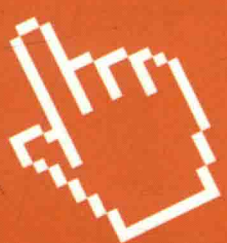
统计软件应用与方法系列丛书

# R语言与应用统计分析实验指导

覃义 南江霞 编

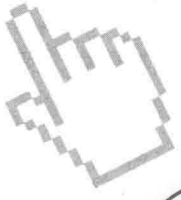


中国统计出版社
China Statistics Press

统计软件应用与方法系列丛书

# R语言与应用统计分析实验指导

覃义 南江霞 编





中国统计出版社
China Statistics Press

图书在版编目（CIP）数据

R 语言与应用统计分析实验指导 / 覃义, 南江霞编.
-- 北京 : 中国统计出版社, 2017.10
ISBN 978-7-5037-8368-5

Ⅰ. ①R… Ⅱ. ①覃… ②南… Ⅲ. ①统计分析－统计程序 Ⅳ. ①C819

中国版本图书馆 CIP 数据核字(2017)第 241054 号

R 语言与应用统计分析实验指导

编　　者/覃　义　南江霞
责任编辑/徐　颖
封面设计/黄俊杰　黄　晨
出版发行/中国统计出版社
通信地址/北京市丰台区西三环南路甲 6 号　邮政编码/100073
电　　话/邮购(010)63376909　书店(010)68783171
网　　址/http://www.zgtjcbs.com/
印　　刷/河北鑫兆源印刷有限公司
经　　销/新华书店
开　　本/ 880mm×1230mm　1/16
字　　数/ 415 千字
印　　张/ 13.5
版　　别/ 2017 年 10 月第 1 版
版　　次/ 2017 年 10 月第 1 次印刷
定　　价/ 48.00 元

# 前　言

在接触R语言之前，我一直用MATLAB，尽管只是初窥门径，但我仍认为，没有MATLAB解决不了的问题（对于不同计算机语言的使用者，会把前面那一句话中的MATLAB换成他们的语言，他们是对的）。一直到2010年，张茂军博士给我推荐了R语言，并把安装包发给了我，如果没有记错的话，当时的版本是2.12.1（其实安装包是可以从官方网站免费获取的）。第一感觉就是：界面和MATLAB很象，但体量要小很多（这个版本只有37M左右，与之相比MATLAB就是一个庞然大物）。但功能一点也不比MATLAB差，关键还是免费的！于是就不可救药的喜欢上了R，开始找一个与R有关的书籍来看，最早是看李洪成翻译的《R语言经典实例》和薛毅编的《R语言实用教程》，到后来，就是方匡南编的《R数据分析：方法与案例详解》，并在自己所教授的统计专业课上，建议我的学生学习并使用R语言。

2016年3月的“人机大战”，AlphaGo在韩国首尔以4:1的总比分战胜了人类的顶尖高手李世石九段；随后，在2017年5月，又在中国浙江乌镇AlphaGo以3:0的绝对优势战胜世界排名第一的柯洁九段。一时之间，大数据和人工智能成为大街小巷热议的话题，成为公众关注的焦点。而与之相对应的开发工具的使用也呈现出上升的趋势，其中又以R语言和Python最为显著。据已公布的数据，2016年PYPL（编程语言流行指数，依据Google上关于语言教程的搜索频率进行统计）R语言排在第9位，在MATLAB之前。而在Indeed（美国最高流量的工作网站之一）上，R语言的使用量排名是第5位，甚至在Python之前，并有继续上升的趋势。

在这个背景之下，我产生了要把我多年来给学生上R语言实验课的讲义整理出版的想法。

关于R的书籍，市面上非常之多，从入门操作到应用实例开发，不一而足。而本书则定位为一本实验教材（当然，R语言的爱好者和自主学习者也可以使用），总共为32个课时，由浅入深以及统计学内容设计了15个常规实验和一个综合实验，每个实验都有一个特定的目的，完成相应的任务即可掌握相应的R的操作。当然，在实际的教学中，也可以根据需要删减掉部分实验，本书假定读者有一定的数据分析与数据挖掘的基础，所以相关的知识，本书做简略的介绍或是不介绍，请读者查阅相关的书籍以获得相关的知识。

本书的主要内容包括：实验1，熟悉R语言的操作环境和基本的使用方法，R语言的数据结构，如，向量、矩阵、数组、列表、数据框的建立和操作，以及数据的读写操作等。实验2，介绍R语言中编写函数的方法，顺序、分支和循环等流程控制的方法。实验3，介绍R语言对数据进行预处理的操作，包括如何发现数据中的缺失值，并如何对缺失值进行插补。实验4，介绍R语言的绘图，包括高、低水平绘图函数及绘图参数的设置。实验5，主要介绍如何用R语言对数据进

行统计性的描述，并介绍如何用 R 实现蒙特卡洛算法计算定积分。实验 6 和实验 7 分别介绍了用 R 语言作假设检验和非参数检验，还包括正态性检验和分布检验。实验 8 和实验 9，主要介绍用 R 内置函数处理回归问题，包括多元线性回归和一元线性、非线性回归。实验 10，主要介绍方差分析，包括单因素和双因素的方差分析。实验 11，主要介绍用 R 语言进行相关性分析。实验 12，主要介绍用 R 语言进行判别分析，包括一次判别和二次判别。实验 13，主要介绍用 R 语言进行聚类分析。实验 14，主要介绍用 R 语言进行主成分分析。实验 15，主要介绍用 R 语言进行关联规则分析。实验 16 是综合实验，主要介绍如何用 R 语言综合解决一个实际问题。

R 语言一个开源工具，是由所有的 R 语言开发者共同维护，R 语言的扩展程序包会在其官方网站（https://www.r-project.org）或者 Github 上找到，本书中某些实例需要下载安装相应的程序包才能够运行。另外，本书的实例都是在 3.3.3 版本下调试运行通过的。

本书的出版得到国家自然科学地区基金项目“违约传染视角下的公司债券定价研究”（项目编号 71461005）、直觉模糊合作博弈的理论及在区域旅游合作中的应用研究：以注北部湾区域为例（项目编号：71561008）、广西高校数据分析与计算重点实验室、桂林电子科技大学统计学专业硕士学位建设经费的资助。此外，本书的完成得到很多人的帮助，在此对他们表示诚挚的感谢！我要感谢张茂军博士从他的项目里给予本书出版的经费支持。张博士具有渊博的学识和广阔的视野，不但把我引入到 R 语言的奇妙世界里，而且在学术上也给予我诸多有益的建议和帮助。我要感谢数学与计算科学学院院长朱志斌教授和书记段复建教授，在他们领导下的数学学院有一个宽松的工作环境和积极向上的氛围，为青年教师的成长和提高提供了许多便利的条件，我是其中的受益者之一。另外，2013 级统计专业学生赵爱萍同学和吴名茜同学、2014 级统计专业学生叶成同学，对本书部分内容的编辑、排版、校对和习题的搜集，做了大量的工作，在此，对他们的辛勤工作表示感谢。中国统计出版社的编辑给予本书很多中肯的修改建议，在此一并谢过。最后，我要特别感谢我的夫人周玲玲女士，感谢她对我工作上的支持和理解，任劳任怨的照顾我的生活，使得本书得以顺利完成；我还要感谢我的儿子覃楚涵，他的欢笑声是我工作中的最好调味品。

限于编者的水平，书中必存在着不足甚至是错误之处，欢迎读者批评指正。

作者的电子邮箱：slqinyi@163.com（覃义）

# 目　录

# 第一章 R语言入门

R语言是主要用于统计分析、绘图的语言和操作环境。R最初是由来自新西兰奥克兰大学的RossIhaka和Robert Gentleman开发，因此称为R。现在有“R开发核心团队”负责开发和维护。R是基于S语言的一个GNU项目，所以也可以当作S语言的一种实现，通常用S语言编写的代码都可以不作修改地在R环境下运行。

# 实验 1 R 语言入门操作

## 1.1 实验目的

1. 掌握 R 语言的基本操作；
2. 掌握 R 语言的基本运算；
3. 掌握 R 语言存取数据的方法。

## 1.2 实验过程

### 1.2.1 赋值操作

#### 1.2.1.1 数值赋值

```
> a <- 100
> a
[1] 100
```

或者

```
> b = 100
> b
[1] 100
```

还可以

```
> 100 -> c
> c
[1] 100
```

以上语句的作用是将 100 赋值给 a 变量，其中“<-”、“->”、“=”都是 R 中的赋值的符号。除此之外，在 R 中，还可以作用函数 assign()来给变量赋值，例如：

```
> (assign("x",100))
[1] 100
```

其中，最外层的括号的作用是为了显示变量 x 的值；命令中的双引号必须在英文状态下输入；在 R 中，是区分大小写的。

我们可以用函数 ls()来查看当前工作空间中的变量：

```
> ls()
[1] "a" "b" "c" "x",
```

#### 1.2.1.2 向量赋值

在 R 语言中，生成向量的方法很多，最简单的方法就是使用连接函数 c() 来生成。例如：把数据 12.1 11.9 12.0 12.3 11.8 赋值给变量 x，可以输入命令：

```
> x <- c(12.1,11.9,12.0,12.3,11.8)
> x
[1] 12.1 11.9 12.0 12.3 11.8
```

函数 c() 不但可以连接数值，还可以连接已有的向量，例如，当输入命令：

```
> y <- c(x,1,x)
```

```
> y
[1] 12.1  11.9  12.0  12.3  11.8  1.0  12.1  11.9  12.0  12.3  11.8
```

如果要生成有一定规律的向量，则要用到 R 中的相应函数。

(1) 步长为 1 的等差数列构成的向量

用符号 ":" 可以产生步为 1 的等差数列，其使用格式为：a:b，表示起始值为 a，步长为 1，终止值为 b 的等差数列，若 b 不在首项为 a 步长 1 的数列里，则产生的数列的末项为小于 b 的最大值，比较以下命令：

```
> (a <- 1 : 10)
 [1]  1  2  3  4  5  6  7  8  9  10
> (b <- 1.1:10)
[1] 1.1  2.1  3.1  4.1  5.1  6.1  7.1  8.1  9.1
> (c <- 10:1)
 [1] 10  9  8  7  6  5  4  3  2  1
```

在这里，符号 ":" 的作用与 matlab 中的作用相同，但要注意的是，该符号并不能指定数列的步长，如果要产生指定步长的等差数列，则要用函数 seq()。

(2)任意步长的等差数列构成的向量

如果要生成任意步长的等差数列，可以用函数 seq()，其基本语法如下：

```
seq(from = 1, to = 1, by = ((to - from)/(length - 1)), length.out = NULL, along.with = NULL, ... )
```

参数说明：

from：　　数值，表示数列的起始值，默认值为 1

to　：　　数值，表示数列的终止值，默认值为 1

by　：　　数值，指定数列的步长

length.out：数值，表示数列的长度

along.with：向量，表示产生的数列与该向量具有相同的长度

注 1.参数 by，length.out，along.with 这三个参数只能选一项。

注 2.当参数 to 指定的终止值不在以 from 为起始值，以 by 指定的步长的数列中的值时，所产生的数列终止值为不超过 to 的最大值。在 R 中输入以下命令，观察其结果。

```
> seq(1,10) #默认的步长是 1
 [1]  1  2  3  4  5  6  7  8  9  10
> seq(10) #默认起始值为 1，步长为 1
 [1]  1  2  3  4  5  6  7  8  9  10
> seq(2,10,by = 2)
[1]  2  4  6  8  10
> seq(1,10,by = 2)
[1]  1  3  5  7  9
> seq(1,10,length.out = 5) #  向量的维数指定为 5
[1]  1.00  3.25  5.50  7.75  10.00
> seq(1,10,along.with = c(1,3,5)) #  向量的维数与向量(1,3,5)的维数相同
[1]  1.0  5.5  10.0
```

(3) 使用重复函数

重复函数 rep()的作用是将变量或是向量复制若干次，函数的调用格式：

```
rep(x, ...)
```

x：　　数量、向量或是数据对象，是要复制的对象

...　　是可选的一些参数，包括

times：　　表示 x 被复制的次数

length.out：　表示复制后输出的向量长度

each：　　表示 x 中每个分量被复制的次数

正整数构成的向量：长度与 x 一致，其分量表示 x 的对应分量被复制的次数

在 R 中输入以下命令，观察其效果。

```
> x = 1 : 4
> rep(x , times = 2)
[1] 1   2   3   4   1   2   3   4
> rep(x , length.out = 10)
 [1] 1   2   3   4   1   2   3   4   1   2
> rep(x , each = 2)
[1] 1   1   2   2   3   3   4   4
> rep(x , c(1 : 4))
 [1] 1   2   2   3   3   3   4   4   4   4
```

对于已构建好的向量 x ，如果想要查看向量 x 中的元素，可以输入 x[i]，其中的索引 i 可以是数字，也可以是向量，输入以下命令，观察其效果。

```
> x <- 1 : 10
> x[1]
[1] 1
> x[1 : 3]
[1] 1   2   3
> x[c(1,4,7)]
[1] 1   4   7
```

注 1：在 R 中引用向量时，用的是[ ]，而 matlab 等编程语言用的是( )，不要混淆。

注 2：在 R 中向量的下标是从 1 开始的，而 C 等编程语言是从 0 开始的，不要混淆。

#### 1.2.1.3 删除和修改元素

如果要删除向量 x 中的元素，可以输入 x[ - i ] ,其中的索引 i 可以是数字，也可以是向量，输入以下命令，观察其效果。

```
> x <- 1 : 10
> x[ - 3 ]
[1]  1   2   4   5   6   7   8   9   10
> x[ - (1 : 4) ]
[1]  5   6   7   8   9   10
> x[ - c(3,6,9) ]
[1]  1   2   4   5   7   8   10
```

注意，以上操作并未真正从变量 x 中删除掉相应的元素，要达到删除的目的，可以把以上操作的结果赋值给其它变量。如果要修改向量中的某些元素，可以直接给相应的分量赋值，输入以下命令，观察其结果。

```
> x[ 3 ] <- 18
> x
```

```
 [1]  1  2  18  4  5  6  7  8  9  10
> x[ 1 : 3 ] <- c(10,20,30)
> x
 [1] 10  20  30  4  5  6  7  8  9  10
> x[ c(3,6,9) ] <- c(12,24,36)
> x
 [1] 10  20  12  4  5  24  7  8  36  10
```

### 1.2.2 算术操作

R 语言提供了方便的算术运算的操作，包括+、-、*、/等，这些运算与 matlab 的运算含义完全一样，都可以对向量进行运算。下面是一些 R 中特有的运算：

(1)求余：%%

如：

```
> 9 %% 4
[1] 1
> (1 : 10) %% 2
[1] 1  0  1  0  1  0  1  0  1  0
```

(2)求商：%/%

如：

```
> 9 %/% 4
[1] 2
> (1 : 10) %/% 2
[1] 0  1  1  2  2  3  3  4  4  5
```

(3)乘方：^

如：

```
> 2 ^ 3
[1] 8
> (1 : 10)^3
[1]    1    8   27   64  125  216  343  512  729  1000
```

(4)开方：sqrt()

如：

```
> sqrt(9)
[1] 3
> sqrt(1 : 10)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
3.000000 3.162278
```

注：函数 sqrt()只能是在实数范围内开平方，如果输入的参数是负数，则会输出 NaN（NaN 是 Not a Number 的缩写，表示不确定），并提示警告信息。输入以下命令，观察其结果。

```
> sqrt(- 9)
[1] NaN
Warning message:
In sqrt(-9) : 产生了 NaNs
```

(5)对数及以 e 为底的指数：log()；log2()；log10()；exp()；

①log()有两个参数，一个为 x，一个是 base，即底数。如果要求以 2 为底数，16 的对数，可以如下操作：

```
> log(16,2)
[1] 4
```

或者

```
> log(x = 16, base = 2)
[1] 4
> log(x = c(2,4,8,16),base = 2)
[1] 1  2  3  4
```

②log2()和 log10()均只有一个参数，即 x，他们分别是以 2 为底和以 10 为底取对数。

③exp()是计算以 e 为底的指数，只有一个参数 x，其作用是计算 e ^ x 如：

```
> exp(1)
[1] 2.718282
> exp(1:3)
[1]  2.718282  7.389056  20.085537
```

在 R 语言中，可以完成各种初等函数的运算，如对数、指数、三角函数和反三角函数以及其他的运算，表 1.1 列出了 R 语言中常用的函数。

表1.1 R语言中的数学函数

| | |
|---|---|
| sqrt | 开平方函数 |
| abs | 绝对值函数 |
| exp | 2.71828..... |
| expm1 | 当 x 的绝对值比 1 小很多的时候，它将能更加正确的计算 exp(x)-1 |
| log | 对数函数 |
| log10 | 对数(底为 10)函数 |
| log2 | 对数(底为 2)函数 |
| sin | 正弦函数 |
| cos | 余弦函数 |
| tan | 正切函数 |
| asin | 反正弦函数 |
| acos | 反余弦函数 |
| atan | 反正切函数 |
| sinh | 双曲正弦函数 |
| cosh | 双曲余弦函数 |
| tanh | 双曲正切函数 |
| asinh | 反双曲正弦函数 |
| acosh | 反双曲余弦函数 |
| atanh | 反双曲正切函数 |
| logb | 和 log 函数一样 |
| log1px | 当 x 的绝对值比 1 小很多的时候，它将能更加正确的计算 log(1+x) |
| gamma | Γ 函数(伽玛函数) |
| lgamma | 等同于 log(gamma(x)) |
| ceiling | 返回大于或等于所给数字表达式的最小整数 |
| floor | 返回小于或等于所给数字表达式的最大整数 |
| trunc | 截取整数部分 |
| round | 四舍五入 |
| signif(x,a) | 数据截取函数，x:有效位，a:到 a 位为止 |

### 1.2.3 比较运算符及逻辑操作

除了以上的四则运算和函数运算之外，各向量之间还可以进行比较运算，其返回值为真(TRUE)或假(FALSE)，比较运算符包括：

| | | | |
|---|---|---|---|
| > | 大于； | >= | 大于等于； |
| < | 小于； | <= | 小于等于； |
| == | 等于； | != | 不等于。 |

如：

```
> 3 == 5
[1] FALSE
```

需要注意的是，当进行比较运算的两个向量的长度不一样长时，R 会将向量补齐，然后再进行比较，输入以下命令，观察其运行效果。

```
> a <- 1:10
> b = c(1,3,5)
> a == b
 [1]  TRUE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
Warning message:
In a == b ：长的对象长度不是短的对象长度的整倍数
```

其补齐的方式是：较短的向量自我复制，直到与较长的向量的长度一致；当较长向量的长度不是较短向量长度的整数倍时，会返回警告信息，但这不会影响运算的结果。在上例中，向量 b 自我复制的结果是：b <- c(1,3,5,1,3,5,1,3,5,1)。

逻辑操作的对象是逻辑值，在 R 语言的逻辑操作包括以下几项：

(1)!

感叹号表示“取非”。

如：

```
> x <- TRUE
> ! x
[1] FALSE
```

又如：

```
> x <- c(T,T,F,T,F)
> ! x
[1] FALSE  FALSE  TRUE FALSE  TRUE
```

(2)&和&&

这两者都是逻辑“与”，操作都是 x & y 和 x && y。

如：

```
> x <- c(T,T,F)
> y <- c(F,T,F)
> x && y
[1]FALSE
> x & y
[1]FALSE  TRUE  FALSE
```

可以看到，&是对每一个元素一一求与，而&&是所有元素求与操作。

(3)|和||

这两者的使用与前者类似：

```
> x <- c(T,T,F)
> y <- c(F,T,F)
> x | y
[1]TRUE   TRUE   FALSE
> x || y
[1]TRUE
```

(4)异或操作 xor()

xor()为异或操作函数，当两个数值相等时，结果为假；当两个数值不等时，结果为真，如

```
> xor(0,1)
[1] TRUE
> x <- c(T,T,F)
> y <- c(F,T,F)
> xor(x,y)
[1]   TRUE   FALSE   FALSE
```

(5) all()和 any()

判断数据中是否存在 TRUE 值，其中 all()是在全部为 TURE 时返回 TURE，any()是在存在任何一个 TRUE 时返回 TRUE，他们都还有另外一个参数，即是否删除 NA 值，即 not available 值：na.rm，其调用格式为：

```
all(x, na.rm = F)
any(x,na.rm = F)
```

当 na.rm = F 时，在逻辑操作过程中不考虑 x 中的 NA 值；否则，在进行逻辑操作过程中考虑 x 中的 NA 值。在 R 中输入以下命令，观察其运行的效果。

```
> x <- c(1,3,5,7,9,NA)
> all(x > 5,na.rm = T)
[1] FALSE
> any(x > 5, na.rm = F)
[1] TRUE
```

另外，如果想判断一个逻辑向量中哪些元素为真，可以用函数 which()，如

```
> which(x > 5)
[1] 4   5
```

若要构造一个初始逻辑向量，可以用函数 logical()，如

```
> logical(5)
[1] FALSE   FALSE   FALSE   FALSE   FALSE
```

### 1.2.4 R 语言的数据结构

#### 1.2.4.1 数组(向量和矩阵)

数组是带有多个下标的且类型相同的元素构成的集合，可以看成是向量和矩阵的推广，可用来储存数值型(numeric)、逻辑型(logical)和字符型(character)三种类型的数据。可以使用函数 array()来生成一个数组，其调用格式为：

```
array(data = NA, dim = length(data), dimnames = NULL)
```

其中：

data：　　保存到数组的数据，可以是空值

dim：　　指定数组的维数

dimnames：　指定数组各维度的名称，默认为整数

```
> (x <- array(1:12,dim = c(3,4))) # 数值型
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> x = array(rep(T,6),dim = c(2,3)) # 逻辑型
> x
```

结果如下：

```
     [,1]  [,2]  [,3]
[1,] TRUE  TRUE  TRUE
[2,] TRUE  TRUE  TRUE
> x = array(rep("a",6),dim = c(2,3))#字符型
> x
```

结果如下：

```
     [,1] [,2] [,3]
[1,] "a"  "a"  "a"
[2,] "a"  "a"  "a"
```

以下的例子展示了如何修改各维度的名称，输入以下命令，观察其运行的效果。

```
> x = array(rpois(6,10),dim = c(2,3),dimnames = list(c("male","female"),c("apple","banana","pear")));x
```

结果如下：

```
       apple banana pear
male      13     12   10
female    11     10    3
```

这里，我们使用了 rpois()来产生了 6 个符合泊松分布的数字。

当然，我们的数组不局限于二维，可以是三维或者更多维。

```
> x = array(rpois(24,10),dim = c(2,3,4),dimnames = list(c("male","female"),c("apple","banana","pear"),
c("Mon","Tue","Wed","Thu")));x
```

结果如下：

```
, , Mon

       apple banana pear
male       8     14   10
female    10     12   10

, , Tue

       apple banana pear
male       8      5    7
female     8      8    6

, , Wed
```

```
          apple banana pear
male         13     13    8
female       12     12    6

, , Thu

          apple banana pear
male          7     17   17
female       15     13    3
```

上面这个例子是一个三维的数组。我们继续来看这个 dimnames。

dimnames 这个参数是用来指定有关的每一个维度的名字的，其中第一个维度的两行分别为 male 和 female，第二个维度的两个分别为 apple，banana 和 pear。

需要注意的是，一维数组跟向量很相近，二维数组就是矩阵。但是一维数组跟向量在某些函数的处理过程中会有区别的对待，如 str()函数：

```
> x = array(rpois(2,10),dim = c(1,2),dimnames = list(c("apple"),c("male","female")))
> str(x)
int [1, 1:2] 16 9
 - attr(*, "dimnames")=List of 2
  ..$ : chr "apple"
  ..$ : chr [1:2] "male" "female"
```

在这里我们用到了函数 list()，其作用是生成一个列表，在后面我们会做详细的介绍。对于数组的引用，可用 A[i,j,...]，输入以下命令，观察其运行效果。

```
> A <- array(data = 1:36,dim = c(3,3,4)) # 产生 3*3*4 维的数组
> A[1,2,3] # 引用第 3 层的第 1 行第 2 列的元素
[1] 22
> A[1,c(1,2),3]
[1] 19 22
> A[1,,3]
[1] 19 22 25
> A[1,,]
     [,1] [,2] [,3] [,4]
[1,]    1   10   19   28
[2,]    4   13   22   31
[3,]    7   16   25   34
```

下面来看看如何定义向量和矩阵。

(1) 向量

在前面我们已经介绍了生成向量的一些方法，而在 R 中可以用函数 vector()来生成一个空向量，其调用格式为：

```
vector(mode = "logical",length = 0)
```

其中

mode：表示的是该向量储存数据的类型，可取的值有："logical"，"integer"，"numeric"，"complex"，"character" 及 "raw"，默认值是"logical"

length：表示该向量的长度，默认值为 0

```
> (x <- vector(mode = "numeric", length = 5))
[1] 0  0  0  0  0
```

与向量相关的函数还有：

as.vector()：将其它类型的对象强制转换成向量

is.vector()：判断对象是否为向量

(2)矩阵

矩阵是一个二维的特殊数组，在 R 中可以用函数 matrix()来生成矩阵，其调用格式为：

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

其中：

data：　　表示该向量要储存的数据

nrow 和 ncol：表示矩阵的行数和列数，注意 data 的个数等于 nrow 乘以 ncol

byrow：　　表示数据是否以行的方式进行排列

dimnames：　指定矩阵各维度的名称，默认为整数

```
> x = matrix(data = 1:6,nrow = 2,ncol = 3,byrow = T); x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> x = matrix(data = 1:6,nrow = 2,ncol = 3,byrow = F);x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

对于矩阵的四则运算(+，-，*，、/)，是对其中对应元素的四则运算，如果要实现线性代数中的矩阵的乘法，则要用%*%，输入以下命令，比较其运算结果：

```
> B <- A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> A*B
     [,1] [,2] [,3]
[1,]    1   16   49
[2,]    4   25   64
[3,]    9   36   81
> A%*%B
     [,1] [,2] [,3]
[1,]   30   66  102
[2,]   36   81  126
[3,]   42   96  150
```

如果要求矩阵的转置，可以使用函数 t()，如：

```
> t(A)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

如果要求矩阵的逆矩阵，可以使用函数 solve()，如：

```
> A <- matrix(data = 1:4, nrow = 2, ncol = 2)
> solve(A)
```