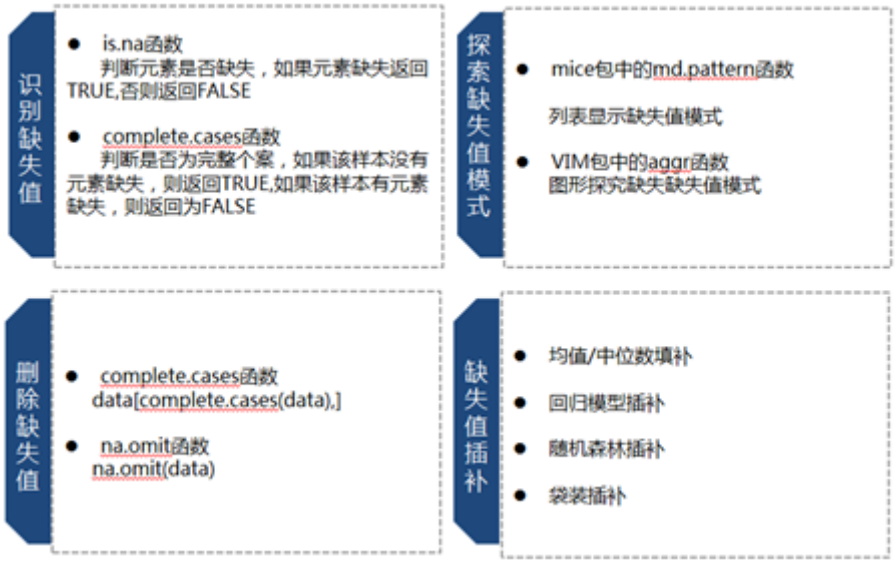
R语言数据挖掘之数据探索

数据质量分析是数据预处理的前提，是数据挖掘分析结论有效性和准确性的基础，其主要任务是检查原始数据中是否存在脏数据，脏数据一般是指不符合要求，以及不能直接进行相应分析的数据。在常见的数据挖掘工作中，脏数据主要值缺失值和异常值。本专场重点介绍了缺失值的判断、缺失值模式探索、缺失值处理及异常值判断的常用方法及R语言实践。
本专题重点学习关于缺失值和异常值的识别和处理技术。

一个完整的处理方法通常包含以下几个步骤

- 识别缺失数据。
- 检查导致数据缺失的原因。
- 删除包含缺失值的实例或用合理的数值代替（插补）缺失值。

下图总结了关于缺失值的判断和处理常用技术。



识别缺失数据

首先，我们先来了解如何识别缺失值。R使用NA（不可得）代表缺失值， NaN（不是一个数）代表不可能值。另外，符号Inf和-Inf分别代表正无穷和负无穷。函数is.na()、is.nan()和is.infinite()可分别用来识别缺失值、不可能值和无穷值。每个返回结果都是TRUE或FALSE。

x	is.na(x)	is.nan(x)	is.infinite(x)
x <- NA	TRUE	FALSE	FALSE
x <- 0 / 0	TRUE	TRUE	FALSE
x <- 1 / 0	FALSE	FALSE	TRUE

这些函数返回的对象与其自身参数的个数相同。若每个元素的类型检验通过，则由TRUE替换，否则用FALSE替换。 例如， 令y <- c(1, 2, 3, NA) ，则is.na(y)返回向量c(FALSE, FALSE,FALSE, TRUE)。

函数complete.cases()可以用来识别矩阵或数据框中没有缺失值的行。若每行都包含完整的实例，则返回TRUE的逻辑向量；若每行有一个或多个缺失值，则返回FALSE。

以VIM包中的睡眠数据集sleep为例进行说明。

```
> # 加载数据集
> data(sleep, package="VIM")
> # 列出没有缺失值的行
> head(sleep[complete.cases(sleep),])
   BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
2    1.000      6.6  6.3   2.0   8.3  4.5   42    3   1      3
5 2547.000   4603.0  2.1   1.8   3.9 69.0  624    3   5      4
6   10.550    179.5  9.1   0.7   9.8 27.0  180    4   4      4
7    0.023      0.3 15.8   3.9  19.7 19.0   35    1   1      1
8  160.000    169.0  5.2   1.0   6.2 30.4  392    4   5      4
9    3.300     25.6 10.9   3.6  14.5 28.0   63    1   2      1
> nrow(sleep[complete.cases(sleep),])
[1] 42
> # 列出有一个或多个缺失值的行
```

```
> head(sleep[!complete.cases(sleep),])
    BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
1  6654.000   5712.0   NA    NA   3.3 38.6  645    3   5      3
3     3.385     44.5   NA    NA  12.5 14.0   60    1   1      1
4     0.920      5.7   NA    NA  16.5   NA   25    5   2      3
13    0.550      2.4  7.6   2.7  10.3   NA   NA    2   1      2
14  187.100    419.0   NA    NA   3.1 40.0  365    5   5      5
19    1.410     17.5  4.8   1.3   6.1 34.0   NA    1   2      1
> nrow(sleep[!complete.cases(sleep),])
[1] 20
```

输出结果显示42个实例为完整数据， 20个实例含一个或多个缺失值。

由于逻辑值TRUE和FALSE分别等价于数值1和0，可用sum()和mean()函数来获取关于缺失数据的有用信息。如：

```
> attach(sleep)
> # 查看变量Dream的缺失个数
> sum(is.na(Dream)) # 等价于sum(!complete.cases(Dream))
[1] 12
> # 查看变量Dream缺失值的占比
> mean(is.na(Dream))
[1] 0.1935484
> # 查看变量Dream非缺失值的占比
> mean(complete.cases(Dream))
[1] 0.8064516
> # 查看数据集sleep中样本有缺失值的占比
> mean(!complete.cases(sleep))
[1] 0.3225806
```


结果表明变量Dream有12个缺失值， 19%的实例在此变量上有缺失值。另外，数据集中32%的实例包含一个或多个缺失值。

对于识别缺失值，有两点需要牢记。第一， complete.cases()函数仅将NA和NaN识别为缺失值，无穷值（Inf和-Inf）被当作有效值。第二，必须使用与本章中类似的缺失值函数来识别R数据对象中的缺失值。像myvar == NA这样的逻辑比较无法实现。

现在你应该懂得了如何用程序识别缺失值，接下来学习一些有助于发现缺失值模式的工具吧。

探索缺失值模式

在决定如何处理缺失数据前，了解哪些变量有缺失值、数目有多少、是什么组合形式等信息非常有用。本节中，我们将介绍探索缺失值模式的图表及相关方法。最后，要知道数据为何缺失，这将为后续深入研究提供许多启示。

列表显示缺失值

mice包中的md.pattern()函数可生成一个以矩阵或数据框形式展示缺失值模式的表格。将函数应用到sleep数据集，可得到：

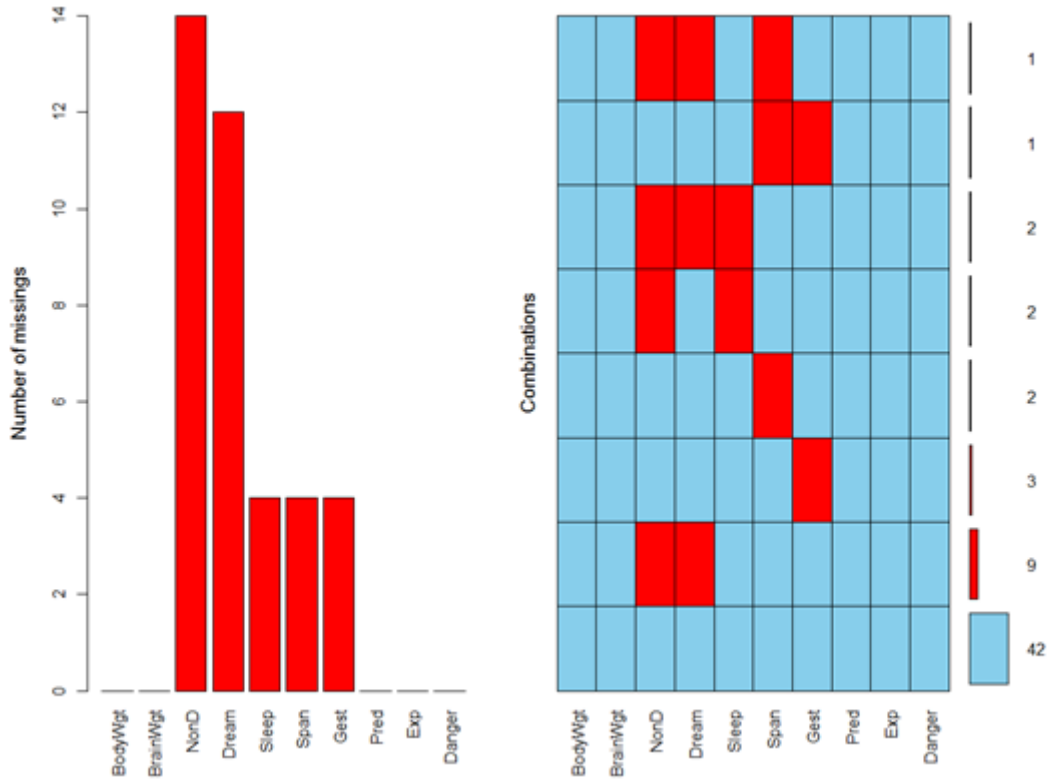
```
> # 探索缺失值模式
> # 列表显示缺失值
> # mice包中的md.pattern()函数
> # 将函数应用到sleep数据集
> library(mice)
载入需要的程辑包：Rcpp
mice 2.25 2015-11-09
Warning message:
程辑包‘Rcpp’是用R版本3.4.0 来建造的
> data(sleep,package="VIM")
> md.pattern(sleep)
   BodyWgt BrainWgt Pred Exp Danger Sleep Span Gest Dream NonD
42       1        1    1   1      1     1    1    1     1    1  0
 2       1        1    1   1      1     1    0    1     1    1  1
 3       1        1    1   1      1     1    1    0     1    1  1
 9       1        1    1   1      1     1    1    1     0    0  2
 2       1        1    1   1      1     0    1    1     1    0  2
 1       1        1    1   1      1     1    0    0     1    1  2
 2       1        1    1   1      1     0    1    1     0    0  3
 1       1        1    1   1      1     1    0    1     0    0  3
         0        0    0   0      0     4    4    4    12   14 38
```


表中的1和0显示了缺失值模式： 0表示变量的列中有缺失值， 1则表示没有缺失值。第一行表述了“无缺失值”的模式（所有元素都为1）。第二行表述了“除了Span之外无缺失值”的模式。第一列表示各缺失值模式的实例个数，最后一列表示各模式中有缺失值的变量的个数。此处可以看到，有42个实例没有缺失值，仅2个实例缺失了Span。9个实例同时缺失了NonD和Dream的值。数据集包含了总共(42×0)+(2×1)+…+(1×3)=38个缺失值。最后一行给出了每个变量中缺失值的数目。

图形探究缺失数据

虽然md.pattern()函数的表格输出非常简洁，但我通常觉得用图形展示模式更为清晰。VIM包提供了大量能可视化数据集中缺失值模式的函数，这边我们介绍aggr函数。aggr()函数不仅绘制每个变量的缺失值数，还绘制每个变量组合的缺失值数。例如：

```
# 图形探究缺失数据
# aggr()函数
library(VIM)
aggr(sleep,prop=FALSE,numbers=TRUE)
```



可以看到，变量NonD有最大的缺失值数（14），有2种哺乳动物缺失了NonD、 Dream和Sleep的评分。 42种动物没有缺失值。代码aggr(sleep, prop=TRUE, numbers=TRUE)将生成相同的图形，但用比例代替了计数。选项numbers=FALSE（默认）删去数值型标签。

删除包含缺失值的实例或用合理的数值代替（插补）缺失值

删除包含缺失值的实例

在完整实例分析中，只有每个变量都包含了有效数据值的观测才会保留下来做进一步的分析。实际上，这样会导致包含一个或多个缺失值的任意一行都会被删除，因此常称作行删除法（listwise）、 个案删除（case-wise）或剔除。大部分流行的统计软件包都默认采用行删除法来处理缺失值，因此许许多多的分析人员在使用诸如回归或者方差分析法来分析数据时，都没有意识到有“缺失值问题”需要处理！

函数complete.cases()可以用来存储没有缺失值的数据框或者矩阵形式的实例（行）：

```
newdata <- mydata[complete.cases(mydata),]
```

同样的结果可以用na.omit函数获得：

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

两行代码表示的意思都是： mydata中所有包含缺失数据的行都被删除，然后结果才存储到
newdata中。

缺失值插补

在数据挖掘中，面对的通常是大型的数据库，它的属性有几十个甚至几百个，因为一个属性值的缺失而放弃大量的其他属性值，这种删除是对信息的极大浪费，最常见的就是通过赋值来解决。用变量均值或中位数来代替缺失值，这样做的优点在于不会减少样本信息，处理起来简单，但缺点在于当缺失数据不是随机出现时会产生偏差。

另一种方法将通过诸如回归、决策树、袋装、贝叶斯定理、随机森林等算法去预测缺失值的最近替代值，也就是把缺失数据所对应的变量当做因变量，其他变量作为自变量，为每个需要进行缺失值赋值的字段分别建立预测模型。

我们先用线性回归模型来对带有缺失值的变量进行预测，达到缺失值插补的目的。

```
#回归模型插补
library(mice)
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
4 60-99 NA NA
10 40-59 NA NA
11 20-39 NA NA
12 40-59 NA NA
15 20-39 29.6 no NA
16 20-39 NA NA
20 60-99 25.5 yes NA
21 20-39 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
nhanes2 # 缺失值处理后的nhanes2数据集
age bmi hyp chl
1 20-39 NA 169
2 40-59 22.7 no 187
3 20-39 NA no 187
4 60-99 NA 225
5 20-39 20.4 no 113
6 60-99 NA 184
7 20-39 22.5 no 118
8 20-39 30.1 no 187
9 40-59 22.0 no 238
10 40-59 NA 203
11 20-39 NA 169
```

```
12 40-59 NA 203
13 60-99 21.7 no 206
14 40-59 28.7 yes 204
15 20-39 29.6 no 169
16 20-39 NA 169
17 60-99 27.2 yes 284
18 40-59 26.3 yes 199
19 20-39 35.3 no 218
20 60-99 25.5 yes 225
21 20-39 NA 169
22 20-39 33.2 no 229
23 20-39 27.5 no 131
24 60-99 24.9 no 225
25 40-59 27.4 no 186
```

由上面的例子可知，我们先需要对数据进行分区，把chl都有值的样本分在一起组成训练集，另外的数据组成测试集，利用训练集来生成线性回归模型，再用模型对测试集数据进行预测完成缺失值的插补。但是这种操作比较繁琐，接下来我们利用随机森林的缺失值插补方式进行处理，用法非常简单，只需要将数据集扔给函数，就可以完成所有变量的缺失值插补。

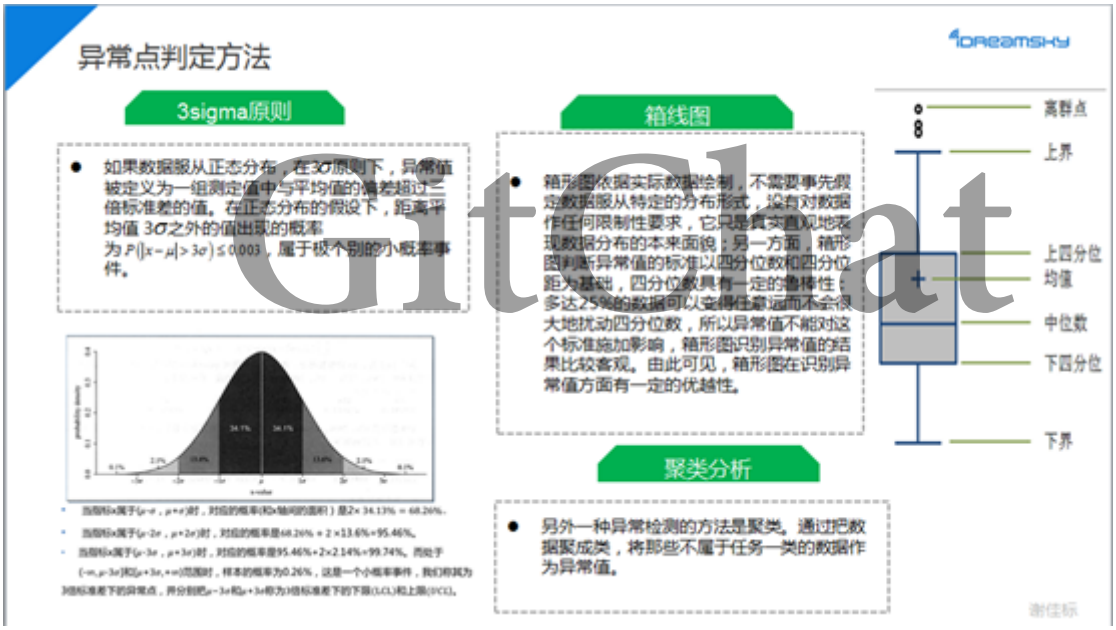

```
> #随机森林缺失值插补
> library(missForest)
> rm(list = ls())
> data("nhanes2")
> z=missForest(nhanes2)
  missForest iteration 1 in progress...done!
  missForest iteration 2 in progress...done!
  missForest iteration 3 in progress...done!
  missForest iteration 4 in progress...done!
> nhanes2.full = z$ximp
> nhanes2.full # 查看结果
     age      bmi hyp      chl
1  20-39 29.76103  no 184.2038
2  40-59 22.70000  no 187.0000
3  20-39 29.31503  no 187.0000
4  60-99 25.74215 yes 228.9633
5  20-39 20.40000  no 113.0000
6  60-99 25.80113  no 184.0000
7  20-39 22.50000  no 118.0000
8  20-39 30.10000  no 187.0000
9  40-59 22.00000  no 238.0000
10 40-59 27.11202 yes 219.4355
11 20-39 29.76103  no 184.2038
12 40-59 27.11202 yes 219.4355
13 60-99 21.70000  no 206.0000
14 40-59 28.70000 yes 204.0000
15 20-39 29.60000  no 184.2038
```

```
16 20-39 29.76103  no 184.2038
17 60-99 27.20000 yes 284.0000
18 40-59 26.30000 yes 199.0000
19 20-39 35.30000  no 218.0000
20 60-99 25.50000 yes 228.9633
21 20-39 29.76103  no 184.2038
22 20-39 33.20000  no 229.0000
23 20-39 27.50000  no 131.0000
24 60-99 24.90000  no 198.3600
25 40-59 27.40000  no 186.0000
```

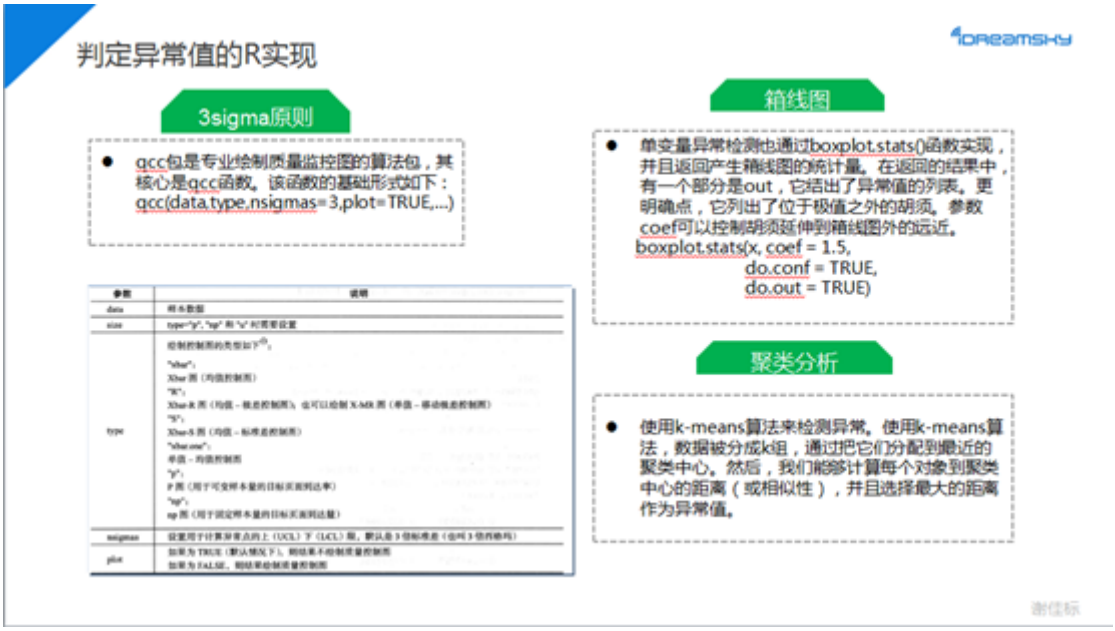
异常值甄别

学完缺失值的一些处理技巧，接下来让我们一起来学习下异常值的常用处理方式。

以下是常用的异常点判定方法：

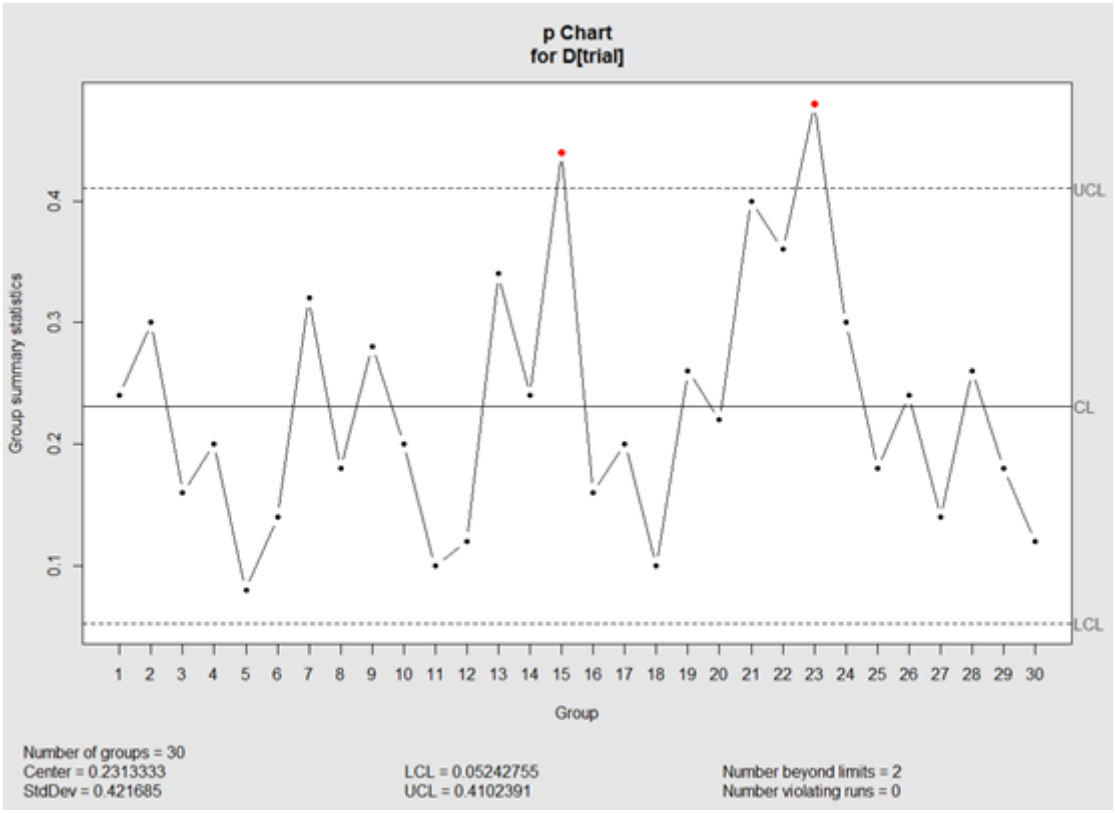


以下是判定异常值的R实现：



首先，我们利用qcc函数来识别异常点。

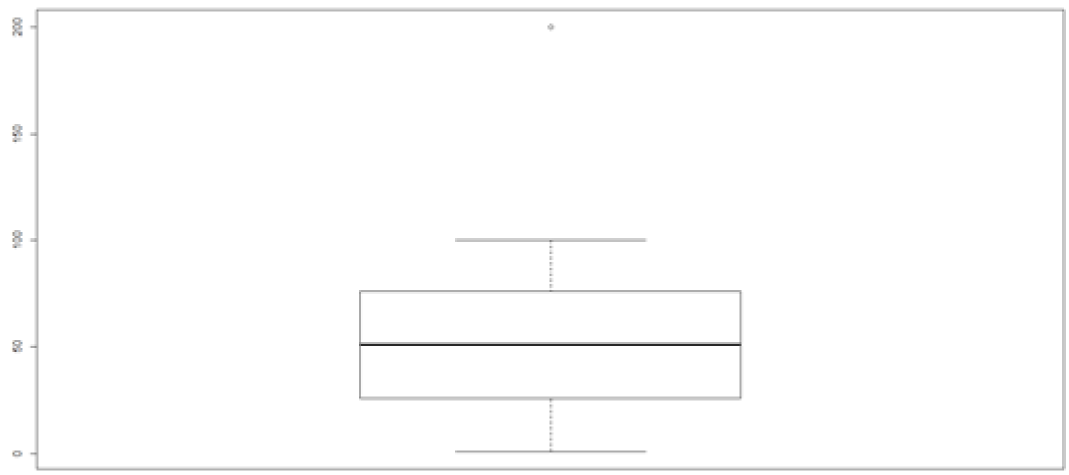
```
> # qcc函数
> # P质量控制图
> library(qcc)
Package 'qcc', version 2.6
Type 'citation("qcc")' for citing this R package in publications.
> data(orangejuice)
> attach(orangejuice)
> qcc(D[trial], sizes=size[trial], type="p")
List of 11
 $ call      : language qcc(data = D[trial], type = "p", sizes =
size[trial])
 $ type      : chr "p"
 $ data.name : chr "D[trial]"
 $ data      : int [1:30, 1] 12 15 8 10 4 7 16 9 14 10 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:30] 0.24 0.3 0.16 0.2 0.08 0.14 0.32
0.18 0.28 0.2 ...
  ..- attr(*, "names")= chr [1:30] "1" "2" "3" "4" ...
 $ sizes     : int [1:30] 50 50 50 50 50 50 50 50 50 50 ...
 $ center    : num 0.231
 $ std.dev   : num 0.422
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 0.0524 0.4102
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```



可见，第15次试验、第23次试验是异常点，在图中用红色表明，这两次的错误率高于正常水平，不符合产品合格标准。

接下来，我们利用boxplot函数绘制箱线图查看数据是否有异常点，并利用boxplot.stats函数将异常值找出来。


```
> # 使用boxplot.stats（）进行异常检测
> x <- c(1:100, 200)
> (b1 <- boxplot.stats(x))
$stats
[1]   1  26  51  76 100
$n
[1] 101
$conf
[1] 43.13921 58.86079
$out
[1] 200
> b1$out
[1] 200
> boxplot(x)
```

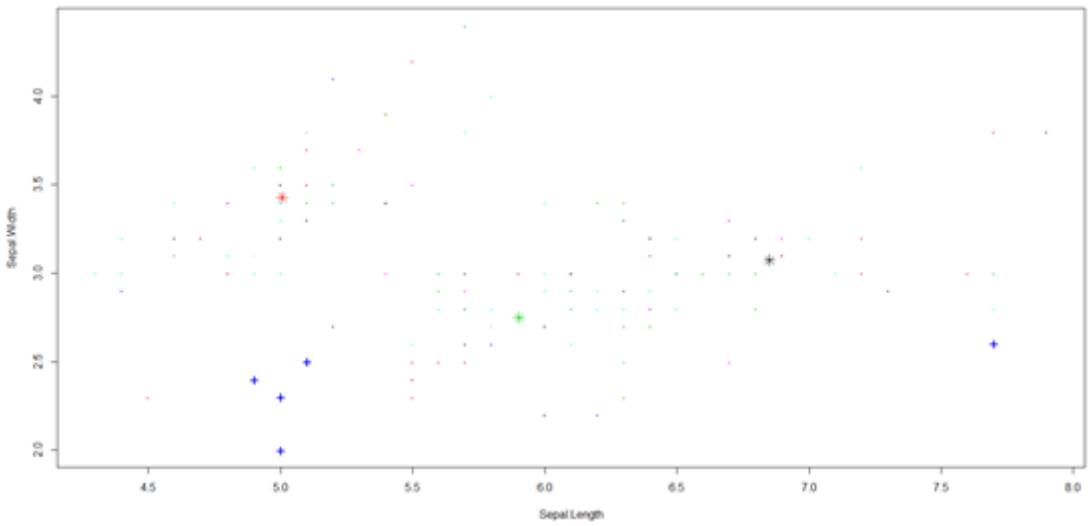


可见，boxplot.stats函数返回的是一个list列表，其中有一个out部分，打印了x数据集的异常值。我们可以通过b1$out将结果提出。

qcc函数或者boxplot.stats函数都是针对单变量甄别异常值的，但是很多时候，我们可能需要同时考虑多个变量的情况，才能判断样本是否为异常样本。此时可以利用K-Means函数实现。
此处我们以鸢尾花数据为例进行演示。先将鸢尾花数据集分为三类，将每个样本距离簇中心最大的五个点作为异常点。


```
> # 通过聚类进行异常检测
> iris1 <- iris[,1:4]
> kmeans.result <- kmeans(iris1,3)
> kmeans.result
K-means clustering with 3 clusters of sizes 38, 50, 62

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.006000    3.428000     1.462000    0.246000
3     5.901613    2.748387     4.393548    1.433871

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3
 [53] 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1
[105] 1 1 3 1 1 1 1 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3
1 1 1 1 3 1 1 1 3 1 1 1 3 1 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:
```

```
[1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> # 找出距离最大的5个样本
> centers <- kmeans.result$centers[kmeans.result$cluster,]
> distances <- sqrt(rowSums((iris1-centers)^2))
> outliers <- order(distances,decreasing = T)[1:5]
> print(outliers)
[1]  99  58  94  61 119
> print(iris1[outliers,])
    Sepal.Length Sepal.Width Petal.Length Petal.Width
99           5.1         2.5          3.0         1.1
58           4.9         2.4          3.3         1.0
94           5.0         2.3          3.3         1.0
61           5.0         2.0          3.5         1.0
119          7.7         2.6          6.9         2.3
> # 画图
> # 画出散点图
> plot(iris1[,c("Sepal.Length","Sepal.Width")],pch="o",
+      col=kmeans.result$centers,cex=0.3)
> # 画出类中心
>
points(kmeans.result$centers[,c("Sepal.Length","Sepal.Width")],col
=1:3,
+        pch=8,cex=1.5)
> # 画出离群点
>
points(iris1[outliers,c("Sepal.Length","Sepal.Width")],pch="+",col
=4,cex=1.5)
```

从上面的知识大家也知道，在做数据质量分析时，不管是缺失值、还是异常值我们在识别和处理都是有严谨的数据挖掘技术手段进行科学地处理，将数据达到建模分析前的质量。