Problem Set 2, PS787

Anthony Cozart

Fri Oct 13, 2017

Question 1

Answer to Question 1a

This pattern in the data, of Hispanics being more likely to have their IDs accepted, may misrepresent our understanding of discrimination if the missing data is correlated with IDs being requested and accepted (i.e., not missing at random). It's easy to imagine a third factor (like appearance of clothes) that is correlated with whether an ID is requested (and ultimately accepted) and the treatment (race). Audit studies tell us that appearance affects whether individuals receive discretionary accommodations, and if you think appearance is correlated with race (it likely is in some professional settings, think Wall St), and that those who dress a certain way will not be asked for IDs, then our estimates are misrepresentative.

Answer to Question 1b

```
setwd("~/Desktop/Michigan/PoliSci 787/Problem Set 2")
library(foreign)
library(MASS)

data = read.csv("q1_data.csv")
dim(data)
```

```
## [1] 217   3
```

```
names(data)
```

```
## [1] "id"       "hispanic" "accepted"
```

```
# create indicator for observed
data$dobserved = !is.na(data$accepted)
summary(data$accepted)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.0000  0.0000  1.0000  0.6861  1.0000  1.0000      45
```

```
Y = data$accepted
Z = data$hispanic
R = data$dobserved

# reproduce table from question prompt
table(Z)              # by treatment
```

```
## Z
##   0   1
## 106 111
```

```
table(R)              # by reported
```

```
## R
## FALSE  TRUE
##    45   172
```

```
table(Z,R)
```

```
##    R
## Z   FALSE TRUE
##   0    28   78
##   1    17   94
```

```
summary(Y[R == 1])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   0.686   1.000   1.000
```

```
summary(Y[R == 0])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##      NA      NA      NA     NaN      NA      NA      45
```

```
# calculate observed ATEs
ave_y1_obs = mean(Y[Z==1 & R==1])
ave_y0_obs = mean(Y[Z==0 & R==1])

summary(Y[Z==1 & R==1]) # mean = likelihood of being accepted if hispanic
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.7234  1.0000  1.0000
```

```
summary(Y[Z==0 & R==1]) # mean = likelihood of being accepted if white
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   0.641   1.000   1.000
```

```
t.test(Y[Z==0 & R==1], Y[Z==1 & R==1])
```

```
##
##  Welch Two Sample t-test
##
## data:  Y[Z == 0 & R == 1] and Y[Z == 1 & R == 1]
## t = -1.149, df = 159.38, p-value = 0.2523
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.22397064  0.05921341
## sample estimates:
## mean of x mean of y
## 0.6410256 0.7234043
```

```
# we fail to reject the null hypothesis that the likelihoods of being accepted are the same.

ate_obs =  ave_y1_obs - ave_y0_obs

# calculate probabilities of reporting by group
summary(R[Z==1])
```

```
##    Mode   FALSE    TRUE
## logical      17      94
```

```
p1 = mean(R[Z==1])

summary(R[Z==0])
```

```
##    Mode   FALSE    TRUE
## logical      28      78
```

```
p0 = mean(R[Z==0])

# set bounds
gamma_H = 1 # all accepted
gamma_L = 0 # all rejected

e0_max = ave_y0_obs * p0 +  gamma_H * (1-p0)
e0_min = ave_y0_obs * p0 +  gamma_L * (1-p0)

e1_max = ave_y1_obs * p1 +  gamma_H * (1-p1)
e1_min = ave_y1_obs * p1 +  gamma_L * (1-p1)

ate_min = e1_min - e0_max
ate_max = e1_max - e0_min
cat("Bounds on ATE: the identifcation region for ATE is [", ate_min, ",", ate_max,"] \n")
```

```
## Bounds on ATE: the identifcation region for ATE is [ -0.1232364 , 0.2940677 ]
```

```
cat("ATE on observed sample only:", ate_obs, "\n")
```

```
## ATE on observed sample only: 0.08237861
```

Answer to Question 1c

```
set.seed(123)
B <- 1000
results <- numeric(B)

# does it make sense to make two draws?
# One each for the likelihood of acceptance E(Y_1|R=0) and E(Y_0|R=0)? Yes. Both could be between 0,1.
for(i in 1:B){
  boot.gamma0<-runif(1)
  boot.gamma1<-runif(1)
  results[i]<- (ave_y1_obs*p1 + boot.gamma1*(1-p1))-(ave_y0_obs*p0 + boot.gamma0*(1-p0))
}
length(results)
```

```
## [1] 1000
```

```
c(sort(results)[25], sort(results)[975])
```

```
## [1] -0.08519119  0.25037808
```

```
# But above is a Monte-Carlo. Instead draw samples from data and calculate the bounds
ate_minb = numeric(B)
ate_maxb = numeric(B)

for(i in 1:B) {
  datab = data[sample(1:nrow(data),replace=TRUE),]

  Yb = datab$accepted
  Zb = datab$hispanic
  Rb = datab$dobserved
  ave_y1_obs = mean(Yb[Zb==1 & Rb==1])
  ave_y0_obs = mean(Yb[Zb==0 & Rb==1])
  p1 = mean(Rb[Zb==1])
  p0 = mean(Rb[Zb==0])

  e0_max = ave_y0_obs * p0 +  gamma_H * (1-p0)
  e0_min = ave_y0_obs * p0 +  gamma_L * (1-p0)

  e1_max = ave_y1_obs * p1 +  gamma_H * (1-p1)
  e1_min = ave_y1_obs * p1 +  gamma_L * (1-p1)

  ate_minb[i] = e1_min - e0_max
  ate_maxb[i] = e1_max - e0_min
}

mean(ate_minb)
```

```
## [1] -0.1214525
```

```
mean(ate_maxb)
```

```
## [1] 0.2947351
```

```
lower.bound = quantile(ate_minb,p=c(0.025))
upper.bound = quantile(ate_maxb,p=c(0.975))
cat("Bounds on ATE: the identifcation region for ATE is [", lower.bound, ",", upper.bound,"] \n")
```

```
## Bounds on ATE: the identifcation region for ATE is [ -0.2368117 , 0.4227215 ]
```

Question 2

```
q2.data = read.csv("question2.csv")
B<-1000
results<-numeric(length=B)

for (i in 1:B) {
  bootsample = q2.data[sample(1:nrow(q2.data),replace=TRUE),]
  bx1 = bootsample$x1
  bx2 = bootsample$x2
  boot.diff = mean(bx1) - mean(bx2)
  results[i] = boot.diff

# Check results
mean(q2.data$x1) - mean(q2.data$x2) # Full sample estimator
mean(results)  # Bootstrap estimator
}
```

Answer to Question 2a

```
c(sort(results)[25], sort(results)[975])
```

```
## [1] -1.1069476 -0.9341379
```

Answer to Question 2b

```
sqrt(var(results))
```

```
## [1] 0.04404263
```

```
# Check: sqrt(var(x1)/length(x1)+var(x2)/length(x2))
```

Answer to Question 2c

The difference in means of x1 and x2 is significantly different from zero. The 95% confidence interval does not contain zero (it's [-1.1034694, -0.9211613]).

Question 3

```
q34.data = read.csv("questions3_4.csv")

# OLS function unchanged from Rocio's code
ols<-function(y,x) {
  n=nrow(x)
  k=ncol(x)
  beta.hats<-ginv(t(x)%*%x)%*%t(x)%*%y
  residu<-y-x%*%beta.hats
  sigma2<-(t(residu)%*%residu)/(n-k)
  varcovar<-sigma2[1,1]*(solve(t(x)%*%x))
  std.dev<-sqrt(diag(varcovar))
  list(betas=beta.hats,std.dev=std.dev)
}

# The hand-made function above takes a matrix. Create one here.
ols.mat<-as.matrix(q34.data)
y<-ols.mat[,1]
x<-cbind(1,ols.mat[,2:ncol(ols.mat)])  # column of 1s is for the constant
data<-cbind(y,x)

# Run OLS estimation (will use this later, when calculating bootstrap bias)
output.ols<-ols(y,x)

# Non-parametric bootstrap
B<-1000
bboot<-matrix(nrow=B,ncol=ncol(data)-1)
dim(bboot)
```

```
## [1] 1000    4
```

```r
k = 1:nrow(data)
for(i in 1:B){
  indxboot <- sample(k,replace=TRUE) # Sample the indices
  datab <-data[indxboot,] #draw row using drawn index place
  yb <-datab[,1]
  xb <-datab[,2:ncol(data)]
  bboot[i,]<-t(ols(yb,xb)$betas)
}
```

Answer to Question 3a

```r
# Boostrapped estimators
apply(bboot,2,mean)
```

```
## [1] 10.309699  2.000246  3.986737  2.986176
```

```r
# Bootstrapped confidence intervals (show basic bootstrap)
apply(bboot,2,function(x) quantile(x,p=c(0.025,0.975)))
```

```
##            [,1]     [,2]     [,3]     [,4]
## 2.5%   9.889701 1.980839 3.957939 2.963438
## 97.5% 10.709143 2.021658 4.016310 3.008874
```

Answer to Question 3b

```r
# create matrix to hold bias for each of B=1000 samples
diff<-matrix(nrow=B,ncol=4)
# calculate bias for each coefficient
for(i in 1:4){
  diff[,i]<-bboot[,i]-output.ols$betas[i]
}
bias<-apply(diff,2,mean)
bias
```

```
## [1] -9.177346e-03 -9.173407e-05  2.055102e-04  5.413126e-04
```

Answer to Question 3c

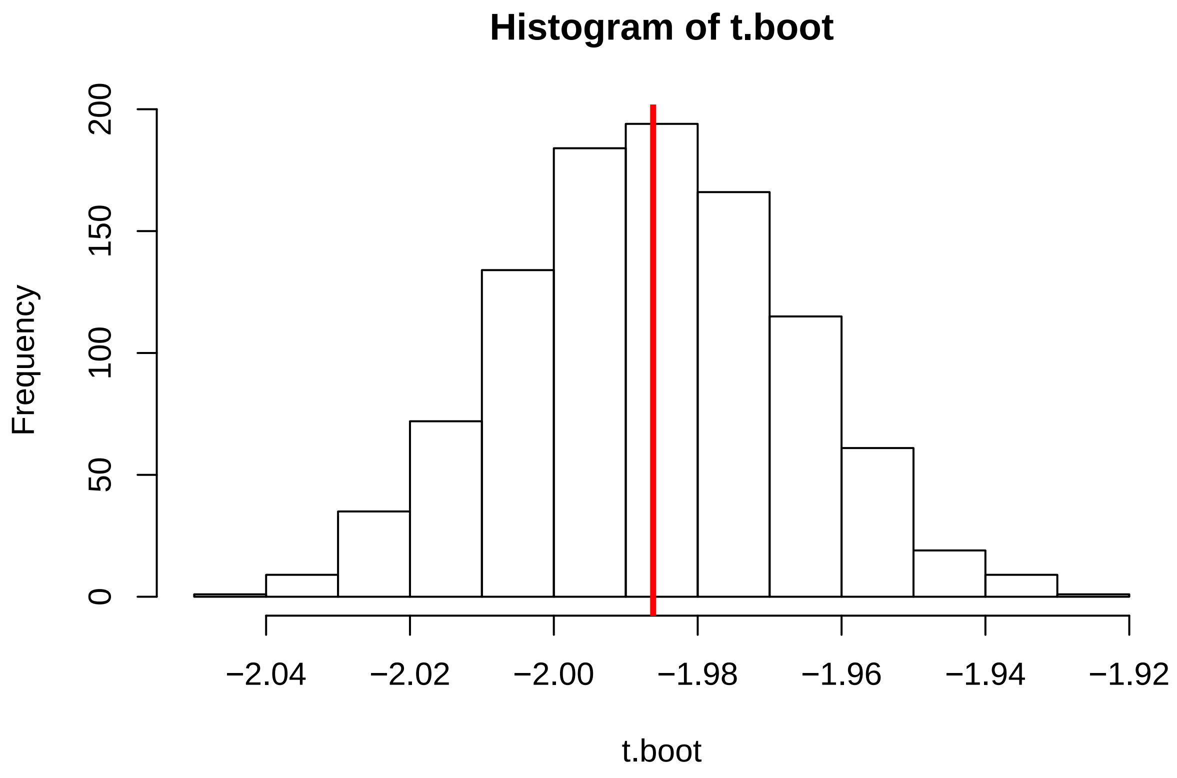
```r
# confidence interval
q3c = c(sort(bboot[,2])[25], sort(bboot[,2])[975])
q3c
```

```
## [1] 1.980778 2.021654
```

```r
cat("The 95% confidence interval of B_1 is [",q3c[1],",",q3c[2],"],
    which does not contain the boostrapped estimate for B_2,
    ",output.ols$betas[3,1]," \n")
```

```
## The 95% confidence interval of B_1 is [ 1.980778 , 2.021654 ],
##     which does not contain the boostrapped estimate for B_2,
##      3.986531
```

```r
# plot
t.boot = bboot[,2] - bboot[,3]
t.sample = output.ols$betas[2,1] - output.ols$betas[3,1]
hist(t.boot)
abline(v = t.sample, col="red", lwd=3)
```



Question 4

```r
# calculate estimated betas for full sample
lmout<-lm(q34.data$Y~q34.data$X1+q34.data$X2+q34.data$X3)
betas.fs<-cbind(as.vector(lmout$coefficients))

# bootstrap
B<-1000
bboot.par<-matrix(nrow=B,ncol=ncol(data)-1) # Subtract 1 because data includes y
for(i in 1:B){
  # sample the estimated residuals, with replacement
  bootresid<-sample(lmout$residuals,replace=TRUE)
  # calculate the fitted value using the sampled residuals
  yboot<-x%*%betas.fs+bootresid
  # regress bootstrapped fitted value on x values, saving betas
  bboot.par[i,]<-t(ols(yboot,x)$betas)
}
```

Answer to Question 4a

```
apply(bboot.par,2,function(x) quantile(x,p=c(0.025,0.975)))
```

```
##            [,1]     [,2]     [,3]     [,4]
## 2.5%   9.916147 1.980537 3.956982 2.965071
## 97.5% 10.744976 2.020364 4.013518 3.005828
```

Answer to Question 4b

```
# Estimate bootstrapped variance-covariance matrix of these OLS estimators
k = dim(bboot.par)[2]
M = matrix(nrow=k,ncol=k)
for(k in 1:4){
  for (r in 1:4){
    M[r,k] = cov(bboot.par[,r],bboot.par[,k])
  }
}
M
```

```
##               [,1]          [,2]          [,3]          [,4]
## [1,]  0.0411521524 -9.972942e-04 -1.097438e-03 -1.659293e-03
## [2,] -0.0009972942  1.082662e-04 -1.349813e-07 -4.009421e-06
## [3,] -0.0010974385 -1.349813e-07  2.174857e-04  2.216347e-06
## [4,] -0.0016592928 -4.009421e-06  2.216347e-06  1.118945e-04
```

Question 5

```
q56.data = read.csv("questions5_6.csv")
ols.mat<-as.matrix(q56.data)
y<-ols.mat[,1]
x<-cbind(1,ols.mat[,2])
data<-cbind(y,x)

output.ols<-ols(y,x)

# Non-parametric bootstrap
B<-1000
bboot<-matrix(nrow=B,ncol=ncol(data)-1)
k = 1:nrow(data)
for(i in 1:B){
  indxboot <- sample(k,replace=TRUE) # Sample the indices
  datab <-data[indxboot,] #draw row using drawn index place
  yb <-datab[,1]
  xb <-datab[,2:ncol(data)]
  bboot[i,]<-t(ols(yb,xb)$betas)
}
```

Answer to Question 5a

```
# Bootstrapped standard errors
sqrt(var(bboot[,1]))
```

```
## [1] 0.1627726
```

```
sqrt(var(bboot[,2]))
```

```
## [1] 0.02658472
```

```
# Bootstrap bias
diff<-matrix(nrow=B,ncol=2)
# calculate bias for each coefficient
for(i in 1:2){
  diff[,i]<-bboot[,i]-output.ols$betas[i]
}
bias<-apply(diff,2,mean)
bias
```

```
## [1] -0.0039499940  0.0008214442
```

Answer to Question 5b

Our non-parametric bootstrap breaks the structure of the data, by sampling rows. Each row is related to each other since this is an AR(1) model (yi is a function of the previous y, $yi - 1$) since this is an AR(1) model.

Answer to Question 5c

The crucial assumption for bootstrapping to work in this AR(1) model is that the error terms are iid. The parametric bootstrap is built by sampling a residual from the full sample, each of which are independent from each other.

```
lmout<-lm(q56.data$y~q56.data$lagy)
betas.fs<-cbind(as.vector(lmout$coefficients))
B<-1000
bboot.par<-matrix(nrow=B,ncol=ncol(data)-1) # Subtract 1 because data includes y
for(i in 1:B){
  # sample the estimated residuals, with replacement. This is okay since they are iid
  bootresid<-sample(lmout$residuals,replace=TRUE)
  yboot<-x%*%betas.fs+bootresid
  bboot.par[i,]<-t(ols(yboot,x)$betas)
}
```

Answer to Question 5d

```
# Bootstrapped standard errors
sqrt(var(bboot.par[,1]))
```

```
## [1] 0.1598648
```

```
sqrt(var(bboot.par[,2]))
```

```
## [1] 0.02621326
```

```
# Bootstrap bias
diff<-matrix(nrow=B,ncol=2)
# calculate bias for each coefficient
for(i in 1:2){
  diff[,i]<-bboot.par[,i]-output.ols$betas[i]
}
bias<-apply(diff,2,mean)
bias
```

```
## [1] -0.005813089  0.001033535
```

Question 6

Answer to Question 6a

```
# Answer to Question 6a
quantile(bboot.par[,1],p=c(0.025,0.975))
```

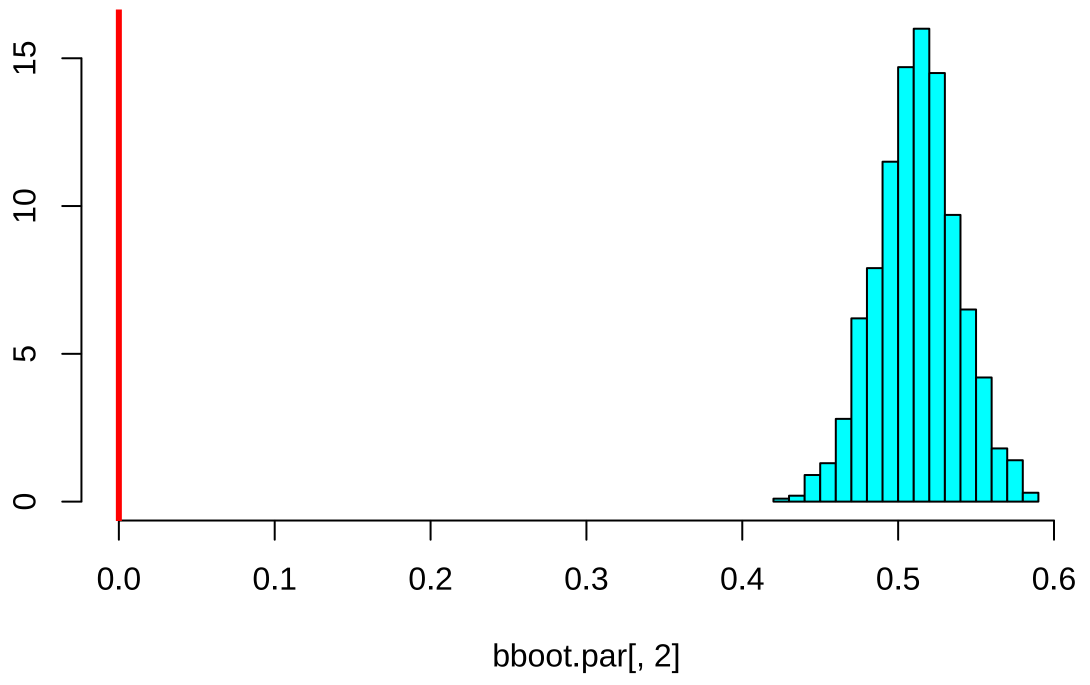
```
##     2.5%    97.5%
## 2.628482 3.259408
```

```
quantile(bboot.par[,2],p=c(0.025,0.975))
```

```
##      2.5%     97.5%
## 0.4600045 0.5640273
```

So we reject the null that p = 0

```
# reject the null
truehist(bboot.par[,2],xlim=c(0,0.6))
abline(v = 0, col="red",lwd=3)
```



```
# bootstrapped p-value is zero (not the most precise answer, admittedly)
```

Answer to Question 6b

```
bboot.mc = matrix(nrow=B,ncol=ncol(data)-1)

for(i in 1:B){
  # first, create new data, from one randomly sampled row from full sample
  u = rnorm(1000,0,1)
  z = numeric(1000)
  # set z1 = 0
  z[1] = 0
  # set z2 = to random draw from data. This will become first value of y.
  indxboot = sample(k,replace=TRUE)[1]
  z[2] = data[indxboot,][1]
  # fill in data, using formula from before
  for (j in 3:1000)  z[j]<-3+0.5*z[j-1]+u[j]
  # create y, lagy from z
  y<-z[2:1000]
  lagy<-z[1:999] # (z1 is the first lagged y)

  lmout<-lm(y~lagy) #can't use $ here - atomic operators?
  betas.mc<-cbind(as.vector(lmout$coefficients))

  x = cbind(1,lagy)
  mc.data = cbind(y,x)
  mc.data = as.matrix(mc.data)

  # now run parametric bootstrap (as in Question 5)
  bootresid<-sample(lmout$residuals,replace=TRUE)
  yboot<-x%*%betas.mc+bootresid
  bboot.mc[i,]<-t(ols(yboot,x)$betas)
}

sqrt(var(bboot.mc[,1]))
```

```
## [1] 0.2465522
```

```
sqrt(var(bboot.mc[,2]))
```

```
## [1] 0.04020221
```

```
diff<-matrix(nrow=B,ncol=2)
for(i in 1:2){
  diff[,i]<-bboot.mc[,i]-output.ols$betas[i]
}
bias<-apply(diff,2,mean)
bias #hmm, it's larger than before...
```

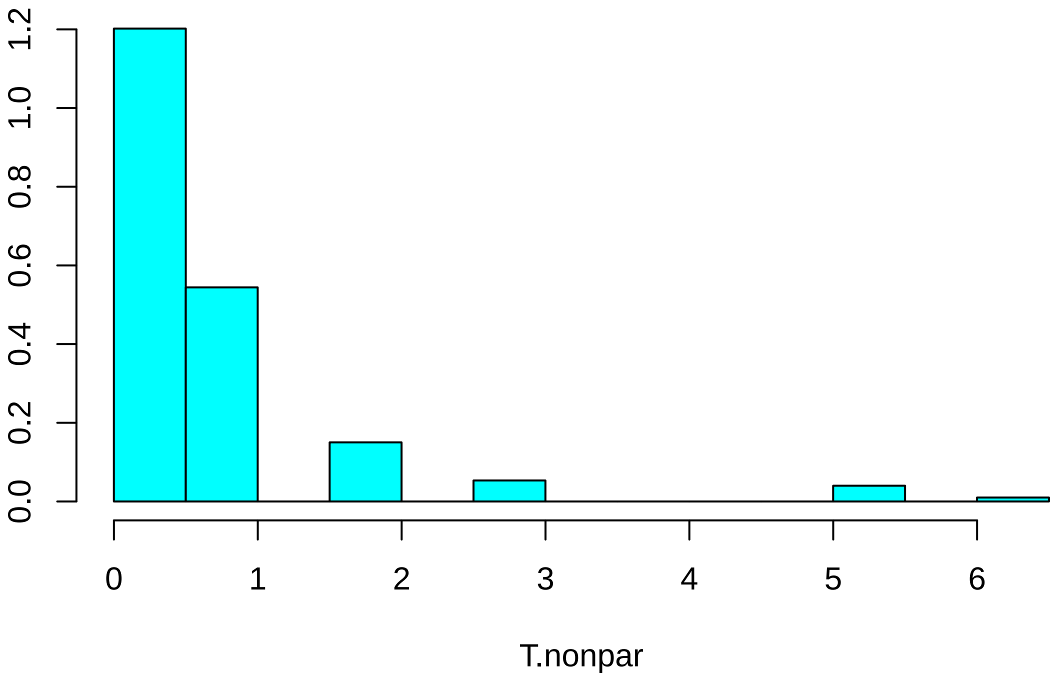
```
## [1]  0.15711999 -0.02804445
```

Question 7

```r
n = 1000
x = runif(n)
max.x = max(x)
B = 599
```

Answer to Question 7a

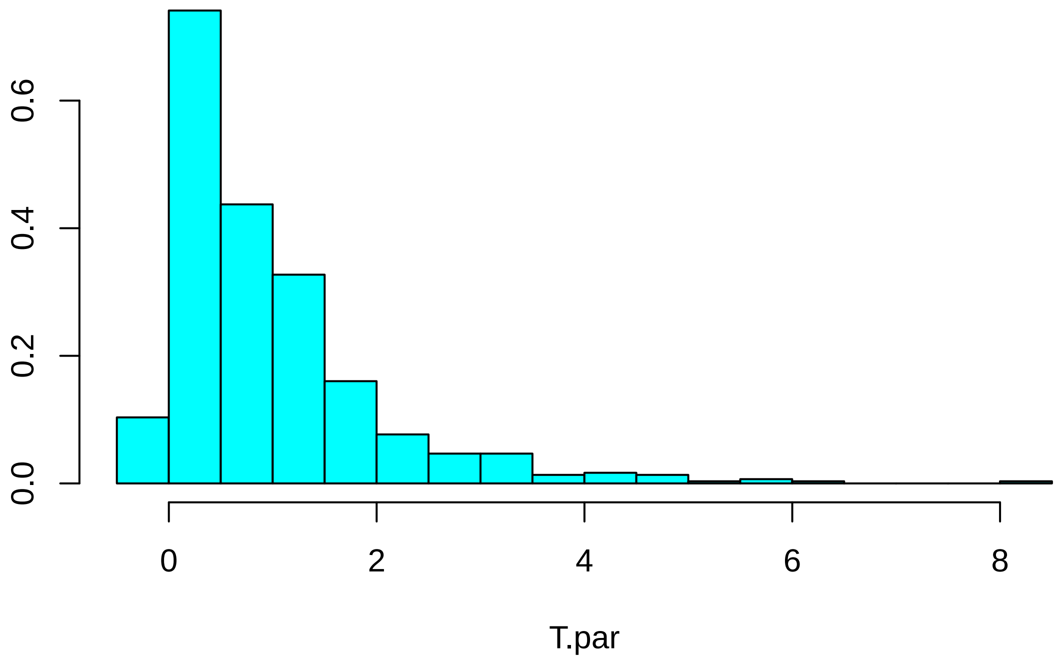
```r
T.nonpar = numeric(B)
for(i in 1:B) {
  xboot = sample(x, replace=TRUE)
  T.nonpar[i] = n * (max.x - max(xboot))
}
truehist(T.nonpar)
```



It coincides with the theoretical exponential(1) distribution, but there's funky bunching close to T=0.

Answer to Question 7b

```r
T.par = numeric(B)
for(i in 1:B) {
  x = runif(n)
  T.par[i] = n * (max.x - max(x))
}
truehist(T.par)
```



Yes, it coincides with the theoretical exponential(1) distribution, and much more so than the non-parametric bootstrap.

Answer to Question 7c

As n increases, we see more bunching in our histogram – the probability that the max(x) in the full sample is equal to the max(x) in the bootstrap increases. The non-parametric bootstrap fails because the distribution of the test statistic is not continuous, but instead truncated at zero.

```r
# non-parametric bootstrap is truncated, but parametric bootstrap is not
min(T.nonpar)
```

```
## [1] 0
```

```r
min(T.par)
```

```
## [1] -0.0597632
```