Assignment 6

Nancy Zhang and Kyle Forrester

2025-10-21

```
set.seed(1234)
nboot  <- 10000 #outer bootstrap reps
nboot2 <- 400 #inner bootstrap reps for SE estimation (nested bootstrap)
alpha  <- 0.05 #this will be for 95% CIs
```

Setup

XY Package This package is not commented, as Professor LuValle stated that if the code inside is not used, it is not necessary to comment. We will only be using the `special.sample` data inside.

```
xy.pck <-
c("xy.pck", "my.boot.xy", "my.boot.xy.conf", "my.boot.xy.reduce",
"my.bootstrap.exp", "special.sample", "my.cp.extract.lars1",
"my.cp.extract.leaps2")
my.boot.xy <-
function(mat.train=x.auto2.leaps,y=y.auto,xstring="leaps",brep=10){
#specialized version of bootstrap
if(xstring=="lars"){
library(lars)
func0<-lars
reduce.function<-my.cp.extract.lars2
}
if(xstring=="leaps"){
library(leaps)
func0<-leaps
reduce.function<-my.cp.extract.leaps2
}
ypredmat<-NULL
residmat<-NULL
betamat<-NULL
n1<-length(mat.train[,1])
#
out0<-reduce.function(func0(mat.train,y),mat.train,y)
beta0<-c(out0$beta)
I1<-rep(T,length(beta0))
beta00<-beta0

betamat<-rbind(betamat,beta00)
#
for(i in 1:brep){
if((i/2)==floor(i/2)){
print(c(i,brep))
```

```
}
#
v1<-sort(sample(n1,n1,replace=T))
#
m0<-mat.train
m1<-(m0[,I1])[v1,]
y1<-(y)[v1]
#
out1<-reduce.function(func0(m1,y1),m1,y1)
#
beta1<-out1$beta
betamat<-rbind(betamat,beta1)

}
#

bagged.beta<-apply(betamat,2,mean)
boxplot(betamat)
list(bagged.beta=bagged.beta,orig.beta=beta00,mat=betamat)
}
my.boot.xy.conf <-
function(mat.train=auto.q.train,mat.test=auto.q.pred,yind=1,xstring="lars",
brep=10000,pred.int=T,alpha=.05){
#specialized version of bootstrap
if(xstring=="lars"){
library(lars)
func0<-lars
reduce.function<-my.cp.extract.lars1
}
if(xstring=="leaps"){
library(leaps)
func0<-leaps
reduce.function<-my.cp.extract.leaps2
}
ypredmat<-NULL
residmat<-NULL
betamat<-NULL
n1<-length(mat.train[,1])
#
out0<-reduce.function(func0(mat.train[,-yind],mat.train[,yind]),mat.train,mat.test,yind)
ypred0<-c(out0$ypredict0)
resid0<-c(out0$resid)
beta0<-c(out0$beta)
ypredmat<-rbind(ypredmat,ypred0)
residmat<-rbind(residmat,resid0)
betamat<-rbind(betamat,beta0)
#
for(i in 1:brep){
if((i/500)==floor(i/500)){
print(c(i,brep))
}
#
v1<-sort(sample(n1,n1,replace=T))
```

```
#
m1<-(mat.train[,-yind])[v1,]
y1<-(mat.train[,yind])[v1]
#
out1<-reduce.function(func0(m1,y1),mat.train,mat.test,yind)
#
ypred1<-c(out1$ypredict0)
resid1<-c(out1$resid)
beta1<-c(out1$beta)
ypredmat<-rbind(ypredmat,ypred1)
residmat<-rbind(residmat,resid1)
betamat<-rbind(betamat,beta1)

}
#
bagged.pred<-apply(ypredmat,2,mean)
bagged.beta<-apply(betamat,2,mean)
quant.boot<-function(x){quantile(x,c(alpha/2,1-alpha/2))}
#
if(pred.int){
main1<-paste("Prediction interval",xstring,"alpha=",alpha)
qboot<-apply(ypredmat+residmat,2,quant.boot)
}
else{
main1=paste("Confidence interval,",xstring,"alpha=",alpha)
qboot<-apply(ypredmat,2,quant.boot)
}
#
y0<-mat.test[,yind]
plot(rep(bagged.pred,5),c(y0,ypred0,bagged.pred,qboot[1,],qboot[2,]),
xlab="Bagged prediction",ylab="Data and intervals",type="n",main=main1)
points(bagged.pred,y0)
lines(bagged.pred,bagged.pred)
o1<-order(bagged.pred)
lines(bagged.pred[o1],ypred0[o1],col=2)
lines(bagged.pred[o1],smooth(qboot[1,o1]),col=3)
lines(bagged.pred[o1],smooth(qboot[2,o1]),col=3)
#
list(bpred=bagged.pred,ypred0=ypred0,type=xstring,bagged.beta=bagged.beta,
orig.beta=beta0,pred.int=pred.int)
}
my.boot.xy.reduce <-
function(mat.train=x.auto2,y=y.auto,xstring="leaps",brep=1000){
#specialized version of bootstrap
if(xstring=="lars"){
library(lars)
func0<-lars
reduce.function<-my.cp.extract.lars2
}
if(xstring=="leaps"){
library(leaps)
func0<-leaps
reduce.function<-my.cp.extract.leaps2
```

```
}
ypredmat<-NULL
residmat<-NULL
betamat<-NULL
n1<-length(mat.train[,1])
#
out0<-reduce.function(func0(mat.train,y),mat.train,y)
beta0<-c(out0$beta)
I1<-abs(beta0)>0
beta00<-beta0[I1]

betamat<-rbind(betamat,beta00)
#
for(i in 1:brep){
if((i/200)==floor(i/200)){
print(c(i,brep))
}
#
v1<-sort(sample(n1,n1,replace=T))
#
m0<-mat.train
m1<-(m0[,I1])[v1,]
y1<-(y)[v1]
#
out1<-reduce.function(func0(m1,y1),m1,y1)
#
beta1<-out1$beta
betamat<-rbind(betamat,beta1)

}
#

bagged.beta<-apply(betamat,2,mean)
boxplot(betamat)
list(bagged.beta=bagged.beta,orig.beta=beta00,mat=betamat)
}
my.bootstrap.exp <-
function(vec0,statfunc,nboot=100)
{
#
    n0<-length(vec0)
    stat0<-statfunc(vec0)
#
    bootvec<-NULL
#
    for( i in 1:nboot){
        vecb<-sample(vec0,replace=T)
#
        statb<-statfunc(vecb)
#
        bootvec<-c(bootvec,statb)
    }
#
```

```
    list(stat0=stat0,bootmean=mean(bootvec),bootvar=var(bootvec))


}
special.sample <-
c(23.840599479756122, 4.3842774727366027, 5.0956688866002846,
2.3566543172090144, -23.949803213259941, 9.3509427290409803,
0.65251206234097481, 6.8365429651532175, 24.896640079698329,
-8.7617700151623517, 25.814657481989627, 5.3152770334854722,
2.8398049478656984, -12.096616427505259, 4.397545337677002, 5.4355953823630578,
18.144806428903586, 1.9786876970902085, 16.530886256647232, -4.9056228227437755,
6.2044959254562855, -8.9589436515690117, 13.812075705778721,
9.6438114456832409, 1.0270954776654413, 29.053179855129486, 8.0951351039111614,
8.7905355421826243, 9.3145597372204065, 3.873737707734108, 8.6849553808569908,
4.1409478062881835, -16.871777369731905, -3.3649728987365961,
16.548116152686003, 1.3435630658641458, -6.5022478735991474,
-8.9998905617268115, 22.518533006100256, -23.222754490905523,
2.9830907224918586, -14.888938206339123, -12.837568901053807,
1.953628201991169, 3.4240511415893384, 3.6827530963346362, -11.468025827420057,
12.27449047523451, 13.405985074580519, 29.737901833066353, 7.1429840913042426,
-16.755755010152104, -13.138900882569185, -1.2381440894678235,
6.7195787066593766, 6.6611230992719683, 1.0132466223852241, -9.3566439929873972,
1.7830456765368581, -4.3367369263888618, 6.4618188533931971,
11.953485022483543, 5.2667434718459845, 8.0009208358824253, 1.5598284753420515,
0.86587769400678294, -3.3896422414109111, -13.331382337117436,
11.487637599416272, 0.89456674007777126, 12.273026674251872,
5.3552341060712934, 12.734275228855134, 0.35275325831025839,
-0.62435123417526484, 5.1114548249170184, -5.647560053005936,
-5.7916095364363205, 4.1071316817436179, 10.312835827049735,
-4.8025107267768403, -14.866977098489048, 13.547340474602702,
12.075118191120149, 2.6460608680859949, -2.4452916979789734,
-25.219439385641106, -0.80708191078156233, 1.6723475670441985,
-1.0564956841990352, 8.1480371737852693, 8.2783658867701888,
1.8028011561703869, -23.964070327659641, 39.920698090685924,
-11.726356807255987, 6.3345346059650183, 2.3614248326048255,
14.125069800344232, 4.207889268836146)
my.cp.extract.lars1 <-
function(str,matrix.train=auto.q.train,matrix.test=auto.q.pred,yindex=1)
{
#str is the structure createing Cp and some choice vector, choosen.part is a string
#first load the leaps library
        #
        I1<-(str$Cp==min(str$Cp))
        s1<-c(1:length(I1))[I1]
        #
        #
        xmat.train<-matrix.train[,-yindex]
        ymat.train<-matrix.train[,yindex]
        yp0<-predict(str,xmat.train,s1)$fit
      resid<-ymat.train-yp0
        xmat.pred<-matrix.test[,-yindex]
        yp1<-predict(str,xmat.pred,s1)$fit
        npred<-length(yp1)
```

```
#
        resp<-sample(resid,npred,replace=T)
        ypout<-yp1+resp
        list(ypredict0=yp1,resid=resp,beta=str$beta[I1,])

}
my.cp.extract.leaps2 <-
function(str,matrix.train,y)
{
#str is the structure creating Cp and some choice vector, choosen.part is a string
#
        #

        I1<-(str$Cp==min(str$Cp))
        which1<-str$which[I1,]
        #
        #
        xmat.train<-(matrix.train)[,which1]
        ymat.train<-y
        ls.train<-lsfit(xmat.train,ymat.train)
        coef0<-ls.train$coef
        which2<-c(1:length(which1))[which1]
        coef1<-which1
      coef1[which1]<-coef0[-1]
      list(beta=coef1)
}
```

```
#bootse
boot_se <- function(vec, stat_fn, nboot2 = 400) {
  n <- length(vec)
  idx <- matrix(sample.int(n, n*nboot2, replace = TRUE), nrow = n)
  Xb<- matrix(vec[idx], nrow = n)
  thetab <- apply(Xb, 2, stat_fn)
  sd(thetab)
}

#nested
boot_t_nested_ci <- function(vec, stat_fn = mean, nboot = 2000, nboot2 = 400, alpha = 0.05) {
  vec <- as.numeric(vec); vec <- vec[is.finite(vec)]
  n <- length(vec); stopifnot(n >= 3)

  theta0 <- stat_fn(vec)
  se0 <- boot_se(vec, stat_fn, nboot2)

  idx <- matrix(sample.int(n, n * nboot, replace = TRUE), nrow = n)
  tstar <- numeric(nboot)

  for (b in seq_len(nboot)) {
    xb <- vec[idx[, b]]
    thetab <- stat_fn(xb)
    seb <- boot_se(xb, stat_fn, nboot2)
    if (!is.finite(seb) || seb == 0) tstar[b] <- NA_real_ else tstar[b] <- (thetab - theta0) / seb
  }
```

```
  tstar <- tstar[is.finite(tstar)]
  stopifnot(length(tstar) > 10)

  qL <- quantile(tstar, 1 - alpha/2, na.rm = TRUE)
  qU <- quantile(tstar, alpha/2, na.rm = TRUE)
  ci <- c(theta0 - qL * se0, theta0 - qU * se0)
  list(estimate = theta0, se = se0, ci = unname(ci))
}

#attempt to do a loopless
boot_loopless_cis <- function(vec, stat_fn = mean, nboot = 2000, alpha = 0.05) {
  vec <- as.numeric(vec)
  vec <- vec[is.finite(vec)]
  n <- length(vec)
  theta0 <- stat_fn(vec)

  idx <- matrix(sample.int(n, n * nboot, replace = TRUE), nrow = n)
  Xb <- matrix(vec[idx], nrow = n)
  thetab <- apply(Xb, 2, stat_fn)

  ci_perc  <- stats::quantile(thetab, c(alpha/2, 1 - alpha/2), names = FALSE)
  q <- stats::quantile(thetab, c(1 - alpha/2, alpha/2), names = FALSE)
  ci_basic <- c(2 * theta0 - q[1], 2 * theta0 - q[2])

  list(estimate = theta0,
       percentile = unname(ci_perc),
       basic = unname(ci_basic))
}

#examples of user prov functions
stat_mean <- function(v) mean(v)
stat_median <- function(v) median(v)
trimmed_mean <- function(v, trim_percent = 20){
    v = sort(v)
    len = length(v)
    mean(v[floor(trim_percent / 200 * len) : len+1-ceiling(trim_percent / 200 * len)])
}
```

```
x <- special.sample
#mean using our new nested bootstrap-t
mean_nested <- boot_t_nested_ci(x, stat_fn = stat_mean,
                                nboot = nboot, nboot2 = nboot2, alpha = alpha)
#median using our new nested bootstrap-t
median_nested <- boot_t_nested_ci(x, stat_fn = stat_median,
                                  nboot = nboot, nboot2 = nboot2, alpha = alpha)
#attempt at loopless comparisons
mean_loopless <- boot_loopless_cis(x, stat_fn = stat_mean,   nboot = nboot, alpha = alpha)
median_loopless <- boot_loopless_cis(x, stat_fn = stat_median, nboot = nboot, alpha = alpha)

#presenting results and CI
tbl <- data.frame(
  Statistic = c("Mean (boot-t, nested)", "Median (boot-t, nested)",
                "Mean (percentile, loopless)", "Mean (basic, loopless)",
                "Median (percentile, loopless)", "Median (basic, loopless)"),
```

```
  Estimate  = c(mean_nested$estimate, median_nested$estimate,
                mean_loopless$estimate, mean_loopless$estimate,
                median_loopless$estimate, median_loopless$estimate),
  CI_Lower  = c(mean_nested$ci[1],    median_nested$ci[1],
                mean_loopless$percentile[1], mean_loopless$basic[1],
                median_loopless$percentile[1], median_loopless$basic[1]),
  CI_Upper  = c(mean_nested$ci[2],    median_nested$ci[2],
                mean_loopless$percentile[2], mean_loopless$basic[2],
                median_loopless$percentile[2], median_loopless$basic[2])
)

#results
knitr::kable(
  tbl,
  digits = 6,
  caption = sprintf(
    "95%% CIs on special.sample (n = %d). Outer nboot = %d, inner nboot2 = %d.",
    length(special.sample), nboot, nboot2))
```

Table 1: 95% CIs on special.sample (n = 100). Outer nboot = 10000, inner nboot2 = 400.

Statistic	Estimate	CI_Lower	CI_Upper
Mean (boot-t, nested)	2.882331	0.528588	5.222612
Median (boot-t, nested)	3.553402	1.661135	5.984231
Mean (percentile, loopless)	2.882331	0.590199	5.198429
Mean (basic, loopless)	2.882331	0.566233	5.174463
Median (percentile, loopless)	3.553402	1.783046	5.266743
Median (basic, loopless)	3.553402	1.840061	5.323759

Conclusion

We found that the point estimate for the mean was lower than that of the median, and that the loopless estimates (basic and percentile) had the same point estimates. However, the loopless confidence intervals were more narrow, implying that the error estimate was smaller. For both mean and median, the percentile estimate had more narrow bounds than the basic estimate. It is likely that the loopless functions yielded smaller intervals than the nested bootstrap because the bootstrap estimate for the standard error was larger than the error estimate used in the loopless function. The point estimates were the same because the mean of the data and the median of the data remain the same regardless of the method used.