统计计算第四次作业

2019302010132 邓凌云

Exercise 7.2

思路分析

Jackknife-after-bootstrap 方法:

对每个"leave-one-out"的Bootstrap样本计算估计，其估计方法如下:

1. 通过 $Bootstrap$ 方法得到样本大小为B的样本 $x_1^*,\ x_2^*,\ \ldots,\ x_B^*$
2. 令 $J(i)$ 表示Bootstrap样本中的不包含 x_i 的样本指标， $B(i)$ 表示不含 x_i 的样本个数
3. 丢掉 $B-B(i)$ 个含有 x_i 的样本后其余样本来计算一个 $Bootstrap$ 重复
4. 得到标准差估计量的 $Jackknife$ 估计

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

代码实现

```
##initialize
library(bootstrap) ##get law data
n <- nrow(law)
B <- 2000
R <- numeric(B)
indices <- matrix(0, nrow = B, ncol = n)

##bootstrap for se(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)
```

```
    #save the indices for the jackknife
    indices[b,] <- i
}

#Jackknife-after-bootstrap
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(R[keep])
}

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

结果解释

```
> print(sd(R))
[1] 0.1344679 ##bootstrap estimate of se(R)
> print(sqrt((n-1) * mean((se.jack - mean(se.jack))^2)))
[1] 0.08361427 ##the standard error of the bootstrap estiamte
```

从最终结果可以看出， $Jackknife-after-bootstrap$ 方法成功的估计了 $se(R)$ 的标准差。

Exercise 7.3

思路分析

在Example 7.2中，通过 $Bootstrap$ 方法估计了 Law 数据集的相关系数，在本问题中，我们将目光投向这一估计的置信区间，并计划采用 $The\ Bootstrap\ t\ interval$ 方法来估计。

该方法虽然叫做 $The\ Bootstrap\ t\ interval$ ，但是实际上由于 $Bootstrap$ 估计的 $\hat{se}(\hat{\theta})$ 的分布未知，因此并未使用 t 分布作为推断分布，而是通过再抽样方法得到一个"t类型"的统计量的分布。其 $100(1-\alpha)\%$ 置信区间为

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

代码实现

首先根据 $The\ Bootstrap\ t\ interval$ 的基本步骤编写应用函数 `boot.t.ci` 。其中

> x为输入数据集；B代表Bootstrap估计的次数，默认为500；R代表估计标准差的重复次数，默认为100
>
> 默认置信水平为 95% ； statistic需要输入一个参数仅包含数据集x的函数

```
##bootstrap t interval method
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, fun) {
            #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)
                fun(x[i, ])
            })
            return(sd(th))
        }

        ##re-sample for
        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, fun = statistic)
        }

        ##estimate the distribution for quantile
        stat0 <- statistic(x) #original statistic
        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)
        return(CI)
    }
```

之后调用该函数，计算law数据的相关系数Bootstrap估计的置信区间：

```
##use function boot.t.ci
library(bootstrap) # get law data
stat <- function(x) { cor(x[, 1], x[,2]) } # calculate the
correlation
ci <- boot.t.ci(law, statistic = stat, B=2000, R=200)
print(ci)
```

结果解释

程序运行结果如下

```
> print(ci)
      2.5%      97.5%
-0.2051370  0.9979151
```

程序运行时便能感受到的明显缺点是，该方法即使在重复次数较少的情况下，也需耗费较长时间。这是由于该方法实际上在Bootstrap中再次嵌套了Bootstrap，重复次数B越大，则计算效率就越低。

Exercise 7.4

思路分析

假定 **aircondit** 中的数据服从指数模型 $Exp(\lambda)$ ，那么接下来的目标就是估计参数 λ 。

首先求出 λ 的 MLE 估计，利用如下公式：

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} x_i}$$

接着我们可以使用 $Bootstrap$ 方法来估计该估计量 $\hat{\lambda}$ 的偏差和标准差。其公式如下：

$$\hat{bias}_B(\hat{\theta}) = \bar{\hat{\theta}}^* - \hat{\theta}$$

这里 $\bar{\hat{\theta}}^* = \frac{1}{B}\sum_{b=1}^{B}(\hat{\theta}^{(b)})$ 。正的偏差值意味着平均来看 MLE 过高估计了 θ。

标准差的计算公式即为

代码实现

可以使用 `boot` 包中的 `boot` 函数直接计算偏差和标准差。

```
library(boot) # get aircondit data # get boot function
mle <- function(x,i){ 1/mean(x[i]) }
aircondit.boot
    <- boot(data = as.matrix(aircondit), statistic = mle, R =
2000)
```

结果解释

最终输出结果为

```
Bootstrap Statistics :
      original      bias    std. error
t1* 0.00925212 0.001353267 0.004214177
```

其中 $oringinal$ 即为对样本直接进行 MLE，$bias$ 代表通过 $Bootstrap$ 方法计算出的估计量的偏差，而 $std.error$ 为通过 $Bootstrap$ 方法计算出的估计量的标准差。

Exercise 7.5

思路分析

针对Example 7.4中通过 $Bootstrap$ 方法获得的结果，利用 $the\ standard\ normal,\ basic,\ percentile$ 和 BCa 方法计算估计量置信区间。

代码实现

直接调用 `boot` 包中的函数 `boot.ci` 来计算各种方式下的置信区间。

```
library(boot) # get aircondit data and function boot.ci
##a simple method
aircondit0 <- as.matrix(aircondit)
meantime <- function(x,i) mean(x[i])
##get boot class
aircondit.boot <- boot(data = aircondit0, statistic = meantime, R
= 5000)
boot.ci(aircondit.boot, type = c("norm", "basic", "perc", "bca"))
```

结果解释

代码运行输出结果如下：

```
> boot.ci(aircondit.boot, type = c("norm", "basic", "perc",
"bca"))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

Intervals :
Level      Normal              Basic
95%   ( 35.0, 181.2 )   ( 27.1, 169.3 )

Level     Percentile            BCa
95%   ( 46.8, 189.1 )   ( 57.1, 223.5 )
Calculations and Intervals on Original Scale
```

四种置信区间估计方法都很好的覆盖了样本平均等待时间估计值 $\hat{\lambda} = 108.1$；

但是可以明显看出，使用四种不同的方法获得的置信区间存在较大差异。

这是由于该估计量的分布较正态分布有较大的偏离，从而使得标准正态置信区间与百分数区间偏离；

而BCa置信区间是对百分数区间作出了修正，其具有更好的理论应用和覆盖率。

Exercise 7.7

思路分析

根据题目可知，对主成分分析中第一主成分的方差贡献率可以通过特征值来计算

$$\theta = \frac{\lambda_1}{\sum_{i=1}^{n} \lambda_i}$$

实际上，往往都是利用样本来对方差贡献率作出估计，其估计公式如下

$$\hat{\theta} = \frac{\hat{\lambda}_1}{\sum_{i=1}^{n} \hat{\lambda}_i}$$

得到样本估计结果后，由于具体总体分布未知，可通过 $Bootstrap$ 方法对估计量 $\hat{\theta}$ 的偏差和标准差进行估计。

代码实现

```
library(bootstrap) # get scor data
```

1. `theta` 函数用来估计样本第一主成分的方差贡献率

```
# default: perform centralized and standardized process
theta <- function(data, center = TRUE, scale = TRUE) {
    Sigma.hat <-
        cov(scale(as.matrix(data),center = center, scale =
scale))
    e.hat <- eigen(Sigma.hat) # calculate the eigenvalue and
eigen-vector

    ## estimate the proportion of the first principal component
    theta.hat <- e.hat$values[1]/sum(e.hat$values)

    return(theta.hat)
}
```

2. 针对 `scor` 所有样本估计

```
## use all sample
theta.all <- theta(scor)
```

3. 利用 $Bootstrap$ 方法估计偏差和标准差

```
## use bootstrap method
 # initial
B <- 2000; n <- nrow(scor)
theta.B <- numeric(B)
 # bootstrap
theta.B <- replicate(B, expr = {
    i <- sample(1:n, size = n, replace = TRUE)
    scor.B <- as.matrix(scor)[i,]
    theta.B <- theta(scor.B)
})

## for bias and standard error
bias <- mean(theta.B-theta.all)
std <- sd(theta.B)
```

结果解释

输出结果显示为

```
> print(result)
    original       bias.B        std.B
 0.636196030 -0.000893874  0.043038363
```

Exercise 7.8

思路分析

仍然使用Exercise 7.7的具体思路，将估算偏差和标准差的方法更换为 $Jackknife$ 方法。

偏差的 $Jackknife$ 估计公式为

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

标准差的 $Jackknife$ 估计公式为

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

代码实现

利用 $Jackknife$ 方法估算偏差和标准差

```
##use jackknife method
 # initial
n <- nrow(scor)
theta.jack <- numeric(n)
 # jackknife
for (i in 1:n)
    theta.jack[i] <- theta(as.matrix(scor)[-i,])

## for bias and standard error
bias.jack <- (n-1)*mean(theta.jack-theta.all)
std.jack <- sqrt((n-1)*mean((theta.jack-mean(theta.jack))^2))
```

结果解释

输出结果为

```
> print(result.jack)
     original     bias.jack      std.jack
 0.6361960298 -0.0003778535  0.0446668393
```

相比 $Bootstrap$ 方法偏差有所减少。

Exercise 7.10

思路分析

该问题中候选的模型有

1. 线性模型： $Y = \beta_0 + \beta_1 X + e$
2. 二次模型： $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + e$
3. 三次模型： $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + e$
4. 指数模型： $log(Y) = log(\beta_0) + \beta_1 X + e$

采用交叉验证来进行模型选择，即通过交叉验证的方法来估计模型的预测误差，并选出误差最小的模型继续进行之后的分析。

代码实现

首先获取 `DAAG` 中的数据 `ironslag`

```
library(DAAG); attach(ironslag) # get ironslag data
```

随后采用 $leave - one - out$ 方法进行交叉验证

```
# for n-fold cross validation
n <- length(magnetic)
e1 <- e2 <- e3 <- e4 <- numeric(n)

# fit models on leave-one-out samples
for (k in 1:n) {
    y <- magnetic[-k]
    x <- chemical[-k]

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

    #quadratic
    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

    #cubic
    J3 <- lm(y ~ x + I(x^2) + I(x^3))
```

```
    yhat3 <- J3$coef[1] + J3$coef[2] * chemical[k] +
        J3$coef[3] * chemical[k]^2 + J3$coef[4] * chemical[k]^3
    e3[k] <- magnetic[k] - yhat3

    #exponential
    J4 <- lm(log(y) ~ x)
    logyhat4 <- J4$coef[1] + J4$coef[2] * chemical[k]
    yhat4 <- exp(logyhat4)
    e4[k] <- magnetic[k] - yhat4
}

# calculate the error and select the minimum
crossv <- c(mean(e1^2), mean(e2^2), mean(e3^2), mean(e4^2))
best.cross.ind <- which(crossv == min(crossv), arr.ind = T)
```

直接使用回归并比较适应的 R^2

```
# fit models on adjusted R squared
L1 <- summary(lm(magnetic ~ chemical))
L2 <- summary(lm(magnetic ~ chemical + I(chemical^2)))
L3 <- summary(lm(magnetic ~ chemical + I(chemical^2) +
I(chemical^3)))
L4 <- summary(lm(log(magnetic) ~ chemical))

# gain adjusted R squared
adj.r <-

 c(L1$adj.r.squared,L2$adj.r.squared,L3$adj.r.squared,L4$adj.r.squ
ared)
best.adj.ind <- which(adj.r == max(adj.r),arr.ind = T)
```

结果解释

最终结果整理后得到

```
> print(best.ind)
best.cross.ind   best.adj.ind
             2              2
```

可以看出，利用两种不同方法最终得到的最优模型均为模型2，即

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + e$$

具体拟合结果为 $Y = 24.49 - 1.39X + 0.05X^2 + e$