Homework 2

Casey Gibson

Problem 1

If we are given

$$X_1, X_2, X_3, X_4$$

a) We know that the number of distinct boostrap samples is given by

$$\binom{2n-1}{n-1} = \binom{2*4-1}{4-1} = 35$$

$$[X_1, X_1, X_3, X_4], [X_1, X_1, X_3, X_3], [X_1, X_1, X_3, X_1], [X_1, X_1, X_3, X_2]$$

b) If $n = 10$ then the probability that $X_1 \in B$ exactly k times is simply binomial

$$\binom{n}{k}(\frac{1}{10})^k(\frac{9}{p})^{10-k}$$

which we can see is

```
prob1 <- function(n,k) {

  return (dbinom(k,n,.1))
}

print (signif(prob1(10,0),3))
```

```
## [1] 0.349
```

```
print (signif(prob1(10,1),3))
```

```
## [1] 0.387
```

```
print (signif(prob1(10,2),3))
```

```
## [1] 0.194
```

```
print (signif(prob1(10,3),3))
```

```
## [1] 0.0574
```

```
print (signif(prob1(10,4),3))
```

```
## [1] 0.0112
```

c)

```
set.seed(1) #set a random seed
n=10 #sample size (or number of subjects in the original sample)
N=100000 #number of bootstrap replications
store=rep(NA,N)
for(i in 1:N){
      store[i]=sum(sample(1:n, rep=TRUE)==1)
```

```
}
store <- table(store)
print (store['0']/N)
```

```
##       0
## 0.34781
```

```
print (store['1']/N)
```

```
##       1
## 0.38891
```

```
print (store['2']/N)
```

```
##       2
## 0.19408
```

```
print (store['3']/N)
```

```
##       3
## 0.05617
```

```
print (store['4']/N)
```

```
##       4
## 0.01124
```

Problem 2

Suppose that $\hat{\phi} = m(\hat{\theta})$ for $m(*)$ is a monotone function. The p-value based on the permutation test based on $\hat{\theta} = P(\hat{\theta}^* \geq \hat{\theta})$

As long as m is a monotone function we can simply take

$$P(m(\hat{\theta}^*) \geq m(\hat{\theta}))$$

which by definition is

$$ASL_{perm}(\hat{\phi})$$

Problem 3

Let

$$\hat{\phi} = \frac{\hat{\theta}}{\bar{\sigma}\sqrt{\frac{1}{n} + \frac{1}{m}}}$$

In order to show that

$$\hat{ASL}_{perm}(\hat{\phi}) = \hat{ASL}_{perm}(\hat{\theta})$$

We can use the result from problem 2 as long as we can show that

$$m(x) = \frac{x}{\bar{\sigma}\sqrt{\frac{1}{n} + \frac{1}{m}}}$$

Is a monotonically increasing function. We know that the denominator is positive regardless of $\bar{z}$ and $\bar{y}$, so clearly this is an increasing function in x. ### Problem 4

Suppose we have

$$3.5, 4, 7, 7.3, 8.6, 12.4, 13.8, 18.1$$

the trimmed mean can be calculated as

$$\frac{1}{4}(7 + 7.3 + 8.6 + 12.4) = 8.825 = \hat{\theta}$$

a)

```
library(boot)

set.seed(1)
data <- c(3.5,4,7,7.3,8.6,12.4,13.8,18.1)

alpha.fn <- function(data,index){
  return (mean(data[index],trim=.25 ))
}
for(i in c(25,100,200,500,1000,2000)){
  bt.results=boot(data,alpha.fn,R=i)
  print (c(i,sd(bt.results$t)))
}
```

```
## [1] 25.000000  1.777637
## [1] 100.000000   2.243242
## [1] 200.000000   2.100808
## [1] 500.000000   2.138062
## [1] 1000.000000    2.081176
## [1] 2000.000000    2.155132
```

b)

```
library(boot)


ses <- matrix(NA,nrow=10,ncol=6)
data <- c(3.5,4,7,7.3,8.6,12.4,13.8,18.1)

alpha.fn <- function(data,index){
  return (mean(data[index],trim=.25 ))
}

for (j in seq(1,10)){
  set.seed(j)
  count <- 1
  for(i in c(25,100,200,500,1000,2000)){
   bt.results=boot(data,alpha.fn,R=i)
    ses[j,count] <- sd(bt.results$t)
    count <- count +1
  }
}
print (ses)
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
##  [1,] 1.777637 2.243242 2.100808 2.138062 2.081176 2.155132
```

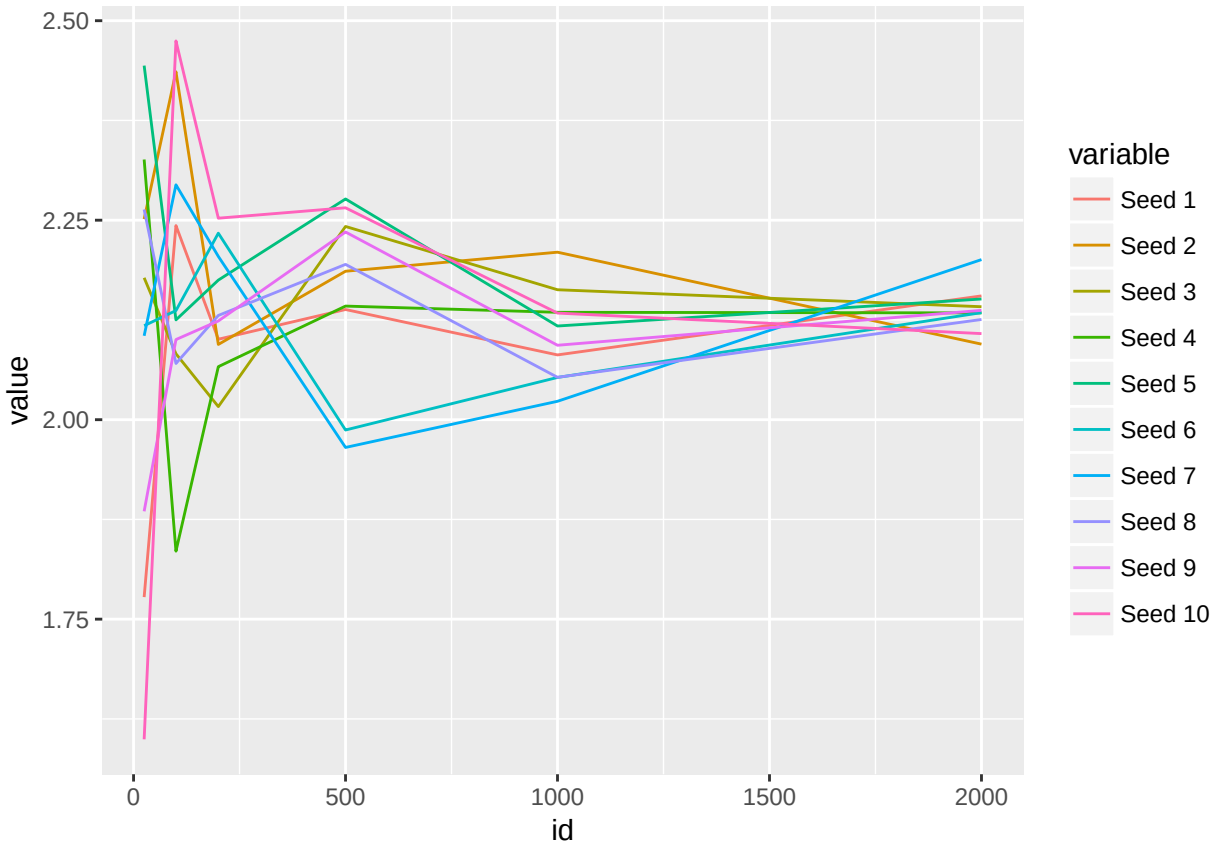
```
##  [2,] 2.251410 2.435993 2.094312 2.186033 2.209901 2.094663
##  [3,] 2.177824 2.082600 2.016397 2.242148 2.162851 2.141547
##  [4,] 2.326071 1.835242 2.066420 2.142356 2.134566 2.133815
##  [5,] 2.443711 2.125084 2.174907 2.276592 2.117404 2.151229
##  [6,] 2.117795 2.136727 2.233733 1.987068 2.052937 2.134137
##  [7,] 2.105117 2.294353 2.204610 1.965216 2.023042 2.200604
##  [8,] 2.263385 2.070422 2.130639 2.194629 2.053079 2.125588
##  [9,] 1.885155 2.100407 2.123717 2.235237 2.093273 2.137008
## [10,] 1.599416 2.474654 2.252496 2.265527 2.133597 2.107874
```

```
library("reshape2")
library("ggplot2")

colnames_ <- c("Seed 1","Seed 2","Seed 3","Seed 4","Seed 5","Seed 6","Seed 7","Seed 8","Seed 9","Seed 1
test_data_long <- as.data.frame(t(ses))
colnames(test_data_long) <- colnames_
test_data_long$id <- c(25,100,200,500,1000,2000)

test_data_long <- melt(test_data_long, id="id")  # convert to long format


ggplot(data=test_data_long,
       aes(x=id, y=value, colour=variable)) +
       geom_line()
```



It looks like once we hit above 1500 bootstrap samples, the results seem to stabilize.

Problem 5

```
n=15
data0=as.data.frame(matrix(0, n, 2))
colnames(data0)=c("x", "y")
data0$X=c(44, 13, 2, 57, 36, 6, 79, 84, 25, 13, 24, 24, 15, 20, 7)
data0$Y=c(34, 41, 6, 24, 29, 13, 9, 4, 7, 19, 19, 3, 1, 1, 19)
x=data0$x
y=data0$y
```

a)

```
R=1000

library(ISLR)
#
alpha.fn=function(data, weight, varname){
  weight.2=weight/mean(weight) #standardize the weights
  return(weighted.mean(unlist(data[,varname]),w=weight.2))
}


alpha.est=alpha.fn(data=data0, weight=rep(1, n),varname = "X")







bayesian.boot=function(data, statistic, R, varname){
  data=as.data.frame(data)
  n1=nrow(data)
  bt.est.original=alpha.fn(data, weight=rep(1, n1),varname = varname )
  bt.est=rep(NULL, R)
  for (i in 1:R){
    weight=rexp(n1, rate=1)
    bt.est[i]=alpha.fn(data, weight,varname = varname)
  }
  std=sd(bt.est)
  results=list(t0=bt.est.original, t=bt.est, std=std)
  return(results)
}

bt.results_x=bayesian.boot(data0,alpha.fn,R=1000, varname="X")

print (var(bt.results_x$t))
```

```
## [1] 38.6669
```

b)

```
bt.results_y=bayesian.boot(data0,alpha.fn,R=1000, varname="Y")

print (var(bt.results_y$t))
```

```
## [1] 9.384961
```

c)

```
print (var(bt.results_x$t - bt.results_y$t))
```

```
## [1] 48.81886
```

```
print (quantile(bt.results_x$t - bt.results_y$t,probs = c(.05)))
```

```
##       5%
## 3.765879
```

$$H_0 = \text{ There is no treatment effect, } \bar{X} - \bar{Y}$$
$$H_A = \text{ There is a treatment effect, } \bar{X} - \bar{Y} \neq 0$$

We can reject H_0 at $\alpha = .05$ level based on the CI above.

d

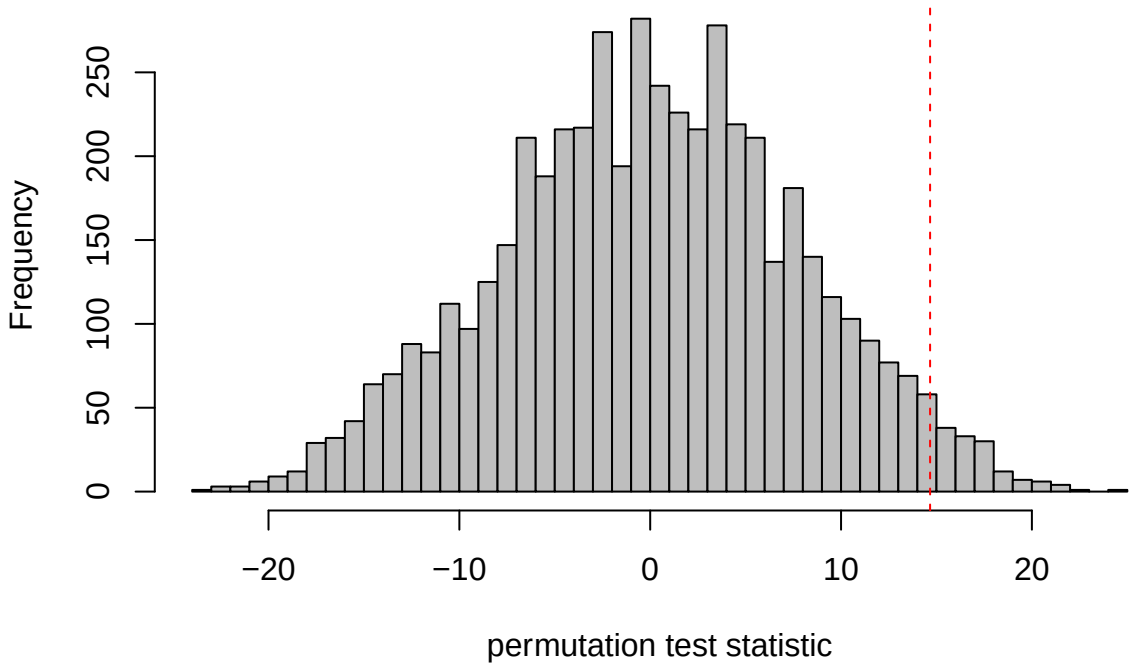
```
perm.test = function(x, y, N){
  #x: observations in group 1
  #y: observations in group2
  #N: number of random samples from permutation distribution
  z=c(x,y)
  z.sort=sort(z)
  n=length(x)
  m=length(z)
  test.stat=rep(NULL,N)
  for(i in 1:N){
    perm.index=sample(1:m, n)
  perm.x = z.sort[perm.index]
  perm.y = z.sort[-perm.index]
  test.stat[i] = mean(perm.x) - mean(perm.y)
  }
  test.stat
}


obs.test.stat=mean(data0$X)-mean(data0$Y)
test.2=perm.test(x=data0$X,y=data0$Y,N=5000)

p.value.1=sum(test.2>=obs.test.stat)/length(test.2)
p.value.1
```

```
## [1] 0.0314
```

```
hist(test.2, nclass=50, col=8, main="", xlab="permutation test statistic")
abline(v=obs.test.stat, col=2, lty="dashed")
```



e)

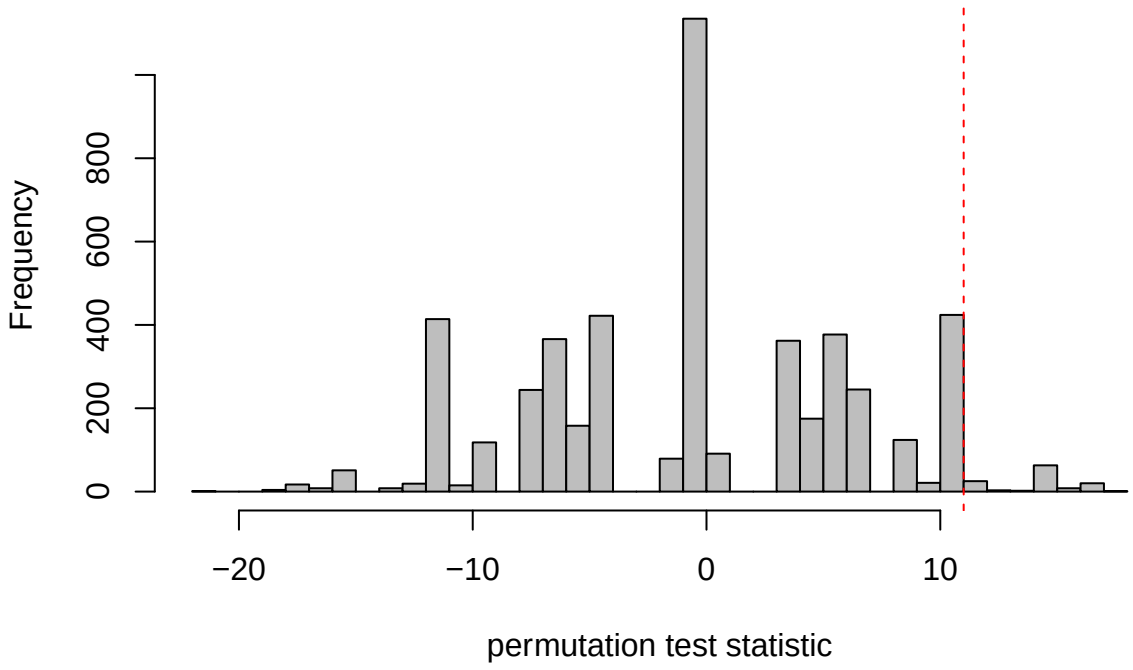
```
perm.test = function(x, y, N){
  #x: observations in group 1
  #y: observations in group2
  #N: number of random samples from permutation distribution
  z=c(x,y)
  z.sort=sort(z)
  n=length(x)
  m=length(z)
  test.stat=rep(NULL,N)
  for(i in 1:N){
    perm.index=sample(1:m, n)
  perm.x = z.sort[perm.index]
  perm.y = z.sort[-perm.index]
  test.stat[i] = median(perm.x) - median(perm.y)
  }
  test.stat
}


obs.test.stat=median(data0$X)-median(data0$Y)
test.2=perm.test(x=data0$X,y=data0$Y,N=5000)

p.value.1=sum(test.2>=obs.test.stat)/length(test.2)
p.value.1
```

```
## [1] 0.1092
```

```
hist(test.2, nclass=50, col=8, main="", xlab="permutation test statistic")
abline(v=obs.test.stat, col=2, lty="dashed")
```



Problem 6

a) Suppose

$$X \sim U(0, \theta)$$

and we draw an iid random sample

$$X_1, X_2, ..., X_n$$

We can write the likelihood as

$$L(\theta | X_1, X_2, ..., X_n) = \prod_{i=1}^{n} \frac{1}{\theta} I_{x_i < \theta}$$

This is only nonzero if all $x_i < \theta$ so the likelihood > 0 occurs for any $\theta > x_i$. However, since we have a θ in the denominator, we penalize larger values of θ so we want to take $\theta = max(X_i)$ in order to achieve the maximum.

b) We know from a previous homework that the probability X_j is included in a bootstrap sample $[X_1^*, X_2^*, ..., X_n^*]$ is $(1 + \frac{1}{n})^n$

In particular we can take $X_j = X_{(n)}$, that is the n-th order statistic (max). The only way that $\hat{\theta^*} = \hat{\theta}$ is if $X_{(n)} \in [X_1^*, X_2^*, ..., X_n^*]$, therefore we see that

$$P(\hat{\theta^*} = \hat{\theta}) = 1 - (1 - \frac{1}{n})^n$$

In the limit this becomes $1 - \frac{1}{e} = .632$

c)

```
set.seed(1) #set a random seed
n=50 #sample size
x=runif(n) #this gives the original sample with 50 observations
```

```
print (max(x))
```

```
## [1] 0.9919061
```

d)

```
alpha.fn <- function(data,index){
  return (max(data[index] ))
}

boot.res <-boot(x,alpha.fn,2000)
print (sd(boot.res$t))
```

```
## [1] 0.02801751
```

e)

```
pb <- runif(50,min=0,max=max(x))
curr_theta <- max(pb)
b_samples <-c(curr_theta)

for (i in seq(1, 2000)){
  pb <- runif(50,0,curr_theta)
  b_samples <-c(b_samples,max(pb))
}
print (sd(b_samples))
```

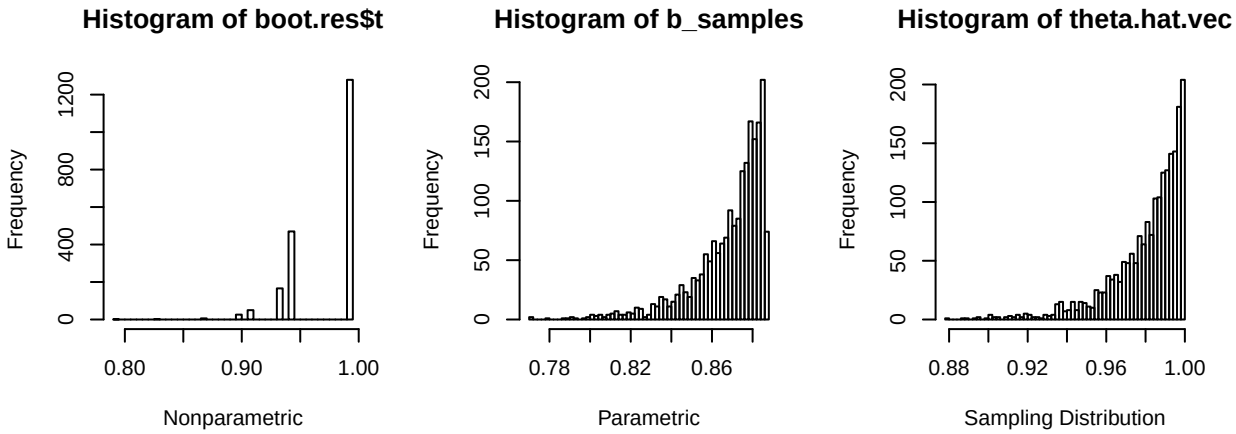
```
## [1] 0.01775496
```

f)

```
set.seed(2)
N=2000
theta.hat.vec=rep(NULL,N)
for (i in 1:N){
    x2=runif(n)
      theta.hat.vec[i]=max(x2)
}
print (sd(theta.hat.vec))
```

```
## [1] 0.01864542
```

```
par(mfrow = c(1, 3), pty = "s")
hist(boot.res$t,
nclass=50, xlab = "Nonparametric")
hist(b_samples,
nclass=50, xlab = "Parametric")
hist(theta.hat.vec,
nclass=50,, xlab = "Sampling Distribution")
```



Estimates based on the parametric bootstrap are much closer to the truth than the non-parametric bootstrap. This is because of the fact that, in the limit, the nonparametric bootstrap only contains the max 63.2% of the time.