

生态统计学

沈国春、李勤

2025-09-21

Contents

前言	5
0.1 课程在线资源	5
0.2 学习方式	6
1 统计编程基础	7
1.1 R 语言介绍	7
1.2 数值向量创建与基本统计计算	15
1.3 字符型数据处理与向量索引操作	17
1.4 数据框结构理解与多类型数据管理	19
1.5 列表数据结构与数据分组管理	22
1.6 外部数据导入与条件筛选分析	25
1.7 缺失值和异常值的识别处理	28
1.8 描述性统计分析与基础数据可视化	32
1.9 条件判断、循环结构与函数编程	36
1.10 现代数据科学工具包应用	41
1.11 图形语法与科学绘图	48
1.12 综合练习	55

前言

这是一本关于生态统计学的教材，旨在为生态学研究者提供实用的数据分析方法。本书结合 R 语言，介绍生态学研究中常用的统计方法。

本书特点：

- 面向生态学研究实际问题
- 基于 R 语言实现
- 包含大量生态数据案例
- 循序渐进的教学安排

本书使用以下 R 包：

```
install.packages(c(  
  "tidyverse", "vegan", "lme4", "ggplot2",  
  "bookdown", "knitr", "rmarkdown"  
)
```

0.1 课程在线资源

课程简明手册

- 网页版 <https://guochunshen.github.io/ecological-statistics>
- PDF 版 <https://gitee.com/gcshen/ecological-statistics/blob/master/docs/ecological-statistics.pdf>
- 原代码 <https://gitee.com/gcshen/ecological-statistics>

0.2 学习方式



Chapter 1

统计编程基础

1.1 R 语言介绍

- 理解为什么生态学专业需要学习 R 语言
- 掌握 R 和 RStudio 的安装和配置
- 建立良好的项目文件组织习惯
- 了解 R 包管理的基本方法

1.1.1 为什么生态学专业需要学习 R?

1.1.1.1 数据分析能力的必要性

现代生态学研究产生海量数据：野外调查数据、实验室测量数据、遥感影像数据、基因序列数据等。传统的 Excel 已无法满足复杂的统计分析需求，而 R 语言提供了完整的数据科学工具链。

1.1.1.2 可重现研究的科学要求

- **重现性危机：**越来越多的科学研究无法被重现，影响科学可信度
- **R 脚本的优势：**
 - 每一步分析都有记录，任何人都可以重现你的分析过程
 - 错误检查：代码可以被审查，减少人为错误
 - 版本控制：分析过程的每次修改都有记录

1.1.1.3 职业发展的核心竞争力

- **学术界要求：**顶级期刊越来越要求提供数据和代码
- **就业市场需求：**环保部门、研究院所、咨询公司都需要数据分析能力
- **跨学科合作：**与计算机科学、统计学等领域合作的桥梁
- **终身学习：**编程思维有助于快速学习新的分析方法

1.1.1.4 开源免费的经济优势

- **成本优势：**SPSS 单机版数万元，SAS 更昂贵，R 完全免费
- **功能更新：**商业软件更新缓慢，R 社区每天都有新功能
- **全球社区：**遇到问题可以在全球社区寻求帮助
- **未来保障：**开源软件不会因为公司倒闭而消失

1.1.1.5 学术发表的必要工具

- **期刊要求:** Nature、Science 等顶级期刊要求提供分析代码
- **同行评议:** 审稿人可以检查你的分析方法是否正确
- **引用优势:** 提供代码的论文被引用次数更高
- **学术诚信:** 透明的分析过程展现严谨的科学态度

1.1.1.6 国际交流的通用语言

- **国际会议:** 生态学国际会议上, R 是数据分析的主流工具
- **合作研究:** 与国外学者合作时, R 是共同的工作语言
- **在线学习:** 全球最优秀的生态学分析教程都使用 R
- **职业流动:** 掌握 R 可以在全球范围内寻求工作机会

1.1.2 R 语言简介与趣味应用

1.1.2.1 什么是 R 语言?

- **统计计算语言:** 专门为统计分析和数据可视化设计的编程语言
- **开源免费:** 由全球统计学家共同维护发展
- **交互式环境:** 可以立即看到代码执行结果
- **扩展性强:** 超过 18,000 个专业扩展包

1.1.2.2 如何获取帮助?

```
# 查看函数帮助文档
?plot
help("plot")

# 搜索帮助文档
??"regression"

# 示例代码演示
example(plot)
example(lm)
demo(persp)
demo(graphics)
demo(Hershey)
demo(plotmath)

# 在线资源:
# - R 官方文档: https://cran.r-project.org/manuals.html
# - RStudio 社区: https://community.rstudio.com
```

1.1.2.3 R 语言的趣味应用示例

```
# 安装必要包 (首次需要)
install.packages(c("ggplot2", "ggridge", "gapminder"))
library(ggplot2)
library(ggridge)
library(gapminder)

# 使用 gapminder 数据集 (包含各国多年生态经济数据)
```

```
# 绘制动态变化图
fig <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, color = continent)) +
  geom_point() +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  labs(title = '年份: {frame_time}', x = '人均 GDP', y = '预期寿命') +
  transition_time(year) +
  ease_aes('linear')

# 渲染并保存动画
# 需要先安装 gifski 包: install.packages("gifske")
animate(fig, renderer = gifske_renderer("./imgs/gapminder_animation.gif"),
        width = 800, height = 600, fps = 10, duration = 15)

# 提示: 运行后会生成展示生态经济数据随时间变化的动画
# 可以清楚地看到不同大陆国家生态经济指标的变化趋势
```

1.1.2.3.1 生态数据动态可视化

1.1.2.4 电子水母

```
# 加载必要的包
library(ggplot2)
library(gganimate) # 用于创建动画
library(dplyr)      # 用于数据处理

# 设置参数 - 减少点数提高性能
i <- 0:2000 # 从 10000 减少到 2000 个点
x <- i %% 200 # R 中的模运算
y <- i / 43

# 预计算固定变量 (不依赖于 t)
k <- 5 * cos(x/14) * cos(y/30)
e <- y/8 - 13
d <- (k^2 + e^2)/59 + 4
a <- atan2(k, e) # 注意 R 中 atan2 的参数顺序是 (y, x)

# 预计算所有帧数据 (向量化操作)
t_values <- seq(0, 10 * pi, by = pi / 10) # 减少帧数: 从 20 到 10, 步长从 /20 到 /10

# 使用矩阵运算代替循环
c_val_matrix <- outer(d / 2 + e / 99, -t_values / 18, "+")
q_matrix <- 60 - 3 * sin(a * e) + outer(k * (3 + 4 / d), sin(d ^ 2 - 2 * t_values), "*")

x_matrix <- q_matrix * sin(c_val_matrix) + 200
y_matrix <- (q_matrix + 9 * d) * cos(c_val_matrix) + 200

# 创建数据框 (更高效的方式)
all_frames <- data.frame(
  t = rep(t_values, each = length(i)),
  X = as.vector(x_matrix),
  Y = as.vector(y_matrix)
)
```

```

# 创建动画
anim <- ggplot(all_frames, aes(X, Y)) +
  geom_point(size = 0.2, alpha = 0.2, color = "white") +
  theme_void() +
  theme(plot.background = element_rect(fill = "black", color = NA),
        panel.background = element_rect(fill = "black", color = NA)) +
  coord_cartesian(xlim = c(0, 400), ylim = c(0, 400)) +
  transition_time(t) +
  shadow_mark(past = FALSE, alpha = 0.05) # 保留轨迹痕迹

# 渲染动画（可能需要几分钟时间）
animate(anim,
        fps = 20,
        duration = 30,
        width = 900,
        height = 900,
        renderer = gifski_renderer())

# 保存动画（可选）
# anim_save("animation.gif", animation = last_animation())

```

```

install.packages("audio")
install.packages("dplyr")
library(audio)
library(dplyr)
notes <- c(A = 0, B = 2, C = 3, D = 5, E = 7, F = 8, G = 10)
pitch <- "D D E D G F# D D E D A G D D D5 B G F# E C5 C5 B G A G"
duration <- c(rep(c(0.75, 0.25, 1, 1, 1, 2), 2),
              0.75, 0.25, 1, 1, 1, 1, 0.75, 0.25, 1, 1, 1, 1, 2)
bday <- data_frame(pitch = strsplit(pitch, " ")[[1]],
                     duration = duration)

bday <-
  bday %>%
  mutate(octave = substring(pitch, nchar(pitch)) %>%
    {suppressWarnings(as.numeric(.))} %>%
    ifelse(is.na(.), 4, .),
    note = notes[substr(pitch, 1, 1)],
    note = note + grep("#", pitch) -
      grep("b", pitch) + octave * 12 +
      12 * (note < 3),
    freq = 2 ^ ((note - 60) / 12) * 440)

tempo <- 120
sample_rate <- 44100

make_sine <- function(freq, duration) {
  wave <- sin(seq(0, duration / tempo * 60, 1 / sample_rate) *
    freq * 2 * pi)
  fade <- seq(0, 1, 50 / sample_rate)
  wave * c(fade, rep(1, length(wave) - 2 * length(fade)), rev(fade))
}

```

```
bday_wave <-
  mapply(make_sine, bday$freq, bday$duration) %>%
  do.call("c", .)

play(bday_wave)
```

1.1.2.4.1 生成音乐

1.1.2.4.2 更多有趣功能

- 动态报告: 用 R Markdown 生成可交互报告
- 网络爬虫: 抓取生态监测站数据
- 地图绘制: 可视化物种分布
- 机器学习: 预测生态变化趋势

1.1.3 环境设置和配置

1.1.3.1 R 语言安装

1.1.3.1.1 下载和安装 R Windows 系统:

1. 访问 <https://cran.r-project.org/bin/windows/base/>
2. 下载最新版 R 安装包 (.exe)
3. 右键以管理员身份运行安装程序
4. 安装路径不要包含中文或空格
5. 勾选“创建桌面快捷方式”

macOS 系统:

1. 访问 <https://cran.r-project.org/bin/macosx/>
2. 下载最新版 R 安装包 (.pkg)
3. 双击安装, 可能需要右键“打开”绕过 Gatekeeper 限制
4. 或通过 Homebrew 安装: `brew install --cask r`

Linux 系统:

- Ubuntu/Debian: `sudo apt-get install r-base`
- CentOS/RHEL: `sudo yum install R`

1.1.3.1.2 下载和安装 RStudio

1. 访问 <https://www.rstudio.com/products/rstudio/download/>
2. 选择适合你系统的 RStudio Desktop 免费版
3. 安装注意事项:
 - Windows: 确保已安装 R 后再安装 RStudio
 - macOS: 可能需要允许来自“未识别开发者”的应用
 - Linux: 可能需要安装依赖库 `libssl-dev` 等

1.1.3.1.3 常见安装问题解决

- 中文路径问题: 安装路径和用户名不要包含中文
- 防火墙拦截: 临时关闭防火墙或添加 R/RStudio 为例外

- 镜像源设置: 安装后运行:

```
options(repos = c(CRAN="https://mirrors.tuna.tsinghua.edu.cn/CRAN/"))
```

- 依赖缺失:

- Windows: 安装 Rtools
- macOS: 安装 Xcode 命令行工具
- Linux: 安装开发工具链

```
# 在 RStudio 中运行这行代码, 应该显示 R 的版本信息
R.version.string

# 检查基本功能是否正常
1+1
plot(1:10)
```

1.1.3.1.4 验证安装

1.1.3.1.5 在 VSCode 中配置 R 环境 为什么选择 VSCode?

- 轻量高效: 启动速度快, 占用资源少
- 扩展性强: 丰富的扩展生态系统
- 多语言支持: 同时支持 R、Python、Markdown 等多种语言
- 版本控制集成: 内置 Git 支持, 便于代码管理
- 远程开发: 支持 SSH、容器、WSL 等远程开发环境

VSCode 安装和配置步骤:

1. 安装 VSCode

- 访问 <https://code.visualstudio.com/>
- 下载适合你操作系统的版本并安装

2. 安装 R 扩展

- 打开 VSCode, 点击左侧扩展图标 (或按 Ctrl+Shift+X)
- 搜索 “R” 并安装 “R” 扩展 (由 REditorSupport 开发)
- 搜索 “R Debugger” 并安装调试器扩展

3. 基本配置

- 打开设置 (Ctrl+,)
- 搜索 “r.rterm” 设置 R 执行路径
- 搜索 “r.rpath” 设置 R 语言服务器路径
- 推荐设置:

```
{
  "r.rterm.windows": "C:\\Program Files\\R\\R-4.3.1\\bin\\x64\\R.exe",
  "r.rterm.mac": "/usr/local/bin/R",
  "r.rterm.linux": "/usr/bin/R",
  "r.lsp.path": "/usr/lib/R/bin/R",
  "r.sessionWatcher": true,
  "r.bracketedPaste": true
}
```

4. 常用功能

- 代码执行: Ctrl+Enter 执行当前行或选中代码
- 代码补全: 自动提供函数和参数建议
- 语法高亮: 彩色显示代码结构
- 调试功能: 设置断点, 逐步调试代码
- 绘图查看: 内置绘图查看器
- Markdown 支持: 直接编写和预览 R Markdown 文档

5. 推荐扩展

- **Rainbow CSV**: 彩色显示 CSV 文件
- **GitLens**: 增强的 Git 功能
- **Bracket Pair Colorizer**: 彩色匹配括号
- **Project Manager**: 项目管理工具
- **Live Share**: 实时协作编程

6. 优势对比

- 相比 **RStudio**: 更轻量, 启动更快, 扩展更多
- 相比纯文本编辑器: 提供完整的 IDE 功能
- 适合场景: 大型项目、多语言开发、远程开发

注意事项:

- 确保已安装 R 后再配置 VSCode
- 路径设置需要根据实际安装位置调整
- 首次使用可能需要安装相关依赖包

1.1.4 项目文件夹结构设置

建立良好的文件组织习惯是数据分析的基础:

```
生态学R语言课程/
第01课-森林调查/
    数据/
    脚本/
    结果/
第02课-物种名称/
    数据/
    脚本/
    结果/
...
参考资料/
```

1.1.5 基本包管理

1.1.5.1 R 包的概念

- **什么是 R 包**: R 包是扩展 R 功能的代码、数据和文档集合
- **包的作用**: 提供专业统计方法、可视化工具、数据导入等功能
- **生态学常用包**: vegan(群落分析)、ggplot2(可视化)、dplyr(数据处理) 等

1.1.5.2 Rtools 的作用

- **Windows 专用**: 用于编译需要 C/C++/Fortran 代码的 R 包
- **适用场景**:
 - 安装需要编译的包 (如部分生态学模型包)
 - 开发自己的 R 包
 - 使用某些高性能计算功能
- **安装方法**: 从 CRAN 下载对应 R 版本的 Rtools 安装包

1.1.5.3 包管理基础

```
# 检查已安装的包
installed.packages()

# 安装新包 (从 CRAN)
install.packages("ggplot2")
```

```
# 从 GitHub 安装开发版包
# install.packages("devtools")
# devtools::install_github(" 作者/包名 ")

# 加载包
library(ggplot2)

# 更新所有包
update.packages()

# 查看包帮助文档
help(package="ggplot2")
```

1.1.5.4 如何寻找合适的包

- CRAN 任务视图: <https://cran.r-project.org/web/views/>
– 如 Environmetrics 任务视图包含生态学相关包
- RStudio 包推荐: 通过 RStudio 的 Packages 面板浏览
- 学术文献: 参考领域内论文使用的方法和包
- 社区推荐: R-bloggers、Stack Overflow 等平台

1.1.6 工作目录设置

1.1.6.1 什么是工作目录?

- 简单理解: 就像你办公桌上的文件夹, R 会默认从这个文件夹里找文件
- 作用: 告诉 R 在哪里读取数据和保存结果
- 类比: 就像在图书馆找书需要知道书架位置一样

1.1.6.2 为什么要设置工作目录?

- 方便管理: 所有课程文件可以分类存放
- 避免错误: R 能准确找到你的数据文件
- 提高效率: 不用每次都输入完整文件路径

1.1.6.3 基本操作

```
# 查看当前工作目录 (就像查看你现在在哪个文件夹)
getwd()

# 设置工作目录 (告诉 R 使用哪个文件夹)
# 注意: 路径中的斜杠方向
setwd("C:/Users/你的用户名/生态学 R 语言课程")

# 列出当前目录下的文件 (看看这个文件夹里有什么)
list.files()

# 小技巧: 在 RStudio 中可以通过菜单设置工作目录更简单:
# Session → Set Working Directory → Choose Directory
```

1.1.6.4 注意事项

- 路径中不要有中文
- Windows 系统使用正斜杠”/“或双反斜杠”\”

- 建议为每节课创建单独的子文件夹

1.1.7 课前准备检查清单

- R 软件安装成功
- 能够运行简单的 R 代码
- 创建了课程文件夹结构
- 理解了学习 R 语言的重要性
- 准备好投入时间学习编程思维

1.2 数值向量创建与基本统计计算

1.2.1 生态学背景

在野外森林调查中，测量树木胸径（DBH，距地面 1.3 米处的直径）是评估森林生长状况的基本方法。我们需要计算样地内树木的平均胸径来了解林分特征。

1.2.2 演示数据

```
# 某样地内 10 棵马尾松的胸径测量值（单位：厘米）
tree_dbh <- c(15.2, 18.7, 22.1, 19.5, 16.8, 20.3, 17.9, 21.4, 19.2, 18.6)
```

1.2.3 课堂演示过程

```
# 1. 创建胸径数据向量
tree_dbh <- c(15.2, 18.7, 22.1, 19.5, 16.8, 20.3, 17.9, 21.4, 19.2, 18.6)

# 2. 查看数据
tree_dbh
length(tree_dbh) # 查看测量了多少棵树

# 3. 计算平均胸径
mean_dbh <- mean(tree_dbh)
mean_dbh

# 4. 计算总树数
tree_count <- length(tree_dbh)
tree_count

# 5. 简单的数学运算
max_dbh <- max(tree_dbh) # 最大胸径
min_dbh <- min(tree_dbh) # 最小胸径
total_dbh <- sum(tree_dbh) # 胸径总和
```

1.2.4 R 语言知识点详解

1.2.4.1 向量创建函数 c()

- 是什么: `c()` 是 `combine` 的缩写，用于将多个值组合成一个向量
- 语法: `c(值 1, 值 2, 值 3, ...)`
- 重要特点:
 - 向量中的所有元素必须是同一类型（数值、字符或逻辑）
 - 如果混合不同类型，R 会自动转换为最通用的类型

- 向量是 R 中最基本的数据结构
- 常见错误：忘记加逗号分隔值，如 `c(1 2 3)` 是错误的
- 最佳实践：给向量起有意义的名字，如 `tree_dbh` 而不是 `x`

1.2.4.2 变量赋值操作符 `<-`

- 是什么：将右边的值赋给左边的变量名
- 语法：变量名 `<-` 值
- 为什么用 `<-` 而不是 `=`：
 - `<-` 是 R 的传统赋值符号，更清晰地表示赋值方向
 - `=` 也可以用，但在某些情况下可能引起混淆
 - 建议统一使用 `<-` 保持代码风格一致
- 变量命名规则：
 - 只能包含字母、数字、点 `(.)` 和下划线 `(_)`
 - 不能以数字开头
 - 区分大小写
 - 建议使用有意义的名称

1.2.4.3 基本统计函数

1.2.4.3.1 `mean()` - 计算平均值

- 语法：`mean(x, na.rm = FALSE)`
- 参数说明：
 - `x`: 数值向量
 - `na.rm`: 是否移除缺失值，默认 `FALSE`
- 返回值：数值，向量的算术平均数
- 注意事项：如果向量包含 `NA` 值，结果会是 `NA`，除非设置 `na.rm = TRUE`

1.2.4.3.2 `length()` - 计算向量长度

- 语法：`length(x)`
- 作用：返回向量中元素的个数
- 应用场景：统计样本数量、检查数据完整性

1.2.4.3.3 `max()` 和 `min()` - 最大值和最小值

- 语法：`max(x, na.rm = FALSE)`, `min(x, na.rm = FALSE)`
- 作用：找出向量中的最大值和最小值
- 参数：与 `mean()` 相同，可以设置 `na.rm` 参数

1.2.4.3.4 `sum()` - 求和

- 语法：`sum(x, na.rm = FALSE)`
- 作用：计算向量所有元素的总和
- 应用：计算总量、累计值等

1.2.4.4 数据查看

- 直接输入变量名：最简单的查看方式，直接显示变量内容
- 自动打印机制：R 会自动显示表达式的结果
- 向量显示格式：会显示 `[1]` 表示第一个元素的位置

1.2.5 课后练习

题目：某湿地调查中测量了 8 棵柳树的树高（单位：米）：`tree_height <- c(4.2, 5.1, 3.8, 4.7, 5.3, 4.9, 4.1, 4.6)`

请完成（仅使用本课学过的向量和基本统计函数）：

1. 创建树高向量并查看数据

2. 计算平均树高
3. 找出最高的树有多高
4. 统计总共测量了多少棵树
5. 计算所有树的总高度
6. 计算树高的标准差（提示：使用 `sd()` 函数）

1.3 字符型数据处理与向量索引操作

1.3.1 生态学背景

在野外鸟类观察中，需要记录观察到的鸟类物种名单，这是生物多样性调查的基础工作。我们要学会如何在 R 中管理物种名称数据。

1.3.2 演示数据

```
# 某公园早晨观察到的鸟类物种
bird_species <- c("白头鵙", "麻雀", "喜鹊", "乌鸦", "红嘴蓝鹊", "大山雀")
```

1.3.3 课堂演示过程

```
# 1. 创建鸟类物种名单
bird_species <- c("白头鵙", "麻雀", "喜鹊", "乌鸦", "红嘴蓝鹊", "大山雀")

# 2. 查看物种名单
bird_species
print(bird_species) # 另一种显示方法

# 3. 统计观察到的物种数量
species_count <- length(bird_species)
species_count

# 4. 访问特定位置的物种
bird_species[1] # 第一个物种
bird_species[3] # 第三个物种

# 5. 添加新观察到的物种
bird_species <- c(bird_species, "燕子")
bird_species

# 6. 字符串操作
paste("今天观察到", length(bird_species), "种鸟类")
```

1.3.4 R 语言知识点详解

1.3.4.1 字符型数据 (Character Data)

- 是什么：用引号包围的文本数据，R 中的基本数据类型之一
- 语法：“文本内容”或‘文本内容’
- 重要特点：

- 单引号和双引号都可以，但要成对使用
- 如果文本中包含引号，需要转义或使用另一种引号包围
- 字符型数据在 R 中以向量形式存储
- **与数值的区别：**
 - 字符型： "123" - 这是文本，不能进行数学运算
 - 数值型： 123 - 这是数字，可以进行数学运算
- **编码注意：**中文字符需要确保 R 的编码设置正确

1.3.4.2 向量索引 (Vector Indexing)

- **是什么：**通过位置编号访问向量中特定元素的方法
- **语法：**向量名 [位置]
- **索引规则：**
 - R 的索引从 1 开始 (不是 0!)
 - 可以使用负数排除特定位置： `bird_species[-1]` (排除第一个)
 - 可以一次访问多个位置： `bird_species[c(1,3,5)]`
- **超出范围：**如果索引超出向量长度，返回 NA
- **实际应用：**在生态学中用于提取特定样本、物种等

1.3.4.3 数据显示函数

1.3.4.3.1 `print()` 函数

- **语法：** `print(x)`
- **与直接输入变量名的区别：**
 - 直接输入：仅在交互模式下显示
 - `print()`：在脚本和函数中也会显示，更可控
- **应用场景：**在循环、函数中需要显示结果时

1.3.4.4 向量合并和扩展

- **添加元素：** `c(原向量, 新元素)`
- **重要概念：** R 中的向量是不可变的，每次“添加”实际上是创建新向量
- **效率考虑：**频繁添加元素效率较低，大量数据建议预先分配空间
- **实际应用：**野外调查中动态添加新发现的物种

1.3.4.5 字符串连接函数 `paste()`

- **语法：** `paste(..., sep = " ", collapse = NULL)`
- **参数详解：**
 - ...：要连接的多个元素
 - `sep`：分隔符，默认是空格
 - `collapse`：如果提供，将结果向量合并为单个字符串
- **相关函数：**
 - `paste0()`：等同于 `paste(..., sep = "")`，不使用分隔符
 - `sprintf()`：格式化字符串，类似其他语言的 `printf`
- **实际应用：**生成报告文本、标签、文件名等

1.3.5 课后练习

题目：某湿地调查中记录的水生植物：`water_plants <- c("荷花", "芦苇", "菖蒲", "水葫芦", "睡莲")`

请完成（使用向量、字符串操作、索引等已学内容）：

1. 创建植物名称向量并显示所有植物名称
2. 计算记录了多少种植物（使用 `length()` 函数）
3. 显示第 2 种和第 4 种植物的名称（使用向量索引）

4. 添加“慈姑”到植物名单中（使用 `c()` 函数合并）
5. 用 `paste()` 函数创建一句完整的调查报告
6. 尝试查找“芦苇”在向量中的位置（提示：使用 `which()` 函数和 `==` 运算符）

1.4 数据框结构理解与多类型数据管理

1.4.1 生态学背景

在植物群落调查中，需要同时记录多种信息：样方编号、物种名称、株高、是否存活等。这些不同类型的数据需要组织在一个表格中，这就需要用到数据框。

1.4.2 演示数据

```
# 某山坡 5 个样方的植物调查数据
plot_data <- data.frame(
  plot_id = c("S001", "S002", "S003", "S004", "S005"),
  species = c("马尾松", "杉木", "樟树", "栎树", "枫香"),
  height_m = c(12.5, 8.3, 15.2, 10.7, 9.8),
  diameter_cm = c(18.2, 12.5, 22.1, 16.8, 14.3),
  alive = c(TRUE, TRUE, FALSE, TRUE, TRUE)
)
```

1.4.3 课堂演示过程

```
# 1. 创建植物调查数据框
plot_data <- data.frame(
  plot_id = c("S001", "S002", "S003", "S004", "S005"),
  species = c("马尾松", "杉木", "樟树", "栎树", "枫香"),
  height_m = c(12.5, 8.3, 15.2, 10.7, 9.8),
  diameter_cm = c(18.2, 12.5, 22.1, 16.8, 14.3),
  alive = c(TRUE, TRUE, FALSE, TRUE, TRUE)
)

# 2. 查看数据框
plot_data
print(plot_data)

# 3. 查看数据框结构
str(plot_data) # 显示每列的数据类型

# 4. 查看数据框维度
nrow(plot_data) # 行数
ncol(plot_data) # 列数
dim(plot_data) # 行数和列数

# 5. 查看前几行和后几行
head(plot_data, 3) # 前 3 行
tail(plot_data, 2) # 后 2 行

# 6. 访问特定列
plot_data$species # 物种列
plot_data$height_m # 高度列
```

```
# 7. 计算统计量
mean(plot_data$height_m) # 平均高度
sum(plot_data$alive)      # 存活植物数量
```

1.4.4 R 语言知识点详解

1.4.4.1 数据框 (Data Frame)

- 是什么: R 中最重要的数据结构, 类似于 Excel 表格或数据库表
- 语法: `data.frame(列名 1 = 向量 1, 列名 2 = 向量 2, ...)`
- 核心特点:
 - 每列可以是不同的数据类型 (数值、字符、逻辑)
 - 但每列内部必须是相同类型
 - 所有列必须有相同的长度 (行数)
 - 每行代表一个观察单位, 每列代表一个变量
- 与矩阵的区别:
 - 矩阵: 所有元素必须是相同类型
 - 数据框: 不同列可以是不同类型
- 生态学应用: 完美适合存储野外调查数据

1.4.4.2 R 中的数据类型系统

1.4.4.2.1 数值型 (Numeric/Double)

- 特点: 包含小数点的数字
- 示例: 12.5, 8.3, 15.2
- 用途: 测量值、计数、比例等

1.4.4.2.2 字符型 (Character)

- 特点: 文本数据, 用引号包围
- 示例: "S001", " 马尾松", " 杉木"
- 用途: 名称、标识符、分类标签

1.4.4.2.3 逻辑型 (Logical)

- 特点: 只有两个值: TRUE 和 FALSE
- 示例: TRUE, FALSE
- 用途: 是/否判断、条件标记
- 运算: 可以进行数学运算 (TRUE=1, FALSE=0)

1.4.4.3 数据框结构查看函数

1.4.4.3.1 `str()` - Structure 函数

- 作用: 显示数据框的完整结构信息
- 显示内容:
 - 数据框类型和维度
 - 每列的数据类型
 - 前几个值的预览
- 读法技巧:
 - 'data.frame': 5 obs. of 5 variables - 5 行 5 列的数据框
 - \$ plot_id : chr - plot_id 列是字符型
 - \$ height_m: num - height_m 列是数值型

1.4.4.3.2 `head()` 和 `tail()`

- 语法: `head(x, n = 6)`, `tail(x, n = 6)`
- 作用: 查看数据框的开头或结尾几行
- 参数: `n` 指定显示的行数, 默认 6 行
- 应用场景:
 - 快速了解数据格式
 - 检查数据导入是否正确
 - 大数据集的初步查看

1.4.4.3.3 维度函数

- `nrow()`: 返回行数 (观察数量)
- `ncol()`: 返回列数 (变量数量)
- `dim()`: 返回维度向量 `c(行数, 列数)`
- `names()` 或 `colnames()`: 返回列名

1.4.4.4 数据框列访问

1.4.4.4.1 美元符号 \$ 操作符

- 语法: 数据框名 `$` 列名
- 特点:
 - 返回该列的向量
 - 支持名称自动补全 (在 RStudio 中)
 - 最常用的列访问方式
- 注意: 列名不需要引号

1.4.4.4.2 双括号 [[]] 操作符

- 语法: 数据框名 `[["` 列名 `"]"]` 或 数据框名 `[[列位置]]`
- 与 `$` 的区别:
 - 支持变量名作为列名: `df[[var_name]]`
 - 可以使用数字索引: `df[[1]]`

1.4.4.4.3 单括号 [] 操作符

- 语法: 数据框名 [行, 列]
- 特点: 返回数据框 (保持原结构)
- 示例: `plot_data[, "species"]` - 选择 species 列但保持数据框格式

1.4.5 课后练习

题目: 某河流生态调查数据:

```
river_survey <- data.frame(
  site_id = c("R01", "R02", "R03", "R04"),
  fish_species = c("草鱼", "鲤鱼", "鲫鱼", "青鱼"),
  length_cm = c(25.3, 18.7, 12.4, 31.2),
  weight_g = c(680, 420, 180, 1200),
  mature = c(TRUE, FALSE, FALSE, TRUE)
)
```

请完成 (使用数据框操作、向量计算等已学内容):

1. 创建数据框并显示整个数据框
2. 使用 `str()` 函数查看数据框的结构

3. 计算鱼类的平均长度和平均重量（使用 `mean()` 和 `$` 操作符）
4. 统计有多少条成熟的鱼（使用 `sum()` 和逻辑值运算）
5. 显示所有鱼类的名称（使用 `$` 操作符）
6. 计算最大和最小的鱼重量（使用 `max()` 和 `min()` 函数）
7. 创建一个包含调查总结的字符串（使用 `paste()` 函数）

1.5 列表数据结构与数据分组管理

1.5.1 生态学背景

不同栖息地类型的物种多样性差异很大。我们需要比较森林、草地、湿地三种生境中的物种数量，这种多组数据的管理需要用到列表结构。

1.5.2 演示数据

```
# 不同栖息地的物种数量调查（每个生境调查了 4 个样点）
forest_species <- c(25, 30, 28, 32)
grassland_species <- c(15, 18, 20, 16)
wetland_species <- c(12, 14, 11, 13)
```

1.5.3 课堂演示过程

```
# 1. 创建各栖息地物种数据
forest_species <- c(25, 30, 28, 32)
grassland_species <- c(15, 18, 20, 16)
wetland_species <- c(12, 14, 11, 13)

# 2. 创建栖息地数据列表
habitats <- list(
  forest = forest_species,
  grassland = grassland_species,
  wetland = wetland_species
)

# 3. 查看列表内容
habitats
str(habitats) # 查看列表结构

# 4. 访问列表中的元素
habitats$forest      # 使用 $ 访问
habitats[["forest"]] # 使用 [[]] 访问
habitats[[1]]         # 使用位置索引

# 5. 计算各生境的平均物种数
forest_mean <- mean(habitats$forest)
grassland_mean <- mean(habitats$grassland)
wetland_mean <- mean(habitats$wetland)

# 6. 创建结果向量
habitat_means <- c(forest_mean, grassland_mean, wetland_mean)
names(habitat_means) <- c("森林", "草地", "湿地")
```

```
# 7. 比较结果
habitat_means
max(habitat_means) # 哪个生境物种最多
which.max(habitat_means) # 物种最多的生境位置
```

1.5.4 R 语言知识点详解

1.5.4.1 列表 (List) 数据结构

- 是什么: R 中最灵活的数据结构, 可以存储不同类型、不同长度的数据
- 语法: list(名称 1 = 数据 1, 名称 2 = 数据 2, ...)
- 核心特点:
 - 每个元素可以是不同的数据类型 (向量、数据框、甚至其他列表)
 - 每个元素可以有不同的长度
 - 元素可以有名称, 也可以没有
 - 是递归数据结构 (可以包含其他列表)
- 与向量、数据框的区别:
 - 向量: 同类型, 一维
 - 数据框: 不同列可以不同类型, 但同列必须同类型, 二维表格
 - 列表: 最灵活, 可以存储任何类型的 R 对象

1.5.4.2 列表元素访问方法

1.5.4.2.1 美元符号 \$ 访问 (推荐)

- 语法: 列表名 \$ 元素名
- 特点:
 - 最直观、最常用的方法
 - 只能用于有名称的元素
 - 支持 RStudio 中的自动补全
 - 返回元素的原始类型

1.5.4.2.2 双括号 [[]] 访问

- 语法: 列表名 [[" 元素名 "]] 或 列表名 [[位置]]
- 特点:
 - 更灵活, 可以使用变量作为索引
 - 可以使用数字位置索引
 - 返回元素的原始类型
- 示例:

```
element_name <- "forest"
habitats[[element_name]] # 使用变量
habitats[[1]] # 使用位置
```

1.5.4.2.3 单括号 [] 访问

- 语法: 列表名 [元素名或位置]
- 特点: 返回包含该元素的子列表 (仍然是列表类型)
- 与 [[]] 的区别:
 - habitats[1] 返回包含第一个元素的列表
 - habitats[[1]] 返回第一个元素本身 (向量)

1.5.4.3 向量命名系统

1.5.4.3.1 names() 函数

- 作用: 为向量的每个元素分配名称
- 语法: `names(向量) <- c(" 名称 1", " 名称 2", ...)`
- 好处:
 - 增加数据的可读性
 - 便于后续的数据访问和处理
 - 在图表中自动显示有意义的标签
- 应用:

```
# 创建时命名
scores <- c(数学 = 95, 英语 = 88, 物理 = 92)

# 后续命名
scores <- c(95, 88, 92)
names(scores) <- c(" 数学", " 英语", " 物理")
```

1.5.4.4 比较和查找函数

1.5.4.4.1 max() 和 min()

- 作用: 找到向量中的最大值或最小值
- 语法: `max(x, na.rm = FALSE)`
- 参数: `na.rm` 控制是否忽略缺失值

1.5.4.4.2 which.max() 和 which.min()

- 作用: 返回最大值或最小值的位置索引
- 语法: `which.max(x)`
- 返回值: 整数, 表示最大值在向量中的位置
- 应用场景: 找到最优样地、最佳条件等
- 注意: 如果有多个相同的最大值, 只返回第一个的位置

1.5.4.5 数据组织策略

- 何时使用列表:
 - 存储相关但结构不同的数据
 - 分组存储实验数据
 - 存储分析结果的不同组成部分
- 命名的重要性:
 - 提高代码可读性
 - 便于数据访问
 - 减少错误发生
- 最佳实践:
 - 使用有意义的名称
 - 保持命名风格一致
 - 适当添加注释说明数据来源

1.5.5 课后练习

题目: 某保护区三个监测站的哺乳动物目击次数:

```
station_a <- c(8, 12, 6, 10)
station_b <- c(15, 18, 14, 16)
station_c <- c(3, 5, 2, 4)
```

请完成（使用向量、列表、命名等已学内容）：

1. 将三个监测站的数据组织成一个列表（使用 `list()` 函数）

2. 计算每个监测站的平均目击次数（使用 `mean()` 和列表访问）
3. 创建一个命名向量显示三个站点的平均值（使用 `names()` 函数）
4. 找出哪个监测站的平均目击次数最高（使用 `which.max()` 函数）
5. 计算所有监测站的总目击次数（使用 `sum()` 和向量合并）
6. 创建一个数据框，包含站点名称和对应的平均目击次数
7. 比较站点 A 和站点 B 的数据变异程度（使用 `sd()` 函数计算标准差）

1.6 外部数据导入与条件筛选分析

1.6.1 生态学背景

长期鸟类监测项目通常将数据保存在 Excel 或 CSV 文件中。我们需要学会将这些外部数据导入 R 中进行分析，并根据研究需要筛选特定时间段的数据。

1.6.2 演示数据文件 (`bird_monitoring.csv`)

```
date,month,species,count,observer
2023-03-15,3, 白头鹎,12, 张三
2023-03-15,3, 麻雀,25, 张三
2023-04-20,4, 喜鹊,8, 李四
2023-04-20,4, 白头鹎,15, 李四
2023-05-10,5, 燕子,20, 王五
2023-06-05,6, 麻雀,18, 张三
2023-06-05,6, 白头鹎,22, 张三
2023-07-12,7, 喜鹊,10, 李四
```

1.6.3 课堂演示过程

```
# 1. 读取鸟类监测数据
bird_data <- read.csv("bird_monitoring.csv", stringsAsFactors = FALSE)

# 2. 查看数据概况
head(bird_data)      # 前几行
tail(bird_data)      # 后几行
str(bird_data)        # 数据结构
summary(bird_data)   # 数据摘要

# 3. 查看数据维度
nrow(bird_data)     # 有多少条记录
ncol(bird_data)      # 有多少个变量

# 4. 查看具体列的信息
unique(bird_data$species)    # 观察到哪些物种
unique(bird_data$observer)    # 有哪些调查员
range(bird_data$month)        # 调查的月份范围
```

```

# 5. 筛选春季数据（3-5 月）
spring_birds <- subset(bird_data, month %in% c(3, 4, 5))
spring_birds

# 6. 筛选特定物种
baijitou_data <- subset(bird_data, species == "白头鵙")
baijitou_data

# 7. 条件组合筛选
spring_bajitou <- subset(bird_data, month %in% c(3, 4, 5) & species == "白头鵙")
spring_bajitou

# 8. 计算统计量
total_count <- sum(bird_data$count)
mean_count <- mean(bird_data$count)
paste("总观察个体数:", total_count, "平均每次观察:", round(mean_count, 1))

```

1.6.4 R 语言知识点详解

1.6.4.1 数据导入函数 `read.csv()`

- 作用：从 CSV（逗号分隔值）文件中读取数据，创建数据框
- 语法：`read.csv(file, header = TRUE, sep = ",", stringsAsFactors = FALSE, ...)`
- 重要参数详解：
 - `file`: 文件路径，可以是本地文件或网络 URL
 - `header = TRUE`: 第一行是否为列名，默认 TRUE
 - `sep = ","`: 字段分隔符，CSV 默认逗号
 - `stringsAsFactors = FALSE`: 是否将字符串转换为因子，建议设为 FALSE
 - `encoding`: 文件编码，中文文件可能需要设置为“UTF-8”
- 文件路径注意事项：
 - Windows 系统使用反斜杠\，但 R 中需要转义\\或使用正斜杠/
 - 使用相对路径时，基于当前工作目录
 - 可用 `getwd()` 查看当前工作目录，`setwd()` 设置工作目录

1.6.4.2 数据概览函数集

1.6.4.2.1 `summary()` - 数据摘要

- 作用：提供每列数据的统计摘要
- 不同数据类型的摘要：
 - 数值型：最小值、第一四分位数、中位数、均值、第三四分位数、最大值
 - 字符型：长度、类别、模式
 - 因子型：各水平的频数
- 应用价值：快速了解数据分布、发现异常值

1.6.4.2.2 `unique()` - 唯一值

- 作用：返回向量中的所有不重复值
- 语法：`unique(x)`
- 应用场景：
 - 查看分类变量的所有类别
 - 检查数据录入是否有错误（如拼写错误）
 - 了解数据的多样性
- 相关函数：`duplicated()` 检查重复值

1.6.4.2.3 `range()` - 值域范围

- 作用: 返回向量的最小值和最大值
- 语法: `range(x, na.rm = FALSE)`
- 返回值: 长度为 2 的向量, `c(最小值, 最大值)`
- 应用: 快速了解数据的取值范围

1.6.4.3 数据筛选系统

1.6.4.3.1 `subset()` 函数 (推荐方式)

- 作用: 根据条件筛选数据框的行
- 语法: `subset(x, subset, select)`
- 参数详解:
 - `x`: 要筛选的数据框
 - `subset`: 逻辑条件表达式
 - `select`: 选择的列 (可选)
- 优势: 语法简洁, 不需要重复写数据框名称

1.6.4.3.2 逻辑操作符详解

1.6.4.3.2.1 `%in%` 操作符

- 作用: 检查左边的值是否在右边的向量中
- 语法: `x %in% y`
- 示例: `month %in% c(3, 4, 5)` 检查月份是否是 3、4、5 中的一个
- 与 `==` 的区别:
 - `==` 只能比较单个值
 - `%in%` 可以同时比较多个值

1.6.4.3.2.2 `==` 等于操作符

- 作用: 检查两个值是否相等
- 注意事项:
 - 区分大小写: `"A" == "a"` 为 FALSE
 - 精确匹配: `"cat" == "cats"` 为 FALSE
 - 用于字符串时必须完全匹配

1.6.4.3.2.3 `&` 逻辑与操作符

- 作用: 连接多个条件, 所有条件都必须为 TRUE
- 语法: 条件 1 & 条件 2 & ...
- 相关操作符:
 - `|`: 逻辑或, 任一条件为 TRUE 即可
 - `!`: 逻辑非, 取反

1.6.4.4 4. 数值处理函数

1.6.4.4.1 `round()` - 四舍五入

- 语法: `round(x, digits = 0)`
- 参数:
 - `x`: 要舍入的数值
 - `digits`: 保留的小数位数
- 应用: 美化输出结果, 控制精度

1.6.4.5 数据导入最佳实践

- **文件检查:** 导入前先用文本编辑器查看文件格式
- **编码处理:** 中文数据注意编码问题
- **数据验证:** 导入后立即检查数据结构和内容
- **备份原始数据:** 避免在原始数据上直接修改
- **路径管理:** 使用项目文件夹, 保持文件路径的一致性

1.6.5 课后练习

题目: 假设有一个植被监测数据文件包含以下列:

- date: 调查日期
- season: 季节 (春、夏、秋、冬)
- plot: 样地编号
- coverage: 植被覆盖度 (%)
- height: 平均高度 (cm)

请完成 (使用数据导入、数据框操作、条件筛选等已学内容):

1. 创建模拟数据或读取数据文件

2. 查看数据的基本信息 (使用 nrow(), ncol(), str(), summary())
3. 筛选夏季的数据 (使用 subset() 函数)
4. 筛选植被覆盖度大于 80% 的记录 (使用 subset() 函数和条件)
5. 计算所有样地的平均植被覆盖度和平均高度 (使用 mean() 函数)
6. 找出覆盖度最高的样地 (使用 which.max() 函数)
7. 创建一个汇总报告 (使用 paste() 函数)

1.7 缺失值和异常值的识别处理

1.7.1 生态学背景

在水质监测中, 由于仪器故障、人为记录错误等原因, 经常出现缺失值和异常值。数据清理是生态学数据分析的重要步骤, 需要识别和处理这些问题数据。

1.7.2 演示数据

```
# 湖泊水质监测数据 (包含缺失值和异常值)
water_quality <- data.frame(
  site_id = c("湖心", "入水口", "出水口", "湖心", "入水口", "出水口"),
  date = c("2023-05-01", "2023-05-01", "2023-05-01", "2023-05-15", "2023-05-15", "2023-05-15"),
  pH = c(7.2, 6.8, NA, 7.5, 6.9, 7.1),
  temperature_C = c(18.5, 19.2, 20.1, 999, 19.8, 20.3), # 999 为仪器错误读数
  dissolved_oxygen = c(8.2, 7.5, 8.8, 8.1, NA, 8.4)
)
```

1.7.3 课堂演示过程

```
# 1. 创建包含问题的水质数据
water_quality <- data.frame(
  site_id = c("湖心", "入水口", "出水口", "湖心", "入水口", "出水口"),
  date = c("2023-05-01", "2023-05-01", "2023-05-01", "2023-05-15", "2023-05-15", "2023-05-15"),
  pH = c(7.2, 6.8, NA, 7.5, 6.9, 7.1),
  temperature_C = c(18.5, 19.2, 20.1, 999, 19.8, 20.3),
```

```

dissolved_oxygen = c(8.2, 7.5, 8.8, 8.1, NA, 8.4)
)

# 2. 查看原始数据
print(water_quality)
str(water_quality)

# 3. 检查缺失值
is.na(water_quality) # 显示所有缺失值位置
sum(is.na(water_quality$pH)) # pH 缺失值个数
sum(is.na(water_quality$dissolved_oxygen)) # 溶解氧缺失值个数

# 4. 识别异常值
summary(water_quality$temperature_C) # 查看温度的统计摘要
water_quality$temperature_C > 50 # 找出不合理的高温值

# 5. 处理缺失值 - 用均值填补
ph_mean <- mean(water_quality$pH, na.rm = TRUE) # 计算 pH 均值 (忽略 NA)
water_quality$pH[is.na(water_quality$pH)] <- ph_mean

# 6. 处理异常值 - 替换为 NA
water_quality$temperature_C[water_quality$temperature_C > 50] <- NA

# 7. 查看清理后的数据
print(water_quality)

# 8. 删除包含 NA 的整行 (如果需要)
clean_data <- na.omit(water_quality)
print(clean_data)

# 9. 计算清理后的统计量
mean(clean_data$temperature_C)
mean(clean_data$pH)
mean(clean_data$dissolved_oxygen)

```

1.7.4 R 语言知识点详解

1.7.4.1 缺失值 (Missing Values) 处理系统

1.7.4.1.1 缺失值的概念

- 什么是 NA: Not Available 的缩写, 表示缺失或不可用的数据
- NA 的特点:
 - 任何包含 NA 的运算结果都是 NA
 - NA 具有传染性: $1 + \text{NA} = \text{NA}$
 - NA 不等于任何值, 包括它自己: $\text{NA} == \text{NA}$ 返回 NA 而不是 TRUE

1.7.4.1.2 is.na() 函数

- 作用: 检测缺失值的位置
- 语法: `is.na(x)`
- 返回值: 与输入同样结构的逻辑向量/矩阵, TRUE 表示缺失
- 应用方式:
 - 检查单个向量: `is.na(vector)`
 - 检查整个数据框: `is.na(data.frame)`

— 统计缺失值数量: `sum(is.na(vector))`

1.7.4.1.3 `na.rm` 参数

- 作用: 在统计计算中移除缺失值
- 语法: `function(x, na.rm = FALSE)`
- 适用函数: `mean()`, `sum()`, `max()`, `min()`, `sd()` 等
- 重要性: 不设置 `na.rm = TRUE` 时, 有 NA 的计算结果都是 NA
- 示例对比:

```
x <- c(1, 2, NA, 4)
mean(x)          # 返回 NA
mean(x, na.rm = TRUE) # 返回 2.33
```

1.7.4.1.4 `na.omit()` 函数

- 作用: 删除包含任何缺失值的完整行
- 语法: `na.omit(x)`
- 返回值: 不含任何 NA 的数据框
- 注意事项:
 - 可能导致大量数据丢失
 - 需要评估删除行对分析的影响
 - 适合缺失值较少且随机分布的情况

1.7.4.2 异常值 (Outliers) 识别与处理

1.7.4.2.1 异常值的识别方法

- 统计方法: 使用 `summary()` 查看数据分布, 识别明显不合理的值
- 业务逻辑: 基于专业知识判断, 如温度 999°C 明显错误
- 可视化方法: 使用箱线图、散点图等发现异常值
- 统计阈值: 如超出 3 倍标准差的值

1.7.4.2.2 异常值处理策略

1. **删除异常值**: 适用于明显的录入错误
2. **替换为 NA**: 保留数据结构, 标记为缺失
3. **替换为合理值**: 用中位数、均值等替换
4. **保留但标记**: 在分析中特殊处理

1.7.4.3 条件替换技术

1.7.4.3.1 逻辑索引替换

- 语法: `data[condition] <- new_value`
- 原理: 通过逻辑条件选择满足条件的元素进行替换
- 示例:

```
# 将所有负值替换为 0
data[data < 0] <- 0

# 将异常高值替换为 NA
data[data > threshold] <- NA
```

1.7.4.3.2 which() 函数

- 作用：返回满足条件的元素位置索引
- 语法：`which(condition)`
- 与直接逻辑索引的区别：
 - 逻辑索引：返回 TRUE/FALSE 向量
 - `which()`：返回位置数字向量
- 应用：当需要知道具体位置时使用

1.7.4.4 数据清理流程和最佳实践

1.7.4.4.1 标准数据清理流程

1. 数据探索：使用 `str()`, `summary()`, `head()`, `tail()` 了解数据
2. 缺失值检查：使用 `is.na()`, `sum(is.na())` 统计缺失情况
3. 异常值识别：结合统计和专业知识识别异常值
4. 清理决策：选择合适的处理方法
5. 执行清理：应用处理方法
6. 验证结果：检查清理后的数据质量

1.7.4.4.2 数据清理的注意事项

- 保留原始数据：清理前备份原始数据
- 记录清理过程：文档化所有清理步骤和决策理由
- 验证合理性：确保清理后的数据符合业务逻辑
- 评估影响：分析清理对后续分析的影响

1.7.4.4.3 缺失值填补方法选择

- 均值填补：适用于数值变量，数据接近正态分布
- 中位数填补：适用于有偏斜的数值变量
- 众数填补：适用于分类变量
- 前向/后向填补：适用于时间序列数据
- 预测模型填补：基于其他变量预测缺失值

1.7.5 课后练习

题目：某森林土壤调查数据：

```
soil_data <- data.frame(
  plot = c("A1", "A2", "A3", "B1", "B2", "B3"),
  organic_matter = c(3.2, NA, 2.8, 3.5, 2.9, 3.1),
  nitrogen_mg = c(45, 52, -10, 48, 51, 49), # -10 为异常负值
  moisture = c(25.5, 28.2, 22.1, NA, 26.8, 24.9)
)
```

- 请完成（使用缺失值处理、条件判断、数据清理等已学内容）：
1. 检查每列的缺失值个数（使用 `is.na()` 和 `sum()` 函数）
 2. 识别 `nitrogen_mg` 列中的异常值（使用逻辑判断和 `which()` 函数）
 3. 用均值填补 `organic_matter` 的缺失值（使用 `mean()` 和 `na.rm` 参数）
 4. 将 `nitrogen_mg` 中的异常值替换为 NA（使用条件赋值）
 5. 创建一个完全没有缺失值的干净数据集（使用 `na.omit()`）
 6. 计算清理后数据的各项平均值（使用 `mean()` 函数）

7. 对比清理前后数据的 summary() 结果

1.8 描述性统计分析与基础数据可视化

1.8.1 生态学背景

不同森林类型的物种多样性存在显著差异。通过比较松林、栎林、混交林的物种数量，我们可以了解森林结构对生物多样性的影响。这需要用到描述性统计和基础可视化。

1.8.2 演示数据

```
# 三种森林类型各 5 个样地的物种数量
pine_forest <- c(22, 25, 20, 28, 24)      # 松林
oak_forest <- c(35, 32, 38, 30, 34)        # 栎林
mixed_forest <- c(45, 42, 48, 40, 46)       # 混交林
```

1.8.3 课堂演示过程

```
# 1. 创建三种森林类型数据
pine_forest <- c(22, 25, 20, 28, 24)
oak_forest <- c(35, 32, 38, 30, 34)
mixed_forest <- c(45, 42, 48, 40, 46)

# 2. 计算描述性统计
# 平均值
pine_mean <- mean(pine_forest)
oak_mean <- mean(oak_forest)
mixed_mean <- mean(mixed_forest)

# 标准差
pine_sd <- sd(pine_forest)
oak_sd <- sd(oak_forest)
mixed_sd <- sd(mixed_forest)

# 最大值和最小值
range(pine_forest)
range(oak_forest)
range(mixed_forest)

# 3. 创建汇总表
forest_summary <- data.frame(
  森林类型 = c("松林", "栎林", "混交林"),
  平均物种数 = c(pine_mean, oak_mean, mixed_mean),
  标准差 = c(pine_sd, oak_sd, mixed_sd),
  最大值 = c(max(pine_forest), max(oak_forest), max(mixed_forest)),
  最小值 = c(min(pine_forest), min(oak_forest), min(mixed_forest))
)
print(forest_summary)

# 4. 箱线图比较
boxplot(pine_forest, oak_forest, mixed_forest,
         names = c("松林", "栎林", "混交林"),
         ylab = "物种数量",
```

```

main = " 不同森林类型物种多样性比较",
col = c("lightgreen", "lightblue", "lightyellow"))

# 5. 添加平均值点
points(1:3, c(pine_mean, oak_mean, mixed_mean),
       col = "red", pch = 19, cex = 1.5)

# 6. 条形图显示平均值
barplot(c(pine_mean, oak_mean, mixed_mean),
         names.arg = c(" 松林", " 榆林", " 混交林"),
         ylab = " 平均物种数",
         main = " 各森林类型平均物种数量",
         col = c("lightgreen", "lightblue", "lightyellow"))

# 7. 方差分析 (简单介绍)
all_data <- c(pine_forest, oak_forest, mixed_forest)
forest_type <- rep(c(" 松林", " 榆林", " 混交林"), each = 5)
forest_df <- data.frame(species_count = all_data, type = forest_type)

```

1.8.4 R 语言知识点详解

1.8.4.1 描述性统计函数深入解析

1.8.4.1.1 sd() - 标准差函数

- 作用：计算样本标准差，衡量数据的离散程度
- 语法：`sd(x, na.rm = FALSE)`
- 数学含义：
 - 标准差越大，数据越分散
 - 标准差越小，数据越集中在均值附近
 - 单位与原数据相同
- 与方差的关系：标准差 = $\sqrt{\text{方差}}$
- 相关函数：
 - `var()`：计算方差
 - `mad()`：计算中位数绝对偏差（对异常值更稳健）

1.8.4.1.2 range() - 值域函数

- 作用：返回最小值和最大值组成的向量
- 语法：`range(x, na.rm = FALSE)`
- 返回值：长度为 2 的数值向量 `c(min, max)`
- 应用：
 - 快速了解数据的取值范围
 - 检查数据是否在合理范围内
 - 设置图形的坐标轴范围

1.8.4.2 R 基础绘图系统详解

1.8.4.2.1 boxplot() - 箱线图函数

- 作用：绘制箱线图，显示数据的分布特征
- 语法：`boxplot(..., names, main, xlab, ylab, col)`
- 箱线图解读：
 - 盒子：第一四分位数 (Q1) 到第三四分位数 (Q3)，包含 50% 的数据
 - 中线：中位数 (Q2)

- 颖线: 延伸到 1.5 倍四分位数间距的范围
- 点: 超出颖线的异常值
- 参数详解:
 - `names`: 各组的标签
 - `main`: 图形标题
 - `xlab, ylab`: x 轴和 y 轴标签
 - `col`: 填充颜色
 - `border`: 边框颜色
 - `notch`: 是否显示置信区间缺口

1.8.4.2.2 `barplot()` - 条形图函数

- 作用: 绘制条形图, 比较不同组的数值
- 语法: `barplot(height, names.arg, main, xlab, ylab, col)`
- 参数详解:
 - `height`: 条形的高度值
 - `names.arg`: 条形的标签
 - `beside`: 并排显示多组数据时设为 TRUE
 - `horiz`: 是否绘制水平条形图
- 应用场景:
 - 比较不同组的均值
 - 显示分类数据的频数
 - 展示比例或百分比

1.8.4.3 图形参数和美化

1.8.4.3.1 颜色参数 `col`

- 预定义颜色: "red", "blue", "green" 等
- 颜色名称: `colors()` 查看所有可用颜色名称
- 十六进制: "#FF0000" (红色)
- RGB 函数: `rgb(1, 0, 0)` (红色)
- 颜色向量: 为不同元素指定不同颜色

1.8.4.3.2 点的形状参数 `pch`

- 常用形状:
 - `pch = 1`: 空心圆
 - `pch = 19`: 实心圆
 - `pch = 2`: 空心三角形
 - `pch = 17`: 实心三角形
 - `pch = 15`: 实心方形
- 字符形状: `pch = "A"` 使用字符 A 作为点

1.8.4.3.3 大小参数 `cex`

- 作用: 控制图形元素的大小
- 默认值: 1.0
- 用法:
 - `cex = 1.5`: 放大 1.5 倍
 - `cex = 0.8`: 缩小为 0.8 倍
- 相关参数:
 - `cex.main`: 标题大小
 - `cex.lab`: 轴标签大小
 - `cex.axis`: 轴数字大小

1.8.4.4 图形叠加和增强

1.8.4.4.1 points() - 添加点

- 作用: 在现有图形上添加点
- 语法: `points(x, y, col, pch, cex)`
- 坐标系统: 使用与原图相同的坐标系统
- 应用: 在箱线图上标记均值、在散点图上突出特定点

1.8.4.4.2 图形叠加的原理

- 图层概念: R 绘图采用图层叠加的方式
- 顺序重要: 后绘制的元素会覆盖先绘制的元素
- 坐标统一: 所有叠加元素必须使用相同的坐标系统

1.8.4.5 数据重组和整理

1.8.4.5.1 rep() - 重复函数

- 作用: 重复向量元素
- 语法: `rep(x, times, each, length.out)`
- 参数说明:
 - `times`: 整个向量重复的次数
 - `each`: 每个元素重复的次数
 - `length.out`: 输出向量的长度
- 示例:

```
rep(c("A", "B"), times = 2)      # "A" "B" "A" "B"  
rep(c("A", "B"), each = 2)       # "A" "A" "B" "B"  
rep(c("A", "B"), length.out = 5) # "A" "B" "A" "B" "A"
```

1.8.4.6 图形设计最佳实践

1.8.4.6.1 色彩选择原则

- 对比度: 确保不同组别容易区分
- 色盲友好: 避免仅依赖红绿色区分
- 一致性: 同一类型数据使用相同色系
- 专业性: 避免过于鲜艳的颜色

1.8.4.6.2 标签和标题

- 信息完整: 包含变量名称和单位
- 简洁明了: 避免过长的标题
- 中文支持: 确保中文字符正确显示

1.8.4.6.3 图形尺寸和比例

- 合适的比例: 避免图形过于压缩或拉伸
- 合理的尺寸: 适合展示媒介的大小
- 留白空间: 给图形元素足够的空间

1.8.5 课后练习

题目: 某保护区三种植被类型的蝴蝶物种数调查:

```
shrubland <- c(12, 15, 10, 14, 13)      # 灌丛
meadow <- c(18, 22, 20, 19, 21)        # 草甸
riparian <- c(25, 28, 23, 27, 26)       # 河岸林
```

请完成（使用描述统计、基础绘图等已学内容）：

1. 计算三种植被类型的平均物种数和标准差（使用 mean() 和 sd() 函数）
2. 创建一个汇总表显示基本统计信息（使用 data.frame()）
3. 绘制箱线图比较三种植被类型（使用 boxplot() 函数）
4. 绘制条形图显示平均物种数（使用 barplot() 函数）
5. 在箱线图上添加平均值点（使用 points() 函数）
6. 判断哪种植被类型的蝴蝶多样性最高（使用 which.max() 和 max() 函数）
7. 计算每种植被类型的变异系数（标准差/平均值 ×100）

1.9 条件判断、循环结构与函数编程

1.9.1 生态学背景

在群落生态学研究中，经常需要根据不同条件对物种进行分类处理，或者对大量样地数据进行批量处理。这需要用到编程中的条件判断和循环结构，让 R 能够自动化完成重复性工作。

1.9.2 演示数据

```
# 某自然保护区不同海拔的物种调查数据
sites_data <- data.frame(
  site_id = paste0("S", 1:10),
  elevation = c(1200, 1450, 1800, 2100, 2350, 1650, 1900, 2200, 1750, 2050),
  species_count = c(45, 52, 38, 28, 22, 48, 35, 25, 42, 30),
  dominant_species = c("栎树", "栎树", "云杉", "冷杉", "高山杜鹃", "栎树", "云杉", "冷杉", "云杉", "冷杉")
)
```

1.9.3 课堂演示过程

1.9.3.1 条件判断基础

```
# 创建示例数据
sites_data <- data.frame(
  site_id = paste0("S", 1:10),
  elevation = c(1200, 1450, 1800, 2100, 2350, 1650, 1900, 2200, 1750, 2050),
  species_count = c(45, 52, 38, 28, 22, 48, 35, 25, 42, 30),
  dominant_species = c("栎树", "栎树", "云杉", "冷杉", "高山杜鹃", "栎树", "云杉", "冷杉", "云杉", "冷杉"
)

# 简单的 if 语句
elevation_threshold <- 2000
if (sites_data$elevation[1] > elevation_threshold) {
  print("高海拔样地")
} else {
  print("低海拔样地")
}
```

```
# if-else 判断所有样地
for (i in 1:nrow(sites_data)) {
  if (sites_data$elevation[i] > 2000) {
    print(paste(sites_data$site_id[i], "是高海拔样地"))
  } else {
    print(paste(sites_data$site_id[i], "是低海拔样地"))
  }
}
```

1.9.3.2 向量化条件判断

```
# 使用 ifelse() 函数进行向量化判断
sites_data$elevation_zone <- ifelse(sites_data$elevation > 2000, "高海拔", "低海拔")
print(sites_data[, c("site_id", "elevation", "elevation_zone")])

# 多层条件判断
sites_data$vegetation_type <- ifelse(sites_data$elevation < 1500, "阔叶林",
                                         ifelse(sites_data$elevation < 2000, "混交林", "针叶林"))
print(sites_data[, c("site_id", "elevation", "vegetation_type")])
```

1.9.3.3 for 循环处理

```
# 计算每个样地的多样性指数类别
diversity_categories <- character(nrow(sites_data))

for (i in 1:nrow(sites_data)) {
  species_num <- sites_data$species_count[i]
  if (species_num >= 40) {
    diversity_categories[i] <- "高多样性"
  } else if (species_num >= 30) {
    diversity_categories[i] <- "中等多样性"
  } else {
    diversity_categories[i] <- "低多样性"
  }
}

sites_data$diversity_category <- diversity_categories
print(sites_data[, c("site_id", "species_count", "diversity_category")])
```

1.9.3.4 自定义函数编写

```
# 编写海拔带判断函数
classify_elevation_zone <- function(elevation) {
  if (elevation < 1500) {
    return("低山带")
  } else if (elevation < 2000) {
    return("中山带")
  } else {
    return("高山带")
  }
}
```

```
# 测试函数
classify_elevation_zone(1800)
classify_elevation_zone(2200)

# 批量应用函数
sites_data$elevation_belt <- sapply(sites_data$elevation, classify_elevation_zone)
print(sites_data[, c("site_id", "elevation", "elevation_belt")])
```

1.9.3.5 复杂条件处理

```
# 编写综合评估函数
assess_conservation_value <- function(elevation, species_count, dominant_sp) {
  score <- 0

  # 海拔因子
  if (elevation > 2000) {
    score <- score + 2
  } else if (elevation > 1500) {
    score <- score + 1
  }

  # 物种多样性因子
  if (species_count > 40) {
    score <- score + 2
  } else if (species_count > 30) {
    score <- score + 1
  }

  # 优势种稀有性因子
  rare_species <- c("高山杜鹃", "冷杉")
  if (dominant_sp %in% rare_species) {
    score <- score + 1
  }

  # 返回保护价值等级
  if (score >= 4) {
    return("极高价值")
  } else if (score >= 3) {
    return("高价值")
  } else if (score >= 2) {
    return("中等价值")
  } else {
    return("一般价值")
  }
}

# 应用综合评估
sites_data$conservation_value <- mapply(assess_conservation_value,
                                          sites_data$elevation,
                                          sites_data$species_count,
                                          sites_data$dominant_species)

print(sites_data[, c("site_id", "conservation_value")])
```

1.9.4 R 语言知识点详解

1.9.4.1 条件判断结构

1.9.4.1.1 if 语句

- 语法: `if (条件) { 执行代码 }`
- 条件: 必须是逻辑值 (TRUE/FALSE)
- 执行规则: 条件为 TRUE 时执行大括号内的代码
- 注意事项: 条件必须是长度为 1 的逻辑向量

1.9.4.1.2 if-else 语句

- 语法:

```
if (条件) {
  # 条件为 TRUE 时执行
} else {
  # 条件为 FALSE 时执行
}
```

- 多重条件: `else if` 可以链式连接
- 最佳实践: 始终使用大括号, 即使只有一行代码

1.9.4.1.3 ifelse() 函数 (向量化)

- 语法: `ifelse(test, yes, no)`
- 优势: 可以处理向量, 一次性判断多个元素
- 参数:
 - `test`: 逻辑向量条件
 - `yes`: 条件为 TRUE 时返回的值
 - `no`: 条件为 FALSE 时返回的值
- 嵌套使用: 可以嵌套实现多重条件判断

1.9.4.2 循环结构

1.9.4.2.1 for 循环

- 语法: `for (变量 in 序列) { 循环体 }`
- 常见用法:

```
# 按索引循环
for (i in 1:10) { }

# 按元素循环
for (item in vector) { }

# 按名称循环
for (name in names(list)) { }
```

- 循环控制:
 - `break`: 跳出循环
 - `next`: 跳过当前迭代

1.9.4.2.2 其他循环类型

- `while` 循环: `while (条件) { 循环体 }`

- `repeat` 循环: `repeat { 循环体; if(条件) break }`

1.9.4.3 函数定义

1.9.4.3.1 基本函数语法

- 语法:

```
函数名 <- function(参数 1, 参数 2 = 默认值) {
  # 函数体
  return(返回值)
}
```

- 参数:

- 必需参数: 调用时必须提供
- 可选参数: 有默认值, 可省略

- 返回值:

- 显式返回: 使用 `return()`
- 隐式返回: 函数最后一个表达式的值

1.9.4.3.2 函数设计原则

- 单一职责: 一个函数只做一件事
- 参数验证: 检查输入参数的有效性
- 错误处理: 使用 `stop()`、`warning()` 处理异常
- 文档化: 添加注释说明函数用途和参数

1.9.4.4 高级应用函数

1.9.4.4.1 sapply() - 简化的 apply

- 作用: 对向量或列表的每个元素应用函数
- 语法: `sapply(X, FUN, ...)`
- 返回值: 简化后的向量或矩阵
- 与 `lapply()` 的区别:
 - `lapply()` 总是返回列表
 - `sapply()` 尝试简化结果

1.9.4.4.2 mapply() - 多变量 apply

- 作用: 同时对多个向量应用函数
- 语法: `mapply(FUN, ..., MoreArgs = NULL)`
- 应用场景: 函数需要多个参数时使用
- 示例: `mapply(function(x, y) x + y, vector1, vector2)`

1.9.4.5 逻辑运算符

1.9.4.5.1 基本逻辑运算符

- `==`: 等于
- `!=`: 不等于
- `>`、`<`、`>=`、`<=`**: 比较运算符
- `&`: 与 (向量化)
- `|`: 或 (向量化)
- `!`: 非
- `&&`、`||`**: 短路逻辑运算符 (只判断第一个元素)

1.9.4.5.2 成员测试

- `%in%`: 检查元素是否在向量中
- `is.na()`: 检查缺失值
- `is.null()`: 检查空值

1.9.4.6 编程最佳实践

1.9.4.6.1 代码组织

- **缩进**: 使用一致的缩进 (建议 2 或 4 个空格)
- **命名**: 使用有意义的变量名和函数名
- **注释**: 解释复杂逻辑和算法思路
- **模块化**: 将复杂任务分解为简单函数

1.9.4.6.2 性能考虑

- **向量化**: 优先使用向量化操作而非循环
- **预分配**: 循环前预分配存储空间
- **避免增长**: 不要在循环中动态增长向量

1.9.4.6.3 调试技巧

- `print()`: 在关键位置输出变量值
- `browser()`: 设置断点进行交互式调试
- `traceback()`: 查看错误调用堆栈
- **分步测试**: 逐步测试函数的各个部分

1.9.5 课后练习

题目：某湿地鸟类监测数据包含以下信息：

```
bird_monitoring <- data.frame(
  site = c("A1", "A2", "B1", "B2", "C1", "C2"),
  water_depth = c(15, 25, 45, 35, 65, 55), # 水深 (cm)
  bird_abundance = c(8, 12, 20, 16, 5, 8), # 鸟类丰度
  season = c("春季", "春季", "夏季", "夏季", "秋季", "秋季")
)
```

请完成（使用 if-else、循环、函数等编程内容，结合之前学过的数据处理方法）：

1. 使用 `ifelse()` 函数，根据水深将栖息地分类 ($<30\text{cm}$ 浅水区, $30\text{-}50\text{cm}$ 中等深度, $>50\text{cm}$ 深水区)
2. 编写函数 `classify_habitat_quality()`, 综合水深和鸟类丰度评估栖息地质量
3. 使用 `for` 循环, 计算每个季节的平均鸟类丰度
4. 创建一个新列, 标记高丰度样地 (丰度 >15 为高丰度, 使用 `ifelse()`)
5. 编写函数处理整个数据集, 输出每个样地的综合评估报告
6. 使用 `apply` 族函数重做第 3 题 (比较循环和向量化方法的差异)

1.10 现代数据科学工具包应用

1.10.1 生态学背景

现代生态学研究产生的数据日益复杂，传统的基础 R 语法在处理复杂数据操作时略显繁琐。tidyverse 是 R 语言的现代数据处理工具包，提供了更直观、更高效的数据处理方法，特别适合处理多变量、多时间点的生态学数据。

1.10.2 演示数据

```
# 模拟某森林样地多年监测数据
library(tidyverse)

# 创建模拟数据
forest_monitoring <- data.frame(
  plot_id = rep(paste0("Plot_", 1:5), each = 12),
  year = rep(2018:2021, times = 15),
  season = rep(c("春", "夏", "秋"), times = 20),
  temperature = rnorm(60, mean = 15, sd = 5),
  humidity = rnorm(60, mean = 70, sd = 10),
  species_richness = rpois(60, lambda = 25),
  tree_height = rnorm(60, mean = 12, sd = 3),
  soil_ph = rnorm(60, mean = 6.5, sd = 0.5)
)
```

1.10.3 课堂演示过程

1.10.3.1 tidyverse 包的加载和数据查看

```
# 安装和加载 tidyverse
# install.packages("tidyverse")
library(tidyverse)

# 创建示例数据（简化版）
forest_data <- tibble(
  plot_id = rep(c("A", "B", "C", "D"), each = 6),
  year = rep(2020:2022, times = 8),
  season = rep(c("春", "夏"), times = 12),
  temperature = c(12, 18, 15, 22, 10, 16, 14, 20, 13, 19, 11, 17,
                 16, 21, 14, 20, 12, 18, 15, 23, 13, 21, 11, 19),
  species_count = c(22, 28, 25, 32, 20, 24, 26, 30, 24, 28, 21, 25,
                   28, 34, 26, 32, 23, 27, 30, 36, 27, 31, 24, 28)
)

# 查看数据结构
glimpse(forest_data)
head(forest_data)
```

1.10.3.2 数据筛选与选择

```
# 使用 filter() 筛选行
summer_data <- forest_data %>%
  filter(season == "夏")

high_diversity <- forest_data %>%
  filter(species_count > 30)

recent_summer <- forest_data %>%
  filter(year >= 2021 & season == "夏")

# 使用 select() 选择列
```

```

temp_species <- forest_data %>%
  select(plot_id, temperature, species_count)

# 选择特定范围的列
core_variables <- forest_data %>%
  select(plot_id:season, species_count)

# 排除特定列
without_year <- forest_data %>%
  select(-year)

```

1.10.3.3 数据变换与新变量创建

```

# 使用 mutate() 创建新变量
forest_enhanced <- forest_data %>%
  mutate(
    temp_category = case_when(
      temperature < 15 ~ "低温",
      temperature < 20 ~ "中温",
      TRUE ~ "高温"
    ),
    diversity_index = species_count / 10, # 简化的多样性指数
    temp_celsius = temperature,
    temp_fahrenheit = temperature * 9/5 + 32
  )

# 查看结果
forest_enhanced %>%
  select(plot_id, temperature, temp_category, species_count, diversity_index)

```

1.10.3.4 数据排序与分组汇总

```

# 使用 arrange() 排序
forest_data %>%
  arrange(desc(species_count)) %>%
  head()

forest_data %>%
  arrange(plot_id, year, season)

# 使用 group_by() 和 summarise() 进行分组统计
plot_summary <- forest_data %>%
  group_by(plot_id) %>%
  summarise(
    n_observations = n(),
    mean_temperature = mean(temperature),
    mean_species = mean(species_count),
    max_species = max(species_count),
    sd_temperature = sd(temperature),
    .groups = 'drop'
  )

print(plot_summary)

```

```
# 多变量分组
season_plot_summary <- forest_data %>%
  group_by(season, plot_id) %>%
  summarise(
    mean_temp = mean(temperature),
    mean_species = mean(species_count),
    .groups = 'drop'
  )

print(season_plot_summary)
```

1.10.3.5 数据重塑：长宽格式转换

```
# 宽格式转长格式 (gather/pivot_longer)
forest_long <- forest_data %>%
  pivot_longer(
    cols = c(temperature, species_count),
    names_to = "variable",
    values_to = "value"
  )

head(forest_long)

# 长格式转宽格式 (spread/pivot_wider)
forest_wide <- forest_long %>%
  pivot_wider(
    names_from = variable,
    values_from = value
  )

head(forest_wide)
```

1.10.3.6 数据连接

```
# 创建额外的样地信息
plot_info <- tibble(
  plot_id = c("A", "B", "C", "D"),
  elevation = c(1200, 1450, 1800, 1600),
  soil_type = c(" 壤土", " 砂土", " 黏土", " 壤土"),
  management = c(" 保护", " 管理", " 保护", " 管理")
)

# 左连接
forest_complete <- forest_data %>%
  left_join(plot_info, by = "plot_id")

head(forest_complete)

# 按管理类型分析
management_analysis <- forest_complete %>%
  group_by(management) %>%
  summarise(
```

```

mean_temperature = mean(temperature),
mean_species = mean(species_count),
.groups = 'drop'
)

print(management_analysis)

```

1.10.4 R 语言知识点详解

1.10.4.1 tidyverse 哲学与管道操作

1.10.4.1.1 管道操作符 %>%

- 作用: 将左侧结果作为右侧函数的第一个参数
- 优势:
 - 代码更易读: 从左到右, 从上到下
 - 减少中间变量: 避免创建临时对象
 - 链式操作: 多个操作连续进行
- 语法: data %>% function()
- 等价写法: function(data)

1.10.4.1.2 tibble vs data.frame

- tibble 特点:
 - 更好的打印输出
 - 更严格的子集操作
 - 保持字符串为字符串 (不自动转因子)
 - 支持列名包含空格和特殊字符

1.10.4.2 数据筛选与选择

1.10.4.2.1 filter() - 行筛选

- 语法: filter(data, condition1, condition2, ...)
- 多条件:
 - 逗号分隔: 逻辑与 (AND)
 - |: 逻辑或 (OR)
 - !: 逻辑非 (NOT)
- 常用条件:
 - ==、 !=: 等于、不等于
 - >、<、>=、<=: 大小比较
 - %in%: 成员检查
 - is.na(): 缺失值检查

1.10.4.2.2 select() - 列选择

- 基本选择: select(data, col1, col2)
- 范围选择: select(data, col1:col3)
- 排除选择: select(data, -col1, -col2)
- 辅助函数:
 - starts_with(): 以某字符开头
 - ends_with(): 以某字符结尾
 - contains(): 包含某字符
 - matches(): 正则表达式匹配

1.10.4.3 数据变换

1.10.4.3.1 mutate() - 新变量创建

- 基本用法: `mutate(data, new_col = expression)`
- 多变量: 可同时创建多个新变量
- 引用新建变量: 在同一个 `mutate()` 中可引用前面创建的变量
- 变量类型转换:
 - `as.numeric()`: 转数值
 - `as.character()`: 转字符
 - `as.factor()`: 转因子

1.10.4.3.2 case_when() - 多条件分类

- 语法:

```
case_when(
  condition1 ~ value1,
  condition2 ~ value2,
  TRUE ~ default_value
)
```

- 优势: 替代复杂的嵌套 `ifelse()`
- 注意: 条件从上到下评估, 满足即停止

1.10.4.4 数据排序与汇总

1.10.4.4.1 arrange() - 数据排序

- 基本排序: `arrange(data, col)`
- 降序排序: `arrange(data, desc(col))`
- 多列排序: `arrange(data, col1, col2)`

1.10.4.4.2 group_by() 与 summarise()

- 分组概念: 将数据按指定变量分组
- 汇总函数:
 - `n()`: 计数
 - `mean()`、`median()`: 均值、中位数
 - `sum()`、`min()`、`max()`: 求和、最小值、最大值
 - `sd()`、`var()`: 标准差、方差
- `.groups` 参数: 控制结果的分组状态

1.10.4.5 数据重塑

1.10.4.5.1 长宽格式概念

- 宽格式: 每个变量一列, 观察单位一行
- 长格式: 变量名和变量值分别存储在不同列中
- 选择原则:
 - 分析时通常用长格式
 - 展示时通常用宽格式

1.10.4.5.2 pivot_longer() - 宽转长

- 语法: `pivot_longer(data, cols, names_to, values_to)`
- 参数:
 - `cols`: 要转换的列

- `names_to`: 存储变量名的新列名
- `values_to`: 存储变量值的新列名

1.10.4.5.3 `pivot_wider()` - 长转宽

- 语法: `pivot_wider(data, names_from, values_from)`
- 参数:
 - `names_from`: 提供新列名的列
 - `values_from`: 提供新列值的列

1.10.4.6 数据连接

1.10.4.6.1 连接类型

- `left_join()`: 保留左表所有行
- `right_join()`: 保留右表所有行
- `inner_join()`: 仅保留匹配行
- `full_join()`: 保留所有行

1.10.4.6.2 连接语法

- 基本语法: `left_join(x, y, by = "key")`
- 多键连接: `by = c("key1", "key2")`
- 不同列名: `by = c("x_key" = "y_key")`

1.10.5 课后练习

题目: 某湿地生物多样性调查数据:

```
wetland_survey <- tibble(
  site_id = rep(c("W1", "W2", "W3"), each = 8),
  date = rep(c("2022-05", "2022-08", "2022-05", "2022-08"), times = 6),
  plant_species = c(15, 18, 12, 16, 20, 25, 18, 22, 8, 12, 6, 10),
  bird_species = c(8, 12, 6, 9, 15, 18, 12, 14, 4, 7, 3, 5),
  water_level = c(45, 38, 50, 42, 35, 28, 40, 33, 55, 48, 60, 53)
)

site_characteristics <- tibble(
  site_id = c("W1", "W2", "W3"),
  area_ha = c(12, 8, 15),
  protection_status = c("保护区", "缓冲区", "实验区")
)
```

请完成 (使用 tidyverse 工具链, 结合之前学过的统计和可视化方法):

1. 筛选出 5 月份的调查数据, 并计算植物和鸟类物种总数 (使用 `filter()` 和 `mutate()`)
2. 按站点分组, 计算各站点的平均物种数和水位变化范围 (使用 `group_by()` 和 `summarise()`)
3. 连接站点特征数据, 创建物种密度指标 (物种数/面积) (使用 `left_join()` 和 `mutate()`)
4. 将数据从宽格式转换为长格式, 便于后续统计分析 (使用 `pivot_longer()`)
5. 根据保护状态和季节, 分析不同组合下的生物多样性特征 (使用 `group_by()` 和统计函数)
6. 使用 `ggplot2` 创建专业的可视化图表展示分析结果
7. 与第 9 课的传统方法对比, 体会 tidyverse 的优势

1.11 图形语法与科学绘图

1.11.1 生态学背景

数据可视化是生态学研究中传达发现和支持论据的关键工具。与基础 R 绘图相比, ggplot2 采用图形语法, 能够创建更加专业、美观的科学图表, 满足期刊发表和学术报告的高标准要求。

1.11.2 演示数据

```
# 某保护区多年生物多样性监测数据
library(ggplot2)
library(dplyr)

biodiversity_data <- data.frame(
  year = rep(2018:2022, each = 12),
  month = rep(1:12, times = 5),
  temperature = rnorm(60, mean = 15 + 5*sin(2*pi*(rep(1:12, times=5)-1)/12), sd = 2),
  species_richness = rpois(60, lambda = 25 + 10*sin(2*pi*(rep(1:12, times=5)-1)/12)),
  habitat = rep(c("森林", "草地", "湿地"), length.out = 60),
  elevation = rep(c(1200, 1000, 800), length.out = 60)
)
```

1.11.3 课堂演示过程

1.11.3.1 ggplot2 基础语法

```
library(ggplot2)
library(dplyr)

# 创建示例数据
bird_data <- data.frame(
  species = c("白头鵙", "麻雀", "喜鹊", "乌鸦", "燕子", "画眉"),
  abundance = c(45, 78, 32, 28, 56, 41),
  habitat = c("森林", "城市", "农田", "城市", "农田", "森林"),
  body_mass = c(25, 15, 180, 350, 18, 35)
)

# 基础散点图
ggplot(bird_data, aes(x = body_mass, y = abundance)) +
  geom_point()

# 添加颜色映射
ggplot(bird_data, aes(x = body_mass, y = abundance, color = habitat)) +
  geom_point(size = 3)

# 添加标题和标签
ggplot(bird_data, aes(x = body_mass, y = abundance, color = habitat)) +
  geom_point(size = 3) +
  labs(
    title = "鸟类体重与丰度关系",
    subtitle = "不同栖息地类型的比较",
    x = "体重 (g)",
    y = "丰度 (个体数)",
```

```
    color = " 栖息地类型"
)
```

1.11.3.2 不同类型的图表

```
# 创建时间序列数据
time_series_data <- data.frame(
  month = rep(1:12, 3),
  species_count = c(
    20, 25, 35, 45, 55, 60, 58, 52, 42, 32, 25, 22, # 2020 年
    22, 28, 38, 48, 58, 65, 62, 55, 45, 35, 28, 25, # 2021 年
    25, 30, 40, 50, 60, 68, 65, 58, 48, 38, 30, 28 # 2022 年
  ),
  year = rep(c("2020", "2021", "2022"), each = 12)
)

# 线图
ggplot(time_series_data, aes(x = month, y = species_count, color = year)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  labs(
    title = " 月度物种数量变化",
    x = " 月份",
    y = " 物种数量",
    color = " 年份"
  ) +
  theme_minimal()

# 柱状图
habitat_summary <- bird_data %>%
  group_by(habitat) %>%
  summarise(
    mean_abundance = mean(abundance),
    se_abundance = sd(abundance) / sqrt(n()),
    .groups = 'drop'
  )

ggplot(habitat_summary, aes(x = habitat, y = mean_abundance, fill = habitat)) +
  geom_col() +
  geom_errorbar(
    aes(ymin = mean_abundance - se_abundance,
        ymax = mean_abundance + se_abundance),
    width = 0.2
  ) +
  labs(
    title = " 不同栖息地的鸟类平均丰度",
    x = " 栖息地类型",
    y = " 平均丰度",
    fill = " 栖息地"
  ) +
  theme_classic()
```

```
# 箱线图
ggplot(bird_data, aes(x = habitat, y = abundance, fill = habitat)) +
  geom_boxplot() +
  geom_jitter(width = 0.2, alpha = 0.6) +
  labs(
    title = "不同栖息地鸟类丰度分布",
    x = "栖息地类型",
    y = "丰度"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

1.11.3.3 多面板图形

```
# 创建多组数据
multi_species_data <- data.frame(
  species = rep(c("鸟类", "哺乳动物", "昆虫"), each = 24),
  month = rep(1:12, 6),
  year = rep(c("2021", "2022"), each = 12, times = 3),
  abundance = c(
    # 鸟类数据
    rnorm(24, mean = 30 + 15*sin(2*pi*(1:24-1)/12), sd = 5),
    # 哺乳动物数据
    rnorm(24, mean = 15 + 8*sin(2*pi*(1:24-1)/12), sd = 3),
    # 昆虫数据
    rnorm(24, mean = 80 + 40*sin(2*pi*(1:24-1)/12), sd = 15)
  )
)

# 分面图
ggplot(multi_species_data, aes(x = month, y = abundance, color = year)) +
  geom_line(size = 1) +
  geom_point() +
  facet_wrap(~ species, scales = "free_y") +
  scale_x_continuous(breaks = c(3, 6, 9, 12)) +
  labs(
    title = "不同类群动物的季节性变化模式",
    x = "月份",
    y = "丰度",
    color = "年份"
  ) +
  theme_bw()
```

1.11.3.4 专业主题和自定义

```
# 创建专业期刊风格的图表
publication_plot <- ggplot(bird_data, aes(x = body_mass, y = abundance)) +
  geom_point(aes(color = habitat), size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", se = TRUE, color = "black", linetype = "dashed") +
  scale_color_manual(values = c("森林" = "#2E8B57", "城市" = "#DC143C", "农田" = "#DAA520")) +
  labs(
    title = "鸟类体重与种群丰度的关系",
    x = "体重 (g)",
```

```

y = " 种群丰度 (个体数)",
color = " 栖息地类型",
caption = " 数据来源: 某自然保护区鸟类调查 (2022)"
) +
theme_minimal() +
theme(
  plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
  axis.title = element_text(size = 12),
  axis.text = element_text(size = 10),
  legend.title = element_text(size = 11),
  legend.text = element_text(size = 10),
  panel.grid.minor = element_blank(),
  plot.caption = element_text(size = 8, color = "gray50")
)
print(publication_plot)

# 保存图片
ggsave("bird_analysis.png", publication_plot,
       width = 8, height = 6, dpi = 300)

```

```

# 群落组成气泡图
community_data <- data.frame(
  site = rep(c(" 样地 A", " 样地 B", " 样地 C"), each = 6),
  species = rep(c(" 物种 1", " 物种 2", " 物种 3", " 物种 4", " 物种 5", " 物种 6"), 3),
  abundance = c(25, 15, 8, 32, 12, 6, # 样地 A
               18, 22, 12, 25, 8, 15, # 样地 B
               12, 8, 25, 18, 20, 17), # 样地 C
  biomass = c(2.5, 3.2, 1.8, 4.1, 2.0, 1.2,
             3.0, 2.8, 2.2, 3.5, 1.5, 2.5,
             2.2, 1.8, 4.0, 2.9, 3.8, 3.2)
)
ggplot(community_data, aes(x = species, y = site)) +
  geom_point(aes(size = abundance, color = biomass), alpha = 0.7) +
  scale_size_continuous(range = c(2, 12), name = " 丰度") +
  scale_color_gradient(low = "lightblue", high = "darkred", name = " 生物量 (kg)") +
  labs(
    title = " 群落物种组成与生物量分布",
    x = " 物种",
    y = " 样地"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid = element_line(color = "gray90", size = 0.3)
)

```

1.11.4 R 语言知识点详解

1.11.4.1 ggplot2 的图形语法

1.11.4.1.1 基本概念

- **数据 (Data)**: 要可视化的数据集
- **美学映射 (Aesthetics)**: 数据变量到图形属性的映射
- **几何对象 (Geometries)**: 用来表示数据的图形元素
- **统计变换 (Statistics)**: 对原始数据的统计总结
- **坐标系统 (Coordinates)**: 数据如何映射到平面
- **分面 (Facets)**: 将数据分割成子集的方法
- **主题 (Themes)**: 控制图形整体外观

```
ggplot(data, aes(x = var1, y = var2)) +
  geom_() +
  scale_() +
  labs() +
  theme_()
```

1.11.4.1.2 基本语法结构

1.11.4.2 美学映射系统

1.11.4.2.1 aes() 函数

- 位置映射: x、y
- 颜色映射: color (边框)、fill (填充)
- 大小映射: size
- 形状映射: shape
- 透明度映射: alpha
- 线型映射: linetype

1.11.4.2.2 映射 vs 设定

- 映射: aes(color = variable), 颜色根据变量值变化
- 设定: geom_point(color = "red"), 所有点都是红色

1.11.4.3 几何对象详解

1.11.4.3.1 点图相关

- **geom_point()**: 散点图
- **geom_jitter()**: 抖动散点图
- 参数: size、shape、alpha、stroke

1.11.4.3.2 线图相关

- **geom_line()**: 线图
- **geom_path()**: 路径图
- **geom_smooth()**: 拟合线
- 参数: size、linetype、method

1.11.4.3.3 柱状图相关

- **geom_col()**: 柱状图 (使用实际值)
- **geom_bar()**: 柱状图 (统计计数)

- `geom_histogram()`: 直方图
- 参数: `width`、`position`

1.11.4.3.4 分布图相关

- `geom_boxplot()`: 箱线图
- `geom_violin()`: 小提琴图
- `geom_density()`: 密度图

1.11.4.4 标度系统

1.11.4.4.1 颜色标度

- 连续型:
 - `scale_color_gradient()`: 双色渐变
 - `scale_color_gradient2()`: 三色渐变
 - `scale_color_viridis_c()`: viridis 调色板
- 离散型:
 - `scale_color_manual()`: 手动设置颜色
 - `scale_color_brewer()`: ColorBrewer 调色板

1.11.4.4.2 坐标轴标度

- 连续型:
 - `scale_x_continuous()`: 连续 x 轴
 - `scale_y_log10()`: 对数 y 轴
- 离散型:
 - `scale_x_discrete()`: 离散 x 轴
- 日期型:
 - `scale_x_date()`: 日期 x 轴

1.11.4.5 分面系统

1.11.4.5.1 `facet_wrap()`

- 用途: 按一个变量分面, 排列成网格
- 语法: `facet_wrap(~ variable, ncol = 2)`
- 参数:
 - `ncol`、`nrow`: 列数和行数
 - `scales`: 坐标轴缩放方式

1.11.4.5.2 `facet_grid()`

- 用途: 按两个变量分面, 形成矩阵
- 语法: `facet_grid(rows ~ cols)`
- 特殊语法:
 - `facet_grid(. ~ variable)`: 仅按列分面
 - `facet_grid(variable ~ .)`: 仅按行分面

1.11.4.6 主题系统

1.11.4.6.1 预设主题

- `theme_minimal()`: 简洁主题
- `theme_classic()`: 经典主题
- `theme_bw()`: 黑白主题
- `theme_void()`: 空白主题

1.11.4.6.2 自定义主题元素

- 文本元素: `element_text()`
 - `size`: 字体大小
 - `color`: 字体颜色
 - `face`: 字体样式 (“bold”、“italic”)
 - `hjust`, `vjust`: 水平和垂直对齐
- 线条元素: `element_line()`
 - `color`: 线条颜色
 - `size`: 线条粗细
 - `linetype`: 线条类型
- 矩形元素: `element_rect()`
 - `fill`: 填充颜色
 - `color`: 边框颜色
- 移除元素: `element_blank()`

1.11.4.7 图片保存

1.11.4.7.1 `ggsave()` 函数

- 语法: `ggsave(filename, plot, width, height, dpi, units)`
- 支持格式:
 - 矢量格式: PDF、SVG、EPS
 - 位图格式: PNG、JPEG、TIFF
- 推荐设置:
 - 期刊投稿: 300-600 DPI
 - 演示文稿: 150-300 DPI
 - 网页使用: 72-150 DPI

1.11.4.8 色彩设计原则

1.11.4.8.1 科学可视化色彩指南

- 连续数据: 使用渐变色, 避免彩虹色
- 分类数据: 使用对比鲜明的颜色
- 色盲友好: 避免红绿组合, 推荐 viridis 调色板
- 发表要求: 考虑黑白印刷效果

1.11.4.8.2 推荐调色板

- Viridis 系列: 色盲友好, 打印友好
- ColorBrewer: 专业的制图调色板
- 自然色彩: 模仿自然界的颜色组合

1.11.5 课后练习

题目: 某国家公园植被多样性调查数据:

```
vegetation_survey <- data.frame(
  transect = rep(c(" 山顶", " 山腰", " 山底"), each = 20),
  species_richness = c(rnorm(20, 15, 3), rnorm(20, 25, 4), rnorm(20, 35, 5)),
  coverage_percent = c(rnorm(20, 60, 10), rnorm(20, 75, 8), rnorm(20, 85, 6)),
  slope_degree = c(rnorm(20, 25, 5), rnorm(20, 15, 3), rnorm(20, 5, 2)),
  soil_depth = c(rnorm(20, 15, 3), rnorm(20, 25, 4), rnorm(20, 40, 6))
)
```

请完成 (使用 ggplot2 高级功能, 结合之前学过的所有内容):

1. 创建物种丰富度与植被覆盖度的散点图，用颜色区分不同海拔带（使用 `geom_point()` 和 `aes()`）
2. 绘制三个海拔带物种丰富度的箱线图，添加个体数据点（使用 `geom_boxplot()` 和 `geom_jitter()`）
3. 创建多面板图，展示不同海拔带的各项指标分布（使用 `facet_wrap()`）
4. 设计一个期刊级别的综合图表，展示海拔梯度上的植被特征变化（使用多个 `geom` 层）
5. 自定义主题，确保图表符合学术发表标准（使用 `theme()` 函数）
6. 保存高质量图片用于论文发表（使用 `ggsave()` 函数）
7. 与第 7 课的基础绘图方法对比，总结 `ggplot2` 的优势
8. 尝试创建动态或交互式可视化（选做，可查阅相关资料）

1.12 综合练习

1.12.1 练习 1

请用上面所学产生一个斐波那契数列的前 46 个值，其中斐波那契数列的具体定义如下：

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2} \quad (n \geq 2, n \in \mathbf{N})$$