Contents

基本数据管理

在前面的章节中，我们讨论了多种导入数据到 R 中的方法。遗憾的是，将你的数据表示为矩阵或数据框这样的矩形形式仅仅是数据准备的第一步。在实际工作中，有多达 60% 的数据分析时间都花在了实际分析前数据的准备上。我敢大胆地说，多数需要处理现实数据的分析师可能都面临着以某种形式存在的类似问题。让我们先看一个例子。

一个实例

假如我们研究主题之一是男性和女性在领导各自企业方式上的不同。典型的问题如下。

- 处于管理岗位的男性和女性在听从上级的程度上是否有所不同？
- 这种情况是否依国家的不同而有所不同，或者说这些由性别导致的不同是否普遍存在？

解答这些问题的一种方法是让多个国家的经理人的上司对其服从程度打分，使用的问题类似于：

这名经理在作出人事决策之前是否会询问我的意见

1	2	3	4	5
非常不同意	不同意	既不同意也不反对	同意	非常同意

结果数据可能类似于下表。各行数据代表了某个经理人的上司对他的评分。

领导行为的性别差异

经理人	日期	国籍	性别	年龄	q1	q2	q3	q4	q5
1	10/24/14	US	M	32	5	4	5	5	5
2	10/28/14	US	F	45	3	5	2	5	5
3	10/01/14	UK	F	25	3	5	5	5	2
4	10/12/14	UK	M	39	3	3	4		
5	05/01/14	UK	F	99	2	2	1	2	1

在这里，每位经理人的上司根据与服从权威相关的五项陈述（q1 到 q5）对经理人进行评分。例如，经理人 1 是一位在美国工作的 32 岁男性，上司对他的评价是惯于顺从，而经理人 5 是一位在英国工作的，年龄未知（99 可能代表缺失）的女性，服从程度评分较低。日期一栏记录了进行评分的时间。一个数据集中可能含有几十个变量和成千上万的观测，但为了简化示例，我们仅选取了 5 行 10 列的

数据。另外，我们已将关于经理人服从行为的问题数量限制为 5。在现实的研究中，你很可能会使用 10 到 20 个类似的问题来提高结果的可靠性和有效性。

示例代码：创建 leadership 数据框

```
manager <- c(1, 2, 3, 4, 5)
date <- c("10/24/08", "10/28/08", "10/1/08", "10/12/08", "5/1/09")
country <- c("US", "US", "UK", "UK", "UK")
gender <- c("M", "F", "F", "M", "F")
age <- c(32, 45, 25, 39, 99)
q1 <- c(5, 3, 3, 3, 2)
q2 <- c(4, 5, 5, 3, 2)
q3 <- c(5, 2, 5, 4, 1)
q4 <- c(5, 5, 5, NA, 2)
q5 <- c(5, 5, 2, NA, 1)
leadership <- data.frame(manager, date, country, gender, age,q1, q2, q3, q4, q5, stringsA
```

为了解决感兴趣的问题，你必须首先解决一些数据管理方面的问题。这里列出其中一部分。

- 五个评分（q1 到 q5）需要组合起来，即为每位经理人生成一个平均服从程度得分。
- 在问卷调查中，被调查者经常会跳过某些问题。例如，为 4 号经理人打分的上司跳过了问题 4 和问题 5。你需要一种处理不完整数据的方法，同时也需要将 99 岁这样的年龄值重编码为缺失值。
- 一个数据集中也许会有数百个变量，但你可能仅对其中的一些感兴趣。为了简化问题，我们往往希望创建一个只包含那些感兴趣变量的数据集。
- 既往研究表明，领导行为可能随经理人的年龄而改变，二者存在函数关系。要检验这种观点，你希望将当前的年龄值重编码为类别型的年龄组（例如年轻、中年、年长）。
- 领导行为可能随时间推移而发生改变。你可能想重点研究最近全球金融危机期间的服从行为。为了做到这一点，你希望将研究范围限定在某一个特定时间段收集的数据上（比如，2009 年 1 月 1 日到 2009 年 12 月 31 日）。

我们将在本章中逐个解决这些问题，同时完成如数据集的组合与排序这样的基本数据管理任务。

创建新变量

有三种方法在数据框中创建新的变量，如下代码所示：

```
#1
mydata<-data.frame(x1 = c(2, 2, 6, 4),x2 = c(3, 4, 2, 8))
mydata$sumx <- mydata$x1 + mydata$x2
mydata$meanx <- (mydata$x1 + mydata$x2)/2
#2
attach(mydata)
mydata$sumx <- x1 + x2
mydata$meanx <- (x1 + x2)/2
detach(mydata)
#3
mydata <- transform(mydata,sumx = x1 + x2,meanx = (x1 + x2)/2)
```

变量的重编码

重编码涉及根据同一个变量和/或其他变量的现有值创建新值的过程。举例来说，你可能想：

- 将一个连续型变量修改为一组类别值；
- 将误编码的值替换为正确值；
- 基于一组分数线创建一个表示及格/不及格的变量。

不妨假设你希望将 leadership 数据集中经理人的连续型年龄变量 age 重编码为类别型变量 agecat（Young、Middle Aged、Elder）。首先，必须将 99 岁的年龄值重编码为缺失值，使用的代码为：

```
leadership$age[leadership$age == 99] <- NA
```

在指定好年龄中的缺失值后，你可以接着使用以下代码创建 agecat 变量：

```
leadership$agecat[leadership$age > 75] <- "Elder"
leadership$agecat[leadership$age >= 55 & leadership$age <= 75] <- "Middle Aged"
leadership$agecat[leadership$age < 55] <- "Young"
```

这段代码可以写成更紧凑的：

```
leadership <- within(leadership,{
agecat <- NA
agecat[age > 75] <- "Elder"
agecat[age >= 55 & age <= 75] <- "Middle Aged"
agecat[age < 55] <- "Young" })
```

看一看经过上述处理后的数据框是什么样子：

```
knitr::kable(leadership)
```

manager	date	country	gender	age	q1	q2	q3	q4	q5	agecat
1	10/24/08	US	M	32	5	4	5	5	5	Young
2	10/28/08	US	F	45	3	5	2	5	5	Young
3	10/1/08	UK	F	25	3	5	5	5	2	Young
4	10/12/08	UK	M	39	3	3	4	NA	NA	Young
5	5/1/09	UK	F	NA	2	2	1	2	1	NA

变量的重命名

如果对现有的变量名称不满意，你可以交互地或者以编程的方式修改它们。假设你希望将变量名 manager 修改为 managerID ，并将 date 修改为 testDate ，那么可以使用语句：

fix(leadership)

来调用一个交互式的编辑器。然后你单击变量名，然后在弹出的对话框中将其重命名。若以编程方式，可以通过 names() 函数来重命名变量。例如：

```
names(leadership)
```

```
##  [1] "manager" "date"    "country" "gender"  "age"     "q1"      "q2"
##  [8] "q3"      "q4"      "q5"      "agecat"
```

```
names(leadership)[2] <- "testDate"
knitr::kable(leadership)
```

manager	testDate	country	gender	age	q1	q2	q3	q4	q5	agecat
1	10/24/08	US	M	32	5	4	5	5	5	Young
2	10/28/08	US	F	45	3	5	2	5	5	Young
3	10/1/08	UK	F	25	3	5	5	5	2	Young
4	10/12/08	UK	M	39	3	3	4	NA	NA	Young
5	5/1/09	UK	F	NA	2	2	1	2	1	NA

以类似的方式，重命名 q1 到 q5 为 item1 到 item5 。

```
names(leadership)[6:10] <- c("item1", "item2", "item3", "item4","item5")
knitr::kable(leadership)
```

manager	testDate	country	gender	age	item1	item2	item3	item4	item5	agecat
1	10/24/08	US	M	32	5	4	5	5	5	Young
2	10/28/08	US	F	45	3	5	2	5	5	Young
3	10/1/08	UK	F	25	3	5	5	5	2	Young
4	10/12/08	UK	M	39	3	3	4	NA	NA	Young
5	5/1/09	UK	F	NA	2	2	1	2	1	NA

缺失值的处理

在任何规模的项目中，数据都可能由于未作答问题、设备故障或误编码数据的缘故而不完整。在 R 中，缺失值以符号 NA （Not Available，不可用）表示。R 提供了一些函数，用于识别包含缺失值的观测。函数 is.na() 允许你检测缺失值是否存在。

```
y <- c(1, 2, 3, NA)
is.na(y)
```

```
## [1] FALSE FALSE FALSE  TRUE
```

注意 is.na() 函数是如何作用于一个对象上的。它将返回一个相同大小的对象，如果某个元素是缺失值，相应的位置将被改写为 TRUE，不是缺失值的位置则为 FALSE 。以下代码将此函数应用到了我们的 leadership 数据集上。

```
is.na(leadership[,6:10])
```

```
##       item1 item2 item3 item4 item5
## [1,] FALSE FALSE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE FALSE
## [3,] FALSE FALSE FALSE FALSE FALSE
## [4,] FALSE FALSE FALSE  TRUE  TRUE
## [5,] FALSE FALSE FALSE FALSE FALSE
```

这里的 leadership[,6:10] 将数据框限定到第 6 列至第 10 列，接下来 is.na() 识别出了缺失值。

重编码某些值为缺失值

```
leadership$age[leadership$age == 99] <- NA
```

在分析中排除缺失值

以下代码计算包含缺失值的项，结果将不正确：

```
x <- c(1, 2, NA, 3)
y <- x[1] + x[2] + x[3] + x[4]
z <- sum(x)
z
```

```
## [1] NA
```

好在多数的数值函数都拥有一个 na.rm=TRUE 选项，可以在计算之前移除缺失值并使用剩余值进行计算：

```
x <- c(1, 2, NA, 3)
y <- sum(x, na.rm=TRUE)
y
```

```
## [1] 6
```

通过函数 na.omit() 移除所有含有缺失值的观测。na.omit() 可以删除所有含有缺失数据的行。在以下代码中，我们将此函数应用到了 leadership 数据集上。

```
leadership
```

```
##   manager testDate country gender age item1 item2 item3 item4 item5 agecat
## 1       1 10/24/08      US      M  32     5     4     5     5     5  Young
## 2       2 10/28/08      US      F  45     3     5     2     5     5  Young
## 3       3  10/1/08      UK      F  25     3     5     5     5     2  Young
## 4       4 10/12/08      UK      M  39     3     3     4    NA    NA  Young
## 5       5   5/1/09      UK      F  NA     2     2     1     2     1   <NA>
```

```
newdata <- na.omit(leadership)
newdata
```

```
##   manager testDate country gender age item1 item2 item3 item4 item5 agecat
## 1       1 10/24/08      US      M  32     5     4     5     5     5  Young
## 2       2 10/28/08      US      F  45     3     5     2     5     5  Young
## 3       3  10/1/08      UK      F  25     3     5     5     5     2  Young
```

在结果被保存到 newdata 之前，所有包含缺失数据的行均已从 leadership 中删除。

日期值

日期值通常以字符串的形式输入到 R 中，然后转化为以数值形式存储的日期变量。函数 as.Date() 用于执行这种转化。其语法为 as.Date(x, “input_format”) ，其中 x 是字符型数据，input_format 则给出了用于读入日期的适当格式（见下表）。

符号	含义	示例
%d	数字表示的日期（0~31）	01~31
%a	缩写的星期名	Mon
%A	非缩写星期名	Monday
%m	月份（00~12）	00~12
%b	缩写的月份	Jan
%B	非缩写月份	January
%y	两位数的年份	07
%Y	四位数的年份	2007

日期值的默认输入格式为 yyyy-mm-dd。语句：

mydates <- as.Date(c(“2007-06-22”, “2004-02-13”))

将默认格式的字符型数据转换为了对应日期。相反，

strDates <- c(“01/05/1965”, “08/16/1975”)

dates <- as.Date(strDates, “%m/%d/%Y”)

则使用 mm/dd/yyyy 的格式读取数据。

在 leadership 数据集中，日期是以 mm/dd/yy 的格式编码为字符型变量的。因此下面的代码使用指定格式读取字符型变量，并将其作为一个日期变量替换到数据框中。

```
myformat <- "%m/%d/%y"
leadership$testDate <- as.Date(leadership$testDate, myformat)
```

有两个函数对于处理时间戳数据特别实用。Sys.Date() 可以返回当天的日期，而 date() 则返回当前的日期和时间。

```
Sys.Date()
```

```
## [1] "2017-04-11"
```

```
date()
```

```
## [1] "Tue Apr 11 09:05:09 2017"
```

你可以使用函数 format(x, format="output_format") 来输出指定格式的日期值，并且可以提取日期值中的某些部分：

```
today <- Sys.Date()
format(today, format="%B %d %Y")
```

```
## [1] "四 月 11 2017"
```

```
format(today, format="%A")
```

```
## [1] "星 期 二"
```

可以用日期相减的方式或使用函数 difftime() 来计算时间间隔，并以星期、天、时、分、秒来表示。假设我出生于 1970 年 10 月 12 日，我现在有多大呢？

```
today <- Sys.Date()
me <- as.Date("1970-10-12")
(today-me)
```

```
## Time difference of 16983 days
```

```
difftime(today, me, units="weeks")
```

```
## Time difference of 2426.143 weeks
```

类型转换

类型转换函数

判断	转换
is.numeric()	as.numeric()
is.character()	as.character()
is.vector()	as.vector()
is.matrix()	as.matrix()
is.data.frame()	as.data.frame()
is.factor()	as.factor()
is.logical()	as.logical()

名为 is.datatype() 这样的函数返回 TRUE 或 FALSE ，而 as.datatype() 这样的函数则将其参数转换为对应的类型。

数据排序

有些情况下，查看排序后的数据集可以获得相当多的信息。例如，哪些经理人最具服从意识？在 R 中，可以使用 order() 函数对一个数据框进行排序。默认的排序顺序是升序。在排序变量的前边加一个减号即可得到降序的排序结果。以下示例使用 leadership 演示了数据框的排序。

```
newdata <- leadership[order(leadership$age),]
newdata
```

```
##   manager   testDate country gender age item1 item2 item3 item4 item5
```

```
## 3        3 2008-10-01       UK       F  25      3      5      5      5      2
## 1        1 2008-10-24       US       M  32      5      4      5      5      5
## 4        4 2008-10-12       UK       M  39      3      3      4     NA     NA
## 2        2 2008-10-28       US       F  45      3      5      2      5      5
## 5        5 2009-05-01       UK       F  NA      2      2      1      2      1
##    agecat
## 3   Young
## 1   Young
## 4   Young
## 2   Young
## 5    <NA>
```

创建了一个新的数据集，其中各行依经理人的年龄升序排序。

下面，将各行依女性到男性、同样性别中按年龄升序排序。

```
attach(leadership)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     age, country, gender, manager
```

```
newdata <- leadership[order(gender, age),]
detach(leadership)
```

将各行依经理人的性别和年龄降序排序。

```
attach(leadership)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     age, country, gender, manager
```

```
newdata <-leadership[order(gender, -age),]
detach(leadership)
```

数据集的合并

要横向合并两个数据框（数据集），请使用 merge() 函数。在多数情况下，两个数据框是通过一个或多个共有变量进行联结的（即一种内联结，inner join）。例如：

total <- merge(dataframeA, dataframeB, by="ID")

将 dataframeA 和 dataframeB 按照 ID 进行了合并。类似地，

total <- merge(dataframeA, dataframeB, by=c("ID","Country"))

将两个数据框按照 ID 和 Country 进行了合并。类似的横向联结通常用于向数据框中添加变量。

要纵向合并两个数据框（数据集），请使用 rbind() 函数：

total <- rbind(dataframeA, dataframeB)

两个数据框必须拥有相同的变量，不过它们的顺序不必一定相同。如果 dataframeA 中拥有 dataframeB 中没有的变量，请在合并它们之前做以下某种处理：

- 删除 dataframeA 中的多余变量；
- 在 dataframeB 中创建追加的变量并将其值设为 NA（缺失）。

纵向联结通常用于向数据框中添加观测。

数据集取子集

R 拥有强大的索引特性，可以用于访问对象中的元素。也可利用这些特性对变量或观测进行选入和排除。

选入（保留）变量

```
newdata <- leadership[, c(6:10)]
newdata
```

```
##   item1 item2 item3 item4 item5
## 1     5     4     5     5     5
## 2     3     5     2     5     5
## 3     3     5     5     5     2
## 4     3     3     4    NA    NA
## 5     2     2     1     2     1
```

从 leadership 数据框中选择了变量 q1 、q2 、q3 、q4 和 q5 ，并将它们保存到了数据框 newdata 中。将行下标留空（,）表示默认选择所有行。

也可以：

```
myvars <- c("item1", "item2", "item3", "item4", "item5")
newdata <-leadership[myvars]
```

其实你可以写：

```
myvars <- paste("item", 1:5, sep="")
newdata <- leadership[myvars]
```

剔除（丢弃）变量

剔除变量的原因有很多。举例来说，如果某个变量中有很多缺失值，你可能就想在进一步分析之前将其丢弃。下面是一些剔除变量的方法。

```
myvars <- names(leadership) %in% c("item3", "item4")
newdata <- leadership[!myvars]
newdata
```

```
##   manager   testDate country gender age item1 item2 item5 agecat
## 1       1 2008-10-24      US      M  32     5     4     5  Young
## 2       2 2008-10-28      US      F  45     3     5     5  Young
## 3       3 2008-10-01      UK      F  25     3     5     2  Young
## 4       4 2008-10-12      UK      M  39     3     3    NA  Young
## 5       5 2009-05-01      UK      F  NA     2     2     1   <NA>
```

选入观测

选入或剔除观测（行）通常是成功的数据准备和数据分析的一个关键方面。

```
newdata <- leadership[1:3,]
newdata <- leadership[leadership$gender=="M" & leadership$age > 30,]
attach(leadership)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     age, country, gender, manager
```

```
newdata <- leadership[gender=='M' & age > 30,]
detach(leadership)
newdata
```

```
##   manager   testDate country gender age item1 item2 item3 item4 item5
## 1       1 2008-10-24      US      M  32     5     4     5     5     5
## 4       4 2008-10-12      UK      M  39     3     3     4    NA    NA
##   agecat
## 1  Young
## 4  Young
```

你可能希望将研究范围限定在 2009 年 1 月 1 日到 2009 年 12 月 31 日之间收集的观测上。怎么做呢？

```
leadership$date <- as.Date(leadership$testDate, "%m/%d/%y")
startdate <- as.Date("2009-01-01")
enddate <- as.Date("2009-10-31")
newdata <- leadership[which(leadership$testDate >= startdate &leadership$date <= enddate)
newdata
```

```
##   manager   testDate country gender age item1 item2 item3 item4 item5
## 5       5 2009-05-01      UK      F  NA     2     2     1     2     1
##   agecat       date
## 5   <NA> 2009-05-01
```

subset() 函数

使用 subset() 函数大概是选择变量和观测最简单的方法了。

```
newdata <- subset(leadership, age >= 35 | age < 24,
select=c(item1, item2, item3, item4))
newdata
```

```
##   item1 item2 item3 item4
## 2     3     5     2     5
## 4     3     3     4    NA
```

```
newdata <- subset(leadership, gender=="M" & age > 25,
select=gender:item4)
newdata
```

```
##   gender age item1 item2 item3 item4
## 1      M  32     5     4     5     5
## 4      M  39     3     3     4    NA
```

随机抽样

sample() 函数能够让你从数据集中（有放回或无放回地）抽取大小为 n 的一个随机样本。你可以使用以下语句从 leadership 数据集中随机抽取一个大小为 3 的样本：

```
mysample <- leadership[sample(1:nrow(leadership), 3, replace=FALSE),]
mysample
```

```
##   manager   testDate country gender age item1 item2 item3 item4 item5
## 1       1 2008-10-24      US      M  32     5     4     5     5     5
## 3       3 2008-10-01      UK      F  25     3     5     5     5     2
## 2       2 2008-10-28      US      F  45     3     5     2     5     5
##   agecat       date
```

```
## 1  Young 2008-10-24
## 3  Young 2008-10-01
## 2  Young 2008-10-28
```

sample() 函数中的第一个参数是一个由要从中抽样的元素组成的向量。在这里，这个向量是 1 到数据框中观测的数量，第二个参数是要抽取的元素数量，第三个参数表示无放回抽样。sample() 函数会返回随机抽样得到的元素，之后即可用于选择数据框中的行。

小结

本章讲解了大量的基础知识。首先我们看到了 R 存储缺失值和日期值的方式，并探索了它们的多种处理方法。接着学习了如何确定一个对象的数据类型，以及如何将它转换为其他类型。还使用简单的公式创建了新变量并重编码了现有变量。你学习了如何对数据进行排序和对变量进行重命名，学习了如何对数据和其他数据集进行横向合并（添加变量）和纵向合并（添加观测）。最后，我们讨论了如何保留或丢弃变量，以及如何基于一系列的准则选取观测。