Ec/ACM/CS 112. Problem set 2

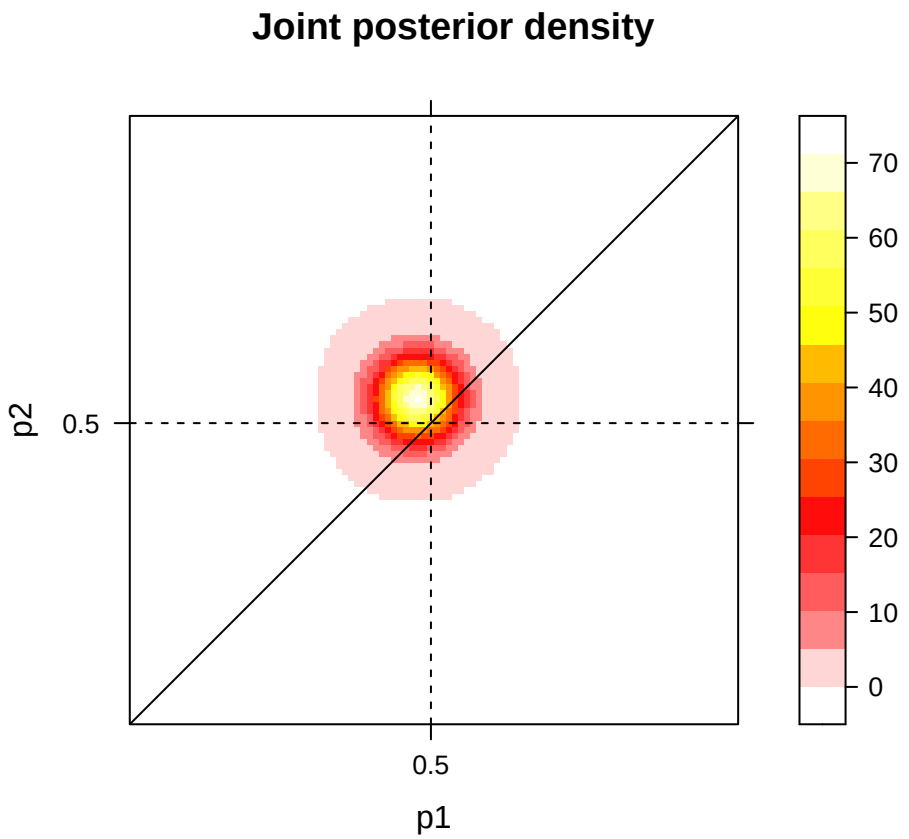
Ben Juarez

PART 1

step 1: Fit using only old data

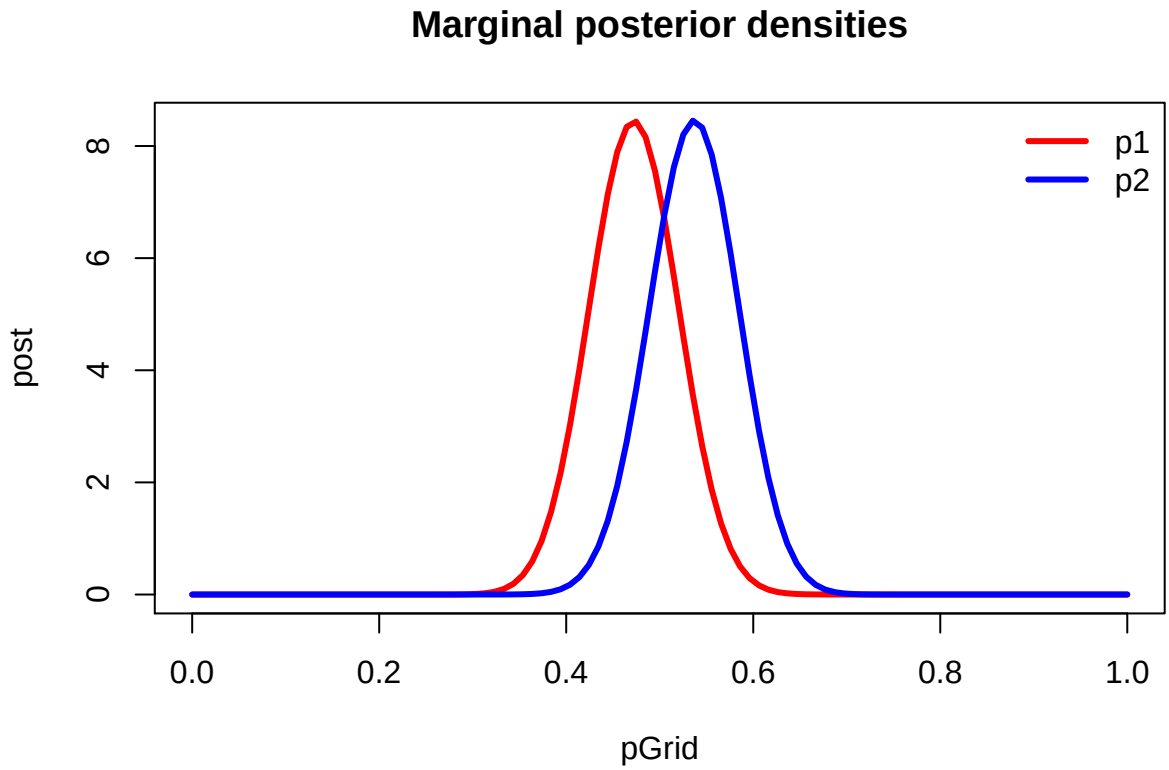
```
### NOTE: Utilized provided code from lecture 2.2
data = read.csv(file = "~/Desktop/PS2_data.csv")
# set random seed
set.seed(123)
# define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 /nGridPoints
# prior params
aPrior = 5
bPrior = 5
# Data collected by ar
data = subset(data, tosser == "ar")
data1 = data$ball_1
data2 = data$ball_2
nWater_1 = sum(data1)
nWater_2 = sum(data2)
n1 = length(data1)
n2 = length(data2)

### Joint posterior density as heat map ###
# define prior matrix
# >> note: assume independent non-uniform priors
prior = dbeta(x = pGrid, shape1 = aPrior, shape2 = bPrior)
priorM = matrix(rep(1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
                byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    priorM[row,col] = prior[row] * prior[col]
  }
}
# compute posterior
postM = matrix(rep(-1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
               byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    p1 = pGrid[row]
    p2 = pGrid[col]
    postM[row,col] = dbinom(nWater_1,n1,p1) * dbinom(nWater_2,n2,p2) * priorM[row,col]
```

```
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(50),
          labels=c(0.5)), y=list(at=c(50), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=50, col = "black", lty=2)
            panel.abline(h=50, col = "black", lty=2)}, main = "Joint posterior density")
```




```
### Marginal posterior densities ###
computePost = function(data, prior) {
  nWater = sum(data)
  nData = length(data)
  likelihood = dbinom(x = nWater, size = nData, prob = pGrid)
  post = likelihood * prior
  post = post / ( sum(post) * gridSize )
  return(post)
}
p1_post = computePost(data1, prior)
p2_post = computePost(data2, prior)
plot(pGrid, p1_post, type="l", col = "red", ylab = "post", lwd = 3,
     main = "Marginal posterior densities")
```

```
lines(pGrid, p2_post, col = "blue", lwd = 3)
legend("topright", legend = c("p1", "p2"), lty = c(1, 1), lwd = c(3, 3),
       col = c("red", "blue"), bty = "n")
```




```
### Mean and standard deviation of posterior marginal distributions ###
p1_alpha = aPrior + nWater_1
p1_beta = bPrior + 100 - nWater_1
p1Mean = p1_alpha / (p1_alpha + p1_beta)
p1SD = sqrt((p1_alpha*p1_beta)/((p1_alpha+p1_beta+1)*(p1_alpha+p1_beta)^2))

p2_alpha = aPrior + nWater_2
p2_beta = bPrior + 100 - nWater_2
p2Mean = p2_alpha / (p2_alpha + p2_beta)
p2SD = sqrt((p2_alpha*p2_beta)/((p2_alpha+p2_beta+1)*(p2_alpha+p2_beta)^2))

paste("p1 mean of posterior distribution", round(p1Mean,3))
```

```
## [1] "p1 mean of posterior distribution 0.473"
```

```
paste("p1 standard deviation of posterior distribution", round(p1SD,3))
```

```
## [1] "p1 standard deviation of posterior distribution 0.047"
```

```
paste("p2 mean of posterior distribution", round(p2Mean,3))
```

```
## [1] "p2 mean of posterior distribution 0.536"
```

```
paste("p2 standard deviation of posterior distribution", round(p2SD,3))
```

```
## [1] "p2 standard deviation of posterior distribution 0.047"
```

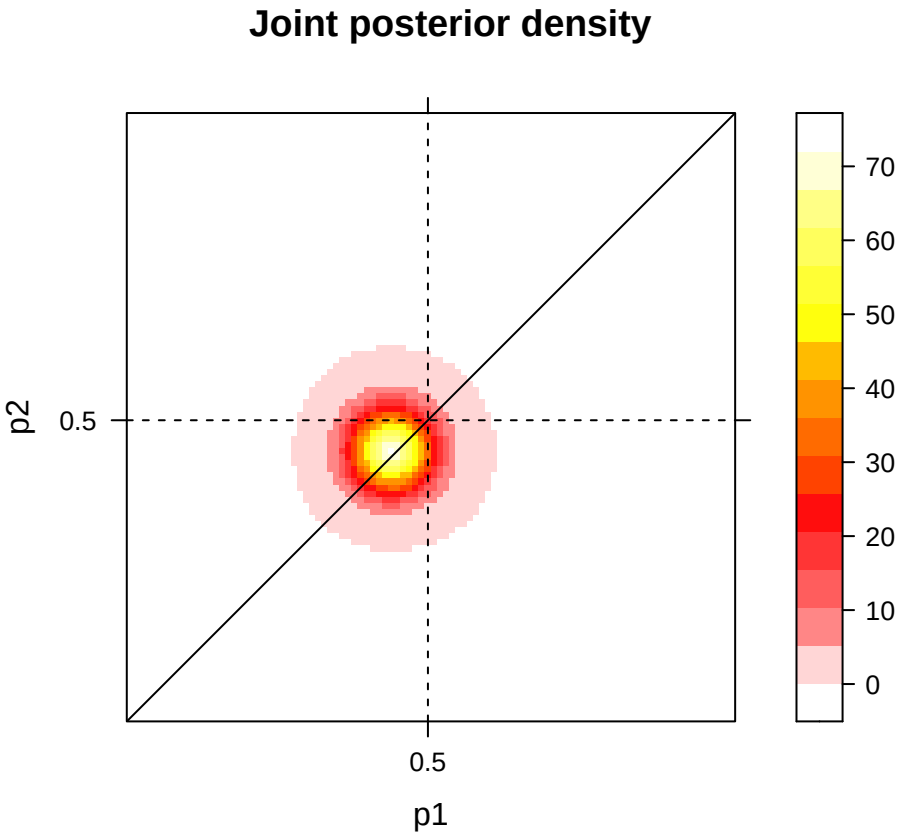
```r
### Posterior probability that p_1 < p_2 ###
sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p1 < p2 = ", round(sum,3))
```

```
## [1] "posterior prob p1 < p2 =  0.847"
```

step 2: Fit only using new data

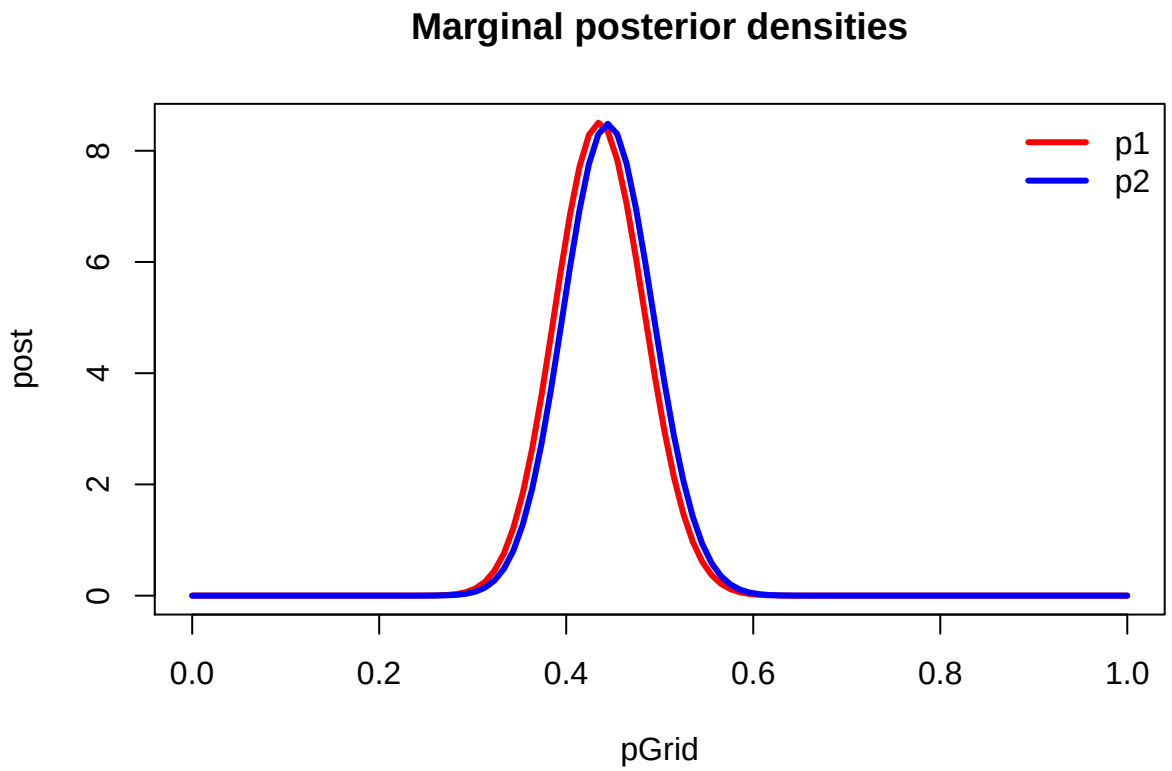
```r
### NOTE: Utilized provided code from lecture 2.2
data = read.csv(file = "~/Desktop/PS2_data.csv")
# set random seed
set.seed(123)
# define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 /nGridPoints
# prior params
aPrior = 5
bPrior = 5
# Data collected by nh
data = subset(data, tosser == "nh")
data1 = data$ball_1
data2 = data$ball_2
nWater_1 = sum(data1)
nWater_2 = sum(data2)
n1 = length(data1)
n2 = length(data2)

### Joint posterior density as heat map ###
# define prior matrix
# >> note: assume independent non-uniform priors
prior = dbeta(x = pGrid, shape1 = aPrior, shape2 = bPrior)
priorM = matrix(rep(1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
                byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    priorM[row,col] = prior[row] * prior[col]
  }
}
# compute posterior
postM = matrix(rep(-1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
               byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    p1 = pGrid[row]
    p2 = pGrid[col]
    postM[row,col] = dbinom(nWater_1,n1,p1) * dbinom(nWater_2,n2,p2) * priorM[row,col]
  }
```

```
}
postM = postM / ( sum(postM) * gridSize ^ 2 )
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(50),
          labels=c(0.5)), y=list(at=c(50), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=50, col = "black", lty=2)
            panel.abline(h=50, col = "black", lty=2)}, main = "Joint posterior density")
```




```
### Marginal posterior densities ###
computePost = function(data, prior) {
  nWater = sum(data)
  nData = length(data)
  likelihood = dbinom(x = nWater, size = nData, prob = pGrid)
  post = likelihood * prior
  post = post / ( sum(post) * gridSize )
  return(post)
}
p1_post = computePost(data1, prior)
p2_post = computePost(data2, prior)
plot(pGrid, p1_post, type="l", col = "red", ylab = "post", lwd = 3,
     main = "Marginal posterior densities")
lines(pGrid, p2_post, col = "blue", lwd = 3)
```

```
legend("topright", legend = c("p1", "p2"), lty = c(1, 1), lwd = c(3, 3),
       col = c("red", "blue"), bty = "n")
```




```
### Mean and standard deviation of posterior marginal distributions ###
p1_alpha = aPrior + nWater_1
p1_beta = bPrior + 100 - nWater_1
p1Mean = p1_alpha / (p1_alpha + p1_beta)
p1SD = sqrt((p1_alpha*p1_beta)/((p1_alpha+p1_beta+1)*(p1_alpha+p1_beta)^2))

p2_alpha = aPrior + nWater_2
p2_beta = bPrior + 100 - nWater_2
p2Mean = p2_alpha / (p2_alpha + p2_beta)
p2SD = sqrt((p2_alpha*p2_beta)/((p2_alpha+p2_beta+1)*(p2_alpha+p2_beta)^2))

paste("p1 mean of posterior distribution", round(p1Mean,3))
```

```
## [1] "p1 mean of posterior distribution 0.436"
```

```
paste("p1 standard deviation of posterior distribution", round(p1SD,3))
```

```
## [1] "p1 standard deviation of posterior distribution 0.047"
```

```
paste("p2 mean of posterior distribution", round(p2Mean,3))
```

```
## [1] "p2 mean of posterior distribution 0.445"
```

```
paste("p2 standard deviation of posterior distribution", round(p2SD,3))
```

```
## [1] "p2 standard deviation of posterior distribution 0.047"
```

```
### Posterior probability that p_1 < p_2 ###
sum = 0
```

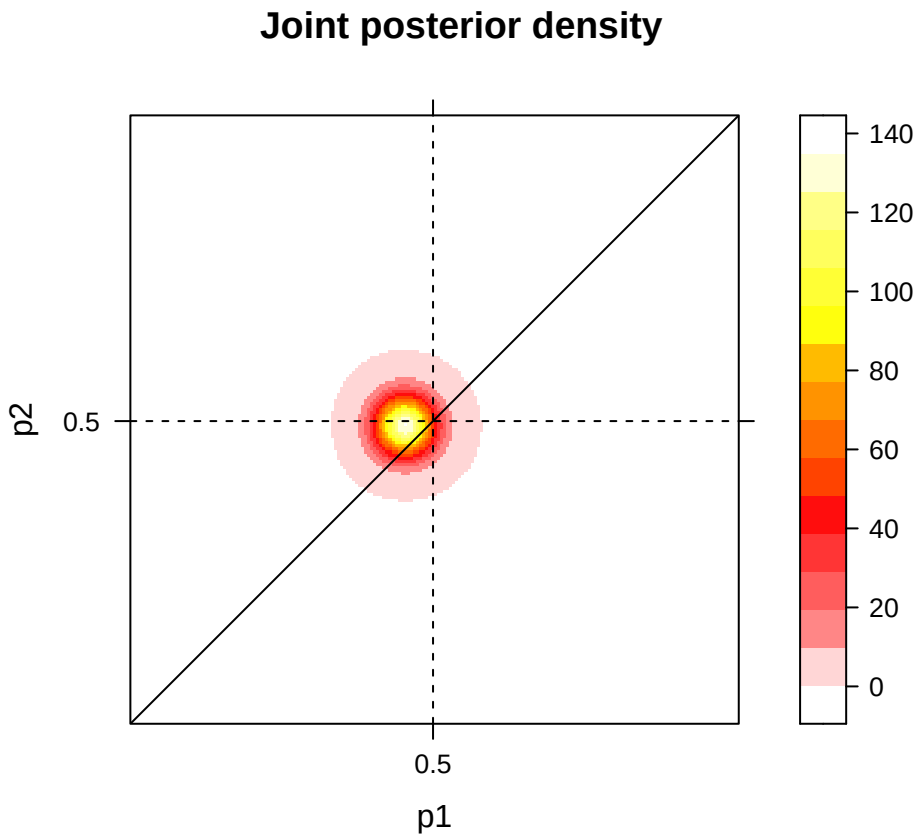
```
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p1 < p2 = ", round(sum,3))
```

```
## [1] "posterior prob p1 < p2 =  0.584"
```

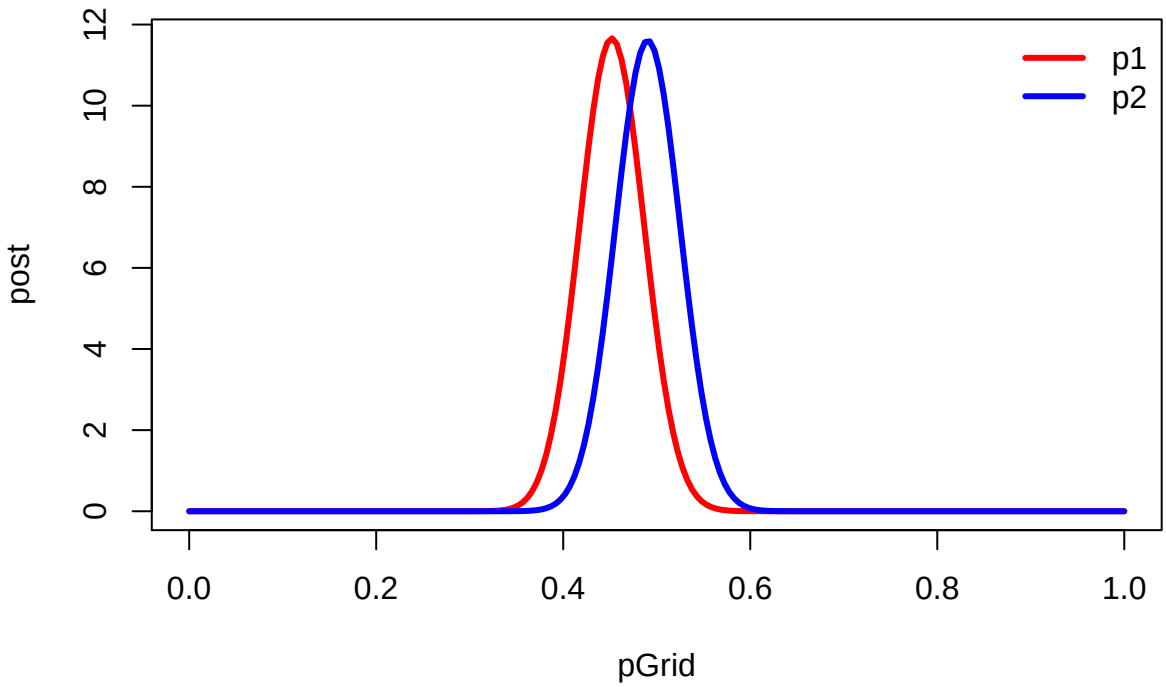
Step 3: Fit using all the data

```
### NOTE: Utilized provided code from lecture 2.2
data = read.csv(file = "~/Desktop/PS2_data.csv")
# set random seed
set.seed(123)
# define grid size
nGridPoints = 200
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 /nGridPoints
# prior params
aPrior = 5
bPrior = 5
# Data collected by ar and nh
data1 = data$ball_1
data2 = data$ball_2
nWater_1 = sum(data1)
nWater_2 = sum(data2)
n1 = length(data1)
n2 = length(data2)

### Joint posterior density as heat map ###
# define prior matrix
# >> note: assume independent non-uniform priors
prior = dbeta(x = pGrid, shape1 = aPrior, shape2 = bPrior)
priorM = matrix(rep(1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
                byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    priorM[row,col] = prior[row] * prior[col]
  }
}
# compute posterior
postM = matrix(rep(-1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
               byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    p1 = pGrid[row]
    p2 = pGrid[col]
    postM[row,col] = dbinom(nWater_1,n1,p1) * dbinom(nWater_2,n2,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )
library(lattice)
```

```
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(100),
          labels=c(0.5)), y=list(at=c(100), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=100, col = "black", lty=2)
            panel.abline(h=100, col = "black", lty=2)}, main = "Joint posterior density")
```




```
### Marginal posterior densities ###
computePost = function(data, prior) {
  nWater = sum(data)
  nData = length(data)
  likelihood = dbinom(x = nWater, size = nData, prob = pGrid)
  post = likelihood * prior
  post = post / ( sum(post) * gridSize )
  return(post)
}
p1_post = computePost(data1, prior)
p2_post = computePost(data2, prior)
plot(pGrid, p1_post, type="l", col = "red", ylab = "post", lwd = 3,
     main = "Marginal posterior densities")
lines(pGrid, p2_post, col = "blue", lwd = 3)
legend("topright", legend = c("p1", "p2"), lty = c(1, 1), lwd = c(3, 3),
       col = c("red", "blue"), bty = "n")
```




```
### Mean and standard deviation of posterior marginal distributions ###
p1_alpha = aPrior + nWater_1
p1_beta = bPrior + 100 - nWater_1
p1Mean = p1_alpha / (p1_alpha + p1_beta)
p1SD = sqrt((p1_alpha*p1_beta)/((p1_alpha+p1_beta+1)*(p1_alpha+p1_beta)^2))

p2_alpha = aPrior + nWater_2
p2_beta = bPrior + 100 - nWater_2
p2Mean = p2_alpha / (p2_alpha + p2_beta)
p2SD = sqrt((p2_alpha*p2_beta)/((p2_alpha+p2_beta+1)*(p2_alpha+p2_beta)^2))

paste("p1 mean of posterior distribution", round(p1Mean,3))
```

```
## [1] "p1 mean of posterior distribution 0.864"
```

```
paste("p1 standard deviation of posterior distribution", round(p1SD,3))
```

```
## [1] "p1 standard deviation of posterior distribution 0.033"
```

```
paste("p2 mean of posterior distribution", round(p2Mean,3))
```

```
## [1] "p2 mean of posterior distribution 0.936"
```

```
paste("p2 standard deviation of posterior distribution", round(p2SD,3))
```

```
## [1] "p2 standard deviation of posterior distribution 0.023"
```

```
### Posterior probability that p_1 < p_2 ###
sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
```

```
  }
}
paste("posterior prob p1 < p2 = ", round(sum,3))
```

```
## [1] "posterior prob p1 < p2 =  0.798"
```

PART 2

Step 1

Functions

```
### Function that returns bivariate normal density ###
bvnd = function(p1, p2, u1, u2, s1, s2, rho) {
   z = (p1-u1)^2/(s1^2) - (2*rho*(p1-u1)*(p2-u2))/(s1*s2) + (p2-u2)^2/(s2^2)
   part1 = 1/(2*pi*s1*s2*sqrt(1-rho^2))
   part2 = exp(-(z)/(2*(1-rho^2)))
   return(part1*part2)
}

### Function that returns prior matrix based on associated bivariate norm. dist. ###
bvnd_priorM = function(u1, u2, s1, s2, rho) {
  priorM = matrix(rep(1, 100^2), nrow = 100, ncol = 100, byrow=TRUE)
  pGrid = seq(from = 0, to = 1, length.out = 100)
  for (row in 1:100) {
    for (col in 1:100) {
      priorM[row, col] = bvnd(pGrid[row], pGrid[col], u1, u2, s1, s2, rho)
    }
  }
  priorM = priorM / ( sum(priorM) * (1/100) ^ 2 )
  return(priorM)
}
sum(bvnd_priorM(.5,.5,1,1,0)*(1/100)^2)
```

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

```
sum(bvnd_priorM(.5,.5,1,1,0.25)*(1/100)^2)
```

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

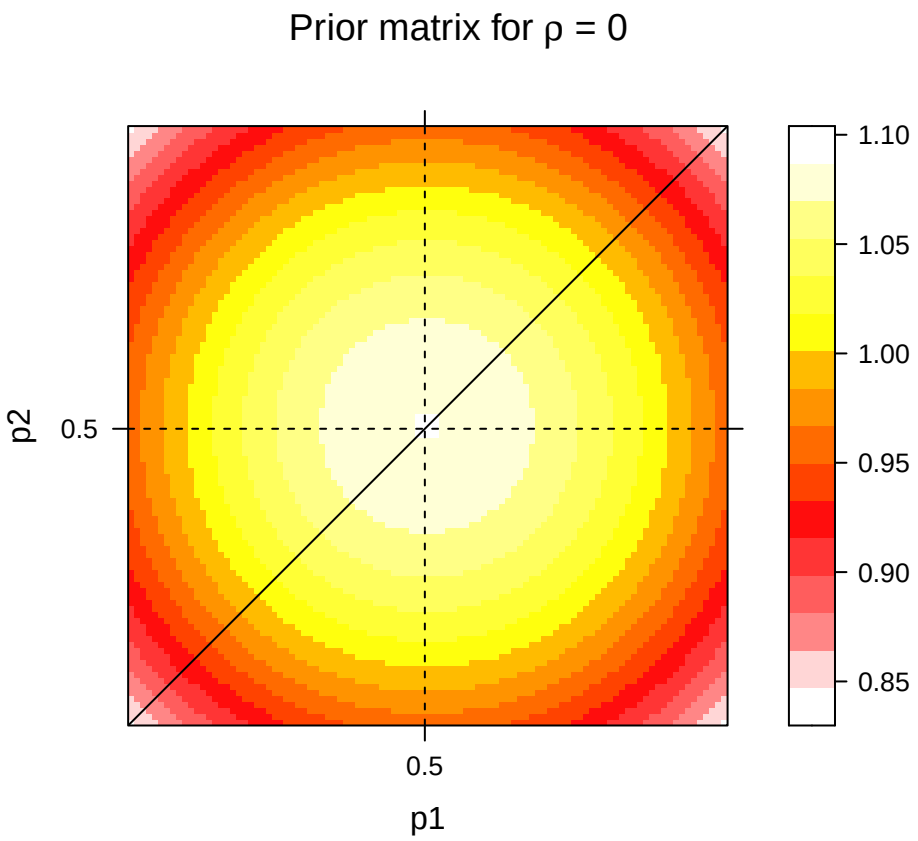
```
sum(bvnd_priorM(.5,.5,1,1,0.5)*(1/100)^2)
```

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

A

```
### Plot of prior matrix as heat map (rho = 0) ###
u1 = 0.5
u2 = 0.5
s1 = 1
s2 = 1
rho = 0
priorM = bvnd_priorM(u1, u2, s1, s2, rho)

library(lattice)
```

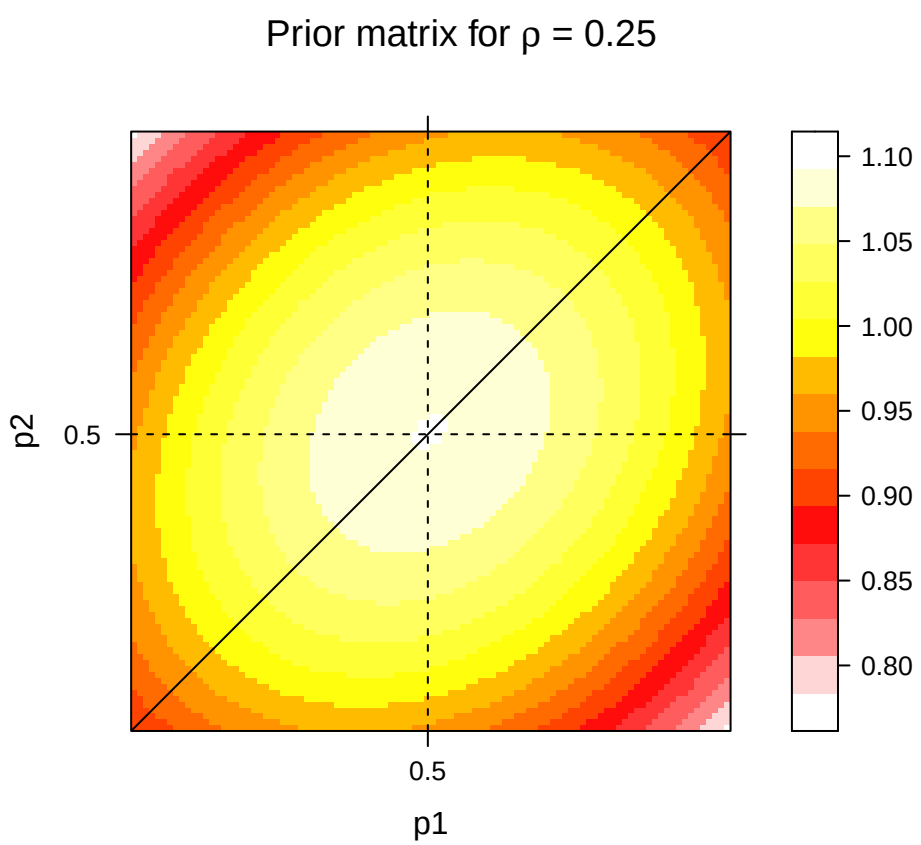
```
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(priorM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(50),
          labels=c(0.5)), y=list(at=c(50), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=50, col = "black", lty=2)
            panel.abline(h=50, col = "black", lty=2)},
          main = expression(paste("Prior matrix for ", rho, " = ", 0)))
```



B

```
### Plot of prior matrix as heat map (rho = 0.25) ###
u1 = 0.5
u2 = 0.5
s1 = 1
s2 = 1
rho = 0.25
priorM = bvnd_priorM(u1, u2, s1, s2, rho)

library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(priorM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(50),
          labels=c(0.5)), y=list(at=c(50), labels=c(0.5))),
```

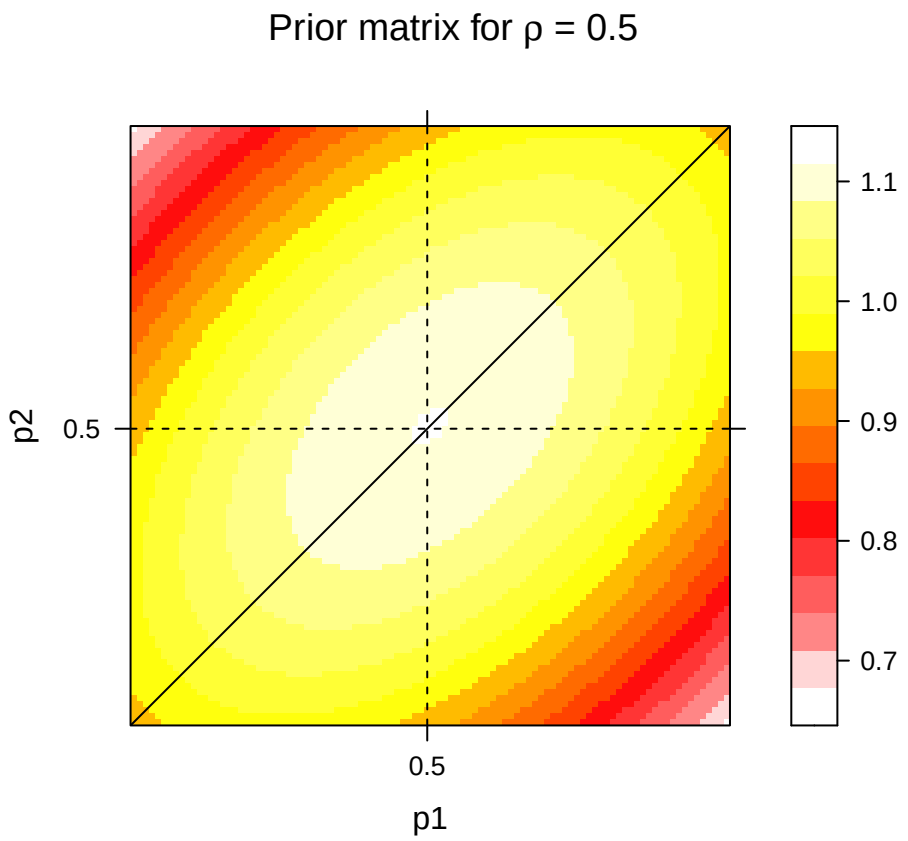
```
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=50, col = "black", lty=2)
            panel.abline(h=50, col = "black", lty=2)},
          main = expression(paste("Prior matrix for ", rho, " = ", 0.25)))
```



C

```
### Plot of prior matrix as heat map (rho = 0.5) ###
u1 = 0.5
u2 = 0.5
s1 = 1
s2 = 1
rho = 0.5
priorM = bvnd_priorM(u1, u2, s1, s2, rho)

library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(priorM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(50),
          labels=c(0.5)), y=list(at=c(50), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=50, col = "black", lty=2)
```

```
          panel.abline(h=50, col = "black", lty=2)},
        main = expression(paste("Prior matrix for ", rho, " = ", 0.5)))
```

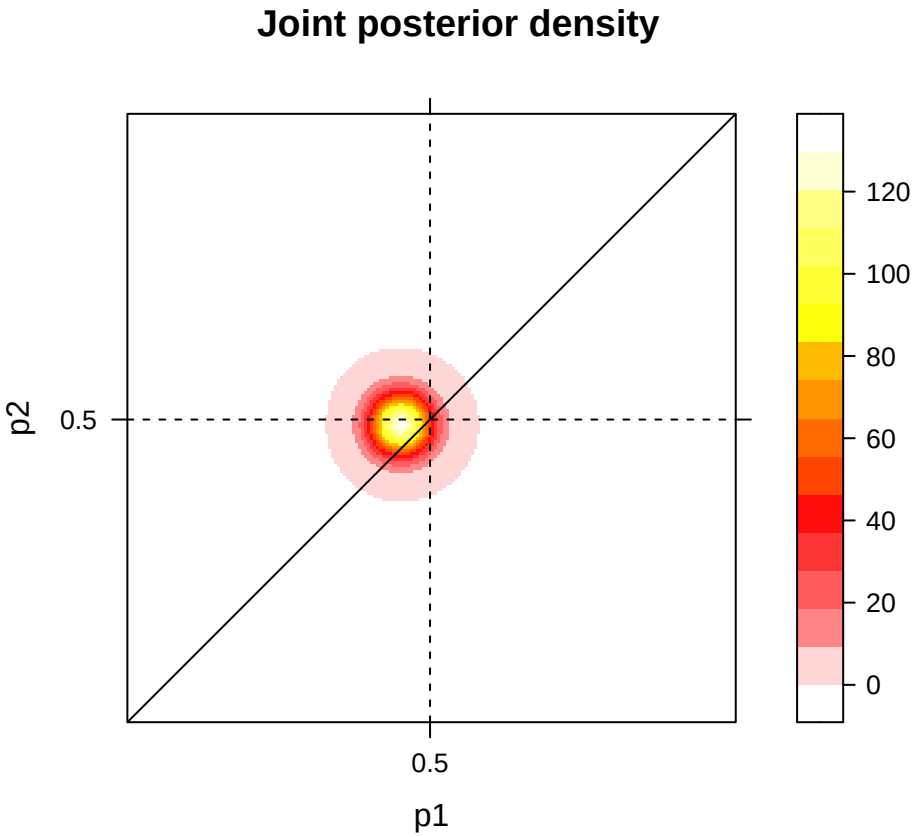


Step 2

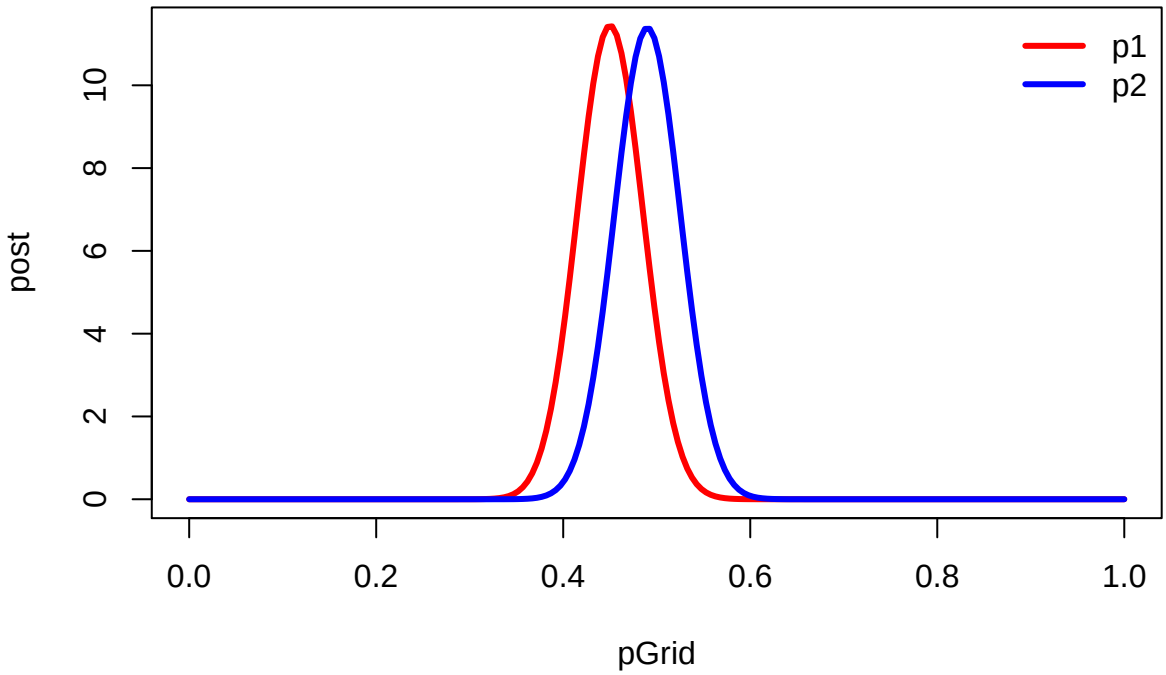
A

```
### NOTE: Utilized provided code from lecture 2.2
data = read.csv(file = "~/Desktop/PS2_data.csv")
# set random seed
set.seed(123)
# define grid size
nGridPoints = 200
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 /nGridPoints
# using all available data
data1 = data$ball_1
data2 = data$ball_2
nWater_1 = sum(data1)
nWater_2 = sum(data2)
n1 = length(data1)
n2 = length(data2)

### Joint posterior density as heat map ###
# define prior matrix
# >> note: assume independent non-uniform priors
```

```
### Function that returns prior matrix based on associated bivariate norm. dist. ###
# redefining with 200 points
bvnd_priorM = function(u1, u2, s1, s2, rho) {
  priorM = matrix(rep(1, 100^2), nrow = 200, ncol = 200, byrow=TRUE)
  pGrid = seq(from = 0, to = 1, length.out = 200)
  for (row in 1:200) {
    for (col in 1:200) {
      priorM[row, col] = bvnd(pGrid[row], pGrid[col], u1, u2, s1, s2, rho)
    }
  }
  priorM = priorM / ( sum(priorM) * (1/200) ^ 2 )
  return(priorM)
}
u1 = 0.5
u2 = 0.5
s1 = 1
s2 = 1
rho = 0
priorM = bvnd_priorM(u1, u2, s1, s2, rho)

# compute posterior
postM = matrix(rep(-1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
               byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    p1 = pGrid[row]
    p2 = pGrid[col]
    postM[row,col] = dbinom(nWater_1,n1,p1) * dbinom(nWater_2,n2,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(100),
          labels=c(0.5)), y=list(at=c(100), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=100, col = "black", lty=2)
            panel.abline(h=100, col = "black", lty=2)}, main = "Joint posterior density")
```




```
### Marginal posterior densities ###
computePost = function(data, prior) {
  nWater = sum(data)
  nData = length(data)
  likelihood = dbinom(x = nWater, size = nData, prob = pGrid)
  post = likelihood * prior
  post = post / ( sum(post) * gridSize )
  return(post)
}
p1_post = computePost(data1, rowSums(priorM))
p2_post = computePost(data2, colSums(priorM))
plot(pGrid, p1_post, type="l", col = "red", ylab = "post", lwd = 3,
     main = "Marginal posterior densities")
lines(pGrid, p2_post, col = "blue", lwd = 3)
legend("topright", legend = c("p1", "p2"), lty = c(1, 1), lwd = c(3, 3),
       col = c("red", "blue"), bty = "n")
```




```
### Mean and standard deviation of posterior marginal distributions ###
p1Mean = sum(pGrid*p1_post*gridSize)
p1SD = sqrt(sum(p1_post*gridSize*(pGrid-p1Mean)^2))
p2Mean = sum(pGrid*p2_post*gridSize)
p2SD = sqrt(sum(p2_post*gridSize*(pGrid-p2Mean)^2))

paste("p1 mean of posterior distribution", round(p1Mean,3))
```

```
## [1] "p1 mean of posterior distribution 0.451"
```

```
paste("p1 standard deviation of posterior distribution", round(p1SD,3))
```

```
## [1] "p1 standard deviation of posterior distribution 0.035"
```

```
paste("p2 mean of posterior distribution", round(p2Mean,3))
```

```
## [1] "p2 mean of posterior distribution 0.49"
```

```
paste("p2 standard deviation of posterior distribution", round(p2SD,3))
```

```
## [1] "p2 standard deviation of posterior distribution 0.035"
```

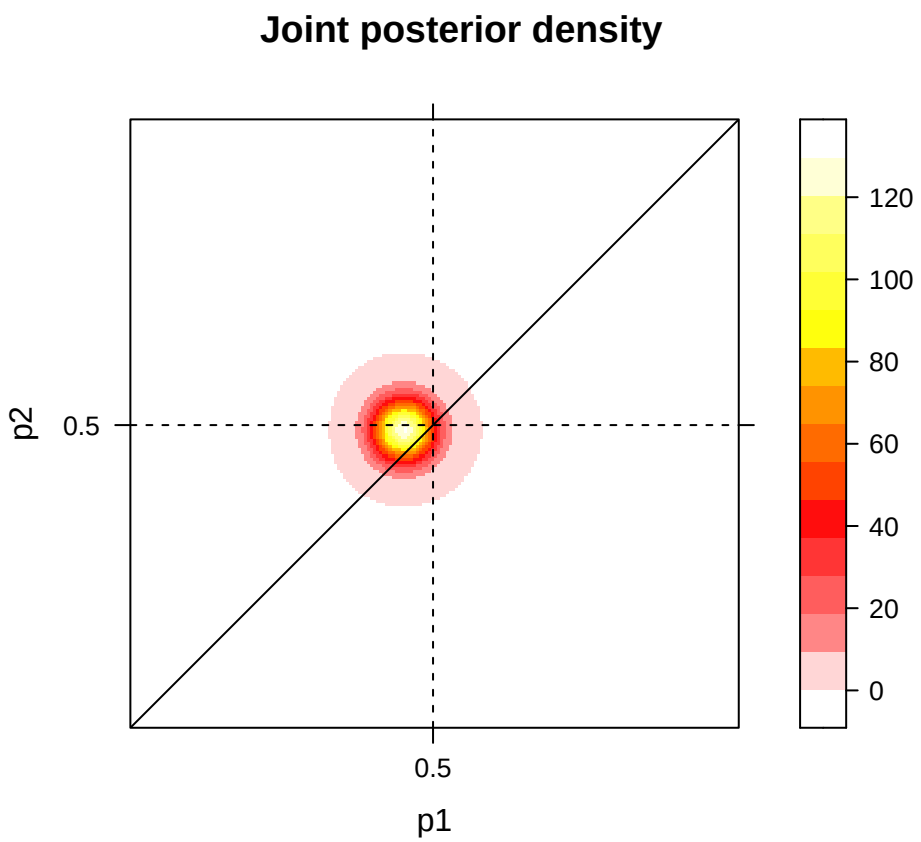
```
### Posterior probability that p_1 < p_2 ###
sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p1 < p2 = ", round(sum,3))
```

```
## [1] "posterior prob p1 < p2 =  0.802"
```

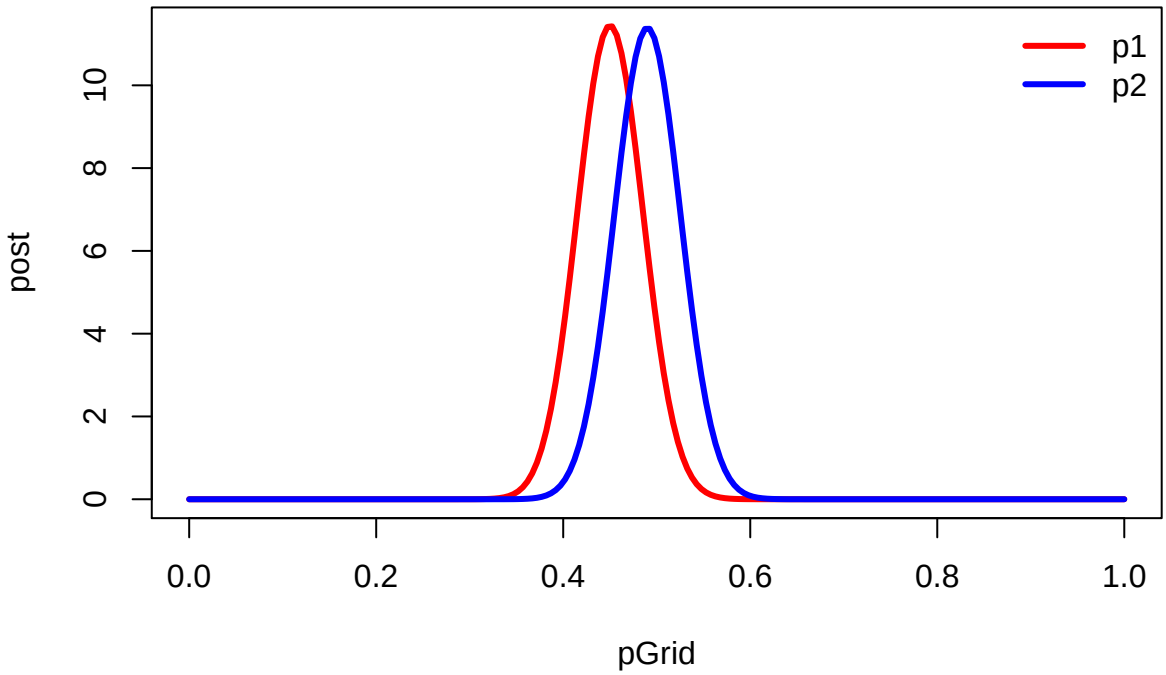
B

```
### NOTE: Utilized provided code from lecture 2.2
data = read.csv(file = "~/Desktop/PS2_data.csv")
# set random seed
set.seed(123)
# define grid size
nGridPoints = 200
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 /nGridPoints
# using all available data
data1 = data$ball_1
data2 = data$ball_2
nWater_1 = sum(data1)
nWater_2 = sum(data2)
n1 = length(data1)
n2 = length(data2)

### Joint posterior density as heat map ###
# define prior matrix
# >> note: assume independent non-uniform priors

### Function that returns prior matrix based on associated bivariate norm. dist. ###
# redefining with 200 points
bvnd_priorM = function(u1, u2, s1, s2, rho) {
  priorM = matrix(rep(1, 100^2), nrow = 200, ncol = 200, byrow=TRUE)
  pGrid = seq(from = 0, to = 1, length.out = 200)
  for (row in 1:200) {
    for (col in 1:200) {
      priorM[row, col] = bvnd(pGrid[row], pGrid[col], u1, u2, s1, s2, rho)
    }
  }
  priorM = priorM / ( sum(priorM) * (1/200) ^ 2 )
  return(priorM)
}
u1 = 0.5
u2 = 0.5
s1 = 1
s2 = 1
rho = 0.25
priorM = bvnd_priorM(u1, u2, s1, s2, rho)

# compute posterior
postM = matrix(rep(-1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
               byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    p1 = pGrid[row]
    p2 = pGrid[col]
    postM[row,col] = dbinom(nWater_1,n1,p1) * dbinom(nWater_2,n2,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )
library(lattice)
```

```
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(100),
          labels=c(0.5)), y=list(at=c(100), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=100, col = "black", lty=2)
            panel.abline(h=100, col = "black", lty=2)}, main = "Joint posterior density")
```




```
### Marginal posterior densities ###
computePost = function(data, prior) {
  nWater = sum(data)
  nData = length(data)
  likelihood = dbinom(x = nWater, size = nData, prob = pGrid)
  post = likelihood * prior
  post = post / ( sum(post) * gridSize )
  return(post)
}
p1_post = computePost(data1, rowSums(priorM))
p2_post = computePost(data2, colSums(priorM))
plot(pGrid, p1_post, type="l", col = "red", ylab = "post", lwd = 3,
     main = "Marginal posterior densities")
lines(pGrid, p2_post, col = "blue", lwd = 3)
legend("topright", legend = c("p1", "p2"), lty = c(1, 1), lwd = c(3, 3),
       col = c("red", "blue"), bty = "n")
```




```
### Mean and standard deviation of posterior marginal distributions ###
p1Mean = sum(pGrid*p1_post*gridSize)
p1SD = sqrt(sum(p1_post*gridSize*(pGrid-p1Mean)^2))
p2Mean = sum(pGrid*p2_post*gridSize)
p2SD = sqrt(sum(p2_post*gridSize*(pGrid-p2Mean)^2))

paste("p1 mean of posterior distribution", round(p1Mean,3))
```

```
## [1] "p1 mean of posterior distribution 0.451"
```

```
paste("p1 standard deviation of posterior distribution", round(p1SD,3))
```

```
## [1] "p1 standard deviation of posterior distribution 0.035"
```

```
paste("p2 mean of posterior distribution", round(p2Mean,3))
```

```
## [1] "p2 mean of posterior distribution 0.49"
```

```
paste("p2 standard deviation of posterior distribution", round(p2SD,3))
```

```
## [1] "p2 standard deviation of posterior distribution 0.035"
```

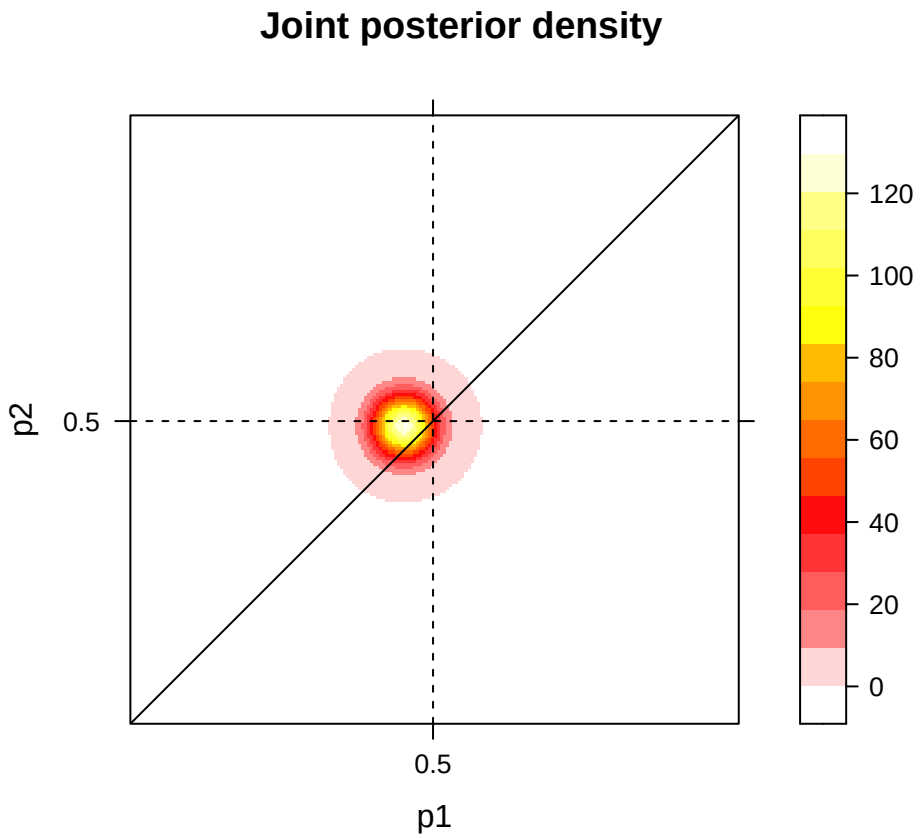
```
### Posterior probability that p_1 < p_2 ###
sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p1 < p2 = ", round(sum,3))
```

```
## [1] "posterior prob p1 < p2 =  0.802"
```

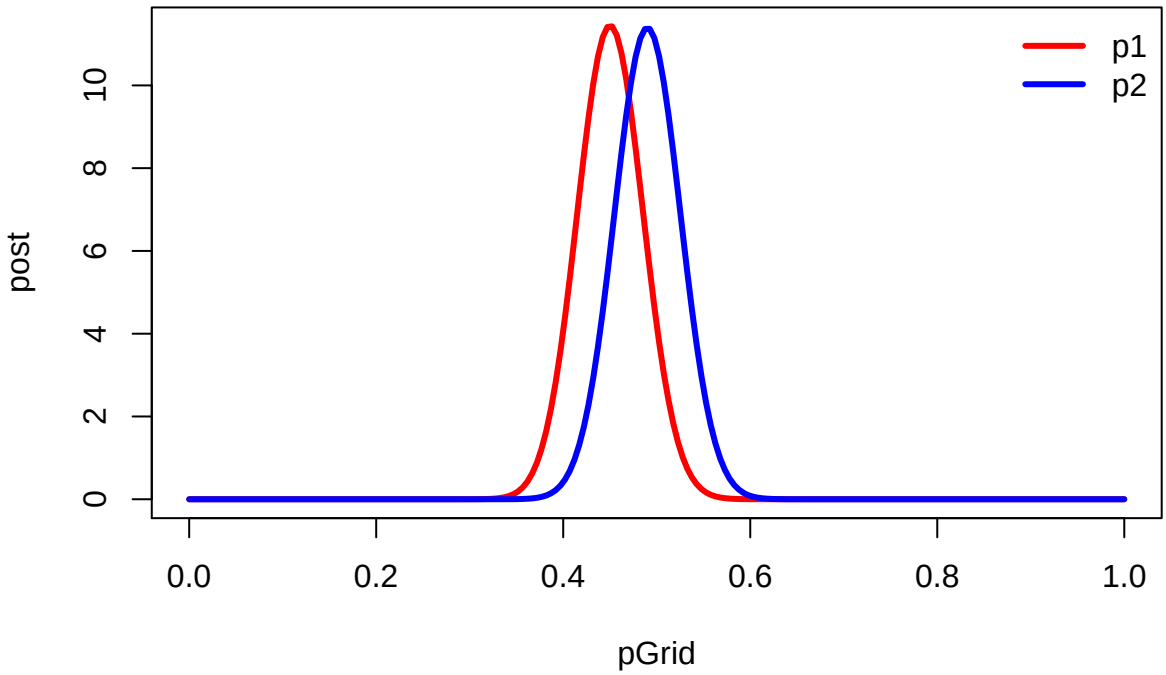
C

```
### NOTE: Utilized provided code from lecture 2.2
data = read.csv(file = "~/Desktop/PS2_data.csv")
# set random seed
set.seed(123)
# define grid size
nGridPoints = 200
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 /nGridPoints
# using all available data
data1 = data$ball_1
data2 = data$ball_2
nWater_1 = sum(data1)
nWater_2 = sum(data2)
n1 = length(data1)
n2 = length(data2)

### Joint posterior density as heat map ###
# define prior matrix
# >> note: assume independent non-uniform priors

### Function that returns prior matrix based on associated bivariate norm. dist. ###
# redefining with 200 points
bvnd_priorM = function(u1, u2, s1, s2, rho) {
  priorM = matrix(rep(1, 100^2), nrow = 200, ncol = 200, byrow=TRUE)
  pGrid = seq(from = 0, to = 1, length.out = 200)
  for (row in 1:200) {
    for (col in 1:200) {
      priorM[row, col] = bvnd(pGrid[row], pGrid[col], u1, u2, s1, s2, rho)
    }
  }
  priorM = priorM / ( sum(priorM) * (1/200) ^ 2 )
  return(priorM)
}
u1 = 0.5
u2 = 0.5
s1 = 1
s2 = 1
rho = 0.5
priorM = bvnd_priorM(u1, u2, s1, s2, rho)

# compute posterior
postM = matrix(rep(-1, nGridPoints ^ 2 ), nrow = nGridPoints, ncol = nGridPoints,
               byrow = TRUE)
for (row in 1:nGridPoints) {
  for (col in 1:nGridPoints) {
    p1 = pGrid[row]
    p2 = pGrid[col]
    postM[row,col] = dbinom(nWater_1,n1,p1) * dbinom(nWater_2,n2,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )
library(lattice)
```

```
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "p1", ylab = "p2", scales=list(x=list(at=c(100),
          labels=c(0.5)), y=list(at=c(100), labels=c(0.5))),
          panel = function(...){
            panel.levelplot(...)
            panel.abline(0,1, col = "black")
            panel.abline(v=100, col = "black", lty=2)
            panel.abline(h=100, col = "black", lty=2)}, main = "Joint posterior density")
```




```
### Marginal posterior densities ###
computePost = function(data, prior) {
  nWater = sum(data)
  nData = length(data)
  likelihood = dbinom(x = nWater, size = nData, prob = pGrid)
  post = likelihood * prior
  post = post / ( sum(post) * gridSize )
  return(post)
}
p1_post = computePost(data1, rowSums(priorM))
p2_post = computePost(data2, colSums(priorM))
plot(pGrid, p1_post, type="l", col = "red", ylab = "post", lwd = 3,
     main = "Marginal posterior densities")
lines(pGrid, p2_post, col = "blue", lwd = 3)
legend("topright", legend = c("p1", "p2"), lty = c(1, 1), lwd = c(3, 3),
       col = c("red", "blue"), bty = "n")
```




```
### Mean and standard deviation of posterior marginal distributions ###
p1Mean = sum(pGrid*p1_post*gridSize)
p1SD = sqrt(sum(p1_post*gridSize*(pGrid-p1Mean)^2))
p2Mean = sum(pGrid*p2_post*gridSize)
p2SD = sqrt(sum(p2_post*gridSize*(pGrid-p2Mean)^2))

paste("p1 mean of posterior distribution", round(p1Mean,3))
```

```
## [1] "p1 mean of posterior distribution 0.451"
```

```
paste("p1 standard deviation of posterior distribution", round(p1SD,3))
```

```
## [1] "p1 standard deviation of posterior distribution 0.035"
```

```
paste("p2 mean of posterior distribution", round(p2Mean,3))
```

```
## [1] "p2 mean of posterior distribution 0.49"
```

```
paste("p2 standard deviation of posterior distribution", round(p2SD,3))
```

```
## [1] "p2 standard deviation of posterior distribution 0.035"
```

```
### Posterior probability that p_1 < p_2 ###
sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p1 < p2 = ", round(sum,3))
```

```
## [1] "posterior prob p1 < p2 =  0.802"
```

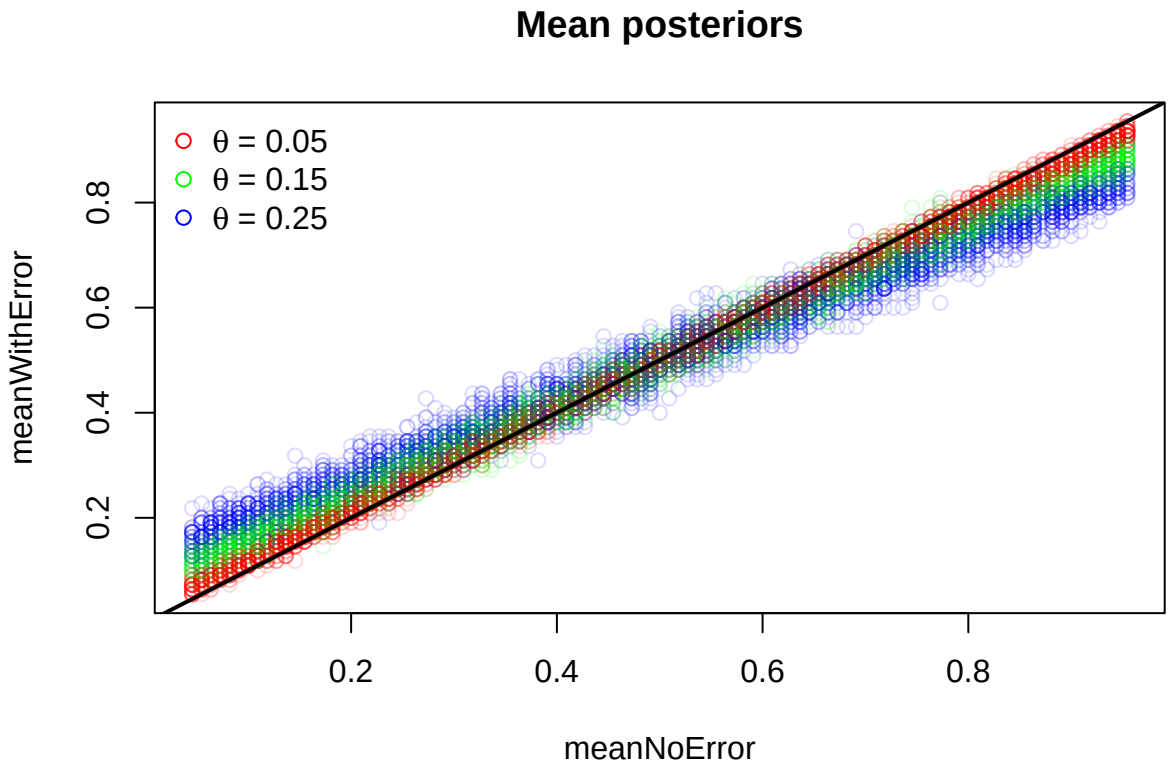
Step 3

The results suggest that there should not be much concern regarding uncorrelated priors given this amount of available data because we ended up with very similar results compared to part 1. Also, the results were consistent for varying rho values.

PART 3

Step 1

```
set.seed(123)
nSim = 10000
postNoError = rep(-1, nSim)
postWithError = rep(-1, nSim)
meanNoError = rep(-1, nSim)
meanWithError = rep(-1, nSim)
thetaVar = rep(-1, nSim)
pGrid = seq(from = 0, to = 1, length.out = 100)
gridSize = 1 / 100
prior = dbeta(x = pGrid, shape1 = 5, shape2 = 5)

computePost = function(data, prior) {
  nWater = sum(data)
  nData = length(data)
  likelihood = dbinom(x = nWater, size = nData, prob = pGrid)
  post = likelihood * prior
  post = post / ( sum(post) * gridSize )
  return(post)
}

for (i in 1:nSim) {
  pTrue = runif(1)
  thetaTrue = sample(c(0.05, 0.15, 0.25),1)
  dataNoError = rbinom(n = 100, size = 1, prob = pTrue)
  dataWithError = dataNoError
  randomIndices = sample.int(100, 100*thetaTrue)
  for (j in 1:length(randomIndices)) {
    dataWithError[randomIndices[j]] = rbinom(1, 1, .5)
  }
  meanNoError[i] = sum(pGrid * computePost(dataNoError, prior) * gridSize)
  meanWithError[i] = sum(pGrid * computePost(dataWithError, prior) * gridSize)
  thetaVar[i] = thetaTrue
}
colors = c(rgb(red=1.0, green=0.0, blue=0.0, alpha=0.15),
           rgb(red=0.0, green=1.0, blue=0.0, alpha=0.15),
           rgb(red=0.0, green=0.0, blue=1.0, alpha=0.15))

plot(meanNoError, meanWithError, col=colors[factor(thetaVar)], main = "Mean posteriors")
abline(a = 0, b = 1, lwd = 2)
legend("topleft", legend = c(expression(paste(theta, " = ", 0.05)),
                             expression(paste(theta, " = ", 0.15)),
                             expression(paste(theta, " = ", 0.25))),
       pch = c(1, 1, 1), col = c("red", "green", "blue"), bty = "n")
```



Step 2

Considering the resulting pattern, we have that the 45 degree line represents the case when the means are equivalent. Therefore, the distribution of colored points showcases the difference between meanNoError and meanWithError. We see that around (0.5,0.5) there is clustering due to the distributions of pTrue (Unif(0,1)) and prior(p) (Beta(5,5)). Furthermore, due to varying values of thetaTrue, we see the effects of measurement error such that with a greater thetaTrue value (blue points), there is a greater difference between meanNoError and meanWithError such that meanWithError is closer to 0.5. So, generally, the mean trends closer to 0.5 with the larger the measurement error. Also, if thetaTrue = 1, then meanWithError trends even closer to 0.5 (basically between 0.4 and 0.6).