用 R 也能做精算—actuar 包学习笔记（一）

李皞
（中国人民大学 统计学院 风险管理与精算）

本文是对 R 中精算学专用包 actuar 使用的一个简单教程。actuar 项目开始于 2005 年，在 2006 年 2 月首次提供公开下载，其目的就是将一些常用的精算功能引入 R 系统。actuar 是一个集成化的精算函数系统，虽然其他 R 包中的很多函数可以供精算师使用，但是为了达到某个目的而寻找某个包的某个函数是一个费时费力的过程，因此，actuar 将精算建模中常用的函数汇集到一个包中，方便了人们的使用。目前，该包提供的函数主要涉及风险理论，损失分布和信度理论，特别是为非寿险研究提供了很多方便的工具。

如题所示，本文是我在学习 actuar 包过程中的学习笔记，主要涉及这个包中一些函数的使用方法和细节，对一些方法的结论也有稍许探讨，因此能简略的地方简略，而讨论的地方可能讲的会比较详细。文章主要是针对 R 语言的初学者，因此每种函数或数据的结构进行了尽可能直白的描述，以便于理解，如有描述不清或者错漏之处，敬请各位指正。闲话少提，下面就正式开始咯！

1 数据描述

本节介绍描述数据的基本方法，数据类型主要分为分组数据和非分组数据。对于非分组数据的描述方法大家会比较熟悉，无论是数量上，还是图形上的，比如均值、方差、直方图、柱形图还有核密度估计等。因此下文的某些部分只介绍如何处理分组数据。

1.1 构造分组数据对象

分组数据是精算研究中经常见到的数据类型，虽然原始的损失数据比分组数据包含有更多的信息，但是某些情况下受条件所限，只能获得某个损失所在的范围。与此同时，将数据分组也是处理原始数据的基本方法，通过将数据分到不同的组中，我们可以看到各组中数据的相对频数，有助于对数据形成直观的印象（比如我们对连续变量绘制直方图）；而且在生存函数的估计中，数据量经常成千上万，一种处理方法是选定合适的时间或损失额度间隔，对数据进行分组，然后再使用分组数据进行生存函数的估计，这样可以有效减小计算量。现在假设我们要把一组连续变量分为 r 组:$(c_0, c_1], (c_1, c_2], \ldots, (c_{r-1}, c_r]$,那么就需要定义 r+1 个边界$c_0, c_1, \ldots, c_r$。实际中的损失数据或生存数据都是取非负值，因此$c_0$经常取 0。

对于分组数据来说，只需要知道每个组的数值范围及落在该组的观测频数，因此要构造一个完整的分组数据只需要提供上面两个信息即可。下面是分组数据的构造函数，注意这个函数是构造一个分组数据的结构，而非对现有连续数据进行分组，该函数返回一个分组数据的对象（grouped data object）。

函数语法：

```
gouped.data(Group=c(…),freq1=c(…),freq2=c(….),….,right=TRUE,ro
w.names=NULL)
```

用法说明：

1）Group 定义的是分组的边界值，freq1 和 freq2（还可以定义 freq3 及更多）是每条分组数据的频数。Group，freq1 和 freq2 可以随意命名，比如我们可以将 Group 改为“分数档”，将 freq1，freq2 改为“一班”，“二班”等，那么“一班”，“二班”就是两条分组数据，有着共同的分组数据边界值。要注意，一定要把边界值向量放在第一个参数的位置！

2）Group 向量要比 freq 向量多出一个长度（边界数比组数多 1）。

3）默认分组区间是左开右闭，如果想变为左闭右开可以设置 right=FALSE。row.names 可以自定义行的名称。

4）返回的是一个数据框。特别要注意对第一列的处理，见下面例子。

例子：

```
> x=grouped.data(Group= c(0, 25, 50, 100, 150,250, 500),
+      Line.1 = c(30, 31, 57, 42, 65, 84),
+      Line.2 = c(26, 33, 31, 19, 16, 11))
> x
      Group Line.1 Line.2
1   (0,  25]     30     26
2  (25,  50]     31     33
3  (50, 100]     57     31
4 (100, 150]     42     19
5 (150, 250]     65     16
6 (250, 500]     84     11
#改成左闭右开区间，自定义行名称
> x=grouped.data(Group= c(0, 25, 50, 100, 150,250, 500),
+      Line.1 = c(30, 31, 57, 42, 65, 84),
+      Line.2 = c(26, 33, 31, 19, 16, 11),
+      right=F,row.names=LETTERS[1:6])
> x
      Group   Line.1 Line.2
A   [0,  25)      30     26
B  [25,  50)      31     33
C  [50, 100)     57     31
D [100, 150)     42     19
E [150, 250)     65     16
F [250, 500)     84     11
#提取某个组的数据（某行）
> x[1,]
    Group Line.1 Line.2
A [0,  25)     30     26
#提取第一条分组数据（某列）
> x[,2]
[1] 30 31 57 42 65 84
#提取边界值。如果引用第一列你期待会出现什么结果呢？
> x[,1]
[1]   0  25  50 100 150 250 500
```

#如同任何对数据框的操作，也可以对数据框中的数据进行修改。特别需要注意的是对第一列的修改，一定要同时指定分组区间的左右的边界值。如：

```
> x[1,1]=c(0,20)
> x
      Group Line.1 Line.2
1   (0,  20]     30     26
2  (20,  50]     31     33
3  (50, 100]     57     31
4 (100, 150]     42     19
5 (150, 250]     65     16
6 (250, 500]     84     11
```

#体会这样修改波及的范围。

```
> x[c(3, 4), 1] <- c(55, 110, 160)
> x
      Group Line.1 Line.2
1   (0,  20]     30     26
2  (20,  55]     31     33
3  (55, 110]     57     31
4 (110, 160]     42     19
5 (160, 250]     65     16
6 (250, 500]     84     11
```

#如果只指定一个边界的话，那就默认左右边界值相同，所以不要这样做。

```
> x[1,1]=10
> x
      Group Line.1 Line.2
1  (10,  10]     30     26
2  (10,  55]     31     33
3  (55, 110]     57     31
4 (110, 160]     42     19
5 (160, 250]     65     16
6 (250, 500]     84     11
```

到这里可能有人会问，如果现在我手中只有一组连续数据，如何实现对数据的分组呢？答案就是使用 cut 函数。

例子：

#生成 100 个均值为 5 的指数分布随机数。

```
> z=rexp(100,rate=0.2)
```

#指定边界点，也是划分点。

```
> break.points=c(0,1,4,8,14,Inf)
> (tz1=table(cut(z,breaks=break.points)))
   (0,1]    (1,4]    (4,8]   (8,14] (14,Inf]
      16       33       33       14        4
> (tz2=as.data.frame(tz1))
     Var1 Freq
```

```
1    (0,1]   16
2    (1,4]   33
3    (4,8]   33
4   (8,14]   14
5 (14,Inf]    4
```
#用汇总结果直接构造分组数据对象。
```
> grouped.data(Group=break.points,freq=tz2[,2])
     Group freq
1  (0,   1]   16
2  (1,   4]   33
3  (4,   8]   33
4  (8,  14]   14
5 (14, Inf]    4
```

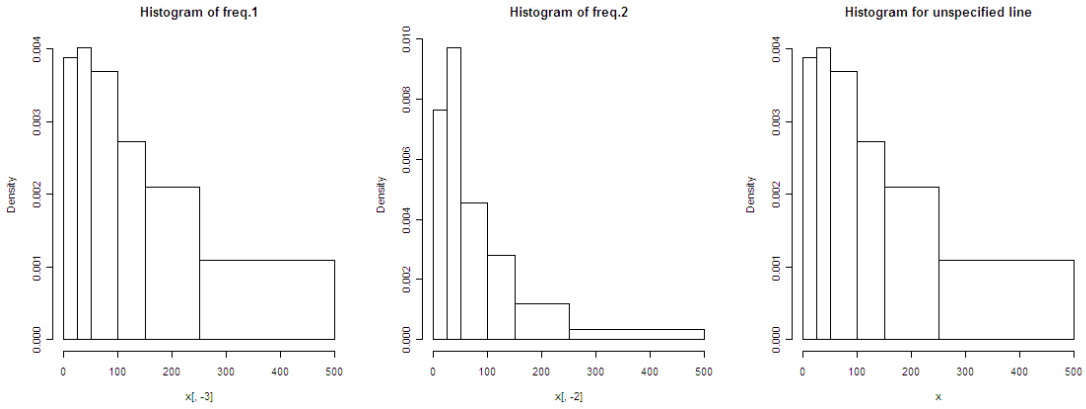
1.2 分组数据分布图

有了 grouped.data 对象，我们就可以对该对象进行一系列操作。首先是绘制分组数据的经验密度函数—直方图，和经验分布函数—拱形图。

1）绘制直方图

由于数据已经被划分好组别，因此 R 会应用分组数据对象 x 的第一列划分组距并绘制直方图，这在绘制非等距直方图时是十分方便的。由于每次只能绘制一组频率，因此绘图时需要指定频率所在的列，如果不指定，默认绘制第一组频率（数据框的第二列）。例子：

```
> layout(matrix(1:3,1,3))
> hist(x[,-3],main='Histogram of freq.1')
> hist(x[,-2],main='Histogram of freq.2')
> hist(x,main='Histogram for unspecified line')
```



2）绘制分组数据经验分布函数

如同对连续的随机变量可以绘制经验分布函数图一样，对于分组数据也可以绘制"拱形图"（ogive），也就是令分组临界点的函数值等于累计频率，对临界点间的函数值使用线

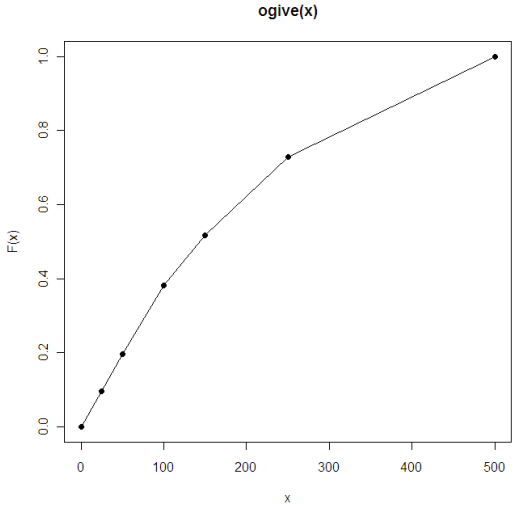
性插值的方法构造的一条曲线。累计频率曲线的公式如下：

$$\tilde{F}_n(x)=\begin{cases}0 & x\le c_0\\ \frac{(c_j-x)F_n(c_{j-1})+(x-c_{j-1})F_n(c_j)}{c_j-c_{j-1}} & c_{j-1}<x\le c_j\\ 1 & x>c_r\end{cases} \tag{1}$$

函数 ogive(x)输入的是分组数据对象 x，返回的是一个阶梯函数对象(Step Function Class)，也就是说实现了分组数据对象向阶梯函数对象的转换。如果给定函数的横坐标，就可以返回相对应的函数值，这点和 ecdf 是相同的。我们可以通过 konts 返回阶梯函数对象的临界点/间断点，通过 plot 绘制阶梯函数。

例子：

```
> Fnt=ogive(x)  #得到一个阶梯函数
> knots(Fnt)  #返回临界点
[1]   0  25  50 100 150 250 500
> Fnt(knots(Fnt))   #返回临界点对应的累积频率值
[1] 0.00000000 0.09708738 0.19741100 0.38187702 0.51779935
0.72815534 1.00000000
> plot(Fnt)   #对函数作图，得到 ogive 曲线
```



1.3 计算数据经验矩

1）经验原点矩

首先是计算经验一阶矩[1]。函数 mean 是一个泛型函数，既可以作用于非分组数据向量对象，也可以作用于分组数据对象 x，计算分组数据均值。非分组数据的经验均值就是将所有数据算术平均，分组数据的经验均值定义为：

$$\frac{1}{n}\sum_{j=1}^{r}n_j\left(\frac{c_{j-1}+c_j}{2}\right) \tag{2}$$

该公式使用每组观测数乘以两端边界点的均值，得到该组的总值，将所有 r 个组的总值相

[1]文中经常会提到“经验**”，比如经验分布函数，经验一阶矩等，个人理解所谓经验就是将样本当成总体去对待，比如经验方差是除以样本量 n，而样本方差是除以 n-1，或者说在 bootstrap 方法中，使用经验分布函数去代替总体。

加再除以样本量 n，就得到每个观测均值的估计。该公式假设每组内的观测值分布是均匀的。
例子：

```
> x
      Group Line.1 Line.2
A   [0,  25)     30     26
B  [25,  50)     31     33
C  [50, 100)     57     31
D [100, 150)     42     19
E [150, 250)     65     16
F [250, 500)     84     11
> mean(x)
   Line.1    Line.2
179.81392  99.90809
```

如果直接用公式(2)计算，Line.1的均值等价于：

```
> ((0+25)/2*30+(25+50)/2*31+(50+100)/2*57+(100+150)/2*42
+ +(150+250)/2*65+(250+500)/2*84)/sum(x[,2])
[1] 179.8139
```

如果说均值函数 mean()只能计算一阶矩，那么 emm 函数则可以计算任意阶的经验原点矩。首先引入 actuar 包中的两个数据集。其中 dental 是非分组数据，gdental 是分组数据。

```
> data(dental)
> dental
 [1]  141   16   46   40  351  259  317 1511  107  567
> data(gdental)
> gdental
           cj   nj
1      (0,  25]  30
2    ( 25,  50] 31
3    ( 50, 100] 57
4    (100, 150] 42
5    (150, 250] 65
6    (250, 500] 84
7   (500, 1000] 45
8  (1000, 1500] 10
9  (1500, 2500] 11
10 (2500, 4000]  3
```

emm 函数可以计算任意阶经验原点矩，其使用方法是这样的： emm(x,order=1)。其中，order 是阶数，可以赋值给它一个向量，这样就能一次性计算多个原点矩。x 可以是数据向量或者是矩阵，对于矩阵，emm 将每一列视为一条数据。
非分组数据 k 阶经验矩的计算公式为：

$$\frac{1}{n}\sum_{j=1}^{n}x_j^k \tag{3}$$

例子：

```
#非分组数据向量
> emm(dental,2)
[1] 293068.3
#数据矩阵
> xx=matrix(1:9,3,3)
> xx
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
#第一行是 xx 第一列的均值和二阶矩，第二行是 xx 第二列的均值和二阶矩，依此类推。
> emm(xx,1:2)
      [,1]      [,2]
[1,]    2  4.666667
[2,]    5 25.666667
[3,]    8 64.666667
```

如果是分组数据，x 也可以是由 grouped.data()生成的分组数据对象。分组数据 k 阶经验矩的计算公式为：

$$\frac{1}{n}\sum_{j=1}^{r}n_j\frac{(c_j^{k+1}-c_{j-1}^{k+1})}{(k+1)\cdot(c_j-c_{j-1})} \tag{4}$$

例子：

```
> emm(gdental,1:3)
[1] 3.533399e+02 3.576802e+05 6.586332e+08
```

2）经验有限期望值

有时候，损失数据会有保单限额 u 的存在（有关保单限额的介绍参见 2.3 节），超过 u 的损失额度被强制定义为 u。elev 函数可以计算经验有限期望值（empirical limited expected value），经验有限期望值都是一阶矩。使用方法是 elev(x)：x 可以是非分组数据，也可以是分组数据。

对于非分组数据，经验有限期望值的公式为：

$$\hat{E}[X\wedge u]=f(u)=\frac{1}{n}\sum_{j=1}^{n}\min(x_j,u) \tag{5}$$

对于分组数据，还要考虑 u 是否是位于分组的边界值上，因此经验有限期望公式比较复杂，在此略去，有兴趣的同学可以参考 ACTEX 出版的 C 的 manual 或者 Loss Models 的第一版。

我们注意到，有限期望值是上限值 u 的函数，不同的 u 计算出的经验有限期望值是不一样的。elev 索性返回一个函数对象，如果要具体计算某一个 u 的经验有限期望值，只需要特

别指定 u 即可。

例子：

#非分组数据

```
> lev=elev(dental)
```

#这里将保单限额设为200

```
> lev(200)
[1] 135
```

#返回lev函数的拐点，注意到拐点都发生在数据点，而观察下图可知拐点间的函数都是线性的，你能从数学上解释这个现象吗？

```
> knots(lev)
 [1]   16   40   46  107  141  259  317  351  567 1511
```

#分组数据

```
> lev2=elev(gdental)
> lev2(200)
[1] 142.4603
```

#作图

```
> par(mfrow=c(1,2))
> plot(lev,type='o',pch=19)
> plot(lev2,type='o',pch=19)
```