天善智能
TIANSHAN SOFT
专注于商业智能BI和大数据的垂直社区平台

数据预处理实战

谢佳标（Daniel.xie）

数据预处理

数据概要

- 取值范围
- 分布特征
- 变异特征
- 频数统计

数据变换

- log转换
- x^(1/n)变换
- x^n变换
- 加权降维

数据清洗

- 去掉异常值
- 统一维度编码
- 离群点、极值处理
- 缺失值处理

数据抽样

- 简单抽样
- 数据分区
- 类失衡处理
- 哑变量处理

创建新变量与重新编码

```
## 创建新变量
# 方法一:
mydata <- iris[,1:2]
mydata$square <- mydata$Sepal.Length*mydata$Sepal.Width
# 方法二:
rm(list = ls())
mydata <- iris[,1:2]
attach(mydata)
mydata$square <- Sepal.Length*Sepal.Width
# 方法三:
rm(list = ls())
mydata <- iris[,1:2]
mydata <- transform(mydata,
                    square = Sepal.Length*Sepal.Length)
library(DT)
datatable(mydata,rownames = F)
### 变量重新编码
rm(list = ls())
mydata <-mtcars
mydata$am <- ifelse(mydata$am==0,"automatic","manual")
```

变量重命名

- reshape包中有一个rename()函数，可用于修改变量名。rename()函数的使用格式为：
  rename(*dataframe*, c(*oldname*="*newname*", *oldname*="*newname*",...))

```
w <- mtcars
library(reshape)
colnames(w)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

```
w <- rename(w,
            c(mpg = "Miles/(US) gallon",cyl = "Number of Cylinders",
              disp = "Displacement(cu.in.)",hp = "Gross horsepower"))
colnames(w)
```

```
##  [1] "Miles/(US) gallon"    "Number of Cylinders"  "Displacement(cu.in.)"
##  [4] "Gross horsepower"     "drat"                 "wt"
##  [7] "qsec"                 "vs"                   "am"
## [10] "gear"                 "carb"
```

```
# 也可以直接用names函数进行重命名
names(w)[5] <- "Rear axle ratio"
names(w)
```

```
##  [1] "Miles/(US) gallon"    "Number of Cylinders"  "Displacement(cu.in.)"
##  [4] "Gross horsepower"     "Rear axle ratio"      "wt"
##  [7] "qsec"                 "vs"                   "am"
## [10] "gear"                 "carb"
```

变量虚拟化

- caret包中有一个dummyVars)函数，可用变量虚拟化批处理。dummyVars()函数的使用格式为：
  dummyVars(formula, data, sep = ".", levelsOnly = FALSE, fullRank = FALSE, ...)

```
customers <- data.frame(
               id=c(10,20,30,40,50),
               gender=c('male','female','female','male','female'),
               mood=c('happy','sad','happy','sad','happy'),
               outcome=c(1,1,0,0,0))
customers
```

```
##   id gender  mood outcome
## 1 10   male happy       1
## 2 20 female   sad       1
## 3 30 female happy       0
## 4 40   male   sad       0
## 5 50 female happy       0
```

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
# 哑变量处理
dmy <- dummyVars(" ~ .", data = customers)
trsf <- data.frame(predict(dmy, newdata = customers))
print(trsf)
```

```
##   id gender.female gender.male mood.happy mood.sad outcome
## 1 10             0           1          1        0       1
## 2 20             1           0          0        1       1
## 3 30             1           0          1        0       0
## 4 40             0           1          0        1       0
## 5 50             1           0          1        0       0
```

转换函数-transform

- 一个数据框中常用的更改变量的函数是transform。形式上transform的定义如下：
  transform(`_data`, ...)
- 在调用这个函数时，首先要指定一个数据框（作为第一个参数），跟着是一系列的表达式，表达式中的变量是数据框中的变量。transform函数会完成每个表达式中的计算，然后返回最终的数据框。

```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> head(transform(airquality,Ozone=-Ozone))
  Ozone Solar.R Wind Temp Month Day
1   -41     190  7.4   67     5   1
2   -36     118  8.0   72     5   2
3   -12     149 12.6   74     5   3
4   -18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6   -28      NA 14.9   66     5   6
> head(transform(airquality,new=-Ozone,Temp=(Temp-23)/1.8))
  Ozone Solar.R Wind     Temp Month Day new
1    41     190  7.4 24.44444     5   1 -41
2    36     118  8.0 27.22222     5   2 -36
3    12     149 12.6 28.33333     5   3 -12
4    18     313 11.5 21.66667     5   4 -18
5    NA      NA 14.3 18.33333     5   5  NA
6    28      NA 14.9 23.88889     5   6 -28
```

排序

- R中涉及排序的基本函数有order、sort和rank三个。下面看下其基本用法：
  sort(x, decreasing = FALSE, ...)
  order(..., na.last = TRUE, decreasing = FALSE)
  rank(x, na.last = TRUE, ties.method = c("average", "first", "random", "max", "min"))
- x表示需要排序的数据，decreasing表示是否按降序排序数据，method表示所使用的排序算法，na.last用来说明如何处理NA值，如果为FALSE，则会删除这些值，如果为TRUE，就会把这些值放到最后。
- 下面通过例子，来更加深刻地理解这些问题：

```
> x<-c(19,84,64,2)
> order(x)
[1] 4 1 3 2
> rank(x)
[1] 2 4 3 1
> sort(x)
[1]  2 19 64 84
```

- 从结果中可以很容易看出三个函数之间的区别，order函数返回的是排序数据所在向量中的索引，rank函数返回该值处于第几位（在统计学上称为秩），sort函数则返回的是按次排好的数据。

选定特定行或者子集

- 很多时候需要根据一定的条件来提取特定的行，主要使用函数subset来实现这个功能。
  subset(x, subset, select, ...)
- x表示原数据，subset是逻辑表达式，表示需要满足的条件，select是一个表达式，表示对那些列来进行选择。
  subset(airquality, Temp > 80, select = c(Ozone, Temp))
  subset(airquality, Day == 1, select = -Temp)
  subset(airquality, select = Ozone:Wind)
  with(airquality, subset(Ozone, Temp > 80))

另一种操作数据框的方法

- 如果掌握了SQL查询语句，可能会觉得在R中对数据框进行操作很笨拙和难以理解。
- 很幸运的是，R也提供了查询语句的便利。可以使用sqldf包来完成这项工作。
- 这个包的名字就显现出这是SQL和df(data.frame)结合的产物。
- 下面通过列子，来试一下使用SQL语句对数据框进行操作。

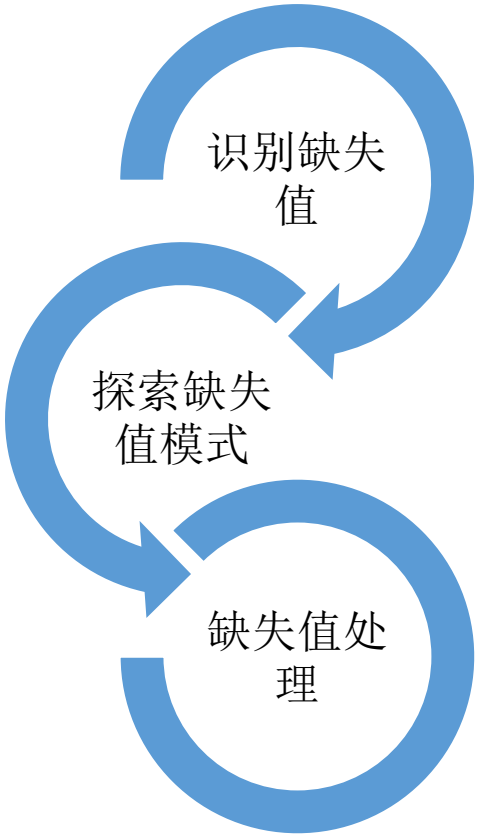
```
> library(sqldf)
> a1<-sqldf("select * from mtcars")
> head(a1)
   mpg cyl disp  hp drat    wt  qsec vs am gear carb
1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
6 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
> # 按照cyl求mpg的均值
> (a<-sqldf("select cyl,avg(mpg) as 'mean.mpg' from mtcars group by cyl"))
  cyl mean.mpg
1   4 26.66364
2   6 19.74286
3   8 15.10000
```

数据等比例抽样-creatDataPartition函数

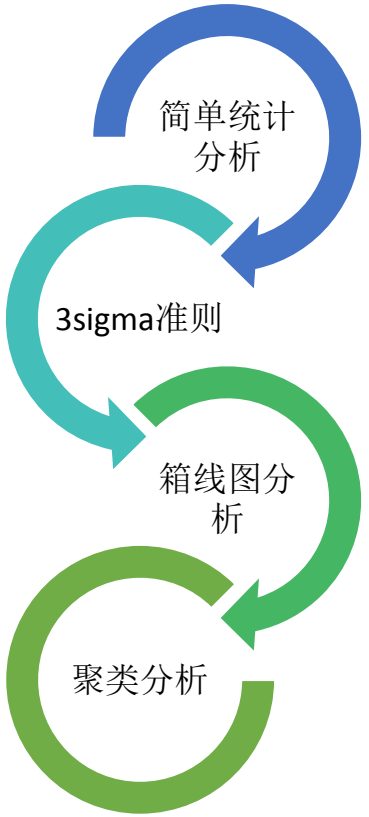
- 现在给大家介绍caret包中的createDataPartition函数，可以快速实现数据按照因子变量的类别进行快速等比例抽样。其函数基本表达形式为：
- createDataPartition(y, times = 1,p = 0.5,list = TRUE,groups = min(5, length(y)))
- 其中y是一个向量，times表示需要需要进行抽样的次数，p表示需要从数据中抽取的样本比例，list表示结果是否是list形式，默认为TRUE，groups:表示果输出变量为数值型数据，则默认按分位数分组进行取样。

数据清洗

缺失值判断及处理



异常值判断及处理



缺失值判断和处理

识别缺失值

- is.na函数
  判断元素是否缺失，如果元素缺失返回TRUE,否则返回FALSE
- complete.cases函数
  判断是否为完整个案，如果该样本没有元素缺失，则返回TRUE,如果该样本有元素缺失，则返回为FALSE

探索缺失值模式

- mice包中的md.pattern函数
  列表显示缺失值模式
- VIM包中的aggr函数
  图形探究缺失缺失值模式

删除缺失值

- complete.cases函数
  data[complete.cases(data),]
- na.omit函数
  na.omit(data)

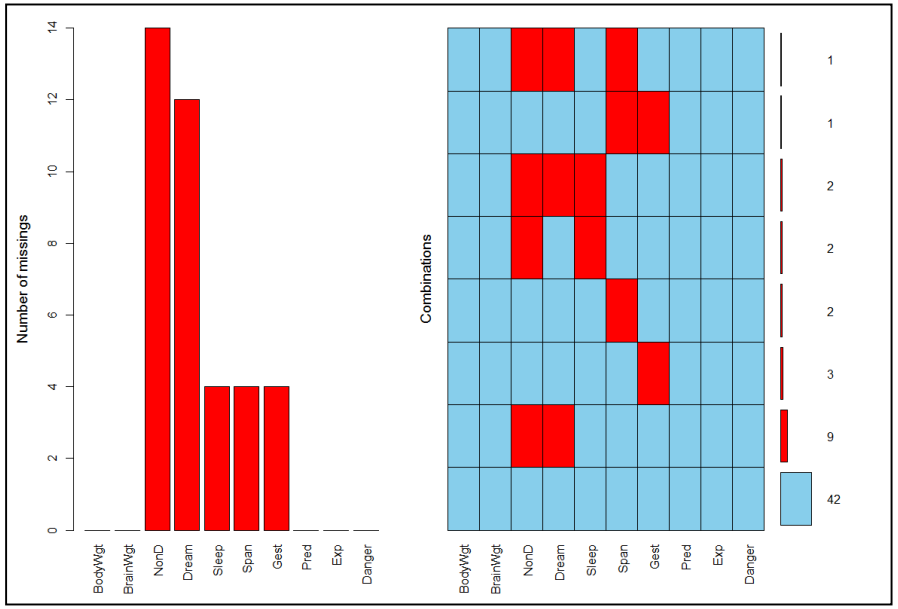
缺失值插补

- 均值/中位数填补
- 回归模型插补
- 随机森林插补
- 袋装插补

案例演示

识别缺失值模式

```
> md.pattern(sleep)
   BodyWgt BrainWgt Pred Exp Danger Sleep Span Gest Dream NonD
42       1        1    1   1      1     1    1    1     1    0
 2       1        1    1   1      1     1    0    1     1    1
 3       1        1    1   1      1     1    1    0     1    1
 9       1        1    1   1      1     1    1    1     0    2
 2       1        1    1   1      1     0    1    1     1    2
 1       1        1    1   1      1     1    0    0     1    2
 2       1        1    1   1      1     0    1    1     0    3
 1       1        1    1   1      1     1    0    1     0    3
         0        0    0   0      0     4    4    4    12   14 38
```



缺失值插补

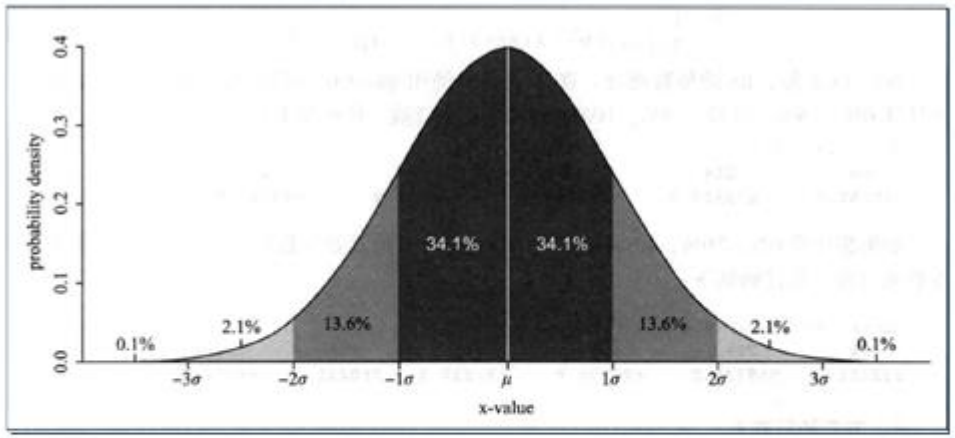
```
# 回归模型插补
library(mice)
sub=which(is.na(nhanes2[,4])==TRUE)
dataTR=nhanes2[-sub,]
dataTE=nhanes2[sub,]
dataTE
lm=lm(chl~age,data=dataTR)
nhanes2[sub,4]=round(predict(lm,dataTE))
head(nhanes2)
```

```
# 随机森林插补
airquality   #有缺失值NA的R自带的数据
complete.cases(airquality) #判断每行有没有缺失值
which(complete.cases(airquality)==F) #缺失值的行号
sum(complete.cases(airquality)) #完整观测值的个数
library(missForest) #用随机森林迭代弥补缺失值
z=missForest(airquality)
air.full=z$ximp  # 随机森林插补后的新数据集
```

异常点判定方法

3sigma原则

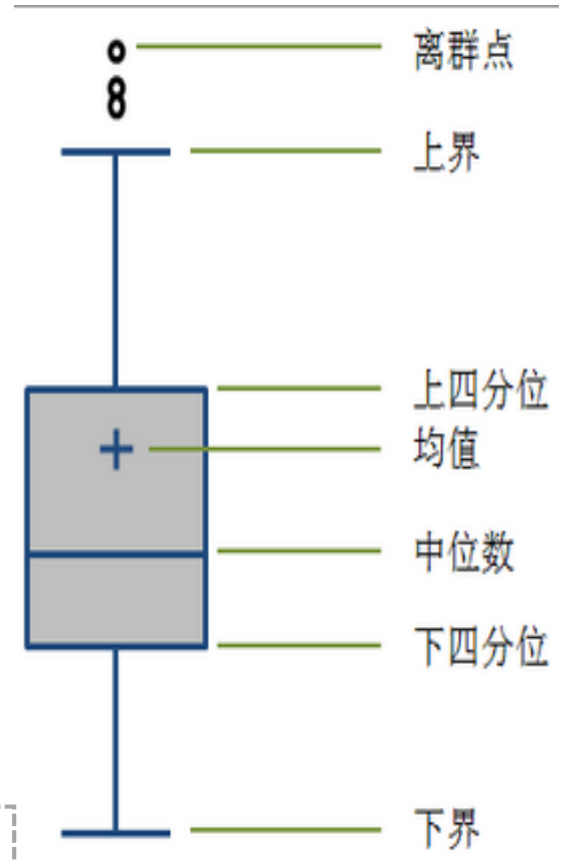
- 如果数据服从正态分布，在3σ 原则下，异常值被定义为一组测定值中与平均值的偏差超过三倍标准差的值。在正态分布的假设下，距离平均值 3σ 之外的值出现的概率为 $P(|x-\mu|>3\sigma)\leq 0.$属于极个别的小概率事件。



- 当指标x属于(μ-σ，μ+σ)时，对应的概率(和x轴间的面积）是2× 34.13% = 68.26%。
- 当指标x属于(μ-2σ，μ+2σ)时，对应的概率是68.26% + 2 ×13.6%=95.46%。
- 当指标x属于(μ-3σ，μ+3σ)时，对应的概率是95.46%+2×2.14%=99.74%。而处于(-∞, μ-3σ]和[μ+3σ,+∞)范围时，样本的概率为0.26%，这是一个小概率事件，我们称其为3倍标准差下的异常点，并分别把$\mu-3\sigma$和$\mu+3\sigma$称为3倍标准差下的下限(LCL)和上限(UCL)。

箱线图

- 箱形图依据实际数据绘制，不需要事先假定数据服从特定的分布形式，没有对数据作任何限制性要求，它只是真实直观地表现数据分布的本来面貌；另一方面，箱形图判断异常值的标准以四分位数和四分位距为基础，四分位数具有一定的鲁棒性：多达25%的数据可以变得任意远而不会很大地扰动四分位数，所以异常值不能对这个标准施加影响，箱形图识别异常值的结果比较客观。由此可见，箱形图在识别异常值方面有一定的优越性。



聚类分析

- 另外一种异常检测的方法是聚类。通过把数据聚成类，将那些不属于任务一类的数据作为异常值。

判定异常值的R实现

3sigma原则

- qcc包是专业绘制质量监控图的算法包，其核心是qcc函数。该函数的基础形式如下：
  qcc(data,type,nsigmas=3,plot=TRUE,...)

参数	说明
data	样本数据
size	type="p", "np" 和 "u" 时需要设置
type	绘制控制图的类型如下⊖： "xbar"： Xbar 图（均值控制图） "R"： Xbar-R 图（均值－极差控制图）；也可以绘制 X-MR 图（单值－移动极差控制图） "S"： Xbar-S 图（均值－标准差控制图） "xbar.one"： 单值－均值控制图 "p"： P 图（用于可变样本量的目标页面到达率） "np"： np 图（用于固定样本量的目标页面到达量）
nsigmas	设置用于计算异常点的上（UCL）下（LCL）限，默认是 3 倍标准差（也叫 3 倍西格玛）
plot	如果为 TRUE（默认情况下），则结果不绘制质量控制图 如果为 FALSE，则结果绘制质量控制图

箱线图

- 单变量异常检测也通过boxplot.stats()函数实现，并且返回产生箱线图的统计量。在返回的结果中，有一个部分是out，它结出了异常值的列表。更明确点，它列出了位于极值之外的胡须。参数coef可以控制胡须延伸到箱线图外的远近。
  boxplot.stats(x, coef = 1.5,
  do.conf = TRUE,
  do.out = TRUE)

聚类分析

- 使用k-means算法来检测异常。使用k-means算法，数据被分成k组，通过把它们分配到最近的聚类中心。然后，我们能够计算每个对象到聚类中心的距离（或相似性），并且选择最大的距离作为异常值。

案例实战

- 案例一：对付费用户数据进行数据抽样
- 案例二：对问卷调研数据的缺失模式进行探索及插补