# Homework on re-sampling methods

P8160 Advanced Statistical Computing

**In this homework, please using parallel computing codes for your implementations.**

## Problem 1: a randomized trial on an eye treatment

An ophthalmologist deisgned 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

**1**

$$E(Y \mid \text{trt} = \text{red}) - E(Y \mid \text{trt} = \text{blue}).$$

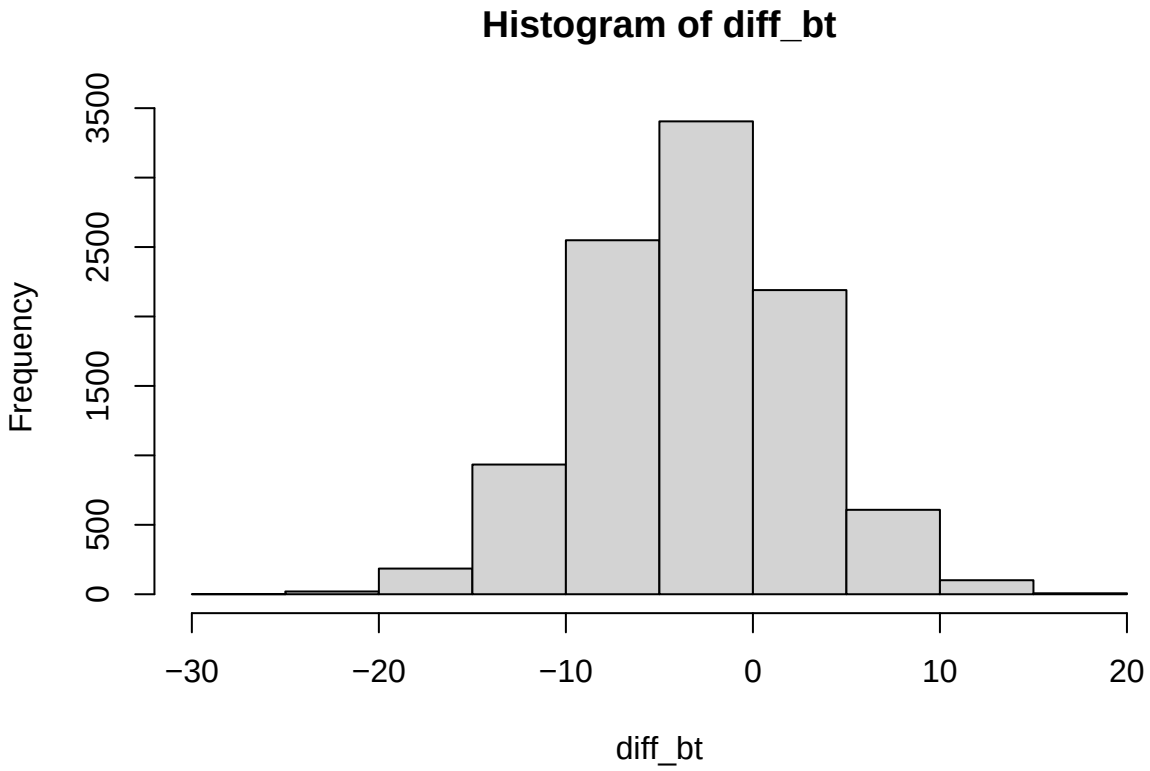
```
mean(red-blue)
```

```
## [1] -3.066667
```

**2**

```r
bootr =
  function(data,return_sample = F){
    bt = sample(data,length(data),replace = T)
    bt_mean = mean(bt)
    bt_bias = mean(data) - mean(bt)
    bt_sd = sd(bt)

    if(!return_sample) bt = NA

    return(list(
      sample = bt,
      mean = bt_mean,
      sd = bt_sd,
      bias = bt_bias
    ))
  }
```

```r
#1e+4 bootstrap

cl = makePSOCKcluster(5)
registerDoParallel(cl)

B = 1e+4

red_bt = foreach(i = 1:B,
                 .combine = rbind,
                 .inorder = F) %do% bootr(red)$mean

blue_bt = foreach(i = 1:B,
                  .combine = rbind,
                  .inorder = F) %do% bootr(blue)$mean

diff_bt = red_bt - blue_bt

hist(diff_bt)
```




```
diff_sd_bt = sqrt(var(diff_bt))

stopCluster(cl)
```

```
alpha = 0.05
lower = max(diff_bt[percent_rank(diff_bt)<alpha/2])
upper = min(diff_bt[percent_rank(diff_bt)>(1-alpha/2)])
lower_aj = 2*mean(red-blue) -upper
upper_aj = 2*mean(red-blue) - lower
```

The unadjusted CI is rc(lower,upper)‘.The adjusted 95% CI is-14.0333333, 8.3333333.Given the bootstrap interval, we see that 0 is within the CI, we cannot reject the Null hypothesis that the interventions have no difference.

## Problem 2 (Continue from Homework 3, the breast cancer sutdy):

The data *breast-cancer.csv* have 569 row and 33 columns. The first column **ID** lables individual breast tissue images; The second column **Diagnonsis** indentifies 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 30 columns correspond to mean, standard deviation and the largest values (points on the tails) of the distributions of the following 10 features computed for the cellnuclei;

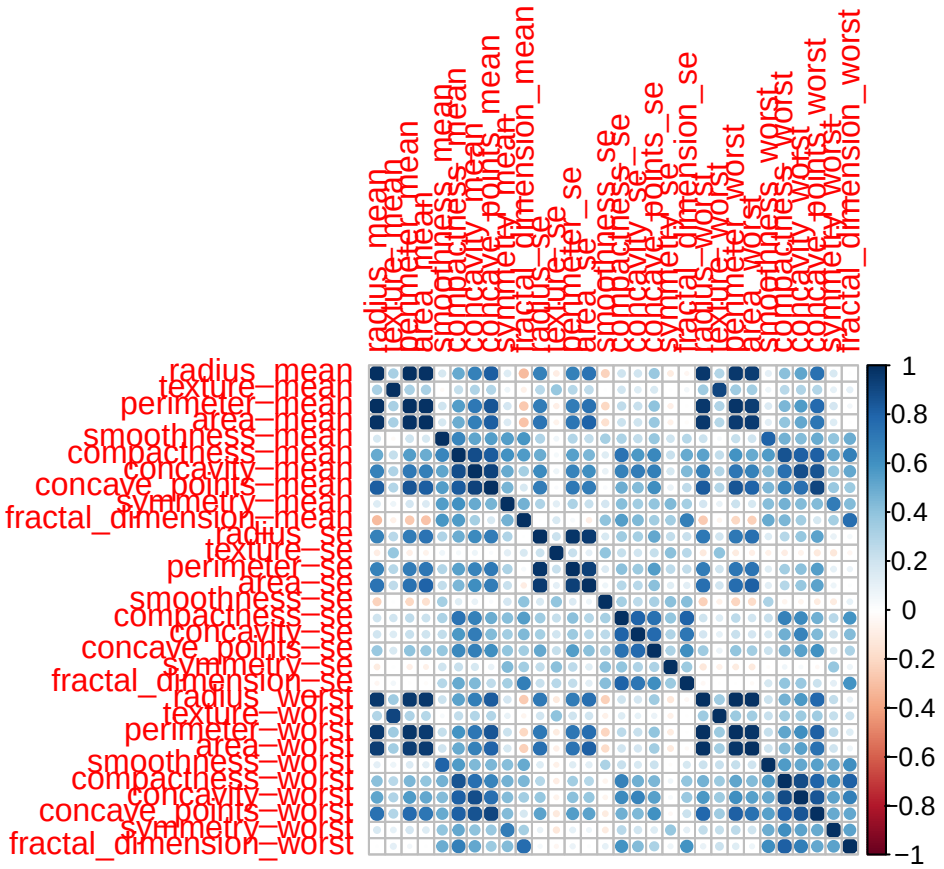
- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)

- 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)

In homework 3, you have developed a path-wise coordinate-wise optimization algorithm for logistic LASSO regressions. Using the algorithm you developed there, complete the following tasks

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 $\lambda$ as the "optimal" 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 bootstraping smoothing idea to re-evaluate the probabilities of malignant. How well the new predictors classify the images?

```
breast =
  read_csv("./breast-cancer.csv") %>%
  janitor::clean_names() %>%
  select(-id,-x33) %>%
  mutate(diagnosis = forcats::fct_relevel(diagnosis,"M")
         )

Y = breast$diagnosis

X = model.matrix(diagnosis~.,breast)[,-1]

set.seed(123123)

test_index = sample(1:length(Y),0.2*length(Y))

corrplot::corrplot(cor(X))
```




```
high_cor = caret::findCorrelation(cor(X),0.8)

X = X[,-high_cor]

Y_ts = Y[test_index]

X_ts = X[test_index,]

Y_tr=Y[-test_index]

X_tr=X[-test_index,]
```

Keeping only variables below correlation of 0.8.

Using predictive error rate as loss function, we tune the lasso logistic as:

```
set.seed(123123)
source("lasso_function.R")

a = cv_Qlasso(Y_tr,X_tr,bfun,intercept = T,return_scale = T)
```

```
## `summarise()` has grouped output by 'lambda'. You can override using the `.groups` argument.
```

```
a
```

```
## $coefficient
##  [1]  0.25176192 -4.06855472  1.05077474 -1.60741625 -1.22448796 -0.59373531
##  [7] -0.93146126 -0.06043354  0.00000000 -2.56138710  0.48805744  0.04240966
## [13] -0.25284346 -0.61150061  0.26188098
##
## $objective
## [1] 0.04832776
##
## $lambda
## [1] 1.05249
##
## $cvtable
## # A tibble: 22 x 5
## # Groups:   lambda [22]
##    objective  lambda     sd coef              `1se`
##        <dbl>   <dbl>  <dbl> <list>            <lgl>
##  1    0.0483 1.05    0.0265 <tibble [15 x 2]> TRUE
##  2    0.0483 2.17    0.0265 <tibble [15 x 2]> TRUE
##  3    0.0505 0.511   0.0253 <tibble [15 x 2]> TRUE
##  4    0.0527 4.46    0.0263 <tibble [15 x 2]> TRUE
##  5    0.0571 0.249   0.0239 <tibble [15 x 2]> TRUE
##  6    0.0593 0.0587  0.0277 <tibble [15 x 2]> TRUE
##  7    0.0593 0.121   0.0277 <tibble [15 x 2]> TRUE
##  8    0.0593 9.17    0.0336 <tibble [15 x 2]> TRUE
##  9    0.0615 0       0.0277 <tibble [15 x 2]> TRUE
## 10    0.0615 0.00674 0.0277 <tibble [15 x 2]> TRUE
## # ... with 12 more rows
```

```
drop_var = a$coefficient==0

names(breast)[-high_cor][-drop_var]
```

```
##  [1] "concave_points_mean"     "symmetry_mean"
##  [3] "fractal_dimension_mean"  "radius_se"
##  [5] "area_se"                 "compactness_se"
##  [7] "concavity_se"            "concave_points_se"
##  [9] "symmetry_se"             "radius_worst"
## [11] "area_worst"              "concave_points_worst"
## [13] "symmetry_worst"          "fractal_dimension_worst"
```

```
X_new = as_tibble(X_tr[,-drop_var])

predict_model = glm(Y_tr~.,data=X_new,family = binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
X_ts_new = X_ts[,-drop_var]

pd = ifelse(predict(predict_model,newdata = as_tibble(X_ts_new),type = "response")>=0.5,"B","M") %>% fo

Y_ts
```

```
##   [1] B B B M B B M B M B B M B M B B B B M M M B B B M M B B B M B B M M B
##  [38] B B B B M B B M M M B B B B B B M M B B B B M B M M B B B B B B B M B B
##  [75] M B B B M M B M B B B M B B M M B M B M M B B M M B M B B B B B M B B B B
## [112] M B
## Levels: M B
```

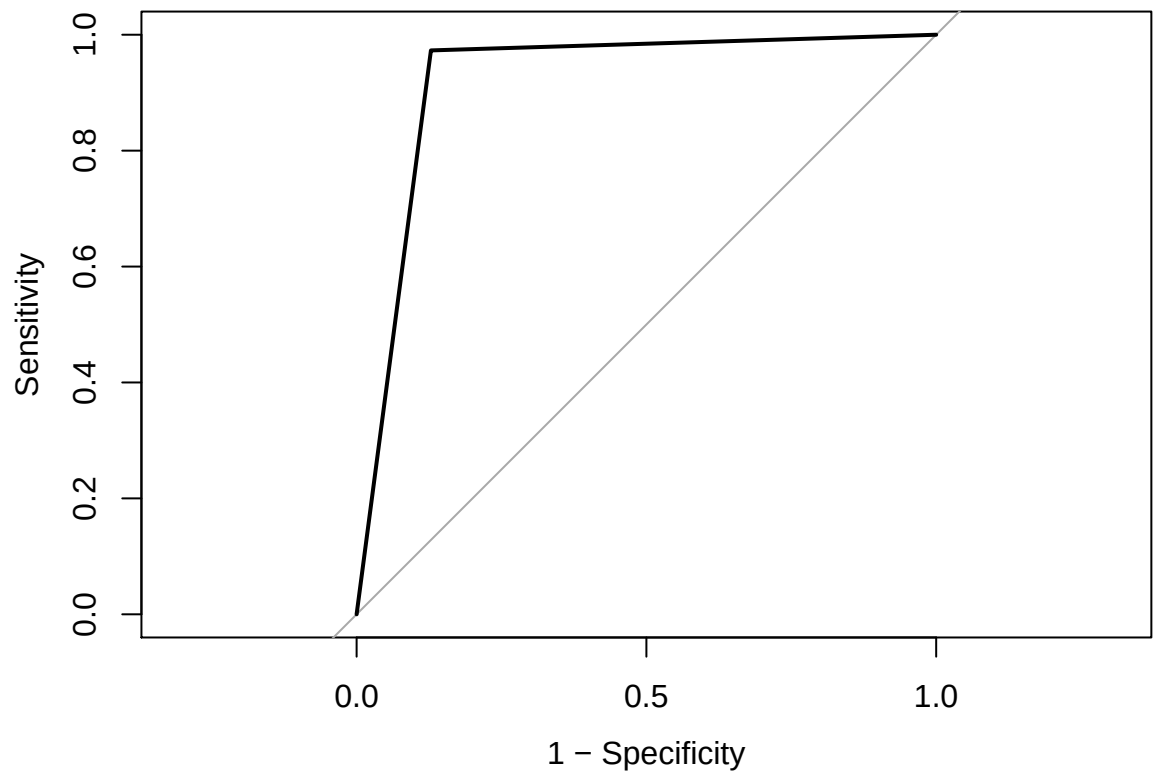
```
mean(Y_ts %>% as.numeric() -1 !=pd)
```

```
## [1] 0.0619469
```

```
plot(roc(Y_ts %>% as.numeric() -1,pd),legacy.axes = T)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
test_data =
  cbind(Y_tr,X_new) %>%
  as_tibble() %>%
  rename(y=Y_tr)

cl = makePSOCKcluster(5)
registerDoParallel(cl)
```

```
bt_err = foreach(i=1:B,
                 .combine = rbind) %dopar% {
                   bt_data= dplyr::sample_n(test_data,nrow(test_data),replace = T)
                   bt_md = glm(y~.,data = bt_data,family = binomial)
                   bt_pd = rep("M",len = length(Y_ts))
                   bt_pd[predict(bt_md,dplyr::as_tibble(X_ts_new),type = "response")>0.5]="B"
                   mean(Y_ts!=bt_pd)
                 }

stopCluster(cl)

hist(bt_err)

abline(v = mean(bt_err),add = T,col=2)
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): "add" is
## not a graphical parameter
```