

# 《R语言编程艺术》

## 图书基本信息

书名：《R语言编程艺术》

13位ISBN编号：9787111423140

10位ISBN编号：7111423143

出版时间：2013-5

出版社：机械工业出版社

作者：（美）Norman Matloff

页数：303

译者：陈堰平,邱怡轩,潘岚锋 等

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：[www.tushu000.com](http://www.tushu000.com)

# 《R语言编程艺术》

## 内容概要

# 《R语言编程艺术》

## 作者简介

Norman Matloff 著名计算机科学家兼统计学家，美国加州大学戴维斯分校计算机科学系教授，曾是该校统计专业的创建者之一，并担任过统计学教授。对并行编程、网络流量、数据挖掘、磁盘系统性能等方面的技术都有深入的研究。乐于分享，撰写了多部广受欢迎的关于软件开发的在线教程，多次为《纽约时报》、《华盛顿邮报》、《福布斯杂志》以及《洛杉矶时报》撰写文章，同时他还是《The Art of Debugging》的作者之一。

## 书籍目录

译者序

前言

致谢

### 第1章 快速入门

1.1 怎样运行R

1.1.1 交互模式

1.1.2 批处理模式

1.2 第一个R会话

1.3 函数入门

1.3.1 变量的作用域

1.3.2 默认参数

1.4 R语言中一些重要的数据结构

1.4.1 向量，R语言中的战斗机

1.4.2 字符串

1.4.3 矩阵

1.4.4 列表

1.4.5 数据框

1.4.6 类

1.5 扩展案例：考试成绩的回归分析

1.6 启动和关闭R

1.7 获取帮助

1.7.1 help()函数

1.7.2 example()函数

1.7.3 如果你不太清楚要查找什么

1.7.4 其他主题的帮助

1.7.5 批处理模式的帮助

1.7.6 互联网资源

### 第2章 向量

2.1 标量、向量、数组与矩阵

2.1.1 添加或删除向量元素

2.1.2 获取向量长度

2.1.3 作为向量的矩阵和数组

2.2 声明

2.3 循环补齐

2.4 常用的向量运算

2.4.1 向量运算和逻辑运算

2.4.2 向量索引

2.4.3 用：运算符创建向量

2.4.4 使用seq()创建向量

2.4.5 使用rep()重复向量常数

2.5 使用all()和any()

2.5.1 扩展案例：寻找连续出现1的游程

2.5.2 扩展案例：预测离散值时间序列

2.6 向量化运算符

2.6.1 向量输入，向量输出

2.6.2 向量输入，矩阵输出

2.7 NA与NULL值

- 2.7.1 NA的使用
- 2.7.2 NULL的使用
- 2.8 筛选
  - 2.8.1 生成筛选索引
  - 2.8.2 使用subset()函数筛选
  - 2.8.3 选择函数which()
- 2.9 向量化的ifelse()函数
  - 2.9.1 扩展案例：度量相关性
  - 2.9.2 扩展案例：对鲍鱼数据集重新编码
- 2.10 测试向量相等
- 2.11 向量元素的名称
- 2.12 关于c()的更多内容
- 第3章 矩阵和数组
  - 3.1 创建矩阵
  - 3.2 一般矩阵运算
    - 3.2.1 线性代数运算
    - 3.2.2 矩阵索引
    - 3.2.3 扩展案例：图像操作
    - 3.2.4 矩阵元素筛选
    - 3.2.5 扩展案例：生成协方差矩阵
  - 3.3 对矩阵的行和列调用函数
    - 3.3.1 使用apply()函数
    - 3.3.2 扩展案例：寻找异常值
  - 3.4 增加或删除矩阵的行或列
    - 3.4.1 改变矩阵的大小
    - 3.4.2 扩展案例：找到图中距离最近的一对端点
  - 3.5 向量与矩阵的差异
  - 3.6 避免意外降维
  - 3.7 矩阵的行和列的命名问题
  - 3.8 高维数组
- 第4章 列表
  - 4.1 创建列表
  - 4.2 列表的常规操作
    - 4.2.1 列表索引
    - 4.2.2 增加或删除列表元素
    - 4.2.3 获取列表长度
    - 4.2.4 扩展案例：文本词汇索引
  - 4.3 访问列表元素和值
  - 4.4 在列表上使用apply系列函数
    - 4.4.1 lapply()和sapply()的使用
    - 4.4.2 扩展案例：文本词汇索引（续）
    - 4.4.3 扩展案例：鲍鱼数据
  - 4.5 递归型列表
- 第5章 数据框
  - 5.1 创建数据框
    - 5.1.1 访问数据框
    - 5.1.2 扩展案例：考试成绩的回归分析（续）
  - 5.2 其他矩阵式操作
    - 5.2.1 提取子数据框

- 5.2.2 缺失值的处理
- 5.2.3 使用rbind()和cbind()等函数
- 5.2.4 使用apply()
- 5.2.5 扩展案例：工资研究
- 5.3 合并数据框
- 5.4 应用于数据框的函数
  - 5.4.1 在数据框上应用lapply()和sapply()函数
  - 5.4.2 扩展案例：应用Logistic模型
  - 5.4.3 扩展案例：学习中文方言的辅助工具
- 第6章 因子和表
  - 6.1 因子与水平
  - 6.2 因子的常用函数
    - 6.2.1 tapply函数
    - 6.2.2 split()函数
    - 6.2.3 by()函数
  - 6.3 表的操作
    - 6.3.1 表中有关矩阵和类似数组的操作
    - 6.3.2 扩展案例：提取子表
    - 6.3.3 扩展案例：在表中寻找频数最大的单元格
  - 6.4 其他与因子和表有关的函数
    - 6.4.1 aggregate()函数
    - 6.4.2 cut()函数
- 第7章 R语言编程结构
  - 7.1 控制语句
    - 7.1.1 循环
    - 7.1.2 对非向量集合的循环
    - 7.1.3 if-else结构
  - 7.2 算术和逻辑运算符及数值
  - 7.3 参数的默认值
  - 7.4 返回值
    - 7.4.1 决定是否显式调用return ()
    - 7.4.2 返回复杂对象
  - 7.5 函数都是对象
  - 7.6 环境和变量作用域的问题
    - 7.6.1 顶层环境
    - 7.6.2 变量作用域的层次
    - 7.6.3 关于ls()的进一步讨论
    - 7.6.4 函数（几乎）没有副作用
    - 7.6.5 扩展案例：显示调用框的函数
  - 7.7 R语言中没有指针
  - 7.8 向上级层次进行写操作
    - 7.8.1 利用超赋值运算符对非局部变量进行写操作
    - 7.8.2 用assign()函数对非局部变量进行写操作
    - 7.8.3 扩展案例：用R语言实现离散事件仿真
    - 7.8.4 什么时候使用全局变量
    - 7.8.5 闭包
  - 7.9 递归
    - 7.9.1 Quicksort的具体实现
    - 7.9.2 拓展举例：二叉查找树

- 7.10 置换函数
  - 7.10.1 什么是置换函数
  - 7.10.2 扩展案例：可记录元素修改次数的向量类
- 7.11 写函数代码的工具
  - 7.11.1 文本编辑器和集成开发环境
  - 7.11.2 edit()函数
- 7.12 创建自己的二元运算符
- 7.13 匿名函数
- 第8章 数学运算与模拟
  - 8.1 数学函数
    - 8.1.1 扩展例子：计算概率
    - 8.1.2 累积和与累积乘积
    - 8.1.3 最小值和最大值
    - 8.1.4 微积分
  - 8.2 统计分布函数
  - 8.3 排序
  - 8.4 向量和矩阵的线性代数运算
    - 8.4.1 扩展示例：向量叉积
    - 8.4.2 扩展示例：确定马尔科夫链的平稳分布
  - 8.5 集合运算
  - 8.6 用R做模拟
    - 8.6.1 内置的随机变量发生器
    - 8.6.2 重复运行时获得相同的随机数流
    - 8.6.3 扩展案例：组合的模拟
- 第9章 面向对象的编程
  - 9.1 S3类
    - 9.1.1 S3泛型函数
    - 9.1.2 实例：线性模型函数lm()中的OOP
    - 9.1.3 寻找泛型函数的实现方法
    - 9.1.4 编写S3类
    - 9.1.5 使用继承
    - 9.1.6 扩展示例：用于存储上三角矩阵的类
    - 9.1.7 扩展示例：多项式回归程序
  - 9.2 S4类
    - 9.2.1 编写S4类
    - 9.2.2 在S4类上实现泛型函数
  - 9.3 S3类和S4类的对比
  - 9.4 对象的管理
    - 9.4.1 用ls()函数列出所有对象
    - 9.4.2 用rm()函数删除特定对象
    - 9.4.3 用save()函数保存对象集合
    - 9.4.4 查看对象内部结构
    - 9.4.5 exists()函数
- 第10章 输入与输出
  - 10.1 连接键盘与显示器
    - 10.1.1 使用scan()函数
    - 10.1.2 使用readline()函数
    - 10.1.3 输出到显示器
  - 10.2 读写文件

- 10.2.1 从文件中读取数据框或矩阵
- 10.2.2 读取文本文件
- 10.2.3 连接的介绍
- 10.2.4 扩展案例：读取PUMS普查数据
- 10.2.5 通过URL在远程计算机上访问文件
- 10.2.6 写文件
- 10.2.7 获取文件和目录信息
- 10.2.8 扩展案例：多个文件内容的和
- 10.3 访问互联网
  - 10.3.1 TCP/IP概述
  - 10.3.2 R中的socket
  - 10.3.3 扩展案例：实现R的并行计算
- 第11章 字符串操作
  - 11.1 字符串操作函数概述
    - 11.1.1 grep()
    - 11.1.2 nchar()
    - 11.1.3 paste()
    - 11.1.4 sprintf()
    - 11.1.5 substr()
    - 11.1.6 strsplit()
    - 11.1.7 regexpr()
    - 11.1.8 gregexpr()
  - 11.2 正则表达式
    - 11.2.1 扩展案例：检测文件名的后缀
    - 11.2.2 扩展案例：生成文件名
  - 11.3 在调试工具edtdbg中使用字符串工具
- 第12章 绘图
  - 12.1 创建图形
    - 12.1.1 基础图形系统的核心：plot()函数
    - 12.1.2 添加线条：abline()函数
    - 12.1.3 在保持现有图形的基础上新增一个绘图窗口
    - 12.1.4 扩展案例：在一张图中绘制两条密度曲线
    - 12.1.5 扩展案例：进一步考察多项式回归
    - 12.1.6 添加点：points()函数
    - 12.1.7 添加图例：legend()函数
    - 12.1.8 添加文字：text()函数
    - 12.1.9 精确定位：locator()函数
    - 12.1.10 保存图形
  - 12.2 定制图形
    - 12.2.1 改变字符大小：cex选项
    - 12.2.2 改变坐标轴的范围：xlim和ylim选项
    - 12.2.3 添加多边形：polygon()函数
    - 12.2.4 平滑散点：lowess()和loess()函数
    - 12.2.5 绘制具有显式表达式的函数
    - 12.2.6 扩展案例：放大曲线的一部分
  - 12.3 将图形保存到文件
    - 12.3.1 R图形设备
    - 12.3.2 保存已显示的图形
    - 12.3.3 关闭R图形设备



## 12.4 创建三维图形

## 第13章 调试

### 13.1 调试的基本原则

#### 13.1.1 调试的本质：确认原则

#### 13.1.2 从小处着手

#### 13.1.3 模块化的、自顶向下的调试风格

#### 13.1.4 反漏洞

### 13.2 为什么要使用调试工具

### 13.3 使用R的调试工具

#### 13.3.1 利用debug()和browser()函数进行逐步调试

#### 13.3.2 使用浏览器命令

#### 13.3.3 设置断点

#### 13.3.4 使用trace()函数进行追踪

#### 13.3.5 使用traceback()和debugger()函数对崩溃的程序进行检查

#### 13.3.6 扩展案例：两个完整的调试会话

### 13.4 更方便的调试工具

### 13.5 在调试模拟数据的代码时请确保一致性

### 13.6 语法和运行时错误

### 13.7 在R上运行GDB

## 第14章 性能提升：速度和内存

### 14.1 编写快速的R代码

### 14.2 可怕的for循环

#### 14.2.1 用向量化提升速度

#### 14.2.2 扩展案例：在蒙特卡罗模拟中获得更快的速度

#### 14.2.3 扩展案例：生成幂次矩阵

### 14.3 函数式编程和内存问题

#### 14.3.1 向量赋值问题

#### 14.3.2 改变时拷贝

#### 14.3.3 扩展案例：避免内存拷贝

### 14.4 利用Rprof()来寻找代码的瓶颈

#### 14.4.1 利用Rprof()来进行监视

#### 14.4.2 Rprof()的工作原理

### 14.5 字节码编译

### 14.6 内存无法装下数据怎么办

#### 14.6.1 分块

#### 14.6.2 利用R软件包来进行内存管理

## 第15章 R与其他语言的接口

### 15.1 编写能被R调用的C/C++函数

#### 15.1.1 R与C/C++交互的预备知识

#### 15.1.2 例子：提取方阵的次对角线元素

#### 15.1.3 编译和运行程序

#### 15.1.4 调试R/C程序

#### 15.1.5 扩展案例：预测离散取值的时间序列

### 15.2 从Python调用R

#### 15.2.1 安装RPy

#### 15.2.2 RPy语法

## 第16章 R语言并行计算

### 16.1 共同外链问题

### 16.2 snow包简介

- 16.2.1 运行snow代码
- 16.2.2 分析snow代码
- 16.2.3 可以获得多少倍的加速
- 16.2.4 扩展案例：K均值聚类
- 16.3 借助于C
  - 16.3.1 利用多核机器
  - 16.3.2 扩展案例：利用OpenMP解决共同外链问题
  - 16.3.3 运行OpenMP代码
  - 16.3.4 OpenMP代码分析
  - 16.3.5 其他OpenMP指令
  - 16.3.6 GPU编程
- 16.4 普遍的性能考虑
  - 16.4.1 开销的来源
  - 16.4.2 简单并行程序，以及那些不简单的
  - 16.4.3 静态和动态任务分配
  - 16.4.4 软件炼金术：将一般的问题转化为简单并行问题
- 16.5 调试R语言并行计算的代码
- 附录A 安装R
- 附录B 安装和使用包

## 精彩短评

- 1、用到了一点点。。
- 2、适合程序员
- 3、前面蛮好的内容,后面有点难
- 4、写的非常好。非常具有操作性。
- 5、也是适合入门用的书，与R实战不同，本书注重从编程的角度入手，提到了很多细节，和R实战一起用作入门教材可谓相辅相成。
- 6、还行
- 7、刚开始，感觉还蛮好看的 隐约看到了好几种语言的影子  
前面几章看得比较粗糙，后面3-4章跳过了没看 mark下。也就先了解了基本的语法之类的啊。。希望能像学awk一样，慢慢的实践上手吧
- 8、这本书系统介绍了R语言的一些运算规则，适合初级R语言开发人才
- 9、入门一阶半的必读书目，对处理技巧、编程规范、优化调试都有很多有益的总结，多数用户是统计导向，交互界面跑命令粘结果完事，这不是正确的打开方式。
- 10、终于读完了，从二月份春节到现在，三月初，不到一个月的时间。比《R语言实战》更引人入胜，真正体会到了"R语言编程“，有些部分值得反复读，希望能有更深入的著作问世。。。R并行值得好好研究。。。
- 11、需要比一般水平略高的C语言功底才能全部读懂得书，如果C一点不会读起来恐怕相当吃力
- 12、比较偏编程方面，比较系统，入门应该找一些侧重统计内容的书籍
- 13、相当于官方文档的中文简版，一些函数功能讲的比较简略，举的很多例子我都跳过了，感觉例子没讲太好，不过网上资料也少，英文文档查的头疼，总得来说还行吧。
- 14、需要做一个 talk，又翻了一遍中文版，非常适合码农读
- 15、书是好书，可我真的看不进去编程教材
- 16、挺好的。对于编程者熟悉R语言有很大帮助，到最后还是要看各种package的文档吧。不涉及数据分析建模之类的。
- 17、对于编程更加侧重，有利于弄清楚各种内在语法和函数，以免糊涂。如果是想现用，用R实战；如果想更好的掌握R，先学R艺术。
- 18、这本书应该是挺好的，但是其中有些地方看不懂，看不明白，不知道是翻译的问题，还是原文的问题。
- 19、一般
- 20、如果用r做项目，我推荐一定要看这本。介绍了很多很实用的小函数，和一些编程需要注意的
- 21、我还是得找个东西亲手做做才上手得快啊
- 22、意思不大
- 23、提纲挈领
- 24、值得反复读几遍！
- 25、很好的入门书
- 26、非常强调函数式编程思维和方式，适合看完R in action入门之后对R对象的结构和编程过程有更深入的了解~
- 27、这本书从语言本身介绍了R的一系列特征，详细介绍了R中的常见数据类型，面向对象，字符串处理，绘图，调试与性能，并行计算等。书中知识点都很纯粹，没有涉及到数据科学中复杂的数学模型和相关算法。如果你想要全面了解R语言，建议读这本书。
- 28、我的R入门书。非常不错。就是讲绘图少了点。
- 29、入门的好书 翻译也不错
- 30、入门来说优于 r in action
- 31、结构喜欢，每章节讲一种数据类型，适合有一定基础，有时有恍然大悟的感觉
- 32、2016年的第一本书。读了好久。后半部分基本囫圇吞枣，什么都没看懂，以后理解深刻了再回来

## 《R语言编程艺术》

看。

33、R语言领域的两部杰作之一！

34、数据结构讲的清楚，不过还得配合官方文档看，google也总有意外惊喜。

35、R语言入门书，感觉稍微有点理解R的一些语言风格了。看完还是不会用R.....

36、上次装逼的时候看过

37、入门很好

38、结构非常清晰，从语言处理的各类元素展开解释，再介绍基础的操作。难度适中的入门读物，能对R的构架有比较扎实的打底。

39、额，是本学计算机的人写的书，过于强调语言

40、这本书很不错，读完之后觉得自己对R语言了解更全面了

41、还是不错滴

42、比较适合有计算机背景的人学习。

43、我的R语言入门书

44、最赞的是case study部分的内容。可以定位为一本“面向程序员而不是分析师的R语言教材”，但是感觉很多东西没有涉及到，难道只能去翻阅文档了么？

45、对于编程什么都不知道的小白来说，只要书里有自己不会的，能帮助自己的，都是好书。

46、主要面对程序员吧，就r语言的缺点提出了解决方案。

47、这本书对于R的描述确实很全面，但是作为一个初学者来说，应该先打好基础再看，对你的帮助就比较大；书中有很多例子都是提前用后来的知识叙述的，建议先看<R语言实战>再来看这个；

48、值得好好的阅读

49、编程角度而非统计角度讲解

50、就是它啦！R语言上的第一步~当做入门的书籍读还是很好的~

不过技能的学习还是要learn by doing~~

## 精彩书评

1、数据挖掘入门到精通—R语言视频教程课程观看地址：<http://www.xuetuwuyou.com/course/59>课程介绍一、课程所用软件：R 3.2.2 (64位) RStudio二、课程涉及到的技术点：1) R语言的基本语法、函数2) R中实用性很强的包3) 模式识别、分类预测算法原理及其实现三、课程学习目标：本课程讲解理论的同时结合大量的案例，让学习者可以快速掌握数据挖掘技能，并利用R数据处理、画图、实现数据挖掘模型的建立。学习完本课程，学习者能达到以下目标：1) 掌握基本R用法；2) 用R进行描述性统计分析、进行数据处理和数据可视化；3) 缺失值的清洗能力；4) 用R语言建立数据挖掘模型；四、课程大纲：第一章：基本概念介绍第1课、数据挖掘、R语言概念介绍第2课、软件安装和数据的读、写、修改第3课、基本概念讲解（向量、矩阵、因子、数据框、列表）第4课、基本图形的讲解和绘制第二章：实用软件包介绍及应用第5课、plyr包主函数讲解第6课、plyr包辅助函数讲解第7课、Ggpolt2介绍第8课、Ggpolt2实践第9课、reshape2包的讲解和实际操作第10课、课缺失值的处理第三章：算法讲解及应用第11课、knn原理简介第12课、knn算法实际操作第13课、决策树的理论讲解第14课、决策树实操第15课、人工神经网络的介绍1第16课、人工神经网络介绍2第17课、人工神经网络实操1第18课、人工神经网络实操2第19课、支持向量机原理介绍第20课、支持向量机的实操

2、这是我所读的R系列中的最佳读本！

3、不确定作者有没有在认真对待这本书的写作。我身为一个统计专业，同时搞算法研究的学生，在编程和统计两方面应该都没有瘸腿，但这本书根本没有让我对R的了解有更多的深入。书对简单易懂的东西谈得太多，而对那些生僻的东西谈得太少。生僻的东西很多都是读者感兴趣的，像是R与C/C++混编、R的数值计算、R的面向对象等等。看过这本书后，我也只是知道了R里有这么些功能，剩下的细则还是要自己到网上搜，搜过了才明白怎么把C程序用R运行等等（作者大面积照顾Linux用户，介绍Linux如何做，这是很让人不爽的）。大篇幅的示例程序也是很让人头疼的一件事，而且纯搞统计的人一般对算法是不敏感的，这些程序对他们而言算是太长了，就我个人来说，我也要费一番功夫才能搞清楚作者究竟在实现什么。同时缺乏数据，即便是读者按部就班地敲进去书中的代码，也就是那么不了了之，没法亲身实测程序结果。另外，书里没有练习，这对学习编程来说是件很痛苦的事。书里对R函数的介绍也不详细，对于一本站在编程角度介绍R的书来说，是不是更应当把书写得像国内matlab教材那样的风格呢？然而，作者始终没有把握住执笔的风格，前面详细过头，后面粗略过头。总之，读这本书不如硬着头皮看R的帮助文档来得实在。

## 章节试读

### 1、《R语言编程艺术》的笔记-第115页

Hadley Wickham的reshape包“让你可以只需使用两个函数melt和cast，就能灵活地重构和汇总数据”。此包可能需要一段时间来学习，但它非常强大。他的plyr包也具有很多用途。你可以从R的CRAN中下载这两个软件包。

### 2、《R语言编程艺术》的笔记-第50页

如果一个函数使用了向量化的运算符，那么它也被向量化了，从而使速度提升成为可能。

eg：

```
u<-c(4,56,7)
```

```
v<-c(12,4,56)
```

```
u>v #向量化运算符
```

```
w<- function (x) return(x+1)
```

```
w(u) #向量化运算符
```

### 3、《R语言编程艺术》的笔记-第226页

每次调用plot()，现有的图形窗口都会直接或间接地被新的图形替代。如果你不希望看到这样的结果，则可以根据你的操作系统来使用下面的命令：

在Linux系统下，执行X11;

在Mac系统下，执行macintosh();

在Windows下，执行windows()。

### 4、《R语言编程艺术》的笔记-第68页

将matrix()的byrow参数设置为T，可以使得矩阵按行排列  
对子矩阵进行赋值

page 68

案例：图像操作——拉什莫尔山图片将罗斯福总统的头像进行模糊处理

图像文件本质上就是矩阵，因为像素点也都是按行和列排列的。如果图像时彩色的话，就需要三个矩阵来存储，分别记录红、黄、蓝的强度值。

生成随机噪声，把目标像素点矩阵和噪声进行加权平均。

page 73

案例：生成协方差矩阵

关键函数：

```
row()&col()
```

```
ifelse()
```

page 74

对矩阵的行和列调用函数

apply()允许用户在矩阵各行或各列上调用指定的函数

apply(m,dimcode,f, fargs) 四个形参分别代表：矩阵、维度编号、应用在维度上的函数、可选参数集。若apply()函数对行进行处理，默认将每一行的结果变成新的一列，so，使用一下转秩就可以了。

案例：寻找异常值

向代码致敬~

```
> findols<-function(x){
+ findol<-function(xrow){
+ mdn<-median(xrow)
+ devs<-abs(xrow)
+ return (which.max(devs))
+ }
+ return (apply(x,1,findol))
+ }
```

如果在循环中每次往矩阵中添加一行，最后矩阵会变成一个大矩阵，这种做法是不可取得，最好一开始就定义好一个大矩阵，在循环的过程中逐行或列进行赋值，避免了在循环中重复创建矩阵，以免每次进行耗时的矩阵内存分配。

案例：找到途中距离最近的一对端点

的计算图中多个端点之间的距离是计算机或统计学中常见的例子，这类问题在聚类算法和基因问题中经常出现。考虑数据量很大、最短点不唯一的情况。

禁止矩阵自动降维：r<-z[,drop=FALSE]

可以将向量转化为矩阵：v<-as.matrix(u)

对矩阵的行和列进行命名：colname(z)<-c("a","b")

## 5、《R语言编程艺术》的笔记-第54页

filtering

$z^*z$  得到的是布尔值向量，等价于  $(z^*z) > 0$

利用筛选来进行替换

$x[x > 3] <- 0$

subset(x,x>5) 可以免去自己移除NA的麻烦

which() #选择函数，找到满足条件的元素的位置

eg: first<-function(x) return(which(x==1)[1])

一般来说，向量化方法会更快，但是如果1出现的位置比较靠后，又要另当别论了

## 6、《R语言编程艺术》的笔记-第66页

避免意外降维

之前编程，遇到一个错误一直不知该如何解决。就是提取矩阵的列时，如果提取的是一列（或一行



) ,R会自动的转成向量,而不是1\*N的矩阵。

这时,会出现一些意想不到的错误,比如,我一直以为dim会工作的很好,但是这种情形会出错。

解决方法也很简单,用Drop=F

```
mat[,s,drop=F]
```

## 7、《R语言编程艺术》的笔记-第58页

\*apply()函数系列是R中最受欢迎同时也是最常用的,该函数系列包括apply()、tapply()和lapply()。

## 8、《R语言编程艺术》的笔记-第60页

在统计学中,“异常值”(outlier)指的是那些和大多数观测值离得很远的少数点。

## 9、《R语言编程艺术》的笔记-第54页

```
mean(x, na.rm=T) #移除NA设置为真
```

### 2.7.2 NULL的使用

null的一个用法是在循环中创建向量,其中每次迭代都在这个向量上增加一个元素(则没有符合条件的增加元素时就是null~)

NULL被当做不存在而计数,是R的一种特殊对象,它没有模式

## 10、《R语言编程艺术》的笔记-第102页

因子(factor)是R语言中许多强大运算的基础,包括许多针对表格数据的运算。因子的设计思想来源于统计学中的名义变量(nominal variables),或称之为分类变量(categorical variables),这种变量的值本质上不是数字,而是对应为分类,例如民主党、共和党和无党派,尽管可以用数字对它们编码。

## 11、《R语言编程艺术》的笔记-第47页

向量的内存分配过程比较耗时,调用c(runs,i)时(runs<-c(runs,i))给新的向量分配了内存空间,每次执行时都会减慢代码的运行速度。一种替代方法是预先分配内存空间,如果确实需要提高速度,可以考虑用C语言重新编码。

扩展案例:预测离散值时间序列

```
cumsum #用来计算向量的累积和
```

关注每次循环中的计算量。

## 12、《R语言编程艺术》的笔记-第68页

对原本就是向量的对象,可以使用as.matrix()函数将其转化为矩阵。

## 13、《R语言编程艺术》的笔记-第61页



案例：对鲍鱼数据集重新编码

关键函数：

`ifelse()`

`which()` & `grps()` & `for()`

R的for循环可以针对字符串向量

eg：

`grps()` & `list()`

`for (gen in c("M","F")) grps[[gen]]&lt;-which(aba==[,1]gen) ???`

测试向量是否相等

`all(x==y)`

`identical(x,y)`要慎用 因为它的意思是“完全一样否” 1和1.0类型不同

向量元素命名

`name(x)`可以用元素的名称来引用

## 14、《R语言编程艺术》的笔记-第25页

`y`&lt;- "abc" #字符串是字符模式的单元素向量

`x`&lt;- c(1,2,3) #数值模式长度为3的数值向量

`z`&lt;- c("abc","29 88") #由两个字符串组成

矩阵由向量生成的方式有两种：行绑定（`rbind`）&列绑定（`cbind`），矩阵乘法的运算符是`%*%`，矩阵索引的下标从1开始（great！）

R语言中列表里的各项可以属于不同的数据类型，可以用组成成分的名称来访问列表中的元素。列表的一种常见用法是把多个值打包组合到一起，然后从函数中返回。

eg：R语言的设计者把`hist()`返回的信息打包到一个R列表中，这样可以通过美元符号来访问，并用其他R语言命令进行操作。（great！）

`str()` #可用来显示任何R对象的内部结构，不只限于列表

`data.frame()` #数据框,通常是通过读取文件或数据库来创建

类被泛型函数所需要，泛型函数是有着相似功能适用于某个特定的类的函数族。举个例子，`plot()`函数是一个泛型函数，可以对任意一个R对象使用`plot()`函数，R会根据对象的类寻找合适的画图函数。

my try：

> z<- c("abc","df")

> a<- paste(z)

> a

[1] "abc" "df"

> a<- paste(z,z)

> a

[1] "abc abc" "df df"

> b<-strsplit(a,"")

```
> b  
[[1]]  
[1] "abc" "abc"
```

```
[[2]]  
[1] "df" "df"
```

以上还有疑问，wait~

关于“类的作用”

15、《R语言编程艺术》的笔记-第38页

接下来我们会用大量时间关注以下话题：

循环补齐：在一定情况下自动延长向量

筛选：提取向量子集

向量化：对向量的某一个元素应用函数

x本质上是一个指针，重赋值是通过将x指向新向量的方式实现的

eg：

若现在需要一个函数去判断向量中第一个1所在位置的索引值，循环如果写成 `for(n in x)` 则无法获得所需元素的索引值，因此引入一个显示循环是必要的，而 `for(i in 1:length(x))` 中需要x的长度。但是如果x的长度为0会产生问题... R的高级函数 `seq()` 出场了...

如果要对向量中的特定元素赋值的话，则要事先声明。原因在于，在函数式语言中，读写向量中的元素，实际上由函数来完成。如果R事先不知道y是一个向量，那么函数将没有执行的对象。对于绑定，由于变量没有事先声明，所以它们的类型不受限制。

eg：

```
y<- vector (length=2)
```

```
y[2]<-1
```

```
x<-c(1,2)
```

向量索引强大的兼容性（great！）

eg：

```
y<-x[c(1,1,3)]
```

负数的下标表示删除，然后看看还剩什么

结合下标可以发现length很有用~

注意运算的优先级

eg:

```
i<-2
```

```
1:i-1 #向量的循环补齐
```

```
seq(from 1.1,to 2,by 0.1) # 生成等差数列的一组向量
```

```
rep(c(2,3,4),3) # rep(x,times)
```

`rep(c(2,3,4), each=2)` #指代交替出现的次数

`any(x>8)`可以用`all` or `any`做判断

记住R是一种函数式语言。

## 16、《R语言编程艺术》的笔记-第23页

只在函数体内部可见的变量对于这个函数来说就是“局部变量”，它们在函数返回值后就撤销了。 . . . R函数中的形式参数是局部变量。 . . . 在计算函数调用的取值时，R会把每个实际参数复制给对应额局部参数变量，继而改变那些在函数外不可见的变量的取值。

全局变量是在函数之外创建的变量，在函数内部也可以访问。

superassignment operator : `&lt;&lt;-` #可以在函数内部给全局变量赋值。

`g &lt;-function(x,y=2,z=T){}```` #其中y、z是默认参数仿佛回到大一的C++课堂 . . . 有的东西就是逃不掉

## 17、《R语言编程艺术》的笔记-第92页

在关系型数据库的世界里，最重要的一个操作是合并：两张表根据某个共同的变量的值组合到一起。相似地，R语言里两个数据框也可以用`merge()`函数合并在一起。

## 18、《R语言编程艺术》的笔记-第85页

在统计领域，R语言中典型矩阵用行表示不同的观测对象，用列表示不同的变量，若数据采集于不同的时间，则时间就成了一个新的维度。在R中，这样的数据成为数组。

eg:

`tests<- array(data=c(firsttest,secondtest),dim=c(3,2,2))`

也可以将三位数组合成四维的，数组的一个最常用的场合是表的计算。

摘抄到时间是一个新的维度时情不自禁地想起了星级穿越，还有爱~

## 19、《R语言编程艺术》的笔记-第52页

如果函数本身的返回值就是向量的话

eg :

`z12<-function(z) return(c(z,z^2))`

`x<-1:8`

`z12(x)`

`matrix(z12(x), ncol=2)` #这样就更清楚了

可以使用函数`sapply`(simplify apply)

sapply(x,f)可以对x的每个元素作用函数f()，并将最后的结果转化为矩阵  
矩阵的每一行表示返回向量的第一个值

## 20、《R语言编程艺术》的笔记-第71页

向量的元素要求都是同类型的，而列表（list）与向量不同，可以组合多个不同类型的对象。如果读者了解其他编程语言的话，可能会觉得R中的列表像python中的字典，也像Perl中的哈希表，或像C中的结构体（struct）类型。列表在R中扮演着一个至关重要的角色，是数据框和面向对象编程的基础。

## 21、《R语言编程艺术》的笔记-第318页

自动安装包：

step1

```
install.packages()
```

eg：&gt; install.packages("mvtnorm","/a/b/c") #将 mvtnorm这个安装包安装到/a/b/c

此时R会自动连接到CRAN下载并编译包，然后将其加载到新的目录a/b/c/mvtnorm

step2wait~被提醒说没有 install.package()这个函数 . . .

# 《R语言编程艺术》

## 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:[www.tushu000.com](http://www.tushu000.com)