第 5 章

数据预处理

我们来设想一个情境，假设你是苹果公司销售部门总管，你需要带领部门人员整理分析近几年来的销售数据。当你着手进行这项工作，仔细地审查公司的数据库和数据仓库，识别并选择应当包含在分析中的属性时，你可能会发现，根本无法进行分析。比如，你想要分析亚太地区 iPhone 5S 的销售情况，但北京地区的销售记录存在一个星期的空白、天津地区某天的销售记录为负、某段时间上海地区的销量远小于新疆地区。这些信息明显与实际情况不符，因此，为了得到准确的分析报告，你必须对这些不符合常理的情况进行处理。

在商务与日常实践中，需要使用数据挖掘技术分析的数据通常是不完整（缺少属性值或某些感兴趣的属性，或仅包含聚集数据）、含噪声（包含错误或存在偏离期望的离群值）、并且不一致的（例如，用于商品分类的部门编码存在差异），这样的数据必须经过预处理，剔除其中噪声、恢复数据完整性和一致性后才能使用数据挖掘技术进行分析。

要知道，若想依照设想来获取有效、准确、可信的数据，从而对其展开挖掘探究并非易事，数据的前期准备相比于后续的分析过程是一项更耗时耗力，需要耐心和技巧的工作。

5.1 数据集加载

在商业分析中，一些变量能够对结果产生直接的、巨大的影响，对于这些变量就需要进行十分严格的控制以保证数据质量；而另一些变量可能仅仅提供一些背景信息或者没有那么重要，对于这样的数据，就不必对数据质量要求十分严格。而分析人员往往并不亲自参加调查，这个项目的数据也可能来自于另一个项目，如何能够分辨变量的重要程度，一件必须要做的事情就是了解数据集背后的内容。

我们需要知道数据收集的目的、方法和途径，这些信息都能够增强对于数据集的理解。花时间去理解数据背后的含义是十分必要的，它的必要性体现在建模之前的构思、对模型含义的理解，以及当模型出现特殊形式或异常时，我们需要重新探索数据结构，以便接下来修正数据集并重新构造更优模型。

本章我们将使用 mice 软件包中的示例数据集来进行数据预处理演示，由于 mice 软件包以软件包 lattice、MASS 及 nnet 为基础建立，因此在加载 mice 软件包前要先安装、加载这三个软件包。

```
> install.packages(lattice)                    # 安装 lattice 软件包
> install.packages(MASS)                       # 安装 MASS 软件包
> install.packages(nnet)                       # 安装 nnet 软件包
> library(lattice)                             # 加载 lattice 软件包
> library(MASS)                                # 加载 MASS 软件包
> library(nnet)                                # 加载 nnet 软件包
```

对于 5.2 节数据清理部分，我们选择使用 nhanes2 数据集进行演示，该数据集是一个含有缺失值的小规模数据集，我们先来对其做一个简单了解。

```
> install.packages(mice)                       # 安装 mice 软件包
> library(mice)                                # 加载 mice 软件包
> data(nhanes2)                                # 获取 nhanes2 数据集
> nrow(nhanes2);ncol(nhanes2)                  # 显示 nhanes2 数据集的行列数
[1] 25
[1] 4
> summary(nhanes2)                             # 获取 nhanes2 数据集的概括信息
age          bmi               hyp          chl
 20-39:12    Min.    :20.40    no    :13    Min.    :113.0
 40-59: 7    1st Qu. :22.65    yes   : 4    1st Qu. :185.0
 60-99: 6    Median  :26.75    NA's  : 8    edian   :187.0
             Mean    :26.56                 Mean    :191.4
             3rd Qu. :28.93                 3rd Qu. :212.0
             Max.    :35.30                 Max.    :284.0
             NA's    :    9                 NA's    :10
```

从中我们可以看出，age 和 hyp 是定性变量，分别分为 3 类和 2 类，bmi 和 chl 是定量变量；age 没有缺失值，bmi 有 9 个缺失值，hyp 有 8 个缺失值，chi 有 10 个缺失值。

需要补充说明的是，数据中存在缺失值的情况主要有两种：其一，当数据输入时其中某个观测值缺失，数据表的相应位置显示为 NA，如上变量中出现 NA 值就是这种情况；其二，当数据表经由其他数据表计算得来时，存在计算错误或计算值不符合要求时，也会显示为 NA。

结合数据背景信息中显示的内容，我们可以对数据集 nhanes2 做一个初步了解。

age：年龄段，取值为 1、2、3，分别代表 20-39、40-59、60-99 这三个年龄段；

bmi：身体质量指数，单位为 kg/m^2；
hyp：是否患高血压，1 代表“否”，2 代表“是”；
chl：血清胆固醇总量，单位为 mg/dL。

为了对我们将要使用的数据集有直观的把握，现将 nhanes 2 数据集的前 6 条数据展示如下。

```
> head(nhanes2)                                    # 输出 nhanes2 数据集的前若干条
  age  bmi hyp chl
1   1   NA  NA  NA
2   2 22.7   1 187
3   1   NA   1 187
4   3   NA  NA  NA
5   1 20.4   1 113
6   3   NA  NA 184
```

5.2　数据清理

除了个别小规模案例，通常数据收集过程不可能保证十分完美。无论多么小心地收集数据，误差不可能完全消除，进行分析之前总要考察一下数据质量。即使对于已经在提高数据质量方面花费了很大努力的数据库，数据中不和谐的部分仍然会存在。经常考察数据质量十分重要，并且要在这方面实时留心。

数据中出现问题的原因多种多样，最经常发生的就是人为输入的错误，比如，小数点输入错误，会把￥150.00 变成￥150000；计算和测量过程中也会存在固有误差；除此之外，外界的一些因素也会导致产生误差。

数据清理是目前数据挖掘研究中一项重要的任务。我们通常在建立模型之前完成数据清理，但对于数据结构的探索和模型的描述及预测会促使我们不断检查数据质量，特别当模型出现预料之外的异常情况时，回过头去检查数据质量十分重要。

一些简单的步骤可以用来对数据质量进行复审。

对于探索性数据，我们通常使用统计图表来探索数据规律。

例如，为了获得收入超过 10 万元的经理的收入情况相关信息，我们对 pay 中 66 个数据进行探索性数据分析，并且分别用直方图、点图、箱形图和 Q-Q 图表示。

```
> pay=c(11,19,14,22,14,28,13,81,12,43,11,16,31,16,23,42,22,26,17,22,13,27,180,16,
  43,82,14,11,51,76,28,66,29,14,14,65,37,16,37,35,39,27,14,17,13,38,28,40,85,32,
  25,26,16,12,54,40,18,27,16,14,33,29,77,50,19,34)# 年薪超过 10 万元的经理收入( 单位为 10 万 )
```

```
> par(mfrow=c(2,2))                    # 将绘图窗口划成 2*2，可同时显示 4 幅图
> hist(pay)                            # 绘制直方图
> dotchart(pay)                        # 绘制点图
> boxplot(pay,horizontal=T)            #绘制箱形图
> qqnorm(pay);qqline(pay)              #绘制 Q-Q 图
```

绘图结果如图 5-1 所示，从这 4 张图中均可发现，第 23 个观测值 180 远离其他变量，可以认为数据 pay 中存在异常值，即第 23 个观测值，需要对异常值进行清理。

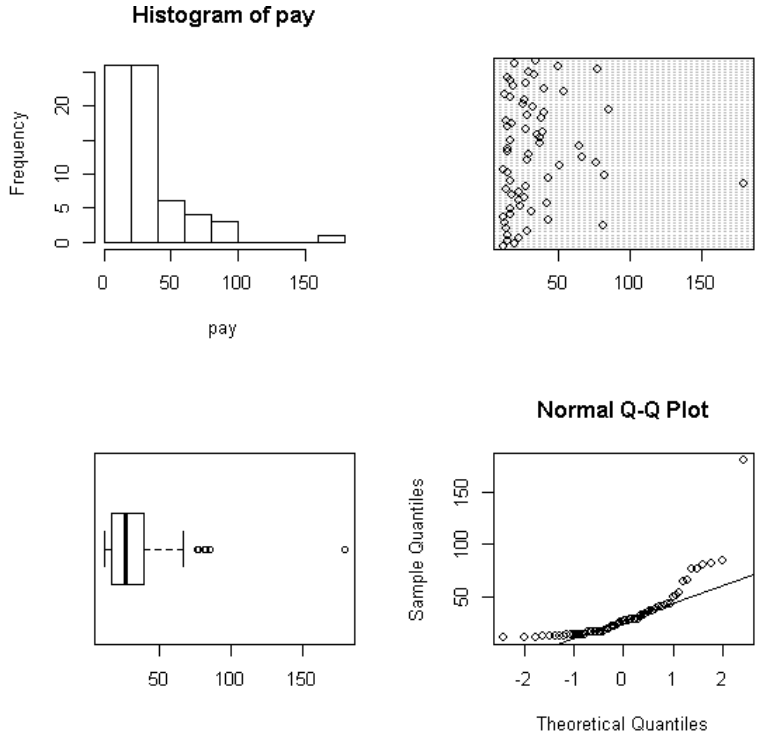

图 5-1 pay 中 66 个数据的分布图

（依次为直方图、点图、箱式图和 Q-Q 图）

在探索过程中，出现任何异常的情况都应该进行解释和处理。例如，对于分类变量，我们应该关注那些频率特别低的类别，这些可能是错误分类或原本应该属于相邻类别的数据，此时就应该回归数据来分析该样本取值是否出现勘误。

例如，在整合来自不同政府机构和公共资源的金融和商务数据时，数据清理的一个主要任务是对姓名和住址的整理，这点在把多个不同资源的数据整合成一个数据集时显得尤为重要。通过一个人的姓名可能会查到很多不同的记录，甚至可能出现 20 到 30 条，进而通过街道地址才能够

进行判断。很多机构都在努力进行数据清理，同时也有很多方法被开发出来以解决数据问题。

5.2.1 缺失值处理

缺失值是数据中经常出现的问题，也是任何数据集中都可能出现的问题，无回答、录入错误等调查中常会出现的现象都会导致缺失数据。缺失值通常会用一些特殊符号进行标记，比如 9999、1990 年 1 月 1 日，或者是“*”、“？”、“#”、“$”等符号。

缺失数据会影响分析工作的进行，进而影响统计工作的效率，还会导致分析的偏差。数据使用者、分析者往往缺乏缺失值处理方面的知识，仅仅对数据进行简单删除或插补会影响数据规模和数据结构，进而影响分析结果。

在数据预处理中，首先要做的通常是判断是否存在缺失值。在 R 语言中缺失值通常以 NA 表示，可以使用函数 is.na 判断缺失值。

```
> sum(is.na(nhanes2))                          #计算 nhanes2 中缺失值的数量
[1] 27
```

另一个常用到的函数是 complete.cases，用来判断某一观测样本是否完整。

```
> sum(complete.cases(nhanes2))                 #计算 nhanes2 中完整样本的数量
[1] 13
```

两个函数结果分别表示“数据中共有 27 个缺失值”以及“数据框中共有 13 条完整观测值”，即有 12 条观测值中存在缺失值。

在存在缺失数据的情况下，需要进一步对数据缺失状况进行观测，判断缺失数据是否随机。我们可以利用 mice 包中的 md.pattern 函数完成这一任务。

```
>md.pattern(nhanes2)  #观测 nhanes2 中缺失值的情况
     age   hyp   bmi   chl
13   1     1     1     1     0
 1   1     1     0     1     1
 3   1     1     1     0     1
 1   1     0     0     1     2
 7   1     0     0     0     3
     0     8     9     10    27
```

其中 1 表示没有缺失数据，0 表示存在缺失数据。第一行第一列的 13 表示有 13 个样本是完整的，即不存在缺失数据。第一列最后一个 7 表示有 7 个样本少了 hyp、bmi、chl 三个变量，最后一行表示各个变量缺失的样本数合计。

对于缺失数据最简单粗暴的方式，即是直接删除含有缺失值的样本，这样做有时也是最为简单有效的方法，但前提是缺失数据的比例较少，且缺失数据是随机出现的，这样删除缺失数据后对分析结果影响不大。

另外，也可采取用变量均值或中位数来代替缺失值的方式，其优点在于不会减少样本信息，处理简单；其缺点在于当缺失数据不是随机出现时会产成偏误。

而多重插补法（Multivariate Imputation）通过变量间的关系对缺失数据进行预测，利用蒙特卡洛方法生成多个完整的数据集，再对这些数据集分别进行分析，最后对分析结果进行汇总处理。

在 R 语言中，可以通过调用 mice 包中的 mice 函数实现，函数的基本形式是：

```
mice(data, m = 5,...)
```

其中，data 代表一个有缺失值的数据框或矩阵，缺失值用 NA 表示；m 表示插补重数，即生成 m 个完整数据集，默认值为 m=5。

举个例子，若要构建以 chl 为因变量，age、hyp、bmi 为自变量的线性回归模型，因为 nhanes 2 数据中存在缺失值，不能直接用来构建模型，因此可以利用 mice()函数，通过如下方式构建模型：

```
> imp=mice(nhanes2,m=4)                    # 生成 4 组完整的数据库并赋给 imp
> fit=with(imp,lm(chl~age+hyp+bmi))        # 生成线性回归模型
> pooled=pool(fit)                         # 对建立的 4 个模型进行汇总
> summary(pooled)                          # 展示 pooled 的内容
```

生成多个完整数据集储存于 imp 中，再对 imp 进行线性回归，最后用 pool 函数对回归结果进行汇总。

```
             Est        se         t            df        Pr(>|t|)    lo 95
(Intercept) -2.810786  70.559221  -0.03983584  5.806503  0.96955821  -176.8672775
age2        43.791589  26.500764  1.65246513   5.128217  0.15788511  -23.8216782
age3        78.743815  30.233726  2.60450247   5.283178  0.04552407  2.2619365
hyp2        -13.339990 27.687074  -0.48181291  4.627705  0.65185343  -86.2690193
bmi         6.327635   2.586265   2.44663041   4.932228  0.05885762  -0.3481417
hi 95 nmis    fmi    lambda
(Intercept) 171.24571   NA     0.5830684     0.4605586
age2        111.40486   NA     0.6259366     0.5038576
age3        155.22569   NA     0.6157695     0.4934649
hyp2        59.58904    NA     0.6604248     0.5397450
bmi         13.00341    9      0.6391340     0.5174711
```

汇总结果的前面部分和普通回归结果相似，nmis 表示了变量中的缺失数据个数，fmi 表示 fraction of missing information，即由缺失数据贡献的变异。

在对是否存在缺失值进行判断后，我们将对缺失值的处理方法进行讲解。

1．删除法

在不影响数据结构的情况下，删除法是最简单的将缺失数据集转变成完整数据集的方法。根据数据处理的不同角度，可以将删除法分为以下 4 种。

（1）删除观测样本。

（2）删除变量：当某个变量缺失值较多且对研究目标影响不大时，可以将整个变量整体删除。

（3）使用完整原始数据分析：当数据存在较多缺失而其原始数据完整时，可以使用原始数据替代现有数据进行分析。

（4）改变权重：当删除缺失数据会改变数据结构时，通过对完整数据按照不同的权重进行加权，可以降低删除缺失数据带来的偏差。

2．插补法

删除数据虽然简单易行，但会带来信息浪费、改变数据结构等问题，因此在条件允许的情况下，找到缺失值的替代值来进行插补，尽可能还原真实数据是更好的方法。

下面介绍均值插补、回归插补、二阶插补、热平台、冷平台、抽样填补等单一变量插补，多变量插补是单变量插补的推广，请读者自行尝试。

在插补方法中，最简单的是从总体中随机抽取某个样本代替缺失样本。

R 程序如下：

```
> sub=which(is.na(nhanes2[,4])==TRUE)        # 返回 nhanes2 数据集中第 4 列为 NA 的行
> dataTR=nhanes2[-sub,]                      # 将第 4 列不为 NA 的数存入数据集 dataTR 中
> dataTE=nhanes2[sub,]                       # 将第 4 列为 NA 的数存入数据集 dataTE 中
> dataTE[,4]=sample(dataTR[,4],length(dataTE[,4]),replace=T)# 在非缺失值中简单抽样
> dataTE
    age      bmi      hyp      chl
1   20-39    NA       <NA>     206
4   60-99    NA       <NA>     204
10  40-59    NA       <NA>     206
11  20-39    NA       <NA>     113
12  40-59    NA       <NA>     229
15  20-39    29.6     no       199
16  20-39    NA       <NA>     218
20  60-99    25.5     yes      187
21  20-39    NA       <NA>     199
24  60-99    24.9     no       238
```

均值法是通过计算缺失值所在变量所有非缺失观测值的均值，使用均值来代替缺失值的插补方法。

类似的，可以使用中位数、四分位数等进行插补，下面仅以均值法为例来对 nhanes2 数据集

的第 4 列进行实现。

```
> sub=which(is.na(nhanes2[,4])==TRUE)      # 返回 nhanes2 数据集中第 4 列为 NA 的行
> dataTR=nhanes2[-sub,]                    # 将第四列不为 NA 的数存入数据集 dataTR 中
> dataTE=nhanes2[sub,]                     # 将第四列为 NA 的数存入数据集 dataTE 中
> dataTE[,4]=mean(dataTR[,4])              # 用非缺失值的均值代替缺失值
> dataTE
    age      bmi      hyp      chl
1   20-39    NA       <NA>     191.4
4   60-99    NA       <NA>     191.4
10  40-59    NA       <NA>     191.4
11  20-39    NA       <NA>     191.4
12  40-59    NA       <NA>     191.4
15  20-39    29.6     no       191.4
16  20-39    NA       <NA>     191.4
20  60-99    25.5     yes      191.4
21  20-39    NA       <NA>     191.4
24  60-99    24.9     no       191.4
```

由于随机插补和均值插补中没有利用到相关变量信息，因此会存在一定偏差，而回归模型是将需要插补变量作为因变量，其他相关变量作为自变量，通过建立回归模型预测出因变量的值对缺失变量进行插补。

```
> sub=which(is.na(nhanes2[,4])==TRUE)      # 返回 nhanes2 数据集中第 4 列为 NA 的行
> dataTR=nhanes2[-sub,]                    # 将第 4 列不为 NA 的数存入数据集 dataTR 中
> dataTE=nhanes2[sub,]                     # 将第 4 列为 NA 的数存入数据集 dataTE 中
> dataTE                                   # dataTE 数据
    Age      bmi      hyp      chl
1   20-39    NA       <NA>     NA
4   60-99    NA       <NA>     NA
10  40-59    NA       <NA>     NA
11  20-39    NA       <NA>     NA
12  40-59    NA       <NA>     NA
15  20-39    29.6     no       NA
16  20-39    NA       <NA>     NA
20  60-99    25.5     yes      NA
21  20-39    NA       <NA>     NA
24  60-99    24.9     no       NA
> lm=lm(chl~age,data=dataTR)
                    # 利用 dataTR 中 age 为自变量，chl 为因变量构建线性回归模型 lm
> nhanes2[sub,4]=round(predict(lm,dataTE))
                    # 利用 dataTE 中数据按照模型 lm 对 nhanes2 中 chl 中的缺失数据进行预测
```

```
> head(nhanes2)       # 缺失值处理后的 nhanes2 的前若干条
    age      bmi      hyp      chl
1   20-39    NA       <NA>     169
2   40-59    22.7     no       187
3   20-39    NA       no       187
4   60-99    NA       <NA>     225
5   20-39    20.4     no       113
6   60-99    NA       <NA>     184
```

热平台插补是指在非缺失数据集中找到一个与缺失值所在样本相似的样本（匹配样本），利用其中的观测值对缺失值进行插补。

```
> accept=nhanes2[which(apply(is.na(nhanes2),1,sum)!=0),]          # 存在缺失值的样本
> donate=nhanes2[which(apply(is.na(nhanes2),1,sum)==0),]          # 无缺失值的样本
> accept[1,]
    age      bmi      hyp      chl
1   20-39    NA       <NA>     NA
> donate[1,]
    age      bmi      hyp      chl
2   40-59    22.7     no       187
```

上述程序按照样本中是否含有缺失值将 nhanes2 分成存在缺失值和无缺失值两个数据表，分别命名为 accept 和 donate。对于 accept 中的每个样本，热平台插补就是在 donate 中找到与该样本相似的样本，用相似样本的对应值代替该样本的缺失值。如，对于 accept 中的第 2 个样本，插补方法如下：

```
> accept[2,]
    age      bmi      hyp      chl
3   20-39    NA       no       187
> sa=donate[which(donate[,1]==accept[2,1]&donate[,3]==accept[2,3]&donate[,4]==
  accept[2,4]),]                          # 寻找与 accept 中第 2 个样本相似的样本
> sa
    age      bmi      hyp      chl
8   20-39    30.1     no       187
> accept[2,2]=sa[1,2]                     # 用找到的样本的对应值替代缺失值
> accept[2,]
    age      bmi      hyp      chl
3   20-39    30.1     no       187
```

在实际操作中，尤其当变量数量很多时，通常很难找到与需要插补样本完全相同的样本，此时可以按照某些变量将数据分层，在层中对缺失值使用均值插补，即采取冷平台插补方法。

```
> level1=nhanes2[which(nhanes2[,3]=="yes"),]          # 按照变量 hyp 分层
> level1
    age      bmi      hyp      chl
```

```
14   40-59    28.7     yes      204
17   60-99    27.2     yes      284
18   40-59    26.3     yes      199
20   60-99    25.5     yes      NA
> level1[4,4]=mean(level1[1:3,4])                    # 用层内均值代替第 4 个样本的缺失值
> level1
    age      bmi      hyp      chl
14   40-59    28.7     yes      204
17   60-99    27.2     yes      284
18   40-59    26.3     yes      199
20   60-99    25.5     yes      229
```

5.2.2　噪声数据处理

噪声是一个测量变量中的随机错误或偏差，包括错误值或偏离期望的孤立点值。在 R 中可以通过调用 outliers 软件包中的 outlier 函数寻找噪声数据，该函数通过寻找数据集中与其他观测值及均值差距最大的点作为异常值，函数的主要形式为：

```
outlier(x, opposite = FALSE, logical = FALSE)
```

其中，x 表示一个数据，通常是一个向量，如果 x 输入的是一个数据框或矩阵，则 outlier 函数将逐列计算；opposite 可输入 TRUE 或者 FALSE，如果值为 TRUE，给出相反值（如果最大值与均值差异最大，则给出最小值）；logical 可输入 TRUE 或者 FALSE，如果值为 TRUE，给向量赋予逻辑值，可能出现噪声的位置用 TRUE 表示。

```
> library(outliers)
> set.seed(1); s1=.Random.seed                        # 设置随机数种子，保证每次出现的随机数相同
> y=rnorm(100)                                        # 生成 100 个标准正态随机数
> outlier(y)                                          # 找出其中离群最远的值
[1] -2.2147
> outlier(y,opposite=TRUE)                            # 找出最远离群值相反的值
[1] 2.401618
> dotchart(y)                                         # 对 y 绘制点图，如图 5-2 所示
```

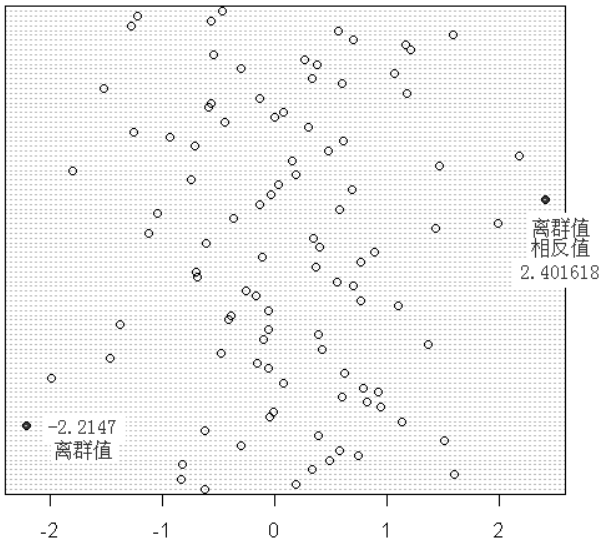

图 5-2 离群值点图

```
> dim(y) <- c(20,5)                    # 将 y 中的数据重新划分成 20 行 5 列的矩阵
> outlier(y)                           # 求矩阵中每列的离群最远值
[1] -2.214700 -1.989352  1.980400  2.401618 -1.523567
> outlier(y,opposite=TRUE)             # 求矩阵中每列的离群最远值的相反值
[1]  1.595281  1.358680 -1.129363 -1.804959  1.586833
> set.seed(1); s1=.Random.seed         # 设置随机数种子，保证每次出现的随机数相同
> y=rnorm(10)                          # 生成 10 个标准正态随机数
> outlier(y,logical=TRUE)              # 返回相应逻辑值，离群最远值用 TRUE 标记
 [1] FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
> plot(y)                              # 绘制散点图，如图 5-3 所示
```

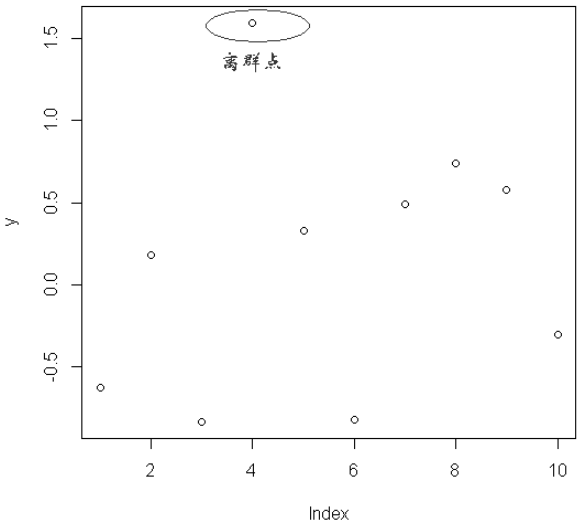

图 5-3　标记出离群值的散点图

离群点检测还可以通过聚类方法进行检测，聚类将类似的取值组织成“群”或“簇”，落在“簇”集合之外的值被视为离群点。聚类方法将在第 7 章进行详细阐述。

在进行噪声检查后，操作实际中常用分箱、回归、计算机检查和人工检查结合等方法“光滑”数据，去掉数据中的噪声。

分箱方法是通过对数据进行排序，利用数据“近邻”来光滑有序数据值的一种局部光滑方法。在分箱方法中，可以使用箱均值、箱中位数或箱边界等进行光滑。箱均值光滑、箱中位数光滑分别为对于每个“箱”，使用其均值或中位数来代替箱中的值；而箱边界光滑则是指将给定箱中的最大值和最小值被视为箱边界，箱中每一个值都被替换为最近边界。一般而言，宽度越大，光滑效果越明显。箱可以是等宽的，即每个箱的区间范围是常量。

下面以等宽箱均值光滑方法为例来介绍。

```
> set.seed(1); s1=.Random.seed          # 设置随机数种子，保证每次出现的随机数相同
> x=rnorm(12)                           # 生成 12 个标准正态随机数
> x=sort(x)                             # 将数据从小到大排序
> dim(x)=c(3,4)                         # 将数据形式转换成 3 行 4 列矩阵，每行代表一个箱
> x[1,]=apply(x,1,mean)[1]              # 用第一行的均值代替第一行中的数据
> x[2,]=apply(x,1,mean)[2]              # 用第二行的均值代替第二行中的数据
> x[3,]=apply(x,1,mean)[3]              # 用第三行的均值代替第三行中的数据
> x                                     # 等宽分箱均值光滑结果
     [,1]          [,2]          [,3]          [,4]
```

```
[1,] -0.003212265    -0.003212265     -0.003212265     -0.003212265
[2,]  0.340596290    0.340596290      0.340596290      0.340596290
[3,]  0.46852902     0.468529029      0.468529029      0.468529029
```

回归是指通过一个函数拟合来对数据进行光滑处理。线性回归涉及找出拟合两个变量的“最佳”直线，使得一个属性可以用来预测另一个；多元线性回归是线性回归的扩充，其中涉及的属性多于两个，并且数据拟合到一个多维曲面。

5.2.3 数据不一致的处理

作为一位数据分析人员，应当警惕编码使用的不一致问题和数据表示的不一致问题（如日期“2004/12/25”和“25/12/2004”）。字段过载是另一种错误源，通常由如下原因导致：开发者将新属性的定义挤压到已经定义的属性的未使用（位）部分（例如，使用一个属性未使用的位，该属性取值已经使用了 32 位中的 31 位）。

编码不一致和数据表示不一致的问题通常需要人工检测，当发现一定规律时可以通过编程进行替换和修改。若存在不一致的数据是无意义数据，可以使用缺失值处理方法进行相应处理。

当对数据进行批量操作时，可以通过对函数返回值进行约束，根据是否提示错误判断、是否存在数据不一致问题，如 vapply 函数。

vapply 函数的作用是对一个列表或向量进行指定的函数操作，其常用格式如下：

```
vapply(X, FUN, FUN.VALUE, ..., USE.NAMES = TRUE)
```

其中 X 是作为输入变量的列表或向量，FUN 是指定函数，FUN.VALUE 是函数要求的返回值，当 USE.NAMES 赋值为 TRUE 且 X 是字符型时，若返回值没有变量名则用 X 作为变量名。

与 vapply 类似的函数还有 lapply 和 sapply，sapply 是 lapply 的友好版本，但可预测性不好。如果是大规模的数据处理，后续的类型判断工作会很麻烦而且很费时。vapply 增加的 FUN.VALUE 参数可以直接对返回值类型进行检查，这样的好处是不仅运算速度快，而且程序运算更安全（因为结果可控）。

下面代码中的 rt.value 变量设置返回值的长度和类型，如果 FUN 函数获得的结果和 rt.value 设置的不一致（长度和类型）都会出错：

```
> x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))# 生成列表
> x
$a
 [1]  1  2  3  4  5  6  7  8  9 10
$beta
[1]  0.04978707  0.13533528  0.36787944  1.00000000  2.71828183  7.38905610
[7] 20.08553692
$logic
[1]  TRUE FALSE FALSE  TRUE
```

```
> probs <- c(1:3/4)
> rt.value <- c(0,0,0)                          # 设置返回值为 3 个数字
> vapply(x, quantile,FUN.VALUE=rt.value,probs=probs)
    a    beta        logic
25% 3.25 0.2516074   0.0
50% 5.50 1.0000000   0.5
75% 7.75 5.0536690   1.0
```

若将 probs <- c(1:3/4) 改成 probs <- c(1:4/4)，会导致返回值与要求格式不一致，进而提示错误。

```
> probs<- c(1:4/4)                              #设置四个分为点
> vapply(x, quantile,FUN.VALUE=rt.value,probs=probs)
Error in vapply(x, quantile, FUN.VALUE = rt.value, probs = probs) :
  values must be length 3,
but FUN(X[[1]]) result is length 4
```

结果显示错误，要求返回值的长度必须为 3，但 FUN(X[[1]])返回的结果长度却是 4，两者不一致导致错误。将要求值长度改为 4，则

```
>rt.value<- c(0,0,0,0)                         # 设置返回值为 4 个数字
>vapply(x, quantile,FUN.VALUE=rt.value,probs=probs)
      a      beta         logic
25%   3.25   0.2516074    0.0
50%   5.50   1.0000000    0.5
75%   7.75   5.0536690    1.0
100%  10.00  20.0855369   1.0
>rt.value<- c(0,0,0,"")                        # 设置返回值为 3 个数字和 1 个字符串
>vapply(x, quantile,FUN.VALUE=rt.value,probs=probs)
Error in vapply(x, quantile, FUN.VALUE = rt.value, probs = probs) :
values must be type 'character',
but FUN(X[[1]]) result is type 'double'
```

由于要求返回值的种类必须是'character'，但 FUN(X[[1]])结果的种类却是'double'，导致产生错误提示。

因此可以根据 vapply 函数的这一功能，使用 FUN.VALUE 参数对数据进行批量检测。

5.3 数据集成

数据集成是指将多个数据源中的数据合并，并存放到一个一致的数据存储（如数据仓库）中。这些数据源可能包括多个数据库、数据立方体或一般文件。

在数据集成时，首先要考虑如何对多个数据集进行匹配，数据分析者或计算机需要识别出能够连接两个数据库的实体信息。例如，要判断一个数据库中的 ID 与另一个数据库中的 NO.是否是相同属性，需要对这两个属性的名字、含义、数据类型和取值范围以及处理空白、零或 NULL 值的空值规则进行比较，这样可以减少模式集成的错误。

在集成期间，当一个数据库的属性与另一个数据库的属性匹配时，必须特别注意数据的结构。这旨在确保源系统中的函数依赖和参照约束与目标系统中的匹配。

例如，在一个系统中，discount 可能用于订单，而在另一个系统中，它用于订单内的商品。如果在集成之前未发现，则目标系统中的商品可能被不正确地打折。

冗余是数据集成的另一个重要问题。两个数据集有两个命名不同但实际数据相同的属性，那么其中一个属性就是冗余的。另外，一个属性若可以通过另一个属性的一定变换得出，那么其中一个属性就可能是冗余的。

可以用相关分析对数据集冗余进行检测。给定两个属性，这种分析可以根据可用的数据，度量一个属性能在多大程度上包含另一个。对于定性数据，可以使用卡方检验，对于定量数据，我们使用相关系数和协方差，它们都能评估一个属性的值如何随另一个变化。

```
> x=cbind(sample(c(1:50),10),sample(c(1:50),10))
                                          # 生成由两列不相关的定性数据组成的矩阵 x
> chisq.test(x)                           # 对矩阵 x 进行卡方检验，检查两列是否相关
  Pearson's Chi-squared test
  data:  x
  X-squared =74.4563,df= 9, p-value = 2.023e-12
```

上述定性数据卡方检验结果显示，p-value<0.05，即在 0.05 显著性水平下拒绝相关的原假设，即变量不相关。

对于定量数据的相关系数，若两个属性的相关系数为 0，表示两个属性独立，并且它们之间不存在相关性，如果该结果小于 0，则两个属性存在负相关，大于 0 则表示存在正相关。相关系数绝对值越大，相关性越强。因此，若两个属性之间存在较大的相关系数，则其中一个可以被视作冗余而删除。

协方差为正，表示两个属性趋向于一起改变，即一个属性的某个观测值如果大于期望，则另一个属性的对应观测值很可能大于其期望。协方差为负，则表示两个属性趋向于相反方向改变。

相关系数和协方差的 R 实现如下：

```
> x=cbind(rnorm(10),rnorm(10))      # 生成由 2 列标准正态随机数组成的矩阵 x
> cor(x)                            # 求两列数据的相关系数
          [,1]      [,2]
[1,] 1.0000000 0.2058453
[2,] 0.2058453 1.0000000
> cov(x)                            # 求两列数据的协方差
          [,1]      [,2]
[1,] 1.0958683 0.2199514
[2,] 0.2199514 1.0418691
```

由上述结果可以看出，两列数据的相关系数为 0.2058453，可以认为相关系数较小，两列之间的相关性不足以将两列数据视为冗余数据，无须删除。从协方差结果可以看出，两列协方差为正，两列趋于一同改变。

除了检测属性间的冗余外，还应该检测观测值是否存在重复。

```
> x=cbind(sample(c(1:10),10,replace=T),rnorm(10),rnorm(10))
    #随机生成数据集，其中第一列为样本编号，若样本编号相同则认为存在重复
> head(x)                           # 去掉重复值前的数据集前若干个观测值
       [,1]  [,2]          [,3]
[1,]    5     -0.1134210    -1.3028591
[2,]    2     -0.3292809    -0.2786583
[3,]    3     0.3734610     -0.1847539
[4,]    10    1.0298881     -0.0817775
[5,]    10    2.7057750     1.6122418
[6,]    5     -1.0349459    -0.4619398
> y=unique(x[,1])                   # 将样本编号去掉重复
> sub=rep(0,length(y))              # 生成列向量备用
> for(i in 1:length(y))             # 循环，根据样本编号筛选数据集，去掉重复观测值
+ sub[i]=which(x[,1]==y[i])[1]
> x=x[sub,]
> head(x)                           # 去掉重复值后的数据集前若干个观测值
       [,1]  [,2]          [,3]
[1,]    5     -0.1134210    -1.3028591
[2,]    2     -0.3292809    -0.2786583
[3,]    3     0.3734610     -0.1847539
[4,]    10    1.0298881     -0.0817775
[5,]    1     1.0777713     0.4964331
```

数据集还涉及数据值冲突的检测与处理。例如，对于现实世界的同一实体，来自不同数据源的属性值可能不同。这可能是因为表示、尺度或编码不同。

例如，重量属性可能在一个系统中以公制单位存放，而在另一个系统中以英制单位存放。对于连锁旅馆，不同城市的房价不仅可能涉及不同的货币，而且可能涉及不同的服务（如免费早餐）和税收。

再比如，不同学校交换信息时，每个学校可能都有自己的课程计划和评分方案。一所大学可能采取学季制，开设 3 门数据库系统课程，用 A+～F 评分；而另一所大学可能采用学期制，开设两门数据库课程，用 1～10 评分。很难在这两所大学之间制定精确的课程成绩变换规则，这使得信息交换非常困难。

对于数据值冲突的检测与处理可以参照 5.2.3 节中数据不一致的处理方法。

5.4 数据变换

在数据变换中，数据被变换成适应于数据挖掘需求的形式，数据变换策略主要包括以下几种。

1．光滑：去掉数据中的噪声，可以通过分箱、回归和聚类等技术实现，具体内容及 R 实现参见 5.2.2 节。

2．属性构造：由给定的属性构造出新属性并添加到数据集中。例如，通过“销售额”和“成本”构造出“利润”，只需要对相应属性数据进行简单变换即可。

3．聚集：对数据进行汇总。如，可以通过日销售数据，计算月和年的销售数据。

4．规范化：把数据单按比例缩放，实质落入一个特定的小区间，如–1.0~1.0 或 0.0~1.0。标准化是比较常用的一种规范化方法，其 R 实现如下：

```
> set.seed(1); s1=.Random.seed      # 设置随机数种子，保证每次出现的随机数相同
> a=rnorm(5)                        # 生成一列随机数
> b=scale(a)                        # 对该列随机数标准化
> b
           [,1]
[1,]  0.3126208
[2,]  0.7162939
[3,] -0.8242656
[4,]  1.6028730
[5,] -0.3749342
attr(,"scaled:center")
[1] 0.2909254
attr(,"scaled:scale")
[1] 1.289492
```

上述 b 中矩阵为标准化后的数据，attr(,"scaled:center")是原数据的均值，attr(,"scaled:scale")是原数据的标准差。

5．离散化：数值属性（例如，年龄）的原始值用区间标签（例如，0～10、11～20 等）或概念标签（例如，youth、adult、senior）替换。可以实现将定量数据向定性数据转化，将连续型数据离散化。

离散化多根据数据情况和分析需求的不同采用不同的划分方式，假设 a 是一组 Logistic 回归的预测值，是取值在 0～1 之间的连续性数据，需要将 a 转换成取值为 0 或 1 的离散型数据。

```
> a=c(0.7063422,0.7533599,0.6675749,0.6100253,0.9341495,0.6069284,0.3462011)
> n=length(a)
> la=rep(0,n)
> la[which(a>0.5)]=1
[1] 1 1 1 1 1 1 0
```

6．由标称数据产生概念分层：属性，如 street，可以泛化到较高的概念层，如 city 或 country。许多标称属性的概念分层都蕴含在数据库的模式中，可以在模式定义级自动定义。

数据泛化可以理解为数据合并，以城市为例，1 表示沈阳、2 表示大连、3 表示盘锦、4 表示抚顺、5 表示广州、6 表示深圳、7 表示珠海、8 表示佛山，可以通过数据合并，将 1、2、3、4 合并为辽宁省，5、6、7、8 合并为广东省。

```
> city=c(6,7,6,2,2,6,2,1,5,7,2,1,1,6,1,3,8,8,1,1)
> province=rep(0,20)
> province[which(city>4)]=1
> province
 [1] 1 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0          #0 表示辽宁省，1 表示广东省
```

5.5　数据归约

数据归约主要是为了压缩数据量，原数据可以用来得到数据集的归约表示，它接近于保持原数据的完整性，但数据量比原数据小得多，与非归约数据相比，在归约的数据上进行挖掘，所需的时间和内存资源更少，挖掘将更有效，并产生相同或几乎相同的分析结果。常用维归约、数值归约等方法实现。

维归约指通过减少属性的方式压缩数据量，通过移除不相关的属性，可以提高模型效率。维归约的方法很多，其中：AIC 准则可以通过选择最优模型来选择属性；LASSO 通过一定约束条件选择变量；分类树、随机森林通过对分类效果的影响大小筛选属性；小波变换、主成分分析通过把原数据变换或投影到较小的空间来降低维数。

AIC 准则是赤池信息准则的简称，通常用来评价模型的复杂度和拟合效果，其计算公式为：

$$AIC=-2\ln(L)+2k$$

其中 L 为似然函数，代表模型的精确度，k 为参数的数量，意味着模型的准确性。当 L 越大

时，模型拟合效果越精确，当 k 越小时，模型越简洁，因此 AIC 兼顾了模型的精确度和简洁性，适合用来对模型进行选择。

使用 AIC 准则进行模型变量选择时，AIC 最小的模型即为最优。

下面我们以 LASSO 为例对其维归约进行阐述。

在 R 中可以使用 glmnet 程序包中的 glmnet()函数实现对不同分布数据进行 LASSO 变量选择，其中：

```
> x=matrix(rnorm(100*20),100,20)        # 生成自变量，为 20 列正态随机数
> y=rnorm(100)                          # 生成一列正态随机数作为因变量
> fit1=glmnet(x,y)                      # 广义线性回归，自变量未分组的，默认为 LASSO
> b=coef(fit1,s=0.01)                   # s 代表λ值，随着λ减小，约束放宽，筛选的变量越多
> b                                     # b 代表变量系数，有值的被选入模型
21 x 1 sparse Matrix of class "dgCMatrix"
1
(Intercept)  0.01637335
V1           0.08325099
V2          -0.02009427
V3          -0.05482563
V4           .
V5           0.11101047
V6          -0.12924568
V7          -0.04121713
V8           .
V9           0.18190221
V10          0.07657682
V11          0.04978051
V12         -0.01395913
V13         -0.01609426
V14         -0.03785605
V15          0.17685794
V16          .
V17         -0.22528254
V18          .
V19          0.07914728
V20          0.07596220
> predict(fit1,newx=x[1:10,],s=c(0.01,0.005)) #λ分别为 0.01 和 0.005 情况下的预测值
       1           2
 [1,]  0.02427191  0.03239116
 [2,] -0.20064653 -0.21305735
 [3,]  0.22854808  0.24957546
 [4,]  0.68076151  0.69852322
```

```
 [5,]  0.13732179  0.12244061
 [6,]  0.36537375  0.36419868
 [7,]  0.83014862  0.84884258
 [8,]  0.29779430  0.30042436
 [9,] -0.08998970 -0.07296359
[10,]  0.17817755  0.15510863
```

对于 LASSO 方法，随着 λ 的减小，约束放松，进入模型的变量增多，当模型拟合值与惩罚函数之和最小时对应的 λ 选择的变量即为最能代表数据集的变量。

数值归约是指用较小的数据表示形式替换原数据。如参数方法中使用模型估计数据，就可以只存放模型参数代替存放实际数据，如回归模型和对数线性模型都可以用来进行参数化数据归约。对于非参数方法，可以使用直方图、聚类、抽样和数据立方体聚集当方法。

有许多其他方法来组织数据归约方法。花费在数据归约上的计算时间不应超过或“抵消”在归约后的数据上挖掘所节省的时间。

5.6　本章汇总

chisq.test()	函数	定性变量相关性检验
complete.cases()	函数	判断是否存在完整观测样本
glmnet	软件包	提供函数 glmnet()
glmnet()	函数	广义线性回归，可以进行 LASSO 模型选择
is.na()	函数	判断是否存在缺失值
lm()	函数	构建线性回归模型
md.pattern()	函数	观测数据集中缺失值分布情况
mice()	函数	对缺失数据进行多重插补
mice	软件包	提供函数 md.pattern()、mice()及 nhanes2 数据集
nhanes2	数据集	由 mice 软件包提供
Outlier()	函数	寻找样本中离群最远的值
outliers	软件包	提供函数 outlier()
predict()	函数	预测
round()	函数	四舍五入求整
scale()	函数	对数据进行标准化处理
vapply()	函数	对列表或向量进行指定函数操作，可用来检测数据是否不一致