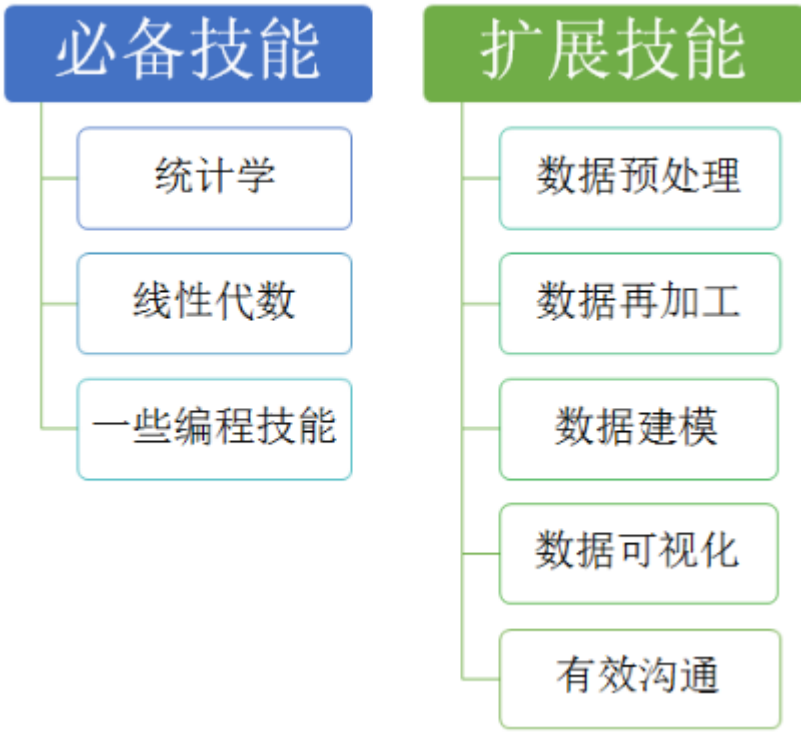
R语言数据挖掘之数据管理

数据的爆炸式增长、广泛可用和巨大数量使得我们的时代成为真正的数据时代。急需功能强大和通用的工具，以便从这些海量数据中发现有价值的信息，把这些数据转化成有组织的知识。这种需求导致了数据挖掘的诞生。这个领域是年青的、动态变化的、生机勃勃的。数据挖掘已经并且将继续在我们从数据时代大步跨入信息时代的历程中做出贡献。

数据挖掘能把大型数据集转换成知识。像Google这样的搜索引擎每天接受数亿次查询。每个查询都被看做一个事务，用户通过事务描述他们的信息需求。随着时间的推移，搜索引擎可以从这些大量的搜索查询中学到什么样的新颖的、有用的知识？有趣的是，从众多用户查询中发现的某些模式能够揭示无价的知识，这些知识无法通过仅读取个体数据项得到。例如，Google的Flu Trends（流感趋势）使用特殊的搜索项作为流感活动的指示器。它发现了搜索流感相关信息的人数与实际具有流感症状的人数之间的紧密联系。当与流感相关的所有搜索都聚集在一起时，一个模式就出现了。使用聚集的搜索数据，Google的Flu Trends可以比传统的系统早两周对流感活动作出评估。这个例子表明，数据挖掘如何把大型数据集转化成知识，帮助我们应对当代的全球性挑战。

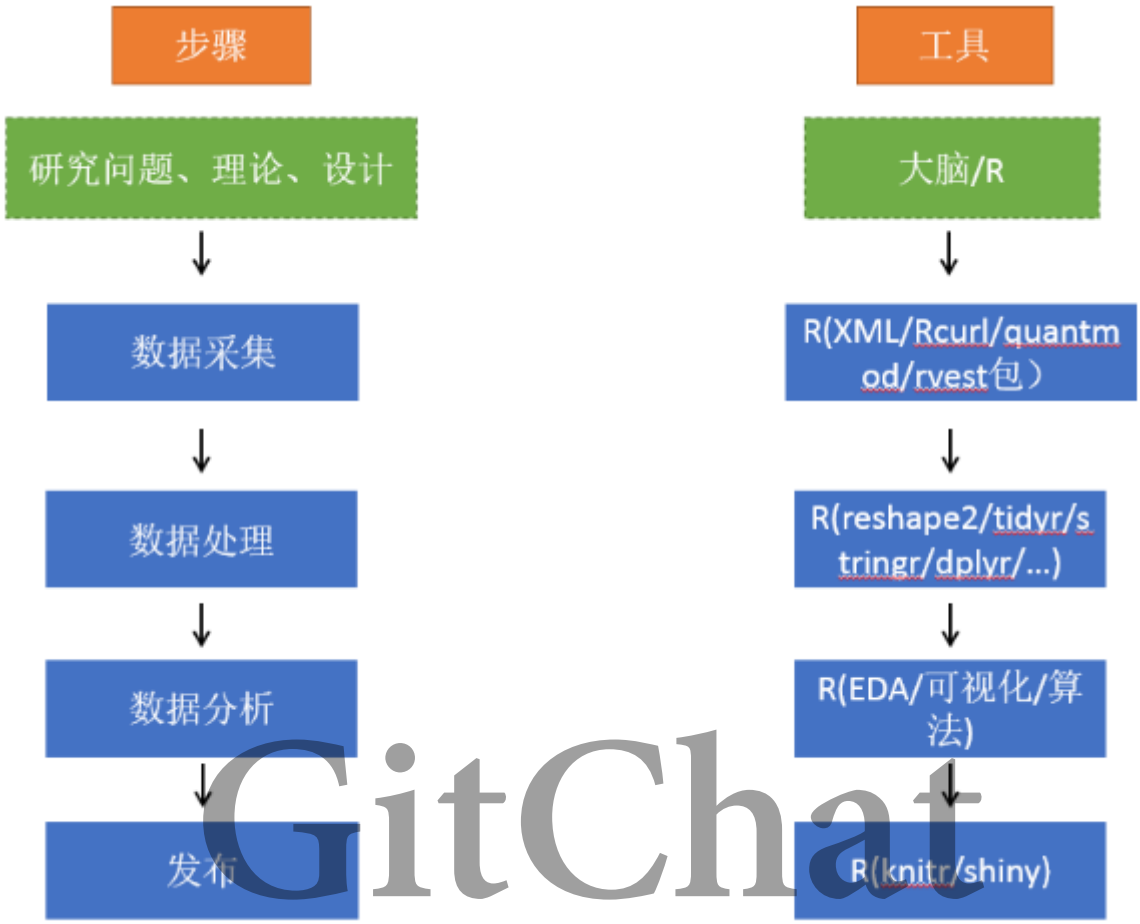
作为一位合格的数据从业者，我们需要掌握数学、统计学、编程能力是基本要求，此外还需要有以下的数据处理、建模、可视化等扩展能力。



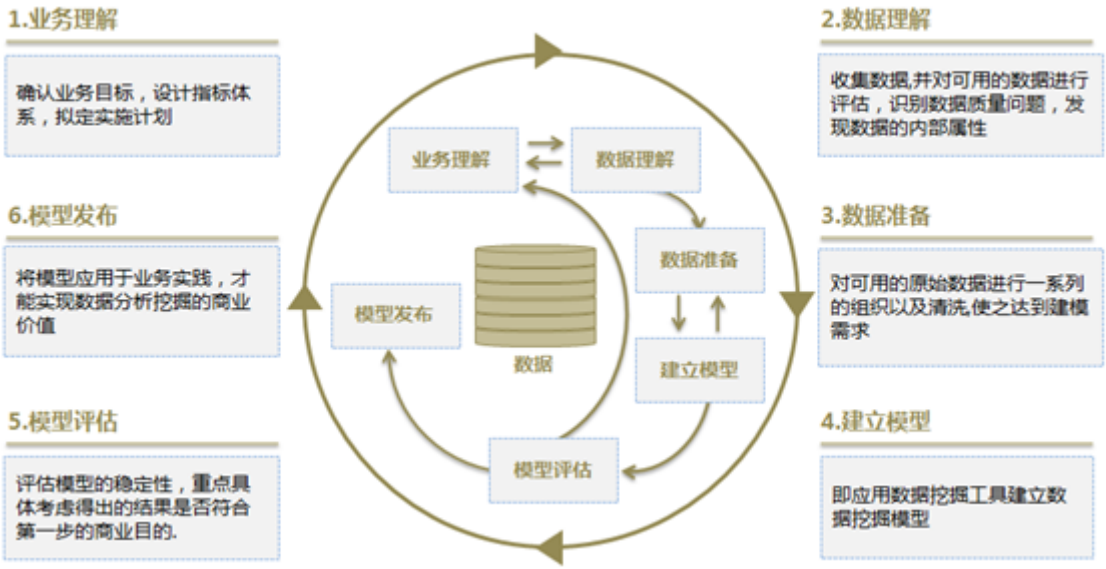
还需要熟练掌握一门工具，能将你的想法迅速实现。这几年流行的R和python都是不错的选择，如果你是期望从事与数据分析、挖掘相关的工作，建议可将R作为首选工具，

其丰富的扩展包可以高效进行数据处理及建模，让你将更多的时间专注在业务和数据本身。

使用R语言进行数据挖掘的各个环节如下：



在正式开始数据挖掘之旅之前，让我们来了解一下数据挖掘流程。按照CRISP-DM方法论，一个游戏挖掘的完整流程包括6个阶段，分别是业务理解、数据理解、数据准备、建立模型、模型评估和模型发布。这6个阶段的顺序并不是固定不变的，在不同的业务场景中，可以有不同的流转方向。



在数据处理方面，R语言在数据概要、数据变换、数据清洗和数据抽样等有非常丰富的选择，让我们能对数据进行高效处理，达到数据分析和建模的目的。

数据概要相对来说比较简单，接下来，就让我们一起来学习下数据变换数据抽样等方面的知识，为将来的数据分析、挖掘建模提供合适的数据集。

数据挖掘最重要的一环就是如何管理你的数据，因为原始数据一般都不能直接用来进行分析，需要对原始数据进行增加衍生变量、数据分箱、数据标准化处理；对因子型变量进行哑变量处理；数据抽样和类失衡数据处理。本专题会详细介绍以上内容的数据挖掘技术及R语言实现。

一、数据转换

对于数据挖掘分析建模来说，数据转换(Transformation)是最常用、最重要，也是最有效的一种数据处理技术。经过适当的数据转换后，模型的效果常常可以有明显的提升，也正因为这个原因，数据转换成了很多数据分析师在建模过程中最喜欢使用的一种数据处理手段。另一方面，在绝大数数据挖掘实践中，由于原始数据，在此主要是指区间型变量(Interval)的分布不光滑(或有噪声)，不对称分布(Skewed Distributions)，也使得数据转化成为一种必须的技术手段。

按照采用的转换逻辑和转换目的的不同，数据转换主要分为以下四大类：

产生衍生变量。

这类转换的目的很直观，即通过对原始数据进行简单、适当地数据公式推导，产生更有商业意义的新变量。例如，我们收集了最近一周的付费人数和付费金额，此时想统计每日的日均付费金额（arpu=revenue/user），此时就可以通过前两个变量快速实现。

```
> # 创建数据集
> w <- data.frame(day = 1:7,
+                 revenue = sample(5000:6000,7),
+                 user = sample(1000:1500,7))
> w
  day revenue user
1   1    5391 1223
2   2    5312 1418
3   3    5057 1343
4   4    5354 1397
5   5    5904 1492
6   6    5064 1113
7   7    5402 1180
> # 增加衍生变量人均付费金额(arpu)
> w$arpu <- w$revenue/w$user
> w
  day revenue user     arpu
1   1    5391 1223 4.408013
2   2    5312 1418 3.746121
3   3    5057 1343 3.765450
4   4    5354 1397 3.832498
5   5    5904 1492 3.957105
6   6    5064 1113 4.549865
7   7    5402 1180 4.577966
```


从中不难发现，得到这些衍生变量所应用到的数据公式很简单，但是其商业意义是明确的，而且跟具体的分析背景和分析思路密切相关。

衍生变量的产生主要依赖于数据分析师的业务熟悉程度和对项目思路的掌握程度，如果没有明确的项目分析思路和对数据的透彻理解，是无法找到有针对性的衍生变量的。

改善变量分布特征的转换，这里主要是指不对称分布所进行的转换。

在数据挖掘实战中，有些数据是不对称的，严重不对称出现在自变量中常常会干扰模型的拟合，最终会影响模型的效果和效率。如果通过各种数学变换，使得变量的分布呈现(或者近似)正态分布，并形成倒钟形曲线，那么模型的拟合常常会有明显的提升，转换后自变量的预测性能也可能得到改善，最终将会提高模型的效果和效率。

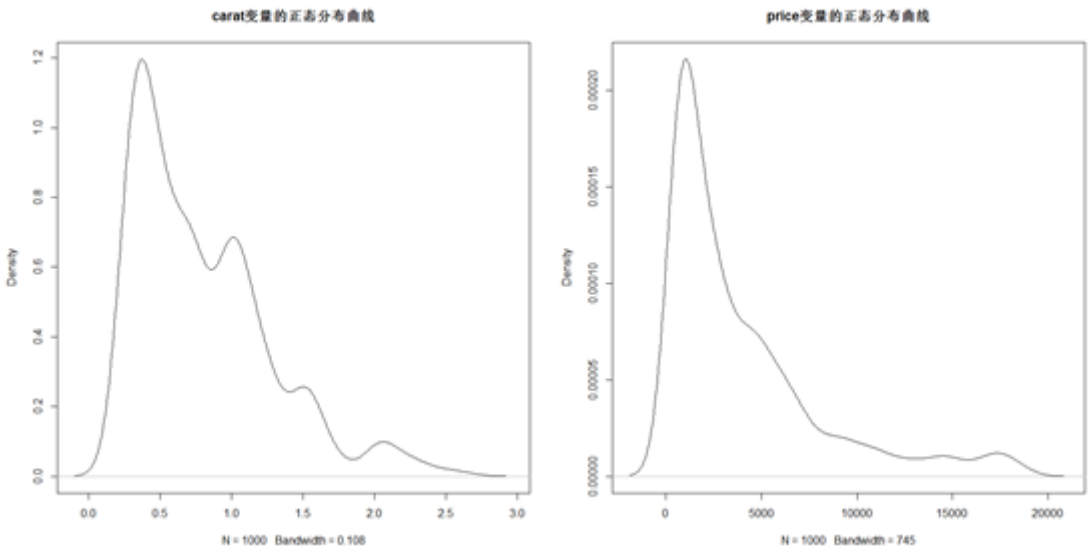
常见的改善分布的转换措施如下：

- 取对数(Log)
- 开平方根(Square Root)
- 取倒数(Inverse)
- 开平方(Square)

- 取指数(Exponential)

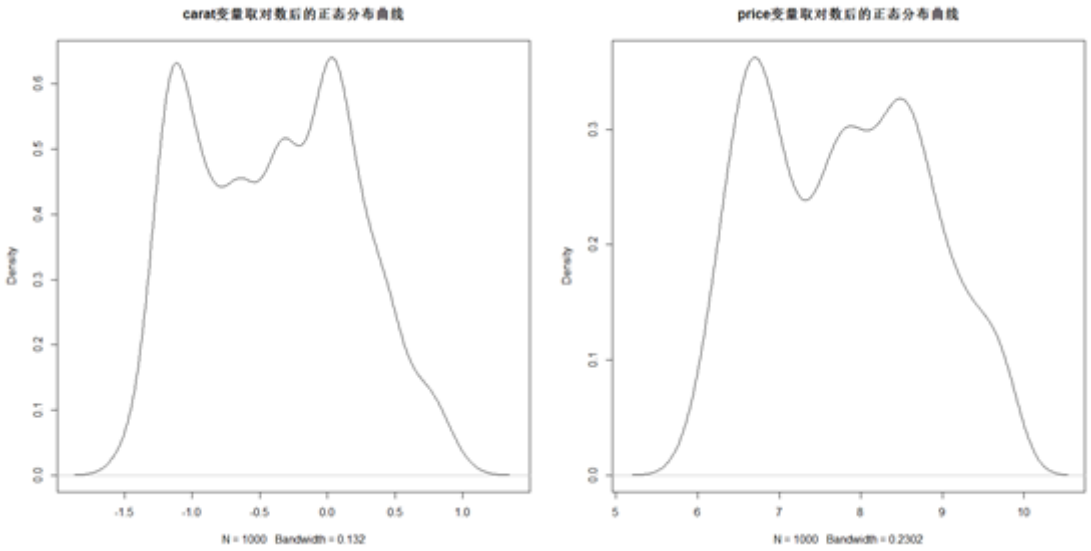
这边以取对数为例进行说明。在R的扩展包ggplot2中自带了一份钻石数据集(diamonds)，我们从中抽取1000个样本最为研究对象，研究数据中变量carat(克拉数)、price(价格)的数据分布情况，并研究两者之间的关系，最后利用克拉数预测钻石的价格。

```
> > library(ggplot2)
> set.seed(1234)
> dsmall <- diamonds[sample(1:nrow(diamonds),1000),] # 数据抽样
> head(dsmall) # 查看数据前六行
      carat       cut color clarity depth table price    x    y
z
6134   0.91     Ideal     G     SI2  61.6    56  3985 6.24 6.22
3.84
33567  0.43   Premium     D     SI1  60.1    58   830 4.89 4.93
2.95
32864  0.32     Ideal     D     VS2  61.5    55   808 4.43 4.45
2.73
33624  0.33     Ideal     G     SI2  61.7    55   463 4.46 4.48
2.76
46435  0.70      Good     H     SI1  64.2    58  1771 5.59 5.62
3.60
34536  0.33     Ideal     G    VVS1  61.8    55   868 4.42 4.45
2.74
> par(mfrow = c(1,2))
> plot(density(dsmall$carat),main = "carat变量的正态分布曲线") # 绘
制carat变量的正态分布曲线
> plot(density(dsmall$price),main = "price变量的正态分布曲线") # 绘
制price变量的正态分布曲线
> par(mfrow = c(1,1))
```

从正态分布图课件，变量carat和price均是严重不对称分布。此时我们利用R语言中的log函数对两者进行对数转换，再次绘制正态密度图。

```
> par(mfrow = c(1,2))
> plot(density(log(dsmall$carat)),main = "carat变量取对数后的正态分布曲线")
> plot(density(log(dsmall$price)),main = "price变量取对数后的正态分布曲线")
> par(mfrow = c(1,1))
```



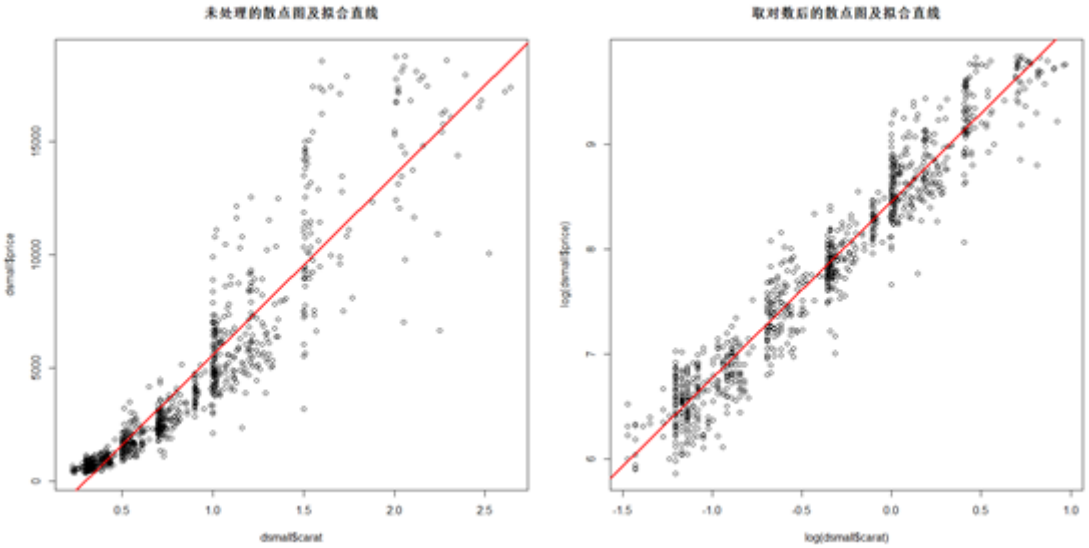
可见，经过对数处理后，两者的正态分布密度曲线就对称很多。最后，让我们一起来验证对原始数据建立线性回归模型与经过对数变量后再建模的区别(关于模型这块，将在数据挖掘模型篇会详细讲解)。


```
> # 建立线性回归模型
> fit1 <- lm(dsmall$price~dsmall$carat,data = dsmall) # 对原始变量进行建模
> summary(fit1) # 查看模型详细结果

Call:
lm(formula = dsmall$price ~ dsmall$carat, data = dsmall)

Residuals:
    Min      1Q  Median      3Q     Max
-8854.8  -821.9   -42.2   576.0  8234.2

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2391.74      97.44  -24.55   <2e-16 ***

dsmall$carat  7955.35     104.45   76.16   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1582 on 998 degrees of freedom
Multiple R-squared:  0.8532,    Adjusted R-squared:  0.8531
F-statistic:  5801 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
> fit2 <-lm(log(dsmall$price)~log(dsmall$carat),data=dsmall) # 对两者进行曲对数后再建模
> summary(fit2) # 查看模型结果

Call:
lm(formula = log(dsmall$price) ~ log(dsmall$carat), data = dsmall)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.07065 -0.16438 -0.01159  0.16476  0.83140 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)        8.451358   0.009937   850.5   <2e-16 ***
log(dsmall$carat) 1.686009   0.014135   119.3   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2608 on 998 degrees of freedom
Multiple R-squared:  0.9345,    Adjusted R-squared:  0.9344 
F-statistic: 1.423e+04 on 1 and 998 DF,  p-value: < 2.2e-16
```


通过对比Multiple R-squared发现，模型1的R平方是0.8532，模型2的R平方是0.9345，R平方的值是越接近1说明模型拟合的越好，所以经过对数处理后建立的模型2优于模型1。我们也可以通过在散点图绘制拟合曲线的可视化方式进行查看。

```
> # 在散点图中 绘制拟合曲线
> par(mfrow=c(1,2))
> plot(dsmall$carat,dsmall$price,
+      main = "未处理的散点图及拟合直线")
> abline(fit1,col="red",lwd=2)
> plot(log(dsmall$carat),log(dsmall$price),
+      main = "取对数后的散点图及拟合直线")
> abline(fit2,col="red",lwd=2)
> par(mfrow=c(1,1))
```



可见，取对数后绘制的散点更集中在红色的线性回归线上。

区间型变量的分箱转换。

分箱转换(Binning)就是把区间型变量 (Interval)转换成次序型变量(Ordinal)，其转换的目的如下：

降低变量(主要是指自变量)的复杂性，简化数据。比如，有一组用户的年龄，原始数据是区间型的，从10~60岁，每1岁都是1个年龄段；如果通过分箱转换，每10岁构成1个年龄组，就可以有效简化数据。R语言中有cut函数可以轻易实现数据分箱操作。


```
age <- sample(10:60,15)
> age <- sample(10:60,15) # 创建年龄变量
> age
 [1] 26 41 59 33 50 16 28 18 52 30 25 44 37 23 47
> age_cut <- cut(age,breaks = seq(10,60,10))
> age_cut
 [1] (20,30] (40,50] (50,60] (30,40] (40,50] (10,20] (20,30]
(10,20] (50,60]
[10] (20,30] (20,30] (40,50] (30,40] (20,30] (40,50]
Levels: (10,20] (20,30] (30,40] (40,50] (50,60]
> table(age_cut) # 查看不同年龄段的人数
age_cut
(10,20] (20,30] (30,40] (40,50] (50,60]
      2       5       2       4       2
```

可见，利用cut函数分箱得到的区间段是左开右闭的，我们通过table函数查看不同区间段的人数，发现有5人在20到30岁之间。

提升自变量的预测能力。如果分箱恰当的，这样就可以显著提升模型的预测效率和效果；尤其是当自变量与因变量有比较明显的非线性关系时，分箱操作更是不错的手段，课用于探索和发现这些相关性；另外，当自变量的偏度很大时，分箱操作也是值得积极尝试的方法。

针对区间型变量进行的标准化操作。

数据的标准化(Normalization)转换也是数据挖掘中常见的数据转换措施之一，数据标准转换的目的是将数据按照比例进行缩放，使之落入一个小的区间范围之内，使得不同的变量经过标准化处理后可以有平等分析和比较的基础。

最简单的数据标准化转换是Min-Max标准化，也叫离差标准化，是对原始数据进行线性变换，使得结果在[0,1]区间，其转换公式如下：

$$x^*=(x-min)/(max-min)$$

其中，max为样本数据的最大值，min为样本数据的最小值。

在R中，我们可以利用max函数和min函数非常轻易地构建一个Normalization函数，实现Min-Max标准化过程。

```
> dat <- sample(1:20,10) # 从1到20中有放回随机抽取10个
> normalization <- function(x) {
+   (x-min(x))/(max(x)-min(x))
+ } # 构建Min-Max标准化的自定义函数
> dat # 原始数据
 [1] 19  5  8 13 15 18 16  1  3  9
> normalization(dat) # 经过Min-Max标准化的 数据
 [1] 1.0000000 0.2222222 0.3888889 0.6666667 0.7777778 0.9444444
0.8333333
 [8] 0.0000000 0.1111111 0.4444444
```


另一种常用的标准化处理是零-均值标准化，即把数据处理称符合标准正态分布。也就是均值为0，标准差为1，转换公式如下：

$$x^*=(x-\mu)/\sigma$$

其中，μ为所有样本数据的均值，σ为所有样本数据的标准差。

在R中，用scale( )函数得到Z-Score标准化。其表达形式为：scale(x, center = TRUE, scale = TRUE)。
总体来说，数据变换的方式多种多样，操作起来简单、灵活、方便，在 实践应用中的价值也是比较明显的。

二、数据抽样

“抽样”对于数据分析和挖掘来说是一种常见的前期数据处理技术和阶段，之所以要采取抽样,主要原因在于如果数据全集的规模太大，针对数据全集进行分析运算不但会消耗更多的运算资源，还会显著增加运算分析的时间，甚至太大的数据量有时候会导致分析挖掘软件运行时的崩溃。而采用了抽样措施，就可以显著降低这些负面的影响；另一个常

见的需要通过抽样来解决的场景就是：我们需要将原始数据进行分区，其中一部分作为训练集，用来训练模型，剩下的部分作为测试集，用来对训练好的模型进行效果评估。

R中的sample( )函数可以实现数据的随机抽样。基本表达形式为：

```
sample(x, size, replace = FALSE, prob = NULL)
```

其中x是数值型向量，size是抽样个数，replace表示是否有放回抽样，默认FALSE是无放回抽样，TURE是有放回抽样。

```
> # sample小例子
> set.seed(1234)
> # 创建对象x，有1~10组成
> x <- seq(1,10);x
 [1]  1  2  3  4  5  6  7  8  9 10
> # 利用sample函数对x进行无放回抽样
> a <- sample(x,8,replace=FALSE);a
[1] 2 6 5 8 9 4 1 7
> # 利用sample函数对x进行有放回抽样
> b <- sample(x,8,replace=TRUE);b
[1]  7  6  7  6  3 10  3  9
> # 当size大于x的长度
> (c <- sample(x,15,replace = F))
Error in sample.int(length(x), size, replace, prob) :
  cannot take a sample larger than the population when 'replace =
FALSE'
> (c <- sample(x,15,replace = T))
 [1]  3  3  2  3  4  4  2  1  3  9  6 10  9  1  5
```


可见，b中抽取的元素有重复值。如果我们要抽取的长度大于x的长度，需要将replace参数设置为T（有放回抽样）。

有时候，我们想根据某一个变量对数据进行等比例抽样（即抽样后的数据子集中的该变量各因子水平占比与原来相同），虽然我们利用sample函数也可以构建，但是这里给大家介绍caret扩展包中的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。

以鸢尾花数据集为例，我们想按照物种分类变量进行等比例随机抽取其中10%的样本进行研究。

```
> # 载入caret包，如果本地未安装就在线安装caret包
> if(!require(caret)) install.packages("caret")
载入需要的程辑包：caret
载入需要的程辑包：lattice
载入需要的程辑包：ggplot2
Warning message:
程辑包‘ggplot2’是用R版本3.3.3 来建造的
> # 提取下标集
> splitindex <-
createDataPartition(iris$Species,times=1,p=0.1,list=FALSE)
> iris_subset <- iris[splitindex,]
> prop.table(table(iris$Species)) # 查看原来数据集Species变量各因子占
比

    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333
> prop.table(table(iris_subset$Species))# 查看子集中Species变量各因
子占比
 setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333
```

可见，抽样后的子集中Species中各类别的占比与原来数据集的相同。

三、类失衡数据处理

另外一个常见的需要通过抽样来解决的场景就是：在很多小概率事件、稀有事件的预测建模过程中，比如信用卡欺诈事件，在整个信用卡用户中，属于恶意欺诈的用户只占0.2%甚至更少，如果按照原始的数据全集、原始的稀有占比来进行分析挖掘，0.2%的稀有事件是很难通过分析挖掘得到有意义的预测和结论的，所有对此类稀有事件的分析建模，通常会采取抽样的措施，即认为增加样本中“稀有事件”的浓度和在样本中的占比。对抽样后得到的分析样本进行分析挖掘，可以比较容易地发现稀有时间与分析变量之间的价值，有意义的一些关联性和逻辑性。

克服类失衡问题常用的技术有以下两种：

- 偏置学习过程的方法，它应用特定的对少数类更敏感的评价指标。
- 用抽样方法来操作训练数据，从而改变类的分布。

有多种抽样方法用于改变数据集中类的失衡，常用的有以下两种：

- 欠采样法，它从多数类中选择一小部分案例，并把它们和少数类个案一起构成一个有更加平衡的类分布的数据集。
- 过采样法，它采用另外的工作模式，使用某些进程来复制少数类个案。

在R中，DMwR包中的SMOTE( )函数可以实现SMOTE方法。主要参数有如下三个：perc.over:过采样时，生成少数类的样本个数;k:过采样中使用K近邻算法生成少数类样本

时的K值，默认是5；perc.under:欠采样时，对应每个生成的少数类样本，选择原始数据多数类样本的个数。例如，perc.over=500表示对原始数据集中的每个少数样本，都将生成5个新的少数样本；perc.under=80表示从原始数据集中选择的多数类的样本是新生的数据集中少数样本的80%。

```
> # 加载DMwR包
> if(!require(DMwR)) install.packages("DMwR")
载入需要的程辑包：DMwR
载入需要的程辑包：lattice
载入需要的程辑包：grid
> dat <- iris[, c(1, 2, 5)]
> dat$Species <- factor(ifelse(dat$Species ==
"setosa","rare","common"))
> table(dat$Species)
common   rare
   100     50
> # 进行类失衡处理
> # perc.over=600:表示少数样本数=50+50*600%=350
> # perc.under=100:表示多数样本数(新增少数样本数*100%=300*100%=300)
> newData <- SMOTE(Species ~ ., dat, perc.over =
600,perc.under=100)
> table(newData$Species)
common   rare
 300    350
> # perc.over=100:表示少数样本数=50+50*100%=100
> # perc.under=200:表示多数样本数(新增少数样本数*200%=50*200%=100)
> newData <- SMOTE(Species ~ ., dat, perc.over =
100,perc.under=200)
> table(newData$Species)
common   rare
   100    100
```


四、数据哑变量处理

虚拟变量 (Dummy Variables) 又称虚设变量、名义变量或哑变量，用以反映质的属性的一个人工变量，是量化了的自变量，通常取值为0或1。引入哑变量可使线形回归模型变得更复杂，但对问题描述更简明，一个方程能达到两个方程的作用，而且接近现实。

举一个例子，假如变量“性别”的取值为：男性、女性。我们可以增加2个哑变量来代替“性别”这个变量，分别为性别.男性(1=男性/0=女性)、性别.女性(1=女性/0=男性)。

caret扩展包中的dummyVars( )函数是专门用来处理变量哑变量处理的函数，其表达形式为：

```
dummyVars(formula, data, sep = ".", levelsOnly = FALSE,  fullRank
= FALSE, ...)
```

其中，formula表示模型公式，data是需要处理的数据集，sep表示列名和因子水平间的连接符号，levelsOnly默认是FALSE，当为TRUE时表示仅用因子水平表示新列名，fullRank默认是FALSE，当为TRUE时表示进行虚拟变量处理后不需要出现代表相同意思的两列。

假如有一份customers数据集，包括id、gender、mood和outcome变量，其中gender和mood都是因子型变量，我们需要将它们进行哑变量处理。

```
> 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)
载入需要的程辑包：ggplot2
Warning message:
程辑包‘ggplot2’是用R版本3.3.3 来建造的
> # 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
```


经过哑变量处理后，原来的变量gender变成两列gender.female和gender.male，变量mood变成mood.happy和mood.sad两列，新生成的列里面的内容均是0/1，已经帮我们做了归一化处理。

在生产环境中，很多时候我们只需要通过简单的数学公式计算得到衍生变量就能很好地解决业务问题，并在建模时候需要对数据进行分区，这些都是在前期数据管理时进行数据抽样工作完成的，所以我们学好数据管理的知识，可能为将来做数据分析提供很大的便利，少走很多弯路。