Springer

R语言应用系列

R语言数据操作

Data Manipulation with R

Phil Spector

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

R语言应用系列

Data Manipulation with R

R语言数据操作

〔美〕菲尔·斯佩克特 Phil Spector 著

University of California, Berkeley

朱钰 柴文义 张颖 译

西安交通大学出版社

Xi'an Jiaotong University Press

陕西省版权局著作权合同登记号 图字 25-2010-114 号

图书在版编目(CIP)数据

R 语言数据操作/[美]斯佩克特(Phil Spector)著；朱钰，柴文义，张颖译. —西安：西安交通大学出版社，2011.7
书名原文：Data Manipulation with R
ISBN 978-7-5605-3873-0

Ⅰ.①R… Ⅱ.①斯… ②朱… ③柴… ④张… Ⅲ.①程序语言-程序设计 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2011)第 043078 号

书　　名	R 语言数据操作
著　　者	[美] 菲尔·斯佩克特
译　　者	朱钰　柴文义　张颖
策划编辑	李　颖
责任编辑	李　颖
出版发行	西安交通大学出版社 (西安市兴庆南路 10 号　邮政编码 710049)
网　　址	http://ligong.xjtupress.com
电　　话	(029)82668357　82667874(发行中心) (029)82668315　82669096(总编办)
传　　真	(029)82669097
印　　刷	西安交通大学印刷厂
开　　本	787mm×1092mm　1/16　**印张** 11.125
印　　数	0001～3000　**字数** 176 千字
版次印次	2011 年 7 月第 1 版　2011 年 7 月第 1 次印刷
书　　号	ISBN 978-7-5605-3873-0/TP·547
定　　价	34.00 元

读者购书、书店添货、如发现印装质量问题，请与本社发行中心联系、调换。
订购热线：(029)82665248　(029)82665249
投稿热线：(029)82665380
读者信箱：banquan1809@126.com

译者序

菲尔·斯佩克特，1983 年获美国德州农工大学统计学博士，之后在 SAS 公司效力，1987 年至今为加州大学伯克利分校统计学系应用程序管理员，1995 年至今为该系副教授。曾著有《S & S-Plus 导论》(An Introduction to S and S-Plus)，其著作还包括与人合著的《SAS 在线性模型中的应用》(SAS system for linear models)及许多关于统计计算和统计软件的文章，对于统计软件 SAS、S-Plus 及 R 有着十分丰富的应用经验和研究。

菲尔·斯佩克特的《R 语言数据操作》是近年来关于 R 软件应用的一部不可多得的好书，本书内容具有综合性、紧凑性和简洁性，是 R 语言数据处理技术的综合指南，对于统计应用和理论研究都很有帮助。

接触过 R 软件的读者对于 R 的数据操作都会有非常深刻的印象——一般是头疼。因为在 R 中有非常丰富和灵活的数据类型和格式，而如果一组数据没有采用适当的格式读入 R，便无法在 R 中进行正确的计算。对于各种数据格式的正确理解和使用往往是使用 R 语言的障碍。而本书正好为读者解决了这些问题。

本书囊括了从各种不同格式的数据文件读取数据的技术以及采用适当的 R 内部数据格式保存数据的技术，对于诸如日期和时间型数据的处理、下标工具的应用、字符型数据的处理以及数据框的应用技术都通过丰富的实际或模拟数据实例作了精彩讲解。

本书的前 7 章的初稿由西安财经学院朱钰提供，对外经济贸易大学柴文义对这些章节进行了审阅，提供了修改意见；第 8 章和第 9 章由柴文义提供初稿，朱钰审阅并提出修改意见。西安财经学院张颖对书稿进行了通篇润色和调整，使全书语言风格统一。西安财经学院统计学系 07 级学生丁静斓、雷丹、黎晓、李垚、李夏茜、刘若珊、乔静、尚文媛、宋娟、王淑睿、卫丹洁、徐卓君和严婕等 13 名同学参加了译前全书内容的讨论及对于书中程序的试验运行。

由于译者水平所限，翻译过程中难免差错，衷心希望广大读者不吝金

玉，谨在此代表参与翻译工作的全体人员提前表示万分的感谢。

朱　钰
于西安财经学院统计学院
zhuyutj628@yahoo.com.cn
2011年5月

前　言

R 语言为数据操作提供了丰富的工作环境，特别是对于用来进行统计建模和图示的数据。除了种类繁多、唾手可得的软件包，它还允许用户既可以使用成熟稳定的统计技术，也可以使用试验性的统计技术。在其它语言中表现尚可的一些技术在 R 中往往是非常低效的，但是，由于 R 的灵活性，这些技术在 R 中也可以实现。一般而言，这些技术的问题在于它们与规模不成比例，也就是说，随着问题规模的扩大，这些技术的运算速度会意想不到的减慢。本书的目标是展示 R 中的各种数据操作技术，充分利用 R 的优势，而非直接使用其它语言中的类似方法。由于这涉及到 R 存储数据的基本概念，本书的第 1 章主要讲述 R 中关于数据的基础知识，对于这一章的掌握是理解其余后面章节的前提。

由于任何一个用 R 语言处理数据的项目，其首要任务是把数据读入 R，使其在 R 中可用。第 2 章涵盖了从各种数据源（文本文件，电子表格，其它程序文件等）读取数据的技术，以及将数据对象用 R 本身的格式及其它程序可以接受的格式进行存储。第 3 章讨论关系数据库的问题，因为大型数据集往往以这样的形式存储。这一章还包括建立和使用数据库并对大型数据集进行操作的一些指导。

第 4 章介绍 R 中关于日期和时间的操作。关于日期和时间的一些操作可以通过使用简单的字符表达来完成，当日期和时间转换为内部形式后，需要进行比较和其它操作时，有更多的操作可以选择。R 中有用于存储日期和时间的各种机制，这一章是要鼓励拥有这种数据的用户尽早将它们转换为适当的类型。

虽然因子在数据建模和图示中具有不可否认的重要性，它们在进行更基本的数据操作时却往往“碍事”。第 5 章解决怎样把因子转换成对象，或者把对象转换成因子，以及在必要时如何避免因子的转换。

第 6 章探讨在 R 中使用下标的多种方式以访问和修改数据。下标（特别是逻辑下标）是 R 最强有力的工具之一。许多需要循环或复杂程序的运

算问题都可以通过使用下标有效而漂亮地解决。

虽然 R 通常被认为是一种处理数值的语言，但是更多的数据以字符串的形式出现，而非数值形式。除断开、整理字符串的基本函数外，R 提供了常见表达的完整操作。与矢量化相结合，大多数字符数据的问题可以简单而有效地解决。第 7 章处理这些领域的问题，重点是字符数据的问题。

由于大多数的分析，不管是基于模型的还是图形化的，都对数据框进行操作，本书的最后两章直接探讨基于数据框的操作。第 8 章讨论综合汇总（聚合）技术，对数据框的内容进行概括，往往按组别进行。第 9 章包括改造和重塑数据框，与第 8 章的内容有些联系。重点放在能够利用 R 优势的方法上，使得随着数据规模的增长，这些运算方法的效率保持在适当的水平。

本书似乎让读者觉得不习惯的一个方面是用等号（=）作为赋值操作符，而不是用传统的“获得”操作符（<-）。我发现使用等号比其它符号更合乎自然，所以我在所有的例子中都用它。仅在一个情况下这会引起麻烦，在（作为一个函数调用的一部分对一个变量赋值）第 8.7 节中讨论。

虽然本书的重点是使用内置于 R 的函数和方法，书中还是介绍了许多来自于 CRAN（R 的综合档案网）的程序包。这些程序包都是我个人觉得在自己的工作中很有用的。没有介绍其它程序包绝不是暗示这些程序包没用。事实上，随着大量由社会贡献的 R 新程序包的出现，建议认真的程序员访问 R 网（http://r-project.org 或最好是访问适当的镜像站点）来获取新的软件包。这个网站上的另一个宝贵资源是 R 简讯，往往提供有关使用新软件包的一些深入的信息。

我想在此表达我对 S 语言的原始开发者、R 核心开发团队以及全体 R 社区成员的最诚挚的谢意，感谢他们创建了这样一个美好的语言，并鼓励使用者以新的、令人激动的方式使用 R 语言！

目　录

第 1 章

R 中的数据

1.1 模式和类

R 中的每一个对象包含多个属性以描述该对象中信息的性质。R 数据 P.1
中最重要的两个属性是模式和类。当管理数据时，重要的一点是了解 R 支持不同类型的数据的差异，当数据出现问题时，问题往往在于处于特定运算中的数据不是正确的模式或类。

mode 函数列示 R 中任何对象的模式，class 函数列示对象的类。当进行数据操作时，最常见的单个对象模式是数字型、字符型和逻辑型。然而，由于 R 中的数据通常围绕一个数据的集合（例如，一个矩阵或数据集）旋转，往往会遇到其它模式。在确定如何在 R 中存储数据时，要考虑的一个重要因素就是所处理数据的模式。有的对象（如矩阵或其它一些数组）要求其中的所有数据属于相同的模式，有的（像列表和数据框）允许单一的对象中存在多种模式的数据。

除了 mode 和 class 函数，typeof 函数有时可以提供关于对象类型的额外信息，虽然其用处一般说来不像 mode 和 class 提供的信息那样大。

当规划数据怎样读入 R 时要考虑的另一个因素是类型数据。R 提供因子类来存储这些类型的数据，而在统计建模和制图函数中对因子自动进

行特殊处理。因为 R 只需要将每个水平存储一次，所以作为因子存储的值比普通存储值需要较少的存储空间。如果你检查一个因子对象的模式，即使它可能显示为字符数据，你会发现，它始终是数值型的，因此当对因子进行操作时，应该特别注意。class 函数或者在 1.3 节描述的其它谓词函数之一可用于识别因子，只要这些因子存储在 R 中。关于因子的更多信息可在第 5 章找到。

P.2 另一个重要的数据类型是日期和时间。虽然这类信息可以作为一个简单的字符形式存储，但是这种形式很难操作。R 提供了一些机制来存储日期，包括内置的 Date，POSIXlt 和 POSIXct 类，以及 R 使用者贡献的 chron 程序包。其间的差别及日期和时间的操作信息在第 4 章有描述。

最后，最常遇到的数据模式是列表。列表是 R 中最灵活的数据存储方式，因为它可以适应不同模式和长度的对象。R 中的许多函数用列表的形式来保存结果，而且列表提供了累积信息增量的一种很有吸引力的方式。当你需要列表的各个组成部分的模式时，可以使用 sapply 函数（第 8.3 节详细讨论），如下例所示：

```
> mylist = list(a=c(1,2,3),b=c("cat","dog","duck"),
+ d=factor("a","b","a"))
> sapply(mylist,mode)
         a          b          d
 "numeric" "character"  "numeric"
> sapply(mylist,class)
         a          b          d
 "numeric" "character"   "factor"
```

1.2 R 的数据存储

单个数值（标量）作为一个 R 过程的中心是非常罕见的，所以在 R 中数据操作所遇到的第一个问题就是用哪一种类型的对象来保存数据集。向量是 R 中存储多个数值的最简单的方式。C 函数（连接或合并的记号）让你在 R 中快速录入数据：

```
> x = c(1,2,5,10)
> x
[1]  1  2  5 10
```

```
> mode(x)
[1] "numeric"
> y = c(1,2,"cat",3)
> y
[1] "1"   "2"   "cat" "3"
> mode(y)
[1] "character"
> z = c(5,TRUE,3,7)
> z
[1] 5 1 3 7
> mode(z)
[1] "numeric"
```

请注意，当使用c合并不同模式的元素时，由此得到的向量模式与其各部分不相同。特别地，如果其中有字符元素，其它元素将被转换为字符；与数值型元素合并的逻辑元素，会被转换成为与数字相当的值，TRUE变成1，FALSE变成0。C函数还可以用来合并向量： P.3

```
> all = c(x,y,z)
> all
 [1] "1"   "2"   "5"   "10"  "1"   "2"   "cat" "3"   "5"
[10] "1"   "3"   "7"
```

由于合并后的向量的一些元素有字符模式，则整个向量转换为字符。

可以给该向量的元素指定名称，这在显示该对象时会用到，也可以用来通过下标访问该向量的元素(第6.1节)。当第一次创建向量时即可为元素指定名称，也可以在创建后用names函数为其添加或更改名称：

```
> x = c(one=1,two=2,three=3)
> x
  one   two three
    1     2     3
> x = c(1,2,3)
> x
[1] 1 2 3
> names(x) = c('one','two','three')
> x
  one   two three
    1     2     3
```

names函数的另一个特征是，它可以用作索引仅修改所选元素的名称：

```
> names(x)[1:2] = c('uno','dos')
> x
  uno   dos three
    1     2     3
```

关于 R 中的向量有一个令人惊讶的事实，即在许多情况下，如果在一次运算中涉及到两个不同长度的向量，R 将会对较短的向量进行循环，从而使长度可比。这其实是如下事实的推广：当运算中涉及一个向量和标量时，R 将默认重复标量，使之与向量的每一个元素相对应。因此，要对向量的每一个元素加一，标量 1 可以如下使用：

P.4

```
> nums = 1:10
> nums + 1
 [1]  2  3  4  5  6  7  8  9 10 11
```

如果加数是一个不同长度的向量，也会发生类似的情况：

```
> nums = 1:10
> nums + c(1,2)
 [1]  2  4  4  6  6  8  8 10 10 12
```

注意值 1 和 2 如何重复，以使运算成功。R 默认这种运算，除非较长对象的长度不是较短对象长度的整数倍：

```
> nums = 1:10
> nums + c(1,2,3)
 [1]  2  4  6  5  7  9  8 10 12 11
Warning message:
longer object length
        is not a multiple of shorter object length in:
                    nums + c(1, 2, 3)
```

请注意，这只是一个警告，该运算仍在进行。

数组是向量的一个多维延伸，和向量一样，数组中的对象都必须是同一个模式。R 最常用的数组是矩阵——一个 2 维数组。矩阵作为向量存储在 R 中，矩阵的列“堆积”在彼此顶部。matrix 函数将向量转换为矩阵。matrix 函数中的 nrow = 和 ncol = 参数分别指定矩阵的行和列数。如果只给出二者之一，R 将根据输入数据的长度计算另一个。

由于矩阵按列存储，matrix 函数假定输入向量将被按列转换为矩阵；当需要按行存储矩阵时，用 byrow = TURE 参数来执行。矩阵的模式就是其构成要素的模式，矩阵类型的报告是 matrix。此外，矩阵有一个属性叫做

dim,dim是一个长度为2的向量,包括矩阵的行数和列数。dim函数将返回这个向量。另外,用nrow或ncol函数可以访问矩阵的单个元素。

矩阵的行、列可以通过matrix函数的dimnames=参数指定名称,或在矩阵创建后通过dimnames或row.names函数指定名称。由于矩阵的行和列数不一定相同,dimnames值必须是一个列表,第一个元素是行名向量,第二个元素是列名向量。和向量一样,这些名称是用于显示的,可用于通过下标访问矩阵的元素。为了给矩阵中的一个维度指定名称,可以对其中不 P.5
希望命名的维度赋值为NULL。例如,要创建一个5×3的随机数矩阵(见第2.2节),并将各列命名为A,B和C,可以使用如下的语句:

```
> rmat = matrix(rnorm(15),5,3,
+                dimnames=list(NULL,c('A','B','C')))
> rmat
             A          B           C
[1,] -1.15822190 -1.1431019  0.464873841
[2,] -0.04083058  0.3705789  0.320723479
[3,] -0.25480412 -0.5972248 -0.004061773
[4,]  0.48423349 -0.8727114 -0.663439822
[5,]  1.93566841 -0.2338928 -0.605026563
```

同样,可以首先建立矩阵,然后单独对其行、列指定名称:

```
dimnames(rmat) = list(NULL,c('A','B','C'))
```

列表提供了一种在单个R对象中存储不同模式的对象的方法。请注意,当形成一个列表时,列表中每个对象的模式保留下来:

```
> mylist = list(c(1,4,6),"dog",3,"cat",TRUE,c(9,10,11))
> mylist
[[1]]
[1] 1 4 6

[[2]]
[1] "dog"

[[3]]
[1] 3

[[4]]
[1] "cat"
```

```
[[5]]
[1] TRUE

[[6]]
[1]  9 10 11

> sapply(mylist,mode)
[1] "numeric"   "character" "numeric"   "character"
[5] "logical"   "numeric"
```

P.6　重要的是，列表不必是同一模式的内容，上述简单的例子同时表明，元素的长度不必相同。

像 R 的其它对象一样，列表中的元素可以被命名，在创建列表时可以命名，如果列表已经存在也可以使用 names 赋值函数命名 。list 函数没有关键词参数，所以可以通过函数对列表中的元素进行命名：

```
> mylist = list(first=c(1,3,5),second=c('one','three','five'),
+               third='end')
> mylist
$first
[1] 1 3 5

$second
[1] "one"   "three" "five"

$third
[1] "end"
```

同样的结果可以通过使用 names 函数在创建列表(未命名)之后得到：

```
> mylist = list(c(1,3,5),c('one','three','five'),'end')
> names(mylist) = c('first','second','third')
```

许多数据分析围绕一个数据集——一组相互联系的值，可以作为一个单元处理——进行。例如，您可能收集到不同公司的资料，其中包括每个公司的名称、行业类型、从业人数、所提供健康计划的类型。对于这些变量，研究中的每一个公司都有相应的值。如果将数据存储在一个矩阵中，以行代表一个观测，以列代表变量，数据就很容易访问，但由于数据集中变量的模式往往不一样，矩阵将强制数值型变量存储为字符型变量。为易于