P8160 Homework 4: Bootstrapping

Renjie Wei

4/11/2022

In this homework, we require the use of parallel computing codes for your implementations.

```
library(parallel)
library(foreach)
library(doParallel)
library(ggplot2)
library(tidyverse)
```

Problem 1: a randomized trial on an eye treatment

An ophthalmologist designed a randomized clinical trial to evaluate a new laser treatment in comparison to the traditional one. The response is visual acuity, measured by the number of letters correctly identified in a standard eye test. 20 patients have both eyes eligible for laser treatment. The ophthalmologist randomized the two laser treatments (new vs traditional) to the two eyes of those patients (i.e. one eye received the new laser treatment and the other receive traditional laser treatment). Another 20 patients had only one suitable eye, so they received one treatment allocated at random.So we have a mixture of paired comparison and two-sample data.

```
> blue <- c(4,69,87,35,39,79,31,79,65,95,68,
           62,70,80,84,79,66,75,59,77,36,86,
           39,85,74,72,69,85,85,72)
> red <-c(62,80,82,83,0,81,28,69,48,90,63,
        77,0,55,83,85,54,72,58,68,88,83,78,
        30,58,45,78,64,87,65)
> acui<-data.frame(str=c(rep(0,20),
            rep(1,10)),red,blue)
```

Answer the following question:

(1) The treatment effect of the new laser treatment is defined as

$$E(Y \mid \text{trt} = \text{new}) - E(Y \mid \text{trt} = \text{traditional}).$$

Estimate the treatment effect using the collected data.

(2) Use bootstrap to construct 95 % confidence interval of the treatment effect. Describe your bootstrap procedure, and what is your conclusion from the bootstrap CI?

Problem 1.1

```
blue <- c(4,69,87,35,39,79,31,79,65,95,68,
           62,70,80,84,79,66,75,59,77,36,86,
```

```
           39,85,74,72,69,85,85,72)
red <- c(62,80,82,83,0,81,28,69,48,90,63,
         77,0,55,83,85,54,72,58,68,88,83,78,
         30,58,45,78,64,87,65)
acui <- data.frame(str = c(rep(0,20),rep(1,10)),red,blue)
```

The treatment effect is defined as $E(Y \mid \text{trt} = \text{new}) - E(Y \mid \text{trt} = \text{traditional})$. So we calculate the raw treatment effect based on the data. Let blue laser be the new treatment.

```
raw_trt_eff <- mean(blue) - mean(red)
```

The raw treatment effect (the observed value of mean difference) is 3.067.

Problem 1.2

Raw treatment may not be a proper way to estimate the mean difference. I am going to use bootstrap sample to estimate the mean difference and its standard error.

Since there are paired structures in our data, when doing bootstrap, we're going to preserve this structure, so instead of bootstrap each observation, we bootstrap subjects for paired data. For un-paired data, we use the simple bootstrap. That is

- For 20 pairs of paired data, in each bootstrap replicate, random sample (with replacement) the number of subject
- For 10 unpaired data in each group, sample with replacement in each bootstrap sample
- Combine the two part of bootstrap replicates as the final bootstrap replicate
- RepeatB times

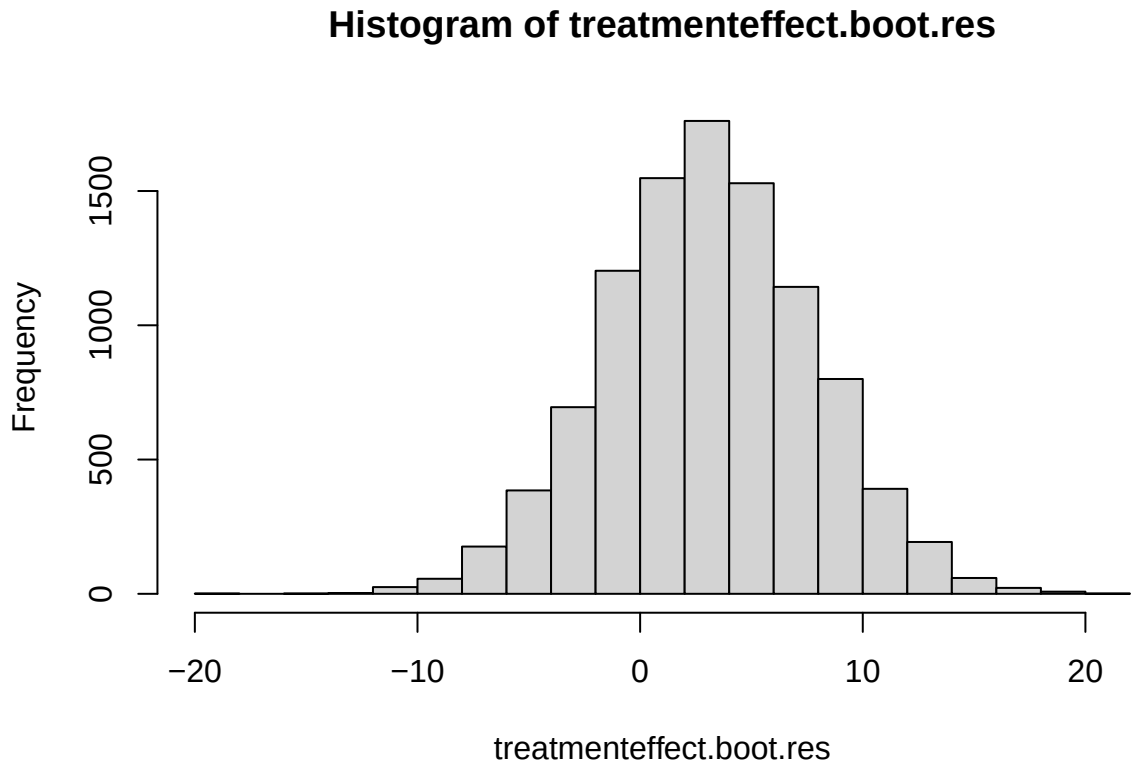
And the implementation of this paired bootstrap is shown in the `pairedboot()` function.

```
set.seed(2022)
# return whole bootstrap sample for future use
pairedboot <- function(paired, unpaired, nboot = 2000){
    numCores <- detectCores()
    registerDoParallel(numCores)
    # parallel computing implementation using foreach
        res <- foreach(icount(nboot), .combine=rbind) %dopar% {
           # bootstrap for paired data
           subject <- nrow(paired)
           pairedboot.ind <- sample(subject, subject, replace = T)
           pairedsamp <- paired[pairedboot.ind,]
           pairedboot.trt <- pairedsamp$blue
           pairedboot.ctrl <- pairedsamp$red
           # bootstrap for unpaired data
           unpaired.trt <- unpaired$blue
           unpaired.ctrl <- unpaired$red
           unpairedboot.trt <- sample(unpaired.trt, replace = T)
           unpairedboot.ctrl <- sample(unpaired.ctrl, replace = T)
           # combine two parts of bootstrap
           boot.trt <- c(pairedboot.trt, unpairedboot.trt)
           boot.ctrl <- c(pairedboot.ctrl, unpairedboot.ctrl)
           # b-th bootstrap estimate of treatment effect
           mean(boot.trt) - mean(boot.ctrl)
        }
        stopImplicitCluster()
```

```
        return(res)
}


acui.paired <- acui[which(acui$str == 0),]
acui.unpaired <- acui[which(acui$str == 1),]

system.time({treatmenteffect.boot.res <- pairedboot(acui.paired, acui.unpaired, 1e4)})
```

```
##    user  system elapsed
##    3.77    0.71    5.43
```

```
treatmenteffect.boot.se <- sqrt(var(treatmenteffect.boot.res))


hist(treatmenteffect.boot.res)
```



From 10000 bootstrap sample, we have a estimation of the treatment effect is 3.073 with a standard error 4.646. And from the histogram of the distribution of bootstrap estimates, we can see that it is approximately normal.

The $100(1-\alpha)\%$ confidence limits for the basic bootstrap confidence interval are (Statistical Computing with R, Page 226)

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

```r
set.seed(2022)
alpha = 0.05
boot.ci <- function(boot_est, alpha, raw_est){
    # use quantile function to get the upper and lower bound
    qt <- quantile(boot_est, c(alpha/2, 1-alpha/2), type = 1)
    names(qt) <- rev(names(qt))
    CI <- rev(2 * raw_est - qt)
    return(CI)
}

boot.CI <- boot.ci(treatmenteffect.boot.res, 0.05, raw_trt_eff)
t(boot.CI) %>%  knitr::kable(digits = 3, caption = "95% confidence interval of bootstrap estimate of tr
```

Table 1: 95% confidence interval of bootstrap estimate of treatment effect

2.5%	97.5%
-6.133	12.2

The 95% confidence interval of the treatment effect $\hat{\theta}$ is shown in the table 1. Since the confidence interval contains 0, so we conclude that at 95% confident level, we cannot say that there is significant difference in treatment effect between the newer treatment and the older one.

Problem 2

The Galaxy data consist of the velocities (in km/sec) of 82 galaxies from 6 well-separated conic sections of an unfilled survey of the Corona region. The structure in the distribution of velocities corresponds to the spatial distribution of galaxies in the far universe. In particular, a multimodal distribution of velocities indicates a strong heterogeneity in the spatial distribution of the galaxies and thus is seen as evidence for the existence of voids and super clusters in the far universe.

Statistically, the question of multimodality can be formulated as a test problem

$$H_0 : n_{\text{mode}} = 1 \quad \text{vs} \quad H_a : n_{\text{mode}} \geq 1$$

where $ n_{mode} $ is the number of modes of the density of the velocities.

Considered nonparametric kernel density estimates

$$\widehat{f}_{K,h}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - X_i}{h})$$

It can be shown that the number of modes in $\widehat{f}_{K,h}(x)$ decreases as h increase. Let H_1 be the minimal bandwidth for which $\widehat{f}_{K,H_1}(x)$ is unimodal. In the galaxy data, $h_1 = 3.05$

Since multimodal densities need more smoothing to become unimodal, the minimal bandwidth H_1 can be used as a test statistic, and one reject the null hypothesis if

$$\text{Prob}(H_1 > h_1) \leq \alpha$$

To evaluating the distribution of H_1 under the null, one could use the following bootstrap algorithm

1. draw B bootstrap samples if size n from $\widehat{f}_{K,h_1}(x)$
2. for each bootstrap, find $h_1^{*(b)}$, the smallest h for which this bootstrap sample has just 1 mode

3. approximate p-value of test is $\frac{\#h_1^{*(b)} > h_1}{B}$

Implement the algorithm above in R, apply it to the galaxy data, and report your findings. You may find the following R codes helpful.

```
library(MASS)
data(galaxies)
plot(density(galaxies/1000, bw=1.5))
plot(density(galaxies/1000, bw=3.5))

#calculate the number of modes in the density
  den <- density(galaxies/1000, bw=1.5)
  den.s <- smooth.spline(den$x, den$y, all.knots=TRUE, spar=0.8)
  s.1 <- predict(den.s, den.s$x, deriv=1)
  nmodes <- length(rle(den.sign <- sign(s.1$y))$values)/2
```

Answers

We are going to bootstrap from the sample distribution, which is the nonparametric kernel density function. Under the null hypothesis, we assume that this density function is unimodal, $h_1 = 3.05$

$$\widehat{f}_{K,h_1}(x) = \frac{1}{nh_1}\sum_{i=1}^{n} K(\frac{x - X_i}{h_1})$$

where K is a kernel function, in this case, we use `density` as an implementation of kernel density estimation. And the parameter `bw` represent the bandwidth we are interest.

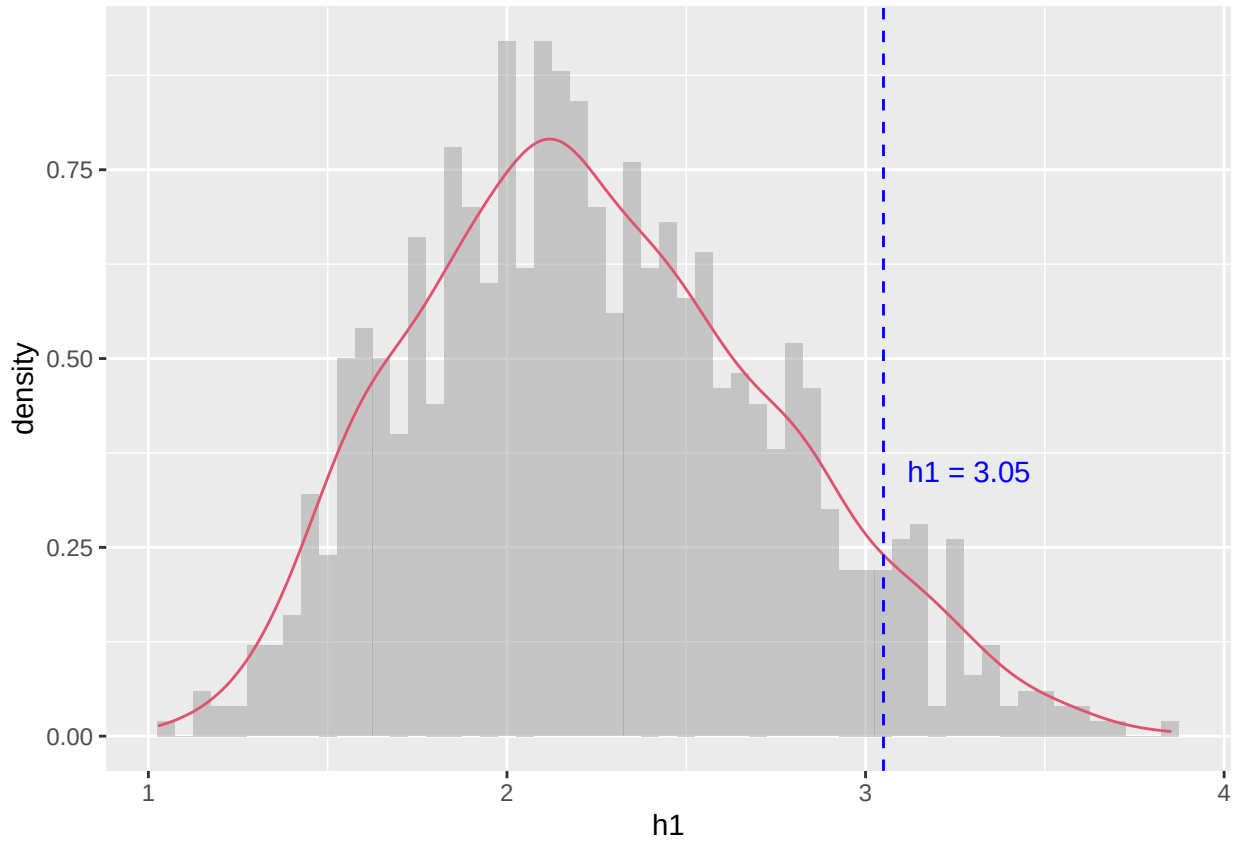
The following function `n.modes()` gives us the number of modes of the sample distribution for any given bandwidth `bw`.

Since in each bootstrap, we want to find $h_1^{*(b)}$, the smallest h for which this bootstrap sample has just 1 mode. Under our null hypothesis, $h_1 = 3.05$. So we apply the `n.modes()` to each bootstrap sample by setting a sequence of bandwidths. Here, we only focus on bandwidths close to h_1 . Since if the null hypothesis is true, the bootstrap estimates $h_1^{*(b)}$ should be centered around the h_1.

```
boot.galaxy <- function(data = galaxies, nboot = 1000){
    # get the number of modes
    n.modes <- function(data = galaxies, bw){
        den <- density(data/1000, bw = bw)
        den.s <- smooth.spline(den$x, den$y, all.knots = TRUE)
        s.1 <- predict(den.s, den.s$x, deriv = 1)
        nmodes <- length(rle(den.sign <- sign(s.1$y))$values)/2
        return(nmodes)
    }
    # try to search h1 in a prespecified sequence
    # the code below is low efficient, replaced by a binary search alg, an idea from Haolin Zhong
    #bwseq <- exp(seq(5, 12 , by = 0.01)/10)
    #get.h1 <- function(bootsamp, bwseq){
    #    nmodes = sapply(bwseq, FUN = function(x) n.modes(bootsamp, x))
    #    res.df <- data.frame(bw = bwseq, nmodes = nmodes)
    #    # return the minimum h gives us the unimodal distribution
    #    h1 <- min(res.df[which(res.df$nmodes == 1),]$bw)
    #    return(h1)
    #}

    # Binary search alg by Haolin
```

```
    # return h1 for the given sample
    get.h1 = function(data, upper = 10, tol = 1e-5){
        l = 1e-9
        r = upper
        while (r - l > tol) {
            bw = (l + r) / 2
            n = n.modes(data, bw)
            if (n == 1) {
                r = bw
            } else {
                l = bw
            }
        }
        # The above algorithm forces l to be slightly smaller than h1
        # So, we use ceiling function to make l slightly bigger,
        # which gives us the true h1 at 2 decimal place
        return(ceiling(100 * l) / 100)
    }


    boot.dens <- function(den){
        return(1000 * sample(den$x, size = 82, replace = T, prob = den$y))
    }

    galaxy.dens <- density(galaxies/1000, bw = 3.05)

    h1.res <- foreach(icount(nboot), .combine = cbind) %dopar% {
        boot <- boot.dens(galaxy.dens)
        get.h1(boot)
    }
    return(h1.res)
}
```

```
#test.nmodes <- sapply(exp(seq(10, 15 , by = 0.001)/10), FUN = function(x) n.modes(galaxies, x) )
cl <- makeCluster(12 , outfile = "")
registerDoParallel(cl)
res.galaxies <- boot.galaxy()
stopCluster(cl)
# calculate the p value of the
p_val_galaxy <- sum(res.galaxies > 3.05)/1000
galaxy_res_df <- data.frame(h1 = t(res.galaxies))


ggplot(galaxy_res_df, aes(x = h1)) +
    geom_histogram(aes(y = ..density..), binwidth = 0.05, alpha = 0.5, fill = 8) +
    geom_density(size = 0.5, col = 2) +
    geom_vline(xintercept = 3.05, lty = 2, color = 'blue') +
    annotate("text", x = 3.3, y = 0.35, label = "h1 = 3.05 ", color = "blue")
```



Our test has given a p-value of $0.073 > 0.05$, we fail to reject the null hypothesis that the number of modes is 1 concluding that the distribution of velocities is unimodal.

Problem 3 (the breast cancer study):

The data *breast-cancer2.csv* have 569 patients. The first column **ID** lables individual breast tissue images; The second column **Diagnonsis** identifies if the image is coming from cancer tissue or benign cases (M=malignant, B = benign). There are 357 benign and 212 malignant cases. The other 10 columns correspond to mean of the distributions of the following 10 features computed for the cellnuclei;

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness ($perimeter^2/area - 1$)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

Consider a logistic LASSO regression to predict cancer cases based on the image features.

1. Propose and implement a 5-fold cross-validation aglorithm to select the turning parameter in the logistic LASSO regression. We call the logistic-LASSO with CV-selected λ as the "optimal" logistic LASSO; The R function for logistic-LASSO
2. Using the selected predictors from the "optimal" logistic LASSO to predict the probability of malignant for each of the images (Note that estimates from logistic-Lasso are biased. You need to re-fit the logistic regression with the selected predictors to estimate the probability.) How well the predictors classify the images?
3. Using the bootstrapping smoothing idea to re-evaluate the probabilities of malignant. How well the new predictors classify the images?
4. Write a summary of your findings.

1. Propose and implement a 5-fold cross-validation algorithm to select the turning parameter in the logistic LASSO regression.

Answers

```
library(glmnet)
breast_dat <- read_csv("breast-cancer-2.csv") %>% select(-id) %>% mutate(diagnosis = as.factor(diagnosis

breast_y = breast_dat$diagnosis
breast_x = as.matrix(breast_dat %>% select(-diagnosis))
set.seed(81602844)

#Spliting the dataset to train and test, with proportion 80% and 20%
rowTrain = sample(1:nrow(breast_x), ceiling(nrow(breast_x) * 0.8))
y_train = breast_y[rowTrain]
x_train = breast_x[rowTrain,]
y_test = breast_y[-rowTrain]
x_test = breast_x[-rowTrain,]
```

```
# this is exactly what we do in each bootstrap sample later
optimalLasso <- function(lambdaseq, x_train, y_train, K =5){
    # By Miron Kursa https://mbq.me
    # fast implementation of auc
    auroc <- function(score, bool) {
      n1 <- sum(!bool)
      n2 <- sum(bool)
      U  <- sum(rank(score)[!bool]) - n1 * (n1 + 1) / 2
      return(1 - U / n1 / n2)
    }

    # for each lambda in lambda seq, do 5-fold cross-validation to get the cv accuracy
    lassocv <- function(cur, x, y, K) {
        folds <- sample(rep(1:K, length.out = nrow(x)))
            auclist <- c()
            for (i in 1:K){
            testidx <- which(folds == i)
            xtest <- x[testidx,]
            ytest <- y[testidx]
            xtrain <- x[-testidx,]
            ytrain <- y[-testidx]
            # model fit with given lambda
            modeltmp <- glmnet(xtrain, ytrain, standardize = TRUE, family = "binomial", alpha = 1, lambd
```

```
            # predict on the validation set
            ypred <- predict(modeltmp, xtest, type = "response")
            # calculate AUC
            auc <- auroc(ypred, ifelse(ytest == "B", FALSE, TRUE))
            # add cv result to list
            auclist <- c(auclist, auc)
        }
        # just return the mean accuracy
        return(mean(auclist))
    }

    cv.res <- sapply(lambdaseq, FUN = function(x) lassocv(x, x_train, y_train, 5))
    cv.res.df <- data.frame(lambdas = lambdaseq, auc = cv.res)
    # return best lambda on cv result
    best.lambda <- cv.res.df[which(cv.res.df$auc == max(cv.res.df$auc)),]$lambdas

    # refit lasso based on best lambda
    best.model <- glmnet(x_train, y_train, standardize = TRUE,  family = "binomial", alpha = 1, lambda =
    coef(best.model)

    best.model.coef <- as.matrix(coef(best.model))
    imp.predictors <- names(best.model.coef[best.model.coef != 0,])
    imp.predictors = imp.predictors[2:length(imp.predictors)]

    optimal.res <- list(lambda = best.lambda,
                        cvResult = cv.res.df,
                        bestModel = best.model,
                        impPredictor = imp.predictors)
    return(optimal.res)
}
```

```
cl2 <- makeCluster(10 , outfile = "")
registerDoParallel(cl2)
clusterExport(cl=cl2, c('optimalLasso','x_train','y_train'))
set.seed(81602844)
optimal.fit <- optimalLasso(exp(seq(-8, -1, length = 201)), x_train, y_train)
stopCluster(cl2)
```

The 5-fold cross-validation gives us a best lambda 0.0010648, and the selected important variables under this lambda are texture_mean, area_mean, smoothness_mean, concavity_mean, concave points_mean, symmetry_mean, fractal_dimension_mean.

2. Using the selected predictors from the "optimal" logistic LASSO to predict the probability of malignant for each of the images (Note that estimates from logistic-Lasso are biased. You need to re-fit the logistic regression with the selected predictors to estimate the probability.) How well the predictors classify the images?

Answer

```
auroc <- function(score, bool) {
    n1 <- sum(!bool)
    n2 <- sum(bool)
    U  <- sum(rank(score)[!bool]) - n1 * (n1 + 1) / 2
    return(1 - U / n1 / n2)
```

```
}

logit.fit <- glmnet(x_train[,optimal.fit$impPredictor], y_train, family = binomial(link = "logit"), lam
logit.train.prob <- predict(logit.fit, x_train[,optimal.fit$impPredictor], type = "response")
logit.test.prob <- predict(logit.fit, x_test[,optimal.fit$impPredictor], type = "response")
logit.train.auc <- auroc(logit.train.prob, ifelse(y_train == "B", FALSE, TRUE))
logit.test.auc <- auroc(logit.test.prob, ifelse(y_test == "B", FALSE, TRUE))
```

By refitting the logistic model using the important variables on the training dataset, we get a AUC = 0.99009 on training set and a AUC = 0.97159 on test set. Which means the selected predictors classify the images pretty well.

3. Using the bootstrapping smoothing idea to re-evaluate the probabilities of malignant. How well the new predictors classify the images?

Answer

From Efron's paper (2014, JASA), classical statistical theory ignores model selection in assessing estimation accuracy. Here we consider bootstrap methods for computing standard errors and confidence intervals that take model selection into account. The methodology involves *bagging*, also known as *bootstrap smoothing*, to tame the erratic discontinuities of selection-based estimators.

Point estimation:

- For each bootstrap in bootstrap with B times, select the best model and get estimates for the coefficient denoted as $t(y_i^*)$ $(i = 1, 2, \ldots, B)$.
- Smooth $\hat{\mu} = t(y)$ by averaging over the bootstrap replications, defining

$$\tilde{\mu} = s(y) = \frac{1}{B}\sum_{i=1}^{B} t(y_i^*)$$

Inference: Define $Y_{ij}^* = \sum_{k=1}^{n} I(y_{ik}^* = y_j)$ representing the number of times that j^{th} data point appears in the $ß^{th}$ bootstrap replicate, thus the vector Y_i^* follows a multinomial distribution with n draws on n categories each of probability $\frac{1}{n}$ in ideal bootstrap. In addition to the percentile confidence interval, the nonparametric delta-method estimate of standard deviation for $s(y)$ in the nonideal case is:

$$\tilde{sd}_B = [\sum_{j=1}^{n} \hat{cov}_j^2]^{1/2}$$

where

$$\hat{cov}_j = \sum_{i=1}^{B}(Y_{ij}^* - Y_{\cdot j}^*)(t_i^* - t_\cdot^*)/B$$

with $Y_{\cdot j}^* = \sum_{i=1}^{B} Y_{ij}^*/B$ and $t_\cdot^* = \sum_{i=1}^{B} t_i^*/B = s(y)$.

```
# return a list contains
# smooth.coef: bootstrap smoothed regression parameter
# smooth.sd: sd of smooth.coef
# smooth.std: standard error of smooth.coef
# coef.matrix: result of bootstrap regression parameters
# chosen.prob: probability of predictors to be choose
# best.lambdas: best lambda in each bootstrap sample

bootSmooth <- function(x_train, y_train, nboot, K){
    # delta-method
    coefInference <- function(x, Ys, coefMtx, predictors, nboot){
```

```
    var.smooth <- 0
    tdot <- mean(coefMtx[, predictors])
    for(j in 1:nrow(x)){
        cov.j <- 0
        yj <- mean(Ys[,j])
        # cant nested foreach loop...
        for (i in nboot) {
            cov.j <- cov.j + (Ys[i,j] - yj) * (coefMtx[i, predictors] - tdot)
        }
        var.smooth <- var.smooth + (cov.j/nboot)^2
    }
    sd.smooth <- sqrt(var.smooth)
    return(sd.smooth)
}

# bootstrap smooth main method
res.bs <- foreach(icount(nboot), .combine = rbind, .packages = c('glmnet')) %dopar% {
    coefList = rep(0, 10 + 1)
    names(coefList) = c("(Intercept)", colnames(x_train))
    # bootstrap
    rowBoot = sample(1:nrow(x_train), replace = TRUE)
    # Yij, the # of jth data appears
    Yij <- rep(0, length(y_train))
    for(j in 1:length(y_train)){
        Yij[j] = sum(j == rowBoot)
    }

    xboot <- x_train[rowBoot,]
    yboot <- y_train[rowBoot]
    # do the cv-lasso optimization
    bootcv <- optimalLasso(exp(seq(-8, -1, length = 201)), xboot, yboot)
    # get the best lambda and coef from the refited logit model
    # as well as the selected predictors in bestModel
    chosen <- bootcv$impPredictor
    logitfit = glmnet(x_train[,chosen], y_train, family = binomial(link = "logit"), lambda = 0)
    logitcoef = as.matrix(coef(logitfit))
    bestlambda <- bootcv$lambda

    for (p in c("(Intercept)", chosen)){
        coefList[p] = logitcoef[p, 1]
    }

    cbind(bestlambda, t(Yij), t(coefList))

}

Yij <- res.bs[, 2:(nrow(x_train) + 1)]
coefMtx <- res.bs[,(nrow(x_train) + 2):ncol(res.bs)]
# the smoothed sd
bs.sd <- sapply(c("(Intercept)", colnames(x_train)), FUN = function(x) coefInference(x_train, Yij,

res <- list(
    smooth.coef = colMeans(coefMtx),
```

```
        coef.smooth.sd = bs.sd,
        coef.std = apply(coefMtx, 2, sd),
        coef.matrix = coefMtx,
        chosen.prob = colMeans(coefMtx != 0),
        best.lambdas = res.bs[,1]
    )
    return(res)

}
```

```
set.seed(81602844)

cl3 <- makeCluster(10 , outfile = "log.out")
registerDoParallel(cl3)
clusterExport(cl = cl3, c('optimalLasso','x_train','y_train'))
bs.res <- bootSmooth(x_train, y_train, 1000, 5)
stopCluster(cl3)
```

```
bs.eta.train <- cbind(1, x_train) %*% bs.res$smooth.coef
bs.eta.test <- cbind(1, x_test) %*% bs.res$smooth.coef

bs.pred.train <- (1 + exp(-bs.eta.train)) ^ (-1)
bs.pred.test <- (1 + exp(-bs.eta.test)) ^ (-1)

bs.auc.train <- auroc(bs.pred.train, ifelse(y_train == "B", FALSE, TRUE))
bs.auc.test <- auroc(bs.pred.test, ifelse(y_test == "B", FALSE, TRUE))
```

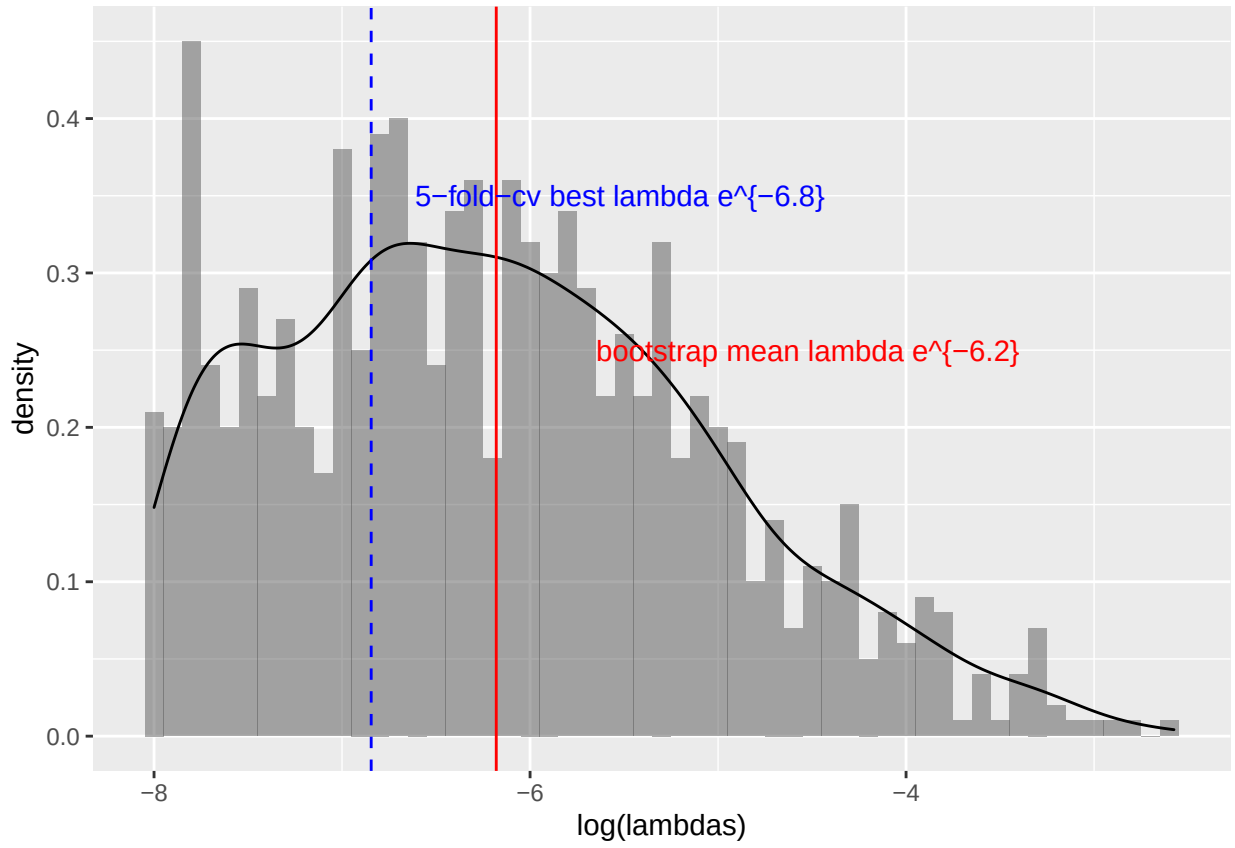
By implementing bagging, we get a AUC = 0.99036 on training set and a AUC = 0.97228 on test set. Which means the bootstrap smoothed model outperform the previous model (train AUC = 0.99009, test AUC = 0.97159).

4. Write a summary of your findings.

Distribution of lambdas from bootstrap replicates

From the histogram and density plot below, we see that the distribution of bootstrap replicates of λ shows a multimodal pattern. Since each bootstrap replicate is a random sample with replacement from the observed data, we are excluding different observations, and the multimodal pattern suggests that this variability in exclusion of observations may have impact in parameter selection.

```
library(ggplot2)
bs.best.lambdas <- data.frame(lambdas = bs.res$best.lambdas)
ggplot(data = bs.best.lambdas, aes(x = log(lambdas))) +
    geom_histogram(aes(y = ..density..), binwidth = 0.1, alpha = 0.5) +
    geom_density() +
    geom_vline(xintercept = log(optimal.fit$lambda), lty = 2, color = 'blue') +
    geom_vline(xintercept = mean(log(bs.best.lambdas$lambdas)), lty = 1, color = 'red') +
    annotate("text", x = -5.5, y = 0.35, label = "5-fold-cv best lambda e^{-6.8} ", color = "blue") +
    annotate("text", x = -4.5, y = 0.25, label = "bootstrap mean lambda e^{-6.2} ", color = "red")
```



Variance reduction of estimated coefficients

```
logisticCoef <- rep(0, 10 + 1)
names(logisticCoef) <- names(bs.res$smooth.coef)
for (pr in names(as.matrix(coef(logit.fit))[,])) {
    logisticCoef[pr] = as.matrix(coef(logit.fit))[pr,]
}

bs.inference <- data.frame(
    predictors = names(bs.res$smooth.coef),
    origin.coef = logisticCoef,
    smooth.coef = bs.res$smooth.coef,
    chosen.prob = scales::percent(bs.res$chosen.prob),
    sd = bs.res$coef.std,
    smooth.sd = bs.res$coef.smooth.sd,
    ci.L = bs.res$smooth.coef - qnorm(0.975) * bs.res$coef.std,
    ci.R = bs.res$smooth.coef + qnorm(0.975) * bs.res$coef.std,
    smoothed.ci.L = bs.res$smooth.coef - qnorm(0.975) * bs.res$coef.smooth.sd,
    smoothed.ci.R = bs.res$smooth.coef + qnorm(0.975) * bs.res$coef.smooth.sd
)

bs.inference %>% knitr::kable(caption = "Bootstrap smoothing result", digits = 3) %>%
    kableExtra::kable_styling(latex_options = c("hold_position", "scale_down"),
                              font_size = 8,full_width = F) %>%
    kableExtra::add_footnote(
```

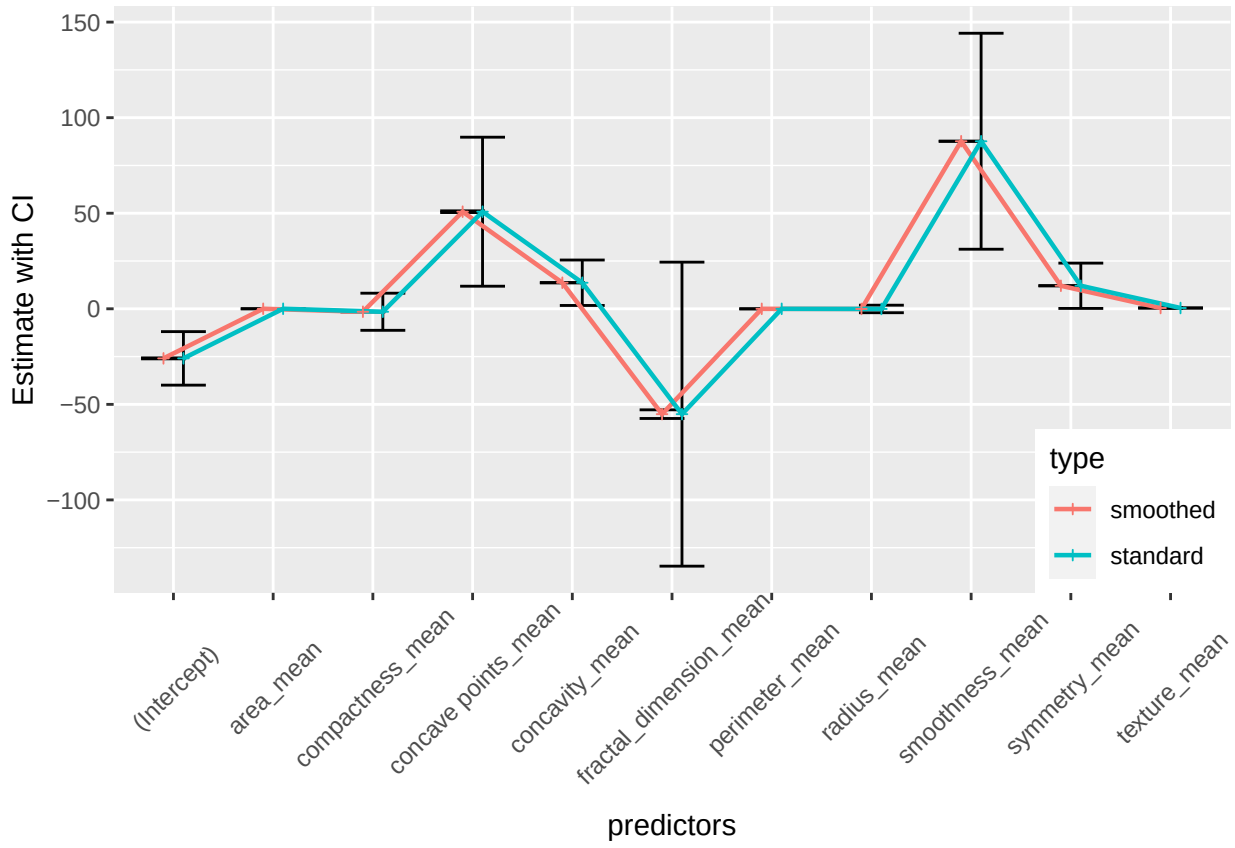
```
        c("Bootstap time=1000",
          "origin.coef: estimation from selected predictors logistic regression",
          "smoothed.coef: estimation from Smoothed Bootstrap",
          "chosen.prob: chosen probability from Smoothed Bootstrap",
          "sd: standard deviation of estimate",
          "smooth.sd: nonparamatric delta-method estimate of standard deviation",
          "ci.L, ci.R:  lower and upper CI of estimate derived from standard deviation",
          "ci.smoothed.L, ci.smoothed.R: lower and upper CI from nonparamatric delta-method estimate"),
        notation = "alphabet")
```

Table 2: Bootstrap smoothing result

	predictors	origin.coef	smooth.coef	chosen.prob	sd	smooth.sd	ci.L	ci.R	smoothed.ci.L	smoothed.ci.R
(Intercept)	(Intercept)	-26.975	-25.942	100.000%	7.143	0.127	-39.942	-11.943	-26.191	-25.693
radius_mean	radius_mean	0.000	-0.086	30.200%	1.004	0.002	-2.053	1.882	-0.089	-0.082
texture_mean	texture_mean	0.424	0.419	99.900%	0.022	0.000	0.376	0.462	0.419	0.419
perimeter_mean	perimeter_mean	0.000	-0.020	11.600%	0.091	0.000	-0.199	0.159	-0.021	-0.019
area_mean	area_mean	0.013	0.015	81.000%	0.012	0.000	-0.008	0.038	0.015	0.015
smoothness_mean	smoothness_mean	99.275	87.670	91.900%	28.828	0.016	31.168	144.171	87.639	87.700
compactness_mean	compactness_mean	0.000	-1.550	46.900%	4.953	0.032	-11.257	8.157	-1.614	-1.487
concavity_mean	concavity_mean	17.409	13.636	89.100%	6.065	0.055	1.749	25.524	13.528	13.744
concave points_mean	concave points_mean	40.173	50.801	97.500%	19.890	0.229	11.818	89.785	50.352	51.251
symmetry_mean	symmetry_mean	14.754	12.047	80.300%	6.046	0.040	0.197	23.898	11.968	12.127
fractal_dimension_mean	fractal_dimension_mean	-90.560	-55.122	68.200%	40.580	1.153	-134.658	24.414	-57.382	-52.862

[a] Bootstap time=1000
[b] origin.coef: estimation from selected predictors logistic regression
[c] smoothed.coef: estimation from Smoothed Bootstrap
[d] chosen.prob: chosen probability from Smoothed Bootstrap
[e] sd: standard deviation of estimate
[f] smooth.sd: nonparamatric delta-method estimate of standard deviation
[g] ci.L, ci.R: lower and upper CI of estimate derived from standard deviation
[h] ci.smoothed.L, ci.smoothed.R: lower and upper CI from nonparamatric delta-method estimate

```
bs.inference %>%
    select(c(predictors, smooth.coef, ci.L, ci.R, smoothed.ci.L, smoothed.ci.R)) %>%
    rename(
        CI_L_standard = ci.L,
        CI_R_standard = ci.R,
        CI_L_smoothed = smoothed.ci.L,
        CI_R_smoothed = smoothed.ci.R,
    ) %>%
    pivot_longer(cols = CI_L_standard:CI_R_smoothed,
                 names_to = c("drop", "LR","type"),
                 names_sep = "_",
                 values_to = "bound" ) %>%
    select(c("predictors", "smooth.coef", "LR", "type", "bound")) %>%
    pivot_wider(names_from = LR, values_from = bound) %>%
    ggplot(aes(x = predictors, y = smooth.coef, color = type, group = type)) +
    geom_errorbar(aes(ymin = L, ymax = R), color = "black", position = position_dodge(width = 0.4)) +
    geom_line(size = .8, position = position_dodge(width = 0.4)) +
    geom_point(size = 1, shape = 3, position = position_dodge(width = 0.4)) +
    ylab("Estimate with CI") +
    labs(fill = "") +
    theme(legend.justification = c(1, 0), legend.position = c(1, 0), axis.text.x = element_text(angle =
```



From the table and error bar plot above, we found that the bootstrap smoothed interval is narrower than the standard interval, corresponding to a reduced variance.

Conclusion

In summary, by implementing the bootstrap smoothing, which takes the model selection process into account, we improved the model performance in prediction. And the bootstrap smoothing also gives us a nonparametric delta-method estimate of standard deviation with a narrower 95% confidence interval.

Bibilography

1. Rizzo, M.L. (2019). Statistical Computing with R (2nd ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9780429192760
2. Bradley Efron (2014) Estimation and Accuracy After Model Selection, Journal of the American Statistical Association, 109:507, 991-1007, DOI: 10.1080/01621459.2013.823775