R语言绘图基础

Fundamentals of R Plotting

主讲人：张陌尘

国家生物信息中心版权所有

邮箱：zhangmochen2019d@big

主讲时间：2025年7月10日

汇报地点：浙江海宁

目录 | CONTENTS

国家生物信息中心版权所有

01 R语言与R studio入门

国家生物信息中心版权所有

R语言概览

R语言是一种用于统计分析、数据可视化和科学计算的开源编程语言和软件环境，广泛应用于生物信息学分析

起源与发展

- 1990年代中期由 Ross Ihaka 与 Robert Gentleman 开发
- 名称源自其前身“S”语言及两位作者姓名首字母

核心优势

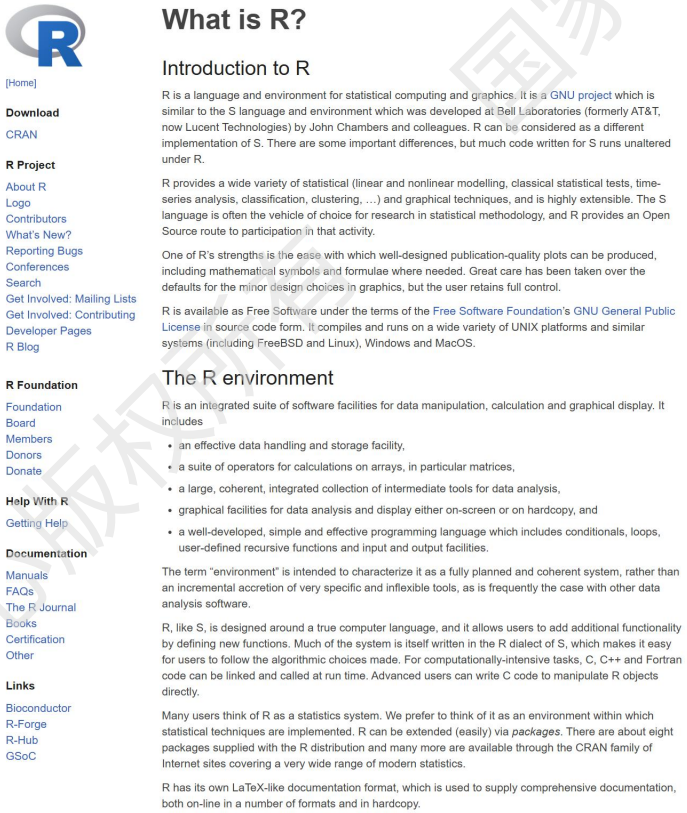
- 丰富的统计分析函数（回归、时序、生存分析等）
- 原生数据结构：向量、矩阵、列表、数据框

生态环境与扩展

- CRAN：超过 2 万个拓展包（如 ggplot2、Bioconductor）
- R Markdown 与 Shiny 支持可重现报告与交互式应用

典型应用场景

- 生物信息学
- 数据科学与机器学习
- 金融建模与社会科学分析


[Home]

Download
CRAN

R Project
About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Get Involved: Contributing
Developer Pages
R Blog

R Foundation
Foundation
Board
Members
Donors
Donate

Help With R
Getting Help

Documentation
Manuals
FAQs
The R Journal
Books
Certification
Other

Links
Bioconductor
R-Forge
R-Hub
GSoC

What is R?

Introduction to R

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

The R environment

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via *packages*. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.

R语言的官方网站：
http://www.r-project.org/

R语言的特点

R语言拥有丰富的内置统计函数、灵活的数据结构和庞大的扩展生态，支持可重现研究与交互式数据探索

开源自由	统计功能丰富	可视化强大	数据结构多样	生态系统庞大	交互与扩展
R采用GPL协议，用户可以免费获取、使用和修改源代码	支持时间序列、生存分析和贝叶斯建模等统计分析	提供灵活的低级绘图接口，满足自定义需求	向量与矩阵高效处理数值计算任务	Bioconductor 专门支持多种生物信息学工作流	Shiny框架可快速构建交互式Web应用，可视化程度高
社区驱动的开发模式确保新功能和补丁能够快速合并，透明度高、无厂商锁定	内置从基本描述统计到高级回归、方差分析、聚类、主成分分析等上百种方法	ggplot2 实现“语法图形学”，通过分层语法快速生成美观的统计图	列表（list）可嵌套任意类型对象，适合组织和管理多层次、结构化的复杂分析结果	社区活跃，定期举办 R 会议和用户组（UseR!、R-Ladies）分享心得	RStudio IDE 集成代码补全、调试、可视化预览和 Git 支持，极大提升开发效率
拥有全球活跃的用户社区和贡献者网络，新手和专家都可在论坛、邮件列表和Stack Overflow 上获得支持	强大的扩展包如 forecast、survival 和 bayesm 提供专业领域的深度统计工具，满足复杂科研需求	plotly、leaflet 等包支持交互式 Web 可视化，无缝嵌入 Shiny 应用或 R Markdown	数据框（data.frame）和 tibble 支持异质列，方便进行分组、汇总和管道式数据处理	CRAN 上超过 2 万个包，涵盖数据清洗（dplyr）、可视化（ggplot2）、机器学习（caret、mlr3）等	支持与 Python、C++、SQL 等多语言互操作，利用 reticulate、Rcpp 等包扩展性能和功能

R扩展软件包的安装与管理

R包管理涵盖 CRAN/Bioconductor 安装、GitHub 私有源获取、renv/packrat 依赖隔离与命名空间加载，保障科研流程中扩展生态的高效可重现

从CRAN安装与卸载 01

1. 安装最新稳定版本

- install.packages("包名")

2. 指定镜像源 -> 加速国内下载（清华镜像）

- install.packages("包名",repos="https://mirrors.tuna.tsinghua.edu.cn/CRAN/")

3. 卸载包

- remove.packages("包名")

4. 查看已安装包

- installed.packages() -> 返回所有包和版本信息
- sessionInfo()$otherPkgs ->当前会话已加载的包

5. 更新包

- update.packages(ask=FALSE) ->更新所有过时包

Bioconductor平台 02

1. 安装BiocManager（仅需一次）

- install.packages("BiocManager")

2. 通过BiocManager安装/更新

（1）安装单个包

- BiocManager::install("DESeq2")

（2）批量安装多个包

- BiocManager::install(c("edgeR","limma"))

（3）更新已安装的所有的Bioconductor包

- BiocManager::install()

3. 常用生信包示例

- GenomicRanges：基因组区间处理
- AnnotationDbi：注释数据库接口

GitHub平台 03

1. 使用remotes包

- install.packages("remotes")
- remotes::install_github("用户名/仓库名")

2. 使用devtools包

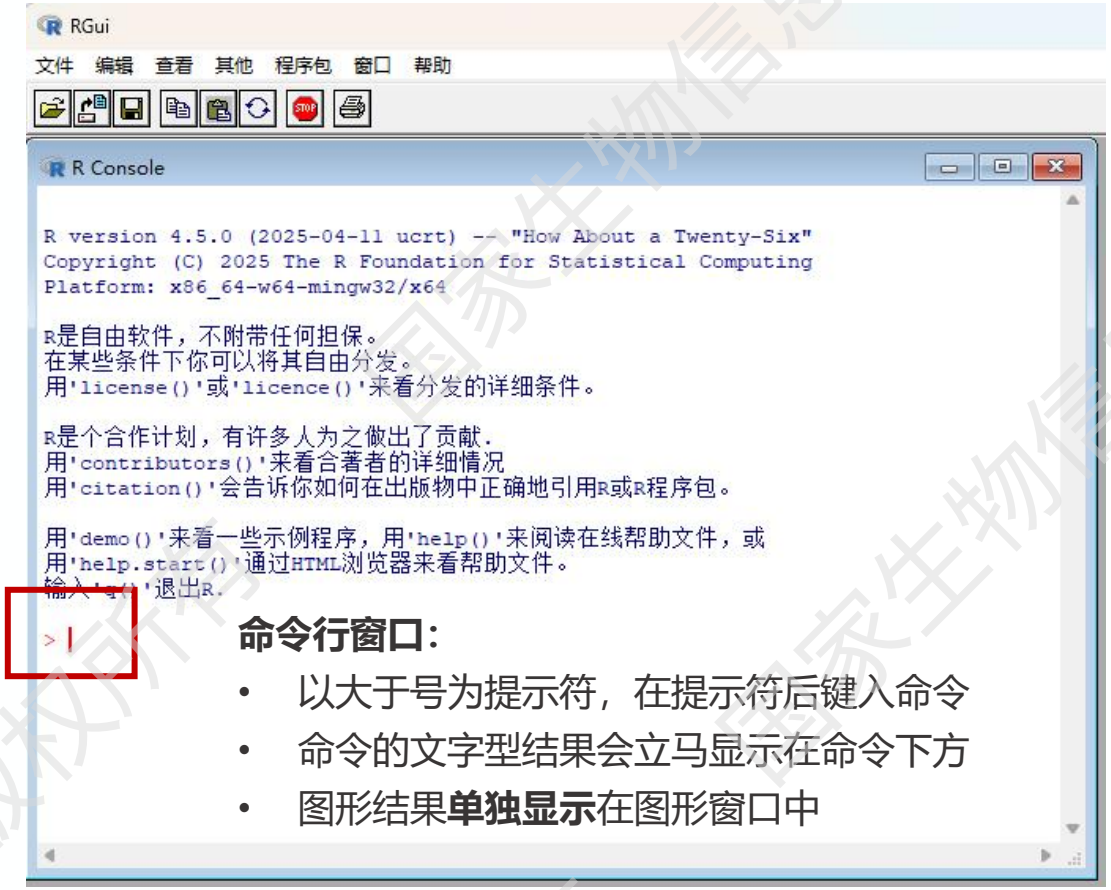
- install.packages("devtools")
- devtools::install_github("用户名/仓库名", ref="branch_or_tag")

3. 安装私有仓库或子目录

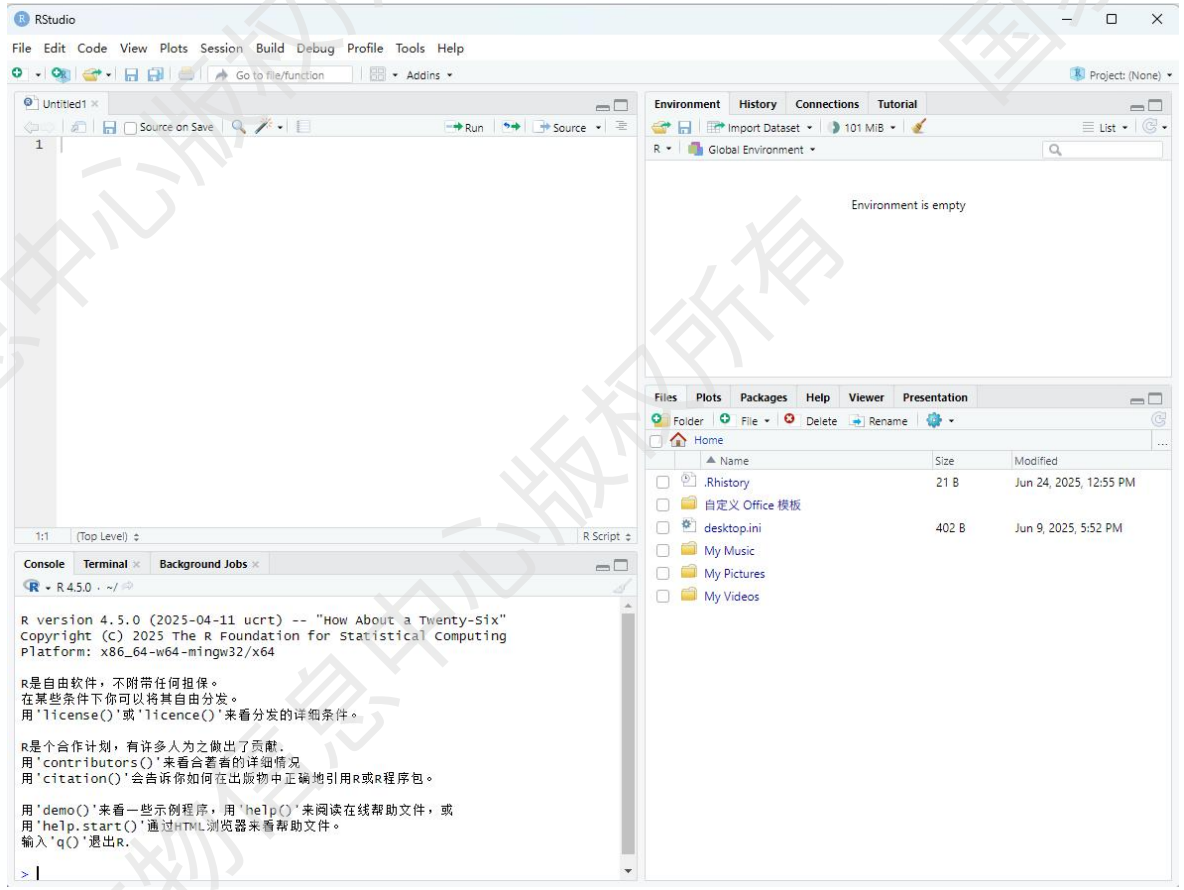
- remotes::install_git("git@github.com:org/privatepkg.git")
- remotes::install_github("org/repo/subdir/pkg")

为什么选择R studio?

RStudio提供了集成的开发环境，将代码编辑、控制台、变量/文件浏览和可视化输出无缝整合，并配备智能补全、调试工具与版本控制接口，大幅提升 R 语言脚本的编写、调试和项目管理效率



图形窗口模式的R软件（R GUI）



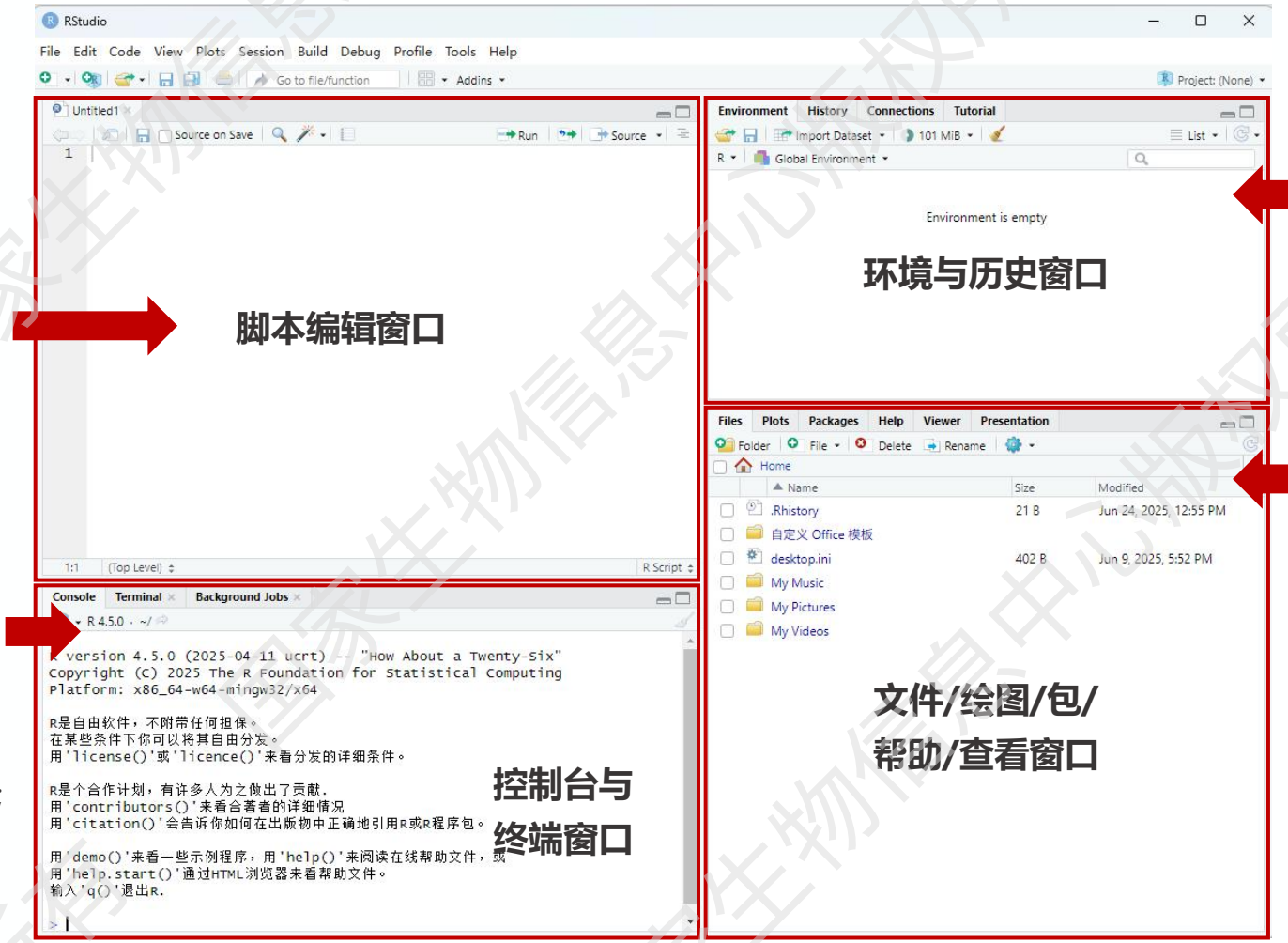
R Studio软件（软件的应用界面与增强系统）

R studio界面四大窗口

通过脚本编辑、控制台与终端、环境与历史以及文件/绘图/包/帮助/查看这四个互联窗口，R Studio 将脚本编辑、命令交互、环境管理和可视化展示无缝集成，大幅提升了 R 语言开发与分析的效率与体验

- 用于编写、打开和管理 R 脚本、R Markdown 文档或函数文件；
- 顶部工具栏提供“Run”（运行当前行/选中代码）、“Source”（执行整个脚本）等按钮；
- 支持**语法高亮、代码折叠和自动补全。**

- **Console**：直接与 R 会话交互的地方，可以手动输入 R 命令并立即执行，适合临时调试；
- **Terminal**：提供普通 shell 访问，允许在 RStudio 内执行系统命令；
- **Background Jobs**：显示使用future等包提交的后台任务状态。

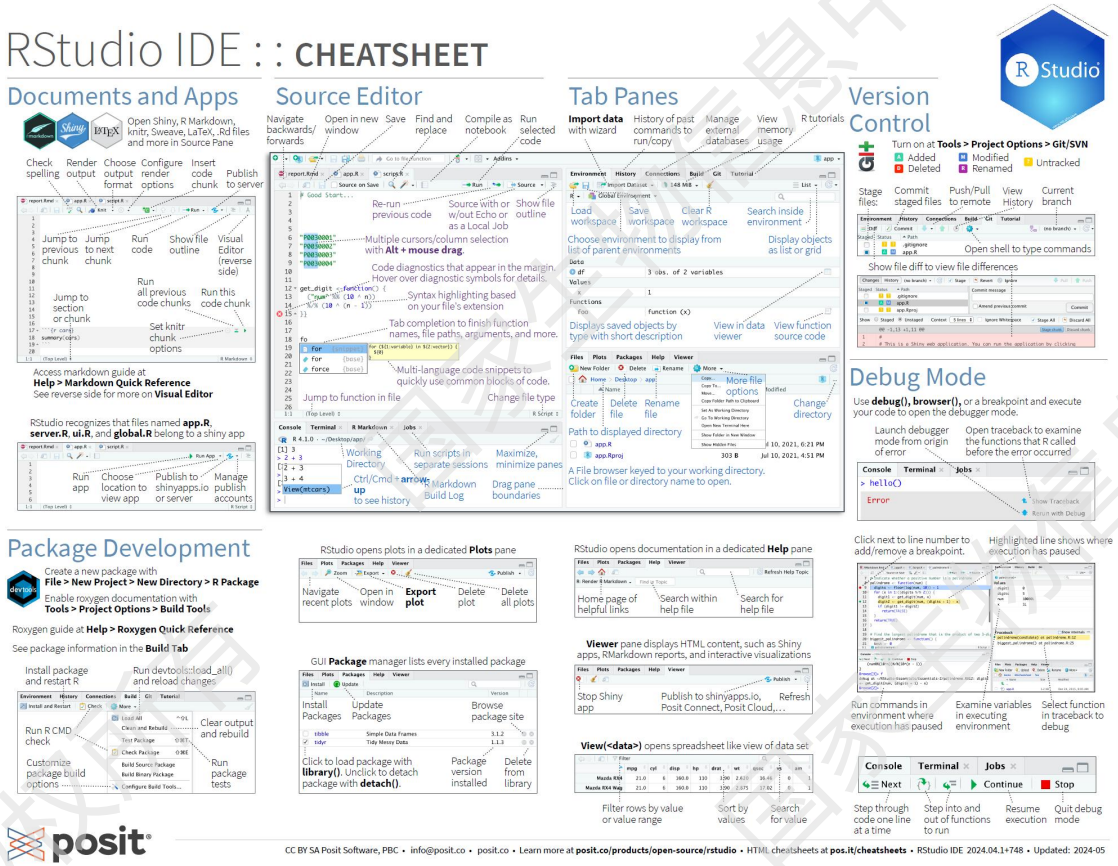


- **Environment**：列出当前 R 会话中所有对象，可点击名称查看其内容或删除；
- **History**：记录所有在控制台执行过的命令，支持筛选和重新发送到控制台；
- **Connections/Tutorial** 等标签可扩展数据库连接或内置教学。

- **Files**：浏览当前工作目录下的文件和文件夹；
- **Plots**：展示执行plot() 等函数生成的图形，配备导出图片的按钮；
- **Packages**：列出已安装的R包，可一键安装、卸载、加载或更新；
- **Help**：搜索并阅读文档与手册；
- **Viewer**：呈现HTML报告/Shiny应用的网页视图。

R studio核心功能一览

RStudio IDE :: Cheatsheet 提供了丰富的功能模块概览、常用快捷键与操作示例，供大家快速查询和参考



下载链接：https://rstudio.github.io/cheatsheets/html/rstudio-ide.html?_gl=1*x77lx7*_ga*MTg5MjQ4MTIwMi4xNzQ4OTE3ODg5*_ga_2C0WZ1JHG0*czE3NDg5MTc4ODkkbzEkZzEkdDE3NDg5MTgxMjkkajU4JGwwJGgw

变量赋值与索引方法

基本赋值符号

<-：R 社区推荐的赋值运算符

=：在函数调用中常用，也可用于全局赋值

->：少见的反向赋值

命名规则

- 变量名区分大小写，以字母或 . 开头（但不以数字或下划线 _ 开头）
- 推荐使用驼峰（geneList）或下划线（gene_list）风格，避免与函数名冲突。

环境与作用域

- 函数内部用 localVar <- ... 定义的变量仅在该函数作用域内可见
- 可用 ls() 或 RStudio Environment 窗格查看当前环境中的变量列表

变量赋值

索引方法

向量

- 整数下标：v[2]　　v <- c(5, 10, 15, 20)
- 逻辑下标：v[v > 10]
- 名称下标：v2 <- c(A=1, B=2, C=3) ->v2["B"]
- 切片与排除：v[2:4]

列表

- [[ ]]提取单个元素：lst[[1]]　　lst <- list(name= "Alice", scores=c (90,85))
- [ ]返回子列表：lst[1] - > list(name = "Alice")
- $ 提取命名元素：lst$name

数据框

- 按列名索引：df[[" Expr "]]和df$Expr
- 按行列下标：df[1,2]
- 逻辑筛选：df[df$Expr > 4,]

国家生物信息中心版权所有

数据类型之向量

向量（Vector）是R中最基本的数据结构，表示同一类型（数值、字符或逻辑值）的有序元素集合

向量 四类创建方式

- **最常用的c()函数**

```
> v1 <- c(2, 4, 6, 8)
> v1
[1] 2 4 6 8
```

- **生成序列**

```
> v2 <- 1:5                 # 1,2,3,4,5
> v3 <- seq(0, 1, by=0.2)
> v2
[1] 1 2 3 4 5
> v3
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

- **重复元素**

```
> v4 <- rep("A", times=3)      # "A" "A" "A"
> v5 <- rep(c(TRUE, FALSE), each=2)
> v4
[1] "A" "A" "A"
> v5
[1]  TRUE  TRUE FALSE FALSE
```

- **类型转换**

```
> v6 <- as.numeric(c("1","2","3"))
> v6
[1] 1 2 3
```

向量 索引与子集

- **整数下标**

v1[2]则返回第二个元素

v1[c(1,3)]则返回第一个和第三个元素

- **逻辑下标**

v1[v1>5]返回所有大于5的元素

- **名称下标（命名向量）**

v7 <- c(A=10, B=20, C=30)

v7["B"]　　# 20

- **切片操作**

v1[2:4] 返回第 2–4 个元素

v1[-1] 返回除第 1 个以外的所有元素

向量 常见运算

- **算数运算**

对向量执行 + - * / ^ 时，R 会按元素逐一计算，并且支持“矢量化”循环。

- **聚合函数**

sum(v1)、min(v1)、max(v1)

mean(v1)、sd(v1)

- **排序与排名**

sort(v1)、order(v1)、rank(v1)

- **集合运算**

union(a,b)、intersect(a,b)

setdiff(a,b)、unique(v1)

国家生物信息中心版权所有

数据类型之矩阵

矩阵（Matrix）是R中的二维同质数据结构，所有元素必须为相同类型（通常为数值）

矩阵 四类创建方式

- **matrix()函数**

```
> # 1. matrix() 函数
> m1 <- matrix(1:9, nrow=3, ncol=3)
> m1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

- **指定填充方向**

```
> # 2. 指定填充方向
> m2 <- matrix(1:6, nrow=2, byrow=TRUE)
> #      [,1] [,2] [,3]
> # [1]    1    2    3
> # [2]    4    5    6
> m2
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

- **将向量转为矩阵**

```
> # 3. 将向量转为矩阵
> v <- 1:12
> m3 <- array(v, dim = c(3,4))  # 与 matrix(v,3,4) 等价
> m3
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

- **行合并/列合并**

```
> # 4. 行合并/列合并
> r1 <- c(5,10,15); r2 <- c(20,25,30)
> m4 <- rbind(r1, r2)  # 2x3
> c1 <- c(1,2); c2 <- c(3,4); c3 <- c(5,6)
> m5 <- cbind(c1, c2, c3)  # 2x3
> m4
    [,1] [,2] [,3]
r1     5   10   15
r2    20   25   30
> m5
     c1 c2 c3
[1,]  1  3  5
[2,]  2  4  6
```

矩阵 索引与子集

- **单元素**

m1[2,3] 取第2行第3列

- **整行/整列**

m1[1,]、m1[, "S2"]

- **子矩阵**

m1[1:2, 2:3]

- **逻辑索引**

m1[m1 > 5] 返回所有大于5的元素（展平向量）

```
> print(m1)
   S1 S2 S3
G1  1  4  7
G2  2  5  8
G3  3  6  9
> # — (4) Indexing & Subsetting — #
>
> # Single element: row 2, column 3
> print("m1[2, 3]:")
[1] "m1[2, 3]:"
> print(m1[2, 3])
[1] 8

> # Entire row or column
> print("m1[1, ] (row 1):")
[1] "m1[1, ] (row 1):"
> print(m1[1, ])
S1 S2 S3
 1  4  7
> print("m1[, 'S2'] (column 'S2'):")
[1] "m1[, 'S2'] (column 'S2'):"
> print(m1[, "S2"])
G1 G2 G3
 4  5  6

> # Submatrix: rows 1-2, columns 2-3
> print("m1[1:2, 2:3]:")
[1] "m1[1:2, 2:3]:"
> print(m1[1:2, 2:3])
   S2 S3
G1  4  7
G2  5  8

> # Logical indexing: all elements > 5 (returns a flattened vector)
> print("m1[m1 > 5]:")
[1] "m1[m1 > 5]:"
> print(m1[m1 > 5])
[1] 6 7 8 9
>
```

矩阵 常见运算

- **对角元素与对角矩阵**

提取对角线元素：diag(m1)；

构建对角矩阵：diag(c(1,2,3)) # 生成3×3对角矩阵

- **行/列绑定**

将多个矩阵按行拼接：rbind(m1, m2)；

按列拼接：cbind(m1, m2)

- **行列子集与切片**

提取子矩阵：m1[1:2, 3:4] （第一、二行与第三、四列）

排除某行或某列：m1[-1,] （去掉第一行）、m1[, -3]（去掉第三列）

国家生物信息中心版权所有

数据类型之列表

列表（list）是 R 中最通用的数据容器，可以包含任意类型的对象 - 向量、矩阵、数据框、函数、甚至其他列表

```
> # 1. 创建一个列表：包含数值、字符和向量
> lst <- list(
+   id     = "Sample01",
+   counts = c(100, 200, 150),
+   meta   = data.frame(tissue=c("Liver","Heart","Brain"), n=3)
+ )
> lst
$id
[1] "Sample01"

$counts
[1] 100 200 150

$meta
  tissue n
1  Liver 3
2  Heart 3
3  Brain 3

>
> # 2. 按名称访问
> lst$id        # "Sample01"
[1] "Sample01"
> lst[["counts"]]  # c(100,200,150)
[1] 100 200 150
>
> # 3. 按位置访问
> lst[[2]]     # 等同于 lst$counts
[1] 100 200 150
>
> # 4. 返回子列表
> lst[1:2]     # 仍是一个含前两元素的列表
$id
[1] "Sample01"

$counts
[1] 100 200 150

>
> # 5. 添加或修改元素
> lst$qc <- TRUE          # 新增布尔型元素
> lst[["meta"]] <- NULL  # 删除名为 meta 的元素
> lst
$id
[1] "Sample01"

$counts
[1] 100 200 150

$qc
[1] TRUE
```

创建列表

```
lst <- list(
  id     = "Sample01",
  counts = c(100, 200, 150),
  meta   = data.frame(tissue=c("Liver","Heart","Brain"), n=3)
) -> # 创建包含数值、字符和向量的列表
```

访问方式

遍历列表

```
lapply(lst, class)
```

合并多个列表：

```
lst2 <- list(status="OK", date=Sys.Date())
merged <- c(lst, lst2)
```

列表扁平化：

```
nested <- list(a=1, b=list(b1=2,b2=3))
purrr::flatten(nested)
```

常用操作与函数

按名称访问

```
lst$id       # "Sample01"
lst[["counts"]]  # c(100,200,150)
```

按位置访问

```
lst[[2]]     # 等同于 lst$counts
```

返回子列表

```
lst[1:2]     # 仍是一个含前两元素的列表
```

添加或修改元素

```
lst$qc <- TRUE        # 新增布尔型元素
lst[["meta"]] <- NULL  # 删除名为 meta 的元素
```

国家生物信息中心版权所有

数据类型之数据框

数据框（data.frame）是一种二维表格结构，类似于数据库中的表或 Excel 中的工作表。每列可以存储不同类型的数据（数值、字符、因子等），行数和列数可动态变化。

```
> # 从向量创建
> df <- data.frame(
+   SampleID = c("S1","S2","S3"),
+   Count    = c(120, 85,  95),
+   Group    = factor(c("A","B","A"))
+ )
>
> # 显示结构（列名、类型与示例值）
> str(df)
'data.frame':   3 obs. of  3 variables:
 $ SampleID: chr  "S1" "S2" "S3"
 $ Count   : num  120 85 95
 $ Group   : Factor w/ 2 levels "A","B": 1 2 1
>
> # 预览前/后几行
> head(df)
  SampleID Count Group
1       S1   120     A
2       S2    85     B
3       S3    95     A
> tail(df)
  SampleID Count Group
1       S1   120     A
2       S2    85     B
3       S3    95     A
>
> # 返回行数与列数
> dim(df)
[1] 3 3
>
> # 列名列表
> names(df)
[1] "SampleID" "Count"    "Group"
```

创建数据框

```
# 从向量创建
df <- data.frame(
  SampleID = c("S1","S2","S3"),
  Count    = c(120, 85,  95),
  Group    = factor(c("A","B","A"))
)
```

查看方式

str(df)：显示结构（列名、类型与示例值）；

head(df) / tail(df)：预览前/后几行；

dim(df)：返回行数与列数；

names(df)：列名列表。

索引与子集

按列名或位置：

```
df$Count        # 提取 Count 列，返回向量
df[["Group"]]   # 同上
df[,2]          # 第 2 列
```

按行号或逻辑条件：

```
df[1, ]         # 第 1 行所有列
df[df$Count > 90, ]  # 筛选 Count 大于 90 的样本
```

混合下标：

```
df[2:3, c("SampleID","Count")]  # 指定行列
```

常用操作

增加/删除列：

```
df$Ratio <- df$Count / sum(df$Count)   # 新增列
df$Group <- NULL                       # 删除列
```

重命名列：

```
names(df)[2] <- "ReadCount"
```

行列绑定：

```
df2 <- rbind(df, data.frame(SampleID="S4")
df3 <- cbind(df, Score=c(8.5,7.2,9.1))
```

排序：

```
df[order(df$Count, decreasing=TRUE), ]
```

数据导入与导出

数据导入

文本文件

- read.table()
- read.csv() -> csv专用
- read.delim() -> 分隔符

快速读入大数据

- data.table::fread()
- data.table包配备的fread函数具备自动检测分隔符、并行读取的功能，对大型文件的读取非常友好。

Excel文件与R原生格式

- readxl::read_excel() ->excel格式
- readRDS() ->由saveRDS保存的单个R对象

数据导出

文本文件

- write.table()
- write.csv() -> csv专用

快速写入大数据

- data.table::fwrite()
- 备注：高速、支持压缩写入（.csv.gz）

Excel文件与R原生格式

- openxlsx::write.xlsx()
- saveRDS() -> 将任意R对象序列化为.rds文件

国家生物信息中心版权所有

本模块小结

R语言与R studio入门模块的主要目标是掌握R基本语法和RStudio使用，能够熟练地创建并操作各种基础数据类型、编写和调试简单脚本、使用台式交互式IDE高效管理项目、完成常见的文件读写任务

R语言基础

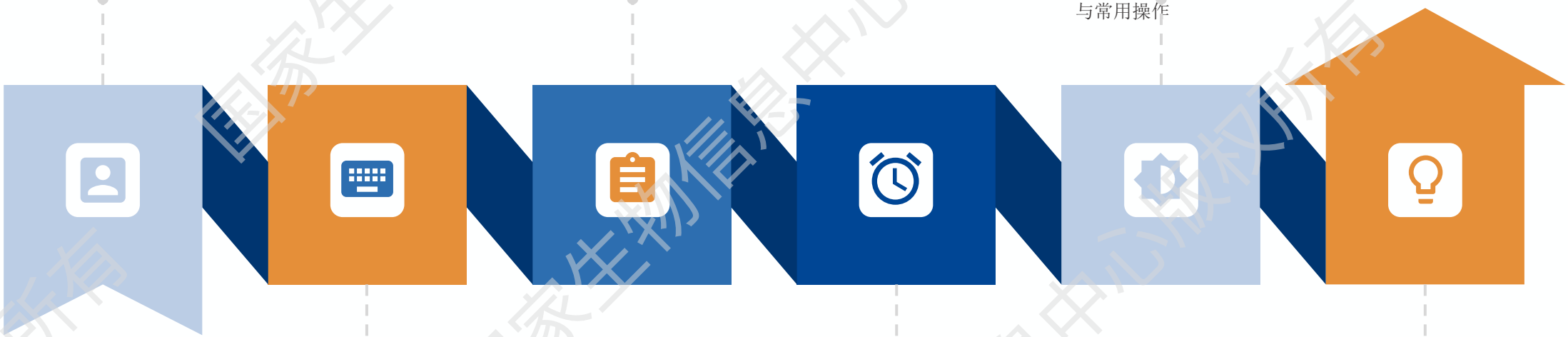
介绍 R 语言的定位、发展历史及生态优势，明确其向量化运算、函数式编程、丰富包资源等核心特性

R Studio核心功能

代码编辑器、控制台、环境/历史窗口、文件/帮助面板的使用技巧

数据类型与数据结构

向量（vector）、矩阵（matrix）、列表（list）、数据框（data.frame）的定义、创建与常用操作



软件安装与环境配置

明确R 及常用扩展包的下载；对比 R GUI，展示 R Studio 四大界面及核心功能一览

基本操作

变量赋值与索引：<-、= 赋值方法；[]、[[ ]]、$ 三种索引方式的区别与应用

数据导入与导出

使用多种函数从文本、CSV 或大数据文件中读取数据，并将处理结果导出

实战练习

练习 1：向量与基因表达统计

1. 从向量 expr <- c(G1=5.2, G2=3.8, G3=6.1, G4=2.9) 中提取表达量大于 5 的基因名称；
2. 计算该向量的平均表达（mean）、中位数（median）和标准差（sd）；
3. 将表达量以 10 倍放缩后，生成新的向量 expr10x。

练习 2：矩阵与批量计算

1. 构建一个 4×3 的表达矩阵 mat，行名为基因 G1–G4，列名为样本 S1–S3，元素随机取自正态分布 rnorm(12, mean=10, sd=2)；
2. 为矩阵添加行名与列名；
3. 计算每个基因（行）的平均表达和每个样本（列）的总表达，并以向量形式输出；
4. 对矩阵中所有表达值做 log2(x+1) 转换，生成新矩阵 mat_log2。

练习 3：列表组织复杂结果

1. 将上一步生成的 mat_log2、及下面的注释信息 anno <- data.frame(Gene=rownames(mat_log2),
   Length=c(1000,1500,1200,800))
   和样本分组信息 group <- rep(c("Ctrl","Treat"), each=2, length.out=3)
   一并保存到一个列表 bio_list <- list(expr=mat_log2, annotation=anno, group=group)；
2. 通过 $和[[ ]] 两种方式访问并展示 annotation 数据框；
3. 使用 lapply() 对 bio_list$expr 中每列执行 mean 计算，并将结果保存回列表中 bio_list$means。

实战要求及答案路径：/home/big/RBase/1-Introduction_to_R_Language_and_RStudio_Module

实战练习

练习 4：数据框操作与索引

1. 从工作目录下导入 CSV 文件 gene_expr.csv（包含列 GeneID, Sample1, Sample2, Group），并赋值给 df；
2. 查看数据结构（str、dim、head）并将 Group 列转换为因子（factor转换）；
3. 筛选出两组（Group=="Control" vs Group=="Treatment"）表达量差异最显著的前 5 个基因（按 abs(Sample1–Sample2)` 排序）；
4. 将筛选结果保存到新数据框 top5 并写出为 top5_diff.csv。

练习 5：数据导入导出综合

1. 使用 read.table() 从制表符分隔的 metadata.txt 导入样本元信息；
2. 将 df（练习 4 中导入的表达数据）与 metadata 按 GeneID 或 Sample 字段进行合并（merge）；
3. 将合并后的数据保存为 TSV 文件 merged_data.tsv（write.table(..., sep="\t")）；
4. 将 bio_list 列表以 RDS 格式保存为 bio_list.rds，并演示如何用 readRDS() 重新载入。

实战要求及答案路径：/home/big/RBase/1-Introduction_to_R_Language_and_RStudio_Module

02 数据整理与预处理操作

国家生物信息中心版权所有

dplyr包之filter

filter用于从数据框（data.frame/tibble）中提取满足逻辑条件的行，是dplyr中最常用的数据子集化操作之一

返回仅包含满足条件行的新数据框，原数据顺序保留

```
library(dplyr)
df_filtered <- df %>% filter(<logical expression>)
```

一个data.frame / tibble　　基于列名的条件，返回布尔向量

基本用法

① **根据单一条件筛选**

```
df %>% filter(Count1 > 100)
```

② **根据多个条件筛选（逻辑与&）**

```
df %>% filter(Count1 > 50, Group == "A")
```

③ **逻辑或 |**

```
df %>% filter(Group == "B" | Count1 < 50)
```

④ **取逻辑否定**

```
df %>% filter(! (Group == "A"))
```

复杂条件与函数配合

① **范围筛选**

```
df %>% filter(Count1 >= 50 & Count1 <= 150)
```

② **向量化函数**

```
df %>% filter(between(Count1, 50, 150)
```

③ **正则匹配 -> 标红部分用于按正则匹配 Gene 名称**

```
df %>% filter(str_detect(Gene, "^G[13]$"))
```

④ **在多个列上使用同一条件**

```
df %>%filter(if_all(starts_with("Count"), ~.> 20))
```

使用建议

① **先小后大**：对大数据集，先用 select() 保留必要列，再 filter()，减少内存开销；

② **链式调用**：与其它 dplyr 语句（mutate()、summarize()、arrange()）组合，构建清晰可读的数据处理管道；

③ **NA 处理**：filter() 默认会排除 NA（因为 NA > 10 返回 NA，不会被选中），如需保留或特别处理，可加条件 is.na()；

④ **调试技巧**：在管道中插入 print() 或 glimpse()，随时查看中间结果。

dplyr包之select

select用于从数据框（data.frame 或 tibble）中挑选（保留）指定的列；配合管道操作，可高效链式调用。该函数支持聚焦分析所需字段、删除冗余列、重命名列、按模式批量选择列等。

```
library(dplyr)
select(.data, ..., .keep = c("all", "unused", "used"))
```

返回仅包含满足条件行的新数据框，原数据顺序保留

输入的数据框　要保留的列名/辅助函数　根据keep中选择的选项，返回满足条件的数据框

辅助函数

① starts_with("prefix")：匹配以某前缀开头的列；

② ends_with("suffix")：匹配以某后缀结尾的列；

③ contains("str")：匹配列名中包含指定字符串；

④ matches("regex")：基于正则表达式匹配列名；

⑤ num_range("x", 1:3)：匹配 x1, x2, x3；

⑥ everything()：代表所有剩余列；

⑦ one_of(c(...))：匹配给定名称向量；

⑧ any_of(c(...))：类似 one_of()，但当某些列不存在时忽略警告。

使用示例

① **保留特定列**

```
df %>% select(id, score1, score2)
```

② **按前缀保留列**

```
df %>% select(starts_with("score"))
```

③ **排除某列**

```
df %>% select(-note)
```

④ **仅保留未选择的列**

```
df %>% select(starts_with("score"), .keep = "unused")
```

注意事项

① **顺序敏感**：select() 会按照你在 ... 中给出的顺序重排列；

② **重命名**：在 select() 内直接用 新名 = 旧名，无需额外调用 rename()；

③ **动态列名**：可结合 all_of()、any_of() 传入一个字符串向量，避免硬编码；

④ **与管道结合**：select() 常与 filter()、mutate()、arrange() 等其他 dplyr 函数组合，构建清晰的数据处理流程。

dplyr包之mutate

mutate() 用于向数据框（data.frame 或 tibble）中添加新列或修改已有列，并自动返回拥有更新后列的新数据框。该函数支持基于现有变量计算派生变量、数据标准化、分组累积计算、条件赋值等

```
library(dplyr)
mutate(.data, across(.cols, .fns, ...), .keep = c("all", "used", "unused", "none"))
```

输入的数据框　对多列一次性应用函数　　控制保留哪些列，与 select() 中 .keep 含义类似

配合函数

① **基本运算与条件：**

if_else(condition, true, false)：向量化条件选择。

case_when(cond1 ~ val1, cond2 ~ val2, TRUE ~ default)：多条件分支。

② **滚动与累积计算：**

lead(x, n) / lag(x, n)：取前/后值。

cumsum(x) / cummean(x) / cumprod(x)：累积和、均值、积。

③ **分组计算（需配合group_by()）：**

row_number()：组内行号。

n()：组内行数。

min_rank()、dense_rank()：组内排名。

使用示例

① **算术派生：新增 total 和 ratio 列**

```
df %>% mutate( total = score1 + score2, ratio = score1 / total )
```

② **条件赋值：根据 score1 生成 pass 列**

```
df %>% mutate(pass = if_else(score1 >= 85, "Yes", "No") )
```

③ **分组计算：计算组内平均 score1 并排名**

```
df %>% select( df %>% group_by(group) %>% mutate( avg_score = mean(score1), rank_in_group = min_rank(desc(score1))) %>% ungroup()
```

④ **插入位置控制：在 score1 之前插入新列 diff**

```
df %>% mutate( diff = score1 - score2, .before = "score1")
```

注意事项

① **按序计算：**在同一次 mutate() 中，新列可引用之前刚创建的列；但不能引用后面才定义的列；

② **覆盖原列：**直接使用已有列名会替换原列，谨防意外丢失数据；

③ **分组后记得 ungroup()：**若不解除分组，后续操作可能受意外影响；

④ **性能考量：**对于超大表格，分组计算和 across() 可能较慢，可考虑拆分步骤或使用数据库后端；

⑤ **NA 传播：**多数数学运算会将 NA 扩散，可结合 replace_na() 或在 if_else() 中指定默认值以控制缺失值行为。

dplyr包之arrange

arrange() 用于对数据框（data.frame/tibble）按指定列进行排序，默认升序，可通过辅助函数实现降序或多列排序。该函数支持对结果进行优先级排序、提取最大/最小值前几行、按时间/分数等字段排序以便后续分析

```
library(dplyr)
arrange(.data, ..., .by_group = FALSE)
```

待排序的数据框

一个或多个排序依据，可以是列名，也可以是表达式

当对已分组的数据框（group_by()）调用时，若设置为 TRUE，则先按组再按 ... 排序；否则跨组全局排序

配合函数

① **desc()**：指定降序排列。

② **across()**：配合 arrange() 对多列执行同一排序方向，需结合 !!! 和 syms 等于编程场景。

③ **row_number()**：生成行号，也可用于 slice_min()、slice_max() 等函数。

④ **order_by()（lubridate）**：基于时间向量排序。

⑤ **coalesce()**：处理含 NA 列排序，将 NA 替换为极大或极小值。

使用示例

① **按单列升序**

```
df %>% arrange(score1)
```

② **按单列降序**

```
df %>% arrange(desc(score1))
```

③ **多列排序：先按 group 再按 score1 降序**

```
df %>% arrange(group, desc(score1))
```

④ **分组后排序**

```
df%>%group_by(group)%>%arrange(score1)
```

注意事项

① **NA 的位置**：默认情况下，NA 值会被排在最后；如需将它们置于开头，可先用 is.na() 或 coalesce() 替换；

② **分组影响**：对已分组的表使用 arrange()，若不想跨组排序，需显式设置 .by_group = TRUE；

③ **稳定性**：arrange() 是稳定排序（stable sort），即相等值的行顺序与原数据顺序一致；

④ **性能考量**：对大表排序开销较大，若只是要最大 / 最小前几行，推荐使用 slice_min() / slice_max()；

⑤ **与管道结合**：常与 filter()、mutate()、slice() 等函数联用，构建清晰的数据处理链。

缺失值筛选及删除

功能及语法

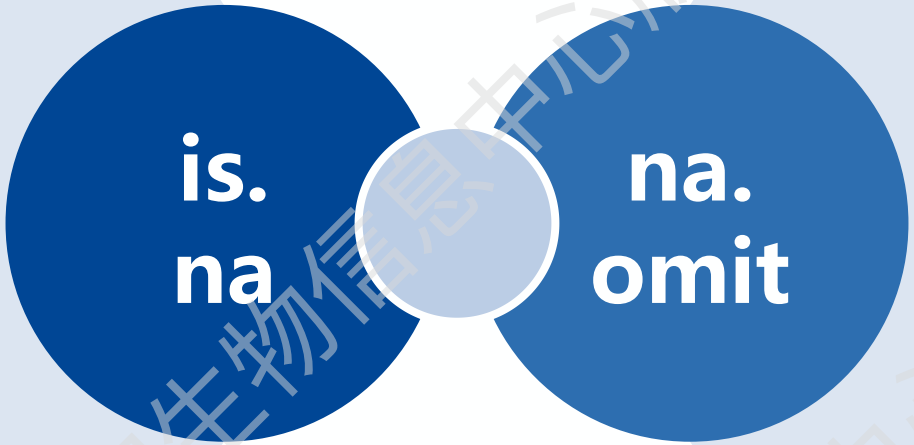
- 功能：返回与输入对象同型的逻辑向量/矩阵/数据框，TRUE 表示对应元素为 NA，否则为 FALSE。
- is.na(x)中x为任意向量、列表、矩阵或数据框。

典型用途

- 筛选含缺失的行：df[!is.na(df$column),]
- 统计缺失数目：sum(is.na(df))
- 可视化缺失模式（配合包如 VIM、naniar）

注意事项

- **信息丢失**：na.omit() 会导致行删除，若缺失比例较高会显著减小样本量；
- **完整案例**：在模型拟合（如 lm()、glm()）时，若不显式处理，R 会默认使用 na.omit()；



功能及语法

- 功能：返回删除了所有含有任何 NA 的行的新数据框或矩阵。
- na.omit(df)中等效于 df[complete.cases(df),]，其中 complete.cases() 返回每行是否“无缺失”。

典型用途

- clean_df<-na.omit(df) #将含 NA 的行全部丢弃
- 数据：一个行数可能减少的数据框；
- 属性：保留原始对象的na.action属性，记录哪些行被移除，用attr(clean_df, "na.action") 查看。

注意事项

- **可控删除**：推荐使用 complete.cases() 或 dplyr 中的 drop_na()，可指定特定列进行删除；
- **缺失机制**：在删除或替换前，应评估缺失模式，以选择合理的处理策略。

国家生物信息中心版权所有

横向合并与纵向拼接

merge 横向合并

功能定位

- merge(x, y, ...)：将两个数据框按一个或多个“键”列横向对齐，执行 SQL 风格的内、外、左、右连接，用于将不同表中的变量合并到同一行。

语法结构

- merge(x, y, by = intersect(names(x), names(y)), by.x = NULL, by.y = NULL, all = FALSE, all.x = all, all.y = all, sort = TRUE, suffixes = c(".x",".y"), incomparables = NULL)

示例演示

- merge(df1, df2)
- merge(df1, df2, all.x = TRUE, sort = FALSE)
- merge(df1, df3, by.x = "id", by.y = "key", all = TRUE)

bind_rows 纵向拼接

功能定位

- bind_rows(..., .id)：将多个数据框按行纵向堆叠，不要求列完全一致，自动对齐列名并补齐缺失，用于将多批次或多片段数据合并成一个整体。

语法结构

- bind_rows(..., .id = NULL)
- .id：若设定为字符（如 "source"），则在结果中新增一列，将输入对象的名字或在列表中位置（1、2、...）填入该列，以标识来源。

示例演示

- df4 <- data.frame(id = 4:5, score = c(80,85))
- bind_rows(df1, df4)
- bind_rows(list(train = df1, test = df4), .id = "set")

国家生物信息中心版权所有

本模块小结

数据整理与预处理操作模块的主要目标是帮助学员掌握R语言中表格数据清洗与预处理的核心技术，从行列筛选、变量生成与排序，到缺失值识别与处理，再到多表合并，构建一条高效、可复现的数据整理流程

行筛选（filter）

运用逻辑表达式从数据框中提取满足条件的子集，支持多条件、范围与正则匹配。

变量派生与排序(mutate、arrange)

利用算术、条件与分组累积函数生成新列；按单列或多列（升序/降序）对表格重新排序，为可视化与下游分析预备输入。

数据合并（merge、bind_rows）

横向合并：merge() 支持内/左/右/全连接，按键列对齐变量；

纵向拼接：bind_rows() 自动补齐不同列名，实现多批次或多片段数据的行级整合，并可通过 .id 标记来源。

列选择与重构（select）

灵活保留、重命名或批量匹配列；结合 tidyselect 辅助函数（starts_with()、matches() 等）精确控制输出格式。

缺失值处理（is.na、na.omit）

is.na() 用于识别和统计缺失模式；na.omit() 删除含 NA 的行，并可结合 replace_na() 进行插补，平衡数据完整性与样本量。

实战练习

某研究项目收集了两批实验数据：

- 样本信息表（sample_info）：包含样本编号、分组（Control/Case）、采集时间及临床指标，存在少量缺失；
- 测序结果表（seq_results）：包含样本编号、基因表达量（GeneA、GeneB）及质量评分（QC_score），两批次列名略有差异；

练习1：样本信息清洗

1. 使用 is.na() 统计 sample_info1 中各列的缺失数。
2. 对 Group 或 Collected 丢失的行，使用 na.omit() 或 drop_na() 删除。
3. 对 Age 中的 NA，用该表中非缺失值的中位数填补（replace_na() 或 mutate() + median()）。
4. 将两批次样本信息纵向拼接为 sample_info_all，并新增一列 .id 标明批次来源。

练习2：测序结果预处理

1. 在 seq_results1 和 seq_results2 中，分别用 select() 保留与样本合并相关的列（SampleID/Sample、GeneA、GeneB、QC_score，[GeneC]）。
2. 使用 rename() 统一两表的样本列名称为 SampleID。
3. 统计并删除 QC_score 小于 30 的样本行（filter(QC_score >= 30)）。
4. 对 seq_results1 中 GeneA、GeneB 的缺失，用该基因列的平均值进行插补（mutate() + if_else(is.na(), mean(), .))。
5. 将清洗后的两批次测序结果使用 bind_rows() 合并为 seq_results_all，并用 .id 标明来源。

实战要求及答案路径：/home/big/RBase/2-Data_Cleaning_and_Preprocessing

实战练习

练习3：横向合并与变量派生

1. 用 merge()（或 left_join()）将 sample_info_all 与 seq_results_all 按 SampleID 横向合并成 full_data，保留所有样本。
2. 用 mutate() 新增：
   - TotalExpr = GeneA + GeneB + coalesce(GeneC, 0)
   - ExprRatio = GeneA / TotalExpr
3. 按 Group 分组后，用 mutate() 计算该组内 TotalExpr 的均值 (GroupMeanExpr)。
4. 使用 arrange() 按 Group 升序、ExprRatio 降序排序，并 slice() 每组提取前 2 名高表达样本。

练习4：导出结果

1. 将最终结果使用 write.csv() 保存为 cleaned_data.csv。
2. 使用 glimpse() 和 summary() 简要检查输出，确保无缺失且各列类型正确。

实战要求及答案路径：/home/big/RBase/2-Data_Cleaning_and_Preprocessing

国家生物信息中心版权所有

03

基础绘图函数与图形系统

国家生物信息中心版权所有

plot() 通用散点/折线/点线图

plot() 是 R 最基础的绘图函数，它提供了对二维散点图、折线图、柱状图等多种图形类型的通用接口

```
plot(x, y = NULL,
     type     = "p",
     main     = NULL,
     sub      = NULL,
     xlab     = NULL,
     ylab     = NULL,
     xlim     = NULL,
     ylim     = NULL,
     pch      = NULL,
     col      = NULL,
     lty      = NULL,
     lwd      = NULL,
     …
)
```

单个向量 x → 绘制 x 值对下标（1,2,...）的图
两个向量 x, y → 绘制 (x[i], y[i]) 的散点

图形类型： p=散点；l=折线；b：点线结合；h=类似直方图的高线；s-阶梯线；n=仅设置坐标系，不画点或线

主标题

副标题

x坐标轴标签

y坐标轴标签

x坐标轴范围

y坐标轴范围

点形，可用数字或字符

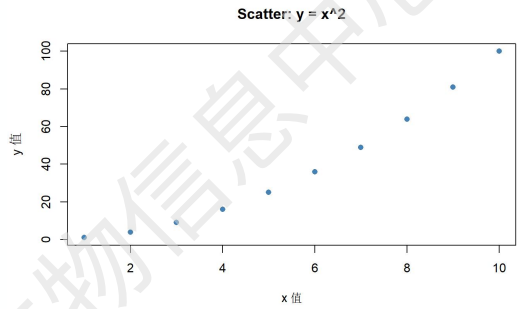
颜色，可用名称或整数/十六进制

线型（1=实线、2=虚线）

线宽

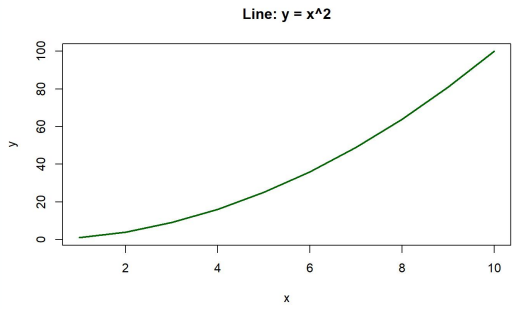
基本散点图

```
x <- 1:10
y <- x^2
plot(x, y,
     type = "p",
     pch  = 16,
     col  = "steelblue",
     main = "Scatter: y = x^2",
     xlab = "x 值",
     ylab = "y 值")
```



基本折线图

```
plot(x, y,
     type = "l",
     lty  = 1,
     lwd  = 2,
     col  = "darkgreen",
     main = "Line: y = x^2")
```



国家生物信息中心版权所有

plot() 通用散点/折线/点线图

plot() 是 R 最基础的绘图函数，它提供了对二维散点图、折线图、柱状图等多种图形类型的通用接口

```
plot(x, y = NULL,
     type     = "p",
     main     = NULL,
     sub      = NULL,
     xlab     = NULL,
     ylab     = NULL,
     xlim     = NULL,
     ylim     = NULL,
     pch      = NULL,
     col      = NULL,
     lty      = NULL,
     lwd      = NULL,
     …
)
```

单个向量 x → 绘制 x 值对下标（1,2,...）的图
两个向量 x, y → 绘制 (x[i], y[i]) 的散点

图形类型：p=散点；l=折线；b：点线结合；h=类似直方图的高线；s-阶梯线；n=仅设置坐标系，不画点或线

主标题

副标题

x坐标轴标签

y坐标轴标签

x坐标轴范围

y坐标轴范围

点形，可用数字或字符

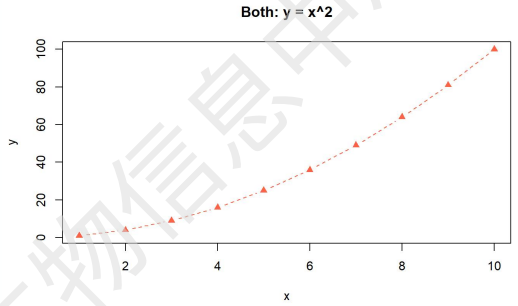
颜色，可用名称或整数/十六进制

线型（1=实线、2=虚线）

线宽

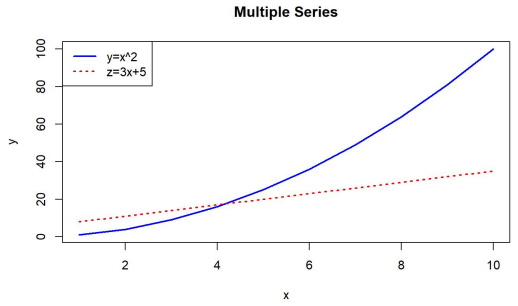
点线结合

```
plot(x, y,
     type = "b",
     pch  = 17,
     lty  = 2,
     col  = "tomato",
     main = "Both: y = x^2")
```



多系列叠加

```
z <- 3*x + 5
plot(x, y, type = "l", col = "blue", lwd = 2, ylim = range(y, z), main = "Multiple Series")
lines(x, z, type = "l", col = "red", lty = 3, lwd = 2)
legend("topleft", legend = c("y=x^2","z=3x+5"), col = c("blue","red"), lty = c(1,3), lwd = 2)
```



国家生物信息中心版权所有

hist() 直方图

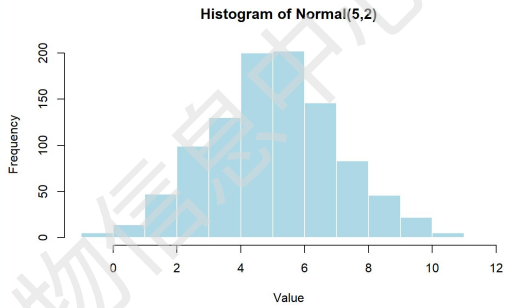
hist() 用于绘制直方图，将连续数值数据分割成若干区间（bins），展示每个区间的频数或密度分布

```
hist(x,
     breaks      = "Sturges",
     freq        = NULL,
     probability = !freq,
     include.lowest = TRUE,
     right       = TRUE,
     col         = NULL,
     border      = NULL,
     main        = NULL,
     xlab        = NULL,
     ylab        = NULL,
     xlim        = NULL,
     ylim        = NULL,
     labels      = FALSE,
     …
)
```

参数	含义
x	数值向量
breaks	分箱方式：可指定数值向量(手动分割点)、也可用字符串方法("Sturges","Scott","FD")
freq	是否绘制频数（TRUE）或密度（FALSE）
probability	与 freq 相反，若 TRUE 绘制密度(y 轴为密度)
include.lowest	是否将第一个区间包含最小值
right	区间是否右闭合
col, border	柱体填充色与边框色
main, xlab, ylab	标题和坐标轴标签
labels	是否在每个柱子上标注数值

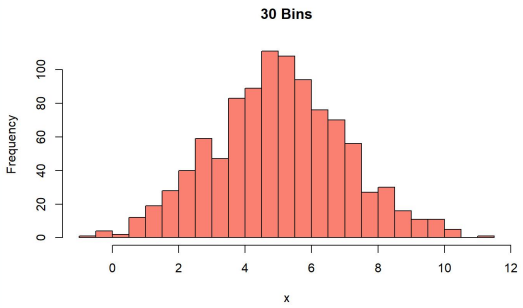
基本直方图

```
set.seed(123)
x <- rnorm(1000, mean = 5, sd = 2)
hist(x,
     main   = "Histogram of Normal (5,2)",
     xlab   = "Value",
     col    = "lightblue",
     border = "white")
```



指定分箱数

```
hist(x,
     breaks = 30,
     main   = "30 Bins",
     col    = "salmon")
```



hist() 直方图

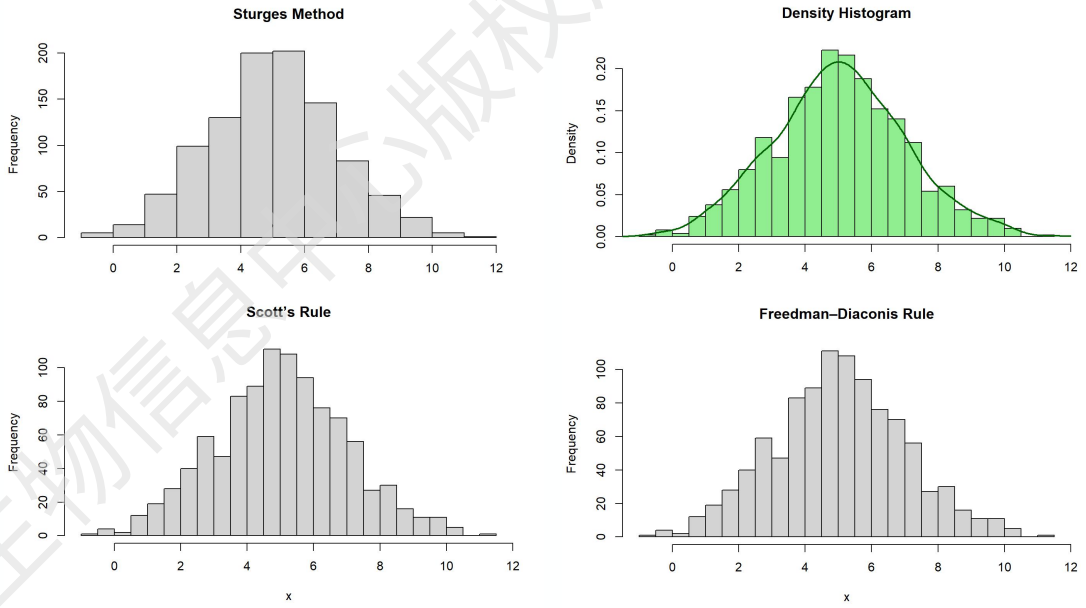
hist() 用于绘制直方图，将连续数值数据分割成若干区间（bins），展示每个区间的频数或密度分布

```
hist(x,
     breaks      = "Sturges",
     freq        = NULL,
     probability = !freq,
     include.lowest = TRUE,
     right       = TRUE,
     col         = NULL,
     border      = NULL,
     main        = NULL,
     xlab        = NULL,
     ylab        = NULL,
     xlim        = NULL,
     ylim        = NULL,
     labels      = FALSE,
     …
)
```

参数	含义
x	数值向量
breaks	分箱方式：可指定数值向量(手动分割点)、也可用字符串方法("Sturges","Scott","FD")
freq	是否绘制频数（TRUE）或密度（FALSE）
probability	与 freq 相反，若 TRUE 绘制密度(y 轴为密度)
include.lowest	是否将第一个区间包含最小值
right	区间是否右闭合
col, border	柱体填充色与边框色
main, xlab, ylab	标题和坐标轴标签
labels	是否在每个柱子上标注数值

使用不同算法自动分箱

```
hist(x, breaks = "Sturges", main = "Sturges Method")
hist(x, breaks = "Scott",  main = "Scott’ s Rule")
hist(x, breaks = "FD",     main = "Freedman–Diaconis Rule")
```



密度直方图

```
hist(x,
    probability = TRUE,
    breaks      = 20,
    col         = "lightgreen",
    main        = "Density Histogram")
lines(density(x), lwd=2, col="darkgreen")
```


国家生物信息中心版权所有

boxplot() 箱线图

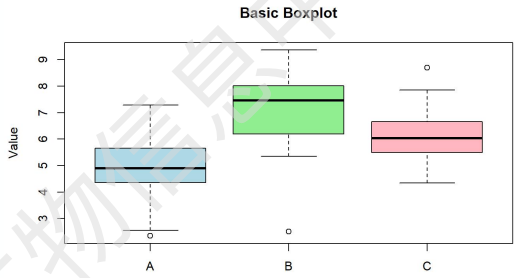
boxplot() 用于绘制箱线图，直观展示数据分布的五数概括（最小值、第一四分位数、中位数、第三四分位数、最大值）及异常值

```
boxplot(x,
        ...,
        formula = NULL,
        data    = NULL,
        subset  = NULL,
        at      = NULL,
        names   = NULL,
        main    = NULL,
        xlab    = NULL,
        ylab    = NULL,
        col     = NULL,
        border  = NULL,
        notch   = FALSE,
        horizontal = FALSE,
        outline    = TRUE,
        varwidth   = FALSE,
        range      = 1.5,
        pars       = list(),
        add        = FALSE
)
```

参数	含义
x	数值向量或数值矩阵，或公式形式 y ~ group
formula	公式接口，如 value ~ group
data, subset	用于公式接口时指定数据框和子集
at	手动指定箱体在坐标上的位置
names	自定义每个箱体的标签
main,xlab,ylab	标题与坐标轴标签
col,border	箱体填充色与边框色
notch	是否绘制缺口（表示中位数95%置信区间）
horizontal	是否水平绘制
outline	是否显示离群点

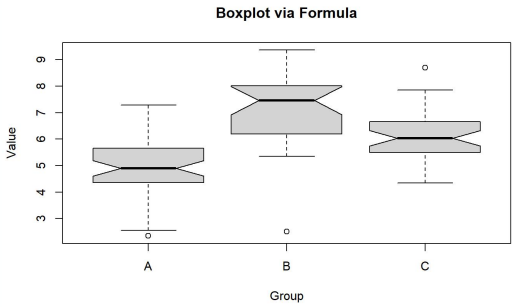
使用不同算法自动分箱

```
grpA <- rnorm(50, mean = 5)
grpB <- rnorm(30, mean = 7, sd = 1.5)
grpC <- rnorm(40, mean = 6, sd = 1)
boxplot(grpA, grpB, grpC,
        names = c("A","B","C"),
        col   = c("lightblue","lightgreen","lightpink"),
        main  = "Basic Boxplot",
        ylab  = "Value")
```



公式接口

```
df <- data.frame(
  group = rep(c("A","B","C"), times = c(50,30,40)),
  value = c(grpA, grpB, grpC))
boxplot(value ~ group, data = df,
        col = "lightgray",
        notch = TRUE,
        main = "Boxplot via Formula",
        xlab = "Group", ylab = "Value")
```



国家生物信息中心版权所有

boxplot() 箱线图

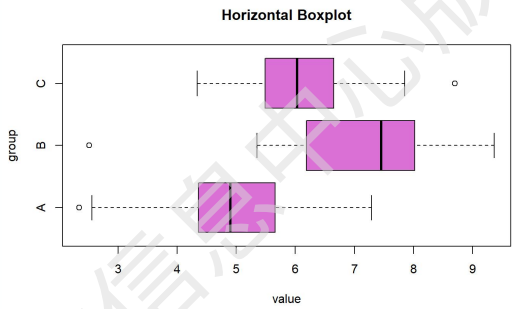
boxplot() 用于绘制箱线图，直观展示数据分布的五数概括（最小值、第一四分位数、中位数、第三四分位数、最大值）及异常值

```
boxplot(x,
        ...,
        formula = NULL,
        data    = NULL,
        subset = NULL,
        at      = NULL,
        names   = NULL,
        main    = NULL,
        xlab    = NULL,
        ylab    = NULL,
        col     = NULL,
        border  = NULL,
        notch   = FALSE,
        horizontal = FALSE,
        outline    = TRUE,
        varwidth   = FALSE,
        range      = 1.5,
        pars       = list(),
        add        = FALSE
)
```

参数	含义
x	数值向量或数值矩阵，或公式形式 y ~ group
formula	公式接口，如 value ~ group
data, subset	用于公式接口时指定数据框和子集
at	手动指定箱体在坐标上的位置
names	自定义每个箱体的标签
main,xlab,ylab	标题与坐标轴标签
col,border	箱体填充色与边框色
notch	是否绘制缺口（表示中位数95%置信区间）
horizontal	是否水平绘制
outline	是否显示离群点

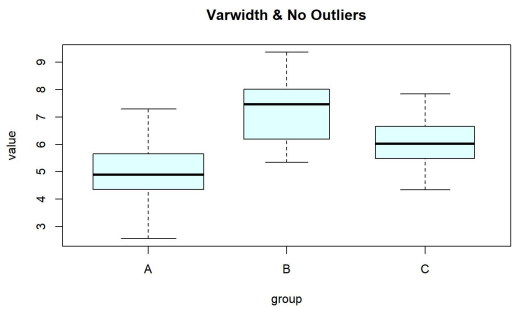
水平箱线图

```
boxplot(value ~ group, data = df,
        horizontal = TRUE,
        col = "orchid",
        main = "Horizontal Boxplot")
```



按样本量调整宽度

```
boxplot(value ~ group, data = df,
        outline  = FALSE,
        varwidth = TRUE,
        col      = "lightcyan",
        main     = "Varwidth & No Outliers")
```



barplot() 箱线图

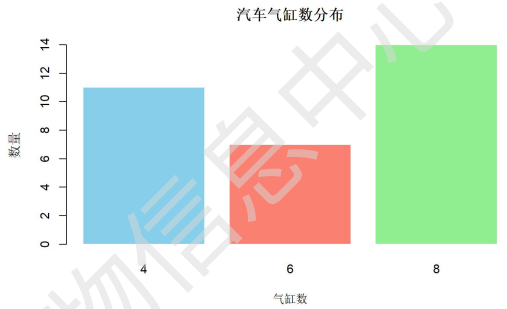
barplot() 用于绘制条形图（柱状图），展示分类或分组数据的频数、百分比或数值大小，适合可视化离散类别的对比，如样本计数、均值差异、条件频率等

```
barplot(height,
        names.arg    = NULL,
        beside       = FALSE,
        horiz        = FALSE,
        col          = NULL,
        border       = NULL,
        main         = NULL,
        xlab         = NULL,
        ylab         = NULL,
        ylim         = NULL,
        las          = 0,
        legend.text  = NULL,
        args.legend  = list(),
        …
)
```

参数	含义
height	数值向量或矩阵：向量时各条高度；矩阵时每列一组分组条形
names.arg	条形标签（类别名称）
beside	是否并排绘制分组条形（TRUE）或堆叠（FALSE，默认）
horiz	是否水平绘制
col, border	条形填充色与边框色
main,xlab,ylab	标题与坐标轴标签
ylim	Y 轴范围，或 xlim（当 horiz=TRUE）
las	标签方向（0-3）
legend.text	图例文字向量
args.legend	传给 legend() 的参数列表

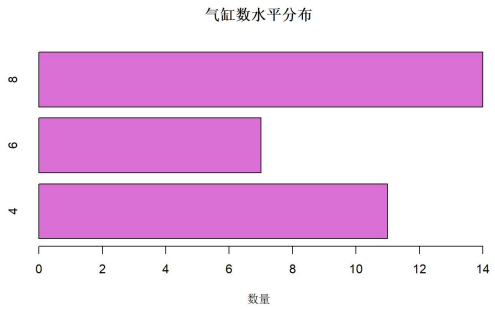
单维频数条形图

```
counts <- table(mtcars$cyl)
barplot(counts,
        main     = "汽车气缸数分布",
        xlab     = "气缸数",
        ylab     = "数量",
        col      = c("skyblue","salmon","lig
htgreen"),
        border   = "white")
```



水平条形图

```
barplot(counts,
        horiz    = TRUE,
        main     = "气缸数水平分布",
        xlab     = "数量",
        col      = "orchid")
```



barplot() 箱线图

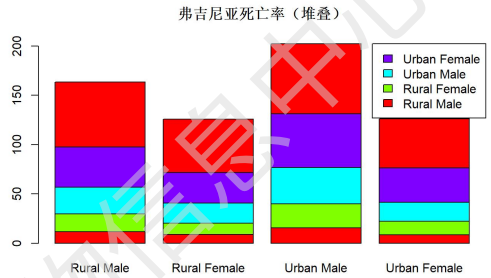
barplot() 用于绘制条形图（柱状图），展示分类或分组数据的频数、百分比或数值大小，适合可视化离散类别的对比，如样本计数、均值差异、条件频率等

```
barplot(height,
        names.arg    = NULL,
        beside       = FALSE,
        horiz        = FALSE,
        col          = NULL,
        border       = NULL,
        main         = NULL,
        xlab         = NULL,
        ylab         = NULL,
        ylim         = NULL,
        las          = 0,
        legend.text  = NULL,
        args.legend  = list(),
        …
)
```

参数	含义
height	数值向量或矩阵：向量时各条高度；矩阵时每列一组分组条形
names.arg	条形标签（类别名称）
beside	是否并排绘制分组条形（TRUE）或堆叠（FALSE，默认）
horiz	是否水平绘制
col, border	条形填充色与边框色
main,xlab,ylab	标题与坐标轴标签
ylim	Y 轴范围，或 xlim（当 horiz=TRUE）
las	标签方向（0-3）
legend.text	图例文字向量
args.legend	传给 legend() 的参数列表

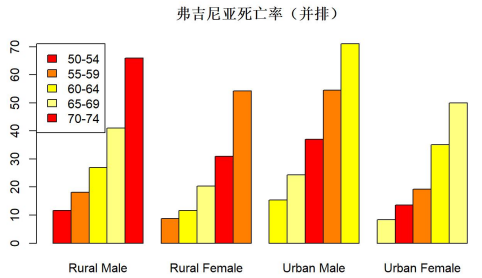
堆叠式分组条形图

```
df <- as.matrix(VADeaths)  #R自带矩阵
barplot(df,
        beside    = FALSE,
        col       = rainbow(ncol(df)),
        legend.text = colnames(df),
        args.legend = list(x = "topright"),
        main      = "弗吉尼亚死亡率（堆叠）
")
```



并排分组条形图

```
barplot(df,
        beside    = TRUE,
        col       = heat.colors(ncol(df)),
        legend.text = TRUE,
        args.legend = list(x = "topleft"),
        main      = "弗吉尼亚死亡率（并排）
")
```



国家生物信息中心版权所有

图形参数控制

明确如何通过 par() 全局设置与各绘图函数参数精细调整颜色、标题、坐标轴与布局，以实现高质量可视化

```r
# 查看当前参数列表（只看部分）
par()[c("mfrow","mar","oma","cex","pch")]

# 设置多图布局：2 行 2 列
par(mfrow = c(2,2))

# 调整图内边距（底、左、顶、右 单位行数）
par(mar = c(4, 4, 2, 1))

# 设置外边距（单位行数）
par(oma = c(1, 1, 1, 1))

# 改变默认点型、字体大小
par(pch = 16, cex = 1.2)

# 恢复默认布局与参数
par(mfrow = c(1,1))
par(mar   = c(5, 4, 4, 2) + 0.1)
par(oma   = c(0, 0, 0, 0))
par(pch   = 1, cex = 1)
```

参数	含义
mfrow/mfcol	多图布局
mar	图内边距：c(bottom,left,top,right)
oma	图外边距（整体外边空白）
pch, cex	默认点型与缩放比例
lty, lwd	默认线型与线宽
col, bg	默认前景色与背景色
las	坐标轴刻度标签方向（0–3）
xpd	绘图范围外是否允许绘制（TRUE 或 FALSE）

颜色相关

1、在绘图函数中：plot(x, y, col = “steelblue”, bg = “lightyellow”);
2、par()函数全局设定：par(col = “black”, bg = “white”);
3、调色板：library(RColorBrewer)、brewer.pal(5, “Set2”)。

标题与注释

1、主标题、副标题：plot(..., main = “Main Title”, sub = “Subtitle”);
2、坐标轴标签：plot(..., xlab = “X Axis Label”, ylab = “Y Axis Label”);
3、字体大小与样式：plot(..., cex.main = 1.5, font.main = 2, cex.lab = 1.2, font.lab = 3) # 轴标签倾斜 + 标题加粗放大

坐标轴与刻度

1、刻度方向：par(las = 1) -> 水平刻度标签； par(las = 2) -> 垂直刻度标签
2、坐标范围：plot(..., xlim = c(0, 10), ylim = c(-5, 5))
3、刻度间隔自定义：axis(1, at = seq(0,10,2), labels = LETTERS[1:6])。

点、线与符号

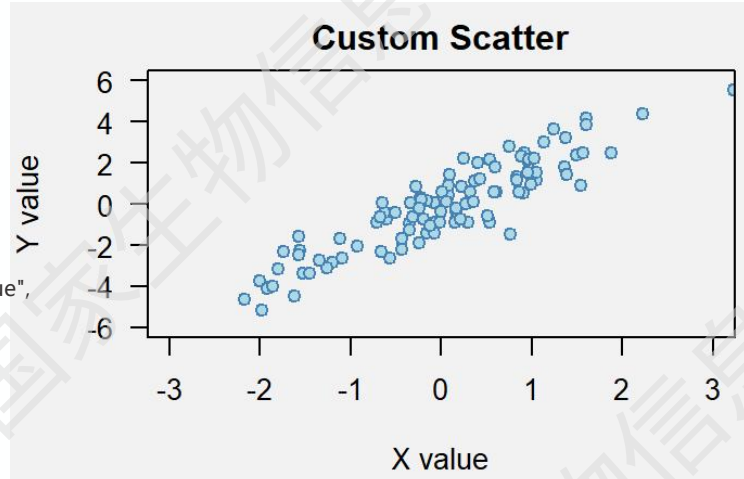
1、点型（pch）：plot(..., pch = 21, bg = “orange”) # 带背景色的圆点；
2、线型（lty）与线宽（lwd）：lines(x, y, lty = 2, lwd = 3)。

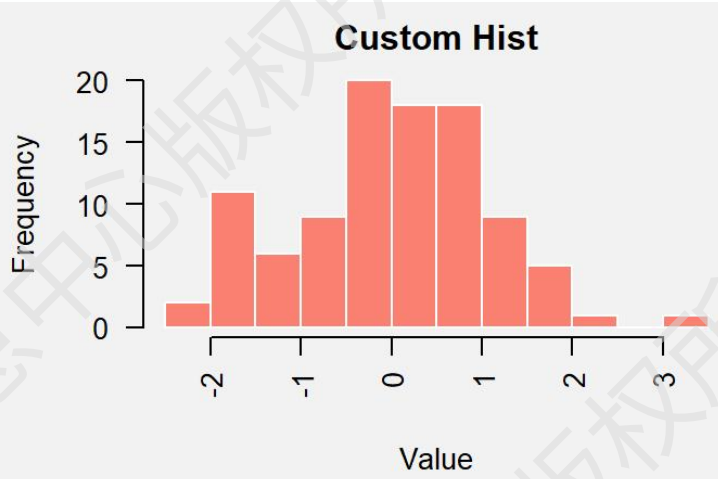
图形参数控制

明确如何通过 par() 全局设置与各绘图函数参数精细调整颜色、标题、坐标轴与布局，以实现高质量可视化

par(mfrow = c(2,2), mar = c(4,4,2,1), bg = "gray95")

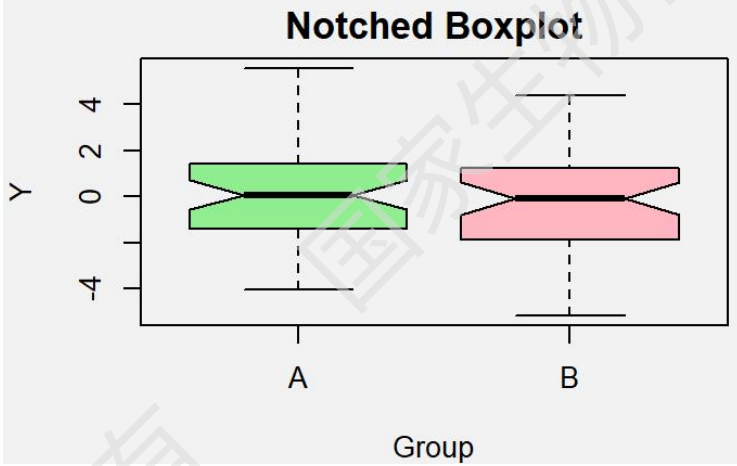
```
x <- rnorm(100); y <- x*2 + rnorm(100)
plot(x, y,
    main = "Custom Scatter",
    xlab = "X value", ylab = "Y value",
    col  = "steelblue", pch  = 21, bg = "lightblue",
    xlim = c(-3,3), ylim = c(-6,6),
    las  = 1)
```

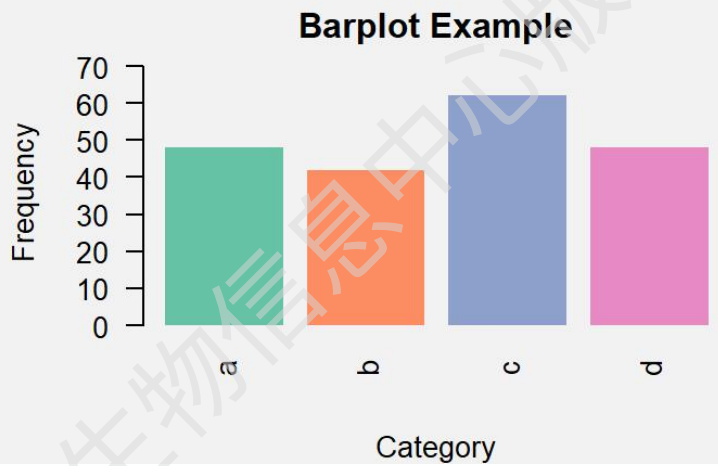





```
hist(x,
    main   = "Custom Hist",
    xlab   = "Value",
    col    = "salmon",
    border = "white",
    breaks = 15,
    las    = 2)
```

```
boxplot(y ~ gl(2,50, labels=c("A","B")),
      main     = "Notched Boxplot",
      xlab     = "Group",
      ylab     = "Y",
      col      = c("lightgreen","lightpink"),
      notch    = TRUE,
      varwidth = TRUE,
      cex.main = 1.3,
      font.lab = 2)
```






```
counts <- table(sample(letters[1:4], 200, replace
= TRUE))
barplot(counts,
      main      = "Barplot Example",
      xlab      = "Category",
      ylab      = "Frequency",
      col       = brewer.pal(4, "Set2"),
      border    = NA,
      las       = 2,
      ylim      = c(0, max(counts)+10),
      args.legend = list(x="topright",
legend=names(counts), fill=brewer.pal(4,"Set2")))
```

图像保存

所有 Base Graphics 绘图都会直接输出到当前图形设备。要将图形导出为文件，需要先启动一个“图形设备”，在绘图完成后再关闭设备。常见设备包括pdf()、png()、jpeg()、tiff()、svg()以及dev.off()

打开设备

（1）单位英寸：pdf("figure.pdf", width = 6, height = 4)

（2）像素及分辨率：png("figure.png", width = 800, height = 600, res = 150)

关闭设备

dev.off()

务必 dev.off()：不关闭，文件可能不完整或无法打开

绘制图形

```
plot(x, y, main = "My Plot", col = "steelblue")
```

1、科学出版首选矢量图（PDF/SVG）、WEB/报告插图首选（PNG/TIFF），需要根据DPI调整分辨率；

2、尺寸与分辨率：建议按期刊要求设定width/height与res，一般论文图建议300 DPI以上；

3、推荐用脚本循环生成多张图并收于同一 PDF（onefile=TRUE）或按命名规则保存 PNG。

pdf()参数详解

```
pdf(file     = "Rplot.pdf",
    width    = 7,         # 图宽（英寸）
    height   = 5,         # 图高（英寸）
    family   = "Helvetica",# 字体族
    pointsize = 12,       # 默认字体大小
    bg       = "white",   # 背景色
    onefile  = TRUE,      # 多页图写入单个 PDF
    paper    = "special"  # 纸张大小
```

png()参数详解

```
png(filename = "Rplot.png",
    width    = 800,     # 宽度（像素）
    height   = 600,     # 高度（像素）
    units    = "px",    # 单位："px","in","cm","mm"
    res      = 150,     # DPI（分辨率）
    bg       = "transparent" # 可选透明背景
)
```

国家生物信息中心版权所有

本模块小结

基础绘图函数与图形系统模块的主要目标是帮助学员掌握R基础绘图系统，从常用图形函数到全局与局部参数定制，再到图像文件输出，构建一条“绘图→美化→导出”的完整工作流，为科研报告可视化展示打下坚实基础

plot()：散点、折线、点线图

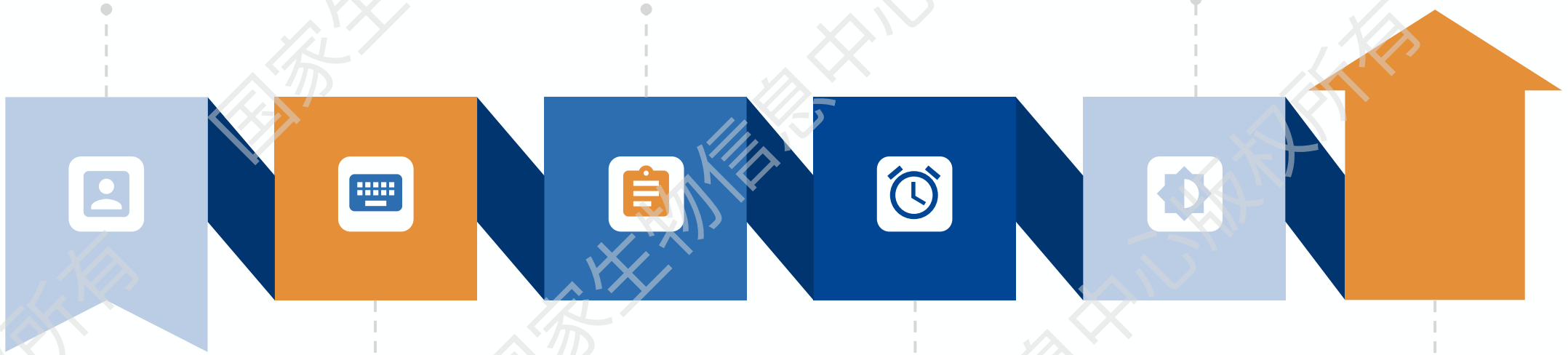
基本用法、样式定制、多图布局

boxplot()：箱线图

公式接口、缺口与宽度、离群点与方向

图形参数控制

全局设置、局部调整、自定义坐标轴



hist()：直方图与密度

分箱策略、频数VS密度、视觉美化

barplot()：条形图

单变量频数、多组并排/堆叠、水平展示

图像导出

矢量输出、位图输出、设备管理

国家生物信息中心版权所有

实战练习

练习1：散点图（plot()）

1. 用 mtcars 绘制 wt vs mpg 的散点图，默认样式。
2. 在此基础上，将点型改为实心圆（pch=19），颜色设为 “steelblue”，并将坐标轴范围限定在 xlim=c(1.5,5.5)、ylim=c(10,35).
3. 使用 par(mfrow=c(1,2)) 布局，先后绘制 wt vs mpg 和 hp vs mpg 两幅图，然后恢复默认布局。

练习2：直方图（hist()）

1. 用 mtcars$mpg 绘制默认直方图。
2. 将分箱数设为 15，填充色为 “salmon”，柱子边框设为白色。
3. 在同一图上绘制密度直方图（probability=TRUE）并叠加核密度曲线。

练习3：箱线图（boxplot()）

1. 用 iris 绘制 Sepal.Length 按 Species 分组的箱线图，显示默认样式。
2. 打开缺口（notch=TRUE）并按样本量调整宽度（varwidth=TRUE），为三组分别着色。
3. 将 Sepal.Width 的箱线图水平绘制（horizontal=TRUE），并设填充色。

实战要求及答案路径：/home/big/RBase/3-Basic_Drawing_Functions_and_Graphics_System_Module

国家生物信息中心版权所有

实战练习

练习4：条形图（barplot()）

1. 统计 mtcars$cyl 的频数并绘制条形图，填充色自选，边框去除。
2. 构造按 gear（行）× cyl（列）的频数矩阵，并绘制并排分组条形图（beside=TRUE）并添加图例。
3. 将 4.1 的条形图改为水平展示（horiz=TRUE）。

练习5：图形参数控制

1. 用 par() 将画布分成 2×2、边距设为 mar=c(4,4,2,1)、默认点型设为 pch=17、字符放大 cex=1.2。
2. 在绘图时，设置主标题字体放大 1.5 倍且加粗（cex.main=1.5,font.main=2），坐标轴标签倾斜显示（cex.lab=1.2,font.lab=3）。
3. 自定义 X 轴刻度：先 xaxt="n" 禁用，再用 axis(1, at=seq(0,10,2), labels=LETTERS[1:6], las=2) 添加。
4. 恢复 par() 默认布局与参数。

练习6：保存图像

1. 将练习 1.2 的散点图保存为 PDF：scatter.pdf，尺寸 6×4 英寸。
2. 将练习 2.2 的直方图保存为 PNG：mpg_hist.png，分辨率 300 DPI。
3. 在同一 PDF (multi.pdf) 中，用 onefile=TRUE 循环绘制三幅正态分布直方图（均值分别为 1, 2, 3）；然后关闭设备。

实战要求及答案路径：/home/big/RBase/3-Basic_Drawing_Functions_and_Graphics_System_Module

国家生物信息中心版权所有

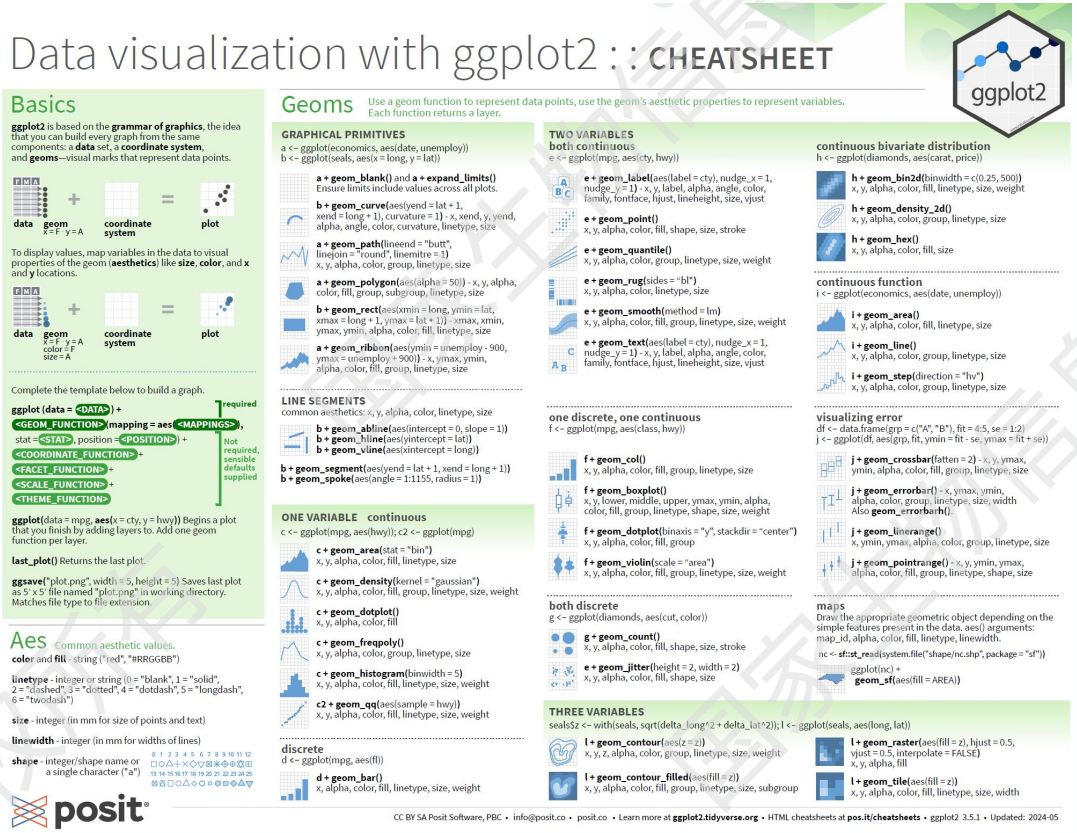
04

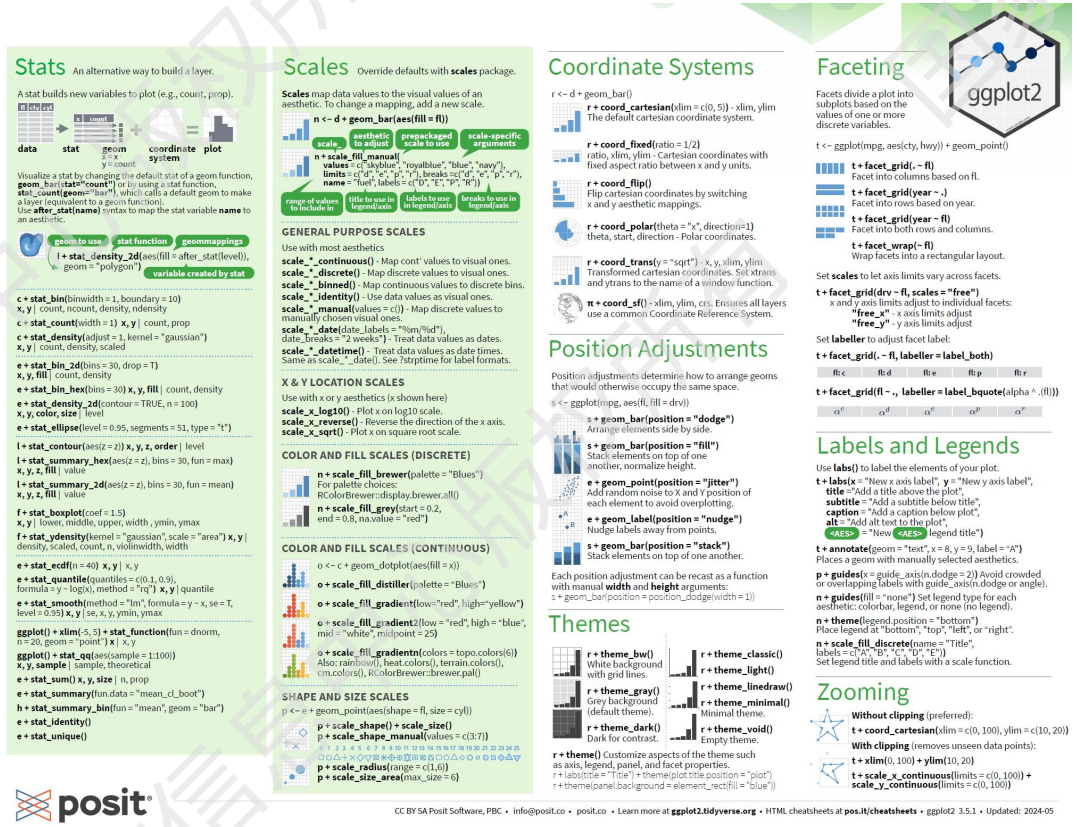
ggplot2语法结构与图层构建

国家生物信息中心版权所有

ggplot2函数核心功能一览

Data Visualization with ggplot2 :: Cheat Sheet 系统梳理了 ggplot2 的核心语法、图层构建、坐标与分面、统计变换与主题自定义等要点，帮助用户快速掌握声明式绘图流程





下载链接：https://rstudio.github.io/cheatsheets/html/data-visualization.html?_gl=1*rwozvi*_ga*MTg5MjQ4MTIwMi4xNzQ4OTE3ODg5*_ga_2C0WZ1JHG0*czE3NDg5MTc4ODkkbzEkZzEkdDE3NDg5MTgzNTckajYwJGwwJGgw

ggplot2语法结构

ggplot2 是 R 语言中最常用的可视化系统，基于“语法图形学（Grammar of Graphics）”，强调将数据映射到图形元素（图层 Layer）上，逐层构建图像

基本构图流程

1、添加基础图对象

```
p <- ggplot(data = df, aes(x = xvar, y = yvar))
```

2、添加图层函数（geom_*()）

```
p + geom_point()  # 散点图
p + geom_bar(stat = "identity")  # 柱状图
p + geom_boxplot()  # 箱线图
```

3、添加标签与主题

```
p + labs(title = "主标题", subtitle = "副标题", x = "X轴名称", y = "Y轴名称") + theme_minimal()
```

核心语法结构

```
ggplot(data = <数据框>, aes(x = <x变量>, y = <y变量>)) +
        <图层函数> +
        <主题或标签设置>
```

1、data：数据来源，必须是数据框；
2、aes()：美学映射，如x/y轴、颜色等；
3、geom_*()：图层函数，定义类型；
4、labs()：设置标题、副标题、轴标签；
5、theme()：美化图标风格，如字体等。

常用图层函数介绍

1、geom_point()：绘制散点图；
2、geom_bar()：绘制柱状图；
3、geom_line()：绘制折线图；
4、geom_boxplot()：绘制箱线图；
5、geom_histogram()：绘制直方图。

国家生物信息中心版权所有

ggplot2语法结构

ggplot2 通过 aes() 映射数据、geom_*() 构建图层、labs() 添加注释、theme() 美化风格，采用图层叠加的语法体系，实现灵活、可复用的可视化构图

美学映射 + 图层叠加

aes(x, y, color, fill, shape, size, alpha, group)

- color/fill：控制颜色；
- shape/size：控制点的形状和大小；
- alpha：控制透明度
- group：用于分组（如线条分类）

组合多个几何图层

```
ggplot(df, aes(x = x, y = y)) +
        geom_point() +
        geom_smooth(method = "lm")
        # 添加回归线
```

标签与主题美化

（1）添加标题、标签

```
+ labs(title = "图主标题", subtitle = "副标题", x = "X轴", y = "Y轴", caption = "数据来源")
```

（2）修改图形主题

```
+ theme_bw()        # 黑白主题（带有清晰网格线）
+ theme_minimal() # 极简主题（无背景和坐标轴线）
+ theme_classic()    # 经典主题（去除灰色背景）
```

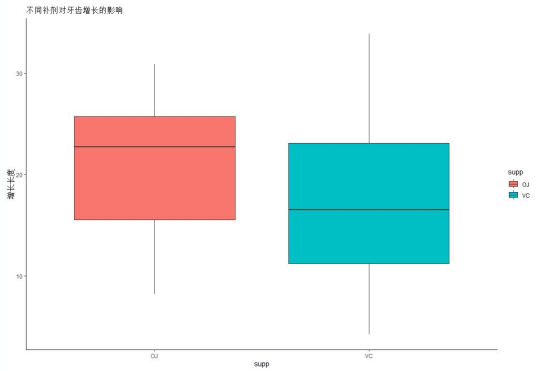
（3）主题细节定制（可选）

```
+ theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"), axis.text.x = element_text(angle = 45, hjust = 1))
```

案例演示

分组箱线图：

```
ggplot(ToothGrowth, aes(x = supp, y = len, fill = supp))
    + geom_boxplot() +
    labs(title = "不同补剂对牙齿增长的影响", y = "增长长度")
    + theme_classic()
```



常用图层介绍：geom_bar()

geom_bar() 用于绘制分类变量的柱状图，它会自动对数据分组并统计频数（默认情况下）

ggplot(data, aes(x = <分类变量>)) + geom_bar()

指定分类变量，如性别、组别、类型等。默认对每个分类自动计数

参数详解

1、mapping：设置映射的变量；

2、stat：设置统计方式（默认自动计数）；

3、position：控制柱子的排布方式；

4、fill：控制柱子的填充颜色；

5、color：控制边框颜色；

6、width：控制柱子宽度；

7、alpha：控制透明度，0-1。

绘图方式分类

1、stat = “count”：

- ggplot(data = mpg, aes(x = class)) + geom_bar()
- class是分类变量，自动统计每类数量。

2、stat = “identity”：

- df <- data.frame(type = c("A", "B", "C"), value = c(10, 23, 15)) ggplot(df, aes(x = type, y = value)) + geom_bar(stat = "identity")
- 必须设置stat为identity，表示y轴用已有数据而不是自动计数

柱状图的位置参数

1、stack（默认）：

- 柱子堆叠显示
- geom_bar(position="stack")

2、dodge:

- 并列柱状图
- geom_bar(position="dodge")

3、fill:

- 百分比柱状图（每组总和为1）
- geom_bar(position="fill")

技巧与注意事项

1、美观性：使用 theme_minimal() 或 theme_classic() 增强专业感；

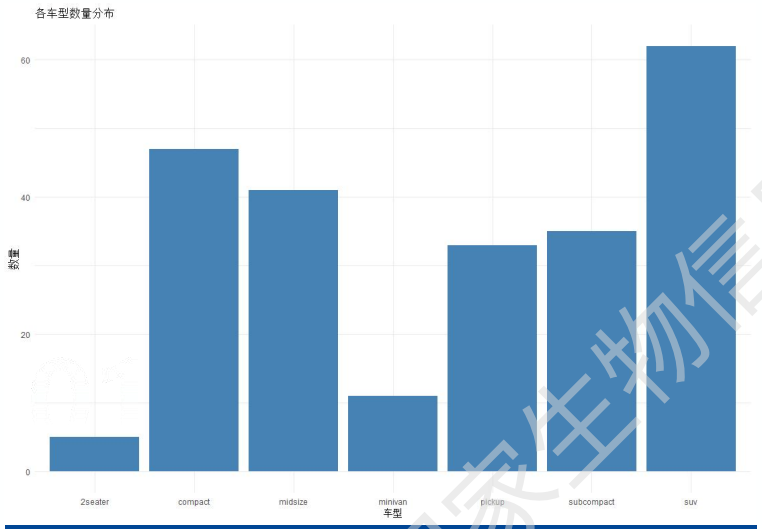
2、色彩：使用 scale_fill_brewer() 设置色板；

3、标签：可加 geom_text() 添加数值标签；

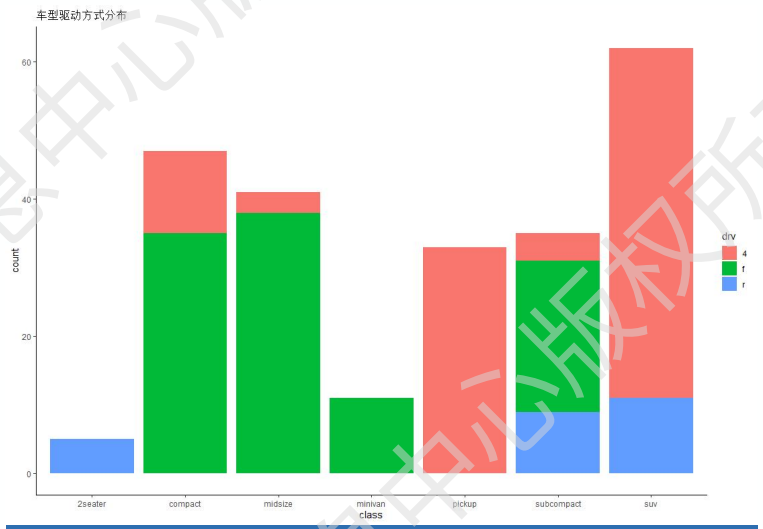
4、顺序：用 factor(x, levels = ...) 自定义类别顺序。

国家生物信息中心版权所有

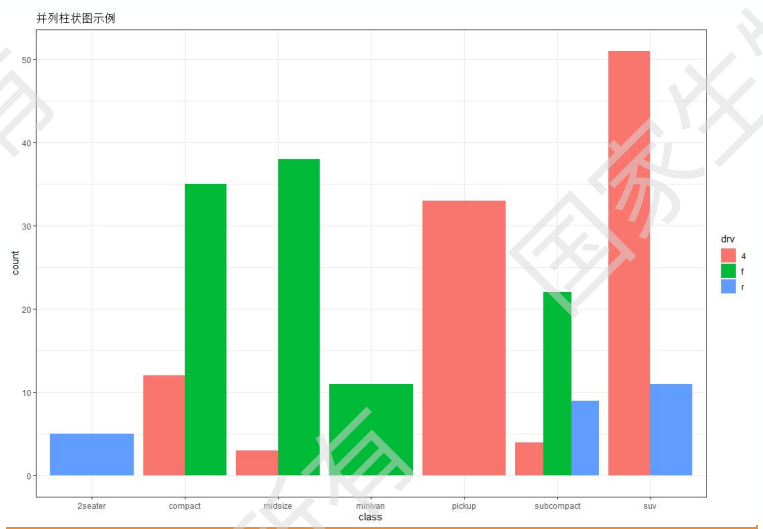
geom_bar()的常见用法与案例展示



示例1



示例2



示例3

基本计数柱状图

```
ggplot(mpg, aes(x = class)) +
  geom_bar(fill = "steelblue") +
  labs(title = "各车型数量分布", x = "车型",
y = "数量") +
  theme_minimal()
```

堆叠柱状图（分组填色）

```
ggplot(mpg, aes(x = class, fill = drv)) +
  geom_bar(position = "stack") +
  labs(title = "车型驱动方式分布") +
  theme_classic()
```

并列柱状图

```
ggplot(mpg, aes(x = class, fill = drv)) +
  geom_bar(position = "dodge") +
  labs(title = "并列柱状图示例") +
  theme_bw()
```

国家生物信息中心版权所有

常用图层介绍：geom_point()

geom_point() 用于绘制二维平面中的散点图，每个点代表一个观测值，位置由 x 和 y 值决定，可用于查看变量之间的关系、分布、聚类等

ggplot(data = 数据框, aes(x = 变量1, y = 变量2)) + geom_point() ➡ 增加散点图层

⬇

aes中的x、y分别表示坐标轴

参数详解

1、mapping：设置映射关系；

2、color：控制点的边框颜色；

3、fill：控制点的填充颜色；

4、size：控制点的大小；

5、alpha：控制点的透明度（0-1）；

6、shape：控制点的形状.

编号	描述
16	实心圆（默认）
17	实心三角形
18	实心菱形
19	实心圆+边框
21	空心圆
22	空心方形

美学映射

ggplot(data, aes(x = var1, y = var2, color = group, size = value)) + geom_point()

1、color：映射分类变量；

aes(color = group)

2、size：映射连续变量；

aes(size = measurement)

3、shape：区分组别；

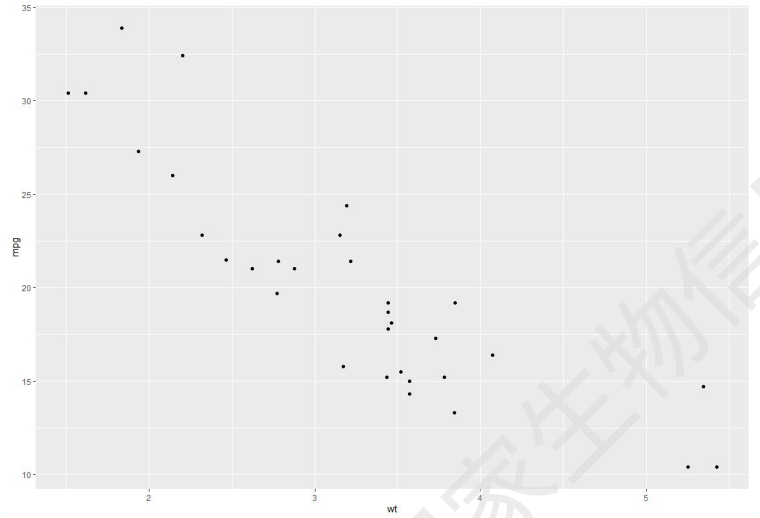
aes(shape = type)

4、alpha：设置透明度。

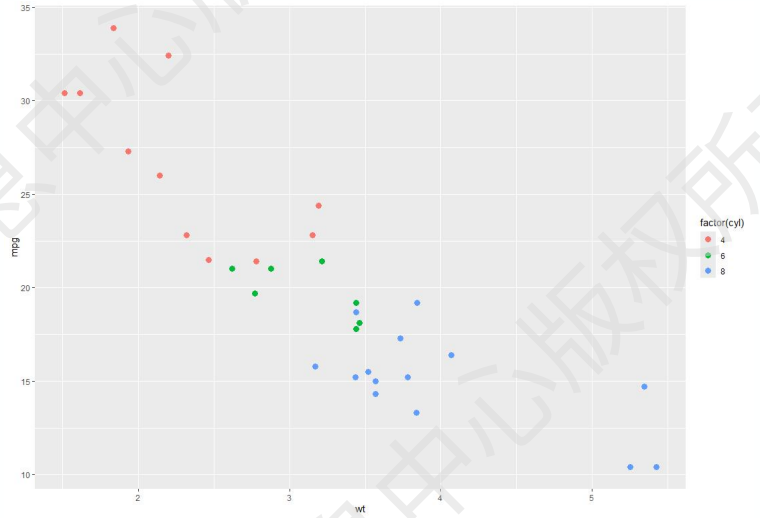
aes(alpha = confidence)

国家生物信息中心版权所有

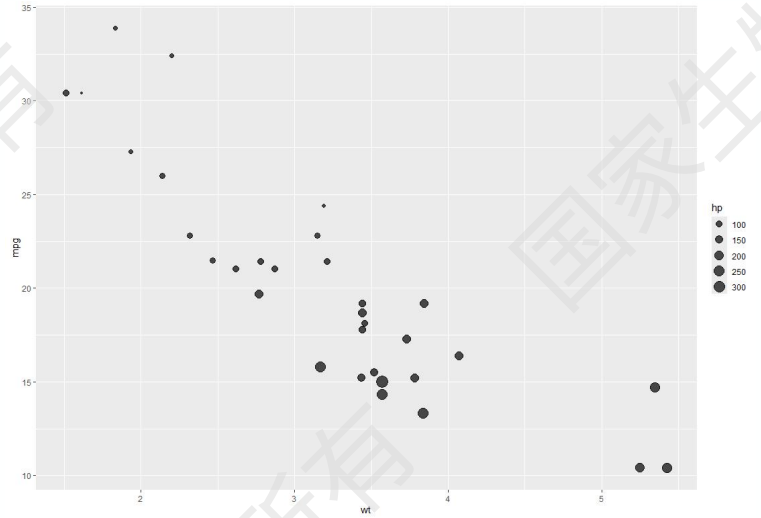
geom_point()的常见用法与案例展示



示例1



示例2



示例3

基本散点图

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

X轴为车重（wt），Y轴为油耗（mpg）

颜色按组分层

```
ggplot(mtcars, aes(x = wt, y = mpg,
color = factor(cyl))) + geom_point(size = 3)
```

根据气缸数（cyl）设置不同颜色

点大小映射连续变量

```
ggplot(mtcars, aes(x = wt, y = mpg,
size = hp)) +geom_point(alpha = 0.7)
```

马力（hp）控制点的大小，增加可视信息

国家生物信息中心版权所有

常用图层介绍：geom_boxplot()

geom_boxplot() 用于绘制箱线图（又称盒须图），适合比较不同组别的数值分布，是科研中非常常用的图表

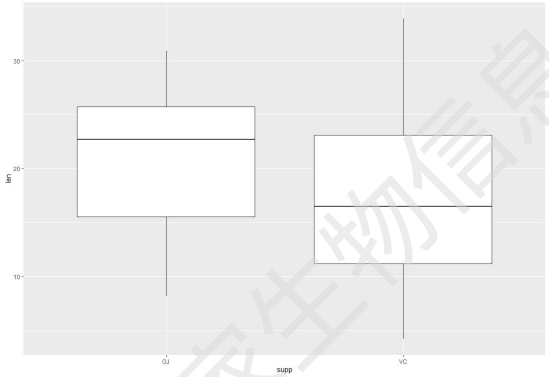
参数详解

1、mapping：设置箱线图的变量映射；
2、fill：填充颜色；
3、color：边框颜色；
4、alpha：透明度；
5、outlier.shape：控制异常值点的形状；
6、outlier.size：异常值点大小；
7、width：箱子的宽度；
8、notch：是否添加缺口；
9、coef：离群值定义的系数。

ggplot(data, aes(x = 分类变量, y = 数值变量)) + geom_boxplot()

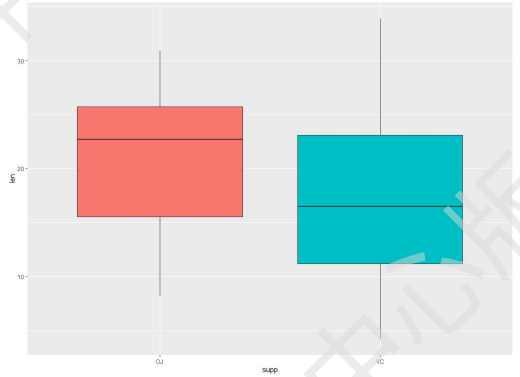
注意：x为分类变量、y为数值型变量

基本散点图



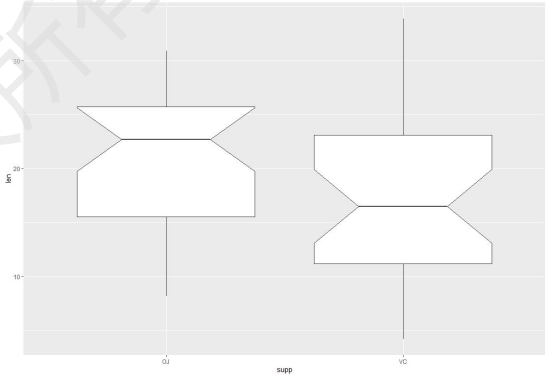
```
ggplot(ToothGrowth,
       aes(x = supp, y = len))
  + geom_boxplot()
```

颜色按组分层



```
ggplot(ToothGrowth,
       aes(x = supp, y = len, fill = supp))
  + geom_boxplot()
```

添加缺口



```
ggplot(ToothGrowth,
       aes(x = supp, y = len))
  + geom_boxplot(notch = TRUE)
```

1、设置theme_minimal()以拥有更干净、专业的图形外观；
2、labs(title, x, y)添加主标题与坐标轴标签；
3、设置scale_fill_brewer()使用预设调色板美化。

国家生物信息中心版权所有

标题与标签设置：labs()

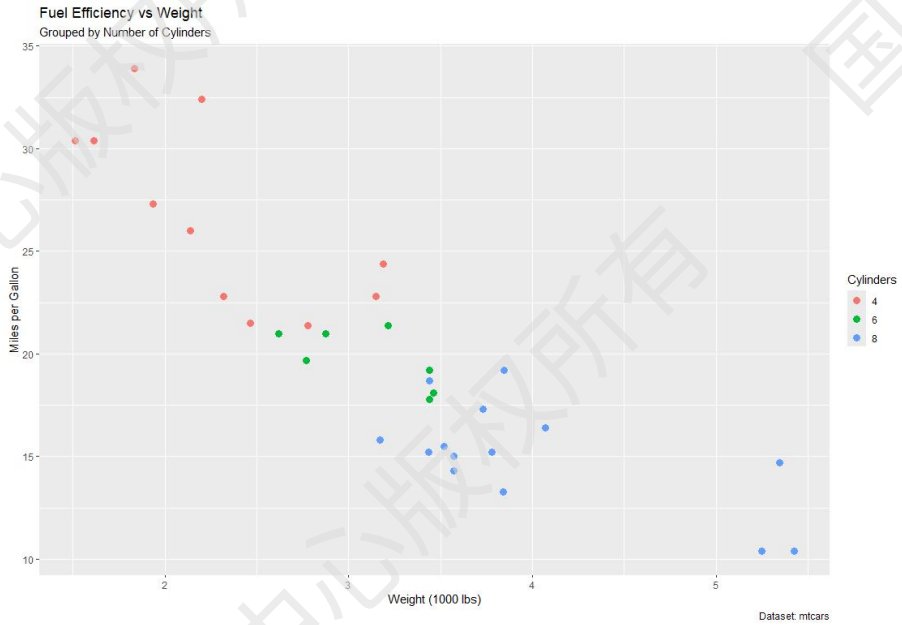
labs() 是 ggplot2 中控制图形注释的核心接口，可用于设置主副标题、坐标轴标签、图例标题与图注，增强图表的表达力和专业性

定义：

- 添加 主标题（title） 和 副标题（subtitle）；
- 指定 X/Y轴标签；
- 定义 图例标题（如颜色、大小、形状）；
- 添加 图注（caption），例如数据来源、备注信息。

```
labs(
    title = "Main Title",
    subtitle = "Subtitle",
    x = "X-axis Label",
    y = "Y-axis Label",
    caption = "Data Source",
    color = "Legend Title")
```

参数名称	说明	示例
title	主标题，位于图形顶部中央	title = " Main Title "
subtitle	副标题，位于主标题下方，用于补充说明	subtitle = " Sub explanation "
x	X 轴标题	x = " X Variable "
y	Y 轴标题	y = " Y Variable "
caption	图注，位于图形左下角，常用于注明数据来源	caption = " Source: ... "
color	映射变量对应的图例标题	color = " Group "
size	图例中控制点大小的标签	size = " Expression Level "
shape	图例中控制点形状的标签	shape = " Category "



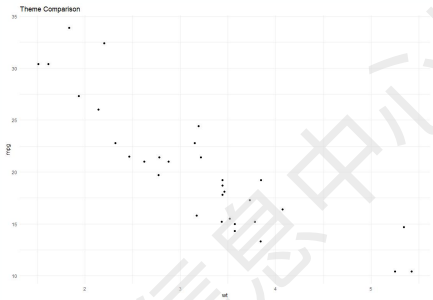
- 使用了颜色、形状等映射时，记得设置图例标题。
- 没有美学映射时，只需设置标题和轴标签。
- 图形用于报告或幻灯片时，必须设置主副标题与坐标轴标签。
- 数据来源或方法说明应统一写入 caption。
- 避免使用默认变量名作为图例标题。

图形主题设置：theme_*()

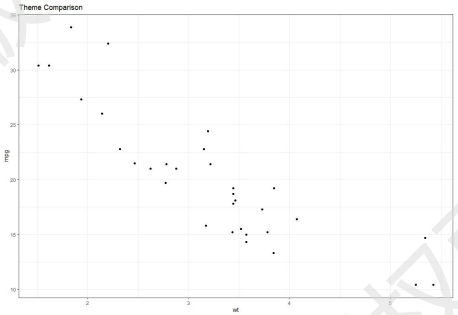
theme_*() 提供预设风格模板，适用于不同的展示需求，是图表专业美化的首选工具，可与 theme() 联合实现风格统一与个性调整

函数名称	风格特征	典型用途
theme_minimal()	极简设计、无边框、无背景、仅保留轴线	幻灯片展示、现代报告
theme_classic()	白底、明确的横纵坐标轴线、无背景色	学术出版、图书图示
theme_bw()	黑白强对比，带细网格线，清晰可打印	论文附件、黑白打印
theme_light()	类似 minimal，但有灰色边框和淡淡网格线	科研报告、演讲演示
theme_gray()（默认）	灰色背景+白网格线，ggplot2 默认风格	一般可视化展示
theme_void()	空白画布，无坐标轴线、无标签	自定义图像、地图展示

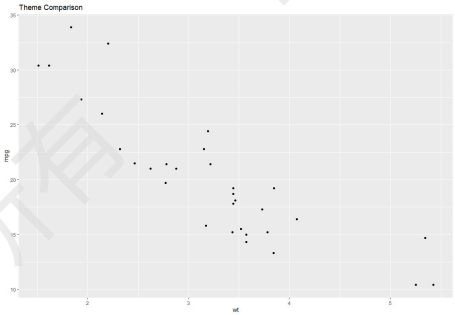
theme_minimal()



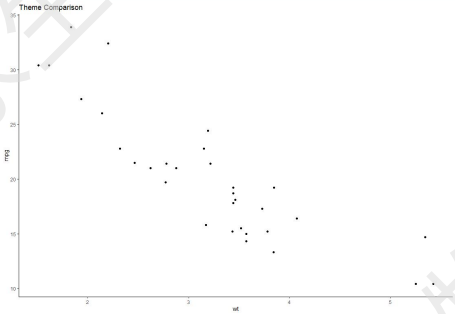
theme_bw()



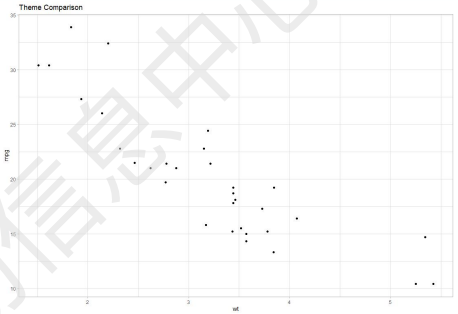
theme_gray()



theme_ classic()



theme_light()



theme_ void()

主题细节自定义：theme()

theme() 是 ggplot2 的“美工系统”，通过精细化控制图形各要素的样式，使图表在保持信息清晰的基础上达到专业与美观的统一

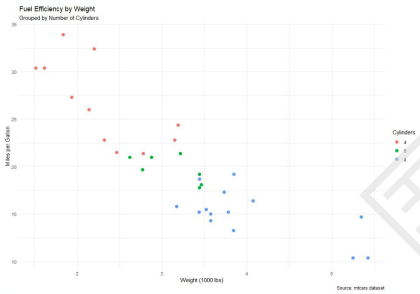
参数名	控制内容	常用设置示例
plot.title	主标题字体、大小、位置	element_text(hjust=0.5, size=16, face="bold")
plot.subtitle	副标题样式	element_text(size=12, hjust=0.5)
axis.title.x / .y	坐标轴标题样式	element_text(size=14, face="italic")
axis.text.x / .y	坐标轴刻度样式	element_text(angle=45, hjust=1)
legend.position	图例位置	"top", "bottom", "left", "none"
legend.title	图例标题字体样式	element_text(face = "bold")
legend.text	图例标签字体样式	element_text(size = 10)
panel.grid.major	主网格线	element_line(color = "gray90")
panel.grid.minor	次网格线	element_blank()（可隐藏）
panel.background	图形绘图区背景	element_rect(fill = "white")
plot.caption	图注样式	element_text(size = 9, color = "gray50")

国家生物信息中心版权所有

主题细节自定义：theme()

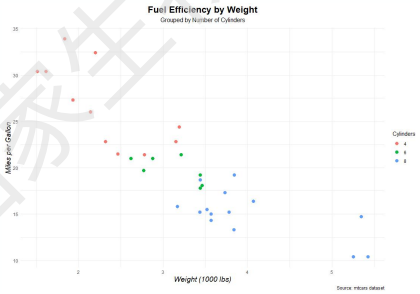
theme() 是 ggplot2 的“美工系统”，通过精细化控制图形各要素的样式，使图表在保持信息清晰的基础上达到专业与美观的统一

套用预设主题



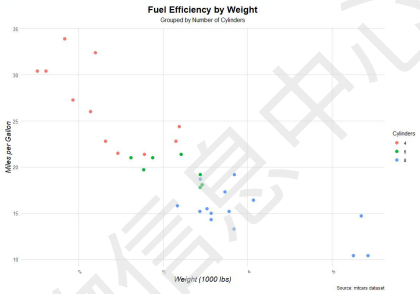

```
ggplot(mtcars, aes(x = wt,
y = mpg, color = factor(cyl)))
+
 geom_point(size = 3) +
 labs(
  title    = "",
  subtitle = "",
  x        = "",
  y        = "",
  color    = "",
  caption  = "")
```

调整多种标题



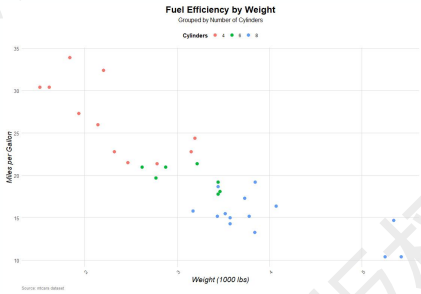

```
p_theme + theme(
   plot.title       =
element_text(hjust = 0.5, face
= "bold", size = 18),
   plot.subtitle    =
element_text(hjust = 0.5, size
= 12, margin = margin(b =
10)),
   axis.title       =
element_text(size = 14, face =
"italic"))
```

处理刻度文字



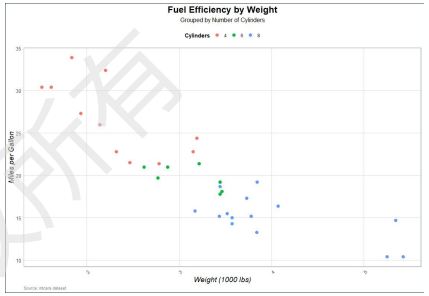

```
p_step2 + theme(
   axis.text.x      =
element_text(angle = 45, hjust
= 1, size = 10),
   axis.text.y      =
element_text(size = 10),
   panel.grid.major =
element_line(color = "gray85"),
   panel.grid.minor =
element_blank())
```

调整图例图注




```
p_step3 + theme(
   legend.position = "top",
   legend.title     =
element_text(face = "bold"),
   legend.text      =
element_text(size = 9),
   plot.caption     =
element_text(size = 8, color =
"gray50", hjust = 0)
)
```

调整背景边框



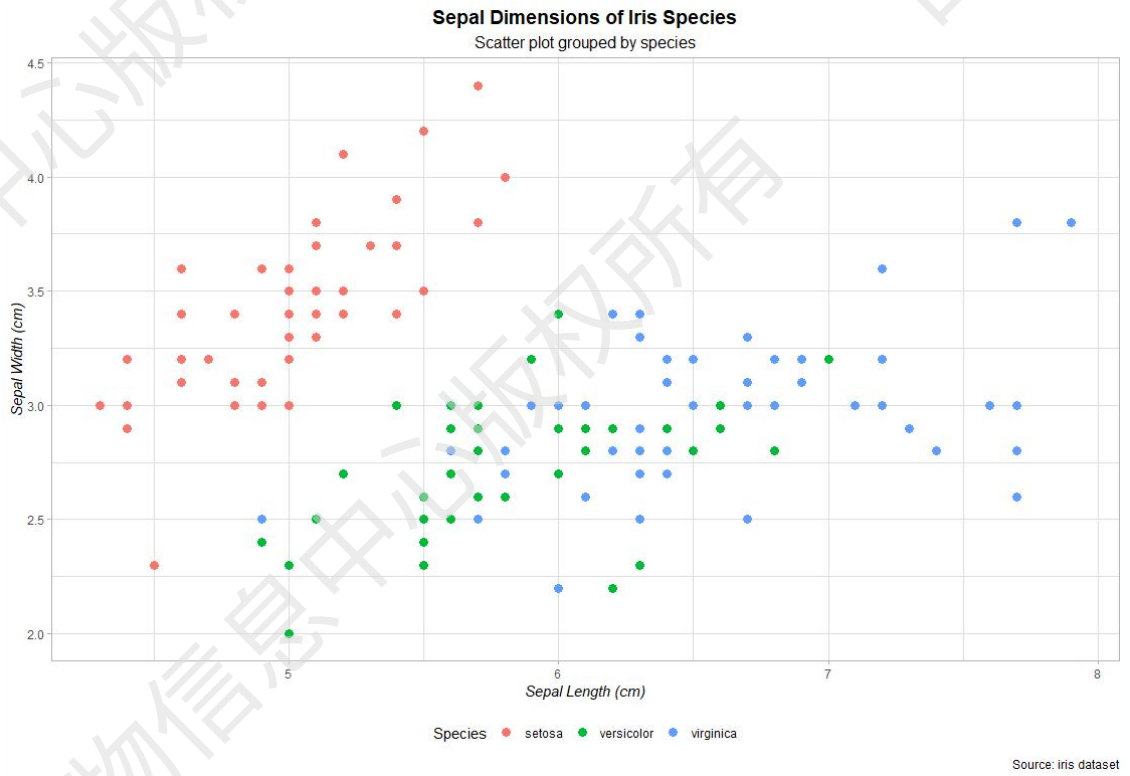

```
p_step4 + theme(
   panel.background =
element_rect(fill = "white"),
   panel.border     =
element_rect(color = "gray70",
fill = NA, linewidth = 0.8),
   plot.background  =
element_rect(fill = "white")
 )
```

国家生物信息中心版权所有

综合案例

该示例综合运用了 labs()、theme_light() 和 theme() 函数，演示了如何在一个完整图形中同时设置标题、副标题、坐标轴标签、图例名称，以及整体风格和细节美化

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point(size = 3) +
  labs(
    title = "Sepal Dimensions of Iris Species",
    subtitle = "Scatter plot grouped by species",
    x = "Sepal Length (cm)",
    y = "Sepal Width (cm)",
    caption = "Source: iris dataset",
    color = "Species"
  ) +
  theme_light() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 12),
    axis.title = element_text(face = "italic"),
    legend.position = "bottom"
  )
```



本模块小结

ggplot2语法结构与图层构建模块的主要目标是帮助学员掌握 ggplot2 的核心语法结构与可视化思维，从图形对象创建到图层叠加、美学映射与注释美化，逐步建立标准的数据可视化流程

ggplot图形构建

基本结构：ggplot(data, aes(...))+图层+注释+美化；

所有图形以加号 + 方式逐层构建，强调“图层式思维”

美学映射aes()

使用 color / fill / shape / size 等将变量映射到图形元素；

理解 group 用于折线图或箱线图分组绘制

图形注释与主题美化

labs() 添加标题、轴标签、图例说明与数据来源；

theme_*() 设置整体风格，theme() 精调字体、颜色、图例与背景。

常用图层函数

geom_point()散点图 geom_bar()柱状图 geom_boxplot() 箱线图注意熟悉 stat = "count" 与 stat = "identity" 的区别。

图层叠加与分面绘图

多图层组合，如 geom_point() + geom_smooth()

国家生物信息中心版权所有

实战练习

练习1：基础箱线图：

1. 使用 ggplot(data, aes(...)) + geom_boxplot() 绘制按组别分组的表达量箱线图；
2. X 轴为 group，Y 轴为 expr。

练习2：添加美学映射aes()

1. 让箱体填充颜色 (fill) 随 group 变化；
2. 观察不同组别颜色变化效果。

练习3：标题与坐标轴注释

1. 添加主标题、子标题、X / Y 轴标签与图例标题；
2. 主标题内容：“Expression Levels of Target Gene”。

练习4：主题美化

1. 选择一种预置主题（如 theme_minimal()）。
2. 在 theme() 中设置：
   - 主标题加粗、字号 16；
   - 子标题斜体、字号 12；
   - X 轴文字旋转 45°；
   - 图例放在右侧。

练习5：加分任务：

计算每组平均表达量，并将均值作为红色文字标注在箱线图上方。

实战要求及答案路径：/home/big/RBase/4-ggplot2_Syntax_Structure_and_Layer_Construction_Module

国家生物信息中心版权所有

05

高级图形绘制与分面布局

国家生物信息中心版权所有

一维分面绘图 – facet_wrap()

facet_wrap() 是 ggplot2 中最常用的分面函数之一，适用于根据某个分类变量对数据进行分组展示，即将一张图按组拆分为若干子图，每组数据独占一个图面，实现横向或纵向的排列，便于快速对比

基本语法结构：

facet_wrap(~ group_variable)

- 使用波浪线表示“以某变量进行分面”
- 注意该变量必须是分类型（factor）或字符串型，否则不建议直接用于分面

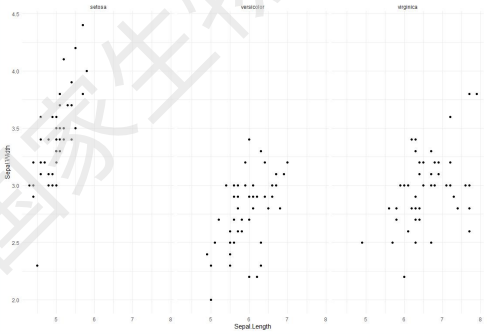
核心参数：

参数	类型	功能描述
nrow	整数	手动指定子图的行数。例如 nrow = 2 表示按两行排列。
ncol	整数	手动指定子图的列数。例如 ncol = 3 表示每行最多三个子图。
scales	字符串	控制子图的坐标轴是否共享。fixed / free / free_x / free_y

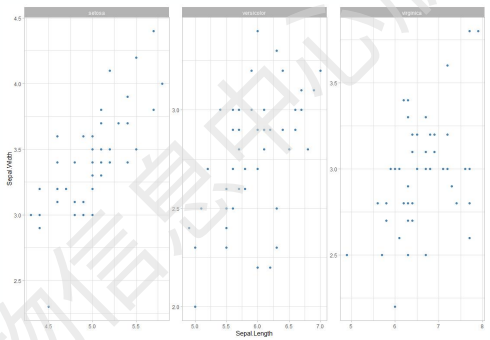
典型使用配合：

```
ggplot(data, aes(x, y)) +
        geom_xxx() +
        facet_wrap(~ 分组变量)
```

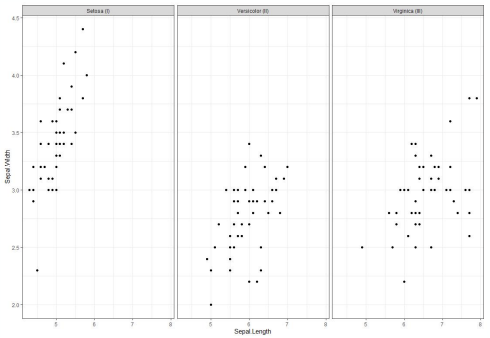
实战示例：



基础一维分面（固定坐标轴）



自由坐标、指定列数、横向排列



重命名分面标签

二维分面绘图 – facet_grid()

facet_grid() 适用于两个分类变量同时拆分图形的情形，它会生成一个按“行变量 × 列变量”交叉排列的网格矩阵，令不同组合在独立子图中呈现

基本调用方式：

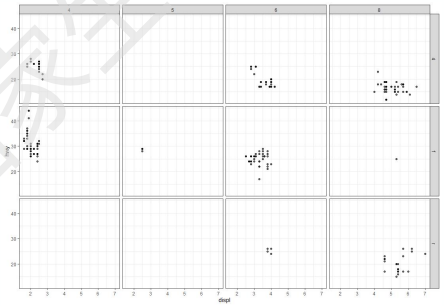
```
ggplot(data, aes(x, y)) +
  geom_xxx() +
  facet_grid(row_var ~ col_var)
```

- row_var：决定行的分面变量；
- col_var：决定列的分面变量；
- 若只想按行或列拆分，可写 facet_grid(group ~ .) 或 facet_grid(. ~ group)。

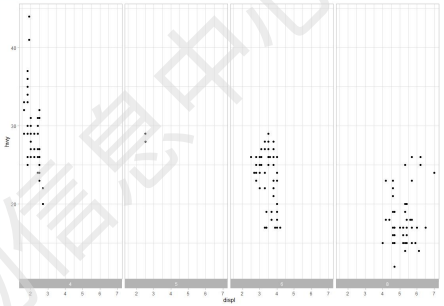
核心参数：

- scales：控制坐标轴是否独立
  - "fixed" 所有子图共用坐标范围；"free" x、y 坐标都独立；"free_x" / "free_y" 放开某一方向
- switch：移动分面标题
  - " x" 把列标题放到底部；"y" 把行标题放到左侧；
- space：子图宽高自适应（需要 scales = "free_x" / "free_y"）
- margins：显示汇总边际图（行或列总览）
- labeller：自定义分面标签

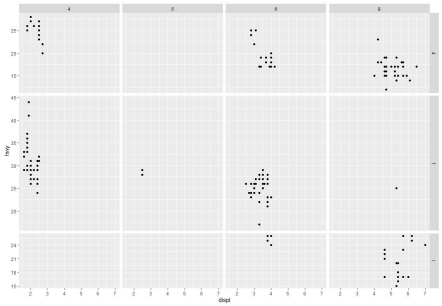
实战示例：



双变量完整交叉



只按列分面并切换标题位置



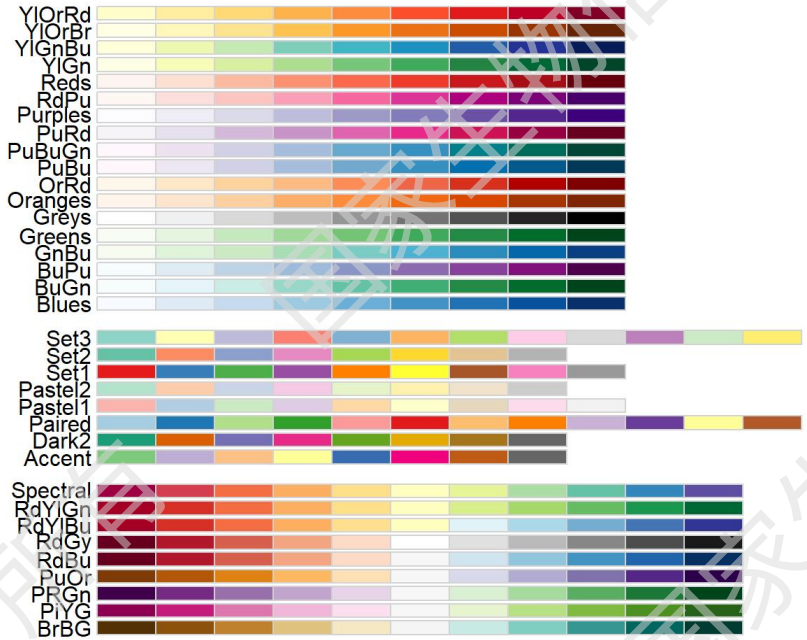
不同y轴范围、自动行高

国家生物信息中心版权所有

色板体系概览

R中常用的色板体系包括RColorBrewer的三大类型色板（离散、连续、双向渐变）、viridis系列（色盲友好、灰度打印友好），以及手动色板的自定义功能

RColorBrewer色板：



- Qualitative：离散/无序分类，常用为Set1、Pastel2；
- Sequential：连续单调渐变，常用为Blues、YlOrRd；
- Diverging：双向渐变强调中点，如 RdBu、PiYG。

viridis色板：

- 设计为色盲友好（色觉安全）
- 灰度打印友好
- 单调可读

手动色板：

- scale_fill_manual()
- scale_color_manual()

支持自定义HEX颜色变量

离散填色

scale_fill_brewer() 是 ggplot2 中最实用的分类填色方案之一，它通过标准化命名的色板为离散数据提供可靠、出版级配色效果。掌握常用色板特性、类别上限和色觉友好度，是提升科研图表质量的关键

```
geom_xxx(aes(fill = group)) + scale_fill_brewer(palette = "Set1", direction = 1)
```

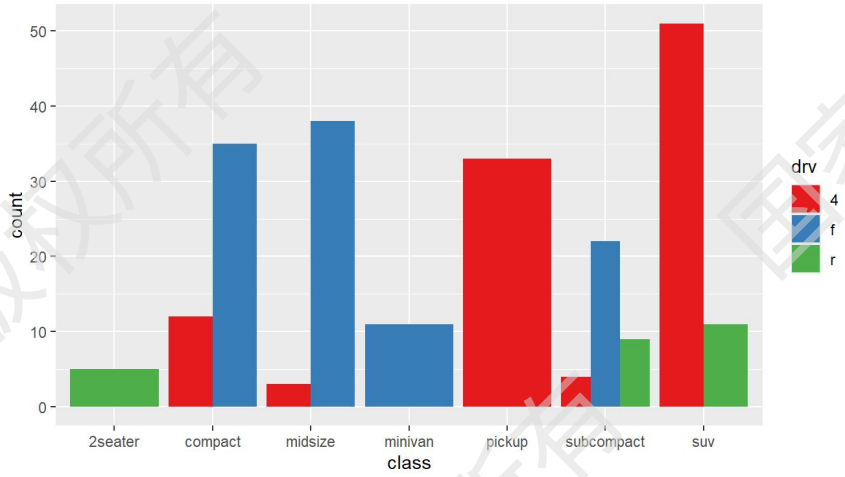
palette：指定使用的预设色板名称（Set1、Dark2等）

1：按默认顺序加载颜色

-1：反转色彩梯度顺序

基本示例：

```
ggplot(mpg, aes(class, fill = drv)) +
  geom_bar(position = "dodge") +
  scale_fill_brewer(palette = "Set1")
```



预设色板介绍：

色板名称	特点	推荐用途	最大类别数
Set1	对比强烈，醒目	分类较少，需突出差异	9
Dark2	深色、高识别度	深色背景下的图形；色觉友好	8
Pastel1	柔和、不刺眼	教学展示、幻灯片	9
Set3	类别较多、色彩轻盈	类别 >10 的演示场景	12
Paired	成对配色 (适用于有偶对含义的类别)	对称分组（如T1/T2， Control/Case)	12

- 根据组数选择合适色板（类别 ≤ 8：Set1、Dark2；类别 9–12：Set3、Paired；类别 > 12：建议使用 viridis::scale_fill_viridis_d() 或 scale_fill_manual() 自定义色彩
- 关注色觉可及性（色盲友好）-- 可选 Dark2、Accent 或 viridis 系列
- 某些类顺序（如“低-中-高”）在色板默认顺序中无法体现梯度感，可反转
- 与主题风格一致 -- 色彩搭配应与 theme_xxx() 一致，比如深灰背景搭配 Pastel1 效果较佳

连续填色

处理连续型变量（如表达量、密度值、频率、浓度）时，合适的渐变色映射对于传递数值差异和图形可读性至关重要。scale_fill_distiller()、scale_fill_gradient()和scale_fill_viridis_c()是用于连续填色的三类主要方法

O1

使用预设渐变色板（Brewer 系列）

- 该函数基于 RColorBrewer 中的 Sequential 或 Diverging 色板，适合表示单调变化或双向变化的连续变量。
- ggplot(df, aes(x, y, fill = value)) + geom_tile() + scale_fill_distiller(palette = "YlGnBu", direction = 1)
- palette：选择渐变色方案，如 "YlGnBu"（黄-绿-蓝）、"OrRd"（橙-红）、"RdBu"（红-蓝，双向分布）。
  direction：是否反转色板顺序，1 为默认，-1 反向。

O2

scale_fill_gradient()系列

- scale_fill_gradient()：指定 low 和 high 两端色（线性渐变）；
  scale_fill_gradient2()：同时指定 low、mid、high 及 midpoint，实现对称渐变；
- scale_fill_gradient(low = “#ffffd4”, high = “#006d2c”) # 简单渐变
  scale_fill_gradient2(low = “blue”, mid = “white”, high = “red”, midpoint = 0) #对称渐变
- 自由度高，适用于对配色要求精；
  可自定义断点和颜色强度，适配非均匀分布数据。

O3

scale_fill_viridis_c()

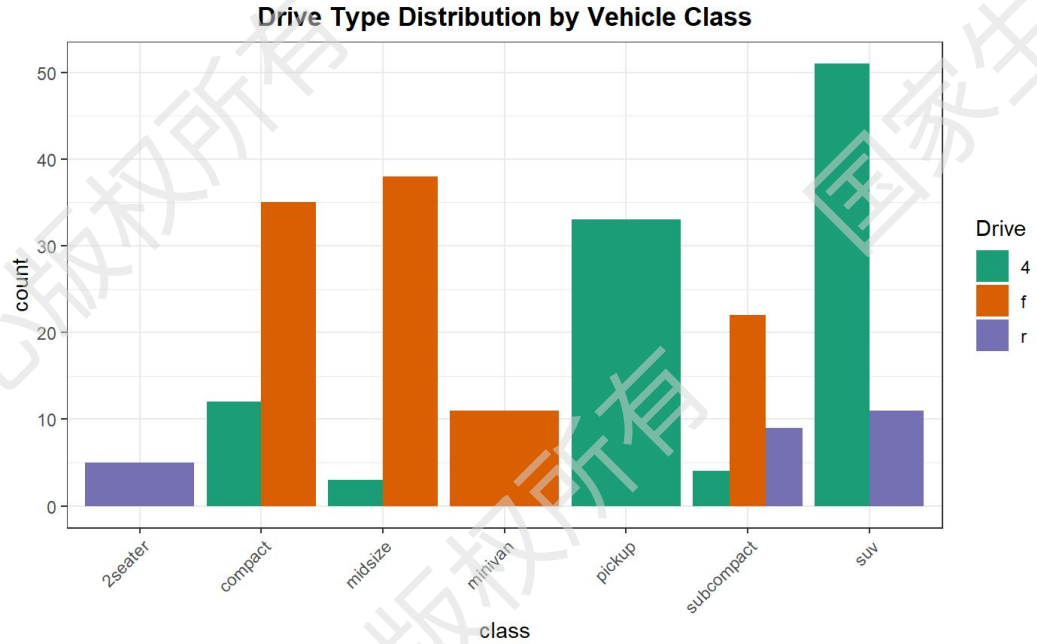
- viridis 色板来自 Python 的 matplotlib，特别设计用于：色觉无障碍（对常见红/绿色盲可读）；单调变化、黑白打印友好；各设备显示一致性好。
- scale_fill_viridis_c(option = "plasma")
- •"viridis"（默认）：深蓝 → 黄绿
  •"plasma"：紫 → 黄（高亮）
  •"inferno"：黑 → 红黄（暗背景效果好）
  •"magma"：黑紫红渐变
  •"cividis"：色盲最友好方案

国家生物信息中心版权所有

进阶示例

Part1：各种车型的驱动类型分布柱状图（离散色板）

```
ggplot(mpg, aes(class, fill = drv)) +
  geom_bar(position = "dodge") +
  scale_fill_brewer(palette = "Dark2")  +
  labs(title = "Drive Type Distribution by Vehicle Class",
       fill  = "Drive") +
  theme_bw() +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)
  )
```



以内置数据集 mpg 为对象，设定横轴映射到 class（车型类别），填充色映射到 drv（驱动方式：4、f、r）。因为未指定 y，后续 geom_bar() 会统计每个类别出现频次。

绘制柱状图并使用 “并排” 布局，使不同驱动方式在同一车型类别下并排显示，方便对比。

采用 RColorBrewer 的 Dark2 色板，为 drv 三个水平提供深色、对比度强且色盲友好的配色。

添加主标题与图例标题；fill = "Drive" 将图例名称从默认变量名 “drv” 改为易读的 “Drive” 。

使用黑框白底主题，增强正式感且便于印刷。

plot.title = element_text(face = "bold", hjust = 0.5)：将标题加粗并居中。

axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)：将横轴文字旋转 45°，同时调整水平和垂直对齐，避免长标签重叠。

进阶示例

Part2：喷发密度热图（连续色板）

```
ggplot(faithfuld, aes(waiting, eruptions, fill = density))
+
  geom_tile() +
  scale_fill_viridis_c(option = "magma") +
  labs(title = "Old Faithful Eruption Density",
       fill  = "Density") +
  theme_minimal()
```

- 使用内置数据框 faithfuld（老忠实间歇泉二维密度估计），将 x 轴映射到 waiting（两次喷发间等待时间，分钟），y 轴映射到 eruptions（喷发持续时间，分钟），填充色映射到 density（核密度估计值）。
- 以网格瓦片形式绘制热图，每个瓦片根据对应 (waiting, eruptions) 组合的密度赋予颜色。
- 采用 viridis 系列连续色板中的 magma 方案，色觉友好且单调递增：黑紫 → 红橙 → 黄白，可突出高密度区域。
- 添加主标题与图例标题，将图例名称设为 “Density”。
- 使用极简主题，去除背景网格和边框，让色块成为视觉焦点。

国家生物信息中心版权所有

本模块小结

facet_wrap()

将数据按一个分类变量拆分为多个子图，自动横向或纵向排列，常用于比较同类指标在不同组别中的表现。

- 适合“单变量拆分”；
- 分类变量较多时建议指定列数；
- 搭配 theme() 调整标题或字体风格以提升可读性。

facet_grid()

同时根据两个变量的组合进行图形拆分，形成网格状排列，适合多因素对比与交互分析。

- 适用于“分组 × 条件”类图；
- 可用于双因子实验数据呈现；
- 注意空组合会生成空白图，需预处理数据或考虑使用 facet_wrap()。

为离散型变量提供对比度清晰、出版级别的色彩映射，常见于柱状图、箱线图等填色图层。

- 各色板最大类别数有限（如 Set1 最多 9 类），超出需使用 scale_fill_manual()；
- 色板名称区分大小写，拼写错误将导致渲染失败；
- fill 对应填充，color 对应边框，勿混用。

scale_fill_brewer()

option = "magma" / "plasma" / "inferno" 等为不同渐变方案；

- geom_tile()、geom_density_2d_filled() 等热图、密度图适用；
- 若为发表、教学场景，优先考虑 viridis 系列；
- 可搭配 theme_minimal()、theme_void() 等突出主图信息。

scale_fill_viridis_c()

国家生物信息中心版权所有

实战练习

练习1：数据探索与清理

1. 加载数据集并检查缺失值。
2. 根据车型类别（class）和驱动方式（drv）进行数据筛选。
3. 检查数值列（如 displ 和 hwy）的分布情况。

练习2：使用 facet_wrap() 进行一维分面

1. 按照 class 分面，展示不同车辆类别的油耗分布。
2. 将不同的驱动方式（drv）映射到填充色上。

练习3：使用 facet_grid() 进行二维分面

1. 根据 drv（驱动方式）和 cyl（气缸数）进行分面，形成一个网格矩阵。
2. 为每个子图设置独立的 Y 轴范围。

练习4：使用配色与主题美化

1. 使用连续色板 scale_fill_viridis_c() 为油耗（hwy）设置颜色映射。
2. 选择合适的主题进行图形美化。具体来说，使用 theme_light() 来突出数据和避免过于复杂的背景，适合数据对比明显的图表；或者使用 theme_bw() 为正式出版风格的图形提供简洁的黑白主题。

练习5：自定义图例与标签

1. 反转图例顺序，确保图例与数据的呈现顺序一致。
2. 自定义分面标签，以提高可理解性，例如将 compact 替换为 “Compact Vehicles”。

实战要求及答案路径：/home/big/RBase/5-Advanced_Plotting_and_Faceted_Layouts_Module

国家生物信息中心版权所有

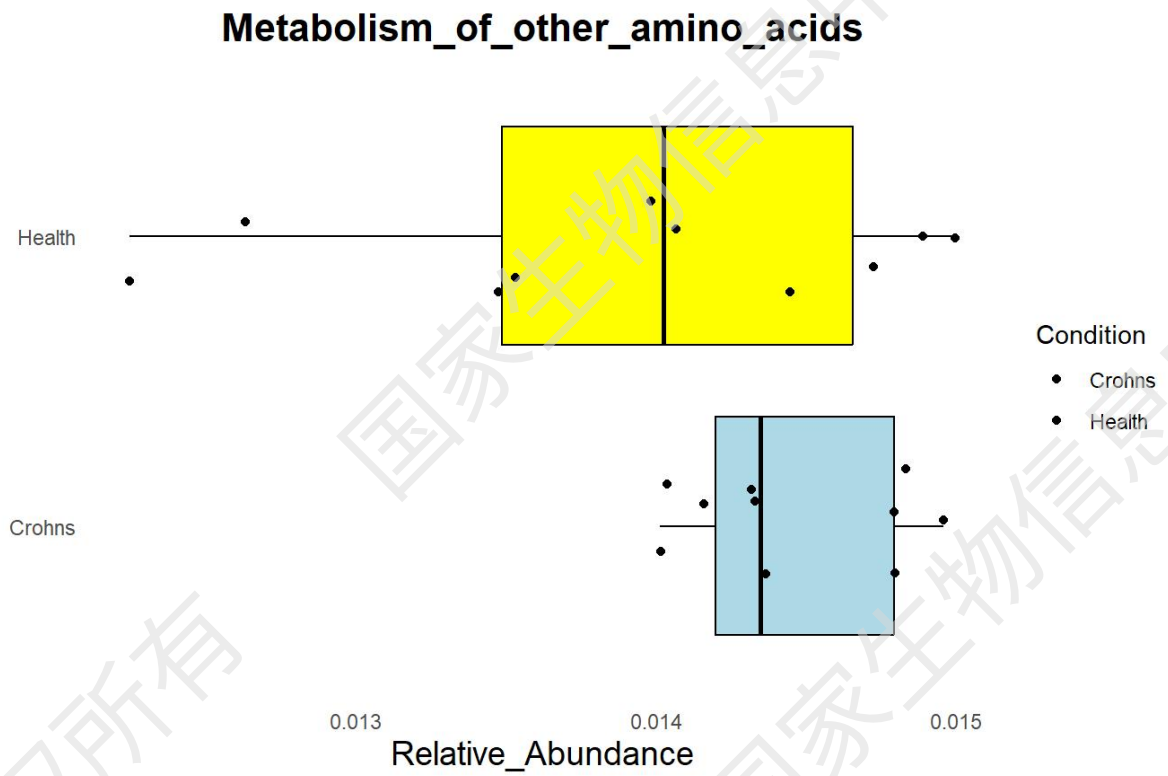
06

科研常用图形类型实战

国家生物信息中心版权所有

案例1：水平箱线图

通过绘制箱线图比较不同组别的相对丰度（或其它数值变量），便于展示数据的分布、离群点等



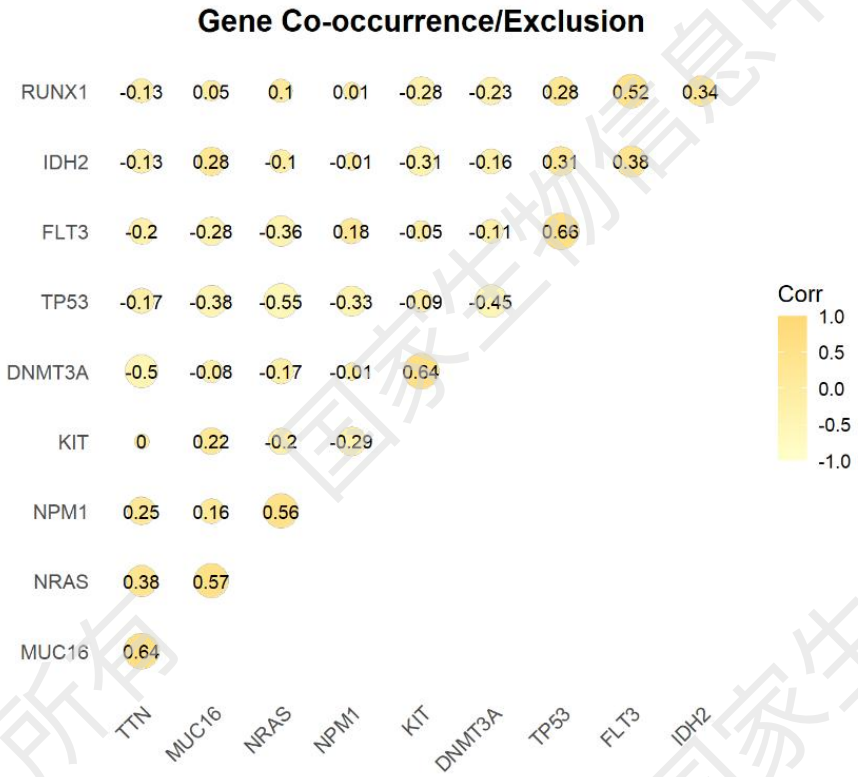
- 使用 ggplot() 函数映射数据，定义 x 和 y 轴，并通过 fill 为不同组别指定颜色。
- 通过 geom_boxplot() 绘制箱线图，设置颜色、边框等属性。
- 使用 geom_jitter() 添加散点，帮助展示个体数据。
- 自定义图表样式、坐标轴、标签等，通过 theme_minimal()、theme() 等函数实现。

基因表达数据、环境数据等比较分析

国家生物信息中心版权所有

案例2：相关性图

绘制基因、变量或样本之间的相关性矩阵图，显示它们之间的相互关系



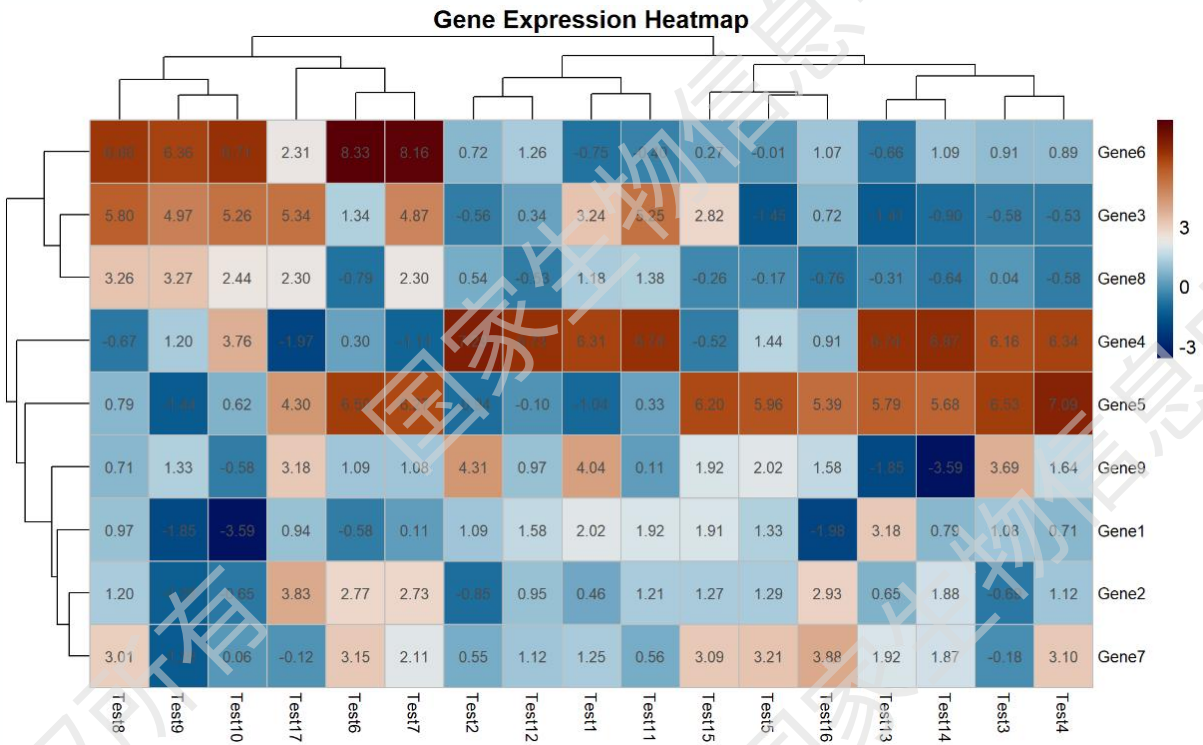
- 使用cor()函数计算数据框中的皮尔逊相关性，得到相关性矩阵。
- 使用ggcorrplot()函数绘制相关性热图，设置颜色、样式、标签等。
- 通过scale_color_manual()定义颜色梯度，确保图表信息准确传达。
- 设置标题、图例、坐标轴标签等，确保图表易读。

基因表达数据的相关性分析，样本间关系分析

国家生物信息中心版权所有

案例3：聚类热图

通过热图展示基因、样本等之间的相似性，采用颜色渐变显示数值范围



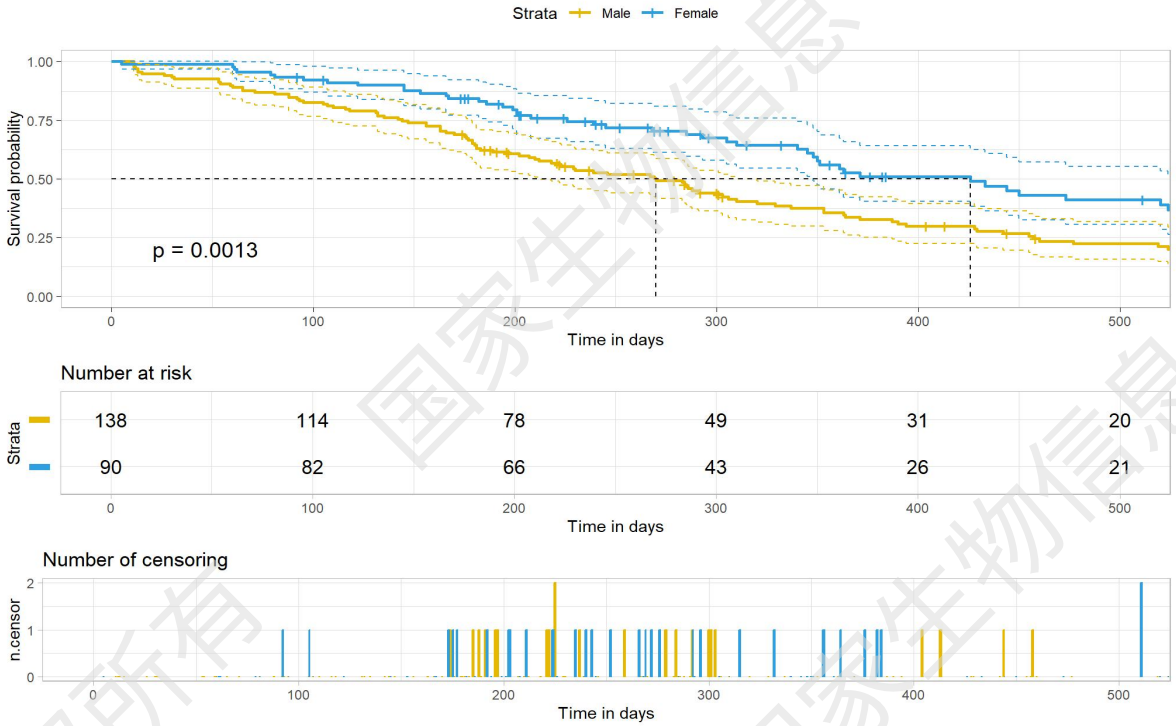
- 使用 pheatmap() 函数绘制热图，支持聚类、颜色渐变和数值显示。
- 通过 cluster_rows 和 cluster_cols 设置行列的聚类方式。
- 使用 color 设置颜色方案，通常配合 RColorBrewer 或 scico 包中的配色方案。
- 显示图例和坐标轴标签，调整字体和边框颜色。

基因表达数据、临床数据聚类分析结果的热图展示

国家生物信息中心版权所有

案例4：生存曲线图

展示生存分析结果，通常使用 Kaplan-Meier 方法进行分析，能够比较不同组别的生存概率



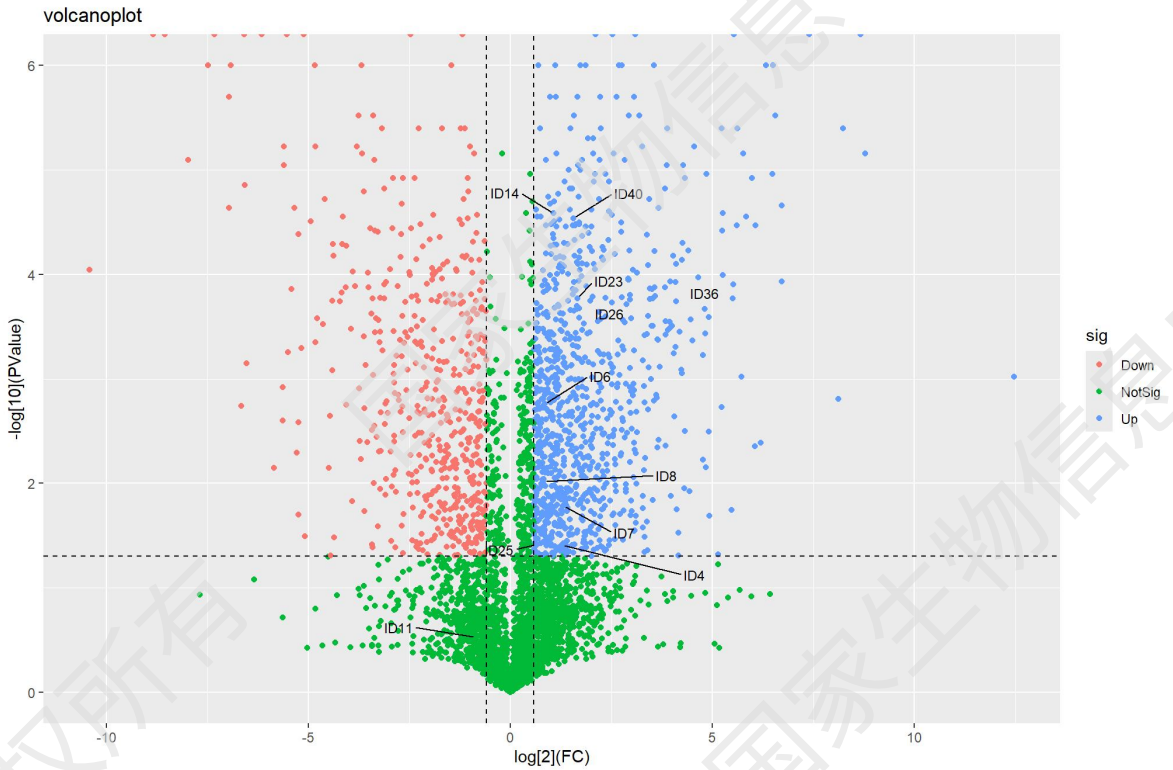
- 使用 survfit() 函数拟合生存模型，依据某一因素（如性别、治疗方法等）进行分组。
- 使用 ggsurvplot() 函数绘制 Kaplan-Meier 生存曲线图，展示每组的生存概率。
- 自定义图例、风险表、置信区间、p 值等内容。

临床数据的生存分析，药物治疗效果评估

国家生物信息中心版权所有

案例5：火山图

通过火山图展示基因表达分析中上调和下调基因的显著性，通常结合 logFC 和 p 值



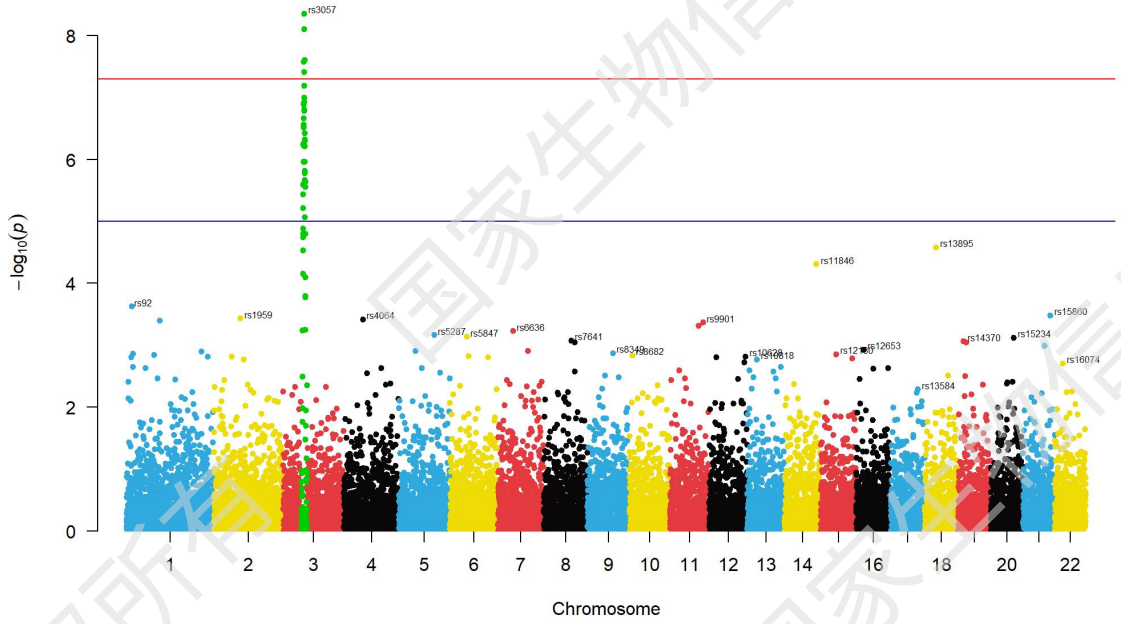
- 使用 ggplot2 绘制火山图，x 轴为 log2(FC)，y 轴为 -log10 (P-value)。
- 设置阈值：通过 geom_vline() 和 geom_hline() 添加显著性阈值（如 logFC、P-value）。
- 使用 geom_text_repel() 给显著的基因加上标签，避免标签重叠。

基因差异表达分析，药物靶点筛选

国家生物信息中心版权所有

案例6：曼哈顿图

在 GWAS（基因组宽度关联研究）中用于展示 SNP 与性状之间的关联性，显示显著的 SNP



- 使用 manhattan() 函数绘制曼哈顿图，输入包含 SNP 名称、染色体编号、碱基位置和 p 值的数据。
- 使用 suggestiveline 和 genomewideline 添加显著性阈值的横线。
- 根据 P 值标记显著的 SNP 并进行注释。

GWAS 数据可视化，用于基因组关联分析

国家生物信息中心版权所有

07

图形导出格式与自动生成

国家生物信息中心版权所有

通用绘图函数的定义与使用

在科研数据分析中，常常需要对多个不同的数据文件进行可视化。为了提高工作效率，我们可以定义一个通用的绘图函数，接受每个数据文件的路径作为输入，并绘制相同风格的图表

创建绘图函数 → **函数示例** → **函数使用**

- 读取数据文件
- 绘制基础图表（如散点图、箱线图等）
- 返回图表对象，方便后续处理和保存

```
# 绘制散点图
  p <- ggplot(data, aes(x = Variable1, y = Variable2)) +
    geom_point() +
    theme_minimal() +
    ggtitle(paste("Plot for", basename(file_path)))
  return(p)
}
```

```
# 传入数据文件路径
plot1 <- plot_data("data_file1.csv")
plot2 <- plot_data("data_file2.csv")

# 显示图表
print(plot1)
print(plot2)
```

国家生物信息中心版权所有

使用ggsave()保存单个图表

ggsave()是ggplot2包提供的一个非常方便的图表保存函数。它支持多种文件格式（如 PNG、PDF、JPEG、SVG 等），可以将绘制的图表保存到指定的路径。

ggsave(filename, plot, width, height, units, dpi, ...)

- filename：保存文件的路径和名称。可以指定文件格式，如 .png、.pdf、.jpeg 等。
- plot：要保存的图表对象。通常是通过 ggplot() 绘制的图表，或者是已经绘制好的图表对象（如 p）。
- width 和 height：指定图表的宽度和高度，单位可以是英寸、厘米或毫米。
- units：指定单位（in 表示英寸，cm 表示厘米，mm 表示毫米）。
- dpi：图像的分辨率（通常为 300 DPI），适用于需要打印的图表。
- ...：其它参数，比如图表的背景颜色、格式等。

根据文件的扩展名，ggsave()自动识别要保存的格式，并将图表保存为指定格式的文件

国家生物信息中心版权所有

批量绘制并保存多个图表

通过 list.files() 函数，我们可以获取指定文件夹下的所有 CSV 文件路径，并对每个文件调用绘图函数，生成对应的图表

```
# 获取多个 CSV 文件的路径
files <- list.files(path = "data_folder", pattern = "*.csv", full.names = TRUE)

# 使用 lapply() 批量绘制图表
lapply(files, function(file) {
  # 调用 plot_data() 函数绘制图表
  plot <- plot_data(file)
  # 保存图表
  ggsave(filename = paste0("output_folder/", tools::file_path_sans_ext(basename(file)), ".png"), plot = plot)
})
```

list.files()：列出data.foler文件夹下所有csv文件

lapply()：遍历每个文件

ggsave()：将每个生成的图表保存为 PNG 格式的文件

-> Next slide

国家生物信息中心版权所有

图表拼接

在科研和数据分析过程中，我们经常需要将多个图表拼接成一个图形文件。R 提供了多个包，如 gridExtra、cowplot 和 patchwork，来实现图表的拼接和排列

gridExtra

gridExtra 包提供了 grid.arrange() 函数，可以将多个图表按指定方式拼接。它的功能非常强大，支持行列拼接以及复杂的布局设计。

```
# 加载 gridExtra 包
library(gridExtra)
# 按照每行两个图表的方式拼接
grid.arrange(p1, p2, p3, ncol = 2)
```

ncol=2代表每行显示两个图表

cowplot

cowplot 包可以更容易地拼接 ggplot2 创建的图表，尤其是在图表的对齐、标题和注释的处理上，提供了更多的灵活性。

```
# 加载 cowplot 包
library(cowplot)
# 拼接每行两个图表
plot_grid(p1, p2, p3, ncol = 2)
```

plot_grid() 可调整图标间距和布局

patchwork

patchwork包提供了非常简便的图表拼接方式。与 gridExtra 和 cowplot 相比，patchwork 更侧重于图表的布局和排版，具有灵活的拼接操作符。

```
# 加载 patchwork 包
library(patchwork)
# 默认p1/p2/p3拼接在一行
combined_plot <- p1 + p2 + p3
```

通过简单的运算符组合多个图表

本模块小结

本模块介绍了如何通过 R 语言实现图表的批量绘制与保存，提升科研数据分析中可视化图表的处理效率

通用绘图函数的定义与使用

数据读取、绘图操作、返回图表对象

使用ggsave()保存多个图表

重要参数：filename、plot、width、height和dpi

图表拼接

gridExtra

cowplot

patchwork

批量绘制多个图表

关键函数：list.files()、lapply()和ggsave()

批量保存多个图表

两种方法：lapply()和for循环

国家生物信息中心版权所有

实战练习

练习1：使用gridExtra拼接多个图表

1. 使用 grid.arrange() 函数将多个图表（如散点图和箱线图）拼接成一个图形文件，并将其保存。

练习2：使用 cowplot 拼接多个图表

1. 使用 plot_grid() 将多个图表拼接，并保存为 PNG 格式。

练习3：使用patchwork拼接多个图表

1. 使用 patchwork 拼接图表，设计一个简单的例子，保存拼接后的图表。

练习4：使用 ggsave() 保存单个图表

1. 使用 ggsave()函数，将p1和p2图表分别保存为 PNG 文件与PDF文件；
2. 要求：设置图表宽度为 8 英寸，高度为 6 英寸，分辨率为 300 DPI。

练习5：批量绘制图表并保存

1. 使用 lapply() 或 for 循环批量绘制并保存散点图和箱线图。

实战要求及答案路径：/home/big/RBase/7-Figure_Export_Formats_and_Automated_Generation_Module

国家生物信息中心版权所有

08

项目实操

国家生物信息中心版权所有

实战练习

Comprehensive Bioinformatics Workflow in R: From Raw Counts to Biological Insights

任务 1：数据准备与导入

1. 加载表达矩阵和样本注释数据（CSV格式）
   - ☑文件正确读取为 counts 和 coldata 两个数据框
   - ☑确保样本名一致，行名设置正确
   - ☑使用 DESeqDataSetFromMatrix() 成功构建对象

任务2：差异表达分析

1. 使用 DESeq2 运行标准差异分析流程
   - ☑得到 dds 对象，并运行 DESeq()
   - ☑使用 results() 提取差异分析结果表格
   - ☑正确添加基因 ID 为列（res$gene）
2. 提取显著差异表达基因（DEGs）
   - ☑筛选条件：padj < 0.05 且 log2FC > 1 或 < -1
   - ☑提取上调、下调基因列表

所需文件路径：/home/big/RBase/pratical_training/Simulate_Reality_Pipeline

国家生物信息中心版权所有

实战练习

Comprehensive Bioinformatics Workflow in R: From Raw Counts to Biological Insights

任务 3：可视化差异分析结果

1. 绘制火山图
   - ☑使用 ggplot2 展示所有基因
   - ☑明确突出显著DEG（红色/尺寸加大）
   - ☑添加标题与坐标轴注释
2. 绘制表达热图
   - ☑使用 vst() 正规化表达数据
   - ☑提取 top 50 DEGs 进行聚类热图展示
   - ☑添加样本分组注释，使用 pheatmap() 绘图

任务4：功能富集分析

1. GO 富集分析（生物过程）
   - ☑使用 enrichGO() 运行分析
   - ☑使用 dotplot() 可视化前 10 个 GO 条目
   - ☑标注图例和标题正确

所需文件路径：/home/big/RBase/pratical_training/Simulate_Reality_Pipeline

国家生物信息中心版权所有

THANK YOU

国家生物信息中心版权所有