Lecture 7: Bootstrap(自助)方法和 Jackknife(刀切)方法

张伟平

Thursday 19$^{\text{th}}$ July, 2018

Contents

Chapter 1
Bootstrap and Jackknife

1.1 The Bootstrap

Efron 在1979,1981和1982年的工作中引入和进一步发展了Bootstarp方法, 此后发表了大量的关于 此方法的研究.

Bootstrap方法是一类非参数Monte Carlo方法, 其通过再抽样对总体分布进行估计. 再抽样方法将 观测到的样本视为一个有限总体, 从中进行随机(再)抽样来估计总体的特征以及对抽样总体作出统计推断. 当目标总体分布没有指定时, Bootstrap方法经常被使用, 此时, 样本是唯一已有的信息.

Bootstrap 一词可以指非参数Bootstrap, 也可以指参数Bootstrap(上一讲中). 参数Bootstrap是指总体分布 完全已知, 利用Monte Carlo方法从此总体中抽样进行统计推断; 而非参数Bootstrap是指总体分布完全未知, 利用再抽样

方法从样本中(再)抽样进行统计推断.

可以视样本所表示的有限总体的分布为一个"伪"总体, 其具有和真实总体类似的特征. 通过从此"伪"总体中 重复(再)抽样, 可以据此估计统计量的抽样分布. 统计量的一些性质, 如偏差, 标准差等也可以通过再抽样来估计.

一个抽样分布的Bootstrap估计类似于密度估计的想法. 我们通过一个样本的直方图来估计密度函数的形状. 直方图 不是密度, 但是在非参数问题中, 可以被视为是密度的一个合理估计. 我们有很多方法从已知的密度中产生随机样本, Bootstrap则从经验分布中产生随机样本. 假设$x = (x_1, \cdots, x_n)$为一个从总体分布$F(x)$中观测到得样本, X^*为从x中随机选择的一个样本, 则

$$P(X^* = x_i) = \frac{1}{n}, \quad i = 1, \cdots, n.$$

从x中有放回的再抽样得到随机样本$X_1^*, \cdots, X_n^*$. 显然随机变量$X_1^*, \cdots, X_n^*$为 $i.i.d$的随机变量, 服从$\{x_1, \cdots, x_n\}$上的均匀分布.

经验分布函数$F_n(x)$是$F(x)$的估计, 可以证明, $F_n(x)$是$F(x)$的充分统计量. 而且另一方面, $F_n(x)$ 本身是$\{x_1, \cdots, x_n\}$上的均匀分布随机变量X^*的分

布函数. 因此在Bootstrap中有这个逼近. F_n逼近到F, Bootstrap重复下的经验分布函数F_n^*是F_n的逼近. 从x中再抽样,等价于从F_n中产生随机样本. 这两种 逼近可以表示为

$$F \to X \to F_n$$

$$F_n \to X^* \to F_n^*$$

从x中产生一个Bootstrap随机样本可以这样实现, 先从$\{1, 2, \cdots, n\}$中有放回的选取n次 得到$\{i_1, \cdots, i_n\}$, 然后得到Bootstrap样本$x^* = (x_{i_1}, \cdots, x_{i_n})$.

假设θ是我们感兴趣的参数(向量), $\hat{\theta}$为θ的估计. 则$\hat{\theta}$的分布的Bootstrap 估计可以通过如下方法得到

1. 对Bootstrap重复的第b次$(b = 1, \ldots, B)$,

   (a) 通过有放回的从$x_1, \cdots, x_n$中抽样得到再抽样样本$x^{*(b)} = x_1^*, \ldots, x_n^*$.

   (b) 根据$x^{*(b)}$计算$\hat{\theta}^{(b)}$.

2. $F_{\hat{\theta}}(\cdot)$的Bootstrap估计为$\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$的经验分布函数.

例1 F_n与Bootstrap抽样 假设我们观察到样本

$$x = \{2, 2, 1, 1, 5, 4, 4, 3, 1, 2\}$$

从x中再抽样依照选择$1, 2, 3, 4, 5$的概率分别为$0.3, 0.3, 0.1, 0.2, 0.1$进行. 从而

从x中随机选择 的一个样本X^*, 其分布函数就是经验分布函数, 即

$$F_{X^*}(x) = F_n(x) = \begin{cases} 0, & x < 1; \\ 0.3, & 1 \le x < 2; \\ 0.6, & 2 \le x < 3; \\ 0.7, & 3 \le x < 4; \\ 0.9, & 4 \le x < 5; \\ 1, & x \ge 5. \end{cases}$$

注意如果F_n没有靠近F_X, 则重复抽样下的分布也不会靠近F_X. 上例中的样本x实际上是从$Poisson(2)$中随机产生的, 从x中大量重复抽样可以很好的估计F_n, 但是不能很好的估计F_X, 因为无论重复多少次再抽样, 得到的Bootstrap样本都没有0.

1.1.1 Bootstrap Estimation of Standard Error

估计量$\hat{\theta}$的标准差的Bootstrap估计, 是Bootstrap重复$\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$ 的样本标准差:

$$\hat{se}_B(\hat{\theta}^*) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}^{(b)} - \overline{\hat{\theta}^*})^2}.$$

其中$\overline{\hat{\theta}^*} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{(b)}$.

根据Efron和Tibshirini(1993), 要得到标准差一个好的估计, 重复的次数B并非需要非常大. $B = 50$常常已经足够了, $B > 200$是很少见的(置信区间除外).

例2 (标准差的Bootstrap估计) **bootstrap**包里的法律院校数据集**law**, 记录了15所法律院校入学考试的平均成绩(LSAT)和GPA(乘了100).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LSAT	576	635	558	578	666	580	555	661	651	605	653	575	545	572	594
GPA	339	330	281	303	344	307	300	343	336	313	312	274	276	288	296

估计LSAT和GPA之间的相关系数, 并求样本相关系数的标准差的Bootstrap估计.

在本例中

1. 数据是成对的$(x_i, y_i), i = 1, \ldots, 15$.

2. 可以通过样本相关系数估计相关系数

$$\hat{\tau} = \frac{n\sum_i x_i y_i - \sum_i x_i \sum_i y_i}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}.$$

3. Bootstrap对这些数据对再抽样.

因此, 算法如下

1. 对Bootstrap重复的第b次$(b = 1, \ldots, B)$,
   (a) 通过有放回的从$x_1, \cdots, x_n$中抽样得到再抽样样本 $x^{*(b)} = x_1^*, \ldots, x_n^*$. 这里$x_i$或者$x_i^*$为一个向量.
   (b) 根据$x^{*(b)}$计算$\hat{\tau}^{(b)}$.
2. $F_{\hat{\tau}}(\cdot)$的Bootstrap估计为$\hat{\tau}^{(1)}, \ldots, \hat{\tau}^{(B)}$的经验分布函数.

样本相关系数为$cor(LSAT, GPA) = 0.7763745$, 使用Bootstrap估计标准差的程序如下:

↑Code

```
library(bootstrap)    #for the law data
print(cor(law$LSAT, law$GPA))
#set up the bootstrap
B <- 200              #number of replicates
n <- nrow(law)        #sample size
R <- numeric(B)       #storage for replicates
#bootstrap estimate of standard error of R
for (b in 1:B) {
    #randomly select the indices
    i <- sample(1:n, size = n, replace = TRUE)
    LSAT <- law$LSAT[i]       #i is a vector of indices
    GPA <- law$GPA[i]
    R[b] <- cor(LSAT, GPA)
}
#output
print(se.R <- sd(R))
hist(R, prob = TRUE)
```

↓Code

$se(\hat{\tau})$的Bootstrap估计为0.1371913, 样本相关系数的标准差的理论值

为0.115.

例3 使用boot函数进行Bootstrap估计标准差 在R中, 包**boot**里的**boot**函数可以进行Bootstrap估计. **boot** 函数中的参数*statistic*是 一个函数, 用来返回感兴趣的统计量值. 这个函数必须至少有两个参数, 其中第一个是数据, 第二个表示Bootstrap 抽样中的指标向量, 频率或者权重等. 因此我们首先写一个函数计算$\hat{\theta}^{(b)}$. 用$i = (i_1, \cdots, i_n)$ 表示指标向量, 则计算相关系数的程序为

↑Code

```
tau<-function(x,i){
xi<-x[i,]
cor(xi[,1],xi[,2])
}
```

↓Code

然后我们就可以使用**boot**函数进行Bootstrap估计:

↑Code

```
library(boot)        #for boot function
obj <- boot(data = law, statistic = tau, R = 2000)
```

```
obj
# alternative method for std.error
y <- obj$t
sd(y)
detach(package:boot)
```

↓Code

观测到的$\hat{\theta}$值用$t1*$标出. 2000次重复下的Bootstrap标准差估计为 0.1326418.

和**boot**函数相似功能的函数是**bootstrap**包里的**bootstrap**函数. 使用此函数重复上述问题的程序如下

↑Code

```
library(bootstrap)        #for boot function
n <- 15
theta <- function(i,x){ cor(x[i,1],x[i,2]) }
results <- bootstrap(1:n,2000,theta,law)
sd(results$thetastar) #0.1325971
detach(package:bootstrap)
```

↓Code

两个函数的用法上有些差异, **bootstrap**包是收录了Efron & Tibshirani的书里的程序和数据. **boot** 包是收录了Davson & Hinkley的书里的程序和数据.

1.1.2 Bootstrap Estimation of Bias

θ的一个估计量$\hat{\theta}$的偏差定义为

$$bias(\hat{\theta}) = E\hat{\theta} - \theta.$$

当$\hat{\theta}$的分布未知或者形式很复杂使得期望的计算不可能(从此分布中抽样变得很困难, Monte Carlo方法不可行), 以及在现实中, 我们也不知道θ的真值时(需要估计), 这种情况下偏差是未知的. 但是我们已经有了样本, $\hat{\theta}$是θ的估计, 而期望$E\hat{\theta}$可以通过Bootstrap方法进行估计. 从而 可以得到偏差的估计:

$$\widehat{bias}_B(\hat{\theta}) = E^*\hat{\theta}^* - \hat{\theta}.$$

E^*表示Bootstrap经验分布.

因此一个估计量的偏差的Bootstrap估计, 是通过使用当前样本下的估计量$\hat{\theta}$来估计θ, 而 使用$\hat{\theta}$的Bootstrap重复来估计$E\hat{\theta}$. 对一个有限样本$x = (x_1, \cdots, x_n)$, 有$\hat{\theta}(x)$的B个i.i.d估计量$\hat{\theta}^{(b)}$. 则$\{\hat{\theta}^{(b)}\}$的均值是期望值$E\hat{\theta}^*$的无偏估计, 因此偏差的Bootstrap估计为

$$\widehat{bias}_B(\hat{\theta}) = \overline{\hat{\theta}^*} - \hat{\theta}.$$

这里$\overline{\hat{\theta}^*} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{(b)}$. 正的偏差意味着$\hat{\theta}$平均来看过高估计了$\theta$; 而负的偏差意味着$\hat{\theta}$平均来看过低估计了$\theta$. 因此, 一个经过偏差修正(Bias-correction)的估计量为

$$\tilde{\theta} = \hat{\theta} - \widehat{bias}_B(\hat{\theta}).$$

例4 Bootstrap偏差估计: 估计上例中样本相关系数的偏差

↑Code

```
theta.hat <- cor(law$LSAT, law$GPA)
#bootstrap estimate of bias
B <- 2000   #larger for estimating bias
```

```
n <- nrow(law)
theta.b <- numeric(B)
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    LSAT <- law$LSAT[i]
    GPA <- law$GPA[i]
    theta.b[b] <- cor(LSAT, GPA)
}
bias <- mean(theta.b - theta.hat)
bias
```

↓Code

这个值和例3中的**boot**函数返回的结果非常相近.

例5 Bootstrap偏差估计: 假设$x = (x_1, \cdots, x_{10}) \sim N(\mu, \sigma^2)$, 求$\sigma^2$的估计量 $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$的偏差

↑Code

```
n<-10
x<-rnorm(n,mean=0,sd=10)
```

```
sigma2.hat<-(n-1)*var(x)/n
#bootstrap estimate of bias
B <- 2000   #larger for estimating bias
sigma2.b <- numeric(B)
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    sigma2.b[b] <-(n-1)*var(x[i])/n
}
bias <- mean(sigma2.b - sigma2.hat)
bias
```

↓Code

在这种情形下, $\hat{\sigma}^2$过低的估计了参数σ^2.

例6 比值参数估计的偏差的Bootstrap估计. 以包**bootstrap**里的**patch**数据为例. 该数据是测量了8个人使用3种不同的药物后血液中某种荷尔蒙的含量. 这三种药物分别是安慰剂, 旧药品(经过FDA审批的), 新药品(某个新工厂相同的工艺下生产的, 按FDA规定, 新工厂生产的药品也要审批). 研究的目的是比较新药和旧药的等价性. 如果可以证明新药和旧药之间的等价性, 则对新

药就不需要完全重新向FDA申请审批了. 等价性的标准是对比值参数

$$\theta = \frac{E(new) - E(old)}{E(old) - E(placebo)}.$$

若$|\theta| \leq 0.20$, 则新药和旧药就等价. 估计θ的统计量为$\bar{Y}/\bar{Z}$. 这两个变量在**patch**数据中给出. 我们的目标是计算此估计偏差的Bootstrap估计.

↑Code

```
data(patch, package = "bootstrap")
patch
n <- nrow(patch)  #in bootstrap package
B <- 2000
theta.b <- numeric(B)
theta.hat <- mean(patch$y) / mean(patch$z)
#bootstrap
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    y <- patch$y[i]
    z <- patch$z[i]
    theta.b[b] <- mean(y) / mean(z)
    }
```

```
bias <- mean(theta.b) - theta.hat
se <- sd(theta.b)
print(list(est=theta.hat, bias = bias,
           se = se, cv = bias/se))
```

Code

1.2 Jackknife

Jackknife(刀切法)是由Quenouille(1949,1956)提出的再抽样方法. Jackknife类似于"leave-one-out"的交叉验证方法. 令$x = (x_1, \ldots, x_n)$为观测到的样本, 定义第i个Jackknife样本为丢掉第i个样本后的剩余样本, 即

$$x_{(i)} = (x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n).$$

若$\hat{\theta} = T_n(x)$, 则定义第i个Jackknife重复为$\hat{\theta}_{(i)} = T_{n-1}(x_{(i)}),\ i = 1, \cdots, n.$

假设参数$\theta = t(F)$, 为分布F的函数. F_n为F的经验分布函数. 则θ的"plug-in"估计 为$\hat{\theta} = t(F_n)$. 称一个"plug-in"估计$\hat{\theta}$是平滑的, 如果数据的小幅变化相应于$\hat{\theta}$ 的小幅变化.

偏差的Jackknife估计

如果$\hat{\theta}$为一个平滑的(plug-in)估计量, 则$\hat{\theta}_{(i)} = t(F_{n-1}(x_{(i)})$, 以及 偏差的Jackknife估计(Quenouille)为

$$\widehat{bias}_{jack} = (n-1)(\overline{\hat{\theta}_{(\cdot)}} - \hat{\theta}),$$

其中$\overline{\hat{\theta}_{(\cdot)}} = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)}$.

我们以θ为总体方差为例来说明为什么偏差的Jackknife估计中系数是$n-1$. 由于方差的"plug-in"估计为

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

估计量$\hat{\theta}$是σ^2的有偏估计, 偏差为 $bias(\hat{\theta}) = E\hat{\theta} - \sigma^2 = -\frac{\sigma^2}{n}$. 每一个Jackknife估计是基于样本量$n-1$的样本构造, 因此Jackknife重复$\hat{\theta}_{(i)}$ 的偏差为$-\frac{\sigma^2}{n-1}$. 所以

$$E[\hat{\theta}_{(i)} - \hat{\theta}] = E[\hat{\theta}_{(i)} - \theta] - E[\hat{\theta} - \theta]$$
$$= bias(\hat{\theta}_{(i)}) - bias(\hat{\theta}) = -\frac{\sigma^2}{n-1} - (-\frac{\sigma^2}{n}) = \frac{bias(\hat{\theta})}{n-1}.$$

所以, 在Jackknife偏差估计的定义中有系数$n-1$.

例7 偏差的Jackknife估计 计算**patch**数据中比值参数的估计偏差的Jackknife估计.

↑Code

```
data(patch, package = "bootstrap")
n <- nrow(patch)
y <- patch$y
z <- patch$z
theta.hat <- mean(y) / mean(z)
print (theta.hat)
```

```
#compute the jackknife replicates, leave-one-out estimates
theta.jack <- numeric(n)
for (i in 1:n)
    theta.jack[i] <- mean(y[-i]) / mean(z[-i])
bias <- (n - 1) * (mean(theta.jack) - theta.hat)
print(bias)  #jackknife estimate of bias
```

↓Code

标准差的Jackknife 估计

对平滑的统计量$\hat{\theta}$, 其标准差的Jackknife估计(Tukey)定义为

$$\hat{se}_{jack} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(i)} - \overline{\hat{\theta}_{(\cdot)}})^2}.$$

比如当θ为总体均值时, $\hat{\theta} = \bar{x}$, 其方差估计为

$$Var(\hat{\theta}) = \frac{\hat{\sigma}^2}{n} = \frac{1}{n(n-1)}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

记$\theta_{(i)} = \frac{n\bar{x}-x_i}{n-1}$, 则$\overline{\hat{\theta}_{(\cdot)}} = \frac{1}{n}\hat{\theta}_{(i)} = \hat{\theta}$, $\hat{\theta}_{(i)} - \overline{\hat{\theta}_{(\cdot)}} = \frac{\bar{x}-x_i}{n-1}$. 因此有

$$\hat{se}_{jack} = \sqrt{Var(\hat{\theta})}.$$

例8 标准差的Jackknife估计 计算**patch**数据中比值参数的估计标准差的Jackknife估计.

↑Code

```
se <- sqrt((n-1) *
    mean((theta.jack - mean(theta.jack))^2))
print(se)
```

↓Code

Jackknife失效情形

若估计量$\hat{\theta}$不够平滑, Jackknife方法就可能会失效. 中位数就是一个不平滑统计量的例子.

例9 (Jackknife方法失效) 用Jackknife方法估计从1,2,...,100中随机抽取的10个数的中位数的标准差.

↑Code

```
set.seed(123) #for the specific example given
#change the seed to see other examples
n <- 10
x <- sample(1:100, size = n)
#jackknife estimate of se
M <- numeric(n)
for (i in 1:n) {         #leave one out
    y <- x[-i]
    M[i] <- median(y)
}
Mbar <- mean(M)
print(sqrt((n-1)/n * sum((M - Mbar)^2)))
#bootstrap estimate of se
Mb <- replicate(1000, expr = {
        y <- sample(x, size = n, replace = TRUE)
        median(y) })
print(sd(Mb))
```

本例中Jackknife估计和Bootstrap估计相差很远, 显然存在问题. 事实上, 由于中位数不是平滑的, Jackknife方法失效了.

1.3 Jackknife-after-Bootstrap

前面我们介绍了使用一个估计量的偏差和标准差的Bootstrap估计. 这些估计本身又是估计量, 那么这些估计量 的方差该如何估计呢? 一种方法就是使用Jackknife方法来估计这些估计量的方差.

注意到$\hat{se}(\hat{\theta})$是B次$\hat{\theta}$的Bootstrap重复统计量的样本标准差, 那么如果我们丢掉 第i个样本, 则Jackknife算法就是对每个i, 从剩余的$n-1$个样本值中再抽样B次, 来计算$\hat{se}(\hat{\theta}_{(i)})$ (Bootstrap过程), 即一个Jackknife重复. 最后我们得到

$$\hat{se}_{jack}(\hat{se}_B(\hat{\theta})) = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{se}_{B(i)}(\hat{\theta}) - \overline{\hat{se}_{B(\cdot)}(\hat{\theta})})^2}$$

其中$\overline{\hat{se}_{B(\cdot)}(\hat{\theta})} = \frac{1}{n}\sum_{i=1}^{n}\hat{se}_{B(i)}(\hat{\theta})$. 即对每个$i$, 我们将重复Bootstrap本身. 这当然是效率低下的, 庆幸的是有方法可以避免重复Bootstrap.

Jackknife-after-Bootstrap 方法是对每个"leave-one-out"的Bootstrap样本计算一个估计. 具体如下:

记$x_i^* = (x_1^*, \ldots, x_n^*)$为一次Bootstrap抽样, $x_1^*, \cdots, x_B^*$表示样本大小为B的Bootstrap样本. 令$J(i)$表示Bootstrap样本 中不含x_i的那些样本指标; $B(i)$表示不含x_i的Bootstrap样本个数, 因此我们可以使用丢掉$B - B(i)$个含有 x_i的样本后其余的样本来计算一个Jackknife重复. 故标准差估计量的Jackknife估计为

$$\hat{se}_{jab}(\hat{se}_B(\hat{\theta})) = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{se}_{B(i)} - \overline{\hat{se}_{B(\cdot)}})^2},$$

其中

$$\hat{se}_{B(i)} = \sqrt{\frac{1}{B(i)}\sum_{j\in J(i)}[\hat{\theta}_{(j)} - \overline{\hat{\theta}_{(J(i))}}]^2},$$

$$\overline{\hat{\theta}_{(J(i))}} = \frac{1}{B(i)} \sum_{j \in J(i)} \hat{\theta}_{(j)}.$$

例 10 (Jackknife-after-Bootstrap), 对例6中标准差的Bootstrap估计$\hat{se}_B(\hat{\theta})$, 使用 Jackknife-after-Bootstrap 方法估计其标准差.

↑Code

```
# initialize
data(patch, package = "bootstrap")
n <- nrow(patch)
y <- patch$y
z <- patch$z
B <- 2000
theta.b <- numeric(B)
# set up storage for the sampled indices
indices <- matrix(0, nrow = B, ncol = n)

# jackknife-after-bootstrap step 1: run the bootstrap
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    y <- patch$y[i]
```

```
    z <- patch$z[i]
    theta.b[b] <- mean(y) / mean(z)
    #save the indices for the jackknife
    indices[b, ] <- i
    }

#jackknife-after-bootstrap to est. se(se)
se.jack <- numeric(n)
for (i in 1:n) {
    #in i-th replicate omit all samples with x[i]
    keep <- (1:B)[apply(indices, MARGIN = 1,
                  FUN = function(k) {!any(k == i)})]
    se.jack[i] <- sd(theta.b[keep])
}

print(sd(theta.b))
print(sqrt((n-1) * mean((se.jack - mean(se.jack))^2)))
```

↓Code

1.4 Bootstrap Confidence Intervals

本节中我们介绍几种在Bootstrap中构造目标参数的渐近置信区间的方法, 其中包括 标准正态Bootstrap置信区间, 基本的Bootstrap置信区间, Bootstrap百分位数(percentile)置信区间和 Bootstrap t 置信区间.

1.4.1 The Standard Normal Bootstrap Confidence Interval

标准正态Bootstrap置信区间是一种比较简单的方法. 假设$\hat{\theta}$是参数 θ的估计量, 以及估计量的标准差为$se(\hat{\theta})$. 若$\hat{\theta}$渐近到正态分布, 即

$$Z = \frac{\hat{\theta} - E\hat{\theta}}{se(\hat{\theta})}$$

渐近服从标准正态分布. 则若$\hat{\theta}$为θ的无偏估计, 那么θ的一个渐近的$100(1-\alpha)\%$ 标准正态 Bootstrap 置信区间为

$$\hat{\theta} \pm z_{\alpha/2} \hat{se}_B(\hat{\theta}),$$

其中$z_{\alpha/2} = \Phi^{-1}(1-\alpha/2)$. 此区间容易计算, 但是有正态性假设或者CLT需成立. 以及 $\hat{\theta}$为θ的无偏估计.

1.4.2 The Percentile Bootstrap Confidence Interval

由形式

$$P(L \leq \hat{\theta} \leq U) = 1-\alpha$$

知, 可以使用Bootstrap重复的样本百分位数来估计L和U. 而$\hat{\theta}$为θ的估计, 因此就取 θ的$1-\alpha$置信区间上下界分别为Bootstrap重复的样本$1-\alpha/2$百分位数$\hat{\theta}^*_{[(B+1)(1-\alpha/2)]}$ 和$\alpha/2$百分位数$\hat{\theta}^*_{[(B+1)\alpha/2]}$.

Efron & Tibshirani 证明了百分位数区间相比于标准正态区间有着更好的理论覆盖率. 下面我们还会介绍 *bias-corrected and accelerated*(BCa) 百分位数区间, 它是百分位数区间的一个修正版本, 有着更好的理论性质以及在应用中有着更好的覆盖率.

1.4.3 The Basic Bootstrap Confidence Interval

由

$$P(L < \hat{\theta} - \theta < U) = 1 - \alpha$$

在$\hat{\theta} - \theta$的分布未知时, 由于Bootstrap重复$\hat{\theta}^*$的样本分位数$\hat{\theta}^*_{[(B+1)\alpha/2]}$和$\hat{\theta}^*_{[(B+1)(1-\alpha/2)]}$满足

$$P(\hat{\theta}^*_{[(B+1)\alpha/2]} - \hat{\theta} \le \theta^* - \hat{\theta} \le \hat{\theta}^*_{[(B+1)(1-\alpha/2)]} - \hat{\theta}) \approx 1 - \alpha.$$

因此$100(1-\alpha)\%$置信区间为

$$(2\hat{\theta} - \hat{\theta}^*_{[(B+1)(1-\alpha/2)]}, 2\hat{\theta} - \hat{\theta}^*_{[(B+1)\alpha/2]})$$

boot包里的函数**boot.ci**计算五种类型的置信区间: 基本的, 正态, 百分位数, 学生化和BCa.

例 11 **patch**数据比值统计量的Bootstrap 置信区间

本例说明如何使用**boot**和**boot.ci**函数得到正态,基本的和百分位数 Bootstrap置信区间. 下面的代码计算 比值统计量的95%置信区间.

↑Code

```
library(boot)       #for boot and boot.ci
 data(patch, package = "bootstrap")
 theta.boot <- function(dat, ind) {
     #function to compute the statistic
     y <- dat[ind, 1]
     z <- dat[ind, 2]
     mean(y) / mean(z)
 }
 y <- patch$y
 z <- patch$z
 dat <- cbind(y, z)
 boot.obj <- boot(dat, statistic = theta.boot, R = 2000)
 print(boot.obj)
 print(boot.ci(boot.obj,
               type = c("basic", "norm", "perc")))
```

↓Code

注意当$|\theta| < 0.2$时, 旧药和新药才被认为是等价的. 因此区间估计没有支持旧药和新药的等价性. 下面 我们根据Bootstrap置信区间的定义计算置信区间, 和上面的结果相对比.

↑Code

```
#calculations for bootstrap confidence intervals
 alpha <- c(.025, .975)
 #normal
 print(boot.obj$t0 + qnorm(alpha) * sd(boot.obj$t))
 #basic
 print(2*boot.obj$t0 -
     quantile(boot.obj$t, rev(alpha), type=1))
 #percentile
 print(quantile(boot.obj$t, alpha, type=6))
```

↓Code

例 12 **patch数据中相关系数的Bootstrap置信区间** 对**law**数据, 计算相关统计量的95%置信区间.

↑Code

```
library(boot)
   data(law, package = "bootstrap")
   boot.obj <- boot(law, R = 2000,
            statistic = function(x, i){cor(x[i,1], x[i,2])})
   print(boot.ci(boot.obj, type=c("basic","norm","perc")))
```

↓Code

三种置信区间都覆盖住了$\rho = .76$(此时通过完整数据集**law82**计算的). 百分位数置信区间和正态 置信区间的差异在于样本相关系数的分布是不是靠近正态分布. 当统计量的分布很靠近正态时, 百分位数 区间和正态区间就会一致.

1.4.4 The Bootstrap t interval

即使当$\hat{\theta}$的分布是正态分布, 且$\hat{\theta}$为θ的无偏估计, $Z = (\hat{\theta} - \theta)/se(\hat{\theta})$ 的分布也不会一定是正态的, 这是因为我们估计了$se(\hat{\theta})$. 我们也不能说Z的分布是t分布, 因为 Bootstrap估计$\hat{se}(\hat{\theta})$的分布未知. Bootstrap t区间并没有使用t分布作为推断分布. 而是使用 再抽样方法得到一个"t类型"的统计量的分布. 假

设$x = (x_1, \ldots, x_n)$为观测到得样本, 则$100(1-\alpha)\%$ Bootstrap t 置信区间为

$$(\hat{\theta} - t^*_{1-\alpha/2}\hat{se}(\hat{\theta}), \hat{\theta} - t^*_{\alpha/2}\hat{se}(\hat{\theta}))$$

其中$\hat{se}(\hat{\theta}), t^*_{\alpha/2}$ 和$t^*_{1-\alpha/2}$由下面的方法计算:

Bootstrap t 区间(学生化的Bootstrap区间)

1. 计算观测到得$\hat{\theta}$.
2. 对每个重复, $b = 1, \cdots, B$:
   (a) 从x中有放回的抽样得到第b个样本$x^{(b)} = (x_1^{(b)}, \ldots, x_n^{(b)})$.
   (b) 由第b个再抽样样本计算$\hat{\theta}^{(b)}$.
   (c) 计算标准差估计$\hat{se}(\hat{\theta}^{(b)})$.(对每个Bootstrap样本$x^{(b)}$, 再单独进行一个Bootstrap估计).
   (d) 计算第b个重复下的"t"统计量: $t^{(b)} = (\hat{\theta}^{(b)} - \hat{\theta})/\hat{se}(\hat{\theta}^{(b)})$.
3. 重复样本$t^{(1)}, \ldots, t^{(B)}$的分布作为推断分布. 找出样本分位数$t^*_{\alpha/2}$和$t^*_{1-\alpha/2}$.
4. 计算$\hat{se}(\hat{\theta})$, 即Bootstrap重复$\{\hat{\theta}^{(b)}\}$的样本标准差.

5. 计算置信界 $(\hat{\theta} - t^*_{1-\alpha/2}\hat{se}(\hat{\theta}), \hat{\theta} - t^*_{\alpha/2}\hat{se}(\hat{\theta}))$.

Bootstrap t区间的一个缺点是要再次使用Bootstrap方法得到标准差的估计$\hat{se}(\hat{\theta}^{(b)})$. 这是在Bootstrap 里面嵌套Bootstrap. 若$B = 1000$, 则Bootstrap t 区间方法需要比别的方法1000倍的时间.

例 13 Bootstrap t区间 本例我们写一个函数来计算一元或者多元样本下Bootstrap t 置信区间. 默认的置信水平为95%, Bootstrap重复数为500, 估计标准差的重复次数默认为100.

↑Code

```
boot.t.ci <-
   function(x, B = 500, R = 100, level = .95, statistic){
       #compute the bootstrap t CI
       x <- as.matrix(x);  n <- nrow(x)
       stat <- numeric(B); se <- numeric(B)
       boot.se <- function(x, R, f) {
           #local function to compute the bootstrap
           #estimate of standard error for statistic f(x)
           x <- as.matrix(x); m <- nrow(x)
```

```
        th <- replicate(R, expr = {
            i <- sample(1:m, size = m, replace = TRUE)
            f(x[i, ])
            })
        return(sd(th))
    }
    for (b in 1:B) {
        j <- sample(1:n, size = n, replace = TRUE)
        y <- x[j, ]
        stat[b] <- statistic(y)
        se[b] <- boot.se(y, R = R, f = statistic)
    }
    stat0 <- statistic(x)
    t.stats <- (stat - stat0) / se
    se0 <- sd(stat)
    alpha <- 1 - level
    Qt <- quantile(t.stats, c(alpha/2, 1-alpha/2), type = 1)
    names(Qt) <- rev(names(Qt))
    CI <- rev(stat0 - Qt * se0)
}
```

↓Code

例14 **patch**数据下比值统计量的Bootstrap t 置信区间

程序如下:

↑Code

```
#boot package and patch data were loaded in Example 7.10
#library(boot)        #for boot and boot.ci
#data(patch, package = "bootstrap")
dat <- cbind(patch$y, patch$z)
stat <- function(dat) {
    mean(dat[, 1]) / mean(dat[, 2]) }
ci <- boot.t.ci(dat, statistic = stat, B=2000, R=200)
print(ci)
```

↓Code

1.5 Better Bootstrap Confidence Intervals

对百分位数区间进行修正可以得到更好的Bootstrap置信区间, 其具有 更好的

理论性质和更好的实际覆盖率. 对$100(1-\alpha)\%$置信区间, 使用两个因子来调整常用的$\alpha/2$和$1-\alpha/2$分位数: 一个偏差(bias)的修正, 一个偏度(skewness)的修正. 偏差 的修正记为z_0, 偏度或者"加速"修正记为a. 更优的Bootstrap置信区间也常 称为BCa.

$100(1-\alpha)\%$ BCa 置信区间为: 先计算

$$\alpha_1 = \Phi(\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1-\hat{a}(\hat{z}_0 + z_{\alpha/2})}),$$

$$\alpha_2 = \Phi(\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1-\hat{a}(\hat{z}_0 + z_{1-\alpha/2})})$$

其中$z_\alpha = \Phi^{-1}(\alpha)$. $\hat{z}_0, \hat{a}$由下面的式子计算. 则 BCa 区间为

$$(\hat{\theta}^*_{\alpha_1}, \hat{\theta}^*_{\alpha_2}).$$

BCa 区间的下界和上界是Bootstrap重复的经验的α_1和α_2分位数.

偏差修正因子实际上是测量$\hat{\theta}$的重复$\hat{\theta}^*$的中位数偏差. 其估计为

$$\hat{z}_0 = \Phi^{-1}(\frac{1}{B}\sum_{b=1}^{B} I(\hat{\theta}^{(b)} < \hat{\theta})).$$

加速因子是从Jackknife重复中估计:

$$\hat{a} = \frac{\sum_{i=1}^{n}(\overline{\theta_{(\cdot)}} - \theta_{(i)})^3}{6\sum_{i=1}^{n}((\overline{\theta_{(\cdot)}} - \theta_{(i)})^2)^{3/2}}.$$

$\hat{a}$之所以称为是加速因子, 是因为它估计的是相对于目标参数θ, $\hat{\theta}$的标准差的变化率. 我们在使用标准正态Bootstrap置信区间时, 是假设方差为一个常数, 与θ无关. 但是很多时候方差 都可能和θ有关. 加速因子的目的就是要考虑到估计量的方差可能会与目标参数有关, 因此对置信界进行调整.

BCa方法的来源可以参看阅读材料.

BCa的性质

BCa Bootstrap置信区间有两个重要的理论性质:

一是不变性, 即若θ的置信区间为$(\hat{\theta}^*_{\alpha_1}, \hat{\theta}^*_{\alpha_2})$, $g(\cdot)$为一一变换函数, 则$g(\theta)$得置信区间为$(g(\hat{\theta}^*_{\alpha_1}), g(\hat{\theta}^*_{\alpha_2}))$.

另外一个性质是二阶精确性, 即误差以$1/n$的速度趋于0.

Bootstrap t 置信区间是二阶精确的, 但是不具有不变性. Bootstrap 百分位数区间有不变性, 但是不是二阶精确的; 标准正态置信区间既没有不变性, 也没有二阶精确性.

例 15 BCa Bootstrap 置信区间 本例是写一个函数来计算BCa Bootstrap置信区间.

↑Code

```
boot.BCa <-
function(x, th0, th, stat, conf = .95) {
    # bootstrap with BCa bootstrap confidence interval
    # th0 is the observed statistic
    # th is the vector of bootstrap replicates
    # stat is the function to compute the statistic
    x <- as.matrix(x)
    n <- nrow(x) #observations in rows
```

```
    N <- 1:n
    alpha <- (1 + c(-conf, conf))/2
    zalpha <- qnorm(alpha)
    # the bias correction factor
    z0 <- qnorm(sum(th < th0) / length(th))
    # the acceleration factor (jackknife est.)
    th.jack <- numeric(n)
    for (i in 1:n) {
        J <- N[1:(n-1)]
        th.jack[i] <- stat(x[-i, ], J)
    }
    L <- mean(th.jack) - th.jack
    a <- sum(L^3)/(6 * sum(L^2)^1.5)
    # BCa conf. limits
    adj.alpha <- pnorm(z0 + (z0+zalpha)/(1-a*(z0+zalpha)))
    limits <- quantile(th, adj.alpha, type=6)
    return(list("est"=th0, "BCa"=limits))
}
```

↓Code

例 16 (BCa 置信区间) 计算**patch**数据中比值统计量的BCa置信区间.

↑Code

```
n <- nrow(patch)
B <- 2000
y <- patch$y
z <- patch$z
x <- cbind(y, z)
theta.b <- numeric(B)
theta.hat <- mean(y) / mean(z)
#bootstrap
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    y <- patch$y[i]
    z <- patch$z[i]
    theta.b[b] <- mean(y) / mean(z)
    }
#compute the BCa interval
stat <- function(dat, index) {
    mean(dat[index, 1]) / mean(dat[index, 2])  }
boot.BCa(x, th0 = theta.hat, th = theta.b, stat = stat)
```

↓Code

在结果中, $\alpha/2 = 0.025$和$1 - \alpha/2 = 0.975$被分别调整为0.0334和0.982.

例 17 使用函数**boot.ci**计算上例中的BCa Bootstrap 置信区间

↑Code

```
#using x from Example 16
 boot.obj <- boot(x, statistic = stat, R=2000)
 boot.ci(boot.obj, type=c("perc", "bca"))
```

↓Code

此例中也计算了百分位数置信区间.

1.6 Application: Cross Validation

交叉验证(Cross Validation)是一种分割数据方法, 其可以用来验证参数估计的稳健性, 分类算法的准确度, 模型的合理性等等. Jackknife 可以被视为是交叉验证的一种特例, 其主要用来估计偏差和估计量的标准差.

最简单的交叉验证方法是所谓的"holdout" 方法. 其将数据随机分为训练集(training set)和测试集(testing set)两个子集. 然后仅使用训练集样本进行建模, 然后通过测试集来对模型进行评估. 其优点是training/testing 比例 不依赖于重复次数. 其缺点是依赖于数据的分割方式, 某些点可能从不进入到测试集中, 而某些点可能多次进入测试集. 这种方法呈现出"Mente Carlo"波动性, 即随机分割不同, 结果会波动.

"K-fold"交叉验证是对"holdout"方法的改进, 其将数据分割为K个子集, 然后重复"holdout"方法K次. 每次第i个子集被作为测试集来评估模型, 其余的$K-1$个子集被作为训练集进行建模. 最后计算K次 的平均误差. 其优点是对数据的分割方式依赖性不是很强, 每个点仅有一次在测试集中, 而在训练集中有$K-1$次. 因此估计的方差会随着K的增加而减少. 缺点是计算的时间复杂度增加.

"leave-one-out" 交叉验证是"K-fold"交叉验证的一个特例($K=n$), 仅使用一个点作为测试集, 其余的点 作为训练集. 其缺点是计算的时间复杂度

可能会比较高.

例 18 模型选择问题 包**DAAG**里的**ironslag**数据描述了两种方法(chemical, magnetic)测量含铁量的53次结果. 散点图显示chemical和magnetic变量是正相关的, 但是关系可能不是线性的. 从散点图上可以看出, 二次多项式, 或者可能一个指数的, 或对数模型能更好的拟合数据.

本例我们使用交叉验证来进行模型选择. 通过交叉验证来估计模型的预测误差. 候选的模型有

1. 线性模型: $Y = \beta_0 + \beta_1 X + e$.
2. 二次的: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + e$.
3. 指数: $log(Y) = log(\beta_0) + \beta_1 X + e$.
4. log-log: $log(Y) = \beta_0 + \beta_1 log(X) + e$.

四种模型的参数估计程序如下

↑Code

```
par(ask = TRUE)
```

```
library(DAAG); attach(ironslag)
a <- seq(10, 40, .1)     #sequence for plotting fits

L1 <- lm(magnetic ~ chemical)
plot(chemical, magnetic, main="Linear", pch=16)
yhat1 <- L1$coef[1] + L1$coef[2] * a
lines(a, yhat1, lwd=2)

L2 <- lm(magnetic ~ chemical + I(chemical^2))
plot(chemical, magnetic, main="Quadratic", pch=16)
yhat2 <- L2$coef[1] + L2$coef[2] * a + L2$coef[3] * a^2
lines(a, yhat2, lwd=2)

L3 <- lm(log(magnetic) ~ chemical)
plot(chemical, magnetic, main="Exponential", pch=16)
logyhat3 <- L3$coef[1] + L3$coef[2] * a
yhat3 <- exp(logyhat3)
lines(a, yhat3, lwd=2)

L4 <- lm(log(magnetic) ~ log(chemical))
plot(log(chemical), log(magnetic), main="Log-Log", pch=16)
```

```
logyhat4 <- L4$coef[1] + L4$coef[2] * log(a)
lines(log(a), logyhat4, lwd=2)
```

↓Code

然后我们使用交叉验证来对每个模型的预测误差进行估计. 算法如下

1. 对$k = 1, \ldots, n$, 令(x_k, y_k)为检验样本, 使用其余样本对模型参数进行估计. 然后计算预测误差.
(a) 使用其余的样本对模型进行拟合.
(b) 计算预测值: $\hat{y}_k = \hat{\beta}_0 + \beta_1 x_k$.
(c) 计算预测误差: $e_k = y_k - \hat{y}_k$.
2. 计算均方预测误差: $\sigma_e^2 = \frac{1}{n}\sum_{i=1}^{n} e_k^2$.

计算程序如下

↑Code

```
n <- length(magnetic)
e1 <- e2 <- e3 <- e4 <- numeric(n)

# for n-fold cross validation
# fit models on leave-one-out samples
```

```
for (k in 1:n) {
    y <- magnetic[-k]
    x <- chemical[-k]

    J1 <- lm(y ~ x)
    yhat1 <- J1$coef[1] + J1$coef[2] * chemical[k]
    e1[k] <- magnetic[k] - yhat1

    J2 <- lm(y ~ x + I(x^2))
    yhat2 <- J2$coef[1] + J2$coef[2] * chemical[k] +
            J2$coef[3] * chemical[k]^2
    e2[k] <- magnetic[k] - yhat2

    J3 <- lm(log(y) ~ x)
    logyhat3 <- J3$coef[1] + J3$coef[2] * chemical[k]
    yhat3 <- exp(logyhat3)
    e3[k] <- magnetic[k] - yhat3

    J4 <- lm(log(y) ~ log(x))
    logyhat4 <- J4$coef[1] + J4$coef[2] * log(chemical[k])
    yhat4 <- exp(logyhat4)
```

```
    e4[k] <- magnetic[k] - yhat4
}
c(mean(e1^2), mean(e2^2), mean(e3^2), mean(e4^2))
```

↓Code

结果表明二次多项式回归的预测误差最小. 我们可以使用**plot(L2)**进行模型诊断.