
南開大學

# 第三讲 数据处理函数及程序设计

# 提　纲



# 1. 数据处理

【问题引入】

假设一组学生参加了数学、科学和英语考试。为了给所有学生确定一个单一的成绩衡量指标，需要将这些科目的成绩组合起来。另外，还需将前20%的学生评定为A，接下来20%的学生评定为B，依次类推。最后，按姓名的字母顺序对学生排序。

# 1. 数据处理

| 学生姓名 | 数　学 | 科　学 | 英　语 |
|---|---|---|---|
| John Davis | 502 | 95 | 25 |
| Angela Williams | 600 | 99 | 22 |
| Bullwinkle Moose | 412 | 80 | 18 |
| David Jones | 358 | 82 | 15 |
| Janice Markhammer | 495 | 75 | 20 |
| Cheryl Cushing | 512 | 85 | 28 |
| Reuven Ytzrhak | 410 | 80 | 15 |
| Greg Knox | 625 | 95 | 30 |
| Joel England | 573 | 89 | 27 |
| Mary Rayburn | 522 | 86 | 18 |

# 1. 数据处理

此数据集有两个明显特征：

- 三科考试的成绩是无法比较的。由于它们的均值和标准差相去甚远，所以对它们求平均值是没有意义的。在组合这些考试成绩之前，必须将其变换为可比较的单元。
- 为了评定等级，需要一种方法来确定某个学生在前述得分上百分比排名。
- 表示姓名的字段只有一个，这让排序任务复杂化了。为了正确地将其排序，需要将姓和名拆开。

# 1. 数据处理

- 数学函数

| 函　数 | 描　述 |
| --- | --- |
| abs(x) | 绝对值<br>abs(-4)返回值为4 |
| sqrt(x) | 平方根sqrt(25)返回值为5<br>和25^(0.5)等价 |
| ceiling(x) | 不小于x的最小整数<br>ceiling(3.475)返回值为4 |
| floor(x) | 不大于x的最大整数<br>floor(3.475)返回值为3 |
| trunc(x) | 向 0 的方向截取的x中的整数部分<br>trunc(5.99)返回值为5 |

当这些函数被应用于数值向量、矩阵或数据框时，它们会作用于每一个独立的值。例如，**sqrt(c(4, 16, 25))**的返回值为**c(2,4,5)**。

# 1. 数据处理

| 函　数 | 描　述 |
| --- | --- |
| round(x, digits=n) | 将x舍入为指定位的小数<br>round(3.475, digits=2)返回值为3.48 |
| signif(x, digits=n) | 将x舍入为指定的有效数字位数<br>signif(3.475, digits=2)返回值为3.5 |
| cos(x)、sin(x)、tan(x) | 余弦、正弦和正切<br>cos(2)返回值为－0.416 |
| acos(x)、asin(x)、atan(x) | 反余弦、反正弦和反正切<br>acos(-0.416)返回值为2 |
| cosh(x)、sinh(x)、tanh(x) | 双曲余弦、双曲正弦和双曲正切<br>sinh(2)返回值为3.627 |
| acosh(x)、asinh(x)、atanh(x) | 反双曲余弦、反双曲正弦和反双曲正切<br>asinh(3.627)返回值为2 |
| log(x,base=n) | 对x取以n为底的对数 |
| log(x) | 为了方便起见 |
| log10(x) | log(x)为自然对数<br>log10(x)为常用对数<br>log(10)返回值为2.3026<br>log10(10)返回值为1 |
| exp(x) | 指数函数<br>exp(2.3026)返回值为10 |

# 1. 数据处理

- 统计函数

| 函　数 | 描　述 |
| --- | --- |
| mean(x) | 平均数<br>mean(c(1,2,3,4))返回值为2.5 |
| median(x) | 中位数<br>median(c(1,2,3,4))返回值为2.5 |
| sd(x) | 标准差<br>sd(c(1,2,3,4))返回值为1.29 |
| var(x) | 方差<br>var(c(1,2,3,4))返回值为1.67 |
| mad(x) | 绝对中位差（median absolute deviation）<br>mad(c(1,2,3,4))返回值为1.48 |
| quantile(x,probs) | 求分位数。其中x为待求分位数的数值型向量，probs为一个由[0,1]之间的概率值组成的数值向量<br># 求x的30%和84%分位点<br>y <- quantile(x, c(.3,.84)) |
| range(x) | 求值域<br>x <- c(1,2,3,4)<br>range(x)返回值为c(1,4)<br>diff(range(x))返回值为3 |
| sum(x) | 求和<br>sum(c(1,2,3,4))返回值为10 |
| diff(x, lag=n) | 滞后差分，lag用以指定滞后几项。默认的lag值为1<br>x<- c(1, 5, 23, 29)<br>diff(x)返回值为c(4, 18, 6) |
| min(x) | 求最小值<br>min(c(1,2,3,4))返回值为1 |
| max(x) | 求最大值<br>max(c(1,2,3,4))返回值为4 |
| scale(x,center=TRUE, scale=TRUE) | 为数据对象x按列进行中心化（center=TRUE）或标准化（center=TRUE,scale=TRUE）；<br>代码清单5-6中给出了一个示例 |

# 1. 数据处理

【操作练习】均值和标准差的计算以及数据标准化(5分钟)

```
x <- c(0:9, 50)
x
mean(x)
mean(x, trim = 0.10, na.rm=TRUE)
sd(x)
newdata <- scale(x) # 默认情况下，函数scale()对矩阵或数据框的指定列进行均值为0、标准差为1的标准化
newdata
newdata <- scale(x)*10+50#标准化为均值50标准差为10的变量
newdata
mean(newdata)
sd(newdata)
```

# 1. 数据处理

【提示】当标准化指定列而不是整个矩阵或数据框时，可以：

```
newdata <- transform(dataframename,
colname=scale(colname)*SD+M)
```

# 1. 数据处理

- 概率函数

概率函数通常用来生成特征已知的模拟数据，以及在用户编写的统计函数中计算概率值。概率函数形如：

**[dpqr]distribution_abbreviation()**

其中：

**d = 密度函数（density）**

**p = 分布函数（distribution function）**

**q = 分位数函数（quantile function）**

**r = 生成随机数（随机偏差），**

其中**distribution_abbreviation()**指分布名称的缩写。

# 1. 数据处理

| 分布名称 | 缩　写 | 分布名称 | 缩　写 |
|---|---|---|---|
| Beta分布 | beta | Logistic分布 | logis |
| 二项分布 | binom | 多项分布 | multinom |
| 柯西分布 | cauchy | 负二项分布 | nbinom |
| (非中心)卡方分布 | chisq | 正态分布 | norm |
| 指数分布 | exp | 泊松分布 | pois |
| F分布 | f | Wilcoxon符号秩分布 | signrank |
| Gamma分布 | gamma | t分布 | t |
| 几何分布 | geom | 均匀分布 | unif |
| 超几何分布 | hyper | Weibull分布 | weibull |
| 对数正态分布 | lnorm | Wilcoxon秩和分布 | wilcox |

# 1. 数据处理

【操作练习】生成服从均匀分布的伪随机数（5分钟）

```
runif(5) #生成0到1区间上服从均匀分布的伪随机数
runif(5)
set.seed(123)  #设定随机数种子
runif(5)
set.seed(123)
runif(5)
```

【提示】通过手动设定种子，可以使生成的随机数结果一致。这有助于我们创建会在未来使用，以及可与他人分享的样本。

# 1. 数据处理

在模拟研究和蒙特卡洛方法中，经常需要获取来自给定均值向量和协方差阵的多元正态分布的数据。MASS包中的mvrnorm()函数可以解决这个问题。

调用格式为：

mvrnorm(n,mean,sigma)

其中：

n是样本大小

mean为均值向量

sigma是方差-协方差矩阵（或相关矩阵）

# 1. 数据处理

【操作练习】从一个参数如下的三元正态分布中抽取500个观测（8分钟）。

| | | | |
|---|---|---|---|
| 均值向量 | 230.7 | 146.7 | 3.6 |
| 协方差阵 | 15360.8 | 6721.2 | -47.1 |
| | 6721.2 | 4700.9 | -16.5 |
| | -47.1 | -16.5 | 0.3 |

# 1. 数据处理

```
library(MASS)
options(digits=3)
set.seed(1234)
mean <- c(230.7, 146.7, 3.6) #指定均值向量
sigma <- matrix( c(15360.8, 6721.2, -47.1,
                   6721.2, 4700.9, -16.5, -47.1,  -16.5,   0.3),
                 nrow=3, ncol=3) #指定协方差阵
mydata <- mvrnorm(500, mean, sigma)   #生成数据
mydata <- as.data.frame(mydata)    #转换成数据框
names(mydata) <- c("y", "x1", "x2")   #命名变量
dim(mydata)              #查看维度
head(mydata, n=10)  #查看前10行
```

# 1. 数据处理

- 字符处理函数（略）

| 函　数 | 描　述 |
| --- | --- |
| nchar(*x*) | 计算x中的字符数量<br>x <- c("ab", "cde", "fghij")<br>length(x)返回值为3（参见表5-7）<br>nchar(x[3])返回值为5 |
| substr(*x*, *start*, *stop*) | 提取或替换一个字符向量中的子串<br>x <- "abcdef"<br>substr(x, 2, 4)返回值为"bcd"<br>substr(x, 2, 4) <- "22222"(*x*将变成"a222ef") |
| grep(*pattern*, *x*, ignore. case=FALSE, fixed=FALSE) | 在*x*中搜索某种模式。若fixed=FALSE，则*pattern*为一个正则表达式。若fixed=TRUE，则*pattern*为一个文本字符串。返回值为匹配的下标<br>grep("A",c("b","A","c"),fixed=TRUE)返回值为2 |

# 1. 数据处理



| 函　数 | 描　述 |
| --- | --- |
| sub(*pattern*, *replacement*, *x*, ignore.case=FALSE, fixed=FALSE) | 在*x*中搜索*pattern*，并以文本*replacement*将其替换。若fixed=FALSE，则*pattern*为一个正则表达式。若fixed=TRUE，则*pattern*为一个文本字符串<br>sub("\\s",".","Hello There")返回值为Hello.There。注意，"\s"是一个用来查找空白的正则表达式；使用"\\s"而不用"\"的原因是，后者是R中的转义字符（参见1.3.3节） |
| strsplit(*x*, *split*, fixed=FALSE) | 在*split*处分割字符向量*x*中的元素。若fixed=FALSE，则*pattern*为一个正则表达式。若fixed=TRUE，则*pattern*为一个文本字符串<br>y <- strsplit("abc", "")将返回一个含有1个成分、3个元素的列表，包含的内容为"a" "b" "c"<br>unlist(y)[2]和sapply(y, "[", 2)均会返回"b" |
| paste(…, sep="") | 连接字符串，分隔符为sep<br>paste("x", 1:3,sep="")返回值为c("x1", "x2", "x3")<br>paste("x",1:3,sep="M")返回值为c("xM1","xM2" "xM3")<br>paste("Today is", date())返回值为Today is Thu Jun 25 14:17:32 2011（我修改了日期以让它看起来更接近当前的时间） |
| toupper(*x*) | 大写转换<br>toupper("abc")返回值为"ABC" |
| tolower(*x*) | 小写转换<br>tolower("ABC")返回值为"abc" |

# 1. 数据处理

- 其他实用函数（略）

| 函　数 | 描　述 |
| --- | --- |
| length(*x*) | 对象x的长度<br>x <- c(2, 5, 6, 9)<br>length(x)返回值为4 |
| seq(*from*, *to*, *by*) | 生成一个序列<br>indices <- seq(1,10,2)<br>indices的值为c(1, 3, 5, 7, 9) |
| rep(*x*, *n*) | 将*x*重复*n*次<br>y <- rep(1:3, 2)<br>y的值为c(1, 2, 3, 1, 2, 3) |
| cut(*x*, *n*) | 将连续型变量*x*分割为有着*n*个水平的因子<br>使用选项ordered_result = TRUE以创建一个有序型因子 |
| pretty(*x*, *n*) | 创建美观的分割点。通过选取*n*+1个等间距的取整值，将一个连续型变量*x*分割为*n*个区间。绘图中常用 |
| cat(… , file ="myfile", append =FALSE) | 连接…中的对象，并将其输出到屏幕上或文件中（如果声明了一个的话）<br>firstname <- c("Jane")<br>cat("Hello" ,firstname, "\n") |

# 1. 数据处理

【操作练习】将函数应用于矩阵和数据框（5分钟）。

```
a <- 5
sqrt(a)
b <- c(1.243, 5.654, 2.99)
round(b)
c <- matrix(runif(12), nrow=3)
c
log(c)
mean(c)    #所有元素的均值
mydata <- matrix(rnorm(30), nrow=6)
mydata
```

# 1. 数据处理

#apply函数经常用来计算矩阵中行或列的均值、和值的函数

```
apply(mydata, 1, mean)     #计算每行的均值
apply(mydata, 2, mean)     #计算每列的均值
apply(mydata, 2, mean, trim=0.2)
                           #计算每列中间60%元素的均值
```

# 2. R语言介绍

**Break Time**

**5 minutes**

# 1. 数据处理

【问题求解】将学生的各科考试成绩组合为单一的成绩衡量指标、基于相对名次（前20%，下20%，等等）给出从A到F的评分、根据学生姓氏和名字的首字母对花名册进行排序。（10分钟）

```
options(digits=3) #限定条件，小数点后3位
Student <- c("John Davis", "Angela Williams", "Bullwinkle Moose", "David Jones", "Janice Markhammer", "Cheryl Cushing", "Reuven Ytzrhak", "Greg Knox", "Joel England", "Mary Rayburn")
Math <- c(502, 600, 412, 358, 495, 512, 410, 625, 573, 522)
Science <- c(95, 99, 80, 82, 75, 85, 80, 95, 89, 86)
English <- c(25, 22, 18, 15, 20, 28, 15, 30, 27, 18)
roster <- data.frame(Student, Math, Science, English, stringsAsFactors=FALSE)
```

# 1. 数据处理

```
z <- scale(roster[,2:4]) #使不同成绩统一标准量化
score <- apply(z, 1, mean) #计算出平均成绩
roster <- cbind(roster, score) #把平均成绩加入到表格中
y <- quantile(score, c(.8, .6, .4, .2)) #设置成绩的分界点
roster$grade[score >= y[1]] <- "A" #对学生成绩分类编组
roster$grade[score < y[1] & score >= y[2]] <- "B"
roster$grade[score < y[2] & score >= y[3]] <- "C"
roster$grade[score < y[3] & score >= y[4]] <- "D"
roster$grade[score < y[4]] <- "F"
```

# 1. 数据处理

```
name <- strsplit((roster$Student), " ")    #分割姓名
name
lastname <- sapply(name, "[", 2)# 提取第二个元素
firstname <- sapply(name, "[", 1)#提取第一个元素

roster <- cbind(firstname,lastname, roster[,-1])
roster <- roster[order(lastname,firstname),] #排序

roster
```

# 2. 程序流程控制

- 语句（statement）是一条单独的R语句或一组复合语句（包含在花括号{ } 中的一组R语句，使用分号分隔）；
- 条件（cond）是一条最终被解析为真（TRUE）或假（FALSE）的表达式；
- 表达式（expr）是一条数值或字符串的求值语句；
- 序列（seq）是一个数值或字符串序列。

# 2. 程序流程控制

- 循环语句

```
for(var in seq)
        statement
```

或者

```
while(cond)
        statement
```

# 2. 程序流程控制

【操作练习】用for语句求1到100的和（5分钟）

```
sum<-0
for (i in 1:100)
  sum=sum+i
print(sum)
```

【问题】如何用while语句求1到100的和？（5分钟）

```
while(cond)
        statement
```

# 2. 程序流程控制

- 分支语句

```
if (cond)
        statement

if (cond) statement1 else statement2

ifelse(cond,statement1,statement2)

switch(expr,…)
```

# 2. 程序流程控制

【操作练习】从键盘输入一个数，如果是奇数，则输出“是奇数”，如果是偶数则输出“是偶数”。（5分钟）

```
number<-scan("")#从键盘输入数据
if(number%%2==0)
  print("是偶数") else  #else一行前面无内容时会报错
  print("是奇数")
```

【运行提示】在左下角提示符>后：

输入数据+回车

Ctrl+z+回车

# 3. 自定义函数

函数的基本结构：

```
myfunction <- function(arg1,arg2,…){
    statements
    return(object)
}
```

【注意】函数中的对象只在函数内部使用。返回对象的数据类型是任意的，从标量到列表皆可。

# 3. 自定义函数

【操作练习】自定义求1…n的和的函数，并调用这个函数求1到1000的和（10分钟）。

◆第1步 定义函数

```
add1To100<-function(n){
sum<-0
for (i in 1:n)
  sum=sum+i
return (sum)
}
```

# 3. 自定义函数

◆第2步 把上面的代码存在文件functionTest.R中，放到"D:/R" 下

◆第3步 调用add函数

```
source("functionTest.R")
result<-add1To100(100)
print(result)
```

# 4. 整合与重构

R中提供了许多用来整合（aggregate）和重塑（reshape）数据的强大方法。在整合数据时，往往将多组观测替换为根据这些观测计算的描述性统计量。在重塑数据时，则会通过修改数据的结构（行和列）来决定数据的组织方式。

# 4. 整合与重构

- 转置

转置（反转行和列）。使用函数t()即可对 一个矩阵或数据框进行转置。对于后者，行名将成为变量（列）名。

【操作练习】2分钟

```
cars <- mtcars[1:5, 1:4]    #mtcars为系统自带数据集
cars
t(cars)
```

# 4. 整合与重构

- 整合数据

调用格式为：

**aggregate(x, by, FUN)**

其中:

**x**是待整合的数据对象；**by**是一个变量名组成的列表（这些变量将被去掉以形成新的观测）；**FUN**则是用来计算描述性统计量的标量函数，它将被用来计算新观测中的值。

# 4. 整合与重构

【操作练习】根据汽缸数和挡位数整合mtcars数据，并返回各个数值型变量的均值。（5分钟）

```
options(digits=3)
attach(mtcars)
mtcars
aggdata <-aggregate(mtcars, by=list(cyl,gear),
   FUN=mean, na.rm=TRUE)
aggdata
```

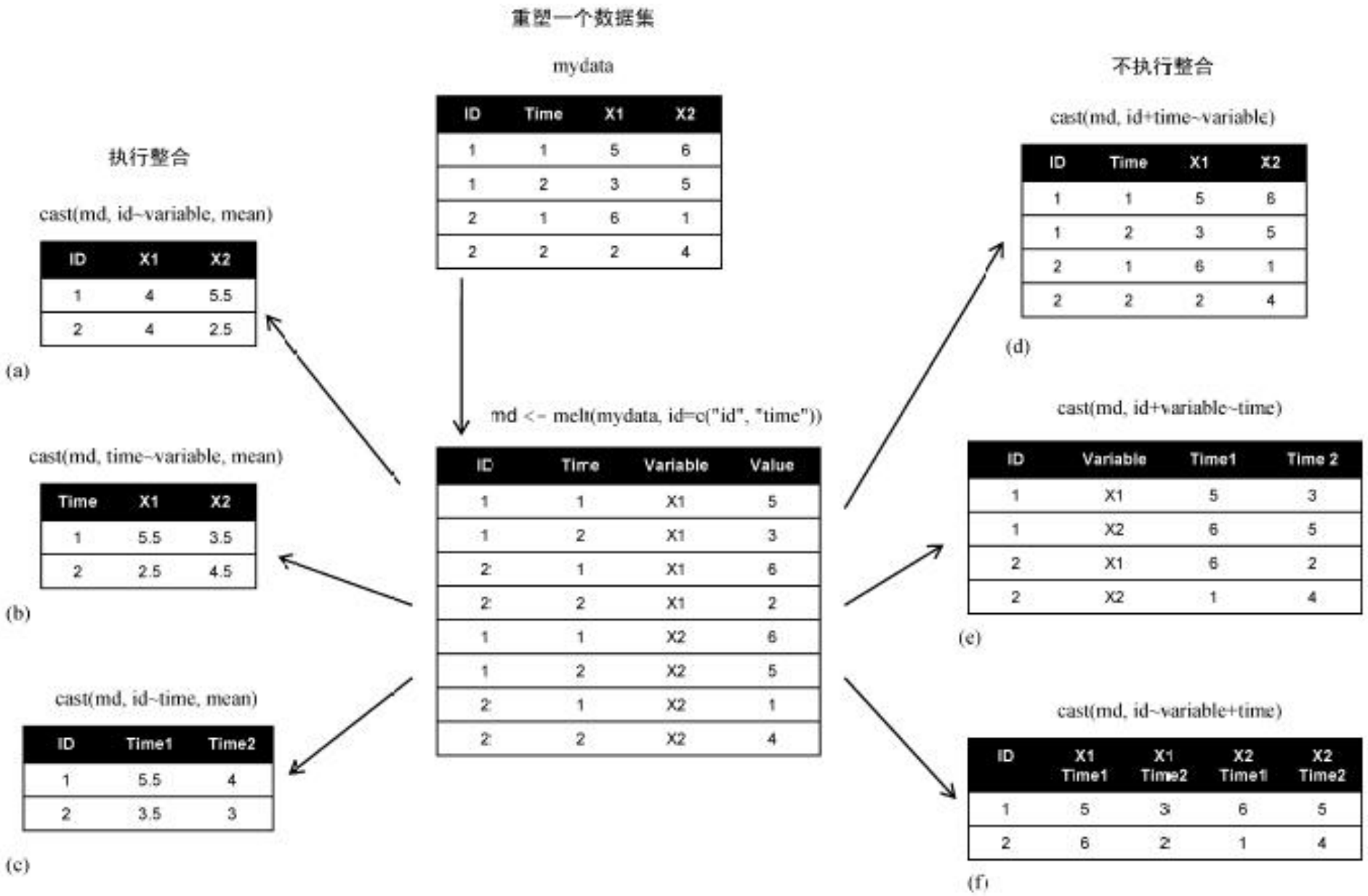
# 4. 整合与重构（略）

- reshape包（略）

Reshape/reshape2包是一套重构和整合数据集的绝妙的万能工具。在第一次使用它之前需要使用install.packages("reshape")进行安装。

你需要首先将数据“融合”（melt），以使每一行都是一个唯一的标识符—变量组合。然后将数据“重铸”（cast）为你想要的任何形状。在重铸过程中，你可以使用任何函数对数据进行整合。

# 4. 整合与重构（略）


重塑一个数据集
mydata
ID Time X1 X2
1 1 5 6
1 2 3 5
2 1 6 1
2 2 2 4
md <- melt(mydata, id=c("id", "time"))
ID Time Variable Value
1 1 X1 5
1 2 X1 3
2 1 X1 6
2 2 X1 2
1 1 X2 6
1 2 X2 5
2 1 X2 1
2 2 X2 4
执行整合
cast(md, id~variable, mean)
ID X1 X2
1 4 5.5
2 4 2.5
(a)
cast(md, time~variable, mean)
Time X1 X2
1 5.5 3.5
2 2.5 4.5
(b)
cast(md, id~time, mean)
ID Time1 Time2
1 5.5 4
2 3.5 3
(c)
不执行整合
cast(md, id+time~variable)
ID Time X1 X2
1 1 5 6
1 2 3 5
2 1 6 1
2 2 2 4
(d)
cast(md, id+variable~time)
ID Variable Time1 Time 2
1 X1 5 3
1 X2 6 5
2 X1 6 2
2 X2 1 4
(e)
cast(md, id~variable+time)
ID X1 Time1 X1 Time2 X2 Time1 X2 Time2
1 5 3 6 5
2 6 2 1 4
(f)

# 课下作业

1、重复练习（但不限于）课上练习的内容，将操作界面截图到Word文档，发word文档到学堂云交作业。

2、你和你的小伙伴使用洪荒之力，完成数据的收集/采集。