Multivariate Data Analysis with R

多元数据分析及其R实现

肖枝洪　朱　强　苏理云
魏正元　袁中尚　编著

科学出版社

多元数据分析及其R实现

肖枝洪　朱　强　苏理云
魏正元　袁中尚　编著

科学出版社
北京

内 容 简 介

数据分析以矩阵和统计软件为工具，根据多因素多指标试验与观测所得到的的数据资料，对研究对象的特征及内在规律进行统计分析和推断，其应用十分广泛。全书共10章，首先对R软件、探索性数据分析、多元正态分布的基本知识进行了简单的介绍，然后介绍了经典的统计分析方法，如多元回归分析、多元相关分析、非线性回归分析、多元聚类分析和判别分析、多元试验数据的主成分分析和因子分析，最后进一步介绍了比较新的数据分析方法，如对应分析、支持向量机、偏最小二乘回归。本书每一章节中的例子都给出了用R语言编写的程序，并对分析结果进行了详尽解释。

本书可作为统计学、信息与计算科学、应用数学、保险学与管理学、精算学、工程技术等专业本科生和研究生教材，也可供相关专业人士参考。

图书在版编目(CIP)数据

多元数据分析及其R实现 / 肖枝洪等编著. —北京: 科学出版社, 2013.6
ISBN 978-7-03-037734-0

Ⅰ. ① 多… Ⅱ. ① 肖… Ⅲ. ①多元分析-高等学校-教材 Ⅳ. ①O212.4

中国版本图书馆 CIP 数据核字(2013)第 120895 号

责任编辑：杨 岭 万 羽/责任校对：万 羽
责任印制：邝志强/封面设计：墨创文化

科学出版社出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com

成都创新包装印刷厂印刷
科学出版社发行 各地新华书店经销
*
2013年8月第 一 版 开本：16 (787×1092)
2013年8月第一次印刷 印张：13.75
字数：300千字
定价：35.00元

序

在当今信息时代, 各行各业都面临大量数据处理问题, 而且数据的表达形式也是多种多样: 既有连续型数据, 又有离散型数据; 既有名义数据, 又有排序数据. 因此, 经典的统计方法对于数据的分析已不够用, 从而, 数据处理方法和软件的实现也随之显得越来越重要.

为了适应时代的需要, 也为了促进统计特色专业的建设与发展, 我们通过广泛查阅资料, 对已有的文献和书籍进行比较和分析, 并根据多年的教学经验, 编写了本书. 同时, 本书也是湖北省省级教改项目“适应新时期人才培养的需要, 构建概率统计类课程体系”和重庆理工大学“统计特色专业建设”的一项研究成果. 提高大学生以及研究生进行数据挖掘、建立统计模型的能力是本项目改革的重点. 因此, 本书在写作上特别注意以下 6 个显著特点.

(1) 数据分析方法全面, 脉络分明: 分别介绍了多元回归分析、正交回归设计、典型相关分析、主成分分析、因子分析、聚类分析、判别分析、对应分析、支持向量机、偏最小二乘法等. 既考虑了经典的数据处理方法, 又考虑了比较近代的数据处理方法, 还考虑了机器学习中的若干方法. 不仅介绍了数据处理的原理和方法, 还对具体步骤进行了解释.

(2) 数据分析方法细腻详实, 所编写的程序可以通过手工输入来验证其准确性.

(3) 结合实际问题密切, 运用适当的方法对所考虑的问题, 例如经济问题、病例分类问题、农作物栽培问题等进行案例分析, 具有示范作用.

(4) 每章内容由浅入深, 浅显易懂, 但又能给人以更多的启示. 同时还配有若干习题, 帮助读者加强对相关知识的理解与巩固.

(5) 本书通篇构成一个有机整体, 读者阅读此书, 会感到逻辑性强, 问题的提出背景清晰.

(6) 本书运用 R 语言对许多例子进行了编程, 以供读者借鉴, 使读者有种工具在手的感觉, 可以增强自信心, 克服畏难的心理.

本书的内容在 64 个学时内可以全部讲完, 也可以根据学生的具体情况对本书的内容做适当取舍, 在 48 个学时下进行重点介绍.

本书由重庆理工大学数学与统计学院肖枝洪教授、华中农业大学理学院朱强博士以及重庆理工大学数学与统计学院苏理云博士和魏正元博士、山东大学公共卫生学院袁中尚博士共同编写而成. 其中, 肖枝洪编写第 1 章、第 4 章至第 8 章, 朱强编写第 9 章, 苏理云编写第 3 章, 魏正元编写第 2 章, 袁中尚编写第 10 章, 所有的 R 程序由肖枝洪、朱强以及袁中尚编写, 全书由肖枝洪统稿. 重庆理工大学数学与统计学院研究生赵忠校对本书文字的录入和校对也做了大量的工作, 在此表示衷心的感谢! 本书的出版得到了科学出版社、重庆理工大学教务处和重庆理工大学数学与统计学院、华中农业大学统计研究所、山东大学公共卫生学院的大力支持, 在此表示衷心的感谢! 同时也对本书的出版给予

关心和支持的同仁，致以衷心的感谢！本书为重庆理工大学“十二五规划教材”.

本书适合大、中专院校从事统计、应用数学、经济管理以及信息与计算机方面工作或学习的本科生、研究生和教师阅读. 章节的安排、内容的组织以及程序的编写，都是由肖枝洪、朱强、袁中尚、苏理云、魏正元等人共同商榷而定的. 虽然我们在编写的过程中作了很大努力，力求准确，但是由于水平有限，书中难免存在一些错误，敬请读者批评指正.

编 者

2013 年 4 月于重庆理工大学花溪校区

目 录

序

第 1 章 R 介绍 1

1.1 R 软件基本操作 1

1.2 R 向量 3

1.3 矩阵及其运算 9

1.4 因子 12

1.5 列表与数据框 14

1.6 输出输入 19

1.7 图形 21

习题 1 33

第 2 章 多元正态分布 34

2.1 p 维标准正态分布 34

2.2 p 维一般正态分布 35

2.3 p 维正态分布的统计推断 39

习题 2 46

第 3 章 多元线性回归 47

3.1 一元线性回归 47

3.2 多元线性回归分析 53

3.3 逐步回归及复共线性 62

习题 3 68

第 4 章 多元线性相关 69

4.1 多个变量的线性相关 69

4.2 两组变量的典型相关分析 77

4.3 典型相关分析的实例 80

习题 4 82

第 5 章 多元非线性回归 84

5.1 非线性回归方程的建立 84

5.2 Logistic 曲线回归 89

5.3 多项式回归 91

5.4 一次回归的正交设计 93

5.5 二次回归的正交组合设计 99

5.6 二次回归的旋转组合设计 103

习题 5 114

第 6 章 多元聚类与判别 115

6.1 聚类的根据 115

6.2 系统聚类法 118

6.3 动态聚类法 133
6.4 Bayes 判别 137
习题 6 143
第 7 章 多元数据的主成分分析 146
7.1 主成分分析法 146
7.2 主成分的应用 152
习题 7 154
第 8 章 多元数据的因子分析 156
8.1 因子分析法 156
8.2 方差极大正交旋转 164
8.3 对应分析法 169
习题 8 176
第 9 章 支持向量机 177
9.1 线性支持向量机 177
9.2 非线性支持向量机 181
9.3 支持向量回归机 186
9.4 模型的评价方法 187
9.5 支持向量机实例及 R 实现 191
习题 9 199
第 10 章 偏最小二乘回归 201
10.1 偏最小二乘回归的基本思想 201
10.2 偏最小二乘回归的基本算法 202
10.3 交叉有效性原则 204
10.4 偏最小二乘回归的实例分析 205
习题 10 209
参考文献 210

第 1 章　R 介 绍

本书的宗旨是介绍如何进行数据分析的原理和方法, 而数据分析的实现必须借助数据处理软件这个强有力的工具. 而目前国际上流行的一种数据处理软件 ——R 软件, 特别适合于做数据分析, 它既有很多现存的统计软件包可以直接利用, 也可以很灵活地用于自编程序, 所以我们选择这个软件来实现数据处理的算法.

R 软件作为一个项目 (project), 1995 年由 Auckland 大学统计系的 Robert Gentleman 和 Ross Ihaka 开始编制, 目前由 R 核心开发小组 (R development core team) 维护, 这个小组认真负责, 完全自愿工作, 并将全球优秀的统计应用软件打包提供给用户. 用户可以通过 R 项目的网站 (http://www.r-project.org) 了解有关 R 的最新信息和使用说明, 得到最新版本的 R 软件和基于 R 的应用统计软件包.

R 软件由一组数据操作、计算和图形展示的工具构成, 相对于其他同类软件, 其特色在于:

(1) 有效的数据处理和保存机制;

(2) 完整的数组和矩阵计算操作符;

(3) 连贯而又完整的数据分析工具;

(4) 图形工具可以对数据直接进行分析和展示, 同时可用于多种图形设备;

(5) 它是一种相当完善、间接而又高效的程序设计语言 (也就是 "S"), 包括条件语句、循环语句、用户定义的递归函数以及各种输入输出接口.

R 是一个非常好的开发新的交互式数据分析方法的工具, 它的开发周期短, 有大量的扩展包 (package) 可以使用. R 有强大的统计功能, 大多数经典的统计方法和最新的技术都可以在 R 中直接得到.

R 是自由软件, 不向使用者收取任何费用. 它是彻底面向对象的编程语言, 和 Matlab 很像. 用户要安装时, 可以进入 http://cran.r-project.org, 下载 "Download and Install R" 栏中的软件, 点击 "Download R for Windows ", 进入 "base", 下载 "Download R xxxx for Windows " (如 Download R 2.15.2 for Windows, 目前最新版本为 R 3.0.1), 然后按提示安装即可.

1.1　R 软件基本操作

R 系统的基本界面是一个交互式命令窗口, 命令提示符是一个大于符号, 即 ">". 命令运行的结果马上显示在命令下面. R 命令有两种形式: 表达式和赋值运算 (用 "< −" 或 "=" 表示赋值运算符). 在命令提示符后键入一个表达式表示计算此表达式, 回车 (用 "↷" 表示) 后立即显示结果. 例如:

```
> 3.5**2+sqrt(9) ↷
[1] 15.25
```

赋值运算把赋值号右边的值计算出来并赋给左边的变量.

```
> x1< − 0:10;x1 ⌒ # 如果一条语句后继续写语句, 需要用 ";" 隔开
[1] 0 1 2 3 4 5 6 7 8 9 10
```

表示此符号以后的这行语句是注释语句, R 软件不会执行紧接在 # 后的语句命令. 以后在 R 语句中省略 "⌒".

我们可以用向上光标键 ("△") 来找回以前运行的命令, 再次运行或修改后再运行. R 是区分大小写的, 所以 x 和 X 表示不同的变量, 特别要加以注意, 这和 SAS 软件有所不同. SAS 软件也是非常好的统计软件, 国际上称为标准的统计软件.

我们给出如下语句来绘制余弦曲线 (如图 1.1 所示).

```
>x1=0:100; x1  # 如果不需要显示 x1 的值, 可以去掉分号后的 "x1"
> x2 < −  x1*2*pi/100; x2
> y=cos(x2); y
> plot(x1,y,type='l')
```

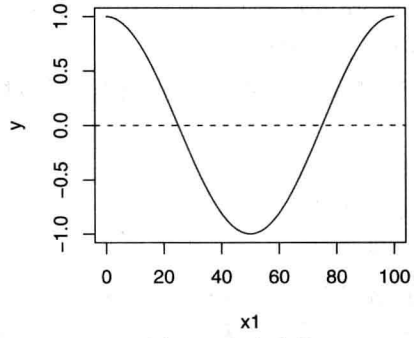

图 1.1 余弦曲线

上述语句中, 0 : 100 表示一个从 0 ∼ 100 的等差数列向量. 从第二个语句可以看出, 我们可以对向量直接进行四则运算. 从第三个语句可以看出, 函数可以输入向量, 并输出向量. 从最后一个语句可以看出, 函数的调用也很自由. 再看一个例子:

```
> vect < −c(10,6,4,7,8)
>mean(vect)
[1] 7
>sd(vect)
[1] 2.236066
>  min(vect)
[1]4
>  median(vect)
[1] 7
>  max(vect)
[1]10
>  boxplot(vect)
```

第一个语句输入若干数据到一个向量, 函数 c( ) 用来把数据组合为一个列向量, 但 R 是以行向量来显示. 不要用 c( ) 来定义其他形式的函数. 后面用了几个函数 (或称命令)

来计算数据的均值、标准差、中位数、最小值以及最大值. 最后的函数表示绘制数据的盒形图.

R 也提供了一般的计算功能. 例如, 求一个矩阵的逆:

```
> A=matrix(c(1,2,7,3),ncol=2,byrow=T); A
>  Ai=solve(A); Ai
```

还可以进行矩阵运算, 例如

```
> b< - c(2,3)
>  x=Ai %*% b
> x
```

这实际上是求解一个线性方程组. "%*%" 表示矩阵的乘法运算. 注意区别 "%*%" 和 "*".

可以把若干行命令保存在一个文本文件 (比如 D: \\ DATA_Analysis \\ R0120906.r) 中, 然后用 source 函数来运行整个文件.

```
> source( "D:\\ DATA_Analysis \\ R0120906.r" )
```

要退出 R, 可以用 q( ) 函数, 也可以直接用窗口上的 "×". R 在退出时会提示是否保存当前工作空间, 它可以把当前定义的所有对象 (有名字的向量、矩阵、列表、函数等) 保存到一个文件中. 本书建议不要保存当前工作空间, 以免以后和其他文件发生混淆. 需要结果时, 将所保存的 R 源程序运行一遍即可. 这和 SAS 软件也有所区别.

1.2 R 向 量

R 是基于对象的语言, 不过其最基本的数据还是预先定义好的数据类型, 如向量、矩阵、列表等. 更复杂的数据用对象表示, 比如, 数据框对象、时间序列对象、模型对象、图形对象等.

R 语言表达式可以使用常量和变量. 变量名的规则是: 由字母、数字和句点组成, 第一个字符必须是字母或句点, 长度没有限制, 区分大小写. 特别注意, 句点可以作为名字的合法成分, 而在其他面向对象的语言中, 句点经常用来分隔对象与成员名.

1.2.1 常量

常量笼统地分为逻辑型、数值型和字符型三种. 实际上数值型数据又可分为整型、单精度、双精度等. 除非特殊需要, 否则不必太关心其具体类型. 例如, 123,123.45,1.2345e30 是数值型常量, "Weight"、"李明" 是字符型 (用双引号或单引号包围). 逻辑真值写为 TRUE(注意区分大小写, 写为 true 没有意义), 逻辑假值写为 FALSE. 复数常量如 2.1−3.5i 这样表示.

R 中的数据可以取缺失值, 用 NA 表示缺失值. 函数 is.na(x) 返回 x 是否缺失值.

1.2.2 向量与赋值

向量 (vector) 是具有相同基本类型的元素序列, 大体相当于其他语言中的一维数组. 在 R 中标量也被看做一维向量. 定义向量最常用的方法是使用函数 c( ). 它把若干个数值或字符串组合为一个向量, 例如:

```
> vect < - c(10,6,4,7,8)
>x=c(1:3,10:13) ;  x
```

```
[1] 1 2 3 10 11 12 13
> x1=c(1,2)
> x2< - c(3,4)
> x< - c(x1,x2); x
[1] 1 2 3 4
```

在显示向量值时, 注意到左边出现一个 “[1]” , 表示该行第一个数的下标, 例如:

```
> 12:78
[1] 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
[26] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
[51] 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
```

第二行输出从第 26 个数开始, 所以在行左边显示 “[26]” .

函数 length(x) 可以计算向量 x 的长度, 例如:

```
> x< - c(12:78)
> x1=length(x); x1
[1] 67
```

1.2.3 向量运算

可以对向量进行加 (+)、减 (−)、乘 (∗)、除 (/) 和乘方 (∗∗或者∧) 运算, 其含义是对向量的每一个元素进行运算. 例如:

```
> x=c(1,2, 6.25)
> y=x**2+1; y
[1] 2.0000 5.0000 40.0625
```

另外, “%/%” 表示整除 (例如, $5\%/\%3 = 1$), “%%” 表示求余数 (例如, $5\%\%3 = 2$).

两个等长度向量间进行加、减、乘、除和乘方运算, 其含义是对应元素间进行四则运算, 例如:

```
> c(1,2,3)+c(10,20,30)/c(2,4,5)
[1] 6 7 9
```

两个长度不同的向量间进行加、减、乘、除和乘方运算时, 长度短的将循环使用, 但长度长的向量的长度应为短的向量的整数倍, 例如:

```
> c(1,2,3)+c(10,20,30,−2, −4, −9,0,0,1)
[1] 11 22 33 −1 − 2 − 6 1 2 4
```

函数 sqrt、log、exp、sin、cos 和 tan 等都可以用向量作为自变量, 结果是对向量的每一个元素取相应的函数值, 例如:

```
> x=c(1, 2, 6.25)
> sqrt(x)
[1] 1.000000 1.414214 2.500000
```

上述例子中出现 6 位小数, 在实际中有时并不需要达到此精度, 可以用命令 options(digits=3) 来控制结果的有效位数的输出. 例如:

```
> options(digits=3)
> sqrt(x)
```

```
[1] 1.00 1.41 2.50
```

函数 min 和 max 分别取自变量向量的最小值和最大值, 函数 sum 计算自变量向量的元素和, 函数 mean 计算均值, 函数 var 计算样本方差, 函数 sd 计算标准差. 但注意, 若变量 x 为矩阵, 则 var(x) 为协方差阵. 函数 range 返回包含最小值和最大值的向量. sort(x) 返回按 x 的元素从小到大排序的结果向量, order(x) 返回使 x 的元素从小到大排序的元素下标向量, 但 x[order(x)] 等效于 sort(x). 例如:

```
> z=c(−3, −9, 8, 2, 0, −8)
> range(z)
[1]  −9   8
> sort(z)
[1] −9 −8 −3  0  2  8
> order(z)
[1] 2 6 1 5 4 3
> z[order(z)]
[1] −9 −8 −3  0  2  8
```

若想将向量值按从大到小排序, 使用函数 sort(z, decreasing = TRUE) 即可.

任何函数值与缺失值的运算结果仍为缺失值. 例如:

```
> c(1,2,NA)−c(1,NA,4)
[1] 0 NA NA
```

1.2.4 产生有规律的数列

调用 R 的函数 numeric(n) 可以产生填充了 n 个零的向量. 在 R 中很容易产生一个等差数列. 例如:

```
> n=5
> 1:n
[1] 1 2 3 4 5
> n:2
[1] 5 4 3 2
> 1:n−1
[1] 0 1 2 3 4
> 1:(n−1)
[1] 1 2 3 4
```

注意区别 1:n−1 与 1:(n−1).

seq 函数是更一般的等差数列函数. 如果只指定一个自变量 n>0, 则 seq(n) 相当于 1:n. 指定两个自变量时, 第一个自变量是开始值, 第二个自变量是结束值, 如 seq(−2,3). 也可以写为 seq(from=−2, to=3). 还可以用一个参数 by 指定等差数列的步长, by 可以放在 seq 内的任何位置, 例如:

```
> seq(−2,3)
[1] −2 −1  0 1 2 3
> seq(from=−2,to=3)
```

```
[1] -2 -1  0 1 2 3
> seq(0,1,0.1)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(0,by=0.2,1)
[1] 0.0 0.2 0.4 0.6 0.8 1.0
> seq(by=0.2,from=0,1)
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

也可以用参数 length 指定数列长度, 例如:

```
> seq(0,length=5)
[1] 0 1 2 3 4
> seq(length=5,to=8)
[1] 4 5 6 7 8
```

seq 函数还可以用一种 seq(along= 向量名) 的格式, 产生该向量的下标序列, 例如:

```
> w <-c(4,-2,7,0)
> seq(along=w)
[1] 1 2 3 4
```

另一个类似的函数是 rep, 它可以重复第一个自变量若干次, 例如:

```
> rep(w,2)
[1] 4 -2 7 0 4 -2 7 0
> rep(c(10,28),c(2,3))
[1] 10 10 28 28 28
```

第一个参数名为变量名, 第二个参数为 times(重复次数). 也可以写为: rep(times=2,w).

1.2.5 逻辑向量

向量可以取逻辑值, 例如:

```
> l <- c(TRUE, TRUE, FALSE); l
[1] TRUE TRUE FALSE
> ll <- w>3;ll
[1] TRUE FALSE TRUE FALSE
```

一个向量与常量比较大小, 结果还是一个向量, 元素为每一对比较大小结果的逻辑值.

两个向量也可以比较, 例如:

```
> log(10*w)
[1] 3.688879 NaN 4.248495 - Inf
```

其中, Inf 表示无穷大.

```
> log(10*w)>w
[1] FALSE NA FALSE FALSE
```

比较运算符包括: <, <=,>=, ==(相等), ! =(不等). 注意区别 == 与 =, 意义不一样.

两个逻辑向量可以进行与运算 (&)、或运算 (|), 结果是对应元素的运算. 对逻辑向量 l 计算 !l 表示取每个元素的非, 例如:

```
> (w> 3)&(c(0,3,5,6)<2)
```

```
[1] TRUE FALSE FALSE FALSE
> !(w>3)
[1] FALSE TRUE FALSE TRUE
```

判断一个逻辑向量是否都为真值的函数是 all, 例如：

```
> all(!(w> 3))
[1] FALSE
```

判断一个逻辑向量是否有真值的函数是 any, 例如：

```
>any(!(w> 3))
[1] TRUE
```

函数 is.na(x) 用来判断 x 的每一个元素是否缺失, 例如：

```
> is.na(c(1,0,NA))
[1] FALSE FALSE TRUE
```

逻辑值可以强制转换为整数值, 如 TRUE 变为 1, FALSE 变为 0. 例如:

```
> age=30
> c( "young" , "old" )[(age> 65) + 1]
[1] "young"
```

1.2.6 字符型向量

向量元素可以取字符串值, 如：

```
> c1=c( "x" , "tan(x)" , "年龄" ) ; c1
[1] "x" "tan(x)" "年龄"
```

函数 paste 用来把它的自变量连成一个字符串, 中间用空格隔开, 例如:

```
> paste( "I", "love" , "homeland" )
[1] "I love homeland"
```

又例如：

```
> paste(c( "X", "Y" ), "=" , 1:4)
[1] "X = 1" "Y = 2" "X = 3" "Y = 4"
```

分隔用的字符可以用 sep 参数指定. 下例产生若干个文件名：

```
> paste( "result." , 1:5, sep= "" )
[1] "result. 1" "esult. 2" "result. 3" "result. 4" "result. 5"
```

如果给 paste( ) 函数指定了参数 collapse, 则 paste( ) 函数可以把一个字符串向量的各个元素连接成一个字符串, 中间用 collapse 指定的值分隔. 例如：

```
> paste(c( "123" , "234" ), collapse= "*" )
[1] "123*234"
```

1.2.7 向量下标运算

访问向量的某一个元素可以用 x[i] 格式, 其中 x 是一个向量名, 或是一个取向量值的表达式, i 是一个下标, 例如：

```
> x=seq(1,8,by=2);x[2]
[1] 3
```

```
> (c(2,4,6)+(−2))[2]
[1] 2
```

还可以单独改变一个元素的值, 例如:

```
> x[2] = 125; x
[1] 1 125 5 7
```

R 提供了 4 种方法访问向量的一部分, 格式为 x[v], x 为向量或向量值的表达式, v 是如下的一种下标向量.

1) 取正整数值的下标向量

在 x[v] 中, v 为一个向量, 元素取值在 1 ~ length(x) 之间, 取值允许重复, 例如:

```
> x[c(1,3)]
[1] 1 5
> x[1:2]
[1] 1 125
> x[c(1,3,2,1)]
[1] 1 5 125 1
> c( “a”, “b”, “c”)[rep(c(2,1,3),3)]
[1] “b” “a” “c” “b” “a” “c” “b” “a” “c”
```

2) 取负整数值的下标向量

在 x[v] 中, v 为一个向量, 元素取值在 −length(x)~ −1 之间, 表示去掉相应位置的元素, 例如:

```
> x[c(−1, −3)]
[1] 125 7
```

3) 取逻辑值的下标向量

在 x[v] 中, v 为和 x 等长度的逻辑向量, x[v] 表示取出所有 v 为真值的元素, 例如:

```
> x> 3
[1] FALSE TRUE TRUE TRUE
> x [x > 3]
[1] 125 5 7
> x[x< (−9)]
numeric(0)
```

可见, x [x>4] 取出所有大于 4 的元素. 这种逻辑值下标是一种强有力的检索工具, 例如,x[tan(x)> 0] 可以取出 x 中所有使正切值为正的元素组成的向量. 如果下标都是假值, 则结果显示一个长度为零的向量, 即 numeric(0).

4) 取字符型值的下标向量

在定义向量时, 可以给元素加上名字, 例如:

```
> ages=c(Li=33,zhang=29,Liu=18)
> ages
Li zhang Liu
33 29 18
```

这样定义的向量可以用通常的方法访问, 另外, 还可以用元素名字来访问元素或元素子集,

例如：

```
> ages[ “zhang”]
zhang
29
> ages[c( “zhang”, “Li”)]
zhang Li
29     33
```

在 R 中, 还可以改变一部分值, 例如：

```
> x[c(1,3)]=c(188, 189)
> x
[1] 188 3 189 7
```

1.3 矩阵及其运算

1.3.1 矩阵

R 中的矩阵 (matrix) 类型和向量类似, 是具有相同基本类型 (如整数、浮点数、字符型) 的数据集合, 其中的元素用两个下标访问. 比如, 一个 $m \times n$ 的矩阵 $\boldsymbol{A}$ 可以保存在 R 矩阵变量 A 中, A 的第 i 行第 j 列元素为 A[i,j].

函数 matrix( ) 用来生成矩阵, 其完全格式为

```
matrix(data=NA,nrow=1,ncol=1, byrow=FALSE,dimnames=NULL)
```

其中, 第一自变量 data 为数组的数据向量 (默认值为缺失值 NA), nrow 为行数, ncol 为列数, byrow 的取值表示数据填入是按行次序还是按列次序. 要注意默认情况下为列次序. dimnames 默认是空值, 否则是一个长度为 2 的列表. 列表的第一个成员是长度与行数相等的字符型向量, 表示每行的标签; 列表的第二个成员是长度与列数相等的字符型向量, 表示每列的标签. 例如, 定义一个 3 行 4 列按行由数据 1:12 依次排列的矩阵如下：

```
> A< - matrix(1:12,ncol=4,nrow=3,byrow=TRUE)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

在有数据的情况下, 只需指定行数和列数之一即可. 指定的数据个数少于所需的数据个数时, 循环使用提供的数据. 例如：

```
> B=matrix(c(1,2),ncol=3,nrow=4)
> B
     [, 1] [, 2] [, 3]
[1,]     1     1     1
[2,]     2     2     2
[3,]     1     1     1
[4,]     2     2     2
```

用 c(A) 可以得到将矩阵 A 的元素按列排列的向量, 在线性代数中称为“按列拉直”, 例如：

```
> c(A)
```

```
[1] 1 5 9 2 6 10 3 7 11 4 8 12
```

函数 cbind( ) 把自变量横向拼成一个大矩阵, 函数 rbind( ) 把自变量纵向拼成一个大矩阵, 例如:

```
> x1=rbind(c(1,2),c(3,4))
> x1
     [, 1] [, 2]
[1,]    1    2
[2,]    3    4
> x2=cbind(c(1,2),c(3,4))
> x2
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

1.3.2 访问矩阵元素和子矩阵

要访问矩阵的某个元素, 只要写出矩阵变量名和方括号内分开的下标即可, 如 A[3,2] 为 10. 也可以给一个矩阵元素赋值来修改该元素. 为了访问矩阵 A 的第 i 行, 只要用A[i,] 格式即可; 类似地, 要访问矩阵 A 的第 j 列, 只需用 A[, j] 格式. 为了访问矩阵中由第 $i_1, i_2, \cdots, i_s$ 行和第 $j_1, j_2, \cdots, j_t$ 列交叉而成的子矩阵, 只要用 A[c(i_1,i_2,$\cdots$,i_s),c(j_1,j_2,$\cdots$,j_t)] 格式即可, 例如:

```
> A[1:2,]
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
> A[1:2,c(1,2,4)]
     [,1] [,2] [,3]
[1,]    1    2    4
[2,]    5    6    8
```

矩阵的行、列下标可以用名字代替. 矩阵的每行可以有一个行名, 每列可以有一个列名, 也可以用名字作为下标. 定义行名用函数 rownames( ), 定义列名用函数 colnames( ), 例如:

```
> rownames(A)<- c("a1", "a2", "a3")
> colnames(A)<- paste("x",1:4,sep="")
> A
   x1 x2 x3 x4
a1  1  2  3  4
a2  5  6  7  8
a3  9 10 11 12
> A["a2", "x4"]
[1] 8
```

1.3.3 矩阵运算

矩阵可以进行四则运算 (+,−,*,/, ∧), 解释为矩阵对应的四则运算, 参加运算的矩阵一般应该是相同形状的矩阵. 例如, 假设 A, B, C 是三个相同形状的矩阵, 则

```
>D<- C+2*A/B
```