BMEG 802 – Advanced Biomedical Experimental Design and Analysis

Assignment 7

Joshua G. A. Cashaback, PhD

Question 1

Gibbs Sampling from the conditional probabilities of a bivariate normal distribution:

$$p(x_{i+1} \mid y_i) = \mathcal{N}\left(\mu_X + \frac{\sigma_X}{\sigma_Y}\rho(y_i - \mu_Y),\ (1-\rho^2)\sigma_X^2\right)$$

$$p(y_{i+1} \mid x_{i+1}) = \mathcal{N}\left(\mu_Y + \frac{\sigma_Y}{\sigma_X}\rho(x_{i+1} - \mu_X),\ (1-\rho^2)\sigma_Y^2\right)$$

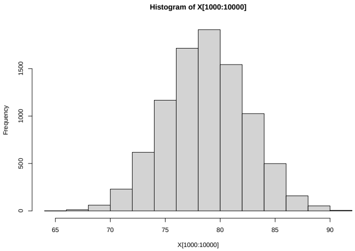
Question 1 Cont’d

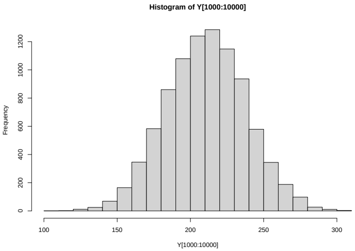
```r
mu_x = 78.8
sigma_x = 3.668
mu_y = 211
sigma_y = 26.904
rho = 0.81
n = 10000 # number of samples to perform
X = array(NA,n) # initialize arrays
Y = array(NA,n)
X[1] = 78.8 # initial guess
Y[1] = 211
for (i in 2:n) {
  mu_c1 = mu_x + sigma_x / sigma_y * rho * (Y[i-1] - mu_y)
  sigma_c1 = ((1 - rho^2) * sigma_x^2)^(1/2) # SD
  X[i] = rnorm(1,mu_c1,sigma_c1)
  mu_c2 = mu_y + sigma_y / sigma_x * rho * (X[i] - mu_x) # NOTE THE DEPENDANCE of p(Y|X) in p(X|Y)
  sigma_c2 = ((1 - rho^2) * sigma_y^2)^(1/2)
  Y[i] = rnorm(1,mu_c2,sigma_c2)
}
```

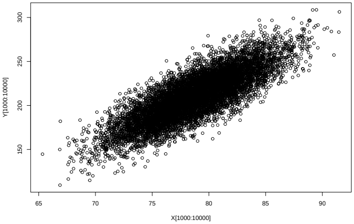
Question 1 Cont'd

Marginal and Joint Posterior Probabilities

```r
hist(X[1000:10000]) # allowing for a 1000 sample "burn-in" period
hist(Y[1000:10000])
plot(X[1000:10000],Y[1000:10000])
```







Question 2

Bootstrap mean and median

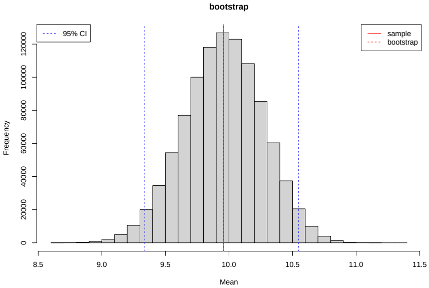
```
X = c(8.453532, 10.025041, 11.495339, 9.367600, 8.333229,
      9.788753, 10.883344, 10.543059, 9.869095, 10.799819)
Xm = mean(X) # mean
MED = median(X) # mean
boot_m = 1000000 # how many simulated experiments?
Xm_boot = array(NA, boot_m) # mean
Med_boot = array(NA, boot_m) # mean
for (i in 1:boot_m) {
  Xb = sample(X, length(X), replace=TRUE) # generate new sample
  Xm_boot[i] = mean(Xb) # compute statistic of interest
  Med_boot[i] = median(Xb) # compute statistic of interest
}
```

Question 2 Cont’d

Mean

```r
# display results
hist(Xm_boot, xlab="Mean", main="bootstrap")
abline(v=Xm, col="red")
abline(v=mean(Xm_boot), col="red", lty=2)
legend(x="topright", lty=c(1,2), col=c("red","red"), legend=c("sample","bootstrap"))

# compute 95% CI
CI95xm = quantile(Xm_boot, probs=c(.025,.975))
abline(v=CI95xm[1], lty=2, col="blue")
abline(v=CI95xm[2], lty=2, col="blue")
legend(x="topleft", lty=2, col="blue", legend="95% CI")
```

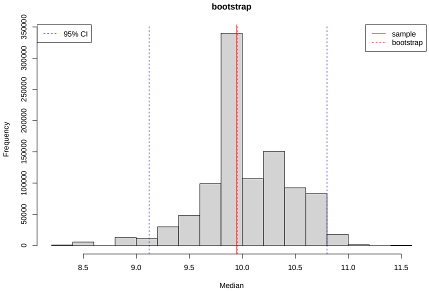


Question 2 Cont’d

Median

```
# display results
hist(Med_boot, xlab="Median", main="bootstrap")
abline(v=MED, col="red")
abline(v=mean(Xm_boot), col="red", lty=2)
legend(x="topright", lty=c(1,2), col=c("red","red"), legend=c("sample","bootstrap"))

# compute 95% CI
CI95med = quantile(Med_boot, probs=c(.025,.975))
abline(v=CI95med[1], lty=2, col="blue")
abline(v=CI95med[2], lty=2, col="blue")
legend(x="topleft", lty=2, col="blue", legend="95% CI")
```



Question 2 Cont'd

Answers should be similar and make sense.

```r
mean(Xm_boot)
```

```
## [1] 9.956119
```

```r
CI95xm[1]
```

```
##     2.5%
## 9.337698
```

```r
CI95xm[2]
```

```
##    97.5%
## 10.54622
```

```r
mean(Med_boot)
```

```
## [1] 10.03176
```

```r
CI95med[1]
```

```
##     2.5%
## 9.121142
```

```r
CI95med[2]
```

```
##    97.5%
## 10.79982
```

Question 3

```
pre = c(22,25,17,24,16,29,20,23,19,20)
post = c(18,21,16,22,19,24,17,21,23,18)
paired_diff0 = post - pre
paired_diff0
```

```
##  [1] -4 -4 -1 -2  3 -5 -3 -2  4 -2
```

```
mean_diff = mean(paired_diff0)
mean_diff
```

```
## [1] -1.6
```

```
paired_diff1 = paired_diff0 - mean_diff
paired_diff1
```

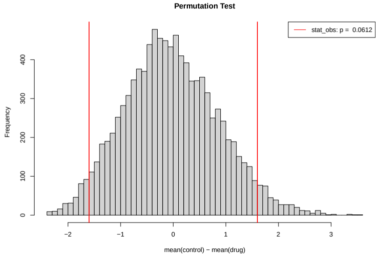
```
##  [1] -2.4 -2.4  0.6 -0.4  4.6 -3.4 -1.4 -0.4  5.6 -0.4
```

```
boot_m = 10000 # how many simulated experiments?
pd_boot = array(NA, boot_m) # create a list to store our bootstrap values
for (i in 1:boot_m) {
  Xb = sample(paired_diff1, length(paired_diff1), replace=TRUE) # generate new sample
  pd_boot[i] = mean(Xb) # compute statistic of interest
}
```

Question 3

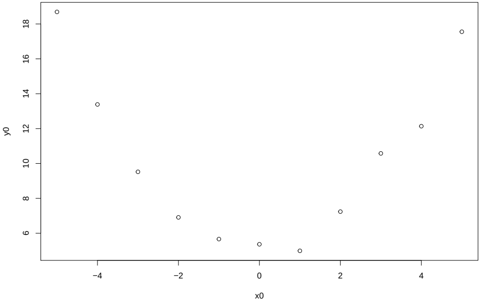
```
# visualize the empirical permutation distribution of our statistic of interest
hist(pd_boot, 50, xlab="mean(control) - mean(drug)", main="Permutation Test")
abline(v=mean_diff, col="red", lwd=2)
abline(v=-mean_diff, col="red", lwd=2)
# how many times in the permutation tests did we observe a stat_perm bigger than or smaller than the stat_obs?
p_perm0 <- length(which(pd_boot <= mean_diff)) / boot_m
p_perm1 <- length(which(pd_boot >= -mean_diff)) / boot_m
p_perm2 <- p_perm0 + p_perm1
legend(x="topright", lty=1, col="red", legend=paste("stat_obs: p = ", p_perm2))
p_perm2
```

```
## [1] 0.0612
```



Question 4

```r
x0 = seq(-5,5,1)
## Can use commented out code to generate data
#noise = rnorm(11,0,1.0)
#a = 5.0
#b = 0.5 # pretend you don't know what the constant is (we are going to estimate this)
#y0 = a + b * x0^2 + noise
#y0
y0 = c(18.691806, 13.385586, 9.522445, 6.908887, 5.664939, 5.367358, 4.991651, 7.239345, 10.578517, 12.138775, 17.554915)
plot(x0,y0)
```



Question 4 Cont'd

Least Squares Loss Function Optimizer

```r
lse <- function(P,X,Y) {
  a = P[1]
  b = P[2]
  x0 = X
  y0 = Y
  mindiff <- sum((y0 - (a + b * x0^2))^2) # sum of least squares
  return(mindiff)
}

x0 = seq(-5,5,1) # x-values
y0 = c(18.691806, 13.385586, 9.522445, 6.908887, 5.664939, 5.367358, 4.991651, 7.239345, 10.578517, 12.138775, 17.554915) #y-values
p_init = c(10,10) # initial guess of the constant, c
opt <- nlm(f = lse, p_init,x0,y0) # using the previously defined loss function
opt$estimate
```

```
## [1] 5.0311330 0.5154706
```

Question 4 Cont’d

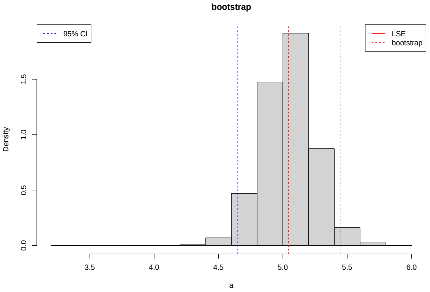
Bootstrapping

```r
x0 = seq(-5,5,1) # x-values
y0 = c(18.691806, 13.385586, 9.522445, 6.908887, 5.664939, 5.367358, 4.991651, 7.239345, 10.578517, 12.138775, 17.554915) #y-values
boot_m = 10000 # how many simulated experiments?
A_boot = array(NA, boot_m) # create a list to store our bootstrap values
B_boot = array(NA, boot_m)
X = array(NA, length(x0))
Y = array(NA, length(y0))
p_init = c(10,10) # initial guess of the constant, c
for (i in 1:boot_m) {
  for (j in 1:length(x0)){# inner loop used to preserve x,y pairs
    k = sample(1:length(x0), 1) # sample random integer
    X[j] = x0[k] # use integer to sample x,y pair
    Y[j] = y0[k]
  }
  P_opt = nlm(f = lse, p_init,X,Y)$estimate
  A_boot[i] = P_opt[1]
  B_boot[i] = P_opt[2]
}
```

Question 4 Cont'd

Plot *a* distribution and confidence intervals

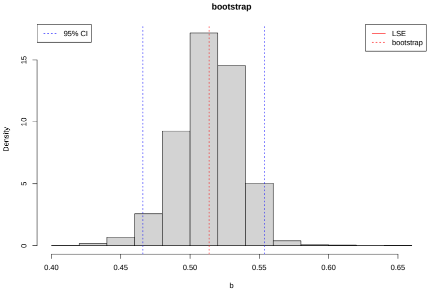
```
# display results
hist(A_boot, xlab="a", main="bootstrap", freq=FALSE)
abline(v=mean(A_boot), col="red", lty=2)
legend(x="topright", lty=c(1,2), col=c("red","red"), legend=c("LSE","bootstrap"))
# compute 95% CI
CI95a = quantile(A_boot, probs=c(.025,.975))
abline(v=CI95a[1], lty=2, col="blue")
abline(v=CI95a[2], lty=2, col="blue")
legend(x="topleft", lty=2, col="blue", legend="95% CI")
```



Question 4 Cont'd

Plot *b* distribution and confidence intervals

```r
# display results
hist(B_boot, xlab="b", main="bootstrap", freq=FALSE)
abline(v=mean(B_boot), col="red", lty=2)
legend(x="topright", lty=c(1,2), col=c("red","red"), legend=c("LSE","bootstrap"))
# compute 95% CI
CI95b = quantile(B_boot, probs=c(.025,.975))
abline(v=CI95b[1], lty=2, col="blue")
abline(v=CI95b[2], lty=2, col="blue")
legend(x="topleft", lty=2, col="blue", legend="95% CI")
```



Question 4 Cont'd

Confidence Intervals

```
CI95a[1]
```

```
##     2.5%
## 4.645516
```

```
CI95a[2]
```

```
##    97.5%
## 5.444637
```

```
CI95b[1]
```

```
##      2.5%
## 0.4659715
```

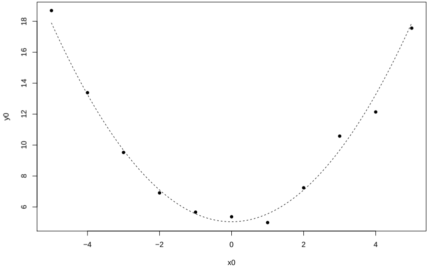
```
CI95b[2]
```

```
##     97.5%
## 0.5535634
```

Question 4 Cont'd

Plot best fit parameters from bootstrap means.

```r
x0 = seq(-5,5,1) # x-values
y0 = c(18.691806, 13.385586, 9.522445, 6.908887, 5.664939, 5.367358, 4.991651, 7.239345, 10.578517, 12.138775, 17.554915) #y-values
xfit = seq(-5,5,0.1)
yfit =  mean(A_boot) + mean(B_boot)*xfit^2
plot(x0,y0,pch=16)
lines(xfit,yfit,lty=2)
```


18
16
14
12
10
8
6
y0
−4
−2
0
2
4
x0