Homework 3 on Bootstrap and resampling methods

Xinyi Lin xl2836

Due: 04/15/2019, Wednesday, by 1pm

Please implement paralle computating into all your bootstrap algorithm and R codes.

Problem 1

Example 1: a randomized trial on eye treatment (See Lecture 7.pdf, page 2)

Answer:

Question 1: How would you analyze the data to investigate whether the expected accuracies between the two treatments are different.

As this dataset a mixture of paired comparison and two-sample data, I used following statistics to test hypotheses.

$$H_0 : \mu_d = 0, H_1 : \mu_d \neq 0$$

$$\bar{d} = \frac{1}{30}[\sum_{i=1}^{20} d_i + \sum_{i=1}^{10}(x_i - y_i)]$$

$$var(\bar{d}) = \frac{20}{30^2} var(d_i) + \frac{10}{30^2} var(x_i) + \frac{10}{30^2} var(y_i)$$

$$d_{stat} = \frac{\bar{d}}{\sqrt{var(\bar{d})}}$$

```
# 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)
two_sample = acui[which(acui$str==1),]
x = two_sample$blue
y = two_sample$red
paired_sample = acui[which(acui$str==0),]
d = paired_sample$blue-paired_sample$red
```

```
teststat <- function(d, x, y) {
  d_bar = (sum(d) + sum(x-y))/30
  d_var = (20*var(d) + 10*var(x) + 10*var(y))/900
  return(list(dhat = d_bar, stat = d_bar/sqrt(d_var)))
  }
boottest1 <- function(d, x, y, nboot=200) {
  combmean_xy <- mean(c(x,y))
  # The mean of x and y
  teststatvec <- rep(NA, nboot)
  estimat_dhat <- rep(NA, nboot)
  adjx <- x - mean(x) + combmean_xy
  # The adjusted X's will have mean=combmean
  adjy <- y - mean(y) + combmean_xy
```

```
  # The adjusted X's will have mean=combmean
  adjd <- d - mean(d)
  for(b in 1:nboot) {
    res <- teststat(sample(adjd, replace = T),
                                sample(adjx, replace = T),
                                sample(adjy, replace = T))
    teststatvec[b] = res$stat
    estimat_dhat[b] = res$dhat
  }
  return(list(bootpval = sum(abs(teststat(d,x,y)$stat) < abs(teststatvec))/nboot,  teststatvec = teststa
}
```

Without using paralle computing

```
set.seed(1)
system.time({
res1 = boottest1(d, x, y)
})
```

```
##    user  system elapsed
##   0.103   0.011   0.117
```

Using paralle computing

```
set.seed(1)
library(parallel)
nCores<-detectCores() # detect numbers of available cores
cl = makeCluster(nCores)
system.time({
res1 = boottest1(d, x, y)
})
```

```
##    user  system elapsed
##   0.032   0.002   0.035
```

```
stopCluster(cl)
res1$bootpval
```

```
## [1] 0.545
```

Accoridng to results, we can know that 1) Using paralle computing helps to decrease running time; 2) As p-value equals to 0.545 which is larger than 0.05, we fail to reject null hypothesis and conclude that with 95% confidence, two treatments do not have different effects.

Question 2: Use bootstrap to construct confidence interval of the treatment effect. What is your conclusion?

```
d_bar = (sum(d) + sum(x-y))/30
d_bar
```

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

```
CI_lower = 2*d_bar-quantile(res1$estimat_dhat,0.975)
CI_upper = 2*d_bar-quantile(res1$estimat_dhat,0.025)
CI=c(CI_lower,CI_upper)
CI
```

```
##    97.5%     2.5%
## -3.11000 14.36833
```

Problem 2

Answer:

Example 2 Number of modes of a density (See Lecture 7.pdf, page 4)

$$H_0 : n_{mode} = 1, H_1 : n_{mode} > 1$$

```
library(MASS)
data(galaxies)
```

```
cal_modes = function(data, bw=3.05){
  den <- density(data/1000, bw=bw)
  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
  return(nmodes)
}
boottest2 = function(data, nboot=200){
  teststatvec <- rep(NA, nboot)
  for (i in 1:nboot) {
    #ndata = sample(data, 1)
    #boot_data = rnorm(82, mean = ndata, sd = 3.05)
    ndata=sample(data,replace = T)
    boot_data=apply(as.data.frame(ndata),1,FUN=function(x) rnorm(1,mean=x,sd=3.05))
    if (cal_modes(boot_data) > 1)
      teststatvec[i] = 1
    else
      teststatvec[i] = 0
    }
    return(list(bootpval = sum(teststatvec)/nboot,  teststatvec = teststatvec))
  }
```

Using paralle computing

```
set.seed(1)
#library(parallel)
#nCores<-detectCores() # detect numbers of available cores
cl = makeCluster(nCores)
res2 = boottest2(galaxies)
stopCluster(cl)
res2$bootpval
```

```
## [1] 0.49
```

According to results, we get p-value equalst to 0.49 which is larger than 0.05, thus we fail to reject null hypothesis and conclude that with 95% confidence, the number of modes equals to 1.

Problem 3

Recall in the lecture of EM algorithm, we studied an ABO blood type data, where we have $N_{obs} = (N_A, N_B, N_{AB}, N_O) = (26, 27, 42, 7)$, and designed EM algorithm to estimate the allele frequencies, P_A, P_B and P_O.

Please design a bootstrap algorithm to estimate the variances of the estimated $\hat{P}_A$, $\hat{P}_B$ and $\hat{P}_O$. Implement your algorithms in R, and present your results..

Answer:

EM algorithm to get estimators of $\hat{P}_A, \hat{P}_B, \hat{P}_O$.

```
# E-step evaluating conditional means E(N_gene | N_obs , p)
getEN <- function(N, p){
  aa = N$a * p$a^2/(p$a^2 + 2*p$a*p$o)
  ao = N$a * 2*p$a*p$o/(p$a^2 + 2*p$a*p$o)
  bb = N$b * p$b^2/(p$b^2 + 2*p$b*p$o)
  bo = N$b * 2*p$b*p$o/(p$b^2 + 2*p$b*p$o)
  ab = N$ab
  oo = N$o
  EN = list(aa=aa, ao=ao, bb=bb, bo=bo, ab=ab, oo=oo)
  return(EN)
}
# M-step - updating the parameters
getp <- function(EN) {
  n <- EN$aa+EN$ao+EN$bb+EN$bo+EN$ab+EN$oo
  a = (2*EN$aa+EN$ao+EN$ab)/(2*n)
  b = (2*EN$bb+EN$bo+EN$ab)/(2*n)
  o = (2*EN$oo+EN$ao+EN$bo)/(2*n)
  p = list(a=a,b=b,o=o)
  return(p)
}
EMmix <- function(N, startp, nreps=20) {
  i <- 0
  EN <- getEN(N,startp)
  newp <- startp
  #res <- c(0, t(as.matrix(newp)))
  res <- newp
  while(i < nreps) {
    i <- i + 1
    newp <- getp(EN)
    EN <- getEN(N,newp)
    #res <- rbind(res, c(i, t(as.matrix(newp))))
    res <- rbind(res, newp)
  }
  return(res[nreps,])
}
```

Test EM algorithm

```
N = list(a=26, b=27, ab=42, o=7)
p = list(a=0.3, b=0.3, o=0.4)
res = EMmix(N,p)
#res = rbind(res,res)
res
```

```
## $a
## [1] 0.3972387
##
```

```
## $b
## [1] 0.40526
##
## $o
## [1] 0.1975014
```

Next, we combind bootstrap and EM al to estimate variances of the estimated $\hat{P_A}, \hat{P_B}, \hat{P_O}$.

```
boottest3 = function(data, startp, nboot=200){
  res_lis = startp
  for (i in 1:nboot) {
    boot_data = sample(data, replace = T)
    a = length(boot_data[boot_data == "a"])
    b = length(boot_data[boot_data == "b"])
    ab = length(boot_data[boot_data == "ab"])
    o = length(boot_data[boot_data == "o"])
    N = list(a=a, b=b, ab=ab, o=o)
    res = EMmix(N, startp = startp)
    res_lis = rbind(res_lis,res)
  }
  var_a = var(as.numeric(res_lis[,1]))
  var_b = var(as.numeric(res_lis[,2]))
  var_o = var(as.numeric(res_lis[,3]))
  return(list(var_a = var_a, var_b = var_b, var_o = var_o))
}
```

Data

```
ABO_data = c(rep("a",26), rep("b",27), rep("ab",42), rep("o",7))
```

Using paralle computing, variances of estimators are shown below.

```
set.seed(1)
#library(parallel)
#nCores<-detectCores() # detect numbers of available cores
cl = makeCluster(nCores)
res3 = boottest3(ABO_data,p)
stopCluster(cl)
res3
```

```
## $var_a
## [1] 0.001230754
##
## $var_b
## [1] 0.001556272
##
## $var_o
## [1] 0.002140086
```