>%
  dplyr::left_join(kegg_df_cln %>%
    dplyr::select(gs_description, category, subcategory) %>%
    dplyr::distinct(),
    by = c("ID" = "gs_description")) %>%
  dplyr::rename(Count = setSize) %>%
  dplyr::mutate(enrichmentDir = ifelse(NES > 0, group_names[1], group_names[2])) %>%
  dplyr::select(Order, ID, everything())

plotdata <- fit_result %>%
  dplyr::filter(!is.na>Description)) %>%
  dplyr::group_by(category) %>%
  dplyr::slice(1:showcase) %>%
  dplyr::ungroup() %>%
  dplyr::mutate(qvalue = as.numeric(qvalue),
    Count = as.numeric(Count),
    enrichmentScore = as.numeric(enrichmentScore)) %>%
  dplyr::mutate(ID = gsub("KEGG_", "", ID),
    ID = tolower(ID)) %>%
  dplyr::arrange(desc(category), enrichmentScore)

plotdata$Description <- factor(plotdata$Description,
  levels = unique(plotdata$Description))
plotdata$category <- factor(plotdata$category)

res <- list(fit = fit,
  result = fit_result,
  pl = pl)

```

```

    return(res)
}

```

15.4 运行

- 使用 GSEA 方法计算所有差异基因的富集 KEGG pathway 结果

```

if (file.exists("./data/result/Function/GSEA_res.RDS")) {
  GSEA_result <- readRDS("./data/result/Function/GSEA_res.RDS")
} else {
  GSEA_result <- get_GSEA(
    genelist = da_res,
    showcase = 10,
    species_type = "Homo sapiens",
    pathway_type = c("C2", "CP:KEGG"))

  saveRDS(GSEA_result, "./data/result/Function/GSEA_res.RDS", compress = TRUE)
}

GSEA_result$pl

```

结果：从图中，可以获得以下比较有用的知识：细胞周期（Cell cycle）和 p53 信号通路（p53 signaling pathway）在肝癌 HCC 中可能具有显著的影响。这两个过程与细胞的生长、增殖和凋亡紧密相关，对癌症的发生和发展至关重要。

- 在 Early 分期富集的 KEGG pathway

```

# Tryptophan metabolism: 14
# Butanoate metabolism: 18
# Nitrogen metabolism: 44
# One carbon pool by folate: 57

enrich_early <- enrichplot::gseaplot2(
  GSEA_result$fit,
  geneSetID = c(14, 18, 44, 57),
  #pvalue_table = TRUE,
  color = c("#D51F26", "#272E6A", "#208A42", "#89288F"),
  ES_geom = "line",

```

```

title = "Enriched in the early stage of HCC")

enrich_early

• 在 Late 分期富集的 KEGG pathway

# Cell cycle: 11
# Mismatch repair: 30
# p53 signaling pathway: 36
# ECM-receptor interaction: 42

enrich_late <- enrichplot::gseaplot2(
  GSEA_result$fit,
  geneSetID = c(11, 30, 36, 42),
  #pvalue_table = TRUE,
  color = c("#faa818", "#41a30d", "#fbdf72", "#367d7d"),
  ES_geom = "line",
  title = "Enriched in the late stage of HCC")

enrich_late

```

15.5 其他可视化方法

通过 **GseaVis**(Zhang 2022)R 包在特定富集通路添加基因标签，展示表达状态。

```

library(GseaVis)

p53_gene <- c("SFN", "SERPINE1", "CDK1", "CHEK1", "CCNB2",
              "CCNB1", "IGFBP3", "GTSE1", "CDK4", "CCNE1")

gseaNb(object = GSEA_result$fit,
        geneSetID = 'p53 signaling pathway',
        addGene = p53_gene,
        # newGsea = TRUE,
        # subPlot = 2,
        addPval = TRUE,
        pvalX = 0.6, pvalY = 0.8,
        pCol = "black",
        newHtCol = c("blue", "white", "red"),

```

```
pHjust = 0)
```

结果：富集结果增加了基因标签。

15.6 输出结果

```
if (!dir.exists("./data/result/Figure")) {
  dir.create("./data/result/Figure", recursive = TRUE)
}

ggsave("./data/result/Figure/Fig3-B.pdf", GSEA_result$pl, width = 8, height = 5, dpi = 600)
ggsave("./data/result/Figure/Fig3-C.pdf", enrich_early, width = 5, height = 4, dpi = 600)
ggsave("./data/result/Figure/Fig3-D.pdf", enrich_late, width = 5, height = 4, dpi = 600)
```

15.7 总结

KEGG 通路富集分析结果揭示在不同 HCC 癌症分期中，富集得到的通路也不一样。比如 p53 signaling pathway 只富集在癌症后期，而富集在前期的是 Tryptophan metabolism 等。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats4      stats       graphics   grDevices datasets   utils       methods
[8] base
```

other attached packages:

```
[1] enrichplot_1.22.0      MSnbase_2.28.1       ProtGenerics_1.34.0
[4] mzR_2.36.0            Rcpp_1.0.12          massdataset_1.0.29
[7] metid_1.2.30           metpath_1.0.8        magrittr_2.0.3
[10] masstools_1.0.13      massdatabase_1.0.10  msigdbr_7.5.1
[13] org.Hs.eg.db_3.18.0   AnnotationDbi_1.66.0  IRanges_2.36.0
[16] S4Vectors_0.40.2      Biobase_2.62.0       BiocGenerics_0.48.1
[19] clusterProfiler_4.10.1 lubridate_1.9.3      forcats_1.0.0
[22] stringr_1.5.1         dplyr_1.1.4          purrr_1.0.2
[25] readr_2.1.5           tidyverse_2.0.0       tibble_3.2.1
[28] ggplot2_3.5.1         tidyverse_2.0.0
```

loaded via a namespace (and not attached):

```
[1] splines_4.3.3          bitops_1.0-7
[3] ggplotify_0.1.2        cellranger_1.1.0
[5] polyclip_1.10-6        preprocessCore_1.64.0
[7] XML_3.99-0.16.1        lifecycle_1.0.4
[9] doParallel_1.0.17       globals_0.16.3
[11] lattice_0.22-6         MASS_7.3-60.0.1
[13] plotly_4.10.4          openxlsx_4.2.5.2
[15] limma_3.58.1           rmarkdown_2.26
[17] yaml_2.3.8             remotes_2.5.0
[19] zip_2.3.1              cowplot_1.1.3
[21] MsCoreUtils_1.14.1     pbapply_1.7-2
[23] DBI_1.2.2              RColorBrewer_1.1-3
[25] abind_1.4-5             zlibbioc_1.48.2
[27] rvest_1.0.4             GenomicRanges_1.54.1
[29] ggraph_2.2.1            RCurl_1.98-1.14
[31] yulab.utils_0.1.4       tweenr_2.0.3
[33] circlize_0.4.16         GenomeInfoDbData_1.2.11
[35] ggrepel_0.9.5            listenv_0.9.1
[37] tidytree_0.4.6           parallelly_1.37.1
[39] DelayedArray_0.28.0      ncdf4_1.22
[41] codetools_0.2-19         xml2_1.3.6
[43] DOSE_3.28.2              ggforce_0.4.2
[45] shape_1.4.6.1            tidyselect_1.2.1
[47] aplot_0.2.2              farver_2.1.1
[49] viridis_0.6.5             matrixStats_1.3.0
[51] jsonlite_1.8.8           GetoptLong_1.0.5
```

```
[53] tidygraph_1.3.1           iterators_1.0.14
[55] foreach_1.5.2            progress_1.2.3
[57] tools_4.3.3              treeio_1.26.0
[59] stringdist_0.9.12        glue_1.7.0
[61] SparseArray_1.2.4         gridExtra_2.3
[63] xfun_0.43                MatrixGenerics_1.14.0
[65] qvalue_2.34.0            GenomeInfoDb_1.38.8
[67] withr_3.0.0              BiocManager_1.30.23
[69] fastmap_1.1.1            fansi_1.0.6
[71] digest_0.6.35            timechange_0.3.0
[73] R6_2.5.1                 gridGraphics_0.5-1
[75] colorspace_2.1-0          GO.db_3.19.1
[77] RSQLite_2.3.6             utf8_1.2.4
[79] generics_0.1.3            renv_1.0.0
[81] data.table_1.15.4          prettyunits_1.2.0
[83] htmlwidgets_1.6.4          S4Arrays_1.2.1
[85] graphlayouts_1.1.1         httr_1.4.7
[87] scatterpie_0.2.2          pkgconfig_2.0.3
[89] gtable_0.3.5              blob_1.2.4
[91] ComplexHeatmap_2.18.0      impute_1.76.0
[93] XVector_0.42.0            furrr_0.3.1
[95] shadowtext_0.1.3          htmltools_0.5.8.1
[97] fgsea_1.28.0              MALDIquant_1.22.2
[99] clue_0.3-65               scales_1.3.0
[101] png_0.1-8                ggfun_0.1.4
[103] knitr_1.46                rstudioapi_0.16.0
[105] tzdb_0.4.0                reshape2_1.4.4
[107] rjson_0.2.21              nlme_3.1-164
[109] curl_5.2.1                cachem_1.0.8
[111] GlobalOptions_0.1.2       parallel_4.3.3
[113] HDO.db_0.99.1             mzID_1.40.0
[115] vsn_3.70.0                pillar_1.9.0
[117] grid_4.3.3                vctrs_0.6.5
[119] pcaMethods_1.94.0          cluster_2.1.6
[121] evaluate_0.23              cli_3.6.2
[123] compiler_4.3.3            rlang_1.1.3
[125] crayon_1.5.2              Rdisop_1.62.0
[127] affy_1.80.0                plyr_1.8.9
[129] fs_1.6.4                  stringi_1.8.4
[131] viridisLite_0.4.2          BiocParallel_1.36.0
```

[133] babelgene_22.9 munsell_0.5.1
[135] Biostrings_2.70.3 lazyeval_0.2.2
[137] GOSemSim_2.28.1 Matrix_1.6-5
[139] hms_1.1.3 patchwork_1.2.0
[141] bit64_4.0.5 future_1.33.2
[143] KEGGREST_1.42.0 statmod_1.5.0
[145] SummarizedExperiment_1.32.0 igraph_2.0.3
[147] memoise_2.0.1 affyio_1.72.0
[149] ggtree_3.10.1 fastmatch_1.1-4
[151] bit_4.0.5 readxl_1.4.3
[153] ape_5.8 gson_0.1.0

第十六章 ssGSEA 富集分析

ssGSEA（单样本基因集富集分析）是一种用于研究基因组学数据的分析方法。ssGSEA 通过将单个样本的基因表达数据与多个基因集进行比较，来揭示不同基因集在个体中的相对活性水平。它可以将基因表达数据映射到已知的基因集上，并计算每个基因集的富集程度，从而提供一种解释个体不同表型基础的新方法。这种方法不需要进行样本间比较，可以直接对单个样本进行基因集富集分析。

16.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(msigdbr)
library(massdatabase)
library(GSVA)
library(ComplexHeatmap)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

16.2 导入数据

- ExpressionSet 来自于章节 九。

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

16.3 所需函数

- `get_KEGG_category` 获得 KEGG pathway 的不同层级关系表;
- `get_GSVA` 基于 *ssGSEA* 方法功能富集分析;
- 使用 `GSVA::gsva`(Hänzelmann, Castelo, 和 Guinney 2013) 做 ssGSEA 分析;
- 使用 `msigdbr::msigdbr`(Dolgalev 2020) 拿到基因和通路关系对应表。
- 下载 KEGG 关系表的函数，它们均来自于 `massdatabase` 包 (Shen 等 2022)。

```
get_KEGG_category <- function(
  pathdir,
  sleeptime = 2,
  organ = "hsa") {

  KEGG_pathway_database <- massdatabase::convert_kegg2metpath(
    data = KEGG_data,
    path = pathdir)

  res <- KEGG_relation %>%
    dplyr::group_by(pathID) %>%
    dplyr::mutate(category = unlist(strsplit(pathClass, ";\\s+"))[1],
                  subcategory = unlist(strsplit(pathClass, ";\\s+"))[2]) %>%
    dplyr::ungroup() %>%
    dplyr::distinct() %>%
    dplyr::select(pathID, category, subcategory, pathName, Describtion)

  return(res)
}

get_GSVA <- function(
  exprset,
  method = c("gsva", "ssgsea"),
  showcase = 10,
  species_type = c("Mus musculus", "Homo sapiens", "zscore", "plage"),
  pathway_type = c("C2", "CP:KEGG"),
  kegg_relation = hsa_KEGG_table) {

  # download background genes
  kegg_df_list <- base::split(
```

```
kegg_df$gene_symbol,  
kegg_df$gs_name)  
  
# profile  
phenotype <- Biobase::pData(exprset) %>%  
  as.data.frame()  
profile <- Biobase::exprs(exprset) %>%  
  as.data.frame() %>%  
  as.matrix()  
  
# KEGG pathway (geneID)  
fit <- GSVA::gsva(  
  expr = profile,  
  gset.idx.list = kegg_df_list,  
  method = method,  
  kcdf = "Gaussian",  
  min.sz = 15,  
  max.sz = 500,  
  mx.diff = TRUE,  
  verbose = FALSE)  
  
fit_result <- phenotype %>%  
  dplyr::inner_join(fit %>%  
    t() %>%  
    as.data.frame() %>%  
    tibble::rownames_to_column("SampleID"),  
    by = c("SampleID"))  
  
fit_result_plot <- fit_result %>%  
  # dplyr::group_by(ProjectID, Group) %>%  
  # dplyr::slice(1:20) %>%  
  # dplyr::ungroup() %>%  
  tibble::column_to_rownames("SampleID") %>%  
  dplyr::arrange(Group, Tumour_Stage)  
  
pl_temp <- ComplexHeatmap::Heatmap(  
  mat = plotdata,  
  top_annotation = top_anno,  
  left_annotation = left_anno,  
  cluster_columns = FALSE, # TRUE
```

```

cluster_rows = TRUE,
show_row_names = TRUE,
show_column_names = FALSE,
heatmap_legend_param = list(
  legend_direction = "horizontal",
  legend_width = unit(6, "cm"),
  title = "Relative Abundance"
)

pl <- draw(pl_temp,
            heatmap_legend_side = "left",
            annotation_legend_side = "bottom")

res <- list(fit = fit,
            result = fit_result,
            pl = pl)

return(res)
}

```

16.4 运行

- KEGG 关系表

```

hsaKEGG_table <- getKEGG_category(
  pathdir = "./data/result/Function/KEGG/hsaKEGG",
  sleeptime = 2,
  organ = "hsa") %>%
  dplyr::filter(!subcategory %in%
    c("Drug resistance: antineoplastic", "Cardiovascular disease",
      "Infectious disease: parasitic", "Neurodegenerative disease",
      "Aging", "Cellular community - eukaryotes", "Infectious disease: bacterial",
      "Cancer: specific types", "Immune disease"))

```

`head(hsaKEGG_table)`

- 使用 ssGSEA 方法计算 KEGG pathway 在组间差异结果

```

GSVA_result <- getGSVA(
  exprset = ExprSet,

```

©Hua

```
method = "gsva",
showcase = 20,
species_type = "Homo sapiens",
pathway_type = c("C2", "CP:KEGG"),
kegg_relation = hsa_KEGG_table)
```

结果：从图中没有看到在 Group 组间非常明显的 KEGG 通路，但能看到 Arginine and proline metabolism 相对 Late Stage 而言，Early Stage 的颜色偏向红色也即是丰度更高一些。

16.5 输出结果

```
if (!dir.exists("./data/result/Figure")) {
  dir.create("./data/result/Figure", recursive = TRUE)
}

pdf("./data/result/Figure/SFig2.pdf", width = 12, height = 6)
print(GSVA_result$pl)
dev.off()
```

16.6 总结

在生物学和医学研究中，为了深入了解细胞或组织在特定条件下的功能状态，单样本富集分析(ssGSEA)成为了一种强有力的工具。ssGSEA 特别适用于单个样本的富集分析，通过预先定义的参考基因集合(如脂肪酸代谢、糖代谢以及免疫等相关基因集)，将基因表达谱数据与这些集合进行映射，进而计算出参考基因集在单样本中的功能表达值。这种方法为研究者提供了在单个样本水平上评估特定生物功能的可能性，对于深入理解生物学过程具有重要意义。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version

locale:
```

```
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] grid      stats4    stats     graphics  grDevices datasets  utils
[8] methods   base

other attached packages:
[1] ComplexHeatmap_2.18.0 GSVA_1.50.5          MSnbase_2.28.1
[4] ProtGenerics_1.34.0  S4Vectors_0.40.2       mzR_2.36.0
[7] Rcpp_1.0.12          Biobase_2.62.0        BiocGenerics_0.48.1
[10] massdataset_1.0.29   metid_1.2.30         metpath_1.0.8
[13] magrittr_2.0.3       masstools_1.0.13      massdatabase_1.0.10
[16] msigdbr_7.5.1       lubridate_1.9.3      forcats_1.0.0
[19] stringr_1.5.1       dplyr_1.1.4          purrr_1.0.2
[22] readr_2.1.5          tidyverse_2.0.0
[25] ggplot2_3.5.1       tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] bitops_1.0-7           cellranger_1.1.0
[3] polyclip_1.10-6        preprocessCore_1.64.0
[5] graph_1.80.0           XML_3.99-0.16.1
[7] lifecycle_1.0.4         doParallel_1.0.17
[9] globals_0.16.3          lattice_0.22-6
[11] MASS_7.3-60.0.1        openxlsx_4.2.5.2
[13] limma_3.58.1           plotly_4.10.4
[15] rmarkdown_2.26          yaml_2.3.8
[17] remotes_2.5.0          zip_2.3.1
[19] MsCoreUtils_1.14.1     pbapply_1.7-2
[21] DBI_1.2.2              RColorBrewer_1.1-3
[23] abind_1.4-5            zlibbioc_1.48.2
[25] rvest_1.0.4             GenomicRanges_1.54.1
[27] ggraph_2.2.1            RCurl_1.98-1.14
[29] tweenr_2.0.3            circlize_0.4.16
[31] GenomeInfoDbData_1.2.11 IRanges_2.36.0
[33] ggrepel_0.9.5           irlba_2.3.5.1
[35] listenv_0.9.1           annotate_1.80.0
[37] parallelly_1.37.1       DelayedMatrixStats_1.24.0
```

```
[39] ncdf4_1.22                  codetools_0.2-19
[41] DelayedArray_0.28.0          xml2_1.3.6
[43] ggforce_0.4.2                tidyselect_1.2.1
[45] shape_1.4.6.1                farver_2.1.1
[47] ScaledMatrix_1.10.0          viridis_0.6.5
[49] matrixStats_1.3.0            jsonlite_1.8.8
[51] GetoptLong_1.0.5              tidygraph_1.3.1
[53] iterators_1.0.14             foreach_1.5.2
[55] tools_4.3.3                 progress_1.2.3
[57] stringdist_0.9.12            glue_1.7.0
[59] gridExtra_2.3                SparseArray_1.2.4
[61] xfun_0.43                   MatrixGenerics_1.14.0
[63] GenomeInfoDb_1.38.8          HDF5Array_1.30.1
[65] withr_3.0.0                 BiocManager_1.30.23
[67] fastmap_1.1.1               rhdf5filters_1.14.1
[69] fansi_1.0.6                  rsvd_1.0.5
[71] digest_0.6.35                timechange_0.3.0
[73] R6_2.5.1                     colorspace_2.1-0
[75] RSQLite_2.3.6                utf8_1.2.4
[77] generics_0.1.3               renv_1.0.0
[79] data.table_1.15.4             prettyunits_1.2.0
[81] graphlayouts_1.1.1            httr_1.4.7
[83] htmlwidgets_1.6.4             S4Arrays_1.2.1
[85] pkgconfig_2.0.3               gtable_0.3.5
[87] blob_1.2.4                   impute_1.76.0
[89] SingleCellExperiment_1.24.0   XVector_0.42.0
[91] furrr_0.3.1                  htmltools_0.5.8.1
[93] MALDIquant_1.22.2            GSEABase_1.64.0
[95] clue_0.3-65                  scales_1.3.0
[97] png_0.1-8                    knitr_1.46
[99] rstudioapi_0.16.0            tzdb_0.4.0
[101] rjson_0.2.21                curl_5.2.1
[103] rhdf5_2.46.1                cachem_1.0.8
[105] GlobalOptions_0.1.2          parallel_4.3.3
[107] AnnotationDbi_1.66.0         mzID_1.40.0
[109] vsn_3.70.0                  pillar_1.9.0
[111] vctrs_0.6.5                 pcaMethods_1.94.0
[113] BiocSingular_1.18.0          beachmat_2.18.1
[115] xtable_1.8-4                cluster_2.1.6
[117] evaluate_0.23               cli_3.6.2
```

[119] compiler_4.3.3
[121] crayon_1.5.2
[123] affy_1.80.0
[125] stringi_1.8.4
[127] BiocParallel_1.36.0
[129] munsell_0.5.1
[131] lazyeval_0.2.2
[133] hms_1.1.3
[135] bit64_4.0.5
[137] Rhdf5lib_1.24.2
[139] statmod_1.5.0
[141] igraph_2.0.3
[143] affyio_1.72.0
[145] readxl_1.4.3
[119] rlang_1.1.3
[121] Rdisop_1.62.0
[123] plyr_1.8.9
[125] viridisLite_0.4.2
[127] babelgene_22.9
[129] Biostrings_2.70.3
[131] Matrix_1.6-5
[133] sparseMatrixStats_1.14.0
[135] future_1.33.2
[137] KEGGREST_1.42.0
[139] SummarizedExperiment_1.32.0
[141] memoise_2.0.1
[143] bit_4.0.5

第五部分

浸润分析

第十七章 免疫浸润细胞

免疫浸润分析在癌症研究中扮演着至关重要的角色，它有助于理解癌症微环境中免疫细胞的组成及其作用。bulk 转录组基因表达数据的反卷积技术，如 CIBERSORT 算法，是实现这一分析的重要工具。

免疫浸润分析在癌症研究中的应用场景非常广泛，主要有以下几个方面：

1. 评估肿瘤免疫微环境状态：免疫浸润分析有助于评估肿瘤免疫微环境的状态，从而了解肿瘤内免疫细胞的组成和活性，这对于指导免疫治疗策略的选择至关重要。
2. 了解病原体与宿主免疫细胞的相互作用：在感染性疾病中，免疫浸润分析有助于了解病原体与宿主免疫细胞的相互作用，为疫苗研发和免疫调节治疗提供依据。
3. 揭示异常免疫细胞的浸润和激活：在自身免疫性疾病中，免疫浸润分析有助于揭示异常免疫细胞的浸润和激活，为诊断和疾病机制研究提供线索。至于免疫浸润分析的优缺点，主要包括：

- **优点：**

1. 深入了解肿瘤免疫微环境：免疫浸润分析能够详细描绘肿瘤内部免疫细胞的种类、数量以及活性状态，从而深入理解肿瘤免疫微环境的状态。
2. 指导免疫治疗策略：通过分析肿瘤免疫微环境的状态，免疫浸润分析可以指导免疫治疗策略的选择，包括选择合适的免疫治疗药物、预测免疫治疗疗效等。

- **缺点：**

1. 数据质量限制：免疫浸润分析依赖于高通量数据，如 RNA 测序或蛋白质质谱数据，这些数据的质量会受到实验操作、测序仪器等多种因素的影响，可能导致偏差和误差。
2. 数据解读复杂：生信分析结果需要经过复杂的统计学和生物学解释，不同的数据分析方法可能得出不同的结果，需要经过仔细验证和进一步研究才能确定其生物学意义。
3. 受益患者比例较低：虽然免疫治疗在理论上具有很大的优势，但目前真正能从免疫治疗中获益的癌症患者比例相对较低。这可能与肿瘤免疫微环境的复杂性、免疫治疗的局限性等因素有关。

免疫浸润分析的算法

表 17.1: 免疫浸润算法比较

method	organism	license	citation
quanTIseq	human	free (BSD)	https://doi.org/10.1186/s13073-019-0638-6
TIMER	human	free (GPL 2.0)	https://doi.org/10.1186/s13059-016-1028-7
CIBERSORT	human	free for non-commercial use only	https://doi.org/10.1038/nmeth.3337
MCPCounter	human	free (GPL 3.0)	https://doi.org/10.1186/s13059-016-1070-5
xCell	human	free (GPL 3.0)	https://doi.org/10.1186/s13059-017-1349-1
EPIC	human	free for non-commercial use only (Academic License)	https://doi.org/10.7554/eLife.26476
ESTIMATE	human	free (GPL 2.0)	https://doi.org/10.1038/ncomms3612
ABIS	human	free (GPL 2.0)	https://doi.org/10.1016/j.celrep.2019.01.041
ConsensusTME	human	free (GPL 3.0)	https://doi.org/10.1158/0008-5472.CAN-18-3560
mMCPCounter	mouse	free (GPL 3.0)	https://doi.org/10.1186/s13073-020-00783-w
seqImmuCC	mouse	free for non-commercial use only	https://doi.org/10.3389/fimmu.2018.01286
DCQ	mouse	free (GPL 2.0)	https://doi.org/10.1002/msb.134947
BASE	mouse	free	https://doi.org/10.1038/ncomms10248

method	organism	license	citation
ImmuneCellAI	human	free	https://doi.org/10.1101/advances.2019.02.880

17.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(Bioconductor)
library(immunedeconv)
library(ImmuneCellAI)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

17.2 导入数据

- `ExpressionSet` 来自于章节 九。

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

17.3 所需函数

- `get_ImmuneCell` 获得免疫细胞的相对丰度表；
- 使用 `immunedeconv::deconvolute(Sturm, Finotello, 和 List 2020)` 反卷积分析。

```
get_ImmuneCell <- function(
  exprset,
  method = c("mcp_counter", "epic", "quantiseq",
```

```

    "xcell", "cibersort", "cibersort_abs",
    "timer", "consensus_tme", "abis",
    "estimate", "ImmuCellAI")) {

profile <- Biobase::exprs(exprset)
metadata <- Biobase::pData(exprset)

if (any(method == "cibersort", method == "cibersort_abs")) {
  immunedeconv::set_cibersort_binary("./data/result/ImmuneCell/CIBERSORT.R")
  immunedeconv::set_cibersort_mat("./data/result/ImmuneCell/LM22.txt")
}

colnames(fit)[1] <- "CellType"

print(fit$CellType)

res <- metadata %>%
  dplyr::inner_join(fit %>%
    tibble::column_to_rownames("CellType") %>%
    t() %>% as.data.frame() %>%
    tibble::rownames_to_column("SampleID"),
    by = "SampleID")

return(res)
}

```

17.4 运行 ImmuCellAI

ImmuCellAI (*Immune Cell Abundance Identifier*) (Miao 等 2020) 是一个免疫浸润计算工具，用于从基因表达数据集（包括 RNA-Seq 和芯片数据）中估计 24 种免疫细胞的丰度。这 24 种免疫细胞由 18 种 T 细胞亚型和 6 种其他免疫细胞组成，包括 B 细胞、NK 细胞、单核细胞、巨噬细胞、中性粒细胞和 DC 细胞。

ImmuCellAI 可用于估计不同人群免疫细胞浸润的差异，以及预测患者对免疫检查点阻断疗法的反应。此外，*ImmuCellAI* 还可用于建立免疫治疗反应预测模型，并且已经过测试，发现其模型精度在 0.80 到 0.91 之间。这表明 *ImmuCellAI* 在肿瘤免疫浸润评估和免疫治疗反应预测中具有强大而独特的功能。

在使用 *ImmuCellAI* 时，用户需要准备全编码蛋白的表达谱基因数据，并选择数据类型和是否预测免疫治疗响应。运行后，用户可以直接下载结果。此外，*ImmuCellAI* 支持 RNA-seq 和芯片两种数据的上传，并且可以通过每一个免疫细胞特征基因的表达来进行免疫浸润评分。

注意：直接使用 `remotes::install_github("lydiaMyr/ImmuneCellAI@main")` 安装是有问题的，建议下载仓库提供的 `ImmuneCellAI_0.1.0.tar.gz` 本地安装

```
Icell_ImmuneCellAI <- get_ImmuneCell(  
  exprset = ExprSet,  
  method = "ImmuneCellAI")  
  
head(Icell_ImmuneCellAI)  
  
Icell_ImmuneCellAI <- get_ImmuneCell(  
  exprset = ExprSet,  
  method = "ImmuneCellAI")  
  
head(Icell_ImmuneCellAI)
```

结果：获得 24 种免疫细胞的丰度表达谱。

17.5 其他免疫浸润方法

- *CIBERSORT* 算法的原理基于一个核心假设：在复杂组织中，细胞类型特异的基因表达模式可以被用来区分不同的细胞类型。具体而言，CIBERSORT 通过比较待测样本的基因表达模式与已知细胞类型的基因表达模式，来推断待测样本中各种细胞类型的相对比例。
- *quantiTseq* 免疫浸润方法的原理主要基于反卷积算法。这种方法利用 bulk samples 的 RNA-seq 数据（即包含多种细胞类型的混合样本的基因表达数据），通过计算每个免疫细胞类型的特异性基因表达特征，来预测肿瘤样本中不同种类免疫细胞的相对丰度或组成。
- *ESTIMATE* 免疫浸润方法是一种基于基因表达数据预测恶性肿瘤组织间质细胞和免疫细胞含量的方法。其原理主要基于单个样本基因集合的富集分析 (ssGSEA)。在 ESTIMATE 方法中，基因集合的富集分析是关键步骤。它基于预先定义的基因集合（如与基质或免疫细胞相关的基因集合），通过计算样本中这些基因的表达水平，来评估特定细胞类型在样本中的富集程度。

实现的 R 代码

```
# cibersort  
Icell_cibersort <- get_ImmuneCell(  
  exprset = ExprSet,  
  method = "cibersort")  
  
# quantiseq  
Icell_quantiseq <- get_ImmuneCell(  
  exprset = ExprSet,
```

```

method = "quantiseq")

# estimate
Icell_estimate <- get_ImmuneCell(
  exprset = ExprSet,
  method = "estimate")

```

17.6 输出结果

```

if (!dir.exists("./data/result/ImmuneCell/")) {
  dir.create("./data/result/ImmuneCell/", recursive = TRUE)
}

write.csv(Icell_ImmuneCellAI, "./data/result/ImmuneCell/HCC_ImmuneCell_ImmuneCellAI.csv", row.names = F)

```

17.7 总结

免疫浸润分析在肿瘤学研究中占据重要位置，其目的在于探究肿瘤微环境中不同免疫细胞类型的分布和状态，以及它们与肿瘤细胞之间的相互作用。这种分析有助于更深入地理解肿瘤免疫微环境的复杂性，并为肿瘤免疫治疗提供重要的参考信息。

为了实现免疫浸润分析，科学家们开发了多种算法，这些算法基于不同的原理来预测肿瘤样本中各种免疫细胞的相对丰度。其中，反卷积算法是一种重要的方法，它通过利用 bulk samples 的 RNA-seq 数据，结合已知的免疫细胞特异性基因表达特征，来估计不同免疫细胞类型在肿瘤样本中的含量。

在本研究中，采用了 *ImmuneCellAI*(Miao 等 2020) 算法对转录组数据进行了反卷积分析。通过运行该算法，成功得到了每个肿瘤样本中 24 种免疫细胞的相对丰度结果。这些结果为进一步探究肿瘤免疫微环境的复杂性、理解免疫细胞与肿瘤细胞之间的相互作用提供了宝贵的数据支持。

系统信息

```

sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version

```

```
locale:  
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8  
  
time zone: Asia/Shanghai  
tzcode source: internal  
  
attached base packages:  
[1] stats      graphics   grDevices datasets  utils      methods    base  
  
other attached packages:  
[1] ImmuCellAI_0.1.0     quadprog_1.5-8       pracma_2.4.4  
[4] e1071_1.7-14        gridExtra_2.3       GSVA_1.50.5  
[7] immunoedconv_2.1.0   EPIC_1.1.7        Biobase_2.62.0  
[10] BiocGenerics_0.48.1 lubridate_1.9.3    forcats_1.0.0  
[13] stringr_1.5.1       dplyr_1.1.4        purrr_1.0.2  
[16] readr_2.1.5         tidyverse_2.0.0     tibble_3.2.1  
[19] ggplot2_3.5.1       tidyverse_2.0.0  
  
loaded via a namespace (and not attached):  
[1] rstudioapi_0.16.0          jsonlite_1.8.8  
[3] magrittr_2.0.3             rmarkdown_2.26  
[5] zlibbioc_1.48.2           vctrs_0.6.5  
[7] memoise_2.0.1             DelayedMatrixStats_1.24.0  
[9] RCurl_1.98-1.14          htmltools_0.5.8.1  
[11] S4Arrays_1.2.1           progress_1.2.3  
[13] curl_5.2.1               cellranger_1.1.0  
[15] Rhdf5lib_1.24.2          SparseArray_1.2.4  
[17] rhdf5_2.46.1             cachem_1.0.8  
[19] lifecycle_1.0.4           ComICS_1.0.4  
[21] pkgconfig_2.0.3           rsvd_1.0.5  
[23] Matrix_1.6-5             R6_2.5.1  
[25] fastmap_1.1.1            GenomeInfoDbData_1.2.11  
[27] MatrixGenerics_1.14.0    digest_0.6.35  
[29] colorspace_2.1-0          AnnotationDbi_1.66.0  
[31] S4Vectors_0.40.2          irlba_2.3.5.1  
[33] GenomicRanges_1.54.1     RSQLite_2.3.6  
[35] beachmat_2.18.1          filelock_1.0.3  
[37] mMCPcounter_1.1.0        testit_0.13  
[39] fansi_1.0.6              timechange_0.3.0
```

©Hua

```
[41] httr_1.4.7                  abind_1.4-5
[43] mgcv_1.9-1                 compiler_4.3.3
[45] proxy_0.4-27                bit64_4.0.5
[47] withr_3.0.0                 BiocParallel_1.36.0
[49] DBI_1.2.2                   HDF5Array_1.30.1
[51] biomaRt_2.58.2              MASS_7.3-60.0.1
[53] rappdirs_0.3.3              DelayedArray_0.28.0
[55] tools_4.3.3                 glue_1.7.0
[57] nlme_3.1-164                rhdf5filters_1.14.1
[59] grid_4.3.3                  generics_0.1.3
[61] sva_3.50.0                  gtable_0.3.5
[63] tzdb_0.4.0                  class_7.3-22
[65] preprocessCore_1.64.0       hms_1.1.3
[67] BiocSingular_1.18.0         ScaledMatrix_1.10.0
[69] xml2_1.3.6                  utf8_1.2.4
[71] XVector_0.42.0              pillar_1.9.0
[73] limma_3.58.1                genefilter_1.84.0
[75] splines_4.3.3               BiocFileCache_2.10.2
[77] lattice_0.22-6              renv_1.0.0
[79] survival_3.7-0              bit_4.0.5
[81] annotate_1.80.0              tidyselect_1.2.1
[83] SingleCellExperiment_1.24.0  locfit_1.5-9.9
[85] Biostrings_2.70.3            knitr_1.46
[87] IRanges_2.36.0              edgeR_4.0.16
[89] SummarizedExperiment_1.32.0  stats4_4.3.3
[91] xfun_0.43                   statmod_1.5.0
[93] matrixStats_1.3.0            stringi_1.8.4
[95] yaml_2.3.8                  evaluate_0.23
[97] codetools_0.2-19             data.tree_1.1.0
[99] BiocManager_1.30.23          graph_1.80.0
[101] cli_3.6.2                  xtable_1.8-4
[103] munsell_0.5.1              Rcpp_1.0.12
[105] GenomeInfoDb_1.38.8         readxl_1.4.3
[107] dbplyr_2.5.0                png_0.1-8
[109] XML_3.99-0.16.1             parallel_4.3.3
[111] blob_1.2.4                  prettyunits_1.2.0
[113] sparseMatrixStats_1.14.0    bitops_1.0-7
[115] GSEABase_1.64.0             scales_1.3.0
[117] crayon_1.5.2                rlang_1.1.3
[119] KEGGREST_1.42.0
```

第十八章 免疫浸润分析

在本章节中，将详细探讨免疫细胞的组成结构、其在不同个体和分组之间的相对丰度差异，并通过热图等可视化手段，对这些差异进行直观而深入的解析。这些分析将有助于科研人员更好地理解免疫细胞在生物体内的分布规律及其在不同条件下的变化特征。

18.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(ggpubr)
library(ggplotify)
library(corrplot)
library(pheatmap)
library(RColorBrewer)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

18.2 导入数据

- ImmueCell 来自于章节 [十七](#)。

```
ImmueCell <- read.csv("./data/result/ImmuneCell/HCC_ImmuneCell_ImmucellAI.csv")
```

18.3 所需函数

- `get_plot` 获得数据分析可视化图；
- 通过 `plotType` 参数选择不同的图形。
 - `boxplot` (箱线图)：一种用于显示一组数据分散情况资料的统计图。因形状如箱子而得名，又称为箱型图、盒须图或盒式图。箱线图通过绘制数据的中位数 (Q2)、四分位数 (Q1 和 Q3)、异常值 (如有) 等信息，可以直观地展示数据的分布特征。
 - `violin` (小提琴图)：小提琴图 (Violin Plot) 是一种用于显示数据分布及其概率密度的图表。这种图表结合了箱形图和密度图的特征，主要用来显示数据的分布形状。在小提琴图中，中间的黑色粗线条表示四分位数的范围，从其上延伸出来的细黑线表示数据的范围，两端为最大值和最小值，而白色点则为中位数。与箱形图相比，小提琴图除了能显示上述的统计数据外，还能更直观地展示数据的整体分布，尤其是在数据量非常大时，小提琴图特别适用。
 - `stackedbar` (堆积图)：堆积图 (Stacked Chart) 或堆积条形图 (Stacked Bar Chart)、堆积柱状图 (Stacked Column Chart) 以及堆积面积图 (Stacked Area Chart) 等是数据可视化中常用的图表类型，它们通过堆叠不同类别的数据来展示数据之间的总量和构成关系。
 - `heatmap` (热图)：热图是一种数据可视化工具，通过颜色的深浅来表示数据的值或密度。它可以直观地展示数据的分布和变化情况，帮助人们更好地理解数据。
 - `corrplot` (相关性矩阵图)：相关性矩阵图 (Correlation Matrix Plot) 是一种用于展示数据集中不同变量之间相关性的可视化工具。它通过矩阵的形式，将每对变量之间的相关性系数以数值或颜色的方式表示出来，从而可以直观地看出哪些变量之间存在较强的相关性，以及这些相关性的正负方向。

```
get_plot <- function(
  object,
  plabel = c("p.signif", "p.format"),
  plotType = c("boxplot", "violin",
              "stackedbar", "heatmap", "corrplot"),
  group,
  group_names = grp_names,
  group_colors = grp_colors,
  plotData = TRUE) {

  colnames(object)[which(colnames(object) == group)] <- "GroupCompvar"

  object <- object %>%
    dplyr::arrange(GroupCompvar, Tumour_Stage)
  object$GroupCompvar <- factor(object$GroupCompvar, levels = group_names)
```

```

phenotype <- object %>%
  dplyr::select(1:8)

profile <- object %>%
  dplyr::select(-c(2:8)) %>%
  tibble::column_to_rownames("SampleID") %>%
  dplyr::select(-all_of(c("Effector_memory", "Gamma_delta",
    "Central_memory")))
}

dat <- profile[, colSums(profile) > 0]

if (!all(rownames(dat) == phenotype$SampleID)) {
  message("wrong order between dat and phenotype")
} else {
  dat <- dat[pmatch(rownames(dat), phenotype$SampleID), ]
}

lvls <- unique(phenotype$GroupCompvar)
dat$GroupCompvar <- phenotype$GroupCompvar
dat$GroupCompvar <- factor(dat$GroupCompvar, levels = lvls)
dat$SampleID <- rownames(dat)

if (plotType %in% c("boxplot", "violin", "stackedbar")) {
  plotdata <- dat %>%
    tidyr::gather(key = "CellType",
                  value = "Composition",
                  -c("SampleID", "GroupCompvar"))

  plot_order <- plotdata[plotdata$GroupCompvar == lvls[1], ] %>%
    dplyr::group_by(CellType) %>%
    dplyr::summarise(md = median(Composition)) %>%
    dplyr::arrange(desc(md)) %>%
    dplyr::pull(CellType)

  plotdata$CellType <- factor(plotdata$CellType,
                                levels = plot_order)

  if (plotType == "boxplot") {

    pl <- ggplot(plotdata, aes(x = CellType, y = Composition)) +

```

```

stat_boxplot(aes(color = GroupCompvar), position = position_dodge(0.5),
             geom = "errorbar", width = 0.2) +
geom_boxplot(aes(fill = GroupCompvar),
             position = position_dodge(0.5),
             width = 0.5,
             outlier.alpha = 0) +
stat_compare_means(aes(group = GroupCompvar),
                   label = plabel,
                   method = "wilcox.test",
                   hide.ns = F) +
scale_fill_manual(values = group_colors) +
scale_color_manual(values = group_colors) +
scale_y_continuous(expand = expansion(mult = c(0.05, 0.1))) +
labs(y = "Ratio", x = NULL) +
theme_bw() +
theme(plot.title = element_text(size = 12, color = "black", hjust = 0.5, face = "bold"),
      axis.title = element_text(size = 11, color = "black", face = "bold"),
      axis.text = element_text(size= 10, color = "black"),
      panel.grid.minor.y = element_blank(),
      panel.grid.minor.x = element_blank(),
      axis.text.x = element_text(angle = 45, hjust = 1 ),
      panel.grid = element_blank(),
      legend.position = "top",
      legend.text = element_text(size = 8),
      legend.title= element_text(size = 10),
      text = element_text(family = "serif"))
} else if (plotType == "stackedbar") {

plotdata$NewSampleID <- paste(plotdata$GroupCompvar, plotdata$SampleID, sep = "_")
mycolors <- colorRampPalette(brewer.pal(8, "Set1"))(length(unique(plotdata$CellType)))

pl <- ggplot(plotdata, aes(x = NewSampleID, y = Composition, fill = CellType)) +
geom_bar(stat = "identity", position = "fill") +
labs(y = "Immune Cells Relative Percentage", x = "") +
scale_y_continuous(expand = c(0, 0),
                  labels = scales::percent) +
scale_fill_manual(values = mycolors) +
guides(fill = guide_legend(title = NULL, ncol = 2)) +
annotate("text", x = as.numeric(table(object$GroupCompvar))[1]/2,
        y = -0.03, label = group_names[1],

```

```

      size = 3, color = "black") +
  annotate("segment", x = 1, xend = as.numeric(table(object$GroupCompvar))[1],
           y = -0.01, yend = -0.01,
           color = group_colors[1], cex = 5) +
  annotate("text", x = (dim(object)[1] - as.numeric(table(object$GroupCompvar))[1])/2 +
               as.numeric(table(object$GroupCompvar))[1],
           y = -0.03, label = group_names[2],
           size = 3, color = "black") +
  annotate("segment", x = as.numeric(table(object$GroupCompvar))[1] + 1,
           xend = dim(object)[1],
           y = -0.01, yend = -0.01,
           color = group_colors[2], cex = 5) +
  theme_classic() +
  theme(axis.title = element_text(size = 11, color = "black", face = "bold"),
        axis.text = element_text(size = 10, color = "black"),
        axis.text.x.bottom = element_blank(),
        axis.line.x.bottom = element_blank(),
        axis.ticks.x.bottom = element_blank(),
        panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank(),
        panel.grid = element_blank(),
        legend.position = "right",
        legend.key.size = unit(0.45, "cm"),
        legend.text = element_text(size = 8),
        text = element_text(family = "serif"))

} else if (plotType == "violin") {

  pl <- ggplot(plotdata, aes(x = CellType, y = Composition)) +
    geom_violin(aes(fill = GroupCompvar), trim = FALSE,
                position = position_dodge(0.5),
                outlier.color = "transparent") +
    stat_compare_means(aes(group = GroupCompvar),
                      label = plabel,
                      method = "wilcox.test",
                      hide.ns = F) +
    scale_fill_manual(values = group_colors) +
    labs(y = "Ratio", x = NULL) +
    theme_bw() +
    theme(plot.title = element_text(size = 12, color = "black", hjust = 0.5, face = "bold"),
          plot.subtitle = element_text(size = 10, color = "black", hjust = 0.5, face = "bold"))
}

```

```

        axis.title = element_text(size = 11, color = "black", face = "bold"),
        axis.text = element_text(size= 10, color = "black"),
        panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1 ),
        panel.grid=element_blank(),
        legend.position = "top",
        legend.text = element_text(size = 8),
        legend.title= element_text(size = 10),
        text = element_text(family = "serif"))

    }

} else if (plotType == "heatmap") {

  plotdata <- dat %>%
    dplyr::arrange(GroupCompvar) %>%
    dplyr::select(-all_of(c("GroupCompvar", "SampleID"))) %>%
    t()

  annotation_col <- data.frame(
    GroupCompvar = dat$GroupCompvar,
    row.names = dat$SampleID)

  temp <- pheatmap::pheatmap(
    plotdata,
    cluster_cols = F,
    # cutree_cols = 2,
    angle_col = "90",
    annotation_col = annotation_col,
    show_colnames = F,
    fontfamily = "serif")

  pl <- as.ggplot(temp)

} else if (plotType == "corrplot") {

  plotdata <- dat %>%
    dplyr::select(-all_of(c("GroupCompvar", "SampleID"))) %>%
    as.matrix()
  fit <- Hmisc::rcorr(plotdata, type = "spearman")

  dataR <- signif(fit$r, 2)
}

```

```
# grDevices::windowsFonts()
par(family = "serif")
pl <- corrplot::corrplot(
    dataR,
    method = "color",
    col = colorRampPalette(c("blue", "white", "red"))(100),
    order = "AOE",
    addCoef.col = "grey50",
    number.cex = .7,
    number.font = 1.5,
    tl.cex = 0.8,
    tl.srt = 45,
    tl.col = "black")

}

if (plotType == "corrplot") {
    datfit <- fit
} else {
    datfit <- NULL
}

if (plotData) {
    res <- list(plot = pl,
                plotdata = plotdata,
                fit = datfit)
} else {
    res <- pl
}

return(res)
}
```

18.4 堆积图

堆积图的意义：

1. 总量和构成：堆积图最直观的功能就是显示不同类别的数据如何组成总体。每个类别的数据块（或柱形、面积）堆叠在一起，形成总体数据的高度或面积。

2. 各部分比例：通过堆叠的高度或面积，可以很容易地看出每个类别在总体中所占的比例。
3. 类别间的对比：在堆积图中，可以方便地对比不同类别的数据大小，以及它们随时间的变化情况。
4. 数据层次：堆积图可以显示数据的层次结构，特别是在数据有多个分类维度时。例如，在一个销售数据中，可以同时展示不同产品类别、不同销售渠道的销售额。
5. 异常值检测：如果某个类别的数据块突然增大或减小，可能意味着该类别出现了异常或重大变化。

```
ImmueCell_42_stackedbar <- get_plot(
  object = ImmueCell,
  plotType = "stackedbar",
  group = "Group",
  plotData = TRUE)
```

```
ImmueCell_42_stackedbar$plot
```

结果：从图中可以看到不同癌症分期的免疫细胞在样本间存在个体特异性。

18.5 箱线图

两组的箱线图的意义：

1. 数据分布的差异：通过比较两组数据的箱子高度、位置和中位数，可以判断两组数据的分散程度、中心位置和对称性是否有显著差异。
2. 异常值的比较：箱线图可以清晰地展示两组数据中异常值的数量和分布情况，从而了解两组数据在极端值方面的差异。
3. 统计检验的基础：箱线图可以为进一步的统计检验（如 Mann-Whitney U 检验）提供直观依据，帮助研究者判断两组数据是否具有统计差异。

```
ImmueCell_boxplot <- get_plot(
  object = ImmueCell,
  plabel = "p.signif",
  group = "Group",
  plotType = "boxplot",
  plotData = TRUE)
```

```
ImmueCell_boxplot$plot
```

结果：从图中可以看到某些免疫细胞如 Tfh, Th2 等在 Early 分期显著富集，这说明不同分期下的 HCC 有着不一样的肿瘤微环境。

18.6 热图

热图的意义：

1. 数据分布：热图通过颜色的变化来展示数据在不同区域的分布情况。颜色越深，表示该区域的数据值越大或密度越高；颜色越浅，则表示数据值越小或密度越低。这有助于人们快速识别数据中的热点和冷点。
2. 趋势分析：通过观察热图中颜色的渐变，可以分析数据的趋势。
3. 异常检测：热图中的异常值通常会以与其他区域明显不同的颜色呈现出来，从而便于人们快速发现数据中的异常情况。
4. 比较分析：通过对不同条件下的热图，可以对不同数据集进行比较分析。例如，在生物实验中，可以对比不同处理组之间的基因表达热图，以找出处理效果对基因表达的影响。

```
ImmueCell_heatmap <- get_plot(
  object = ImmueCell,
  plotType = "heatmap",
  group = "Group",
  plotData = TRUE)
```

结果：从图中可以看到不同癌症分期的免疫细胞在样本间存在个体特异性。

18.7 相关性矩阵图

相关性矩阵图意义：

1. 相关性分析：相关图可以直观地展示数据集中变量之间的相关性，从而帮助研究人员快速了解哪些变量之间可能存在关系，以及关系的强度和方向。
2. 模式识别：通过相关图，研究人员可以识别数据中的隐藏模式，例如哪些变量组合在一起时相关性较强，或者哪些变量与多个其他变量都有较强的相关性。
3. 假设检验：在统计假设检验中，相关图可以作为一个初步的工具来检查数据是否符合预期的相关性模式。
4. 数据探索：在数据探索阶段，相关图可以帮助研究人员了解数据的整体结构和变量之间的关系，为后续的分析和建模提供基础。

```
ImmueCell_corrplot <- get_plot(
  object = ImmueCell,
  plotType = "corrplot",
  group = "Group",
  plotData = TRUE)
```

结果：从图中可以看到不同免疫细胞存在强相关。

18.8 输出结果

```
if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

ggsave("./data/result/Figure/Fig4-A.pdf", ImmueCell_42_stackedbar$plot, width = 10, height = 6, dpi = 600)
ggsave("./data/result/Figure/Fig4-B.pdf", ImmueCell_boxplot$plot, width = 10, height = 6, dpi = 600)
ggsave("./data/result/Figure/Fig4-C.pdf", ImmueCell_heatmap$plot, width = 8, height = 6, dpi = 600)

pdf("./data/result/Figure/Fig4-D.pdf", width = 8, height = 6)
temp_list <- get_plot(
  object = ImmueCell,
  plotType = "corrplot",
  group = "Group",
  plotData = TRUE)
dev.off()
```

18.9 总结

在本章节中，通过对比免疫细胞在不同癌症分期的表现，深入探究了不同癌症分期人群的肿瘤微环境差异。为了清晰展示这些差异，采用了多种可视化工具，包括堆积图、箱线图、热图以及相关矩阵图等，对收集到的数据进行了详细的分析和呈现。经过细致的研究，最终发现，某些特定的免疫细胞类型在不同癌症分期组中表现出了显著的差异，这些差异为理解癌症发展与免疫细胞活性之间的关系提供了重要线索。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version
```

©Hua

```
locale:  
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8  
  
time zone: Asia/Shanghai  
tzcode source: internal  
  
attached base packages:  
[1] stats      graphics   grDevices datasets  utils      methods    base  
  
other attached packages:  
[1] RColorBrewer_1.1-3  pheatmap_1.0.12   corrplot_0.92    ggplotify_0.1.2  
[5] ggpubr_0.6.0       lubridate_1.9.3   forcats_1.0.0   stringr_1.5.1  
[9] dplyr_1.1.4        purrrr_1.0.2     readr_2.1.5     tidyverse_2.0.0  
[13] tibble_3.2.1       ggplot2_3.5.1    tidyverse_2.0.0  
  
loaded via a namespace (and not attached):  
[1] yulab.utils_0.1.4    utf8_1.2.4        generics_0.1.3  
[4] renv_1.0.0          rstatix_0.7.2    stringi_1.8.4  
[7] hms_1.1.3           digest_0.6.35    magrittr_2.0.3  
[10] evaluate_0.23      grid_4.3.3       timechange_0.3.0  
[13] fastmap_1.1.1      jsonlite_1.8.8   backports_1.4.1  
[16] BiocManager_1.30.23 fansi_1.0.6     scales_1.3.0  
[19] abind_1.4-5         cli_3.6.2       rlang_1.1.3  
[22] munsell_0.5.1      cachem_1.0.8    withr_3.0.0  
[25] yaml_2.3.8          tools_4.3.3     tzdb_0.4.0  
[28] memoise_2.0.1      ggsignif_0.6.4   colorspace_2.1-0  
[31] broom_1.0.5         gridGraphics_0.5-1 vctrs_0.6.5  
[34] R6_2.5.1            lifecycle_1.0.4   fs_1.6.4  
[37] car_3.1-2          pkgconfig_2.0.3   pillar_1.9.0  
[40] gtable_0.3.5        glue_1.7.0       xfun_0.43  
[43] tidyselect_1.2.1    rstudioapi_0.16.0 knitr_1.46  
[46] htmltools_0.5.8.1   rmarkdown_2.26    carData_3.0-5  
[49] compiler_4.3.3
```

第六部分

标记筛选

第十九章 LASSO LR

采用了 LASSO (Least Absolute Shrinkage and Selection Operator) 结合 LR (Logistic Regression: 适合本教程二分类分组数据) 的方法，对差异基因（参考章节 十一）进行了特征筛选。通过这种方法，能够从大量的候选基因中识别出与特定生物过程或疾病状态最为相关的关键基因。筛选出的这些重要基因不仅具有统计学上的显著性，而且在实际应用中具有较高的预测能力。这些经过 LASSO+LR 筛选的重要基因将被用于后续的生物信息学分析、疾病机制探究以及潜在治疗靶点的识别等研究中，以期进一步推动相关领域的科学进展。

19.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(Biobase)
library(glmnet)
library(caret)
library(ROCR)
library(pROC)
library(MLmetrics)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

19.2 导入数据

- 差异结果来自于章节 十一；

- ExpressionSet 来自于章节 九。

```
da_res <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma_select.csv")
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

19.3 准备数据

- 差异基因：结果解析见表 11.1

```
signif_DEG <- da_res %>%
  dplyr::filter(Enrichment != "Nonsignif") %>%
  dplyr::slice(-grep("\\.", FeatureID))
```

```
head(signif_DEG[, 1:6])
```

- 基因表达谱：行名是样本，列名是基因 ID

```
profile <- exprs(ExprSet) %>%
  as.data.frame()
rownames(profile) <- make.names(rownames(profile))
```

```
head(profile_DEG[, 1:6])
```

- 临床表型表：包含分组等信息

```
metadata <- pData(ExprSet)
head(metadata[, 1:6])
```

19.4 机器学习特征筛选

特征筛选步骤：

1. 数据分割：

将原始数据集划分为训练集、验证集和测试集。通常，训练集用于模型训练，验证集用于调整超参数和选择最佳模型，测试集用于评估最终模型的性能。划分比例可以根据数据集的大小和特性进行调整，但一般常见的比例是训练集占 60%-80%，验证集和测试集各占 10%-20%。

2. 数据转换：

数据清洗：处理缺失值、异常值、重复值等。对于缺失值，可以使用均值、中位数、众数等填充，或采用插值法、机器学习预测等方法进行填充。

特征编码：对于分类变量，需要进行编码以便模型能够处理。常见的编码方式包括独热编码（One-Hot Encoding）、标签编码（Label Encoding）等。

特征离散化：将连续特征转换为离散类别，例如通过分箱操作。这有助于处理一些具有非线性关系的特征。

特征标准化或归一化：对于不同量纲或不同分布的特征，需要进行标准化或归一化，以便模型能够更好地处理它们。

3. 特征筛选：

相关性分析：计算每个特征与目标变量之间的相关性，如皮尔逊相关系数或斯皮尔曼等级相关系数。通过相关性分析，可以初步筛选出与目标变量强相关的特征。

树模型重要性：利用决策树、随机森林等树模型评估特征的重要性。这些模型可以根据特征在树中分裂的贡献度来评估特征的重要性。

启发式搜索策略：如前向序列选择方法，从空的候选集合出发，逐步添加与目标变量相关性最强的特征。

全局最优搜索策略：如穷举法或分支界定法，从所有可能的特征组合中挑选出表现最优的特征子集。

4. 训练集构建模型：使用经过特征筛选的训练集数据构建机器学习模型。选择合适的机器学习算法，如逻辑回归、支持向量机、决策树、随机森林、神经网络等。
5. 调参：使用验证集对模型进行调参，优化模型的超参数。超参数是机器学习算法中需要人为设定的参数，如学习率、迭代次数、正则化参数等。可以采用网格搜索（Grid Search）、随机搜索（Random Search）等方法进行调参。
6. 测试集评估模型：使用测试集对经过调参的模型进行评估，计算模型的性能指标，如准确率、精确率、召回率、F1 分数、AUC-ROC 等。根据评估结果选择性能最佳的模型作为最终模型。

19.4.1 数据分割

在机器学习的实践中，数据分割是一个至关重要的步骤，用于将原始数据集划分为训练集和测试集（使用了 `caret::createDataPartition` ([Kuhn 2008](#))）。

```
set.seed(123)

trainIndex <- caret::createDataPartition(
  metadata$Group,
  p = 0.8,
  list = FALSE,
  times = 1)
```

©Hua

```

y_train <- metadata[trainIndex, ]
X_train <- profile_DEG[rownames(profile_DEG) %in% rownames(y_train), ] %>%
  as.matrix()

y_test <- metadata[-trainIndex, ]
X_test <- profile_DEG[!rownames(profile_DEG) %in% rownames(y_train), ] %>%
  as.matrix()

```

19.4.2 调参

在构建机器学习模型时，特别是在使用 Lasso+LR 时，选择合适的正则化参数（在 Lasso 中通常称为 λ ）是非常重要的。正则化参数用于控制模型的复杂度，以避免过拟合。交叉验证（Cross-Validation, CV）是一种常用的技术，用于评估模型在不同参数设置下的性能，并选取最优的参数。

在 `cv.glmnet` 函数中（来自于 `glmnet`(Friedman, Hastie, 和 Tibshirani 2010)），`family` 参数用于指定模型的类型。当 `family = "binomial"` 时，你正在指定模型应该使用逻辑回归（logistic regression: LR）来拟合二元（binary）响应变量（即，响应变量只取两个可能的值，如 0 和 1）。

```

set.seed(123)

cv.fit <- cv.glmnet(
  x = X_train,
  y = y_train$Group,
  family = "binomial")

plot(cv.fit)

```

19.4.3 测试集验证

首先，将训练好的模型应用于测试集数据，以预测每个样本的分类标签。预测结果将与测试集的真实标签进行对比，以计算模型在各个类别上的分类准确性。

- 预测测试集的标签

```

test_group_label <- predict(cv.fit, X_test) %>%
  as.data.frame() %>%
  setNames("Predicted") %>%
  cbind(data.frame(Actual = y_test$Group)) %>%
  head()

```

```

head(test_group_label)

• assess.glmnet 函数给出整体评估指标

assess.glmnet(cv.fit, newx = X_test, newy = y_test$Group, s = "lambda.min")

• 混淆矩阵 (Confusion matrix)

cnf <- confusion.glmnet(
  object = cv.fit,
  newx = X_test)

print(cnf)

• ROC 曲线

rocs <- roc.glmnet(
  object = cv.fit$fit$prevval,
  newy = y_train$Group)

best <- cv.fit$index["min", ]
plot(rocs[[best]], type = "l")
invisible(sapply(rocs, lines, col="grey"))
lines(rocs[[best]], lwd = 2, col = "red")
lines(c(0, 1), c(0, 1), col = "black", lty = 9, lwd = 2)

```

19.4.4 最终分类模型

```

lasso_model <- glmnet(
  X_train,
  y_train$Group,
  family = "binomial",
  alpha = 1,
  lambda = cv.fit$lambda.min)

ggplot(roc_df, aes(x = FPR, y = TPR)) +
  geom_line(color = "blue") +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +

```

```

geom_text(aes(label = paste("Youden Index =", round(youden_index, 2))),
          x = 0.2, y = 0.8, color = "black", size = 4) +
geom_text(aes(label = paste("Specificity =", round(roc_data$specificity[which.max(roc_data$sensitivity)],
          x = 0.2, y = 0.7, color = "black", size = 4) +
labs(x = "False Positive Rate", y = "True Positive Rate",
     title = "ROC Curve for LASSO Regression") +
theme_bw()

```

- 混淆矩阵

```
print(caret::confusionMatrix(as.factor(pred_raw[, 1]), factor(y_test$Group)))
```

结果: **特异度 (Specificity)** 在上述所有模型评估指标中是偏低的 ($Specificity : 0.2857$), 结合先前机器学习基础小节 4.3, 可以推测得知该模型对阴性样本也即是 Late Stage 的患者预测效果不佳。

- AUROC 曲线: 使用 `pROC::roc(Robin 等 2011)` 函数

```

AUROC <- function(
  DataTest,
  PredProb = pred_prob,
  nfeature) {

  # DataTest = y_test
  # PredProb = pred_prob
  # nfeature = 41

  # ROC object
  rocobj <- roc(DataTest$Group, PredProb[, 1])

  # plot
  pl <- ggplot(data = roc, aes(x = fpr, y = tpr)) +
    geom_path(color = "red", size = 1) +
    geom_abline(intercept = 0, slope = 1,
                color = "grey", linewidth = 1, linetype = 2) +
    labs(x = "False Positive Rate (1 - Specificity)",
         y = "True Positive Rate",
         title = paste0("AUROC (", nfeature, " Features)")) +
    annotate("text",
             x = 1 - rocobj_df$specificities[max_value_row] + 0.15,
             y = rocobj_df$sensitivities[max_value_row] - 0.05,
             label = paste0(threshold, " (",
                           rocobj_df$specificities[max_value_row], ",",
                           rocobj_df$sensitivities[max_value_row], ")"))
}
```

```

rocobj_df$sensitivities[max_value_row], ")"),
  size = 5, family = "serif") +
  annotate("point",
    x = 1 - rocobj_df$specificities[max_value_row],
    y = rocobj_df$sensitivities[max_value_row],
    color = "black", size = 2) +
  annotate("text",
    x = .75, y = .25,
    label = roc_CI_lab,
    size = 5, family = "serif") +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1)) +
  theme_bw() +
  theme(panel.background = element_rect(fill = "transparent"),
    plot.title = element_text(size = 12, color = "black", face = "bold"),
    axis.title = element_text(size = 11, color = "black", face = "bold"),
    axis.text = element_text(size = 10, color = "black"),
    axis.ticks.length = unit(0.4, "lines"),
    axis.ticks = element_line(color = "black"),
    axis.line = element_line(size = .5, color = "black"),
    text = element_text(size = 8, color = "black", family = "serif"))

res <- list(rocobj = rocobj,
            roc_CI = roc_CI_lab,
            roc_pl = pl)

return(res)
}

AUROC_res <- AUROC(
  DataTest = y_test,
  PredProb = pred_prob,
  nfeature = 20)

AUROC_res$roc_pl

```

- AUPRC 曲线：使用 pROC::roc(Robin 等 2011) 函数

```

AUPRC <- function(DataTest, PredProb, nfeature) {
  # plot

```

```

pl <- ggplot(data = prc, aes(x = recall, y = precision)) +
  geom_path(color = "red", size = 1) +
  labs(x = "Recall",
       y = "Precision",
       title = paste0("AUPRC (", nfeature, " Features)")) +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1)) +
  theme_bw() +
  theme(panel.background = element_rect(fill = "transparent"),
        plot.title = element_text(color = "black", size = 14, face = "bold"),
        axis.ticks.length = unit(0.4, "lines"),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(size = .5, color = "black"),
        axis.title = element_text(color = "black", size = 12, face = "bold"),
        axis.text = element_text(color = "black", size = 10),
        text = element_text(size = 8, color = "black", family = "serif"))

res <- list(dat_PR = dat_PR,
            PC_pl = pl)

return(res)
}

AUPRC_res <- AUPRC(
  DataTest = y_test,
  PredProb = pred_prob,
  nfeature = 20)

AUPRC_res$PC_pl

```

- 模型评估参数: 使用 `pROC::roc`(Robin 等 2011) 和 `MLmetrics::PRAUC`(Y. Yan 2016) 函数

```

Evaluate_index <- function(DataTest, PredProb, label, PredRaw) {

threshold <- rocobj_df$threshold[max_value_row]
sen <- round(TP / (TP + FN), 3) # caret::sensitivity(con_matrix)
spe <- round(TN / (TN + FP), 3) # caret::specificity(con_matrix)
acc <- round((TP + TN) / (TP + TN + FP + FN), 3) # Accuracy
pre <- round(TP / (TP + FP), 3) # precision
rec <- round(TP / (TP + FN), 3) # recall
#F1S <- round(2 * TP / (TP + TN + FP + FN + TP - TN), 3) # F1-Score

```

```

F1S <- round(2 * TP / (2 * TP + FP + FN), 3) # F1-Score
youden <- sen + spe - 1 # youden index

# AUROC
AUROC <- round(as.numeric(auc(DataTest$Group, PredProb[, 1])), 3)

# AUPRC
AUPRC <- round(MLmetrics::PRAUC(y_pred = PredProb[, 1],
                                   y_true = DataTest$Group), 3)

index_df <- data.frame(Index = c("Threshold", "Sensitivity",
                                  "Specificity", "Accuracy",
                                  "Precision", "Recall",
                                  "F1 Score", "Youden index",
                                  "AUROC", "AUPRC"),
                        Value = c(threshold, sen, spe,
                                  acc, pre, rec, F1S,
                                  youden, AUROC, AUPRC)) %>%
  stats::setNames(c("Index", "label"))

return(index_df)
}

Evaluate_index(
  DataTest = y_test,
  PredProb = pred_prob,
  label = group_names[1],
  PredRaw = pred_raw)

```

结果：从表中我们可以看到以下指标及其对应的数值：

- **阈值 (Threshold)**: 0.337。阈值用于将模型的预测结果转换为具体的类别（例如，在二分类问题中，通常将预测概率大于阈值的样本视为正类，小于阈值的视为负类）。
- **灵敏度 (Sensitivity)**: 0.937。也称为真正率 (True Positive Rate, TPR) 或召回率 (Recall)，它表示所有实际为正样本的样本中，被模型正确预测为正样本的比例。
- **特异度 (Specificity)**: 0.286。也称为真负率 (True Negative Rate, TNR)，它表示所有实际为负样本的样本中，被模型正确预测为负样本的比例。
- **准确率 (Accuracy)**: 0.737。它表示模型在所有样本上的正确分类的比例。

- **精度 (Precision)**: 0.747。它表示所有被模型预测为正样本的样本中，实际为正样本的比例。
- **召回率 (Recall)**: 0.937 (注意这个与 Sensitivity 的值是一样的，因为在二分类问题中，Sensitivity 和 Recall 是等价的)。
- **F1 得分 (F1Score)**: 0.831。它是精度和召回率的调和平均值，用于综合衡量两者的表现。
- **Youden 指数**: 0.223。它是一个用于评估诊断测试性能的指标，等于灵敏度与特异度之和减去 1。
- **AUROC (Area Under the Receiver Operating Characteristic Curve)**: 0.732。AUROC 是一种衡量分类模型性能的指标，它表示 ROC 曲线下的面积。
- **AUPRC (Area Under the Precision-Recall Curve)**: 0.573。AUPRC 是另一种衡量分类模型性能的指标，特别适用于正样本数量远少于负样本的情况，它表示 Precision-Recall 曲线下的面积。

19.5 标记基因

在运用 Lasso 回归进行模型训练的过程中，通过 L1 正则化的作用，模型能够自动筛选出那些对预测结果具有显著影响的特征，这些特征对应的 Lasso 系数不为零，因此被称为标记基因或显著特征。

```
optimal_feature <- coef(cv.fit, s = "lambda.min") %>%
  as.matrix() %>%
  as.data.frame() %>%
  setNames("Coef") %>%
  dplyr::filter(Coef != 0) %>%
  tibble::rownames_to_column("FeatureID") %>%
  dplyr::slice(-1)

head(optimal_feature)
```

- 可视化特征

```
main_theme <-
  theme(
    panel.background = element_blank(),
    panel.grid = element_blank(),
    axis.line.x = element_line(linewidth = 0.5, color = "black"),
    axis.line.y = element_line(linewidth = 0.5, color = "black"),
    axis.ticks = element_line(color = "black"),
    axis.title = element_text(size = 11, color = "black", face = "bold"),
    axis.text = element_text(color = "black", size = 10),
```

```

legend.position = "right",
legend.background = element_blank(),
legend.key = element_blank(),
legend.text = element_text(size = 12),
text = element_text(family = "serif"))

optimal_feature_pl <- optimal_feature %>%
  ggplot(aes(x = FeatureID, y = Coef)) +
  geom_bar(stat = "identity", fill = "white", color = "blue") +
  labs(x = "", y = "Coefficient") +
  coord_flip() +
  main_theme

optimal_feature_pl
  
```

- 提取特征的表达谱

```

profile_LASSO <- profile[pmatch(optimal_feature$FeatureID,
                                   rownames(profile)), ]

head(profile_LASSO[, 1:6])
  
```

19.6 输出结果

```

if (!dir.exists("./data/result/ML/LASSO")) {
  dir.create("./data/result/ML/LASSO", recursive = TRUE)
}

write.csv(optimal_feature, "./data/result/ML/LASSO/HCC_LASSO_feature.csv", row.names = F)
write.table(profile_LASSO, "./data/result/ML/LASSO/HCC_LASSO_profile.tsv", row.names = T, sep = "\t")
save(AUROC_res, file = "./data/result/ML/LASSO/HCC_LASSO_AUROC.RData")

if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

pdf("./data/result/Figure/SFig3-A.pdf", width = 5, height = 4)
plot(cv.fit)
  
```

© Hua

```
dev.off()

ggsave("./data/result/Figure/SFig3-B.pdf", AUROC_res$roc_pl, width = 5, height = 4, dpi = 600)
ggsave("./data/result/Figure/SFig3-C.pdf", optimal_feature_pl, width = 5, height = 4, dpi = 600)
```

19.7 总结

采用了一种结合差异基因分析和机器学习（LASSO+LR 模型）的方法，来进行基因筛选和分类预测。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version 3.8.0

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets   utils      methods    base

other attached packages:
[1] MLmetrics_1.1.3      pROC_1.18.5        ROCR_1.0-11
[4] caret_6.0-94         lattice_0.22-6     glmnet_4.1-8
[7] Matrix_1.6-5         Biobase_2.62.0     BiocGenerics_0.48.1
[10] lubridate_1.9.3     forcats_1.0.0      stringr_1.5.1
[13] dplyr_1.1.4          purrr_1.0.2       readr_2.1.5
[16] tidyverse_1.3.1      tibble_3.2.1       ggplot2_3.5.1
[19] tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] gtable_0.3.5        shape_1.4.6.1      xfun_0.43
```

[4] recipes_1.0.10 tzdb_0.4.0 vctrs_0.6.5
[7] tools_4.3.3 generics_0.1.3 stats4_4.3.3
[10] parallel_4.3.3 fansi_1.0.6 ModelMetrics_1.2.2.2
[13] pkgconfig_2.0.3 data.table_1.15.4 lifecycle_1.0.4
[16] compiler_4.3.3 munsell_0.5.1 codetools_0.2-19
[19] htmltools_0.5.8.1 class_7.3-22 yaml_2.3.8
[22] prodlim_2023.08.28 pillar_1.9.0 MASS_7.3-60.0.1
[25] gower_1.0.1 iterators_1.0.14 rpart_4.1.23
[28] foreach_1.5.2 parallely_1.37.1 lava_1.8.0
[31] nlme_3.1-164 tidyselect_1.2.1 digest_0.6.35
[34] future_1.33.2 stringi_1.8.4 reshape2_1.4.4
[37] listenv_0.9.1 splines_4.3.3 fastmap_1.1.1
[40] grid_4.3.3 colorspace_2.1-0 cli_3.6.2
[43] magrittr_2.0.3 survival_3.7-0 utf8_1.2.4
[46] future.apply_1.11.2 withr_3.0.0 scales_1.3.0
[49] timechange_0.3.0 rmarkdown_2.26 globals_0.16.3
[52] nnet_7.3-19 timeDate_4032.109 hms_1.1.3
[55] evaluate_0.23 knitr_1.46 hardhat_1.3.1
[58] rlang_1.1.3 Rcpp_1.0.12 glue_1.7.0
[61] BiocManager_1.30.23 renv_1.0.0 ipred_0.9-14
[64] rstudioapi_0.16.0 jsonlite_1.8.8 R6_2.5.1
[67] plyr_1.8.9

第二十章 Boruta RF

采用了 Boruta 结合 RF (Random Forest) 的方法，对差异基因（参考章节 [十一](#)）进行了特征筛选。

20.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(BioBase)
library(randomForest)
library(caret)
library(mlbench)
library(Boruta)
library(pROC)
library(Hmisc)
library(MLmetrics)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

20.2 导入数据

- 差异结果来自于章节 [十一](#)；
- `ExpressionSet` 来自于章节 [九](#)。

```
da_res <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma_select.csv")

ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

20.3 准备数据

- 差异基因：结果解析见表 11.1

```
signif_DEG <- da_res %>%
  dplyr::filter(Enrichment != "Nonsignif") %>%
  dplyr::slice(-grep("\\"., FeatureID))

head(signif_DEG[, 1:6])
```

- 基因表达谱：行名是样本，列名是基因 ID

```
profile <- exprs(ExprSet) %>%
  as.data.frame()
rownames(profile) <- make.names(rownames(profile))

head(profile_DEG[, 1:6])
```

- 去除共线性特征

在机器学习中，数据预处理中使用去除共线性特征的方法，其主要目的是为了提升模型的性能、稳定性和准确性。以下是具体的几个原因：

这里通过 `Hmisc:::rcorr`(Harrell Jr 和 Harrell Jr 2019) 采用了相关系数分析去除共线性。

```
if (file.exists("./data/result/ML/RF/HCC_RF_RM_profile.tsv")) {
  profile_remain <- fread("./data/result/ML/RF/HCC_RF_RM_profile.tsv", sep = "\t") %>%
    tibble::column_to_rownames("V1")
} else {
  profile_remain <- profile_DEG %>%
    dplyr::select(all_of(rownames(feature_remain)))
}

write.table(profile_remain, "./data/result/ML/RF/HCC_RF_RM_profile.tsv",
            row.names = T, sep = "\t", quote = F)
}
```

 print(dim(profile_remain))

- 临床表型表：包含分组等信息

metadata <- pData(ExprSet)

head(metadata[, 1:6])

- 合并数据

```
MergeData <- metadata %>%
```

```
dplyr::select(SampleID, Group) %>%
```

```
dplyr::inner_join(profile_remain %>%
```

```
  tibble::rownames_to_column("SampleID"),
```

```
  by = "SampleID") %>%
```

```
tibble::column_to_rownames("SampleID") %>%
```

```
dplyr::mutate(Group = recode(Group, "Early Stage" = "Early",
```

```
"Late Stage" = "Late")) %>%
```

```
dplyr::mutate(Group = factor(Group))
```

```
head(MergeData[, 1:6])
```

20.4 机器学习特征筛选

特征筛选步骤：

1. 数据分割：

将原始数据集划分为训练集、验证集和测试集。通常，训练集用于模型训练，验证集用于调整超参数和选择最佳模型，测试集用于评估最终模型的性能。划分比例可以根据数据集的大小和特性进行调整，但一般常见的比例是训练集占 60%-80%，验证集和测试集各占 10%-20%。

2. 数据转换：

数据清洗：处理缺失值、异常值、重复值等。对于缺失值，可以使用均值、中位数、众数等填充，或采用插值法、机器学习预测等方法进行填充。

特征编码：对于分类变量，需要进行编码以便模型能够处理。常见的编码方式包括独热编码（One-Hot Encoding）、标签编码（Label Encoding）等。

特征离散化：将连续特征转换为离散类别，例如通过分箱操作。这有助于处理一些具有非线性关系的特征。

特征标准化或归一化：对于不同量纲或不同分布的特征，需要进行标准化或归一化，以便模型能够更好地处理它们。

3. 特征筛选：

相关性分析：计算每个特征与目标变量之间的相关性，如皮尔逊相关系数或斯皮尔曼等级相关系数。通过相关性分析，可以初步筛选出与目标变量强相关的特征。

树模型重要性：利用决策树、随机森林等树模型评估特征的重要性。这些模型可以根据特征在树中分裂的贡献度来评估特征的重要性。

启发式搜索策略：如前向序列选择方法，从空的候选集合出发，逐步添加与目标变量相关性最强的特征。

全局最优搜索策略：如穷举法或分支界定法，从所有可能的特征组合中挑选出表现最优的特征子集。

4. 训练集构建模型：使用经过特征筛选的训练集数据构建机器学习模型。选择合适的机器学习算法，如逻辑回归、支持向量机、决策树、随机森林、神经网络等。
5. 调参：使用验证集对模型进行调参，优化模型的超参数。超参数是机器学习算法中需要人为设定的参数，如学习率、迭代次数、正则化参数等。可以采用网格搜索（Grid Search）、随机搜索（Random Search）等方法进行调参。
6. 测试集评估模型：使用测试集对经过调参的模型进行评估，计算模型的性能指标，如准确率、精确率、召回率、F1 分数、AUC-ROC 等。根据评估结果选择性能最佳的模型作为最终模型。

20.4.1 数据分割

在机器学习的实践中，数据分割是一个至关重要的步骤，用于将原始数据集划分为训练集和测试集（使用了 `caret::createDataPartition` ([Kuhn 2008](#))）。

```
set.seed(123)

trainIndex <- caret::createDataPartition(
    MergeData$Group,
    p = 0.8,
    list = FALSE,
    times = 1)

trainData <- MergeData[trainIndex, ]
X_train <- trainData[, -1]
y_train <- trainData[, 1]

testData <- MergeData[-trainIndex, ]
X_test <- testData[, -1]
y_test <- testData[, 1]
```

20.4.2 基础模型

使用 `randomForest` 函数 (Liaw, Wiener, 等 2002) 构建随机森林的基础模型，获得特征的重要性得分矩阵。

```
set.seed(123)
base.fit <- randomForest(
  Group ~ .,
  data = trainData,
  importance = TRUE)

base.fit

• 特征的重要性得分

head(imp_biomarker)
```

20.4.3 Boruta 特征筛选

Boruta 特征筛选 (使用 Boruta (Kursa 和 Rudnicki 2010) R 包) 是一种基于随机森林的特征选择方法，旨在从给定的特征集合中找到真正重要的特征，并区分无关的特征。它的核心思想是通过创建随机生成的“影子”特征 (shadow features) 来模拟随机选择的特征，并将这些影子特征与原始特征进行比较，以确定每个原始特征的重要性。

```
if (!file.exists("./data/result/ML/RF/HCC_RF_Boruta.RData")) {
  set.seed(123)

  bank_df <- Boruta:::attStats(boruta.bank)
  print(bank_df)

  if (!dir.exists("./data/result/ML/RF")) {
    dir.create("./data/result/ML/RF", recursive = TRUE)
  }

  save(X_train, y_train, X_test, y_test,
        fs_Boruta, boruta.bank, bank_df,
        file = "./data/result/ML/RF/HCC_RF_Boruta.RData")
} else {
  load("./data/result/ML/RF/HCC_RF_Boruta.RData")
}
```

```

©Hua
plot(boruta.bank, xlab = "", xaxt = "n")
lz <- lapply(1:ncol(boruta.bank$ImpHistory), function(i) {
  boruta.bank$ImpHistory[is.finite(boruta.bank$ImpHistory[, i])], i]
})
names(lz) <- colnames(boruta.bank$ImpHistory)
Labels <- sort(sapply(lz, median))
axis(side = 1, las = 2, labels = names(Labels),
at = 1:ncol(boruta.bank$ImpHistory), cex.axis = 0.7)

trainData_select <- trainData %>%
  dplyr::select(all_of(c("Group", rownames(feature_Boruta))))
X_train_select <- trainData_select[, -1]
y_train_select <- trainData_select[, 1]

testData_select <- testData %>%
  dplyr::select(all_of(c("Group", rownames(feature_Boruta))))
X_test_select <- testData_select[, -1]
y_test_select <- testData_select[, 1]

```

20.4.4 调参

为了优化随机森林模型的性能，采用了重复交叉验证（repeated cross-validation）的方法来调整其两个关键参数：*mtry*（每棵树中随机选择的特征数）和*ntree*（树的总数）。在调参过程中，设定了不同的*mtry* 和 *ntree* 参数组合，并使用重复交叉验证来评估每个组合下的模型性能。这种方法通过多次重复地划分训练集和验证集，并对每个参数组合进行多次评估，从而获得了更加稳定和可靠的性能指标。最终，根据这些性能指标，选择出能够使模型性能达到最优的*mtry* 和 *ntree* 参数值。

```

set.seed(123)
## Print the best tuning parameter that maximizes model accuracy
optimalVar <- data.frame(tune_fit$results[which.max(tune_fit$results[, 3]), ])
# print(optimalVar)

## Plot model accuracy vs different values of Cost
print(plot(tune_fit))

```

- 最佳特征：通过重复运行获得误差确定最佳特征数目

```

n.var <- as.numeric(rownames(error.cv))
colnames(error.cv) <- paste("error", 1:5, sep = ".")

```

```

err.mean <- apply(error.cv, 1, mean)
err.df <- data.frame(num = n.var,
                      err.mean = err.mean,
                      error.cv)
head(err.df[, 1:6])

• 最佳特征可视化

optimal <- err.df$num[which(err.df$err.mean == min(err.df$err.mean))]
main_theme <-
  theme(
    panel.background = element_blank(),
    panel.grid = element_blank(),
    axis.line.x = element_line(linewidth = 0.5, color = "black"),
    axis.line.y = element_line(linewidth = 0.5, color = "black"),
    axis.ticks = element_line(color = "black"),
    axis.title = element_text(size = 11, color = "black", face = "bold"),
    axis.text = element_text(color = "black", size = 10),
    legend.position = "right",
    legend.background = element_blank(),
    legend.key = element_blank(),
    legend.text = element_text(size = 12),
    text = element_text(family = "serif"))

pl <-
  ggplot(data = err.df, aes(x = err.df$num)) +
  geom_line(aes(y = err.df$error.1), color = "grey", linewidth = 0.5) +
  geom_line(aes(y = err.df$error.2), color = "grey", linewidth = 0.5) +
  geom_line(aes(y = err.df$error.3), color = "grey", linewidth = 0.5) +
  geom_line(aes(y = err.df$error.4), color = "grey", linewidth = 0.5) +
  geom_line(aes(y = err.df$error.5), color = "grey", linewidth = 0.5) +
  geom_line(aes(y = err.df$err.mean), color = "black", linewidth = 0.5) +
  geom_vline(xintercept = optimal, color = "red", lwd = 0.36, linetype = 2) +
  coord_trans(x = "log2") +
  scale_x_continuous(breaks = c(1, 5, 10, 20, 30)) +
  labs(x = "Number of Features ", y = "Cross-validation error rate") +
  annotate("text",
    x = optimal - 6,
    y = max(err.df$err.mean),
    label = paste("Optimal = ", optimal, sep = ""),
    color = "red") +

```

```
main_theme  
pl
```

20.4.5 最终分类模型

在已经确定了 *mtry* 和 *ntree* 这两个关键参数的最优值，并选择了合适的特征数目之后，将这些参数和特征应用到随机森林模型的构建中。

```
selected_biomarker <- imp_biomarker %>%  
  dplyr::filter(FeatureID %in% rownames(feature_Boruta)) %>%  
  dplyr::slice(1:optimal)  
  
selected_columns <- c("Group", selected_biomarker$FeatureID)  
  
trainData_optimal <- trainData_select %>%  
  dplyr::select(all_of(selected_columns))  
  
testData_optimal <- testData_select %>%  
  dplyr::select(all_of(selected_columns))  
  
set.seed(123)  
rf_fit_optimal
```

结果：相比基础模型来说，最终模型降低了错误率。

20.4.6 测试集验证

- 混淆矩阵

```
print(caret::confusionMatrix(pred_raw, testData_optimal$Group))
```

- AUROC 曲线：使用 pROC::roc(Robin 等 2011) 函数

```
AUROC <- function(  
  DataTest,  
  PredProb = pred_prob,  
  nfeature) {  
  
  # plot
```

```

pl <- ggplot(data = roc, aes(x = fpr, y = tpr)) +
  geom_path(color = "red", size = 1) +
  geom_abline(intercept = 0, slope = 1,
              color = "grey", linewidth = 1, linetype = 2) +
  labs(x = "False Positive Rate (1 - Specificity)",
       y = "True Positive Rate",
       title = paste0("AUROC (", nfeature, " Features)")) +
  annotate("text",
           x = 1 - rocbj_df$specificities[max_value_row] + 0.15,
           y = rocbj_df$sensitivities[max_value_row] - 0.05,
           label = paste0(threshold, " (",
                         rocbj_df$specificities[max_value_row], ",",
                         rocbj_df$sensitivities[max_value_row], ")"),
           size=5, family="serif") +
  annotate("point",
           x = 1 - rocbj_df$specificities[max_value_row],
           y = rocbj_df$sensitivities[max_value_row],
           color = "black", size = 2) +
  annotate("text",
           x = .75, y = .25,
           label = roc_CI_lab,
           size = 5, family = "serif") +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1)) +
  theme_bw() +
  theme(panel.background = element_rect(fill = "transparent"),
        plot.title = element_text(size = 12, color = "black", face = "bold"),
        axis.title = element_text(size = 11, color = "black", face = "bold"),
        axis.text = element_text(size= 10, color = "black"),
        axis.ticks.length = unit(0.4, "lines"),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(size = .5, color = "black"),
        text = element_text(size = 8, color = "black", family = "serif"))

res <- list(rocobj = rocobj,
            roc_CI = roc_CI_lab,
            roc_pl = pl)

return(res)
}

```

```
AUROC_res <- AUROC(
```

```
  DataTest = testData_select,
  PredProb = pred_prob,
  nfeature = optimal)
```

```
AUROC_res$roc_pl
```

- AUPRC 曲线：使用 `pROC::roc`(Robin 等 2011) 函数

```
AUPRC <- function(DataTest, PredProb, nfeature) {
```

```
  # plot
```

```
  pl <- ggplot(data = prc, aes(x = recall, y = precision)) +
    geom_path(color = "red", size = 1) +
    labs(x = "Recall",
         y = "Precision",
         title = paste0("AUPRC (", nfeature, " Features)")) +
    coord_cartesian(xlim = c(0, 1), ylim = c(0, 1)) +
    theme_bw() +
    theme(panel.background = element_rect(fill = "transparent"),
          plot.title = element_text(color = "black", size = 14, face = "bold"),
          axis.ticks.length = unit(0.4, "lines"),
          axis.ticks = element_line(color = "black"),
          axis.line = element_line(size = .5, color = "black"),
          axis.title = element_text(color = "black", size = 12, face = "bold"),
          axis.text = element_text(color = "black", size = 10),
          text = element_text(size = 8, color = "black", family = "serif"))
```

```
  res <- list(dat_PR = dat_PR,
              PC_pl = pl)
```

```
  return(res)
}
```

```
AUPRC_res <- AUPRC(
```

```
  DataTest = testData_select,
  PredProb = pred_prob,
  nfeature = optimal)
```

```
AUPRC_res$PC_pl
```

- 模型评估参数：使用 `pROC::roc`(Robin 等 2011) 和 `MLmetrics::PRAUC`(Y. Yan 2016) 函数

```

Evaluate_index <- function(DataTest, PredProb, label, PredRaw) {

  # ROC object
  rocobj <- roc(DataTest$Group, PredProb[, 1])

  # confusionMatrix
  con_matrix <- table(PredRaw, DataTest$Group)

  threshold <- rocobj_df$threshold[max_value_row]
  sen <- round(TP / (TP + FN), 3) # caret::sensitivity(con_matrix)
  spe <- round(TN / (TN + FP), 3) # caret::specificity(con_matrix)
  acc <- round((TP + TN) / (TP + TN + FP + FN), 3) # Accuracy
  pre <- round(TP / (TP + FP), 3) # precision
  rec <- round(TP / (TP + FN), 3) # recall
  #F1S <- round(2 * TP / (TP + TN + FP + FN + TP - TN), 3) # F1-Score
  F1S <- round(2 * TP / (2 * TP + FP + FN), 3) # F1-Score
  youden <- sen + spe - 1 # youden index

  # AUROC
  AUROC <- round(as.numeric(auc(DataTest$Group, PredProb[, 1])), 3)

  # AUPRC
  AUPRC <- round(MLmetrics::PRAUC(y_pred = PredProb[, 1],
                                    y_true = DataTest$Group), 3)

  index_df <- data.frame(Index = c("Threshold", "Sensitivity",
                                   "Specificity", "Accuracy",
                                   "Precision", "Recall",
                                   "F1 Score", "Youden index",
                                   "AUROC", "AUPRC"),
                           Value = c(threshold, sen, spe,
                                     acc, pre, rec, F1S,
                                     youden, AUROC, AUPRC)) %>%
    stats::setNames(c("Index", label))

  return(index_df)
}

```

```

        }

Evaluate_index(
  DataTest = testData_select,
  PredProb = pred_prob,
  label = group_names[1],
  PredRaw = pred_raw)

```

结果：从表中我们可以看到以下指标及其对应的数值：

- **阈值 (Threshold)**: 0.585。阈值用于将模型的预测结果转换为具体的类别（例如，在二分类问题中，通常将预测概率大于阈值的样本视为正类，小于阈值的视为负类）。
- **灵敏度 (Sensitivity)**: 0.861。也称为真正率 (True Positive Rate, TPR) 或召回率 (Recall)，它表示所有实际为正样本的样本中，被模型正确预测为正样本的比例。
- **特异度 (Specificity)**: 0.429。也称为真负率 (True Negative Rate, TNR)，它表示所有实际为负样本的样本中，被模型正确预测为负样本的比例。
- **准确率 (Accuracy)**: 0.728。它表示模型在所有样本上的正确分类的比例。
- **精度 (Precision)**: 0.773。它表示所有被模型预测为正样本的样本中，实际为正样本的比例。
- **召回率 (Recall)**: 0.861（注意这个与 Sensitivity 的值是一样的，因为在二分类问题中，Sensitivity 和 Recall 是等价的）。
- **F1 得分 (F1Score)**: 0.814。它是精度和召回率的调和平均值，用于综合衡量两者的表现。
- **Youden 指数**: 0.290。它是一个用于评估诊断测试性能的指标，等于灵敏度与特异度之和减去 1。
- **AUROC (Area Under the Receiver Operating Characteristic Curve)**: 0.704。AUROC 是一种衡量分类模型性能的指标，它表示 ROC 曲线下的面积。
- **AUPRC (Area Under the Precision-Recall Curve)**: 0.220。AUPRC 是另一种衡量分类模型性能的指标，特别适用于正样本数量远少于负样本的情况，它表示 Precision-Recall 曲线下的面积。

20.5 标记基因

通过 Boruta 特征筛选和随机森林交叉误差率，模型能够自动筛选出那些对预测结果具有显著影响的特征，因此被称为标记基因或显著特征。

```

optimal_feature <- imp_biomarker %>%
  dplyr::filter(FeatureID %in% rownames(feature_Boruta)) %>%
  dplyr::slice(1:optimal) %>%

```

```
dplyr::select(FeatureID, MeanDecreaseAccuracy) %>%
dplyr::arrange(MeanDecreaseAccuracy) %>%
dplyr::mutate(FeatureID = forcats::fct_inorder(FeatureID))

head(optimal_feature)
```

- 可视化特征

```
optimal <- 39
```

```
optimal_feature_pl <- imp_biomarker %>%
  dplyr::filter(FeatureID %in% rownames(feature_Boruta)) %>%
  dplyr::slice(1:optimal) %>%
  dplyr::select(FeatureID, MeanDecreaseAccuracy) %>%
  dplyr::arrange(MeanDecreaseAccuracy) %>%
  dplyr::mutate(FeatureID = forcats::fct_inorder(FeatureID)) %>%
  ggplot(aes(x = FeatureID, y = MeanDecreaseAccuracy)) +
  geom_bar(stat = "identity", fill = "white", color = "blue") +
  labs(x = "", y = "Mean decrease accuracy") +
  coord_flip() +
  main_theme
```

```
optimal_feature_pl
```

- 提取特征的表达谱

```
profile_RF <- profile[pmatch(optimal_feature$FeatureID,
                                rownames(profile)), ]
```

```
head(profile_RF[, 1:6])
```

20.6 输出结果

```
if (!dir.exists("./data/result/ML/RF")) {
  dir.create("./data/result/ML/RF", recursive = TRUE)
}

write.csv(optimal_feature, "./data/result/ML/RF/HCC_RF_feature.csv", row.names = F)
```

```
write.table(profile_RF, "./data/result/ML/RF/HCC_RF_profile.tsv", row.names = T, sep = "\t", quote = FALSE)
save(AUROC_res, file = "./data/result/ML/RF/HCC_RF_AUROC.RData")

if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

pdf("./data/result/Figure/SFig3-D.pdf", width = 5, height = 4)
plot(boruta.bank, xlab = "", xaxt = "n")
lz <- lapply(1:ncol(boruta.bank$ImpHistory), function(i) {
  boruta.bank$ImpHistory[is.finite(boruta.bank$ImpHistory[, i])], i
})
names(lz) <- colnames(boruta.bank$ImpHistory)
Labels <- sort(sapply(lz, median))
axis(side = 1, las = 2, labels = names(Labels),
     at = 1:ncol(boruta.bank$ImpHistory), cex.axis = 0.7)
dev.off()

ggsave("./data/result/Figure/SFig3-E.pdf", AUROC_res$roc_pl, width = 5, height = 4, dpi = 600)
ggsave("./data/result/Figure/SFig3-F.pdf", optimal_feature_pl, width = 5, height = 5, dpi = 600)
```

20.7 总结

经过对差异基因进行 **Boruta** 特征筛选，成功识别出一组与特定生物学现象或条件紧密相关的基因子集。接着，利用随机森林模型进行进一步的特征选择，以找到在预测性能方面表现最佳的模型特征向量。通过这一详尽的模型训练和评估过程，最终确定了 **39** 个特征，这些特征在预测任务中展现出了相对不错的性能。这 **39** 个特征将被用于后续的分析，以进一步揭示这些差异基因在生物学过程中的作用机制和潜在价值。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v
```

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats graphics grDevices datasets utils methods base

other attached packages:
[1] MLmetrics_1.1.3 Hmisc_5.1-2 pROC_1.18.5
[4] Boruta_8.0.0 mlbench_2.1-5 caret_6.0-94
[7] lattice_0.22-6 randomForest_4.7-1.1 Biobase_2.62.0
[10] BiocGenerics_0.48.1 data.table_1.15.4 lubridate_1.9.3
[13]forcats_1.0.0 stringr_1.5.1 dplyr_1.1.4
[16] purrr_1.0.2 readr_2.1.5 tidyverse_2.0.0
[19] tibble_3.2.1 ggplot2_3.5.1 tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] tidyselect_1.2.1 timeDate_4032.109 fastmap_1.1.1
[4] digest_0.6.35 rpart_4.1.23 timechange_0.3.0
[7] lifecycle_1.0.4 cluster_2.1.6 survival_3.7-0
[10] magrittr_2.0.3 compiler_4.3.3 rlang_1.1.3
[13] tools_4.3.3 utf8_1.2.4 yaml_2.3.8
[16] knitr_1.46 htmlwidgets_1.6.4 plyr_1.8.9
[19] foreign_0.8-86 withr_3.0.0 nnet_7.3-19
[22] grid_4.3.3 stats4_4.3.3 fansi_1.0.6
[25] colorspace_2.1-0 future_1.33.2 globals_0.16.3
[28] scales_1.3.0 iterators_1.0.14 MASS_7.3-60.0.1
[31] cli_3.6.2 rmarkdown_2.26 generics_0.1.3
[34] rstudioapi_0.16.0 future.apply_1.11.2 reshape2_1.4.4
[37] tzdb_0.4.0 splines_4.3.3 parallel_4.3.3
[40] BiocManager_1.30.23 base64enc_0.1-3 vctrs_0.6.5
[43] hardhat_1.3.1 Matrix_1.6-5 jsonlite_1.8.8
[46] hms_1.1.3 htmlTable_2.4.2 Formula_1.2-5
[49] listenv_0.9.1 foreach_1.5.2 gower_1.0.1
[52] recipes_1.0.10 glue_1.7.0 parallely_1.37.1
[55] codetools_0.2-19 stringi_1.8.4 gtable_0.3.5
[58] munsell_0.5.1 pillar_1.9.0 htmltools_0.5.8.1
[61] ipred_0.9-14 lava_1.8.0 R6_2.5.1

[64] evaluate_0.23 backports_1.4.1 renv_1.0.0
[67] class_7.3-22 Rcpp_1.0.12 checkmate_2.3.1
[70] gridExtra_2.3 nlme_3.1-164 prodlim_2023.08.28
[73] xfun_0.43 pkgconfig_2.0.3 ModelMetrics_1.2.2.2

第二十一章 REF SVM

采用了 **REF** (**R**ecursive **F**eature **E**limination) 结合 **SVM** (**S**upport **V**ector **M**achine) 的方法，对差异基因（参考章节 [十一](#)）进行了特征筛选。通过这种方法，能够从大量的候选基因中识别出与特定生物过程或疾病状态最为相关的关键基因。筛选出的这些重要基因不仅具有统计学上的显著性，而且在实际应用中具有较高的预测能力。这些经过 **REF+SVM** 筛选的重要基因将被用于后续的生物信息学分析、疾病机制探究以及潜在治疗靶点的识别等研究中，以期进一步推动相关领域的科学进展。

21.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(BioBase)
library(data.table)
library(caret)
library(e1071)
library(mlbench)
library(pROC)
library(Hmisc)
library(MLmetrics)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

21.2 导入数据

- 差异结果来自于章节 十一；
- ExpressionSet 来自于章节 九。

```
da_res <- read.csv("./data/result/DA/HCC_Early_vs_Late_limma_select.csv")
```

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

21.3 准备数据

- 差异基因：结果解析见表 11.1

```
signif_DEG <- da_res %>%
  dplyr::filter(Enrichment != "Nonsignif") %>%
  dplyr::slice(-grep("\\.", FeatureID))

head(signif_DEG[, 1:6])
```

- 基因表达谱：行名是样本，列名是基因 ID

```
profile <- exprs(ExprSet) %>%
  as.data.frame()
rownames(profile) <- make.names(rownames(profile))

head(profile[, 1:6])
```

- 去除共线性特征

在机器学习中，数据预处理中使用去除共线性特征的方法，其主要目的是为了提升模型的性能、稳定性和准确性。以下是具体的几个原因：

1. 提高模型准确性：共线性特征指的是在数据集中存在高度相关性的特征。这些特征在建模时可能会提供冗余的信息，甚至可能导致模型过拟合，从而降低预测的准确性。通过去除共线性特征，可以减少这些冗余信息，使模型更加专注于关键信息，从而提高预测的准确性。
2. 提高模型稳定性：当数据集中存在共线性特征时，模型的参数可能会变得不稳定，对训练数据的微小变化都可能会产生较大的影响。通过去除共线性特征，可以降低模型的复杂性，提高模型的稳定性，使模型在面对新的、未见过的数据时能够保持较好的性能。

3. 加速模型训练：在训练模型时，如果数据集中存在大量的共线性特征，那么模型可能需要更多的时间和计算资源来找到最优的参数组合。通过去除共线性特征，可以减少模型的输入维度，降低模型的复杂性，从而加速模型的训练过程。

常用的去除共线性特征的方法包括：

1. 方差膨胀因子（VIF）：VIF 用于量化特征之间的共线性程度。当 VIF 值较高时，表示该特征与其他特征之间存在较强的共线性关系。可以通过设定一个阈值，将 VIF 值超过该阈值的特征进行剔除或合并。
2. 主成分分析（PCA）：PCA 是一种常用的降维方法，它通过找到数据中的主要变化方向（即主成分）来降低数据的维度。PCA 可以有效地去除共线性特征，因为它会生成一组新的、不相关的特征（即主成分），这些特征能够捕获原始数据中的大部分信息。
3. 相关系数分析：通过分析特征之间的相关系数，可以找出存在高度相关性的特征对。对于相关系数较高的特征对，可以选择其中一个特征进行保留，而将另一个特征进行剔除或合并。

这里通过 `Hmisc::rcorr`(Harrell Jr 和 Harrell Jr 2019) 采用了相关系数分析去除共线性。

```
if (file.exists("./data/result/ML/SVM/HCC_SVM_RM_profile.tsv")) {
  profile_remain <- fread("./data/result/ML/SVM/HCC_SVM_RM_profile.tsv", sep = "\t") %>%
    tibble::column_to_rownames("V1")
} else {

  write.table(profile_remain, "./data/result/ML/SVM/HCC_SVM_RM_profile.tsv",
              row.names = T, sep = "\t", quote = F)
}

print(dim(profile_remain))
```

- 临床表型表：包含分组等信息

```
metadata <- pData(ExprSet)
head(metadata[, 1:6])
```

- 合并数据

```
MergeData <- metadata %>%
  dplyr::select(SampleID, Group) %>%
  dplyr::inner_join(profile_remain %>%
    tibble::rownames_to_column("SampleID"),
    by = "SampleID") %>%
  tibble::column_to_rownames("SampleID") %>%
```

```
dplyr::mutate(Group = recode(Group, "Early Stage" = "Early",
                               "Late Stage" = "Late")) %>%
dplyr::mutate(Group = factor(Group))

head(MergeData[, 1:6])
```

21.4 机器学习特征筛选

特征筛选步骤：

1. 数据分割：

将原始数据集划分为训练集、验证集和测试集。通常，训练集用于模型训练，验证集用于调整超参数和选择最佳模型，测试集用于评估最终模型的性能。划分比例可以根据数据集的大小和特性进行调整，但一般常见的比例是训练集占 60%-80%，验证集和测试集各占 10%-20%。

2. 数据转换：

数据清洗：处理缺失值、异常值、重复值等。对于缺失值，可以使用均值、中位数、众数等填充，或采用插值法、机器学习预测等方法进行填充。

特征编码：对于分类变量，需要进行编码以便模型能够处理。常见的编码方式包括独热编码（One-Hot Encoding）、标签编码（Label Encoding）等。

特征离散化：将连续特征转换为离散类别，例如通过分箱操作。这有助于处理一些具有非线性关系的特征。

特征标准化或归一化：对于不同量纲或不同分布的特征，需要进行标准化或归一化，以便模型能够更好地处理它们。

3. 特征筛选：

相关性分析：计算每个特征与目标变量之间的相关性，如皮尔逊相关系数或斯皮尔曼等级相关系数。通过相关性分析，可以初步筛选出与目标变量强相关的特征。

树模型重要性：利用决策树、随机森林等树模型评估特征的重要性。这些模型可以根据特征在树中分裂的贡献度来评估特征的重要性。

启发式搜索策略：如前向序列选择方法，从空的候选集合出发，逐步添加与目标变量相关性最强的特征。

全局最优搜索策略：如穷举法或分支界定法，从所有可能的特征组合中挑选出表现最优的特征子集。

4. 训练集构建模型：使用经过特征筛选的训练集数据构建机器学习模型。选择合适的机器学习算法，如逻辑回归、支持向量机、决策树、随机森林、神经网络等。

5. 调参：使用验证集对模型进行调参，优化模型的超参数。超参数是机器学习算法中需要人为设定的参数，如学习率、迭代次数、正则化参数等。可以采用网格搜索（Grid Search）、随机搜索（Random Search）等方法进行调参。

6. 测试集评估模型：使用测试集对经过调参的模型进行评估，计算模型的性能指标，如准确率、精确率、召回率、F1 分数、AUC-ROC 等。根据评估结果选择性能最佳的模型作为最终模型。

21.4.1 数据分割

在机器学习的实践中，数据分割是一个至关重要的步骤，用于将原始数据集划分为训练集和测试集（使用了 `caret::createDataPartition` ([Kuhn 2008](#))）。

```
set.seed(123)

trainIndex <- caret::createDataPartition(
    MergeData$Group,
    p = 0.8,
    list = FALSE,
    times = 1)

trainData <- MergeData[trainIndex, ]
X_train <- trainData[, -1]
y_train <- trainData[, 1]

testData <- MergeData[-trainIndex, ]
X_test <- testData[, -1]
y_test <- testData[, 1]
```

21.4.2 基础模型

```
set.seed(123)

base.fit <- e1071::svm(
    Group ~ .,
    data = trainData,
    kernel = "radial")

base.fit
```

21.4.3 Recursive Feature Elimination 特征筛选

递归特征消除 (Recursive Feature Elimination, RFE) 是一种基于模型的特征选择方法，其原理是通过反复训练模型和剔除最不重要特征的方式来选择最优的特征子集。以下是 RFE 特征筛选的具体步骤：

```
if (!file.exists("./data/result/ML/SVM/SVM_preData.RData")) {  
    set.seed(123)  
  
    if (!dir.exists("./data/result/ML/SVM")) {  
        dir.create("./data/result/ML/SVM", recursive = TRUE)  
    }  
  
    save(X_train, y_train, X_test, y_test, fs_rfe,  
          file = "./data/result/ML/SVM/SVM_preData.RData")  
} else {  
    load("./data/result/ML/SVM/SVM_preData.RData")  
}  
  
print(fs_rfe)  
#list the chosen features  
predictors(fs_rfe)[1:41]  
plot(fs_rfe, type = c("g", "o"))  
  
trainData_select <- trainData %>%  
    dplyr::select(all_of(c("Group", feature_rfe)))  
X_train_select <- trainData_select[, -1]  
y_train_select <- trainData_select[, 1]  
  
testData_select <- testData %>%  
    dplyr::select(all_of(c("Group", feature_rfe)))  
X_test_select <- testData_select[, -1]  
y_test_select <- testData_select[, 1]
```

21.4.4 调参

```
set.seed(123)
```

```

if (file.exists("./data/result/ML/SVM/HCC_SVM_tuneFit.RData")) {
  load("./data/result/ML/SVM/HCC_SVM_tuneFit.RData")
} else {

  set.seed(123)
  tune_fit <- train(
    Group ~.,
    data = trainData_select,
    method = "svmLinear", # svmRadial svmLinear svmRadialCost
    trControl = myControl,
    tuneGrid = tuneGrid)

  save(tune_fit, file = "./data/result/ML/SVM/HCC_SVM_tuneFit.RData")
}

## Plot model accuracy vs different values of Cost
print(plot(tune_fit))

## Print the best tuning parameter that maximizes model accuracy
optimalVar <- data.frame(tune_fit$results[which.max(tune_fit$results[, 3]), ])
print(optimalVar)

```

21.4.5 最终分类模型

在已经通过适当的参数调优确定了 *Cost* 关键参数的值之后，可以基于这些参数构建最终的 SVM（支持向量机）模型。这个模型将使用选定的 *Cost* 值来控制对误差的容忍度，从而确保模型在训练数据上具有良好的拟合能力，并在未知数据上保持优秀的泛化性能。

```

optimal <- length(fs_rfe$optVariables[1:selected_num])
selected_columns <- c("Group", fs_rfe$optVariables[1:selected_num])

trainData_optimal <- trainData_select %>%
  dplyr::select(all_of(selected_columns))

testData_optimal <- testData_select %>%
  dplyr::select(all_of(selected_columns))

set.seed(123)

```

```
svm_fit_optimal
```

21.4.6 测试集验证

首先，将训练好的模型应用于测试集数据，以预测每个样本的分类标签。预测结果将与测试集的真实标签进行对比，以计算模型在各个类别上的分类准确性。

为了更详细地了解模型的性能，构建混淆矩阵。混淆矩阵是一个 $N \times N$ 的表格（其中 N 为分类类别数），用于显示每个类别下的真实标签与预测标签之间的对比情况。通过混淆矩阵，可以计算出精确度 (Precision)、召回率 (Recall)、F1 分数 (F1-Score) 等评估指标，这些指标能够全面反映模型在各类别上的分类效果。

此外，还可以绘制 ROC 曲线来评估模型的性能。ROC 曲线是通过设置不同的分类阈值，计算真正例率 (True Positive Rate, TPR) 和假正例率 (False Positive Rate, FPR) 得到的。ROC 曲线越靠近左上角，说明模型在保持较低假正例率的同时，能够获得较高的真正例率，即模型的性能越好。

通过混淆矩阵和 ROC 曲线，可以全面、客观地评估机器学习模型在测试集上的性能。

- 混淆矩阵

```
print(caret::confusionMatrix(pred_raw, testData_optimal$Group))
```

- AUROC 曲线：使用 pROC::roc(Robin 等 2011) 函数

```
AUROC <- function(
  DataTest,
  PredProb = pred_prob,
  nfeature) {

  # plot
  pl <- ggplot(data = roc, aes(x = fpr, y = tpr)) +
    geom_path(color = "red", size = 1) +
    geom_abline(intercept = 0, slope = 1,
                color = "grey", linewidth = 1, linetype = 2) +
    labs(x = "False Positive Rate (1 - Specificity)",
         y = "True Positive Rate",
         title = paste0("AUROC (", nfeature, " Features)")) +
    annotate("text",
             x = 1 - rocbj_df$specificities[max_value_row] + 0.15,
             y = rocbj_df$sensitivities[max_value_row] - 0.05,
             label = paste0(threshold, " (",
                           rocbj_df$specificities[max_value_row], ",",
                           rocbj_df$sensitivities[max_value_row], ")")),
```

©Hua

```

        size = 5, family = "serif") +
  annotate("point",
    x = 1 - rocobj_df$specificities[max_value_row],
    y = rocobj_df$sensitivities[max_value_row],
    color = "black", size = 2) +
  annotate("text",
    x = .75, y = .25,
    label = roc_CI_lab,
    size = 5, family = "serif") +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1)) +
  theme_bw() +
  theme(panel.background = element_rect(fill = "transparent"),
    plot.title = element_text(size = 12, color = "black", face = "bold"),
    axis.title = element_text(size = 11, color = "black", face = "bold"),
    axis.text = element_text(size = 10, color = "black"),
    axis.ticks.length = unit(0.4, "lines"),
    axis.ticks = element_line(color = "black"),
    axis.line = element_line(size = .5, color = "black"),
    text = element_text(size = 8, color = "black", family = "serif"))

res <- list(rocobj = rocobj,
            roc_CI = roc_CI_lab,
            roc_pl = pl)

```

```

return(res)
}
```

```

AUROC_res <- AUROC(
  DataTest = testData_optimal,
  PredProb = pred_prob,
  nfeature = optimal)
```

```
AUROC_res$roc_pl
```

- AUPRC 曲线：使用 pROC:::roc(Robin 等 2011) 函数

```

AUPRC <- function(DataTest, PredProb, nfeature) {

  # plot
  pl <- ggplot(data = prc, aes(x = recall, y = precision)) +
```

```

geom_path(color = "red", size = 1) +
  labs(x = "Recall",
       y = "Precision",
       title = paste0("AUPRC (", nfeature, " Features)")) +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1)) +
  theme_bw() +
  theme(panel.background = element_rect(fill = "transparent"),
        plot.title = element_text(color = "black", size = 14, face = "bold"),
        axis.ticks.length = unit(0.4, "lines"),
        axis.ticks = element_line(color = "black"),
        axis.line = element_line(size = .5, color = "black"),
        axis.title = element_text(color = "black", size = 12, face = "bold"),
        axis.text = element_text(color = "black", size = 10),
        text = element_text(size = 8, color = "black", family = "serif"))

res <- list(dat_PR = dat_PR,
            PC_pl = pl)

return(res)
}

AUPRC_res <- AUPRC(
  DataTest = testData_optimal,
  PredProb = pred_prob,
  nfeature = optimal)

AUPRC_res$PC_pl
  
```

- 模型评估参数：使用 `pROC::roc`(Robin 等 2011) 和 `MLmetrics::PRAUC`(Y. Yan 2016) 函数

```

Evaluate_index <- function(DataTest, PredProb, label, PredRaw) {

  threshold <- rocbj_df$threshold[max_value_row]
  sen <- round(TP / (TP + FN), 3) # caret::sensitivity(con_matrix)
  spe <- round(TN / (TN + FP), 3) # caret::specificity(con_matrix)
  acc <- round((TP + TN) / (TP + TN + FP + FN), 3) # Accuracy
  pre <- round(TP / (TP + FP), 3) # precision
  rec <- round(TP / (TP + FN), 3) # recall
  #F1S <- round(2 * TP / (TP + TN + FP + FN + TP - TN), 3) # F1-Score
  F1S <- round(2 * TP / (2 * TP + FP + FN), 3) # F1-Score
  
```

©Huawei

```

youden <- sen + spe - 1 # youden index

# AUROC
AUROC <- round(as.numeric(auc(DataTest$Group, PredProb[, 1])), 3)

# AUPRC
AUPRC <- round(MLmetrics::PRAUC(y_pred = PredProb[, 1],
                                    y_true = DataTest$Group), 3)

index_df <- data.frame(Index = c("Threshold", "Sensitivity",
                                  "Specificity", "Accuracy",
                                  "Precision", "Recall",
                                  "F1 Score", "Youden index",
                                  "AUROC", "AUPRC"),
                        Value = c(threshold, sen, spe,
                                  acc, pre, rec, F1S,
                                  youden, AUROC, AUPRC)) %>%
  stats::setNames(c("Index", "label"))

return(index_df)
}

Evaluate_index(
  DataTest = testData_optimal,
  PredProb = pred_prob,
  label = group_names[1],
  PredRaw = pred_raw)

```

结果：从表中我们可以看到以下指标及其对应的数值：

- **阈值 (Threshold)**: 0.598。阈值用于将模型的预测结果转换为具体的类别（例如，在二分类问题中，通常将预测概率大于阈值的样本视为正类，小于阈值的视为负类）。
- **灵敏度 (Sensitivity)**: 0.911。也称为真正率 (True Positive Rate, TPR) 或召回率 (Recall)，它表示所有实际为正样本的样本中，被模型正确预测为正样本的比例。
- **特异度 (Specificity)**: 0.343。也称为真负率 (True Negative Rate, TNR)，它表示所有实际为负样本的样本中，被模型正确预测为负样本的比例。
- **准确率 (Accuracy)**: 0.737。它表示模型在所有样本上的正确分类的比例。
- **精度 (Precision)**: 0.758。它表示所有被模型预测为正样本的样本中，实际为正样本的比例。

- **召回率 (Recall)**: 0.911 (注意这个与 Sensitivity 的值是一样的，因为在二分类问题中，Sensitivity 和 Recall 是等价的)。
- **F1 得分 (F1Score)**: 0.828。它是精度和召回率的调和平均值，用于综合衡量两者的表现。
- **Youden 指数**: 0.254。它是一个用于评估诊断测试性能的指标，等于灵敏度与特异度之和减去 1。
- **AUROC (Area Under the Receiver Operating Characteristic Curve)**: 0.663。AUROC 是一种衡量分类模型性能的指标，它表示 ROC 曲线下的面积。
- **AUPRC (Area Under the Precision-Recall Curve)**: 0.231。AUPRC 是另一种衡量分类模型性能的指标，特别适用于正样本数量远少于负样本的情况，它表示 Precision-Recall 曲线下的面积。

21.5 标记基因

通过 REF 特征筛选，模型能够自动筛选出那些对预测结果具有显著影响的特征，因此被称为标记基因或显著特征。

```
optimal_feature <- fs_rfe$variables %>%
  dplyr::rename(FeatureID = var) %>%
  dplyr::filter(FeatureID %in% fs_rfe$optVariables[1:selected_num]) %>%
  dplyr::inner_join(fs_rfe$results, by = "Variables") %>%
  dplyr::select(FeatureID, Accuracy, Kappa, AccuracySD, KappaSD) %>%
  dplyr::arrange(Accuracy) %>%
  dplyr::mutate(FeatureID = forcats::fct_inorder(FeatureID))

optimal_feature <- optimal_feature[pmatch(unique(optimal_feature$FeatureID),
                                         optimal_feature$FeatureID), ,]

head(optimal_feature)
```

- 提取特征的表达谱

```
profile_SVM <- profile[pmatch(optimal_feature$FeatureID,
                                rownames(profile)), ]

head(profile_SVM[, 1:6])
```

21.6 输出结果

```

if (!dir.exists("./data/result/ML/SVM")) {
  dir.create("./data/result/ML/SVM", recursive = TRUE)
}

write.csv(optimal_feature, "./data/result/ML/SVM/HCC_SVM_feature.csv", row.names = F)
write.table(profile_SVM, "./data/result/ML/SVM/HCC_SVM_profile.tsv", row.names = T, sep = "\t", quote =
save(AUROC_res, file = "./data/result/ML/SVM/HCC_SVM_AUROC.RData")

if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

pdf("./data/result/Figure/SFig3-G.pdf", width = 5, height = 4)
plot(fs_rfe, type = c("g", "o"))
dev.off()

ggsave("./data/result/Figure/SFig3-H.pdf", AUROC_res$roc_pl, width = 5, height = 4, dpi = 600)

```

21.7 总结

经过对差异基因进行 **REF** 特征筛选，成功识别出一组与特定生物学现象或条件紧密相关的基因子集。接着，利用支持向量机构建预测模型。通过这一详尽的模型训练和评估过程，最终确定了 **41** 个特征，这些特征在预测任务中展现出了相对不错的性能（需要注意，它在 *Late* 分期预测不好）。这 **41** 个特征将被用于后续的分析，以进一步揭示这些差异基因在生物学过程中的作用机制和潜在价值。

⚠ 警告

相比其他两种机器学习方法（见章节 [十九](#) 和章节 [二十](#)）得到的预测模型，**REF+SVM** 的预测模型在 *Late* 分期样本预测效果相对不佳。

系统信息

```

sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

```

```
Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets  utils      methods    base

other attached packages:
[1] MLmetrics_1.1.3      Hmisc_5.1-2          pROC_1.18.5
[4] mlbench_2.1-5        e1071_1.7-14       caret_6.0-94
[7] lattice_0.22-6       data.table_1.15.4 Biobase_2.62.0
[10] BiocGenerics_0.48.1 lubridate_1.9.3 forcats_1.0.0
[13] stringr_1.5.1       dplyr_1.1.4        purrr_1.0.2
[16] readr_2.1.5         tidyrr_1.3.1      tibble_3.2.1
[19] ggplot2_3.5.1       tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] tidyselect_1.2.1     timeDate_4032.109 fastmap_1.1.1
[4] digest_0.6.35       rpart_4.1.23       timechange_0.3.0
[7] lifecycle_1.0.4      cluster_2.1.6     survival_3.7-0
[10] magrittr_2.0.3      compiler_4.3.3   rlang_1.1.3
[13] tools_4.3.3        utf8_1.2.4       yaml_2.3.8
[16] knitr_1.46          htmlwidgets_1.6.4 plyr_1.8.9
[19] foreign_0.8-86     withr_3.0.0       nnet_7.3-19
[22] grid_4.3.3          stats4_4.3.3     fansi_1.0.6
[25] colorspace_2.1-0    future_1.33.2    globals_0.16.3
[28] scales_1.3.0        iterators_1.0.14 MASS_7.3-60.0.1
[31] cli_3.6.2           rmarkdown_2.26   generics_0.1.3
[34] rstudioapi_0.16.0   future.apply_1.11.2 reshape2_1.4.4
[37] tzdb_0.4.0           proxy_0.4-27    splines_4.3.3
[40] parallel_4.3.3     BiocManager_1.30.23 base64enc_0.1-3
[43] vctrs_0.6.5          hardhat_1.3.1   Matrix_1.6-5
[46] jsonlite_1.8.8      hms_1.1.3       htmlTable_2.4.2
```

[49] `Formula_1.2-5` `listenv_0.9.1` `foreach_1.5.2`
[52] `gower_1.0.1` `recipes_1.0.10` `glue_1.7.0`
[55] `parallelly_1.37.1` `codetools_0.2-19` `stringi_1.8.4`
[58] `gttable_0.3.5` `munsell_0.5.1` `pillar_1.9.0`
[61] `htmltools_0.5.8.1` `ipred_0.9-14` `lava_1.8.0`
[64] `R6_2.5.1` `evaluate_0.23` `backports_1.4.1`
[67] `renv_1.0.0` `class_7.3-22` `Rcpp_1.0.12`
[70] `checkmate_2.3.1` `gridExtra_2.3` `nlme_3.1-164`
[73] `prodlim_2023.08.28` `xfun_0.43` `pkgconfig_2.0.3`
[76] `ModelMetrics_1.2.2.2`

第二十二章 交集特征

在数据分析和机器学习项目中，特征选择是一个至关重要的步骤，它有助于识别数据集中与目标变量最相关的特征。当通过不同的机器学习方法筛选出重要特征时，对这些特征取交集以识别核心特征，是一种有效的策略，能够确保我们专注于那些在不同模型中都表现出显著影响的特征。

接下来，我们将三种方法筛选出的重要特征进行交集计算。这意味着我们要找出在三种不同特征选择方法中都被认定为重要的那些特征。这些特征的集合就是我们所谓的“核心特征”。

22.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(ggvenn)
library(UpSetR)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

22.2 导入数据

- `LASSO_feature` 结果来自于章节 [十九](#)；
- `RF_feature` 结果来自于章节 [二十](#)；
- `SVM_feature` 结果来自于章节 [二十一](#)。

©Hua
LASSO_feature <- read.csv("./data/result/ML/LASSO/HCC_LASSO_feature.csv")
RF_feature <- read.csv("./data/result/ML/RF/HCC_RF_feature.csv")
SVM_feature <- read.csv("./data/result/ML/SVM/HCC_SVM_feature.csv")

22.3 重叠的重要特征

在通过三种不同的机器学习方法筛选特征后，取这些特征集合的交集。位于这个交集内的基因，将其视为具有显著重要性的特征，这些特征对于所研究的问题具有共同的、关键的影响。

```
feature_list <- list(  
  LASSO = LASSO_feature$FeatureID,  
  RF_Boruta = RF_feature$FeatureID,  
  SVM_RFE = SVM_feature$FeatureID  
)  
  
over_LASSO_RF <- intersect(LASSO_feature$FeatureID,  
                           RF_feature$FeatureID)  
over_SVM_RF <- intersect(SVM_feature$FeatureID,  
                           RF_feature$FeatureID)  
over_LASSO_SVM <- intersect(LASSO_feature$FeatureID,  
                           SVM_feature$FeatureID)  
  
over_gene_three <- df_int_gene %>%  
  dplyr::filter(int %in% c("LASSO|RF_Boruta|SVM_RFE"))  
  
head(over_gene_three)
```

22.4 重要特征的韦恩图

- 采用 **UpSetR**(Conway, Lex, 和 Gehlenborg 2017)R 包画交集图

```
upset_pl <- UpSetR::upset(  
  data = UpSetR::fromList(feature_list),  
  nsets = 3,  
  sets = c("LASSO", "RF_Boruta", "SVM_RFE"),  
  sets.bar.color = c("#CD534CFF", "#EFC000FF", "#0073C2FF"))
```

upset_pl

- 采用 `ggvenn`(L. Yan 和 Yan 2021)R 包画交集图

```
venn_pl <- ggvenn(
  data = feature_list,
  fill_color = c("#0073C2FF", "#EFC000FF", "#CD534cff"),
  stroke_size = 0.5,
  set_name_size = 4,
  show_percentage = FALSE)
```

venn_pl

22.5 输出结果

```
if (!dir.exists("./data/result/Biomarker/")) {
  dir.create("./data/result/Biomarker/", recursive = TRUE)
}

write.csv(over_gene_three, "./data/result/Biomarker/Biomarker_LR_RF_SVM.csv", row.names = F)

if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

pdf("./data/result/Figure/Fig5-A1.pdf", width = 7, height = 5, onefile = FALSE)
upset_pl
dev.off()

ggsave("./data/result/Figure/Fig5-A2.pdf", venn_pl, width = 5, height = 4, dpi = 600)
```

22.6 总结

采用了三种不同的机器学习算法来筛选与表型或疾病状态紧密相关的基因特征。这三种算法分别是 *LASSO+LR*、Boruta+RF 和 REF+SVM*，它们各自基于不同的统计原理和假设，以识别与目标变

量显著相关的基因。

系统信息

```
sessionInfo()
```

```
R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] grid      stats     graphics   grDevices datasets  utils      methods
[8] base

other attached packages:
[1] UpSetR_1.4.0      ggvenn_0.1.10      data.table_1.15.4 lubridate_1.9.3
[5] forcats_1.0.0     stringr_1.5.1     dplyr_1.1.4       purrrr_1.0.2
[9] readr_2.1.5       tidyverse_2.0.0    tibble_3.2.1      ggplot2_3.5.1
[13] tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] gtable_0.3.5      jsonlite_1.8.8     compiler_4.3.3
[4] BiocManager_1.30.23 renv_1.0.0      Rcpp_1.0.12
[7] tidyselect_1.2.1   gridExtra_2.3     scales_1.3.0
[10] yaml_2.3.8       fastmap_1.1.1     plyr_1.8.9
[13] R6_2.5.1        generics_0.1.3    knitr_1.46
[16] munsell_0.5.1    pillar_1.9.0     tzdb_0.4.0
[19] rlang_1.1.3      utf8_1.2.4       stringi_1.8.4
[22] xfun_0.43        timechange_0.3.0   cli_3.6.2
[25] withr_3.0.0      magrittr_2.0.3    digest_0.6.35
[28] rstudioapi_0.16.0 hms_1.1.3       lifecycle_1.0.4
```

[31] vctrs_0.6.5 evaluate_0.23 glue_1.7.0
[34] fansi_1.0.6 colorspace_2.1-0 rmarkdown_2.26
[37] tools_4.3.3 pkgconfig_2.0.3 htmltools_0.5.8.1

第二十三章 验证特征

在成功识别出核心特征之后，为了验证这些特征的有效性和可靠性，我们在发现数据集和验证数据集上进行了进一步的评估。这一步骤旨在确保这些特征在不同数据集上的表达值具有一致性，并验证它们对区分肝癌（HCC）早晚期的能力。

1. 表达值一致性评估
2. ROC 曲线评估
3. 确定最终候选核心特征
4. 结果解释与后续工作

23.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(Biobase)
library(pROC)
library(multipleROC)
library(ggdist)
library(gghalves)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

23.2 导入数据

- common_feature 结果来自于章节 [二十二](#)；
- ExpressionSet 来自于章节 [九](#)。

```
common_feature <- read.csv("./data/result/Biomarker/Biomarker_LR_RF_SVM.csv")
```

```
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

```
ExprSet_GSE14520 <- readRDS("./data/result/ExpSetObject/GSE14520_ExpSet_counts.RDS")
```

23.3 函数

- get_raincloud: 组间表达值的云雨图，使用 *ggdist*([Kay 2023](#)) 和 *gghalves*([Tiedemann 2022](#))R 包；
- get_ROC: 特征的 ROC 曲线，使用 *multipleROC*([Moon 2019](#))R 包。

```
get_raincloud <- function(
  dat,
  group,
  group_names = grp_names,
  group_colors = grp_colors,
  measures,
  angle = 30) {

  MergeData <- metadata %>%
    dplyr::select(SampleID, Group) %>%
    dplyr::inner_join(profile %>%
      tibble::rownames_to_column("SampleID"),
      by = "SampleID") %>%
    tibble::column_to_rownames("SampleID") %>%
    dplyr::mutate(Group = factor(Group, levels = grp_names))

  # group for plot x-label
  dat_cln2 <- MergeData
  colnames(dat_cln2)[which(colnames(dat_cln2) == group)] <- "Group_new"

  if (group_names[1] == "all") {
```

```

tempdata <- dat_cln2
} else {
  tempdata <- dat_cln2 %>%
    dplyr::filter(Group_new %in% group_names)
}
tempdata$Group_new <- factor(tempdata$Group_new, levels = group_names)

if (length(measures) > 1) {
  pl <- ggplot(plotdata, aes(x = Group_new, y = Values, fill = Group_new)) +
    ggdist::stat_halfeye(adjust = 0.5, width = 0.3,
                          .width = 0, justification = -0.3, point_colour = NA) +
    stat_boxplot(aes(color = Group_new), geom = "errorbar", width = 0.1) +
    geom_boxplot(width = 0.1, outlier.shape = NA) +
    gghalves::geom_half_point(side = "l", range_scale = 0.4, alpha = 0.5) +
    stat_summary(geom = "crossbar", width = 0.08, fatten = 0, color = "white",
                 fun.data = function(x){c(y = median(x), ymin = median(x), ymax = median(x)))} +
    labs(x = "", y = "Value") +
    scale_fill_manual(values = group_colors) +
    scale_color_manual(values = group_colors) +
    guides(fill = "none", color = "none") +
    scale_y_continuous(expand = expansion(mult = c(0.1, 0.1))) +
    ggpubr::stat_compare_means(method = "wilcox.test",
                               comparisons = cmp) +
    facet_wrap(.~ Index, scales = "free", nrow = 2) +
    theme_bw() +
    theme(axis.title.y = element_text(size = 11, face = "bold"),
          axis.text.y = element_text(size = 10),
          axis.text.x = element_text(size = 9, hjust = .5, vjust = .5, angle = angle),
          strip.text = element_text(size = 10, face = "bold", color = "black"),
          text = element_text(family = "serif"))
} else {
  pl <- ggplot(plotdata, aes(x = Group_new, y = Values, fill = Group_new)) +
    ggdist::stat_halfeye(adjust = 0.5, width = 0.3,
                          .width = 0, justification = -0.3, point_colour = NA) +
    stat_boxplot(aes(color = Group_new), geom = "errorbar", width = 0.1) +
    geom_boxplot(width = 0.1, outlier.shape = NA) +
    gghalves::geom_half_point(side = "l", range_scale = 0.4, alpha = 0.5) +
    stat_summary(geom = "crossbar", width = 0.08, fatten = 0, color = "white",
                 fun.data = function(x){c(y = median(x), ymin = median(x), ymax = median(x)))} +
    labs(x = "", y = measures) +

```

```
scale_fill_manual(values = group_colors) +
  scale_color_manual(values = group_colors) +
  guides(fill = "none", color = "none") +
  scale_y_continuous(expand = expansion(mult = c(0.1, 0.1))) +
  ggpubr::stat_compare_means(method = "wilcox.test",
    comparisons = cmp) +
  theme_bw() +
  theme(axis.title.y = element_text(size = 11, face = "bold"),
    axis.text.y = element_text(size = 10),
    axis.text.x = element_text(size = 9, hjust = .5, vjust = .5, angle = angle),
    strip.text = element_text(size = 10, face = "bold", color = "black"),
    text = element_text(family = "serif"))
}

return(pl)
}

get_ROC <- function(
  dat,
  group,
  index,
  type = c(1, 2, 3)) {

  if (type == 1) {
    print(with(dat_cln, roc(Stage ~ Feature)))
  } else if (type == 2) {
    pROC_obj <- roc(
      dat_cln$Stage,
      dat_cln$Feature,
      smoothed = TRUE,
      # arguments for ci
      ci = TRUE,
      ci.alpha = 0.9,
      stratified = FALSE,
      # arguments for plot
      plot = TRUE,
      auc.polygon = TRUE,
      max.auc.polygon = TRUE,
      grid = TRUE,
      print.auc = TRUE,
```

```

show.thres = TRUE)

sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type = "shape", col = "lightblue")
plot(sens.ci, type = "bars")

} else if (type == 3) {
multipleROC(Stage ~ Feature, data = dat_cln)
} else if (type == 4) {

ROC_fun <- function (formula, data, plot = TRUE, threshold) {
  fit = glm(formula, data = data, family = "binomial")
  fit = glm(Stage ~ Feature, data = dat_cln, family = "binomial")
  df = calSens(fit$fitted.values, fit$y)
  no = which.max(df$sum)
  cutpoint = threshold #df$x[no]
  sens = paste("Sens:", sprintf("%03.1f", df[no, ]$sens *
    100), "%\n", "Spec:", sprintf("%03.1f", df[no, ]$spec *
    100), "%\n", "PPV:", sprintf("%03.1f", df[no, ]$ppv *
    100), "%\n", "NPV:", sprintf("%03.1f", df[no, ]$npv *
    100), "%\n", sep = ""))
  auc = simpleAUC(df)
  cutoff = fit$model[which(fit$fitted == cutpoint), ][-1]
  result = list(fit = fit, df = df, cutpoint = cutpoint, sens = sens,
    auc = auc, cutoff = cutoff)
  class(result) = "multipleROC"
  if (plot)
    print(plot(result))
  invisible(result)
}
ROC_fun(Stage ~ Feature, data = dat_cln, threshold = cutoff)
}
}

```

23.4 重要特征的表达

- 采用云雨图展示组间差异结果（发现数据集）

```

rain_dis_pl <- get_raincloud(
  dat = ExprSet,

```

```
group = "Group",
group_names = grp_names,
group_colors = grp_colors,
measures = common_feature$gene,
angle = 30)
```

rain_dis_pl

结果：重要特征在发现数据集的组间表达（wilcox.test 的 p 值作为显著性水平）

1. 在癌症早期 Early 富集的基因：FTCD, CYP2C9, CNGA1, ACSL6;
2. 在癌症晚期 Late 富集的基因：SLC6A8, ANGPT2, ENO1, KCNJ15, SLC39A4, ETV1;
 - 采用云雨图展示组间差异结果（验证数据集 GSE14520）

```
rain_val_pl <- get_raincloud(
  dat = ExprSet_GSE14520,
  group = "Group",
  group_names = grp_names,
  group_colors = grp_colors,
  measures = common_feature$gene,
  angle = 30)
```

rain_val_pl

- 重要特征的表达结果总结

```
common_feature_censis_all <- data.frame(
  FeatureID = c("FTCD", "CYP2C9", "CNGA1", "SLC6A8", "ANGPT2", "ENO1",
               "ACSL6", "KCNJ15", "ETV1", "SLC39A4"),
  Enrich = c(rep("Both_Early", 3), rep("Both_Late", 3),
            rep("Non-consist", 4))
)
```

```
common_feature_censis <- data.frame(
  FeatureID = c("FTCD", "CYP2C9", "CNGA1", "SLC6A8", "ANGPT2", "ENO1"),
  Enrich = c(rep("Both_Early", 3), rep("Both_Late", 3))
)

head(common_feature_censis)
```

23.5 ROC 分析

ROC 曲线 (Receiver Operating Characteristic Curve) 来评估每个核心特征区分肝癌 (HCC) 早晚期的能力。

- 准备数据：发现和验证数据集合并元数据和表达谱

```
MergeData_dis <- pData(ExprSet) %>%
  dplyr::select(SampleID, Group) %>%
  dplyr::inner_join(exprs(ExprSet)) %>%
    as.data.frame() %>%
    t() %>%
    as.data.frame() %>%
    tibble::rownames_to_column("SampleID"),
    by = "SampleID") %>%
  tibble::column_to_rownames("SampleID") %>%
  dplyr::mutate(Group = factor(Group, levels = grp_names))

head(MergeData_dis[, 1:6])
```

```
MergeData_val <- pData(ExprSet_GSE14520) %>%
  dplyr::select(SampleID, Group) %>%
  dplyr::inner_join(exprs(ExprSet_GSE14520)) %>%
    as.data.frame() %>%
    t() %>%
    as.data.frame() %>%
    tibble::rownames_to_column("SampleID"),
    by = "SampleID") %>%
  tibble::column_to_rownames("SampleID") %>%
  dplyr::mutate(Group = factor(Group, levels = grp_names))

head(MergeData_val[, 1:6])
```

- ROC 曲线展示区分癌症分期能力（发现数据集）

```
for (gene in common_feature_consis$FeatureID) {

  temp_roc_pl <- get_ROC(
    dat = MergeData_dis,
```

```

    group = "Group",
    index = gene,
    type = 3)
}

```

- ROC 曲线展示区分癌症分期能力（验证数据集 *GSE14520*）

```

for (gene in common_feature_consist$FeatureID) {
  temp_roc_pl <- get_ROC(
    dat = MergeData_val,
    group = "Group",
    index = gene,
    type = 3)
}

```

23.6 汇总筛选结果

通过表达值一致性和 AUC 两个指标，我们确定了最终的候选核心特征，它们便是 **SLC6A8** 和 **ANGPT2**，但 **SLC6A8** 的 AUC 在验证数据集更高一些。现在我们汇总这两个核心特征的表达值一致性和 AUC 结果。

- 采用云雨图展示组间差异结果

```

rain_dis_SLC6A8 <- get_raincloud(
  dat = ExprSet,
  group = "Group",
  group_names = grp_names,
  group_colors = grp_colors,
  measures = "SLC6A8",
  angle = 30)

```

`rain_dis_SLC6A8`

```

rain_val_SLC6A8 <- get_raincloud(
  dat = ExprSet_GSE14520,
  group = "Group",
  group_names = grp_names,
  group_colors = grp_colors,

```

©Hua
measures = "SLC6A8",
angle = 30)

rain_val_SLC6A8

结果：核心特征 **SLC6A8** 在数据集的组间表达

- ROC 曲线展示区分癌症分期能力

```
dis_roc_SLC6A8 <- get_ROC(  
  dat = MergeData_dis,  
  group = "Group",  
  index = "SLC6A8",  
  type = 3)
```

```
val_roc_SLC6A8 <- get_ROC(  
  dat = MergeData_val,  
  group = "Group",  
  index = "SLC6A8",  
  type = 3)
```

23.7 输出结果

```
if (!dir.exists("./data/result/Biomarker/")) {  
  dir.create("./data/result/Biomarker/", recursive = TRUE)  
}  
  
write.csv(common_feature_consist_all, "./data/result/Biomarker/Biomarker_LR_RF_SVM_validation.csv", row.names = FALSE)  
  
if (!dir.exists("./data/result/Figure/")) {  
  dir.create("./data/result/Figure/", recursive = TRUE)  
}  
  
ggsave("./data/result/Figure/Fig5-B.pdf", rain_dis_SLC6A8, width = 5, height = 4, dpi = 600)  
ggsave("./data/result/Figure/Fig5-C.pdf", rain_val_SLC6A8, width = 5, height = 4, dpi = 600)  
  
pdf("./data/result/Figure/Fig5-D.pdf", width = 5, height = 4)  
dis_roc_SLC6A8 <- get_ROC(  
  dat = MergeData_val,  
  group = "Group",  
  index = "SLC6A8",  
  type = 3)
```

©Hua

```
dat = MergeData_dis,
group = "Group",
index = "SLC6A8",
type = 3)

dev.off()

pdf("./data/result/Figure/Fig5-E.pdf", width = 5, height = 4)
val_roc_SLC6A8 <- get_ROC(
  dat = MergeData_val,
  group = "Group",
  index = "SLC6A8",
  type = 3)
dev.off()

ggsave("./data/result/Figure/SFig4-A.pdf", rain_dis_pl, width = 9, height = 7, dpi = 600)
ggsave("./data/result/Figure/SFig4-B.pdf", rain_val_pl, width = 9, height = 7, dpi = 600)

for (gene in common_feature_consis$FeatureID) {

  filename <- paste0("./data/result/Figure/", "SFig4-C-discovery-", gene, ".pdf")
  pdf(filename, width = 5, height = 4)
  temp_roc_pl_dis <- get_ROC(
    dat = MergeData_dis,
    group = "Group",
    index = gene,
    type = 3)
  dev.off()
}

for (gene in common_feature_consis$FeatureID) {

  filename <- paste0("./data/result/Figure/", "SFig4-D-validation-", gene, ".pdf")
  pdf(filename, width = 5, height = 4)
  temp_roc_pl_val <- get_ROC(
    dat = MergeData_val,
    group = "Group",
    index = gene,
    type = 3)
  dev.off()
}
```

23.8 总结

在深入分析数据集的过程中，我们特别关注于识别那些对区分样本早晚期状态具有重要意义的特征。通过对发现数据集和验证数据集上的关键特征进行细致比较，我们评估了这些特征的表达一致性和其基于受试者工作特性曲线（ROC）的区分能力。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets   utils      methods    base

other attached packages:
[1] gghalves_0.1.4      ggdist_3.3.2        multipleROC_0.1.1
[4] pROC_1.18.5        Biobase_2.62.0       BiocGenerics_0.48.1
[7] data.table_1.15.4   lubridate_1.9.3    forcats_1.0.0
[10] stringr_1.5.1      dplyr_1.1.4        purrrr_1.0.2
[13] readr_2.1.5        tidyverse_2.0.0      tibble_3.2.1
[16] ggplot2_3.5.1      tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] utf8_1.2.4          generics_0.1.3      renv_1.0.0
[4] stringi_1.8.4       hms_1.1.3           digest_0.6.35
[7] magrittr_2.0.3      evaluate_0.23      grid_4.3.3
```

[10] timechange_0.3.0 fastmap_1.1.1 plyr_1.8.9
[13] jsonlite_1.8.8 BiocManager_1.30.23 fansi_1.0.6
[16] scales_1.3.0 cli_3.6.2 rlang_1.1.3
[19] munsell_0.5.1 withr_3.0.0 yaml_2.3.8
[22] tools_4.3.3 tzdb_0.4.0 colorspace_2.1-0
[25] vctrs_0.6.5 R6_2.5.1 lifecycle_1.0.4
[28] pkgconfig_2.0.3 pillar_1.9.0 gtable_0.3.5
[31] glue_1.7.0 Rcpp_1.0.12 xfun_0.43
[34] tidyselect_1.2.1 rstudioapi_0.16.0 knitr_1.46
[37] htmltools_0.5.8.1 rmarkdown_2.26 compiler_4.3.3
[40] distributional_0.4.0

第七部分

关联分析

第二十四章 关联分析

在成功获取核心特征集之后，我们计划深入地探究这些特征与免疫浸润细胞之间的关联性，这是因为免疫浸润细胞在癌症的进程中扮演着至关重要的角色。免疫浸润细胞，如 T 细胞、B 细胞、巨噬细胞等，是肿瘤微环境中的重要组成部分，它们通过直接杀伤肿瘤细胞、释放细胞因子、激活免疫应答等多种机制参与抗肿瘤免疫过程。

24.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(BioconductorManager)
library(ggpmisc)
library(ggpubr)
library(ggExtra)
library(cowplot)
library(pheatmap)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

24.2 导入数据

- `ImmuneCell` 结果来自于章节 [十七](#)；

- common_feature 结果来自于章节 二十三；
- ExpressionSet 来自于章节 九。

```
ImmuneCell <- read.csv("./data/result/ImmuneCell/HCC_ImmuneCell_ImmucellAI.csv")
common_feature <- read.csv("./data/result/Biomarker/Biomarker_LR_RF_SVM_validation.csv")
ExprSet <- readRDS("./data/result/ExpSetObject/MergeExpSet_VoomSNM_VoomSNM_LIRI-JP_TCGA-LIHC.RDS")
```

24.3 函数

- get_cor: 免疫细胞和重要特征之间的相关性 `pheatmap::pheatmap(Kolde 等 2019)` 热图；
- get_lollipop: 免疫细胞和单个重要特征之间的相关性棒棒图, `cowplot::plot_grid(Wilke 2019)` 组图；
- get_cor_scatterplot: 单个免疫细胞和单个重要特征之间的相关性散点图, `ggExtra::ggMarginal(Dean Attali 2019)` 边沿图。

```
get_cor <- function(
  object,
  genelist,
  immunescore) {

  metadata <- pData(object)
  profile <- exprs(object) %>%
    data.frame()

  Iscore <- immunescore %>%
    dplyr::select(-c(2:8)) %>%
    tibble::column_to_rownames("SampleID") %>%
    dplyr::select(-all_of(c("Effector_memory", "Gamma_delta",
                           "Central_memory")))
  Iscore <- Iscore[, colSums(Iscore) > 0]

  sid <- intersect(colnames(profile), rownames(Iscore))
  profile_final <- profile[rownames(profile) %in% genelist$FeatureID,
                           pmatch(sid, colnames(profile))]
  Iscore_final <- Iscore[pmatch(sid, rownames(Iscore)), ]

  if (!all(colnames(profile_final) == rownames(Iscore_final))) {
```

```

message("wrong order bettween profile and Iscore")
} else {
  profile_final <- profile_final[, pmatch(rownames(Iscore_final),
                                         colnames(profile_final))]
}

# calculate the association by stats::cor.test
mat_rho <- datRho
mat_signif <- as.matrix(datPva)
mat_signif[mat_signif > 0.05] <- ""
mat_signif[mat_signif > 0.01 & mat_signif != ""] <- "*"
mat_signif[mat_signif <= 0.01 & mat_signif != "" & mat_signif != "*"] <- "+"

pl <- pheatmap::pheatmap(
  mat = t(mat_rho),
  color = colorRampPalette(c("green", "black", "red"))(100), # "green", "black", "red"
  cluster_rows = T,
  cluster_cols = F,
  display_numbers = t(mat_signif),
  number_color = "white",
  fontsize_number = 15,
  fontsize_row = 8,
  fontsize_col = 8,
  fontfamily = "serif")

res <- list(Rho = datRho,
            Pvalue = datPva,
            pl = pl)

return(res)
}

get_lollipop <- function(
  dat_rho,
  dat_pva,
  biomarker) {

  plotdata <- rho %>%
    dplyr::inner_join(pval, by = "Cell") %>%

```

```

dplyr::arrange(Rho)
plotdata$Cell <- factor(plotdata$Cell, levels = unique(plotdata$Cell))

p1 <- ggplot(plotdata, aes(x = Rho, y = Cell)) +
  scale_color_manual(name = "pvalue",
                     values = c("#E69F00", "#56B4E9", "#009E73", "gray")) +
  geom_segment(aes(x = 0, y = Cell, xend = Rho, yend = Cell), size = 1) +
  geom_point(aes(size = Rho_label, color = Pval_label)) +
  labs(title = biomarker,
       x = "Spearman Correlation Coefficient", y = "") +
  guides(size = guide_legend(order = 1, title = "abs(cor")),
         color = guide_legend(order = 2, title = "pvalue")) +
  theme_bw() +
  theme(plot.title = element_text(size = 12, color = "black", hjust = 0.5, face = "bold"),
        axis.title = element_text(size = 11, color = "black", face = "bold"),
        axis.text.y = element_text(size = 10, color = "black"),
        legend.position = "right",
        text = element_text(family = "serif"))

p2 <- ggplot() +
  geom_text(plotdata, mapping = aes(x = 0, y = Cell,
                                    color = Pval_label2,
                                    label = Pval)) +
  scale_color_manual(name = "",
                     values = c("red", "black", ""))+

  theme_void() +
  guides(color = "none")

pl <- cowplot::plot_grid(
  p1, p2, ncol = 2,
  align = "hv",
  rel_widths = c(1, 0.1)
)

return(pl)
}

get_cor_scatterplot <- function(
  object,
  genelist,

```

```

    immunescore,
    biomarker_gene,
    biomarker_cell,
    group_names = grp_names,
    group_colors = grp_colors) {

mdat <- Iscore %>%
  dplyr::select(all_of(c("Group", biomarker_cell))) %>%
  tibble::rownames_to_column("SampleID") %>%
  dplyr::inner_join(profile %>%
    t() %>% as.data.frame() %>%
    dplyr::select(biomarker_gene) %>%
    tibble::rownames_to_column("SampleID"),
    by = "SampleID")
plotdata <- mdat
colnames(plotdata)[3:4] <- c("Cell", "Gene")
plotdata$Group <- factor(plotdata$Group, levels = group_names)

xlab <- paste(biomarker_gene, "expression")
ylab <- biomarker_cell

pl <- ggExtra::ggMarginal(
  pl_temp,
  type = "density",
  groupFill = TRUE)

return(pl)
}

```

24.4 重要特征与免疫细胞的相关热图

所有重要特征和免疫浸润细胞的相关性分析，解析特征和免疫细胞之间的相关性也即是特征对癌症的免疫微环境的关联情况。这种关联可以是正相关（一个变量增加时，另一个也增加），也可以是负相关（一个变量增加时，另一个减少）。相关性的知识点：

- 关系的强度和方向：相关系数的值（如皮尔逊相关系数）可以表示两个变量之间关系的强度和方向。系数的绝对值越接近 1，关系越强；接近 0 则关系越弱。正数表示正相关，负数表示负相关。
- 是否存在依赖关系：虽然相关性不等于因果性（即一个变量导致另一个变量变化），但它可以指示

两个变量之间是否存在某种依赖关系。这种依赖关系可能是直接的，也可能是通过其他变量间接的。

- 预测能力：高度相关的变量可能具有预测另一个变量的能力。例如，如果两个变量高度正相关，那么知道一个变量的值可能有助于预测另一个变量的值。
- 数据模式：相关性分析可以帮助我们识别数据中的模式或趋势，这有助于理解数据的性质和行为。

```
cell_cor <- get_cor(
  object = ExprSet,
  genelist = common_feature,
  immunescore = ImmuneCell)

cell_cor$pl
```

24.5 SLC6A8 关联图

综合先前验证结果小节 23.6 和上述相关热图结果小节 24.4，对 **SLC6A8** 单独做相关性分析和可视化进一步了解其特性。

```
SLC6A8_pl <- get_lollipop(
  dat_rho = cell_cor$Rho,
  dat_pva = cell_cor$Pvalue,
  biomarker = "SLC6A8")

SLC6A8_pl
```

24.6 SLC6A8 与特定免疫细胞

针对上述相关结果小节 24.5，为了进一步探索 **SLC6A8** 和 **Bcell**（树突状细胞）之间的具体相关情况，我们进行了散点图的可视化分析。

```
SLC6A8_Bcell <- get_cor_scatterplot(
  object = ExprSet,
  genelist = common_feature,
  immunescore = ImmuneCell,
  biomarker_gene = "SLC6A8",
  biomarker_cell = "Bcell")
```

```
SLC6A8_Bcell
```

针对上述相关结果小节 24.5，为了进一步探索 SLC6A8 和 CD8_naive（未成熟的 CD8+ T 淋巴细胞）之间的具体相关情况，我们进行了散点图的可视化分析。

```
SLC6A8_CD8_naive <- get_cor_scatterplot(
  object = ExprSet,
  genelist = common_feature,
  immunescore = ImmuneCell,
  biomarker_gene = "SLC6A8",
  biomarker_cell = "CD8_naive")
```

```
SLC6A8_CD8_naive
```

24.7 SLC6A8 与其他免疫细胞

除了上述两种免疫细胞外，SLC6A8 与其他细胞的结果可以作为附图保存。

```
IC_names <- colnames(ImmuneCell)[9:32]
IC_names <- IC_names[!IC_names %in%
  c("Effector_memory", "Gamma_delta",
    "Central_memory", "CD4_T", "Tfh",
    "Th1", "MAIT", "iTreg", "Th2",
    "Macrophage", "CD8_naive", "Bcell")]

SLC6A8_ImmuneCell_list <- list()
for (j in 1:length(IC_names)) {

  IC <- IC_names[j]
  temp_plot <- get_cor_scatterplot(
    object = ExprSet,
    genelist = common_feature,
    immunescore = ImmuneCell,
    biomarker_gene = "SLC6A8",
    biomarker_cell = IC)

  SLC6A8_ImmuneCell_list[[j]] <- temp_plot
}
```

©HuaJ

```
names(SLC6A8_ImmuneCell_list) <- IC_names
```

合并上述画图对象

```
SLC6A8_ImmuneCell_pl <- cowplot::plot_grid(
  SLC6A8_ImmuneCell_list$CD4_naive, SLC6A8_ImmuneCell_list$Cytotoxic,
  SLC6A8_ImmuneCell_list$Exhausted, SLC6A8_ImmuneCell_list$Tr1,
  SLC6A8_ImmuneCell_list$nTreg, SLC6A8_ImmuneCell_list$Th17,
  SLC6A8_ImmuneCell_list$NKT, SLC6A8_ImmuneCell_list$DC,
  SLC6A8_ImmuneCell_list$Monocyte, SLC6A8_ImmuneCell_list$NK,
  SLC6A8_ImmuneCell_list$Neutrophil, SLC6A8_ImmuneCell_list$CD8_T,
  ncol = 4, align = "hv",
  labels = LETTERS[1:length(IC_names)])
```

```
SLC6A8_ImmuneCell_pl
```

24.8 输出结果

```
if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

ggsave("./data/result/Figure/Fig6-A.pdf", cell_cor$pl, width = 8, height = 5, dpi = 600)
ggsave("./data/result/Figure/Fig6-B.pdf", SLC6A8_pl, width = 6, height = 4, dpi = 600)
ggsave("./data/result/Figure/Fig6-C.pdf", SLC6A8_Bcell, width = 6, height = 4, dpi = 600)
ggsave("./data/result/Figure/Fig6-D.pdf", SLC6A8_CD8_naive, width = 6, height = 4, dpi = 600)

ggsave("./data/result/Figure/SFig6.pdf", SLC6A8_ImmuneCell_pl, width = 11, height = 9, dpi = 600)
```

24.9 总结

经过对关键特征的细致筛选与深入分析，我们针对肿瘤微环境中的免疫浸润细胞进行了详尽的相关性研究。在此过程中，**SLC6A8** 的显著重要性得以进一步凸显，特别是在肝细胞癌（HCC）晚期的显著富集小节 23.6，为我们揭示了其在癌症进程中的潜在作用。

具体而言，我们的研究观察到 **SLC6A8** 与 **Bcell** (**B 淋巴细胞**) 和 **CD8_naive** (**未成熟的 CD8+ T 淋巴细胞**) 之间存在显著的相关性。一方面，**SLC6A8** 与 **Bcell** 呈显著正相关，这一发现表明 **SLC6A8**

可能与 **B 细胞**的活化、增殖或功能执行存在紧密联系，进而在肿瘤免疫应答中发挥关键作用。另一方面，**SLC6A8** 与 **CD8_naive** 则呈显著负相关，这一结果可能揭示了 **SLC6A8** 在调节 **CD8+ T 淋巴细胞**成熟和活化过程中的潜在抑制作用。

B 细胞和 **CD8+ T 淋巴细胞**作为免疫系统的关键组成部分，在癌症的演变和进展过程中扮演着至关重要的角色。**B 细胞**通过产生抗体参与体液免疫应答，而 **CD8+ T 淋巴细胞**则主要执行细胞毒性功能，直接杀伤被感染的细胞或肿瘤细胞。

此外，根据我们之前的免疫浸润分析小节 18.5，我们发现 **B 细胞**显著富集在 **HCC 晚期分组**，而 **CD8_naive** 则显著富集在 **HCC 早期分组**。这一发现进一步强调了 **SLC6A8** 与这两种免疫细胞之间关系的重要性，并可能为我们理解 HCC 进展的免疫机制提供新的线索。

综上所述，**SLC6A8** 与 **B 细胞**和 **CD8_naive** 之间的显著相关性，以及其在 HCC 早晚期的显著富集，为我们揭示了该基因在肿瘤免疫应答中的潜在作用，并为进一步的研究和临床应用提供了有价值的参考。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets   utils      methods    base

other attached packages:
[1] pheatmap_1.0.12      cowplot_1.1.3        ggExtra_0.10.1
[4] ggpubr_0.6.0         ggpmisc_0.5.6       ggpp_0.5.7
[7] Biobase_2.62.0       BiocGenerics_0.48.1  data.table_1.15.4
[10] lubridate_1.9.3     forcats_1.0.0        stringr_1.5.1
[13] dplyr_1.1.4          purrrr_1.0.2        readr_2.1.5
```

[16] tidyverse_2.0.0

loaded via a namespace (and not attached):

[1] gtable_0.3.5	xfun_0.43	rstatix_0.7.2
[4] lattice_0.22-6	tzdb_0.4.0	vctrs_0.6.5
[7] tools_4.3.3	generics_0.1.3	fansi_1.0.6
[10] pkgconfig_2.0.3	Matrix_1.6-5	RColorBrewer_1.1-3
[13] lifecycle_1.0.4	compiler_4.3.3	MatrixModels_0.5-3
[16] munsell_0.5.1	carData_3.0-5	SparseM_1.81
[19] httpuv_1.6.15	quantreg_5.97	htmltools_0.5.8.1
[22] yaml_2.3.8	later_1.3.2	pillar_1.9.0
[25] car_3.1-2	MASS_7.3-60.0.1	abind_1.4-5
[28] mime_0.12	tidyselect_1.2.1	digest_0.6.35
[31] stringi_1.8.4	splines_4.3.3	fastmap_1.1.1
[34] grid_4.3.3	colorspace_2.1-0	cli_3.6.2
[37] magrittr_2.0.3	survival_3.7-0	utf8_1.2.4
[40] broom_1.0.5	withr_3.0.0	promises_1.3.0
[43] scales_1.3.0	backports_1.4.1	timechange_0.3.0
[46] rmarkdown_2.26	ggsignif_0.6.4	hms_1.1.3
[49] shiny_1.8.1.1	evaluate_0.23	knitr_1.46
[52] miniUI_0.1.1.1	rlang_1.1.3	Rcpp_1.0.12
[55] xtable_1.8-4	glue_1.7.0	polynom_1.4-1
[58] BiocManager_1.30.23	renv_1.0.0	rstudioapi_0.16.0
[61] jsonlite_1.8.8	R6_2.5.1	

第八部分

单细胞分析

第二十五章 单细胞数据处理

在生物信息学和数据分析领域，对公开发表的单细胞表达谱数据和元数据进行整合与处理是至关重要的一步。为了进行后续的分析和挖掘，这些原始数据需要被转换成适合处理的 *Seurat* 数据对象 (Hao 等 2024)。以下是对这一过程的书面化描述：

25.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(data.table)
library(Seurat)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

25.2 导入数据

- GSE149614 所有数据获取方式见小节 6.5

```
phenotype <- fread("./data/GSE149614_scRNA/GSE149614_HCC.metadata.updated.txt")
profile <- fread("./data/GSE149614_scRNA/GSE149614_HCC.scRNAseq.S71915.count.txt")
```

25.3 原始数据转换成 Seurat 对象

Seurat 对象 (Hao 等 2024) 是单细胞分析常见的数据对象, `Seurat::CreateSeuratObject` 函数可以将原始数据转换成 **Seurat 对象**。

Seurat 是一个集成了多种处理单细胞 RNA 测序数据功能的 R 包。它允许研究人员方便地对单细胞数据进行探索、预处理、分析和可视化。通过 Seurat 对象, 研究人员可以执行细胞类型分类、基因表达分析、细胞群聚等操作。Seurat 数据对象是一种高度灵活的数据结构, 主要用于存储单细胞 RNA 测序数据。

```
CreateObject <- function(
  count,
  metadata,
  proj = "GSE149614"){

  res <- Seurat::CreateSeuratObject(
    counts = count.cln,
    assay = "RNA",
    min.cells = 3,
    # min.features = 400,
    meta.data = metadata.cln,
    project = proj)
  res[["Batch"]] <- proj

  return(res)
}

Seurat_raw <- CreateObject(
  count = profile,
  metadata = phenotype,
  proj = "GSE149614")

Seurat_raw

Seurat_raw <- readRDS("./data/result/scRNA/Seurat_raw.RDS")

Seurat_raw
```

结果: **Seurat 对象**需要注意的列信息:

- *orig.ident*: 该列将包含已知的样本身份。如果在加载数据时提供了 `project` 参数的值, 它将会默认使用这个值 (通过 `Seurat_raw@meta.data$orig.ident` 访问);

- *nCount_RNA/nUMI*: 表示每个细胞中 UMI (Unique Molecular Identifier, 唯一分子标识符) 的数量。UMI 通常用于区分 PCR 扩增产生的相同序列的分子，以便更准确地估计基因的表达水平 (通过 `Seurat_raw@meta.data$nCount_RNA` 访问);
- *nFeature_RNA/nGene*: 表示每个细胞中检测到的基因数量。这里的“基因”可能指的是检测到的转录本或独特的 RNA 分子。这一数据通常用于质量控制，例如排除基因表达量过低或过高（可能是噪声或异常值）的细胞 (通过 `Seurat_raw@meta.data$nFeature_RNA` 访问)。

Seurat 对象的 count matrix

```
GetAssayData(Seurat_raw, assay = "RNA", layer = "counts") [c("ODF4", "MC4R", "UBOX5-AS1"), 1:10]
```

结果：矩阵中的`.`值代表 0 (未检测到分子)。由于 scRNA-seq 矩阵中的大多数值都是 0，Seurat 在可能的情况下使用稀疏矩阵表示法。这可以为 Drop-seq/inDrop/10x 数据节省大量内存和速度。

```
dense.size <- object.size(as.matrix(GetAssayData(Seurat_raw, assay = "RNA", layer = "counts")))
dense.size
sparse.size <- object.size(GetAssayData(Seurat_raw, assay = "RNA", layer = "counts"))
sparse.size
dense.size / sparse.size
```

25.4 数据过滤

为了更容易地识别不同的细胞类型群体，我们需要过滤数据以仅包含高质量的真实细胞。同时，我们需要识别任何失败的样本，并尝试挽救数据或将其从分析中移除，此外，还要尝试理解样本失败的原因。

数据过滤的挑战在 1) 区分质量差的细胞和复杂性较低的细胞；2) 选择合适的过滤阈值，以便保留高质量细胞而不移除生物学上相关的细胞类型。因此，在进行质量控制之前，请明确您期望样本中存在的细胞类型。例如，您是否期望样本中有低复杂性的细胞或线粒体表达水平较高的细胞？如果是这样，那么在评估数据质量时，我们需要考虑这种生物学特性。

25.4.1 过滤指标

- *number of genes per UMI for each cell*: 只需要取每个细胞检测到的基因数量的以 10 为底的对数，以及每个细胞 UMI 数量的以 10 为底的对数，然后将基因数量的对数值除以 UMI 数量的对数值。

```
Seurat_raw$log10GenesPerUMI
```

- *Mitochondrial Ratio*: 映射到线粒体基因的转录本比例

```
Seurat_raw$mitoRatio <- PercentageFeatureSet(object = Seurat_raw, pattern = "^\u00d7T-", assay = "RNA")
Seurat_raw$mitoRatio <- Seurat_raw@meta.data$mitoRatio / 100
```

- 使用分布图可视化过滤指标在样本或者细胞等分布

```
Seurat_raw@meta.data %>%
  ggplot(aes(x = sample, fill = sample)) +
  geom_bar() +
  geom_hline(yintercept = c(500, 1800)) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

Seurat_raw@meta.data %>%
  ggplot(aes(x = nCount_RNA, color = sample, fill = sample)) +
  geom_density(alpha = 0.2) +
  scale_x_log10() +
  geom_vline(xintercept = 500) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

Seurat_raw@meta.data %>%
  ggplot(aes(x = nFeature_RNA, color = sample, fill = sample)) +
  geom_density(alpha = 0.2) +
  scale_x_log10() +
  geom_vline(xintercept = 300) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

Seurat_raw@meta.data %>%
  ggplot(aes(x = log10GenesPerUMI, color = sample, fill = sample)) +
  geom_density(alpha = 0.2) +
  scale_x_log10() +
  geom_vline(xintercept = 0.8) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```

```
Seurat_raw@meta.data %>%
  ggplot(aes(x = mitoRatio, color = sample, fill= sample)) +
  geom_density(alpha = 0.2) +
  scale_x_log10() +
  geom_vline(xintercept = 0.2) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```

结果：从每个过滤指标的分布图确定最近过滤阈值。

```
VlnPlot(Seurat_raw,
  features = c("nCount_RNA", "nFeature_RNA",
              "log10GenesPerUMI", "mitoRatio"),
  pt.size = 0, ncol = 2)
```

25.4.2 过滤处理

根据上述过滤指标的阈值，过滤数据。以下是过滤的原因：

过滤不符合要求的细胞或基因

```
Seurat_filter <- subset(
  x = Seurat_raw,
  subset = (nCount_RNA > 500) & (nCount_RNA < 150000) &
  (nFeature_RNA > 200) & (nFeature_RNA < 7500) &
  (log10GenesPerUMI > 0.8) &
  (mitoRatio < 0.2))
```

```
Seurat_filter
```

结果：有 2497 个细胞不符合上述过滤指标阈值要求被丢弃。

```
VlnPlot(Seurat_filter,
  features = c("nCount_RNA", "nFeature_RNA",
              "log10GenesPerUMI", "mitoRatio"),
  pt.size = 0, ncol = 2)
```

25.5 输出结果

```

if (!dir.exists("./data/result/scRNA/")) {
  dir.create("./data/result/scRNA/", recursive = TRUE)
}

saveRDS(Seurat_raw, file = "./data/result/scRNA/Seurat_raw.RDS", compress = TRUE)
saveRDS(Seurat_filter, file = "./data/result/scRNA/Seurat_filter.RDS", compress = TRUE)

```

25.6 总结

整个过程涉及数据获取、数据转换、数据过滤和数据保存等多个步骤，是生物信息学和数据分析中不可或缺的一部分。通过这一过程，我们可以从公开发表的单细胞表达谱数据中提取有价值的信息，为后续研究核心特征如 **SLC6A8** 等在癌症早晚期的细胞水平表达研究提供有力的支持。接下来需要对该数据进行数据标准化等操作。

系统信息

```

sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets   utils      methods    base

other attached packages:
[1] Seurat_5.0.3        SeuratObject_5.0.2  sp_2.1-4       data.table_1.15.4
[5] lubridate_1.9.3    forcats_1.0.0      stringr_1.5.1    dplyr_1.1.4
[9] purrrr_1.0.2       readr_2.1.5      tidyverse_1.3.1  tibble_3.2.1

```

[13] ggplot2_3.5.1 tidyverse_2.0.0

loaded via a namespace (and not attached):

[1] deldir_2.0-4	pbapply_1.7-2	gridExtra_2.3
[4] rlang_1.1.3	magrittr_2.0.3	RcppAnnoy_0.0.22
[7] spatstat.geom_3.2-9	matrixStats_1.3.0	ggridges_0.5.6
[10] compiler_4.3.3	reshape2_1.4.4	png_0.1-8
[13] vctrs_0.6.5	pkgconfig_2.0.3	fastmap_1.1.1
[16] utf8_1.2.4	promises_1.3.0	rmarkdown_2.26
[19] tzdb_0.4.0	xfun_0.43	jsonlite_1.8.8
[22] goftest_1.2-3	later_1.3.2	spatstat.utils_3.0-4
[25] irlba_2.3.5.1	parallel_4.3.3	cluster_2.1.6
[28] R6_2.5.1	ica_1.0-3	spatstat.data_3.0-4
[31] stringi_1.8.4	RColorBrewer_1.1-3	reticulate_1.37.0
[34] parallelly_1.37.1	scattermore_1.2	lmtest_0.9-40
[37] Rcpp_1.0.12	knitr_1.46	tensor_1.5
[40] future.apply_1.11.2	zoo_1.8-12	sctransform_0.4.1
[43] httpuv_1.6.15	Matrix_1.6-5	splines_4.3.3
[46] igraph_2.0.3	timechange_0.3.0	tidyselect_1.2.1
[49] abind_1.4-5	rstudioapi_0.16.0	yaml_2.3.8
[52] spatstat.random_3.2-3	spatstat.explore_3.2-7	codetools_0.2-19
[55] miniUI_0.1.1.1	listenv_0.9.1	plyr_1.8.9
[58] lattice_0.22-6	shiny_1.8.1.1	withr_3.0.0
[61] ROCR_1.0-11	evaluate_0.23	Rtsne_0.17
[64] future_1.33.2	fastDummies_1.7.3	survival_3.7-0
[67] polyclip_1.10-6	fitdistrplus_1.1-11	pillar_1.9.0
[70] BiocManager_1.30.23	KernSmooth_2.23-22	renv_1.0.0
[73] plotly_4.10.4	generics_0.1.3	RcppHNSW_0.6.0
[76] hms_1.1.3	munsell_0.5.1	scales_1.3.0
[79] globals_0.16.3	xtable_1.8-4	glue_1.7.0
[82] lazyeval_0.2.2	tools_4.3.3	RSpectra_0.16-1
[85] RANN_2.6.1	leiden_0.4.3.1	dotCall64_1.1-1
[88] cowplot_1.1.3	grid_4.3.3	colorspace_2.1-0
[91] nlme_3.1-164	patchwork_1.2.0	cli_3.6.2
[94] spatstat.sparse_3.0-3	spam_2.10-0	fansi_1.0.6
[97] viridisLite_0.4.2	uwot_0.2.2	gttable_0.3.5
[100] digest_0.6.35	progressr_0.14.0	ggrepel_0.9.5
[103] htmlwidgets_1.6.4	htmltools_0.5.8.1	lifecycle_1.0.4
[106] httr_1.4.7	mime_0.12	MASS_7.3-60.0.1

第二十六章 单细胞数据标准化

scRNA-seq 的标准化是一个重要的预处理步骤，目的是消除技术变异（比如测序深度和基因长度等因素），使基因表达和/或样本之间的比较更加可靠。标准化方法可以是简单的全局缩放和统一转换，也可以是更复杂的基于每个基因的校正。重要的是要认识到不同方法可能适用于不同的数据集和研究目标，并且了解这些方法的假设和局限性对于正确解释结果至关重要。

26.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(Seurat)
library(cowplot)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

26.2 导入数据

- `seurat_obj` 数据来自于章节 [二十五](#)

```
seurat_obj <- readRDS("./data/result/scRNA/Seurat_filter.RDS")

seurat_obj
```

26.3 消除测序深度影响

首先了解以下两个概念：

```
seurat_norm <- NormalizeData(
  object = seurat_obj,
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  verbose = FALSE)

seurat_norm
```

26.4 评估细胞周期的影响

细胞周期 (cell cycle) 是指细胞从一次分裂完成开始到下一次分裂结束所经历的全过程，包括间期和分裂期。评估细胞周期有助于了解细胞所处的具体阶段（如 G1 期、S 期、G2 期和 M 期），这对于理解细胞的功能状态、代谢活动和生长特性至关重要。

```
load("./data/result/scRNA/Homo_sapiens_cycle.rda")

seurat_phase@meta.data %>%
  ggplot(aes(x = S.Score, y = G2M.Score)) +
  geom_point(aes(color = Phase)) +
  theme_minimal()
```

26.5 识别高度可变的特征

可以使用 Seurat 函数 `FindVariableFeatures` 选择 `vst` (*Variance Stabilizing Transformation*, 方差稳定变换) 方法（筛选标准是通过基因在不同细胞间的平均表达量和变异程度），并设置高变异基因数量为 2000。

```
seurat_var <- FindVariableFeatures(
  object = seurat_phase,
  selection.method = "vst",
  nfeatures = 2000,
  verbose = FALSE)

LabelPoints(plot = VariableFeaturePlot(seurat_var),
```

```
points = head(VariableFeatures(seurat_var), 10),  
repel = TRUE)
```

26.6 缩放数据

```
seurat_scale <- ScaleData(  
  object = seurat_var,  
  verbose = FALSE)  
  
seurat_scale
```

26.7 降维聚类

- 查看细胞周期的影响

```
seurat_pca <- RunPCA(  
  object = seurat_scale,  
  features = VariableFeatures(object = seurat_scale),  
  verbose = FALSE)  
  
DimPlot(seurat_pca,  
        reduction = "pca",  
        group.by = "Phase")
```

26.8 输出结果

```
if (!dir.exists("./data/result/scRNA/")) {  
  dir.create("./data/result/scRNA/", recursive = TRUE)  
}  
  
saveRDS(seurat_pca, file = "./data/result/scRNA/Seurat_pca.RDS", compress = TRUE)
```

26.9 总结

我们采取了严谨的数据预处理流程。这一流程的核心在于通过数据标准化来降低技术变异，并妥善处理生物学变异，以确保分析结果的准确性和可靠性。

系统信息

```
sessionInfo()
```

```
R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2
```

```
Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version
```

```
locale:
```

```
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: Asia/Shanghai
tzcode source: internal
```

```
attached base packages:
```

```
[1] stats      graphics   grDevices datasets  utils      methods    base
```

```
other attached packages:
```

```
[1] cowplot_1.1.3      Seurat_5.0.3       SeuratObject_5.0.2  sp_2.1-4
[5] lubridate_1.9.3    forcats_1.0.0      stringr_1.5.1      dplyr_1.1.4
[9] purrrr_1.0.2       readr_2.1.5       tidyverse_2.0.0
[13] ggplot2_3.5.1      tidyverse_2.0.0
```

```
loaded via a namespace (and not attached):
```

```
[1] deldir_2.0-4        pbapply_1.7-2      gridExtra_2.3
[4] rlang_1.1.3         magrittr_2.0.3     RcppAnnoy_0.0.22
[7] spatstat.geom_3.2-9 matrixStats_1.3.0   ggridges_0.5.6
[10] compiler_4.3.3     reshape2_1.4.4     png_0.1-8
[13] vctrs_0.6.5        pkgconfig_2.0.3   fastmap_1.1.1
[16] utf8_1.2.4         promises_1.3.0    rmarkdown_2.26
[19] tzdb_0.4.0          xfun_0.43        jsonlite_1.8.8
[22] goftest_1.2-3      later_1.3.2      spatstat.utils_3.0-4
```

[25] irlba_2.3.5.1 parallel_4.3.3 cluster_2.1.6
[28] R6_2.5.1 ica_1.0-3 spatstat.data_3.0-4
[31] stringi_1.8.4 RColorBrewer_1.1-3 reticulate_1.37.0
[34] parallelly_1.37.1 scattermore_1.2 lmtest_0.9-40
[37] Rcpp_1.0.12 knitr_1.46 tensor_1.5
[40] future.apply_1.11.2 zoo_1.8-12 sctransform_0.4.1
[43] httpuv_1.6.15 Matrix_1.6-5 splines_4.3.3
[46] igraph_2.0.3 timechange_0.3.0 tidyselect_1.2.1
[49] abind_1.4-5 rstudioapi_0.16.0 yaml_2.3.8
[52] spatstat.random_3.2-3 spatstat.explore_3.2-7 codetools_0.2-19
[55] miniUI_0.1.1.1 listenv_0.9.1 plyr_1.8.9
[58] lattice_0.22-6 shiny_1.8.1.1 withr_3.0.0
[61] ROCR_1.0-11 evaluate_0.23 Rtsne_0.17
[64] future_1.33.2 fastDummies_1.7.3 survival_3.7-0
[67] polyclip_1.10-6 fitdistrplus_1.1-11 pillar_1.9.0
[70] BiocManager_1.30.23 KernSmooth_2.23-22 renv_1.0.0
[73] plotly_4.10.4 generics_0.1.3 RcppHNSW_0.6.0
[76] hms_1.1.3 munsell_0.5.1 scales_1.3.0
[79] globals_0.16.3 xtable_1.8-4 glue_1.7.0
[82] lazyeval_0.2.2 tools_4.3.3 data.table_1.15.4
[85] RSpectra_0.16-1 RANN_2.6.1 leiden_0.4.3.1
[88] dotCall64_1.1-1 grid_4.3.3 colorspace_2.1-0
[91] nlme_3.1-164 patchwork_1.2.0 cli_3.6.2
[94] spatstat.sparse_3.0-3 spam_2.10-0 fansi_1.0.6
[97] viridisLite_0.4.2 uwot_0.2.2 gtable_0.3.5
[100] digest_0.6.35 progressr_0.14.0 ggrepel_0.9.5
[103] htmlwidgets_1.6.4 htmltools_0.5.8.1 lifecycle_1.0.4
[106] httr_1.4.7 mime_0.12 MASS_7.3-60.0.1

第二十七章 单细胞聚类分析

单细胞聚类分析的目的在于从单细胞转录组数据中识别和分类细胞类型，以揭示细胞种类之间的差异以及它们在不同生理或病理状态下的变化。通过聚类算法将相似基因表达模式的细胞聚集在一起，可以识别出不同的细胞类型，从而有助于理解细胞组成和功能，并为研究细胞发展、疾病发生机制等提供重要线索。

27.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(Seurat)
library(cowplot)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")
grp_shapes <- c(15, 16)
```

27.2 导入数据

- `seurat_obj` 数据来自于章节 [二十六](#)，已经做过 PCA 降纬处理。

```
seurat_obj <- readRDS("./data/result/scRNA/Seurat_pca.RDS")

seurat_obj
```

27.3 确定 PC 纬度

在单细胞聚类分析过程中，对细胞数据进行 PCA（主成分分析）以提取主成分（PC）是常见的预处理步骤。由于 PC 的选取直接关系到聚类结果的质量，因此选择恰当的 PC 数量至关重要。理想的 PC 数量应能充分保留原始数据中的关键信息，同时避免引入过多的冗余信息，以确保聚类结果的有效性和准确性。

```
# seurat_pca <- JackStraw(seurat_pca, num.replicate = 100)
# seurat_pca <- ScoreJackStraw(seurat_pca, dims = 1:30)
# JackStrawPlot(seurat_pca, dims = 1:20)

ElbowPlot(seurat_obj, ndims = 50, reduction = "pca")
```

结果：可以观察到 *PC10 - 20* 周围是“肘部”，这表明大部分真实信号是在前 20 个 PC 捕获的，为了保留更多信息，我们选择 40 个 PC。

27.4 寻找邻居

Seurat 使用基于图的聚类方法，该方法采用了 K-最近邻方法，然后尝试将这个图划分为高度相互连接的“簇”。基于 PCA 空间中的欧氏距离来构建一个 K-最近邻（K-nearest neighbor，KNN）图。

```
seurat_obj <- FindNeighbors(
  object = seurat_obj,
  dims = 1:40,
  verbose = FALSE)

seurat_obj
```

27.5 寻找聚类

Seurat 将以优化标准模块化函数为目标，迭代地将细胞组合在一起。我们将使用 `FindClusters()` 函数来执行基于图的聚类。分辨率是一个重要的参数，它设定了下游聚类的“粒度”，并且需要针对每个独立实验进行优化。对于 3000 - 5000 个细胞的数据集，分辨率设置在 0.4 - 1.4 之间通常会产生良好的聚类效果。

```
seurat_cluster <- FindClusters(
  object = seurat_obj,
  resolution = seq(0.4, 1.4, 0.2),
```

```
verbose = FALSE)
```

```
seurat_cluster
```

结果：不同分辨率的聚类结果可以通过 `seurat_obj@meta.data` 查看，会看到以 `RNA_snn_res` 开头（如果是整合多个来源数据的数据集可能是 `integrated_snn_res` 开头）。

- 选择聚类分辨率，增加分辨率值会导致聚类数量增加，这通常对于更大的数据集是必要的。我们这里选择最大的 **1.4** 分辨率。

```
Idents(object = seurat_cluster) <- "RNA_snn_res.1.4"
```

```
seurat_cluster
```

27.6 可视化聚类

Seurat 提供了几种非线性降维技术，如 tSNE 和 UMAP，以可视化和探索这些数据集。这些算法的目标是学习数据的底层流形，以便在低维空间中将相似的细胞放置在一起。基于图的聚类中确定的细胞应该在这些降维图上共定位。作为 UMAP 和 tSNE 的输入，我们建议使用与聚类分析相同的主成分（PCs）作为输入。

```
seurat_cluster <- RunUMAP(
  object = seurat_cluster,
  reduction = "pca",
  dims = 1:40,
  verbose = FALSE)
```

```
seurat_cluster <- RunTSNE(
  object = seurat_cluster,
  reduction = "pca",
  dims = 1:40,
  check_duplicates = FALSE,
  verbose = FALSE)
```

```
DimPlot(seurat_cluster,
  reduction = "umap",
  label = TRUE,
  label.size = 6) + NoLegend()
```

```
DimPlot(seurat_cluster,
        reduction = "tsne",
        label = TRUE,
        label.size = 6) + NoLegend()
```

结果：总计识别出 **52** 个细胞类别。可以通过观察同一个细胞聚类簇它们是否聚集在一起，若它们是分散在不同区域，说明该分辨率下聚类效果不好请重新选择分辨率，但还是没有改善可能需要重新进行数据预处理等步骤了。比如，假定 *26* 和 *40* 在某次聚类显示是一个细胞类群，但它们却是分散在很远的两端，这很大概率是聚类错误。

- 使用 `ggh4x`(van den Brand 2023) 和 `ggunchull`(Lyu 2021)R 包生成符合出版社的图片。

```
library(ggh4x)
library(ggunchull)

umap_pl <- ggplot(umap_data, aes(x = umap_1, y = umap_2, fill = ident, color = ident)) +
  geom_point(size = 0.5, show.legend = FALSE) +
  guides(color = "none", x = umap_axis, y = umap_axis) +
  scale_x_continuous(breaks = NULL) +
  scale_y_continuous(breaks = NULL) +
  theme(aspect.ratio = 1,
        panel.background = element_blank(),
        panel.grid = element_blank(),
        axis.line = element_line(arrows = arrow(type = "closed")),
        axis.title = element_text(hjust = 0.05, face = "italic"))

umap_pl
```

27.7 输出结果

```
if (!dir.exists("./data/result/scRNA/")) {
  dir.create("./data/result/scRNA/", recursive = TRUE)
}

saveRDS(seurat_cluster, file = "./data/result/scRNA/Seurat_cluster.RDS", compress = TRUE)
```

27.8 总结

我们从降维后的数据中筛选出了能够代表足够多原始信息的主成分。这些被筛选出的主成分不仅保留了原始数据的关键特征，而且减少了数据的冗余和噪声，为后续的聚类分析提供了更加清晰和准确的数据

基础。

系统信息

```
sessionInfo()
```

```
R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets  utils      methods    base

other attached packages:
[1] cowplot_1.1.3     Seurat_5.0.3       SeuratObject_5.0.2  sp_2.1-4
[5] lubridate_1.9.3  forcats_1.0.0      stringr_1.5.1      dplyr_1.1.4
[9] purrrr_1.0.2      readr_2.1.5       tidyverse_2.0.0
[13] ggplot2_3.5.1

loaded via a namespace (and not attached):
[1] deldir_2.0-4        pbapply_1.7-2      gridExtra_2.3
[4] rlang_1.1.3         magrittr_2.0.3     RcppAnnoy_0.0.22
[7] spatstat.geom_3.2-9 matrixStats_1.3.0   ggridges_0.5.6
[10] compiler_4.3.3     reshape2_1.4.4     png_0.1-8
[13] vctrs_0.6.5        pkgconfig_2.0.3    fastmap_1.1.1
[16] utf8_1.2.4         promises_1.3.0     rmarkdown_2.26
[19] tzdb_0.4.0          xfun_0.43         jsonlite_1.8.8
[22] goftest_1.2-3      later_1.3.2       spatstat.utils_3.0-4
[25] irlba_2.3.5.1     parallel_4.3.3    cluster_2.1.6
[28] R6_2.5.1            ica_1.0-3         spatstat.data_3.0-4
[31] stringi_1.8.4      RColorBrewer_1.1-3 reticulate_1.37.0
```

[34] parallelly_1.37.1 scattermore_1.2 lmtest_0.9-40
[37] Rcpp_1.0.12 knitr_1.46 tensor_1.5
[40] future.apply_1.11.2 zoo_1.8-12 sctransform_0.4.1
[43] httpuv_1.6.15 Matrix_1.6-5 splines_4.3.3
[46] igraph_2.0.3 timechange_0.3.0 tidyselect_1.2.1
[49] abind_1.4-5 rstudioapi_0.16.0 yaml_2.3.8
[52] spatstat.random_3.2-3 spatstat.explore_3.2-7 codetools_0.2-19
[55] miniUI_0.1.1.1 listenv_0.9.1 plyr_1.8.9
[58] lattice_0.22-6 shiny_1.8.1.1 withr_3.0.0
[61] ROCR_1.0-11 evaluate_0.23 Rtsne_0.17
[64] future_1.33.2 fastDummies_1.7.3 survival_3.7-0
[67] polyclip_1.10-6 fitdistrplus_1.1-11 pillar_1.9.0
[70] BiocManager_1.30.23 KernSmooth_2.23-22 renv_1.0.0
[73] plotly_4.10.4 generics_0.1.3 RcppHNSW_0.6.0
[76] hms_1.1.3 munsell_0.5.1 scales_1.3.0
[79] globals_0.16.3 xtable_1.8-4 glue_1.7.0
[82] lazyeval_0.2.2 tools_4.3.3 data.table_1.15.4
[85] RSpectra_0.16-1 RANN_2.6.1 leiden_0.4.3.1
[88] dotCall64_1.1-1 grid_4.3.3 colorspace_2.1-0
[91] nlme_3.1-164 patchwork_1.2.0 cli_3.6.2
[94] spatstat.sparse_3.0-3 spam_2.10-0 fansi_1.0.6
[97] viridisLite_0.4.2 uwot_0.2.2 gtable_0.3.5
[100] digest_0.6.35 progressr_0.14.0 ggrepel_0.9.5
[103] htmlwidgets_1.6.4 htmltools_0.5.8.1 lifecycle_1.0.4
[106] httr_1.4.7 mime_0.12 MASS_7.3-60.0.1

第二十八章 单细胞细胞识别

单细胞细胞簇识别有两种方式自动识别和手动识别，可以先通过自动识别后再手动纠正细胞类群。

- 手动注释：通过查询一些常见的细胞类型的标记基因
- 自动细胞注释网站和软件：

两者相辅相成对聚类结果进行细胞注释。

28.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(Seurat)
library(cowplot)
library(pheatmap)
library(ggh4x)
library(ggunchull)
library(scCATCH)
library(SingleR)
library(UCell)
library(patchwork)
library(ggtree)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")

cell_types <- c("Hepatocyte", "T/NK",
```

```

    "Myeloid", "B",
    "Endothelial", "Fibroblast")

cell_colors <- c("#A01FF0", "#1F78B4", "#4EB29D",
                 "#DA3F4C", "#F1EE97", "#08306B")

```

28.2 导入数据

- `seurat_obj` 数据来自于章节 [二十七](#)

```

seurat_obj <- readRDS("./data/result/scRNA/Seurat_cluster.RDS")

seurat_obj

```

28.3 细胞簇的标记基因

细胞簇的标记基因是表征不同细胞簇的特异性基因，这些基因在特定细胞簇中表现出独特的表达模式，从而成为区分不同细胞簇的重要特征。

```

if (file.exists("./data/result/scRNA/All_Markers_raw.csv")) {
  all_markers_raw <- read.csv("./data/result/scRNA/All_Markers_raw.csv")
} else {
  # 找出每个细胞簇的标记物，与所有剩余的细胞进行比较，只报告阳性细胞
  all_markers_raw <- FindAllMarkers(
    object = seurat_obj,
    only.pos = TRUE,
    min.pct = 0.25,
    logfc.threshold = 0.25,
    verbose = FALSE)

  write.csv(all_markers_raw, "./data/result/scRNA/All_Markers_raw.csv", row.names = FALSE)
}

```

```

topN_markers <- all_markers_raw %>%
  dplyr::group_by(cluster) %>%
  dplyr::top_n(n = 5, wt = avg_log2FC)

```

```
# DoHeatmap(object = seurat_obj,
#           features = top10_markers$gene) +
#   NoLegend() +
#   scale_fill_gradientn(colors = c("white","grey","firebrick3"))

head(topN_markers)
```

28.4 ATC 细胞注释

生成 ATC 网站需要的输入文件。

```
topN_markers_label <- topN_markers %>%
  dplyr::group_by(cluster) %>%
  dplyr::summarise(label = paste0("Cluster", cluster, ":", paste(gene, collapse = ","))) %>%
  dplyr::distinct() %>%
  dplyr::ungroup()

# cat(paste(topN_markers_label$label, collapse = "\n"))
```

结果：在[Annotation of Cell Types, ATC](#)选择好 Human 和 liver 后，每次输入的簇数目必须小于 50。注释结果不够理想，很多 cluster 无法注释出结果，比如 cluster0 没有注释到细胞类群。

28.5 scCATCH 细胞注释

[scCATCH](#)(Shao 等 2020) 是浙江大学团队在 2020 年发表的自动注释 R 包，它提供 `findmarkergene` 和 `findcelltype` 函数用于自动注释。通过查看[数据中的组织和癌症列表](#)判断是否适合自己的数据。

```
sc_data <- GetAssayData(seurat_obj, assay = "RNA", layer = "scale.data")
sc_cluster <- FetchData(object = seurat_obj, vars = "ident")

obj <- createscCATCH(data = sc_data, cluster = as.character(sc_cluster$ident))

# find marker gene for each cluster
obj <- findmarkergene(
  object = obj,
  species = "Human",
  cluster = "All",
  marker = cellmatch,
```

```

use_method = "1",
tissue = "Liver",
cancer = "Liver Cancer",
cell_min_pct = 0.25,
logfc = 0.25,
pvalue = 0.05,
verbose = TRUE)

# find cell type for each cluster
obj <- findcelltype(obj)

```

结果：代码运行错误，报错信息 *No potential marker genes found in the matrix! You may adjust the cell clusters by applying other clustering algorithm and try again.*

28.6 SingleR 细胞注释

- SingleR 自建数据集，细胞注释。metadata 已经包含文章提供的细胞注释信息，我们直接从 seurat_obj 构建数据库文件。

```

seurat_SGR <- seurat_obj

pred_label <- SingleR(
  test = GetAssayData(object = seurat_obj,
                       assay = "RNA",
                       layer = "data"),
  ref = SGT_ref,
  labels = SGT_ref$CellType)

plotScoreHeatmap(pred_label)

seurat_SGR_label <- seurat_obj
seurat_SGR_label$singleR <- pred_label$labels
seurat_SGR_label$singleR <- factor(seurat_SGR_label$singleR,
                                      levels = cell_types)

DimPlot(seurat_SGR_label,
        reduction = "tsne",
        label = FALSE,
        group.by = "singleR",
        cols = cell_colors,

```

```
pt.size = 0.5)
```

结果：和小节 28.7 的结果还是存在部分差异，说明该方法自动注释效果还是不太行。

28.7 内部细胞注释

- seurat_obj@meta.data\$celltype 列已经包含了公开数据提供的细胞注释信息，我们可以直接使用该信息画图即可。

```
tSNE_columns <- c("celltype", "tSNE_1", "tSNE_2")
tSNE_data <- FetchData(
  object = seurat_obj,
  vars = tSNE_columns) %>%
  dplyr::mutate(celltype = factor(celltype, levels = cell_types))

tSNE_axis <- ggh4x::guide_axis_truncated(
  trunc_lower = unit(0, "npc"),
  trunc_upper = unit(3, "cm"))

tSNE_pl <- ggplot(tSNE_data, aes(x = tSNE_1, y = tSNE_2, fill = celltype, color = celltype)) +
  geom_point(size = 0.5, show.legend = FALSE) +
  guides(color = "none", fill = guide_legend(title = NULL),
         x = tSNE_axis, y = tSNE_axis) +
  scale_fill_manual(values = cell_colors) +
  scale_color_manual(values = cell_colors) +
  scale_x_continuous(breaks = NULL) +
  scale_y_continuous(breaks = NULL) +
  theme(aspect.ratio = 1,
        panel.background = element_blank(),
        panel.grid = element_blank(),
        axis.line = element_line(arrow = arrow(type = "closed")),
        axis.title = element_text(hjust = 0.05, face = "italic"))

tSNE_pl
```

结果：在 tsne 图中，同一细胞类群没有聚集在一起，这和文章提供的图表现不一致，原因是仅仅保留了 tumor 患者的细胞。

- 修改默认 identity 为细胞注释结果

```
seurat_cell <- seurat_obj

Idents(object = seurat_cell) <- "celltype"
levels(seurat_cell) <- cell_types

DimPlot(seurat_cell,
        reduction = "tsne",
        label = FALSE,
        cols = cell_colors,
        pt.size = 0.5)
```

结果：从图示数据分析中观察到，部分簇内包含了多个细胞类群，这可能是由于新方法的聚类效果未能达到如先前研究那样的高度一致性。为了更准确地描述这些簇，我们需要重新对每个簇进行命名，命名规则将依据簇内出现概率最高的细胞类群来确定。

28.8 重命名簇

```
seurat_cell_rename <- seurat_obj

table(seurat_obj$seurat_clusters, seurat_obj$celltype)

seurat_cell_rename <- RenameIdents(
  object = seurat_cell_rename, "0" = "Hepatocyte",
  "1" = "Hepatocyte", "2" = "T/NK", "3" = "Myeloid",
  "4" = "Myeloid", "5" = "Hepatocyte", "6" = "Hepatocyte",
  "7" = "Myeloid", "8" = "Myeloid", "9" = "Hepatocyte",
  "10" = "Hepatocyte", "11" = "Hepatocyte", "12" = "Endothelial",
  "13" = "T/NK", "14" = "Hepatocyte", "15" = "Fibroblast",
  "16" = "Hepatocyte", "17" = "T/NK", "18" = "Myeloid",
  "19" = "T/NK", "20" = "T/NK", "21" = "Hepatocyte",
  "22" = "Myeloid", "23" = "Fibroblast", "24" = "Myeloid",
  "25" = "T/NK", "26" = "Hepatocyte", "27" = "Hepatocyte",
  "28" = "Endothelial", "29" = "T/NK", "30" = "Myeloid",
  "31" = "Hepatocyte", "32" = "T/NK", "33" = "Endothelial",
  "34" = "B", "35" = "B", "36" = "Myeloid",
  "37" = "Fibroblast", "38" = "Myeloid", "39" = "Hepatocyte",
  "40" = "Myeloid", "41" = "T/NK", "42" = "Myeloid",
  "43" = "Myeloid", "44" = "Myeloid", "45" = "Hepatocyte",
  "46" = "Hepatocyte", "47" = "T/NK", "48" = "Endothelial")
```

```
"49" = "T/NK", "50" = "T/NK", "51" = "Myeloid")  
  
levels(seurat_cell_rename) <- cell_types  
  
DimPlot(seurat_cell_rename,  
        reduction = "tsne",  
        label = FALSE,  
        cols = cell_colors,  
        pt.size = 0.5)
```

28.9 展示分组基因表达

使用气泡图展示不同细胞类型的差异表达基因。

```
if (file.exists("./data/result/scRNA/All_Markers_celltype.csv")) {  
  all_markers_cell <- read.csv("./data/result/scRNA/All_Markers_celltype.csv")  
} else {  
  # 找出每个细胞簇的标记物，与所有剩余的细胞进行比较，只报告阳性细胞  
  # devtools::install_github('immunogenomics/presto')  
  require(presto)  
  all_markers_cell <- FindAllMarkers(  

```

```

  labs(x = "", y = "") +
  scale_color_gradientn(colors = c('#330066', '#336699', '#66CC66', '#FFCC33')) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))

ggplot(DotPlot_markers$data, aes(x = id, y = features.plot, size = pct.exp)) +
  geom_point(shape = 21, aes(fill = avg.exp.scaled), position = position_dodge(0)) +
  scale_size_continuous(range = c(1, 10)) +
  scale_fill_gradient(low = "#E54924", high = "#498EA4") +
  labs(x = "", y = "", size = "Percent Express",
       fill = "Average Expression") +
  # guides(size = guide_legend(title = "Percent Express", order = 1),
  #         fill = guide_legend(title = "Average Expression", order = 2)) +
  theme_bw() +
  theme(legend.position = "right",
        legend.box = "vertical",
        legend.margin = margin(t = 0, unit = 'cm'),
        legend.spacing = unit(0, "in"),
        axis.text.x = element_text(color = "black", size = 11, angle = 45,
                                   vjust = 0.5, hjust = 0.5),
        axis.text.y = element_text(color = "black", size = 10),
        legend.text = element_text(size = 10, color = "black"),
        legend.title = element_text(size = 10, color = "black")))

```

结果：从气泡图可以看出每个细胞类型都有自己独有的差异基因。

28.10 美化气泡图

对小节 28.9 的细胞类群的基因表达状况气泡图进行美化，体现不同的原始聚类结果。

```

cell_bubble_pl <- plot_spacer() + plot_spacer() + ggtree_plot_col +
  plot_spacer() + plot_spacer() + labels +
  plot_spacer() + plot_spacer() + plot_spacer() +
  ggtree_plot + plot_spacer() + dotplot +
  plot_spacer() + plot_spacer() + legend +
  plot_layout(ncol = 3, widths = c(0.7, -0.1, 4), heights = c(0.9, 0.1, -0.1, 4, 1)) &
  theme(text = element_text(family = "serif"))

```

cell_bubble_pl

结果：气泡图

28.11 单细胞评分

单细胞评分分析 UCell(Andreatta 和 Carmona 2021)R 包，对细胞类群进行评分。

```
markers <- list()
markers$Hepatocyte <- c("SPINK1", "COX7B2", "TFF2")
markers`T/NK` <- c("CD2", "TRAC", "CD3E")
markers$Myeloid <- c("MS4A7", "C5AR1", "FPR3")
markers$B <- c("MS4A1", "BANK1", "TNFRSF13C")
markers$Endothelial <- c("CLEC14A", "EMCN", "TM4SF18")
markers$Fibroblast <- c("COL14A1", "PLN", "MFAP4")

marker_score <- AddModuleScore_UCell(
  obj = seurat_cell_rename,
  features = markers)

FeaturePlot(object = marker_score,
            features = colnames(marker_score@meta.data) %>%
              stringr::str_subset("_UCell"),
            reduction = "tsne",
            order = T,
            ncol = 3,
            cols = viridis::viridis(256)) &
  theme(plot.title = element_text(size = 12),
        axis.text = element_text(size = 10),
        legend.title = element_text(size = 10))
```

28.12 输出结果

```
if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

pdf(file = "./data/result/Figure/Fig7-A.pdf", width = 7, height = 5)
tSNE_pl
dev.off()

if (!dir.exists("./data/result/scRNA/")) {
```

```

dir.create("./data/result/scRNA/", recursive = TRUE)
}

saveRDS(seurat_cell_rename, file = "./data/result/scRNA/Seurat_celltype.RDS", compress = TRUE)

if (!dir.exists("./data/result/Figure/")) {
  dir.create("./data/result/Figure/", recursive = TRUE)
}

ggsave("./data/result/Figure/Fig7-A.pdf", tSNE_pl, width = 7, height = 5, dpi = 600)

```

28.13 总结

在进行生物信息学或单细胞测序数据的分析中，聚类分析是识别不同细胞类型的关键步骤。一旦聚类完成，对类群进行标记基因的识别、细胞自动注释以及文献结果的整合注释，最后对注释结果进行打分评估，是评估聚类效果的重要流程。

接下来，我们针对每个聚类类群进行标记基因的识别。这些标记基因通常是在该类群中特异性高表达，而在其他类群中表达量较低的基因。通过比较不同类群间的基因表达差异，我们可以筛选出每个类群的特异性标记基因，为后续细胞类型的注释提供基础。除此之外，我们还采用了自动注释方法，但由于数据库的局限性和实验条件的差异，自动注释的结果可能并不完全准确。需要结合手动注释结果和文献验证。

最后，我们需要对注释结果进行打分评估，以验证聚类效果的好坏。

系统信息

```

sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

```

```
attached base packages:
[1] stats4      stats       graphics   grDevices datasets   utils       methods
[8] base

other attached packages:
[1] ggtree_3.10.1           patchwork_1.2.0
[3] UCell_2.6.2             SingleR_2.4.1
[5] SummarizedExperiment_1.32.0 Biobase_2.62.0
[7] GenomicRanges_1.54.1     GenomeInfoDb_1.38.8
[9] IRanges_2.36.0          S4Vectors_0.40.2
[11] BiocGenerics_0.48.1    MatrixGenerics_1.14.0
[13] matrixStats_1.3.0      scCATCH_3.2.2
[15] ggchull_1.0.1          ggh4x_0.2.8
[17] pheatmap_1.0.12        cowplot_1.1.3
[19] Seurat_5.0.3           SeuratObject_5.0.2
[21] sp_2.1-4                lubridate_1.9.3
[23]forcats_1.0.0           stringr_1.5.1
[25] dplyr_1.1.4             purrr_1.0.2
[27] readr_2.1.5             tidyverse_2.0.0
[29] tibble_3.2.1             ggplot2_3.5.1
[31]

loaded via a namespace (and not attached):
[1] RcppAnnoy_0.0.22           splines_4.3.3
[3] later_1.3.2                ggplotify_0.1.2
[5] bitops_1.0-7                R.oo_1.26.0
[7] polyclip_1.10-6            fastDummies_1.7.3
[9] lifecycle_1.0.4              globals_0.16.3
[11] lattice_0.22-6             MASS_7.3-60.0.1
[13] magrittr_2.0.3              plotly_4.10.4
[15] rmarkdown_2.26              yaml_2.3.8
[17] httpuv_1.6.15              sctransform_0.4.1
[19] spam_2.10-0                spatstat.sparse_3.0-3
[21] reticulate_1.37.0           pbapply_1.7-2
[23] RColorBrewer_1.1-3         abind_1.4-5
[25] zlibbioc_1.48.2            Rtsne_0.17
[27] R.utils_2.12.3              RCurl_1.98-1.14
[29] yulab.utils_0.1.4           GenomeInfoDbData_1.2.11
[31] ggrepel_0.9.5               irlba_2.3.5.1
```

[33] listenv_0.9.1
[35] tidytree_0.4.6
[37] RSpectra_0.16-1
[39] fitdistrplus_1.1-11
[41] DelayedMatrixStats_1.24.0
[43] codetools_0.2-19
[45] alphahull_2.5
[47] aplot_0.2.2
[49] sgeostat_1.0-27
[51] jsonlite_1.8.8
[53] progressr_0.14.0
[55] survival_3.7-0
[57] tools_4.3.3
[59] treeio_1.26.0
[61] Rcpp_1.0.12
[63] gridExtra_2.3
[65] xfun_0.43
[67] BiocManager_1.30.23
[69] fansi_1.0.6
[71] digest_0.6.35
[73] timechange_0.3.0
[75] mime_0.12
[77] scattermore_1.2
[79] spatstat.data_3.0-4
[81] utf8_1.2.4
[83] renv_1.0.0
[85] prettyunits_1.2.0
[87] htmlwidgets_1.6.4
[89] uwot_0.2.2
[91] gtable_0.3.5
[93] SingleCellExperiment_1.24.0
[95] htmltools_0.5.8.1
[97] scales_1.3.0
[99] gfun_0.1.4
[101] knitr_1.46
[103] tzdb_0.4.0
[105] nlme_3.1-164
[107] zoo_1.8-12
[109] parallel_4.3.3
[111] pillar_1.9.0
[33] spatstat.utils_3.0-4
[35] goftest_1.2-3
[37] spatstat.random_3.2-3
[39] parallelly_1.37.1
[41] leiden_0.4.3.1
[43] DelayedArray_0.28.0
[45] tidyselect_1.2.1
[47] ScaledMatrix_1.10.0
[49] spatstat.explore_3.2-7
[51] BiocNeighbors_1.20.2
[53] ggridges_0.5.6
[55] dbscan_1.1-12
[57] progress_1.2.3
[59] ica_1.0-3
[61] glue_1.7.0
[63] SparseArray_1.2.4
[65] withr_3.0.0
[67] fastmap_1.1.1
[69] rsvd_1.0.5
[71] gridGraphics_0.5-1
[73] R6_2.5.1
[75] colorspace_2.1-0
[77] tensor_1.5
[79] R.methodsS3_1.8.2
[81] generics_0.1.3
[83] data.table_1.15.4
[85] httr_1.4.7
[87] S4Arrays_1.2.1
[89] pkgconfig_2.0.3
[91] lmtest_0.9-40
[93] XVector_0.42.0
[95] dotCall64_1.1-1
[97] png_0.1-8
[99] splancs_2.01-44
[101] rstudioapi_0.16.0
[103] reshape2_1.4.4
[105] cachem_1.0.8
[107] KernSmooth_2.23-22
[109] miniUI_0.1.1.1
[111] grid_4.3.3

[113] vctrs_0.6.5
[115] promises_1.3.0
[117] beachmat_2.18.1
[119] cluster_2.1.6
[121] cli_3.6.2
[123] rlang_1.1.3
[125] future.apply_1.11.2
[127] fs_1.6.4
[129] stringi_1.8.4
[131] viridisLite_0.4.2
[133] munsell_0.5.1
[135] spatstat.geom_3.2-9
[137] RcppHNSW_0.6.0
[139] sparseMatrixStats_1.14.0
[141] shiny_1.8.1.1
[143] memoise_2.0.1
[145] ape_5.8

RANN_2.6.1
BiocSingular_1.18.0
xtable_1.8-4
evaluate_0.23
compiler_4.3.3
crayon_1.5.2
interp_1.1-6
plyr_1.8.9
BiocParallel_1.36.0
deldir_2.0-4
lazyeval_0.2.2
Matrix_1.6-5
hms_1.1.3
future_1.33.2
ROCR_1.0-11
igraph_2.0.3

第二十九章 核心特征单细胞表达

在完成细胞类群的注释并获取准确信息后，我们进一步深入分析了先前确定的核心特征在 bulk-RNA 水平上的表达情况。接下来，为了验证这些特征在细胞层面的表达是否与 bulk-RNA 数据相符，我们特别关注了这些核心特征在不同细胞类群的早晚期表达情况。

29.1 加载 R 包

使用 `rm(list = ls())` 来清空环境中的所有变量。

```
library(tidyverse)
library(Seurat)
library(cowplot)
library(rstatix)
library(viridis)
library(ggpubr)

rm(list = ls())
options(stringsAsFactors = F)
options(future.globals.maxSize = 10000 * 1024^2)

grp_names <- c("Early Stage", "Late Stage")
grp_colors <- c("#8AC786", "#B897CA")

cell_types <- c("Hepatocyte", "T/NK",
                 "Myeloid", "B",
                 "Endothelial", "Fibroblast")

cell_colors <- c("#A01FF0", "#1F78B4", "#4EB29D",
                  "#DA3F4C", "#F1EE97", "#08306B")
```

29.2 导入数据

- seurat_obj 数据来自于章节 [二十八](#)
- common_feature 数据来自于章节 [二十三](#)

```
seurat_obj <- readRDS("./data/result/scRNA/Seurat_celltype.RDS")
common_feature <- read.csv("./data/result/Biomarker/Biomarker_LR_RF_SVM_validation.csv")
```

29.3 细胞类群的 tSNE

查看细胞类群的可视化结果

```
DimPlot(seurat_obj,
        reduction = "tsne",
        label = FALSE,
        cols = cell_colors,
        pt.size = 0.5)
```

结果：从 tSNE 图得知，有 6 种类型的细胞。

29.4 核心特征表达分布

查看核心特征在单细胞水平的表达密度分布图

```
feature_FeaturePlot <- FeaturePlot(
  object = seurat_obj,
  features = common_feature %>%
    dplyr::filter(Enrich %in% c("Both_Early", "Both_Late")) %>%
    dplyr::pull(FeatureID),
  reduction = "tsne",
  ncol = 3)

feature_FeaturePlot
```

结果：6 个核心特征在不同细胞类群的分布不同，在章节 [二十四](#) 发现的 **SLC6A8** 富集在 **Hepatocyte** 细胞群。

29.5 核心特征点图

查看核心特征在单细胞水平的表达的点图

```
feature_DotPlot <- DotPlot(
  object = seurat_obj,
  features = common_feature %>%
    dplyr::filter(Enrich %in% c("Both_Early", "Both_Late")) %>%
    dplyr::pull(FeatureID)) +
  labs(y = "")

feature_DotPlot
```

29.6 SLC6A8 表达分布

单独生成关注的核心特征 **SLC6A8** 在细胞水平的表达分布情况。

```
SLC6A8_DotPlot <- DotPlot(
  object = seurat_obj,
  features = "SLC6A8")

SLC6A8_DotPlot
```

29.7 SLC6A8 差异结果

在先前 bulk RNA 结果小节 23.6 中，我们发现核心特征 **SLC6A8** 富集在 HCC 晚期。现在在细胞水平上，它的表达分布情况是如何？

SLC6A8_vlnplot

结果：在细胞水平上，**SLC6A8** 在 HCC 早期和晚期的差异结果。**SLC6A8** 分别显著富集在 **Hepatocyte** 和 **Myeloid** 的晚期分组。

29.8 输出结果

```
ggsave("./data/result/Figure/Fig7-B.pdf", SLC6A8_FeaturePlot, width = 5, height = 4, dpi = 600)
ggsave("./data/result/Figure/Fig7-C.pdf", SLC6A8_DotPlot, width = 5, height = 3, dpi = 600)
```

```
ggsave("./data/result/Figure/Fig7-D.pdf", SLC6A8_vlnplot, width = 8, height = 4, dpi = 600)
ggsave("./data/result/Figure/SFig6.pdf", feature_DotPlot, width = 8, height = 4, dpi = 600)
```

29.9 总结

在深入研究细胞类群的信息后，我们针对六个核心特征进行了详尽的分析，特别关注了 **SLC6A8** 这一基因在细胞层面的表达情况。

系统信息

```
sessionInfo()

R version 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK version

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats      graphics   grDevices datasets   utils      methods    base

other attached packages:
[1] ggpubr_0.6.0      viridis_0.6.5       viridisLite_0.4.2   rstatix_0.7.2
[5] cowplot_1.1.3     Seurat_5.0.3        SeuratObject_5.0.2  sp_2.1-4
[9] lubridate_1.9.3  forcats_1.0.0       stringr_1.5.1      dplyr_1.1.4
[13] purrrr_1.0.2     readr_2.1.5        tidyverse_2.0.0
[17] ggplot2_3.5.1     tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] RColorBrewer_1.1-3    rstudioapi_0.16.0    jsonlite_1.8.8
[4] magrittr_2.0.3         spatstat.utils_3.0-4  rmarkdown_2.26
[7] vctrs_0.6.5           ROCR_1.0-11          spatstat.explore_3.2-7
[10] htmltools_0.5.8.1     broom_1.0.5          sctransform_0.4.1
```

[13] parallelly_1.37.1 KernSmooth_2.23-22 htmlwidgets_1.6.4
[16] ica_1.0-3 plyr_1.8.9 plotly_4.10.4
[19] zoo_1.8-12 igraph_2.0.3 mime_0.12
[22] lifecycle_1.0.4 pkgconfig_2.0.3 Matrix_1.6-5
[25] R6_2.5.1 fastmap_1.1.1 fitdistrplus_1.1-11
[28] future_1.33.2 shiny_1.8.1.1 digest_0.6.35
[31] colorspace_2.1-0 patchwork_1.2.0 tensor_1.5
[34] RSpectra_0.16-1 irlba_2.3.5.1 progressr_0.14.0
[37] fansi_1.0.6 spatstat.sparse_3.0-3 timechange_0.3.0
[40] httr_1.4.7 polyclip_1.10-6 abind_1.4-5
[43] compiler_4.3.3 withr_3.0.0 backports_1.4.1
[46] carData_3.0-5 fastDummies_1.7.3 ggsignif_0.6.4
[49] MASS_7.3-60.0.1 tools_4.3.3 lmtest_0.9-40
[52] httpuv_1.6.15 future.apply_1.11.2 goftest_1.2-3
[55] glue_1.7.0 nlme_3.1-164 promises_1.3.0
[58] grid_4.3.3 Rtsne_0.17 cluster_2.1.6
[61] reshape2_1.4.4 generics_0.1.3 gtable_0.3.5
[64] spatstat.data_3.0-4 tzdb_0.4.0 data.table_1.15.4
[67] hms_1.1.3 car_3.1-2 utf8_1.2.4
[70] spatstat.geom_3.2-9 RcppAnnoy_0.0.22 ggrepel_0.9.5
[73] RANN_2.6.1 pillar_1.9.0 spam_2.10-0
[76] RcppHNSW_0.6.0 later_1.3.2 splines_4.3.3
[79] lattice_0.22-6 renv_1.0.0 survival_3.7-0
[82] deldir_2.0-4 tidyselect_1.2.1 miniUI_0.1.1.1
[85] pbapply_1.7-2 knitr_1.46 gridExtra_2.3
[88] scattermore_1.2 xfun_0.43 matrixStats_1.3.0
[91] stringi_1.8.4 lazyeval_0.2.2 yaml_2.3.8
[94] evaluate_0.23 codetools_0.2-19 BiocManager_1.30.23
[97] cli_3.6.2 uwot_0.2.2 xtable_1.8-4
[100] reticulate_1.37.0 munsell_0.5.1 Rcpp_1.0.12
[103] globals_0.16.3 spatstat.random_3.2-3 png_0.1-8
[106] parallel_4.3.3 dotCall64_1.1-1 listenv_0.9.1
[109] scales_1.3.0 ggridges_0.5.6 leiden_0.4.3.1
[112] rlang_1.1.3

第九部分

撰写文章

第三十章 框架

一篇论文的组成通常遵循一定的结构和规范，以确保信息的清晰传递和研究的科学严谨性。它一般包含标题 (title)、摘要 (abstract)、介绍 (introduction)、方法和材料 (methods and materials)、结果 (results)、讨论 (discussion) 和结论 (conclusion) 等。这些组成之间有什么区别又该如何来撰写它们呢。

30.1 标题 (title)

标题是论文的简短概括，旨在吸引读者的注意并快速了解论文的主题。标题应简洁明了，包含关键词，能够准确反映论文的主要内容和研究焦点。避免使用过于复杂或冗长的句子。

30.2 摘要 (abstract)

摘要是对论文内容的简要总结，为读者提供论文的整体概览。摘要应包括研究背景、目的、方法、主要结果和结论。通常不超过 200-300 字，语言应精炼，结构清晰。

30.3 介绍 (introduction)

介绍部分阐述研究的背景、意义、目的和理论基础，为后续的研究内容做铺垫。首先概述相关领域的研究现状和研究空白，然后明确研究目的和研究问题，最后简要介绍研究方法和预期结果。

30.4 方法和材料 (methods and materials)

方法和材料部分详细描述研究过程中采用的研究方法、实验设计、样本选择、数据收集和分析方法等。确保详细描述研究方法的每个步骤，使读者能够复现实验。同时，也要注明所使用的材料和设备，以及数据的来源和处理方式。

30.5 结果 (results)

结果部分呈现研究的发现和数据，通常以表格、图表或文字描述的形式呈现。清晰、准确地描述研究结果，避免主观解释。同时，对结果进行必要的统计分析，以支持研究结论。

30.6 讨论 (discussion)

讨论部分对研究结果进行解释和分析，将结果与已有研究进行比较，指出研究的限制和不足之处，并提出未来研究的方向。在讨论中，要客观地评价研究结果，避免夸大或缩小研究价值。同时，要深入探讨研究结果的深层含义和可能的影响。

30.7 结论 (conclusion)

结论部分是对全文的总结，强调研究的主要发现和贡献，以及研究的局限性和未来展望。结论应简洁明了，概括研究的主要结果和贡献。同时，也要指出研究的局限性和不足之处，并提出对未来研究的建议。

第三十一章 结果

在串联文章结果的过程中，我们需要首先明确研究的目标或问题，并围绕这一目标或问题来组织我们的结果。以下是根据本教程数据分析的结果展示了如何将关键发现串联起来：

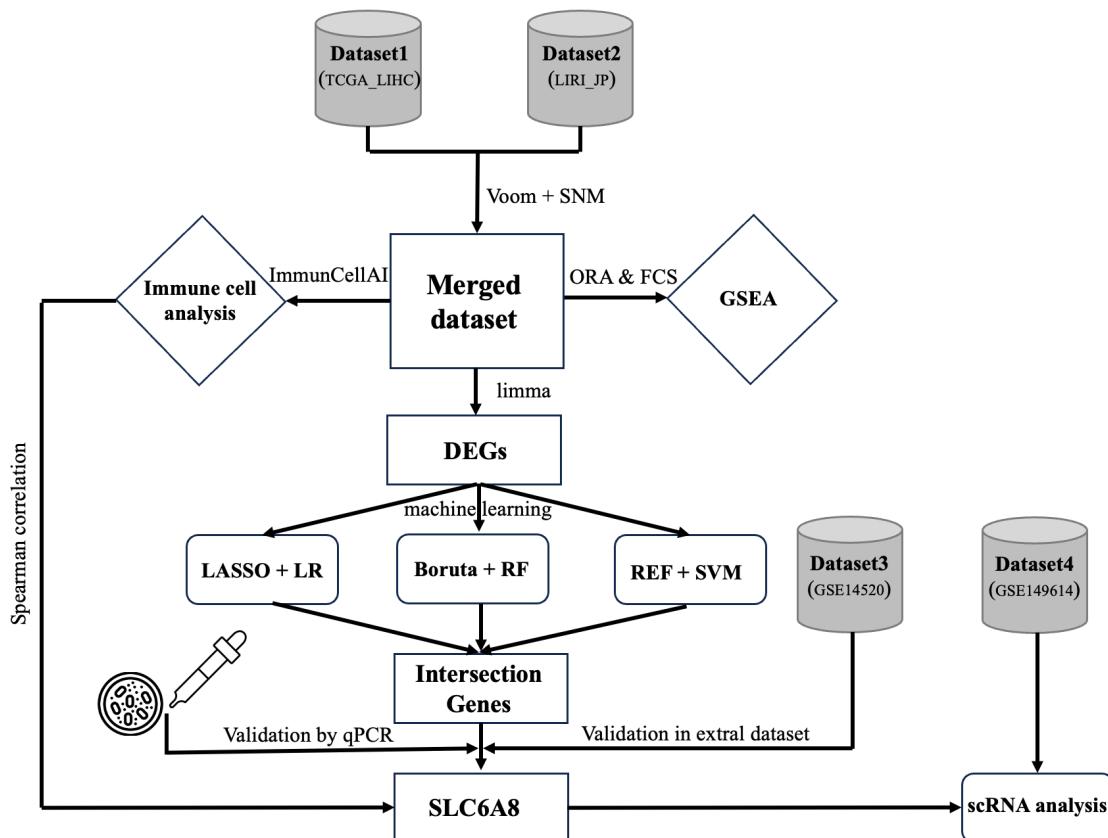


图 31.1: Figure 1: Schematic workflow of this study

Figure 1: Schematic workflow of this study.

首先，我们通过整合两个大的数据来源，构建了一个整合的肝癌数据集。这一步骤为我们后续的差异基因分析提供了坚实的基础。基于这个数据集，我们运用统计方法识别出了在早晚期肝癌中表达差异显著的基因。这些差异基因不仅为我们揭示了肝癌发展过程中的分子变化，而且为候选标记物的筛选提供了候选列表。

接下来，我们利用机器学习技术，从差异基因中筛选出具有潜在临床价值的候选标记物。通过构建分类模型并进行交叉验证，我们确保了候选标记物在区分早晚期肝癌方面的准确性和可靠性。这些候选标记物不仅具有高度的区分能力，而且与肝癌的分期等临床信息密切相关。

为了验证候选标记物的有效性，我们进行了关联分析和外部数据验证。通过比较候选标记物与肝癌患者的免疫浸润细胞的相关性，我们进一步证实了这些标记物在肝癌诊断中的潜在价值。同时，我们利用外部数据集对候选标记物进行了验证，确保了其在不同人群中的普适性和可靠性。

此外，我们还对差异基因进行了功能分析，以深入了解这些基因在肝癌发生和发展中的潜在作用。这些发现不仅为我们提供了关于肝癌生物学特性的新见解，而且为未来的治疗策略提供了理论基础。

综合上面的分析过程，本次研究的结果暂时可以设置为六个部分，它们分别是（差异基因、功能探究、免疫浸润分析、筛选标记物、目标基因关联分析、单细胞水平验证目标基因）：

1. 肝癌早晚期的差异基因识别；
2. 肝癌早晚期的功能区别；
3. 肝癌早晚期的免疫浸润细胞差异；
4. 机器学习识别的潜在候选基因标记物；
5. 基因标记物 SLC6A8 和免疫细胞的相关性；
6. 基因标记物 SLC6A8 在单细胞水平上表达一致性分析.

31.1 Supplemental Figure1

31.2 Figure2

31.3 Figure3

在流程图中，功能分析是从差异基因衍生出的分析环节，其之所以独立成为一个章节，是因为相较于其他分析方向，此处更适宜作为上下文的衔接点。

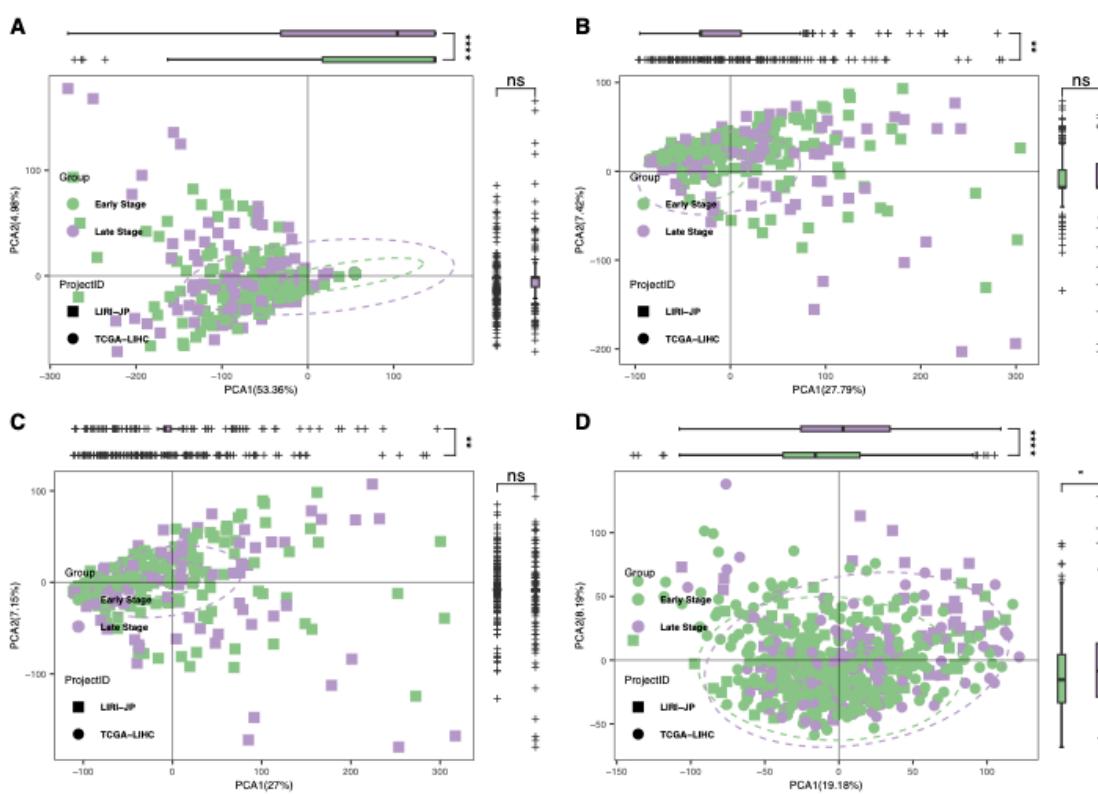


图 31.2: **Supplemental Figure 1:** PCA of multiple types of gene expression matrix for batch removal on merge data cohort (LIHC-US and LIRI-JP).

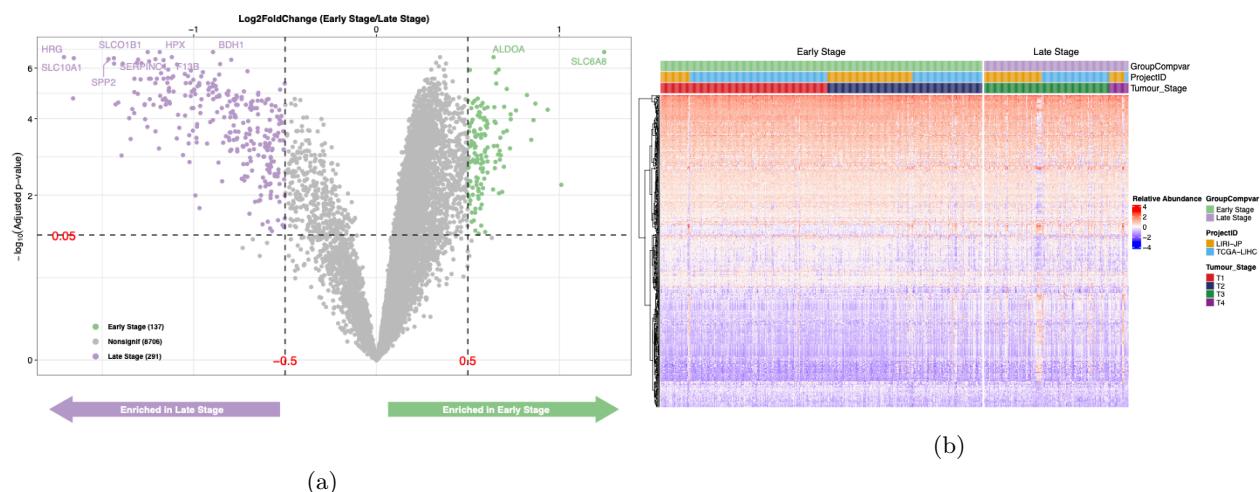


图 31.3: **Figure 2:** DEGs between the early-stage group and the late-stage group in HCC merge dataset (LIHC-US and LIRI-JP).

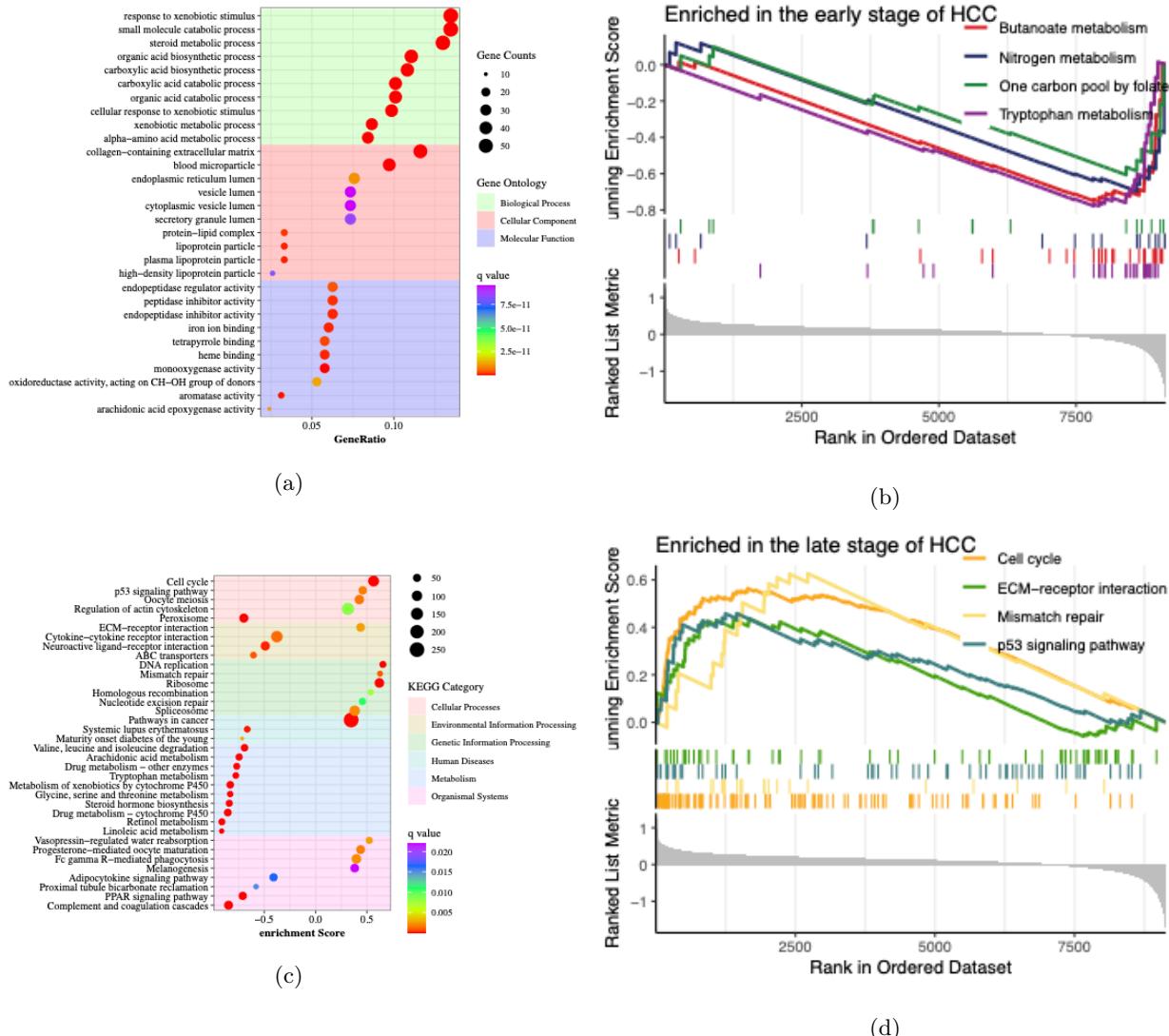


图 31.4: **Figure 3:** Functional analysis of DEGs between the early-stage group and the late-stage group in HCC merge cohort.

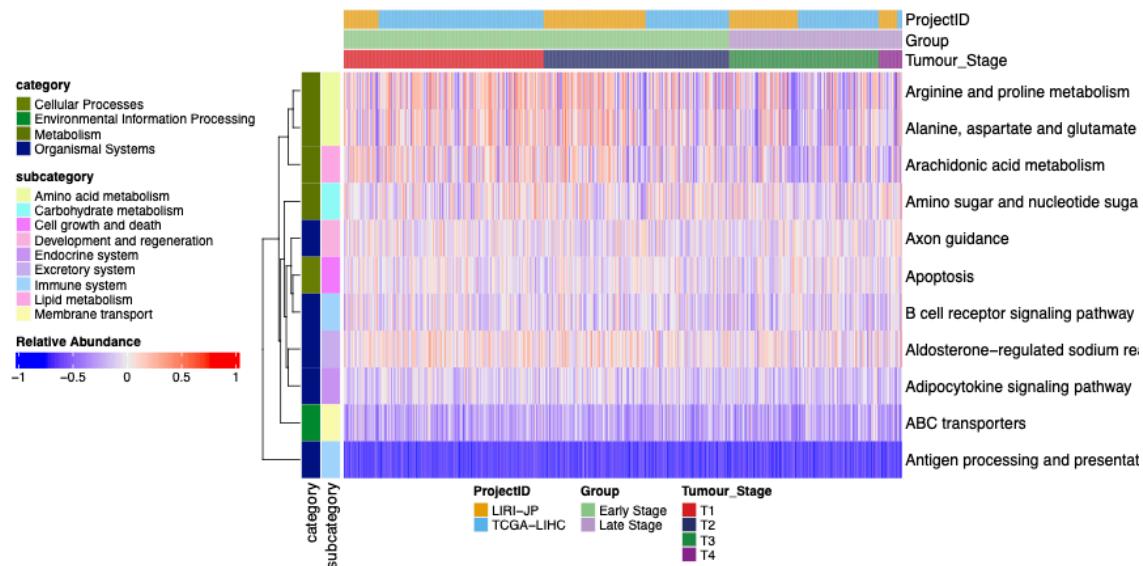


图 31.5: Supplemental Figure 2: GSVA enrichment analysis of gene expression matrix between the HCC early stage and late stage in merge data.

31.4 Supplemental Figure2

31.5 Figure4

31.6 Supplemental Figure3

31.7 Figure5

31.8 Supplemental Figure4

31.9 Figure6

31.10 Figure7

31.11 撰写结果

现在可以将文章的研究成果划分为以下几个主要部分：

1. Identification of DEGs in the HCC early stage and late stage;
2. Functional analysis of DEGs by GO and KEGG enrichment analysis;

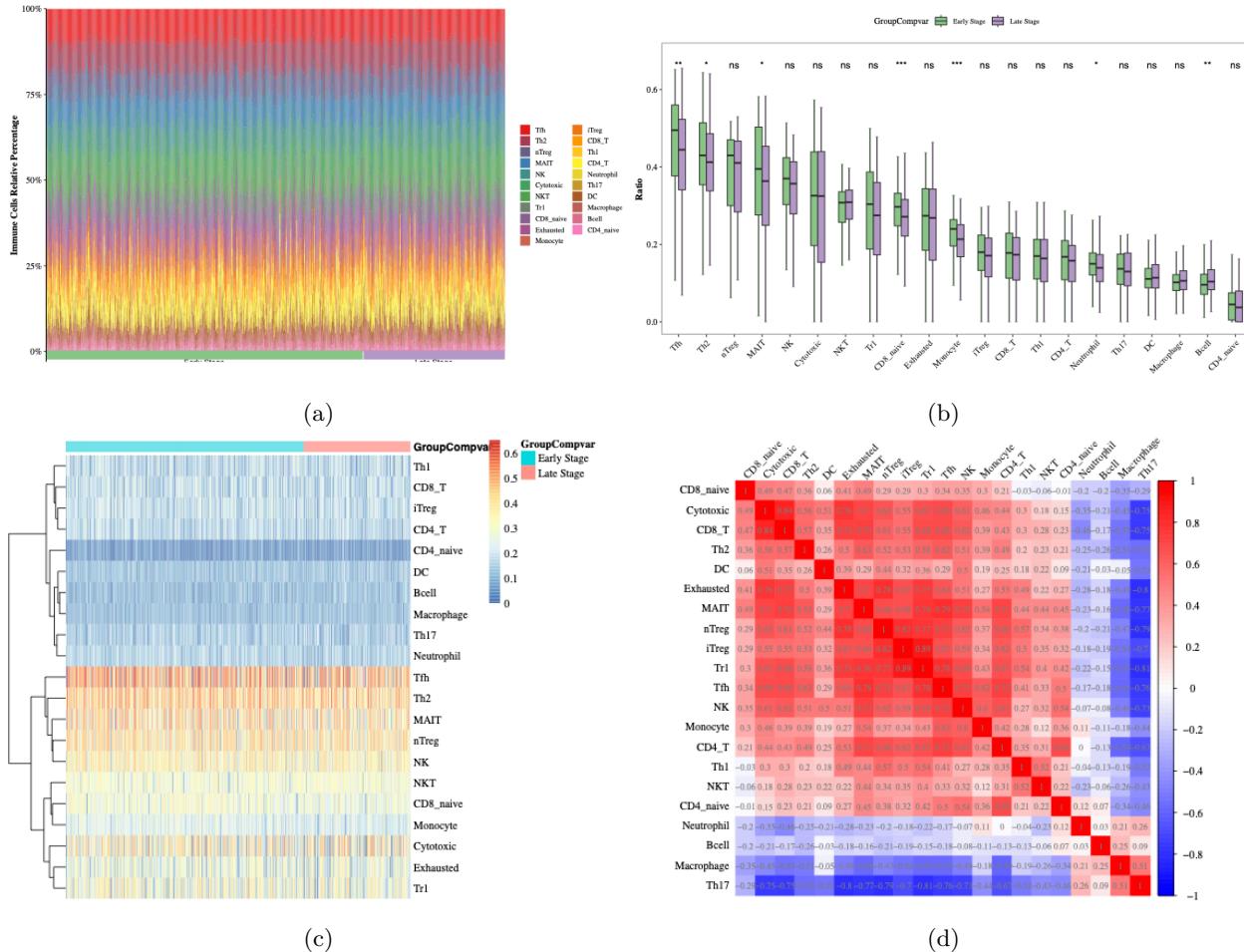


图 31.6: Figure 4: Evaluation and visualization of immune cell infiltration between the HCC early stage and late stage in merge cohort.

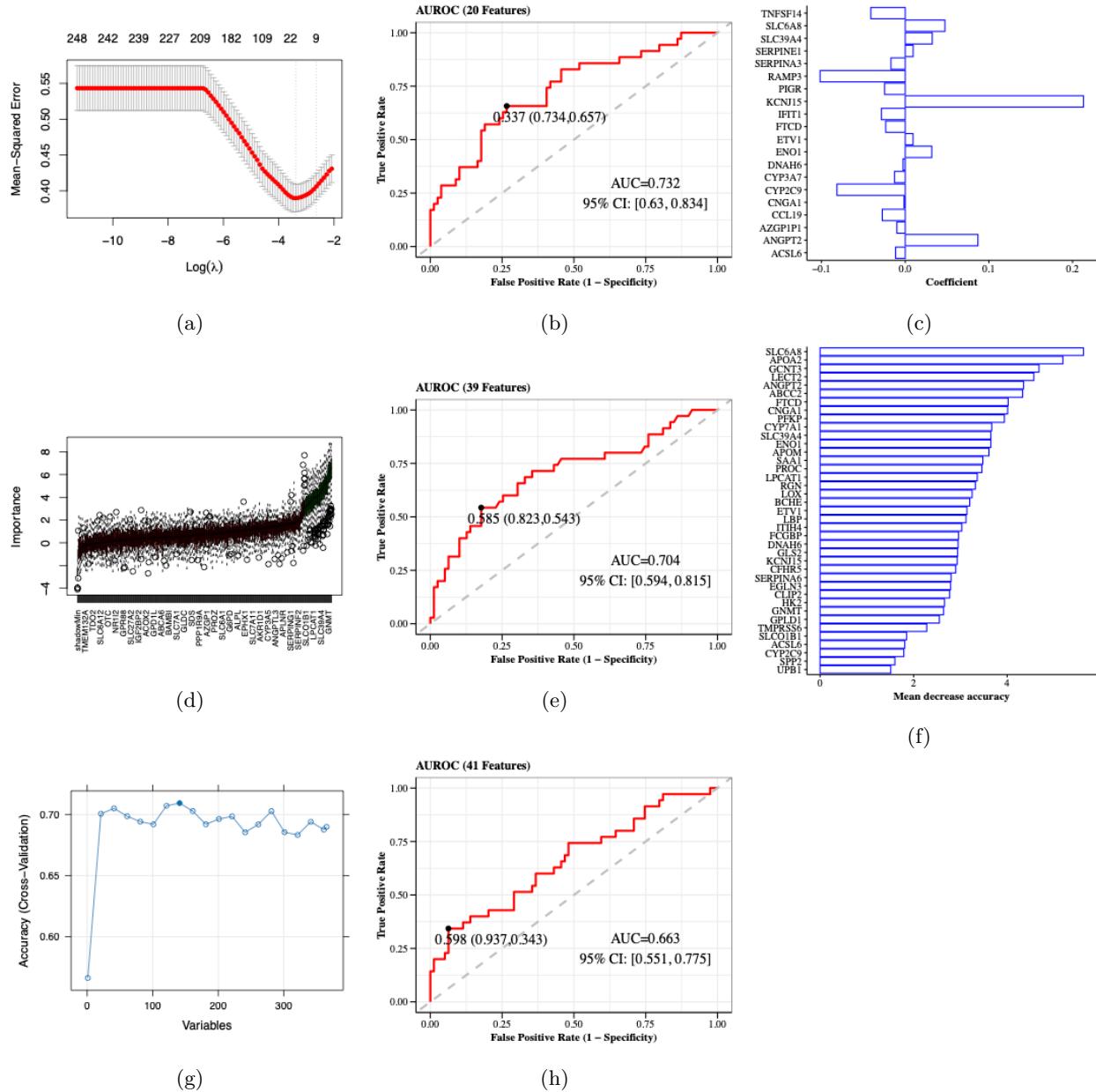


图 31.7: Supplemental Figure 3: Machine learning algorithms to screen potential diagnostic markers from DEGs between the HCC early stage and late stage in merge data.

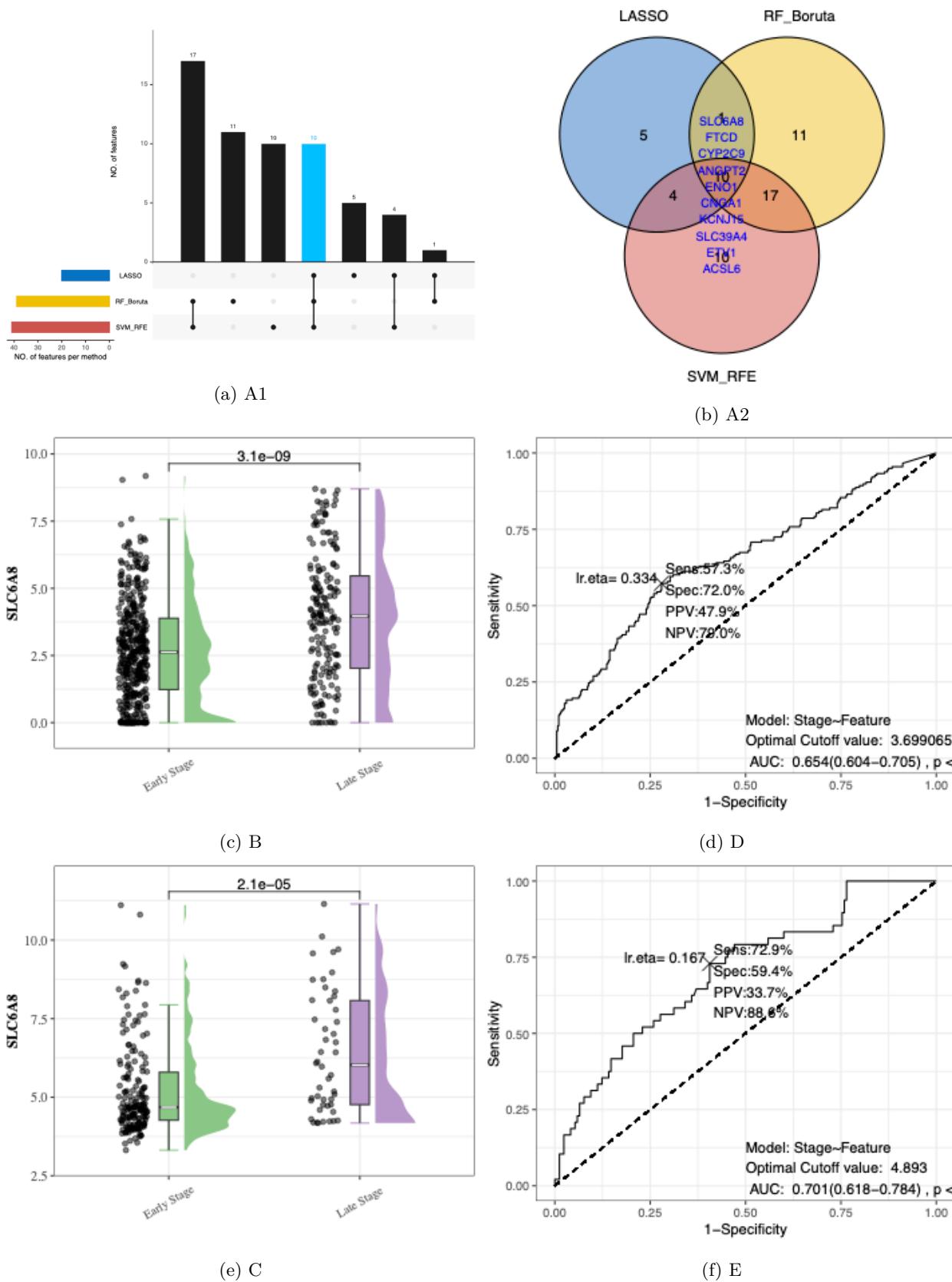


图 31.8: Figure 5: Validation and ROC of SLC6A8 diagnostic marker identified from the merge datasets by multiple machine learning algorithms.

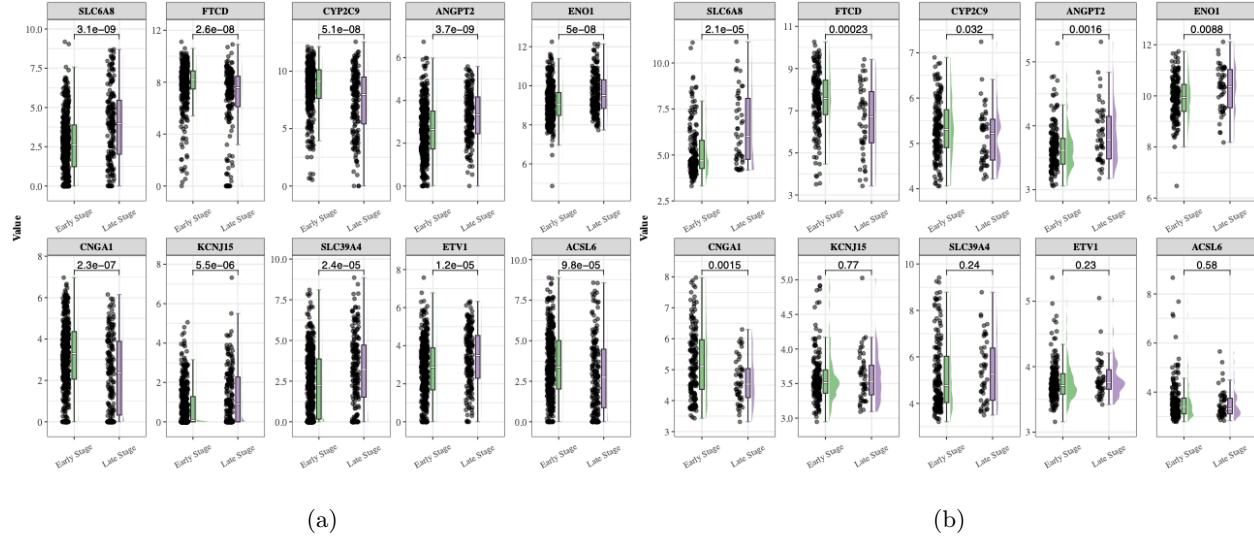


图 31.9: Supplemental Figure 4: 10 overlapping genes screened from machine learning algorithms were validated in the validation cohort.

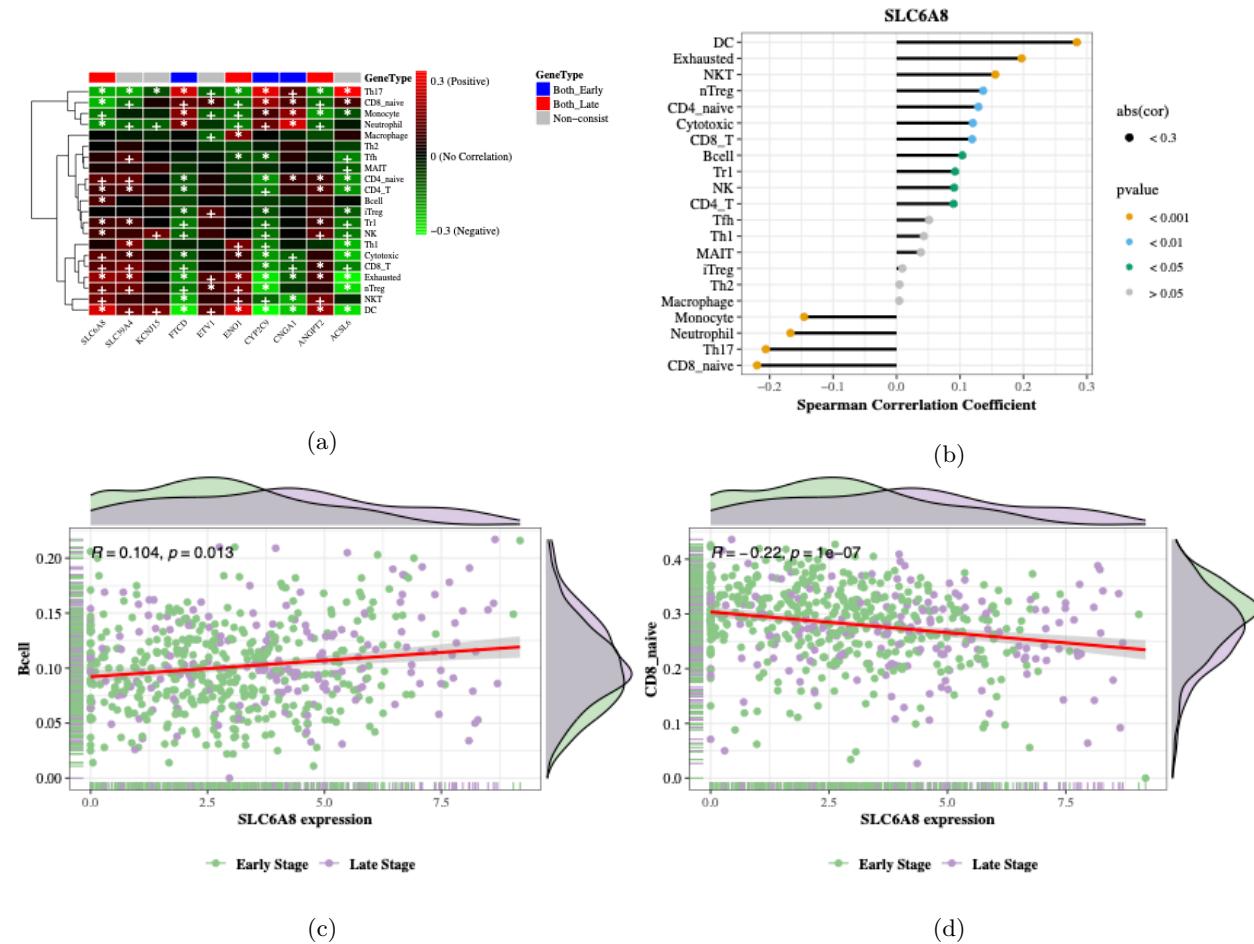


图 31.10: Figure 6: Correlation between SLC6A8 and 21 immune cells.

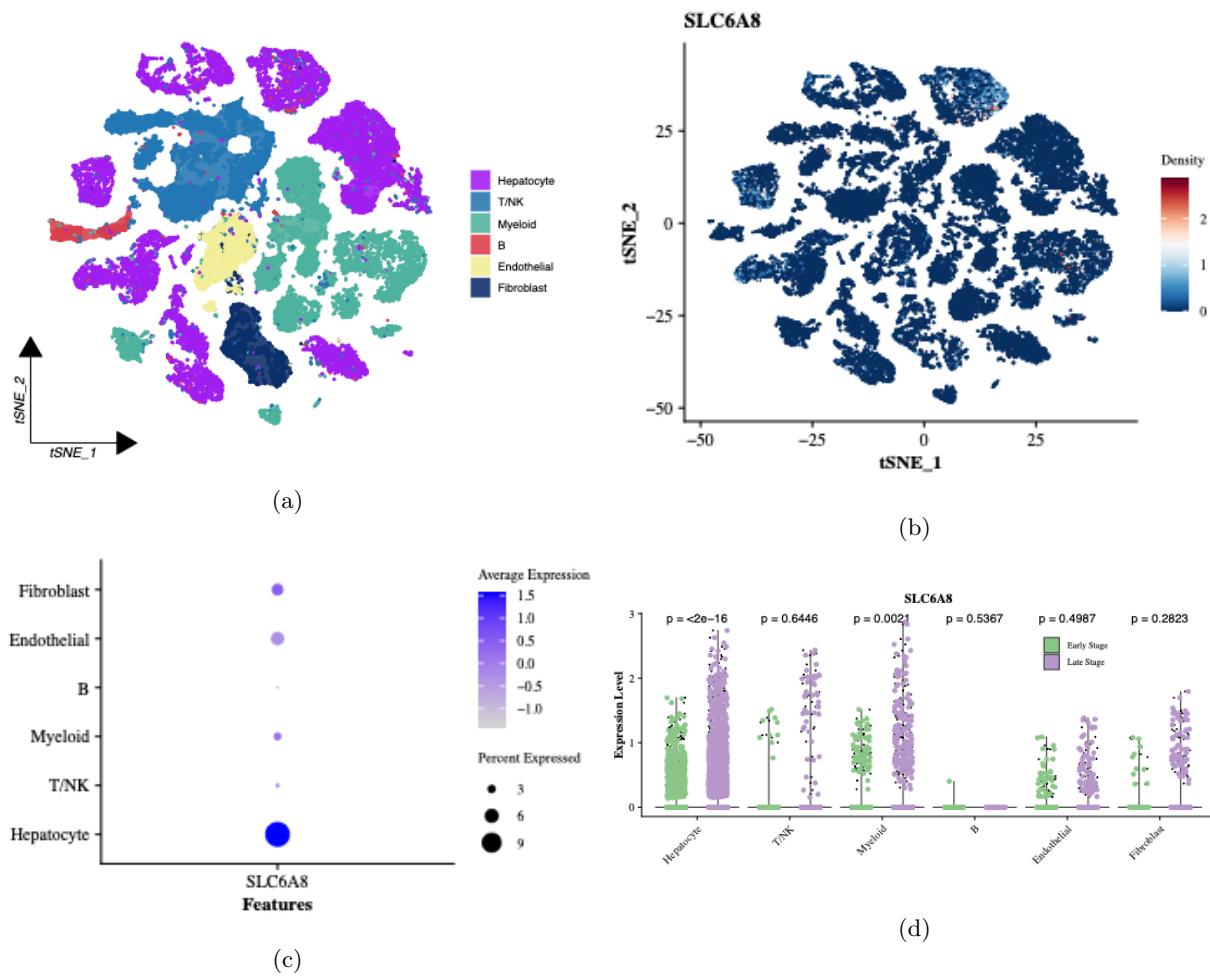


图 31.11: Figure 7: *SLC6A8* expression in single-cell GSE149614 transcriptomic dataset.

3. Significant changes between two stages in immune cells by ImmuneCellAI;
4. Potential gene biomarkers identified by multiple machine learning approaches;
5. Correlation analysis between SLC6A8 and immune cells;
6. Expression level of SLC6A8 in single-cell transcriptomic data;

现在主要的点在于怎么撰写结果。在撰写研究结果部分时，建议首先明确地提出几个数据分析中观察到的最显著的趋势或模式。这些关键点应简洁地反映出研究的核心成果。比如：

- 第一部分

We conducted an analysis to identify differentially expressed genes

A total of 137 genes exhibited significant enrichment in the early-stage group

- 第二部分

To delve into the potential biological implications of differentially expressed genes (DEGs)

Furthermore, through Gene Set Enrichment Analysis (GSEA)

- 第三部分

Utilizing the ImmuCellAI, a cutting-edge immune cell infiltration algorithm

- 第四部分

To identify potential gene biomarkers that can distinguish between the early and late stages of HCC

Subsequently, we extracted 10 intersection biomarkers

To identify diagnostic markers among the six genes

- 第五部分

Utilizing Spearman correlation,

- 第六部分

Utilizing t-SNE visualization based on the Seurat-class object

Subsequently, we segregated the cells into two groups analysis.

在撰写研究结果时，可以采取一种逐步细化的方法。

第三十二章 方法

本文涉及到的材料和方法包含了以下内容：

- 数据收集和整理: Data collection and download
- 数据整合: Data collection and Integration
- 差异基因分析: Differential Expression Genes
- 功能富集分析: Functional enrichment analysis
- 免疫浸润分析: Immune cell infiltration
- 候选标记物识别: Potential biomarkers selection
- 诊断 ROC 曲线: ROC of diagnostic biomarker
- 单细胞分析: Single cell transcriptome data processing and analyzing
- 统计方法: Statistical analysis

从这里可以看出，它们组织和结果部分是一致的。

32.1 数据收集和整理

在生物信息学研究中，尤其是当研究基于公开数据时，详细记录数据的收集和下载过程是至关重要的。这种透明度对于确保研究的可靠性和可重复性至关重要。通过清晰地描述数据的来源、收集方法和下载步骤，我们为其他研究者提供了一个清晰的指南，使他们能够理解数据的合理性和适用性。此外，详细说明数据的获取过程也有助于建立研究的信誉，因为它允许读者评估数据的质量和相关性。这包括提供数据集的详细信息，如数据集的名称、来源链接、访问日期以及任何必要的数据集描述或特征。

Standardized RNA-Seq reads from the LIHC-US and LIRI-JP projects (Release 28) were retrieved from The Cancer Genome Atlas (TCGA, <https://cancergenome.nih.gov/>) and the International Cancer Genome Consortium (ICGC, <https://dcc.icgc.org/>), respectively.

Based on the TNM staging system, HCC is classified into four stages: stage I, stage II, stage III, and stage IV.

32.2 数据整合

在分析不同来源的转录组数据时，我们面临着批次效应的挑战，这些效应可能会对生物学变异产生干扰。批次效应的存在可能会影响我们对数据的解释，因此，采取适当的方法来降低这些效应是必要的。在研究中，我们明确指出了用于减少批次效应的策略和技术，包括但不限于数据标准化、批次校正算法或多变量分析方法。我们详细描述了所采用的方法。这些信息对于其他研究者来说是至关重要的，因为它们不仅有助于理解我们如何控制批次效应，而且也为其他研究提供了可能的解决方案。通过这种方法，我们确保了研究结果的生物学解释更加准确，减少了批次效应对数据的潜在影响，从而提高了研究的质量和可信度。

A consolidated dataset incorporating the LIHC-US and LIRI-JP datasets was created using the R packages “limma” (version 3.58.11) and “SNM” (version 1.50.0).

32.3 差异基因分析

在执行差异分析的过程中，我们首先详细说明了所使用的软件工具和版本信息，确保了分析过程的透明度和标准化。此外，我们明确了用于判断基因表达差异显著性的标准，包括使用的统计测试、阈值设定以及校正多重比较的方法。

To identify differentially expressed genes (DEGs) in the merged dataset, we applied a threshold of $|\log FC| > 0.5$ and an adjusted-pvalue < 0.05 using the “limma” (version 3.58.11) R package.

32.4 功能富集分析

功能富集分析需要做和差异分析类型的处理。

Gene Ontology (GO) enrichment analysis was carried out to explore the biological significance of the differentially expressed genes (DEGs) utilizing the clusterProfiler (version 4.10.1) R package.

32.5 免疫浸润分析

同理免疫浸润分析也是类似的道理。

The ImmuCellAI, a gene set signature-driven approach, serves as a deconvolution algorithm capable of estimating the abundance of 24 distinct immune cell types.

32.6 候选标记物识别

该部分是我们研究方法学的核心，详细描述了如何利用机器学习技术识别候选生物标记物。我们采用了三种不同的机器学习方法，并明确了每种方法所涉及的软件工具、分析步骤以及用于评估模型性能的判断标准。

In conclusion, we consolidated the overlapping genes identified by the LASSO-LR, SVM-RFE, and Brouta-RF algorithms for further analysis.

32.7 诊断 ROC 曲线

在诊断生物标记物的研究中，接收者操作特征（ROC）曲线扮演着至关重要的角色。ROC 曲线的绘制和分析对于判断标记物的诊断效能至关重要。为了确保评估过程的透明度和结果的可重复性，我们明确指出了用于生成 ROC 曲线的软件工具及其版本。

The effectiveness of each diagnostic biomarker was evaluated using receiver operating characteristic (ROC) curves generated by the “pROC” (version 1.18.5) and “multipleROC” (version 0.1.1) R packages.

32.8 单细胞分析

在进行单细胞分析的过程中，我们特别强调了所使用的软件工具及其版本的重要性。详细记录了软件的配置参数和分析流程，包括数据预处理、质量控制、细胞类型鉴定、基因表达模式的聚类分析以及差异表达基因的识别。这些步骤的详细描述不仅有助于其他研究者理解我们的分析方法，而且也便于他们在自己的研究中应用相似的分析策略。

The GSE149614 raw data was retrieved from GEO databases.

32.9 统计方法

统计方法是文章必不可少的部分，明确软件的版本等信息。

Statistical analysis was performed using R (version 4.3.3) and RStudio (version 2023.12.1+402).

第三十三章 讨论

33.1 段落逻辑

在第一段中，我们首先重申了研究的背景，阐明了进行这项研究的必要性和重要性。我们指出了研究的主要目的，并简述了我们选择的研究方法和设计。我们旨在为读者提供一个清晰的框架，帮助他们理解研究的核心内容。这不仅为后续的深入讨论提供了基础，而且也突出了研究的创新点和科学价值。

HCC, characterized by high morbidity and mortality rates, has emerged as a significant cause of cancer-related deaths globally.

在该段讨论部分，我们综合考虑了第一、第二和第三结果，以探讨它们在早晚期肿瘤发展中的基因表达、功能变化和免疫细胞浸润方面的综合影响。我们的目标是揭示这些差异的生物学基础，并评估它们对临床实践的潜在指导意义。通过对这些结果的深入分析，我们不仅能够更好地理解肿瘤发展的分子机制，而且能够为临床医生提供关于疾病监测、治疗选择和患者管理的新见解。

In this study, we identified 137 down-regulated genes and 291 up-regulated genes, and found that some of these differentially expressed genes (DEGs) were enriched in cancer-related pathways such as the p53 signaling pathway and tryptophan metabolism.

在讨论部分的后续内容中，我们专注于对第四部分中识别出的标记物进行深入的生物学探讨。特别是，我们将重点分析 SLC6A8 基因，以验证其与肝细胞癌（HCC）的生物学过程的相关性，并探讨其作为潜在诊断标记物的科学依据。通过对 SLC6A8 基因的功能、其在 HCC 发展中的潜在作用机制，以及其在现有文献中的相关性进行综合分析，我们提供了有力的证据，支持 SLC6A8 基因与 HCC 的发生和发展密切相关。

Utilizing LASSO analysis, RF-Boruta, and SVM-REF, we screened ten differentially expressed genes (DEGs) at the intersection of various gene sets: SLC6A8, FTCD, CYP2C9, ANGPT2, ENO1, CNGA1, KCNJ15, SLC39A4, ETV1, and ACSL6. These genes have been previously implicated in a diverse array of malignancies

接着我们又对第五部分标记物尤其是 SLC6A8 和免疫细胞的相关性进行了讨论，探究 SLC6A8 和免疫微环境的相互关系在生物学上的意义

There was a disparity in immune capacity between the two stages of HCC, yet the specific immunity associated with SLC6A8 remained unclear.

然后我们对第六部分单细胞研究的结果进行讨论（这一部分有点羸弱，暂时没有看到什么好的单细胞文章验证 SLC6A8 的结果）。

We further examined whether the expression of SLC6A8 at the single-cell transcriptome level and in vitro experimentation concurred with bulk-RNA sequencing findings.

最后我们需要对整体讨论做一个总结。

In summary, our transcriptomic data analysis revealed a significant upregulation of SLC6A8 in the late stage of HCC.

还需要补充上研究的局限性。

In our study, we have identified potential diagnostic biomarkers for both the early and late stages of HCC, supported by transcriptomic data and validated through single-cell transcriptomic analysis and in vitro experiments. While one limitation

第三十四章 介绍

本研究通过以下部分来组织引言：

- 肝细胞癌（HCC）作为一种高致命性的癌症，其早期诊断和有效治疗一直是医学领域面临的重要挑战。HCC 的复杂性以及肿瘤微环境的异质性使得其诊断和治疗变得尤为困难。因此，深入理解 HCC 的免疫微环境，并发展出高效、准确的早期诊断方法，对于提高 HCC 患者的生存率和生活质量具有重要意义。

Hepatocellular carcinoma (HCC), which ranked as the fourth leading cause of death globally in 2018, remains a significant global health concern

- 近年来，基于基因等单分子诊断预测的研究在 HCC 领域展现出了巨大的潜力和前景。通过深入研究 HCC 的基因组、转录组等分子特征，我们可以更准确地了解肿瘤的生物学特性和演变规律，为疾病的诊断和治疗提供有力支持。

In recent decades, HCC has garnered significant attention in the research landscape. For instance, certain genes like CLTA, TALDO1, and CSTB, identified as a gene signature via single-cell RNA sequencing, have been linked to survival outcomes and immunotherapy responses

- 在数据驱动的时代背景下，二代测序技术和机器学习的结合为单分子诊断模型的发展提供了新的思路和方法。通过高通量测序技术获取海量的基因数据，结合机器学习算法进行深度分析和挖掘，我们可以更准确地预测 HCC 的发生、发展和转移风险，为临床决策提供科学依据。

Recently, the rapid advancements in next-generation sequencing technologies have yielded vast amounts of RNA sequencing data for hepatocellular carcinoma (HCC).

- 本研究正是在这样的背景下展开，通过综合运用二代测序和机器学习技术，我们深入探究了 HCC 的免疫微环境，并构建了一种高效、准确的单分子诊断模型。本研究的主要结果将为 HCC 的早期诊断和精准治疗提供新的思路和方法，具有重要的临床意义和科研价值。

In this study, we harnessed well-established bioinformatic tools to search for potential biomarkers that could indicate the advanced stage of hepatocellular carcinoma (HCC).

第三十五章 结论和摘要

这两部分都是总结性的部分，因此将它们放置在一起。

35.1 结论

撰写一篇科学研究文章的结论部分，需要对研究结果进行精炼的总结，并强调其对领域内的贡献和意义。本研究核心发现在于找到了 **SLC6A8** 作为 HCC 早晚期诊断标记物。

In summary

35.2 摘要

文章的摘要是整篇文章的缩影，它简洁地概述了研究的主要内容和结论。摘要通常包括以下几个要素：
1) 研究目的：探究 HCC 的早晚期诊断标记物。2) 研究方法：基于大数据的多种机器学习方法。3) 研究结果：机器学习筛选到 10 个候选标记物，其中以 SLC6A8 为最佳。4) 结论：SLC6A8 可能可以作为临床诊断标记物。

Hepatocellular carcinoma (HCC), a chronic liver disease marked by persistent tumor development, represents a substantial global burden, contributing significantly to mortality rates.

初稿

⚠ Pay Attention!!!

该文章已发表了，请勿使用该初稿投递任何期刊，它仅作为学习的参考材料。

经过对研究背景、目的、方法、结果和结论的详细阐述，我们将文章的各个部分初步整合成一篇完整的初稿。然而，需要注意的是，尽管初稿已经形成，但为了确保文章的质量和可读性，它仍需经过一系列的精修和校对过程。

- **结构审查：**首先，我们需要对初稿的结构进行审查，确保每个部分都符合逻辑顺序，并且与研究的主题紧密相关。
- **内容完善：**接着，对每个部分的内容进行深入的完善，包括对研究方法的详细描述、对结果的准确呈现以及对结论的有力论证。
- **语言润色：**文章的语言需要流畅、准确，避免使用模糊不清的表达。对初稿中的语法错误、拼写错误以及不恰当的术语进行修正。
- **格式调整：**根据目标期刊或会议的格式要求，对初稿的格式进行调整，包括引用格式、图表标注、页边距等。
- **同行评审：**在提交前，可以让同行或导师对初稿进行评审，以获得宝贵的反馈和建议。
- **最终校对：**根据反馈进行修改后，进行最后的校对，确保文章在提交前达到最高的学术标准。
- **准备提交：**完成所有修订后，准备相关的提交材料，如封面信、作者信息等，以备投稿。

Title

Multiple machine learning algorithms have pinpointed SLC6A8 as a diagnostic biomarker for the late stage of Hepatocellular carcinoma.

Abstract

Introduction

Materials and methods

Data collection and download

Data Integration

Differential Expression Genes

Functional enrichment analysis

Immune cell infiltration

Potential biomarkers selection

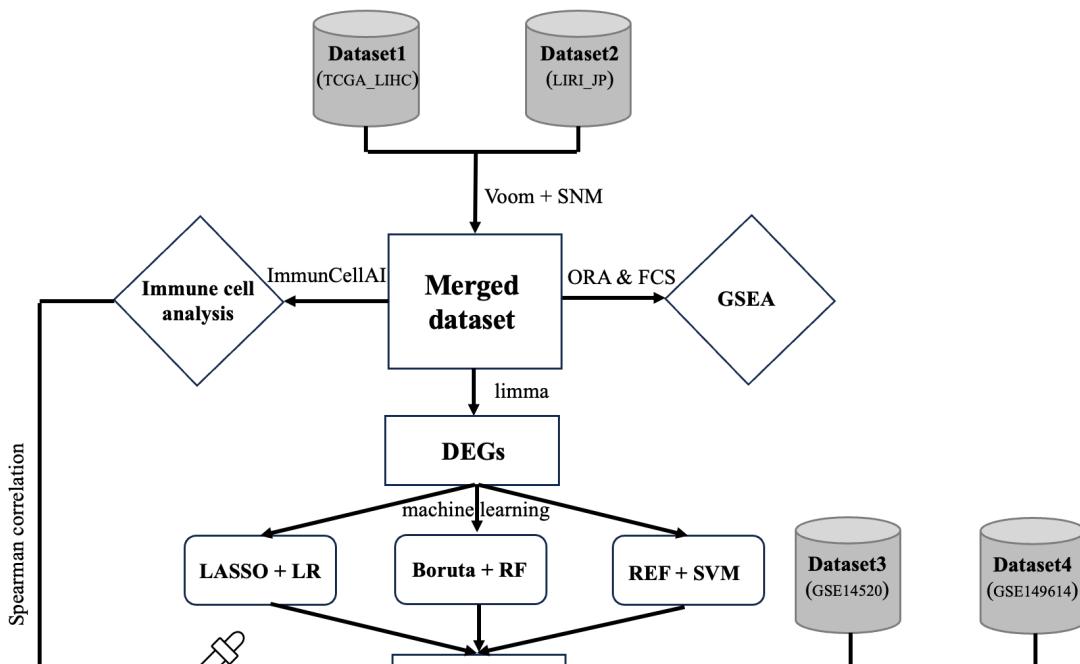
ROC of diagnostic biomarker

Single cell transcriptome data processing and analyzing

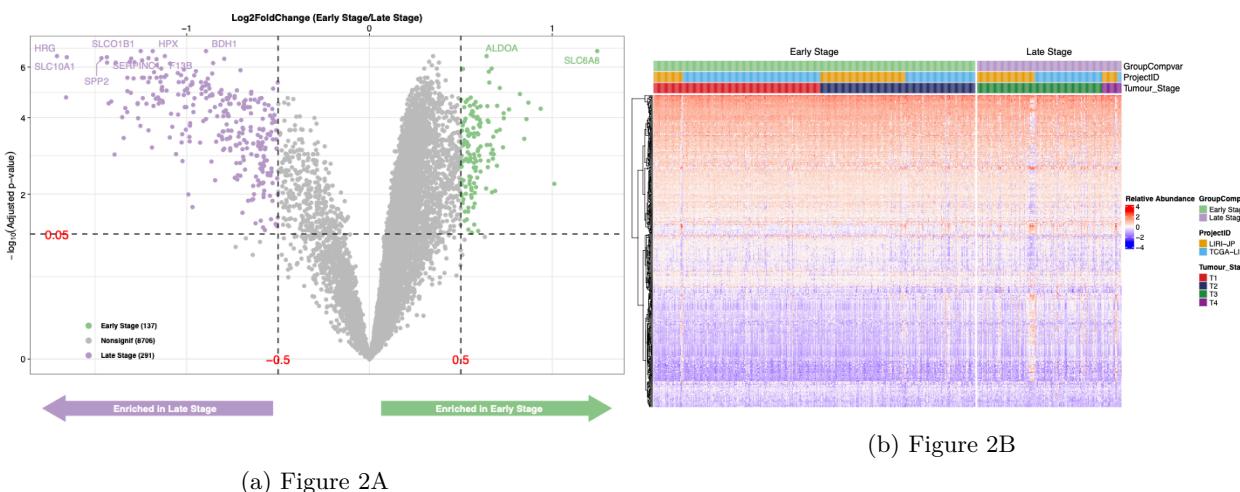
Statistical analysis

Results

Identification of DEGs in the HCC early stage and late stage



©Hua



(a) Figure 2A

Functional analysis of DEGs by GO and KEGG enrichment analysis

Significant changes between two stages in immune cells by ImmuneCellAI

Potential gene biomarkers identified by multiple machine learning approaches

Correlation analysis between SLC6A8 and immune cells

Expression level of SLC6A8 in single-cell transcriptomic data

Discussion

Conclusion

Limitation of the study

Supplemental Figures

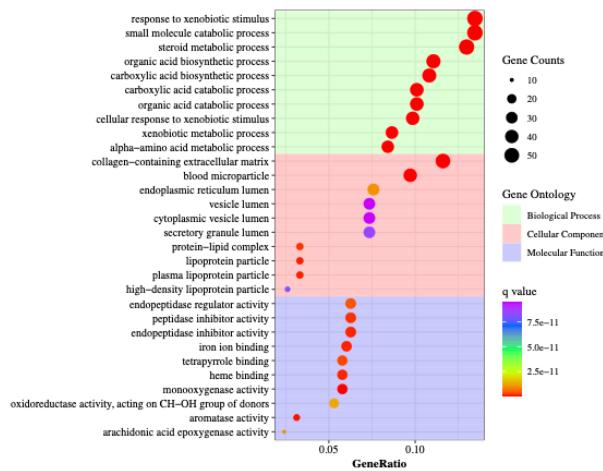


图 35.3: Figure 3A

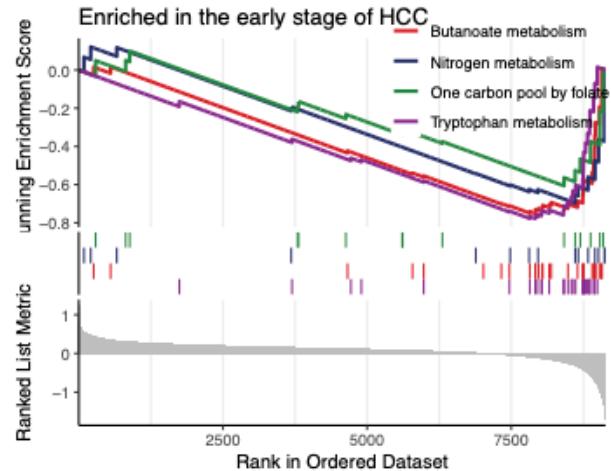


图 35.4: Figure 3C

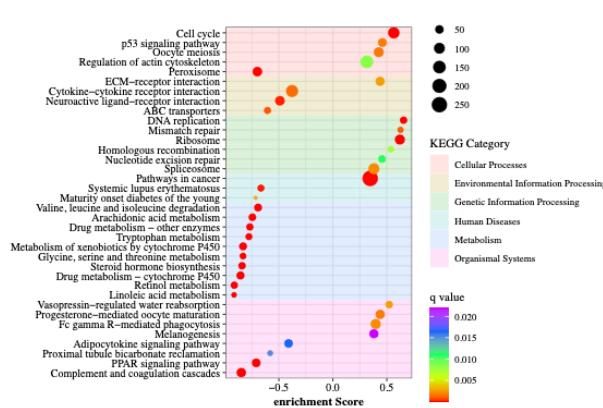


图 35.5: Figure 3B

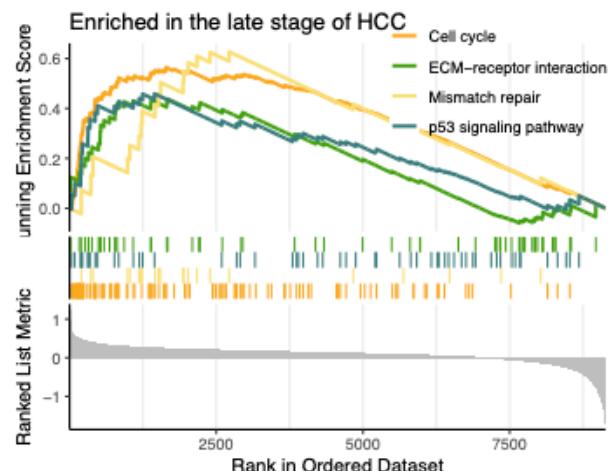


图 35.6: Figure 3D

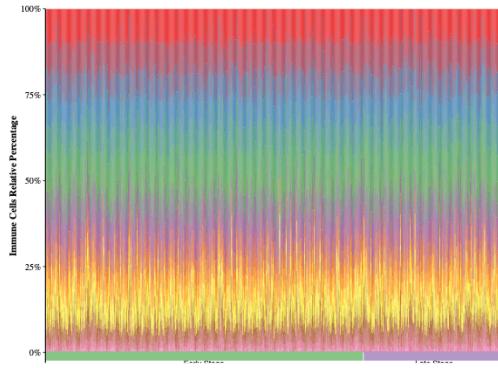


图 35.7: Figure 4A

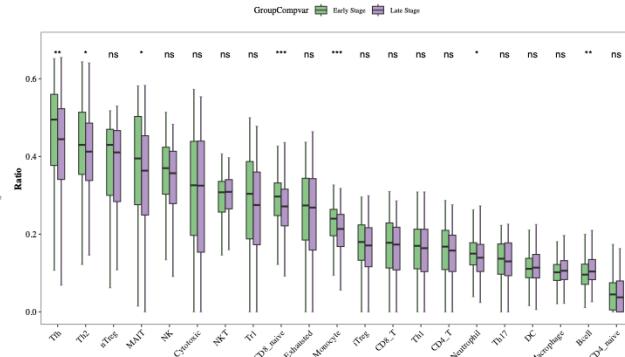


图 35.8: Figure 4B

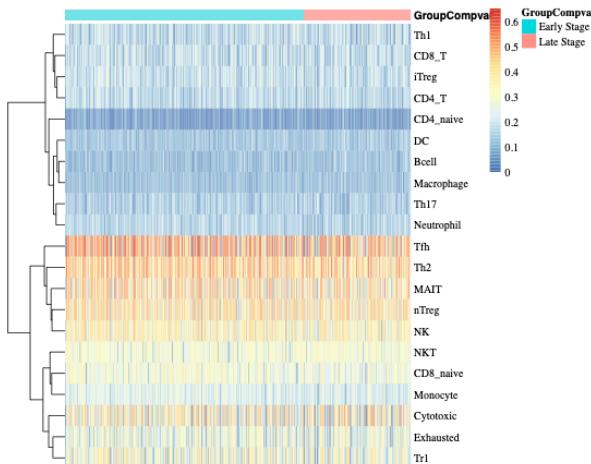


图 35.9: Figure 4C

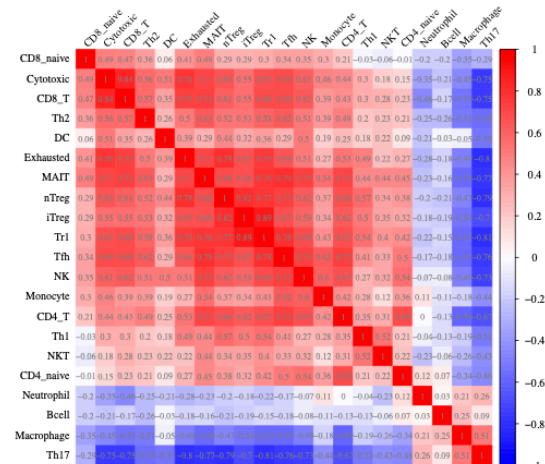


图 35.10: Figure 4D

©Hua

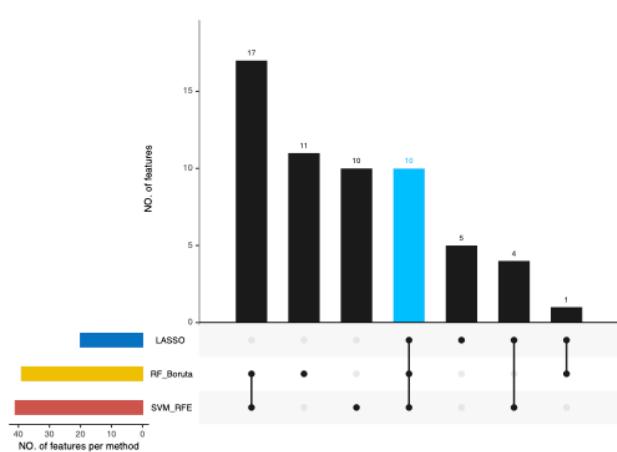


图 35.11: Figure 5A1

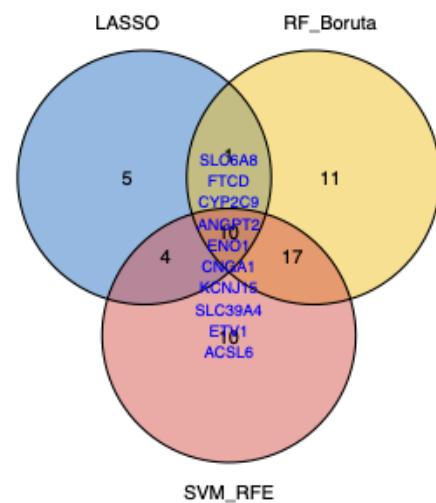


图 35.12: Figure 5A2

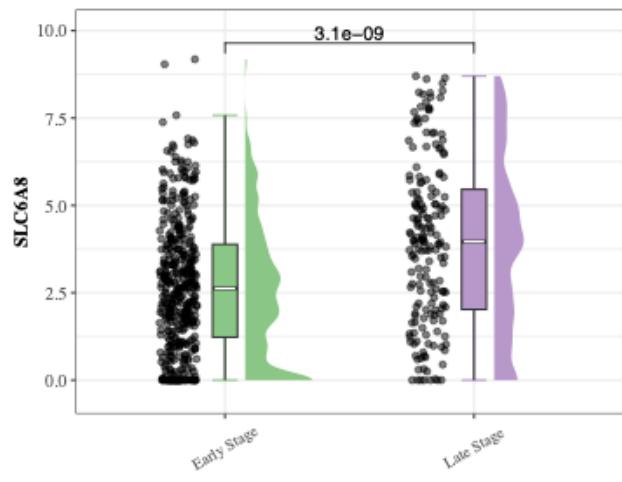


图 35.13: Figure 5B

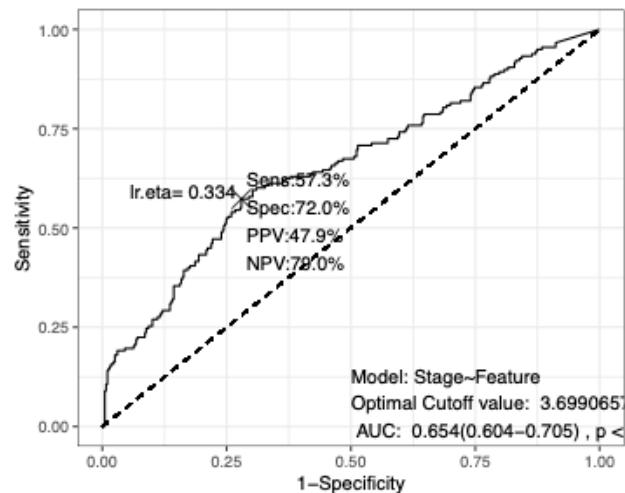


图 35.14: Figure 5D

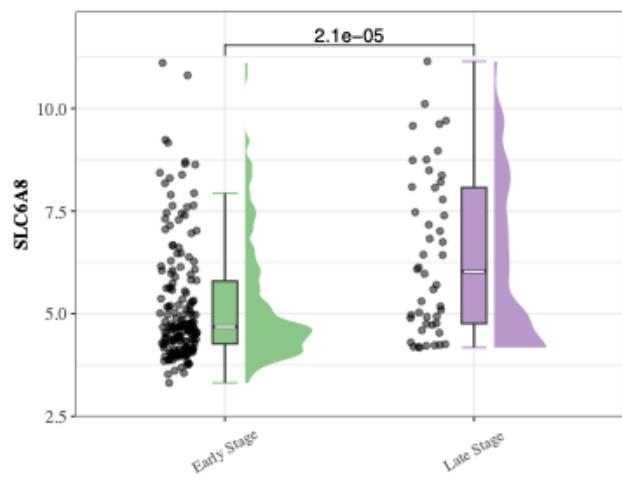


图 35.15: Figure 5C

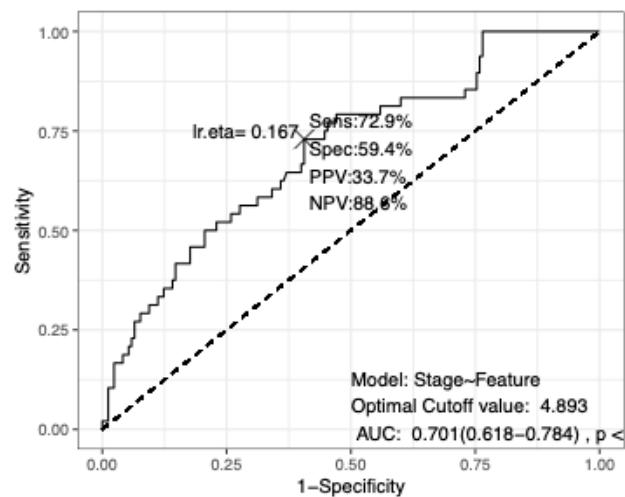


图 35.16: Figure 5E

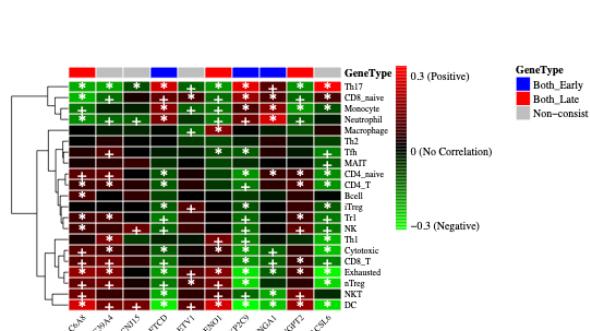


图 35.17: Figure 6A

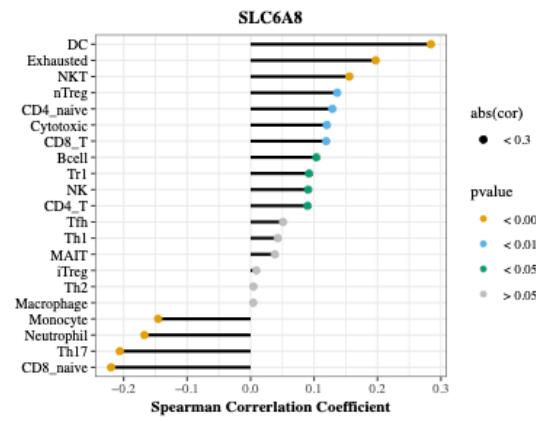


图 35.18: Figure 6B

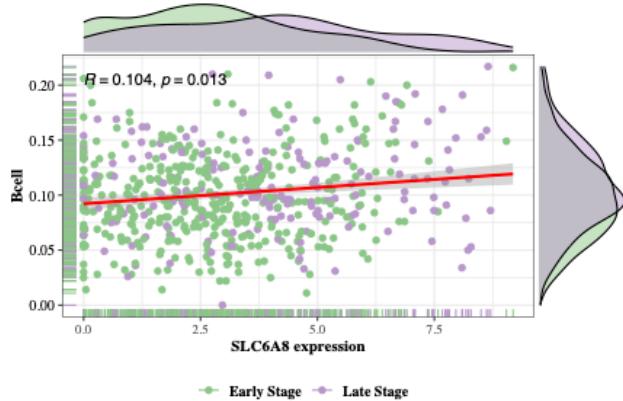


图 35.19: Figure 6C

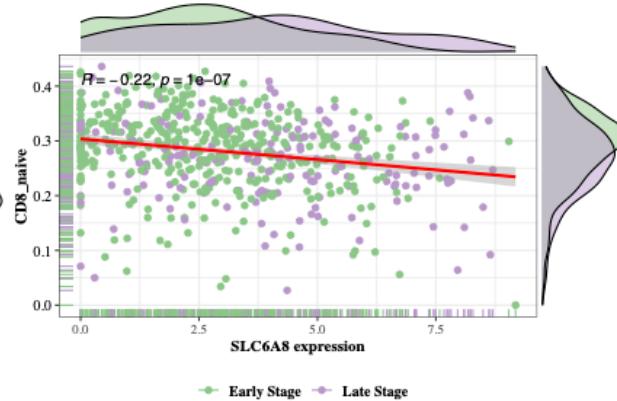


图 35.20: Figure 6D

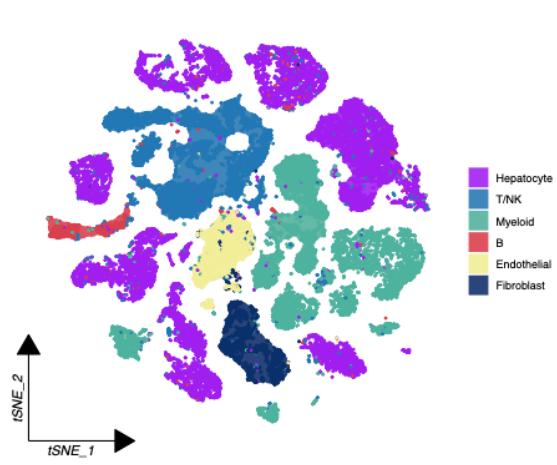


图 35.21: Figure 7A

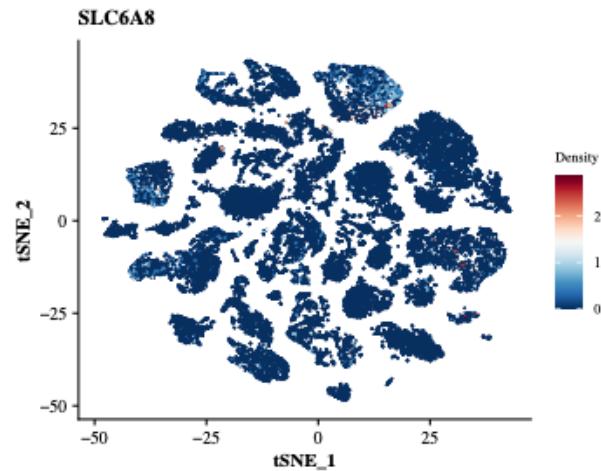


图 35.22: Figure 7B

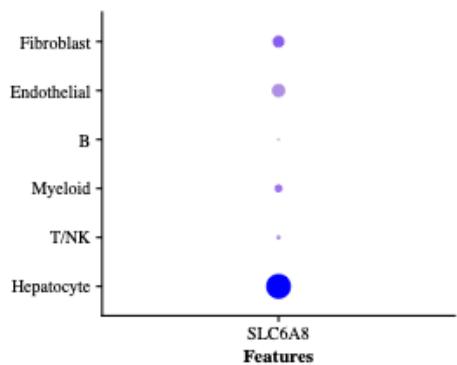


图 35.23: Figure 7C

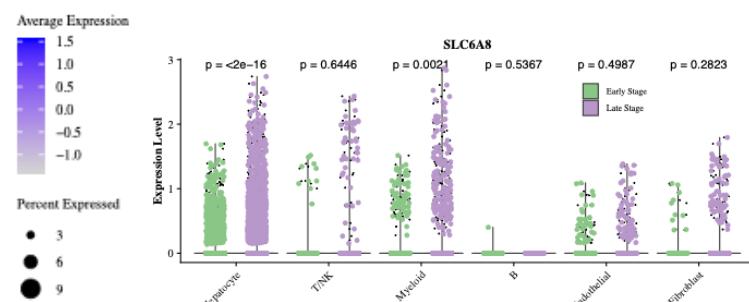


图 35.24: Figure 7D

References

- Anderson, Marti J. 2014. «Permutational multivariate analysis of variance (PERMANOVA)» . *Wiley statsref: statistics reference online*, 1–15.
- Andreatta, Massimo, 和 Santiago J Carmona. 2021. «UCell: Robust and scalable single-cell gene signature scoring» . *Computational and structural biotechnology journal* 19: 3796–98.
- Conway, Jake R, Alexander Lex, 和 Nils Gehlenborg. 2017. «UpSetR: an R package for the visualization of intersecting sets and their properties» . *Bioinformatics* 33 (18): 2938–40.
- Dean Attali, Christopher Baker. 2019. «ggExtra: Add Marginal Histograms to 'ggplot2', and More 'ggplot2' Enhancements» . <https://github.com/dattali/ggExtra>.
- Dolgalev, Igor. 2020. «msigdbr: MSigDB gene sets for multiple organisms in a tidy data format» . *R package version 7* (1).
- Friedman, Jerome, Trevor Hastie, 和 Rob Tibshirani. 2010. «Regularization paths for generalized linear models via coordinate descent» . *Journal of statistical software* 33 (1): 1.
- Gu, Zuguang, Roland Eils, 和 Matthias Schlesner. 2016. «Complex heatmaps reveal patterns and correlations in multidimensional genomic data» . *Bioinformatics* 32 (18): 2847–49.
- Gu, Zuguang, Lei Gu, Roland Eils, Matthias Schlesner, 和 Benedikt Brors. 2014. «”Circlize” implements and enhances circular visualization in R» .
- Hänzelmann, Sonja, Robert Castelo, 和 Justin Guinney. 2013. «GSVA: gene set variation analysis for microarray and RNA-seq data» . *BMC bioinformatics* 14: 1–15.
- Hao, Yuhan, Tim Stuart, Madeline H Kowalski, Saket Choudhary, Paul Hoffman, Austin Hartman, Avi Srivastava, 等. 2024. «Dictionary learning for integrative, multimodal and scalable single-cell analysis» . *Nature biotechnology* 42 (2): 293–304.
- Harrell Jr, Frank E, 和 Maintainer Frank E Harrell Jr. 2019. «Package ‘hmisc’» . *CRAN2018* 2019: 235–36.
- Huber, Wolfgang, Vincent J Carey, Robert Gentleman, Simon Anders, Marc Carlson, Benilton S Carvalho, Hector Corrada Bravo, 等. 2015. «Orchestrating high-throughput genomic analysis with Bioconductor» . *Nature methods* 12 (2): 115–21.
- Kay, Matthew. 2023. «ggdist: Visualizations of Distributions and Uncertainty in the Grammar of Graphics» . *IEEE Transactions on Visualization and Computer Graphics*.
- Kolde, Raivo 等. 2019. «Pheatmap: pretty heatmaps» . *R package version 1* (2): 726.
- Kuhn, Max. 2008. «Building predictive models in R using the caret package» . *Journal of statistical*

- software 28: 1–26.
- Kursa, Miron B, 和 Witold R Rudnicki. 2010. «Feature selection with the Boruta package» . *Journal of statistical software* 36: 1–13.
- Leek, Jeffrey T, 和 John D Storey. 2007. «Capturing heterogeneity in gene expression studies by surrogate variable analysis» . *PLoS genetics* 3 (9): e161.
- Liaw, Andy, Matthew Wiener, 等. 2002. «Classification and regression by randomForest» . *R news* 2 (3): 18–22.
- Lyu, Yulin. 2021. «ggunchull: A ggplot extension for drawing smooth non-convex irregular hulls around groups of points» . <https://github.com/sajuukLyu/ggunchull>.
- Mecham, Brigham H, Peter S Nelson, 和 John D Storey. 2010. «Supervised normalization of microarrays» . *Bioinformatics* 26 (10): 1308–15.
- Miao, Ya-Ru, Qiong Zhang, Qian Lei, Mei Luo, Gui-Yan Xie, Hongxiang Wang, 和 An-Yuan Guo. 2020. «ImmuneCellAI: a unique method for comprehensive T-cell subsets abundance prediction and its application in cancer immunotherapy» . *Advanced science* 7 (7): 1902880.
- Moon, Keon-Woong. 2019. «The multipleROC package» . <https://github.com/cardiomoon/multipleROC>.
- Poore, Gregory D, Evguenia Kopylova, Qiyun Zhu, Carolina Carpenter, Serena Fraraccio, Stephen Wandro, Tomasz Kosciolek, 等. 2020. «Microbiome analyses of blood and tissues suggest cancer diagnostic approach» . *Nature* 579 (7800): 567–74.
- Ritchie, Matthew E, Belinda Phipson, DI Wu, Yifang Hu, Charity W Law, Wei Shi, 和 Gordon K Smyth. 2015. «limma powers differential expression analyses for RNA-sequencing and microarray studies» . *Nucleic acids research* 43 (7): e47–47.
- Robin, Xavier, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, 和 Markus Müller. 2011. «pROC: an open-source package for R and S+ to analyze and compare ROC curves» . *BMC bioinformatics* 12: 1–8.
- Sarkar, Deepayan. 2008. *lattice: Multivariate Data Visualization with R*. New York: Springer. <http://lmdvr.r-forge.r-project.org>.
- Shao, Xin, Jie Liao, Xiaoyan Lu, Rui Xue, Ni Ai, 和 Xiaohui Fan. 2020. «scCATCH: automatic annotation on cell types of clusters from single-cell RNA sequencing data» . *Iscience* 23 (3).
- Shen, Xiaotao, Hong Yan, Chuchu Wang, Peng Gao, Caroline H Johnson, 和 Michael P Snyder. 2022. «TidyMass an object-oriented reproducible analysis framework for LC–MS data» . *Nature Communications* 13 (1): 4365.
- Sturm, Gregor, Francesca Finotello, 和 Markus List. 2020. «Immunedeconv: an R package for unified access to computational methods for estimating immune cell fractions from bulk RNA-sequencing data» . *Bioinformatics for cancer immunotherapy: methods and protocols*, 223–32.
- Tiedemann, Frederik. 2022. «The gghalves package» . <https://github.com/erocoar/gghalves>.
- van den Brand, Teun. 2023. *ggh4x: Hacks for 'ggplot2'*. <https://github.com/teunbrand/ggh4x>.
- Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. 2nd 本. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.
- Wilke, Claus O. 2019. «cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'» . <https://ggplot2.tidyverse.org>.

- //github.com/wilkelab/cowplot.
- Xie, Yihui. 2015. *Dynamic Documents with R and knitr*. 2nd 本. Boca Raton, Florida: Chapman; Hall/CRC. <https://yihui.org/knitr/>.
- Yan, Linlin, 和 Maintainer Linlin Yan. 2021. «Package 〈ggvenn.〉» Google Scholar.
- Yan, Yachen. 2016. «MLmetrics: Machine Learning Evaluation Metrics» . <https://github.com/yanchen/MLmetrics>.
- Yang, Yuwei, Yan Cao, Xiaobo Han, Xihui Ma, Rui Li, Rentao Wang, Li Xiao, 和 Lixin Xie. 2023. «Revealing EXPH5 as a potential diagnostic gene biomarker of the late stage of COPD based on machine learning analysis» . *Computers in Biology and Medicine* 154: 106621.
- Yu, Guangchuang. 2014. «ggplotify: Convert Plot to 'grob' or 'ggplot' Object» . <https://github.com/GuangchuangYu/ggplotify>.
- Yu, Guangchuang, Li-Gen Wang, Yanyan Han, 和 Qing-Yu He. 2012. «clusterProfiler: an R package for comparing biological themes among gene clusters» . *Omics: a journal of integrative biology* 16 (5): 284–87.
- Zhang, Jun. 2022. «GseaVis: An Implement R Package to Visualize GSEA Results» . <https://github.com/junjunlab/GseaVis>.
- Zou, Hua. 2022. «MicrobiomeAnalysis: An R package for analysis and visualization in metagenomics» . <https://github.com/HuaZou/MicrobiomeAnalysis/>.

附录 A 补充知识

A.1 数据库

i Pay Attention

持续更新生物医学研究相关的数据库列表...

这些数据库和资源是生物医学研究中常用的工具，它们提供不同类型的数据和分析服务，帮助研究人员在各自的领域中找到创新点和进行深入研究。下面是对每个数据库的简要介绍，包括它们的网址和主要作用：

- [PubMed](#): PubMed 是一个免费的搜索引擎，主要提供生物医学和生命科学领域的文献检索服务。它包含大量的期刊文章、会议记录、书籍章节等，是查找最新研究和文献的重要资源。

A.2 R 学习材料

互联网上有很多 R 语言学习材料，本书仅仅列出部分：

- 首推四川师范大学研究生公选课[数据科学中的 R 语言](#)。

A.3 R 与统计

- [应用统计学与 R 语言实现学习笔记](#)，适合结合 R 语言了解统计概念。

索引

Quarto, 1