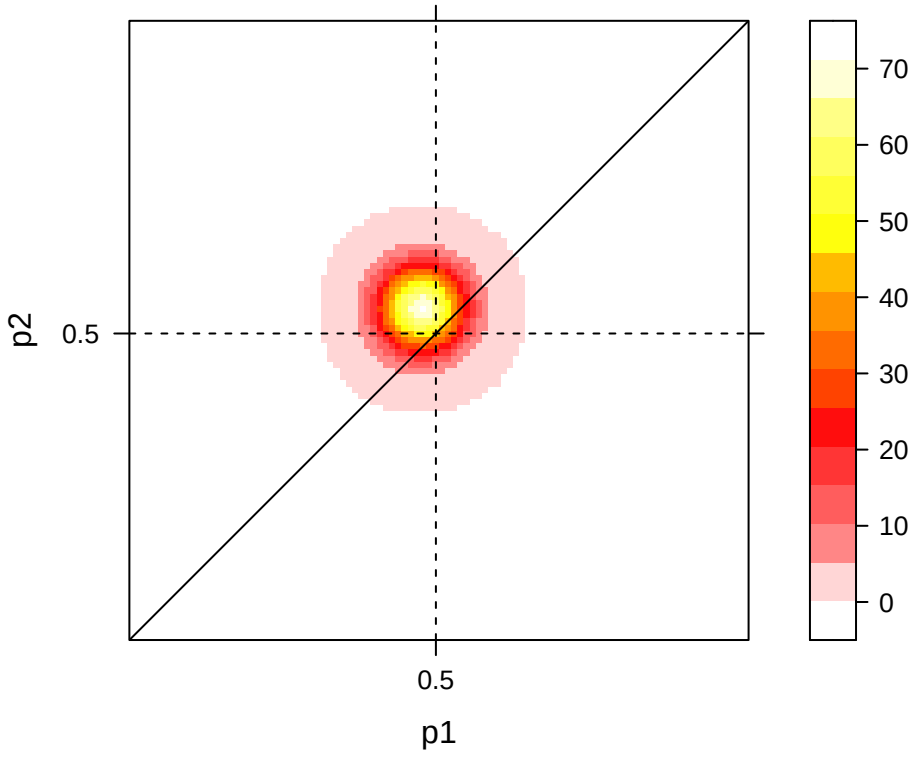
Ec/ACM/CS 112. Problem set 2. Solutions

PART 1

step 1: Fit using only old data

```
# clear environment
rm(list = ls())

#define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 / nGridPoints

# prior params
aPrior = 5
bPrior = 5

data = read.csv("~/Desktop/PS2_data.csv")
data1_ar = data$ball_1[1:100]
data2_ar = data$ball_2[1:100]
nWater_1_ar = sum(data1_ar)
nWater_2_ar = sum(data2_ar)
n1_ar = length(data1_ar)
n2_ar = length(data2_ar)

# 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_ar,n1_ar,p1) * dbinom(nWater_2_ar,n2_ar,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )

# Plot heat map

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)})
```

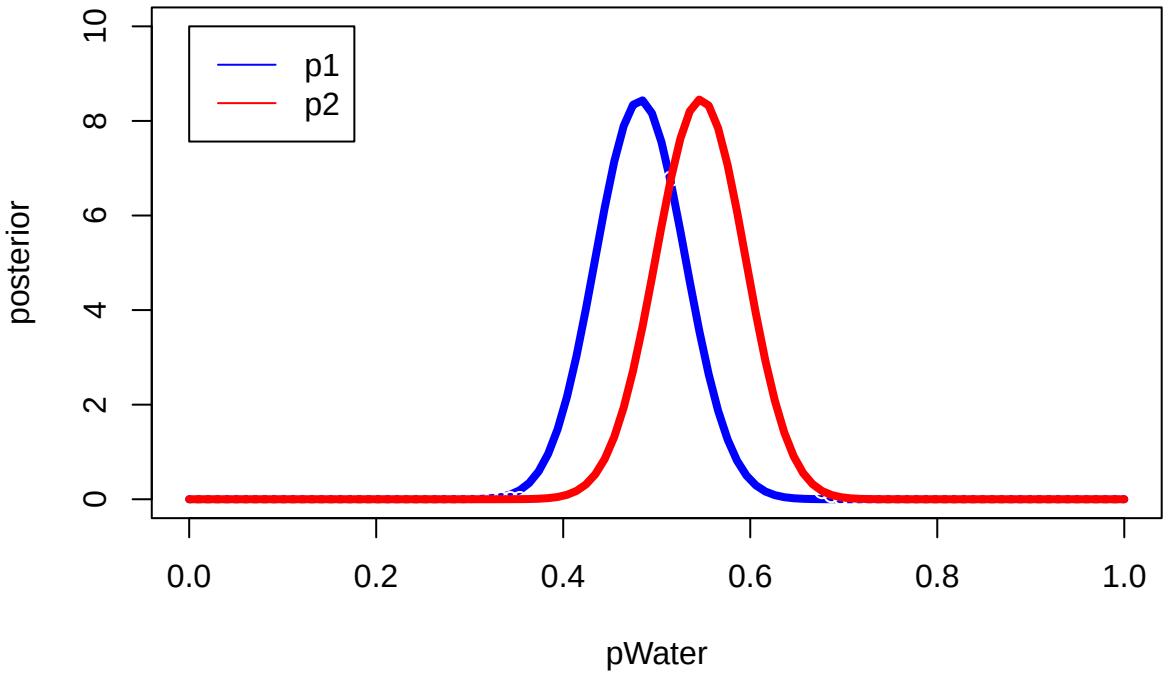



```
# Compute Probability of p2>p1

sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
```

```r
paste("posterior prob p2 > p1 = ", sum)
```

```
## [1] "posterior prob p2 > p1 =  0.847112714105597"
```

```r
##########
# Compute marginal posterior densities
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)

marg_post_1 <- function(p1){
  for (i in 1:nGridPoints){
    P <- sum(postM[floor(p1 * 100), ])* 1/100
  }
  P
}
marg_post_2 <- function(p2){
  for (i in 1:nGridPoints){
    P <- sum(postM[ , floor(p2 * 100)])* 1/100
  }
  P
}

marg_post_1_grid <- pGrid
marg_post_2_grid <- pGrid

for (i in 1:nGridPoints){
  marg_post_1_grid[i] <- marg_post_1(p1 = pGrid[i])
}
marg_post_1_grid <- marg_post_1_grid / (sum(marg_post_1_grid)* (1/nGridPoints))
for (i in 1:nGridPoints){
  marg_post_2_grid[i] <- marg_post_2(p2 = pGrid[i])
}
marg_post_2_grid <- marg_post_2_grid / (sum(marg_post_2_grid) * (1/nGridPoints))

# Plot marginal distributions
plot(pGrid, marg_post_1_grid, type="l", lwd=4,
     xlab = "pWater", ylab = "posterior",
     ylim = c(0,10), main = "Marginal Posterior Distributions", col="blue")
points(pGrid, marg_post_2_grid, ylim = c(0,10), col="white")
lines(pGrid, marg_post_2_grid,  lwd=4, ylim = c(0,10), col="red")
legend(0,10, legend = c("p1", "p2"),
       col = c("blue","red"), lty=c(1,1) )
```




```
##########

# Compute means and standard deviations

mean_1_ar <- 0
mean_2_ar <- 0
i2_1_ar <- 0
i2_2_ar <- 0
for (i in 1:nGridPoints) {
  mean_1_ar <- mean_1_ar + marg_post_1_grid[i] * i * (1/nGridPoints)^2
  i2_1_ar <- i2_1_ar + marg_post_1_grid[i] * i^2 * (1/nGridPoints)^3
  mean_2_ar <- mean_2_ar + marg_post_2_grid[i] * i * (1/nGridPoints)^2
  i2_2_ar <- i2_2_ar + marg_post_2_grid[i] * i^2 * (1/nGridPoints)^3
}

sd_1_ar <- sqrt(i2_1_ar - mean_1_ar^2)
sd_2_ar <- sqrt(i2_2_ar - mean_2_ar^2)
paste("mean of p1", mean_1_ar)
```

```
## [1] "mean of p1 0.488"
```

```
paste("mean of p2", mean_2_ar)
```

```
## [1] "mean of p2 0.551"
```

```
paste("sd of p1", sd_1_ar)
```

```
## [1] "sd of p1 0.0469133761827552"
```

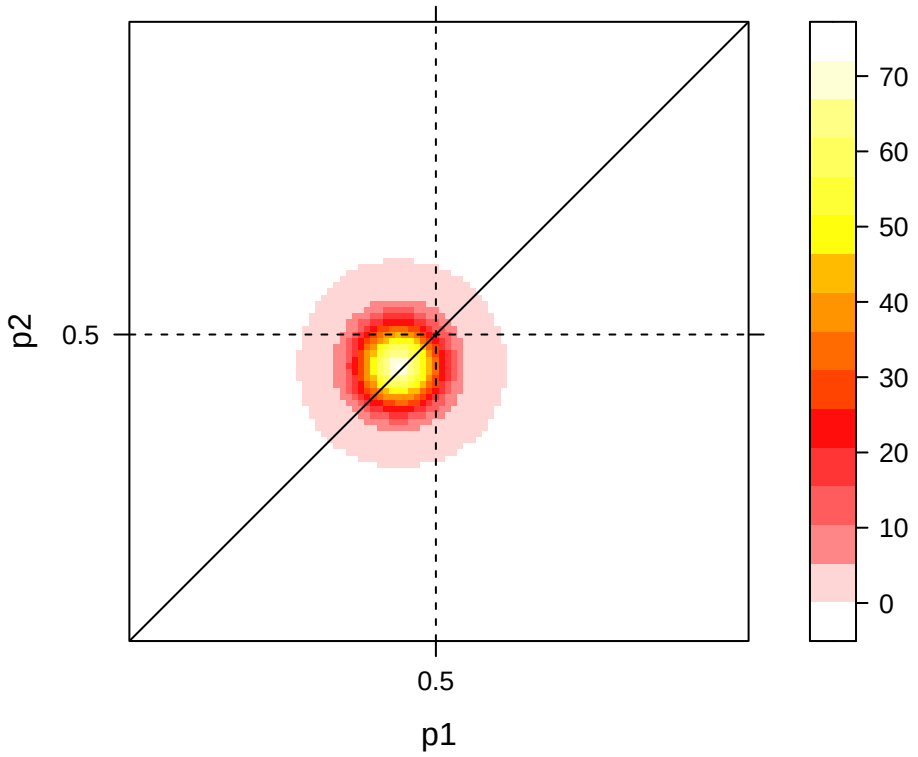
```
paste("sd of p2", sd_2_ar)
```

```
## [1] "sd of p2 0.0468589026414095"
```

step 2: Fit only using new data

```
#define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 / nGridPoints

# prior params
aPrior = 5
bPrior = 5

data = read.csv("~/Desktop/PS2_data.csv")
data1_ar = data$ball_1[101:200]
data2_ar = data$ball_2[101:200]
nWater_1_ar = sum(data1_ar)
nWater_2_ar = sum(data2_ar)
n1_ar = length(data1_ar)
n2_ar = length(data2_ar)

# 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_ar,n1_ar,p1) * dbinom(nWater_2_ar,n2_ar,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )

# Plot heat map

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)})
```

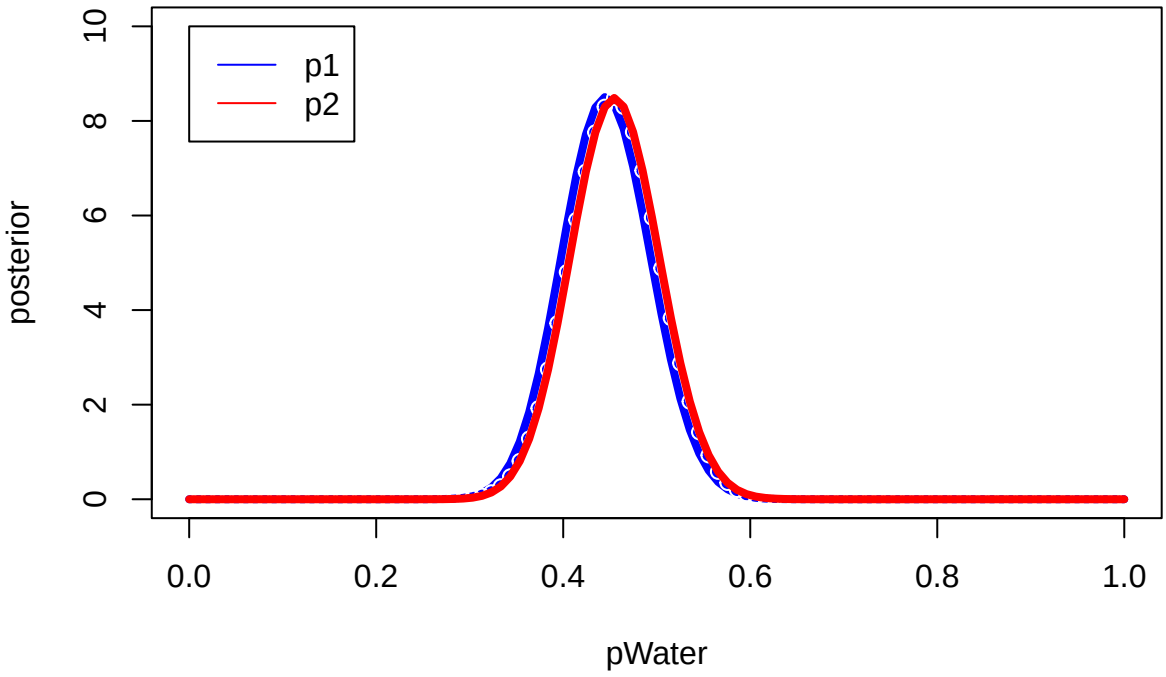



```
# Compute Probability of p2>p1

sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p2 > p1 = ", sum)
```

```
## [1] "posterior prob p2 > p1 =  0.583992971565739"
```

```
# Plot marginal posterior densities

##########
# Compute marginal posterior densities
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)

marg_post_1 <- function(p1){
  for (i in 1:nGridPoints){
    P <- sum(postM[floor(p1 * 100), ])* 1/100
```

```
  }
  P
}
marg_post_2 <- function(p2){
  for (i in 1:nGridPoints){
    P <- sum(postM[ , floor(p2 * 100)])* 1/100
  }
  P
}

marg_post_1_grid <- pGrid
marg_post_2_grid <- pGrid

for (i in 1:nGridPoints){
  marg_post_1_grid[i] <- marg_post_1(p1 = pGrid[i])
}
marg_post_1_grid <- marg_post_1_grid / (sum(marg_post_1_grid)* (1/nGridPoints))
for (i in 1:nGridPoints){
  marg_post_2_grid[i] <- marg_post_2(p2 = pGrid[i])
}
marg_post_2_grid <- marg_post_2_grid / (sum(marg_post_2_grid) * (1/nGridPoints))

# Plot marginal distributions
plot(pGrid, marg_post_1_grid, type="l", lwd=4,
     xlab = "pWater", ylab = "posterior",
     ylim = c(0,10), main = "Marginal Posterior Distributions", col="blue")
points(pGrid, marg_post_2_grid, ylim = c(0,10), col="white")
lines(pGrid, marg_post_2_grid,  lwd=4, ylim = c(0,10), col="red")
legend(0,10, legend = c("p1", "p2"),
       col = c("blue","red"), lty=c(1,1) )
```




```
##########

# Compute means and standard deviations

mean_1_ar <- 0
mean_2_ar <- 0
i2_1_ar <- 0
i2_2_ar <- 0
for (i in 1:nGridPoints) {
  mean_1_ar <- mean_1_ar + marg_post_1_grid[i] * i * (1/nGridPoints)^2
  i2_1_ar <- i2_1_ar + marg_post_1_grid[i] * i^2 * (1/nGridPoints)^3
  mean_2_ar <- mean_2_ar + marg_post_2_grid[i] * i * (1/nGridPoints)^2
  i2_2_ar <- i2_2_ar + marg_post_2_grid[i] * i^2 * (1/nGridPoints)^3
}

sd_1_ar <- sqrt(i2_1_ar - mean_1_ar^2)
sd_2_ar <- sqrt(i2_2_ar - mean_2_ar^2)
paste("mean of p1", mean_1_ar)
```

```
## [1] "mean of p1 0.452"
```

```
paste("mean of p2", mean_2_ar)
```

```
## [1] "mean of p2 0.461"
```

```
paste("sd of p1", sd_1_ar)
```

```
## [1] "sd of p1 0.0466012411387916"
```

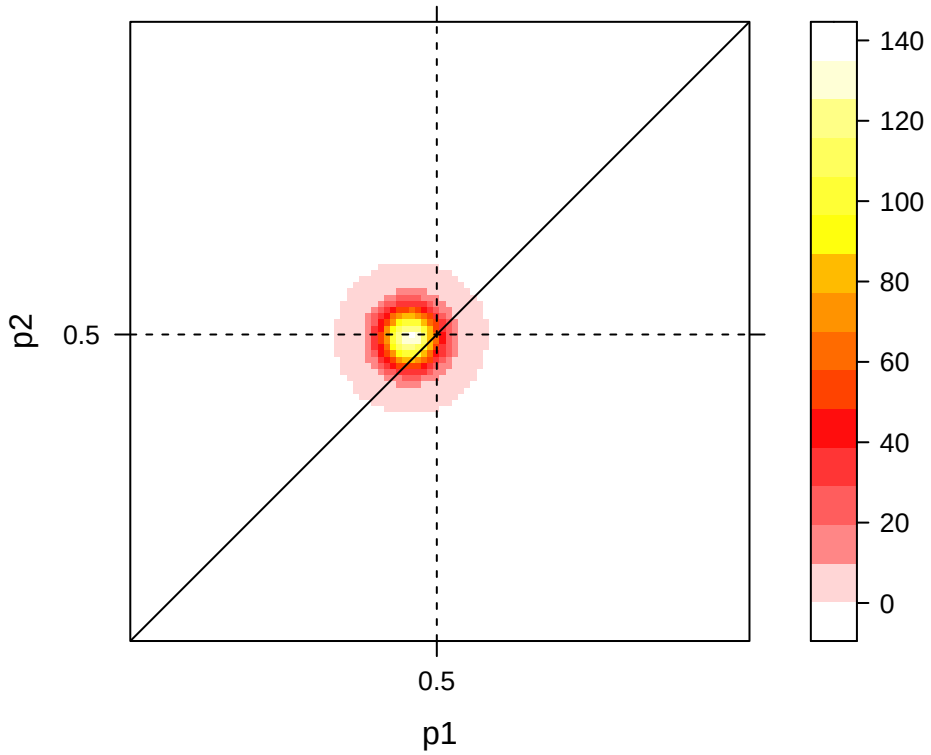
```
paste("sd of p2", sd_2_ar)
```

```
## [1] "sd of p2 0.0467029138508725"
```

Step 3: Fit using all the data

```
#define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 / nGridPoints

# prior params
aPrior = 5
bPrior = 5

data = read.csv("~/Desktop/PS2_data.csv")
data1_ar = data$ball_1[1:200]
data2_ar = data$ball_2[1:200]
nWater_1_ar = sum(data1_ar)
nWater_2_ar = sum(data2_ar)
n1_ar = length(data1_ar)
n2_ar = length(data2_ar)

# 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_ar,n1_ar,p1) * dbinom(nWater_2_ar,n2_ar,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )

# Plot heat map

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)})
```

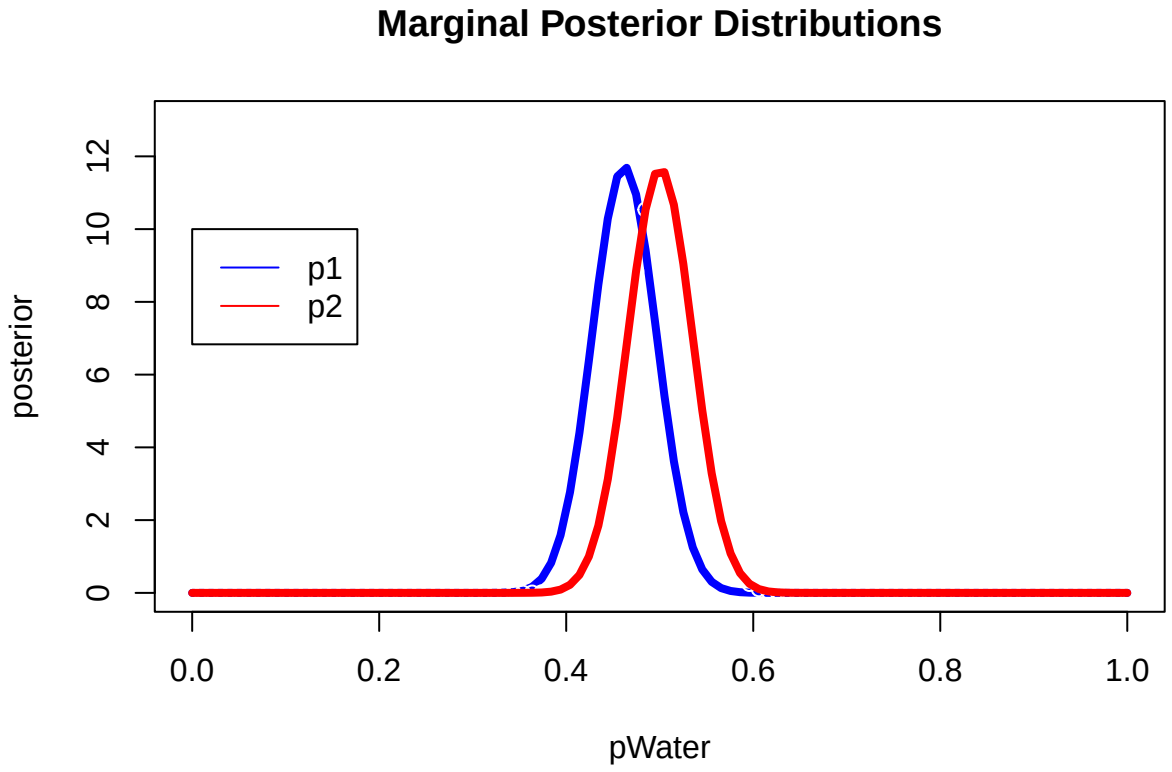



```
# Compute Probability of p2>p1

sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p2 > p1 = ", sum)
```

```
## [1] "posterior prob p2 > p1 =  0.813035878062756"
```

```
# Plot marginal posterior densities
##########
# Compute marginal posterior densities
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)

marg_post_1 <- function(p1){
  for (i in 1:nGridPoints){
    P <- sum(postM[floor(p1 * 100), ])* 1/100
  }
```

```
  P
}
marg_post_2 <- function(p2){
  for (i in 1:nGridPoints){
    P <- sum(postM[ , floor(p2 * 100)])* 1/100
  }
  P
}

marg_post_1_grid <- pGrid
marg_post_2_grid <- pGrid

for (i in 1:nGridPoints){
  marg_post_1_grid[i] <- marg_post_1(p1 = pGrid[i])
}
marg_post_1_grid <- marg_post_1_grid / (sum(marg_post_1_grid)* (1/nGridPoints))
for (i in 1:nGridPoints){
  marg_post_2_grid[i] <- marg_post_2(p2 = pGrid[i])
}
marg_post_2_grid <- marg_post_2_grid / (sum(marg_post_2_grid) * (1/nGridPoints))

# Plot marginal distributions
plot(pGrid, marg_post_1_grid, type="l", lwd=4,
     xlab = "pWater", ylab = "posterior",
     ylim = c(0,13), main = "Marginal Posterior Distributions", col="blue")
points(pGrid, marg_post_2_grid, ylim = c(0,13), col="white")
lines(pGrid, marg_post_2_grid,  lwd=4, ylim = c(0,13), col="red")
legend(0,10, legend = c("p1", "p2"),
       col = c("blue","red"), lty=c(1,1) )
```



```
##########

# Compute means and standard deviations

mean_1_ar <- 0
mean_2_ar <- 0
i2_1_ar <- 0
i2_2_ar <- 0
for (i in 1:nGridPoints) {
  mean_1_ar <- mean_1_ar + marg_post_1_grid[i] * i * (1/nGridPoints)^2
  i2_1_ar <- i2_1_ar + marg_post_1_grid[i] * i^2 * (1/nGridPoints)^3
  mean_2_ar <- mean_2_ar + marg_post_2_grid[i] * i * (1/nGridPoints)^2
  i2_2_ar <- i2_2_ar + marg_post_2_grid[i] * i^2 * (1/nGridPoints)^3
}

sd_1_ar <- sqrt(i2_1_ar - mean_1_ar^2)
sd_2_ar <- sqrt(i2_2_ar - mean_2_ar^2)
paste("mean of p1", mean_1_ar)
```

```
## [1] "mean of p1 0.467857142857143"
```

```
paste("mean of p2", mean_2_ar)
```

```
## [1] "mean of p2 0.505571428571429"
```

```
paste("sd of p1", sd_1_ar)
```

```
## [1] "sd of p1 0.0339223081558159"
```

```
paste("sd of p2", sd_2_ar)
```

```
## [1] "sd of p2 0.034071022978893"
```

PART 2

step 1

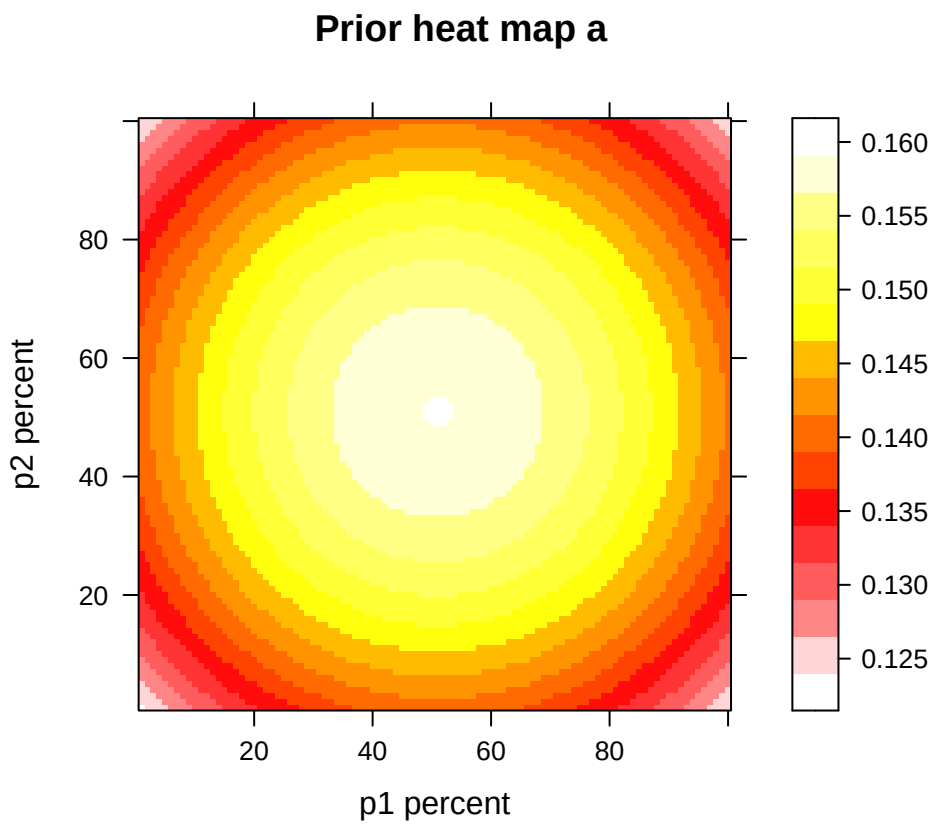
A

```
# function generates a 100x100 prior matrix with a bivariate normal distribution
bnd <- function(mu1, mu2, s1, s2, rho){
  priorMat = matrix(rep(1, 100 ^ 2 ),
                nrow = 100,
                ncol = 100,
                byrow = TRUE)
  for (row in 1:100) {
    for (col in 1:100) {
      x1 <- (col-1) / 100
      x2 <- (row-1) / 100
      z <- (x1 - mu1)^2 / s1^2 - (2 * rho* (x1 - mu1) * (x2 - mu2)) / (s1 * s2) + (x2 - mu2)^2 / s2^2
      P <- (1 / (2 * pi * s1 * s2 * sqrt(1 - rho^2))) * exp(-z / (2 * (1 - rho^2)))
      priorMat[row,col] = P
  }
  }
```

```
  priorMat
}

# function retrieves a specific matrix element (p1,p2) from the prior matrix
get_prior_val <- function(pvec, Mu1, Mu2, S1, S2, Rho){
  pmat <- bnd(mu1 = Mu1, mu2 = Mu2, s1 = S1, s2 = S2, rho = Rho)
  x1 <- floor(100 * pvec[1])
  x2 <- floor(100 * pvec[2])
  pmat[x1, x2]
}

# test normalization
pmat <- bnd(.5, .5, .1, .1, 0)
int = sum(pmat) * 0.01^2
print(int)
```

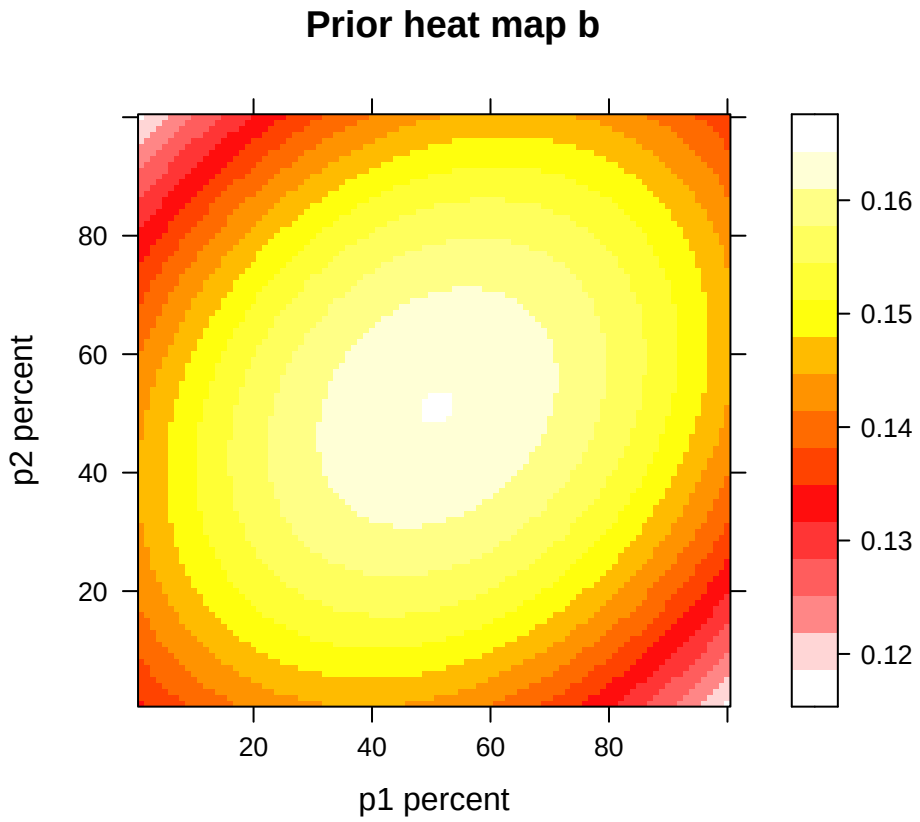
```
## [1] 0.9999988
```

```
# part a: heat map for mu1 = mu2 = 0.5, s1 = s2 = 1, rho = 0
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(bnd(mu1 = 0.5, mu2 = 0.5, s1 = 1, s2 = 1, rho = 0), col.regions=new.palette(20),
          xlab = "p1 percent", ylab = "p2 percent", main="Prior heat map a",
          panel = function(...){
            panel.levelplot(...)
            })
```



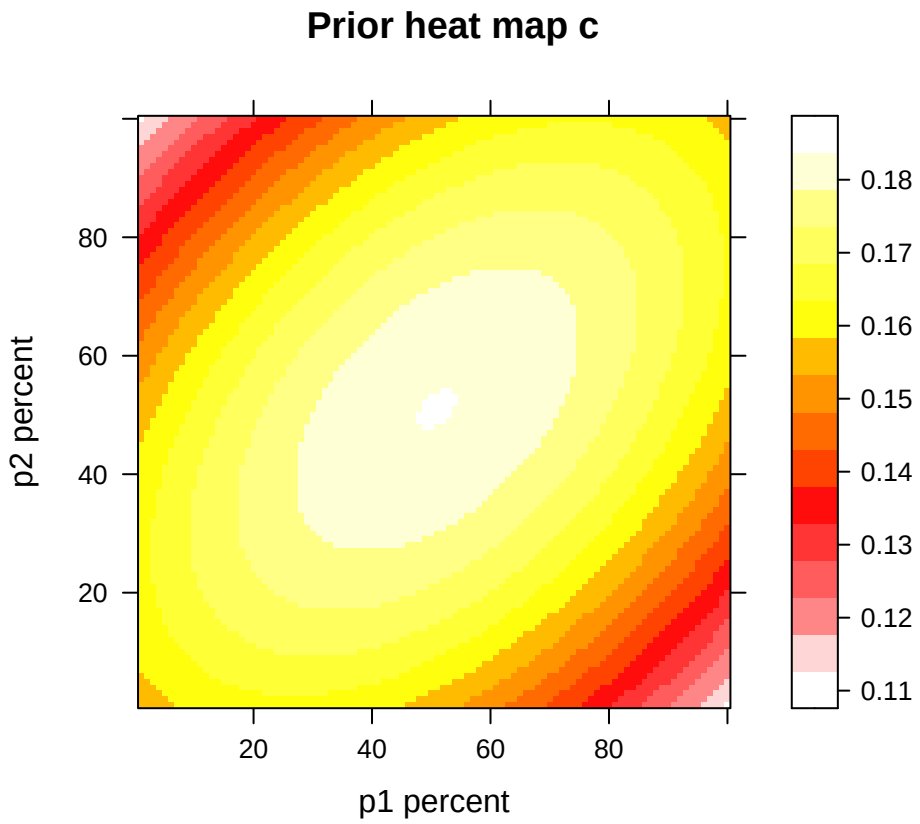
B

```
# part b: heat map for mu1 = mu2 = 0.5, s1 = s2 = 1, rho = 0.25
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(bnd(mu1 = 0.5, mu2 = 0.5, s1 = 1, s2 = 1, rho = 0.25), col.regions=new.palette(20),
          xlab = "p1 percent", ylab = "p2 percent", main="Prior heat map b",
          panel = function(...){
            panel.levelplot(...)
            })
```



C

```
# part c: heat map for mu1 = mu2 = 0.5, s1 = s2 = 1, rho = 0.5
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(bnd(mu1 = 0.5, mu2 = 0.5, s1 = 1, s2 = 1, rho = 0.5), col.regions=new.palette(20),
          xlab = "p1 percent", ylab = "p2 percent", main="Prior heat map c",
          panel = function(...){
            panel.levelplot(...)
            })
```

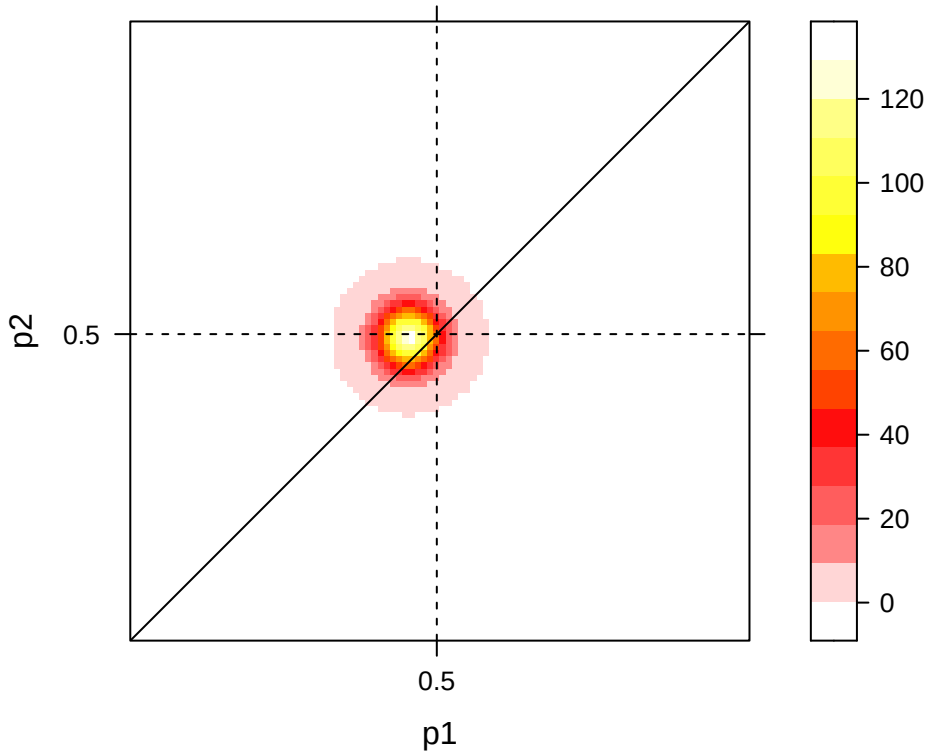


Step 2

A

```
# estimate using all data and prior a
#define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 / nGridPoints

# data
data = read.csv("~/Desktop/PS2_data.csv")
data1_ar = data$ball_1[1:200]
data2_ar = data$ball_2[1:200]
nWater_1_ar = sum(data1_ar)
nWater_2_ar = sum(data2_ar)
n1_ar = length(data1_ar)
n2_ar = length(data2_ar)

# Define prior matrix
priorM = bnd(mu1 = 0.5, mu2 = 0.5, s1 = 1, s2 = 1, rho = 0)

# 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_ar,n1_ar,p1) * dbinom(nWater_2_ar,n2_ar,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )

# Plot heat map

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)})
```

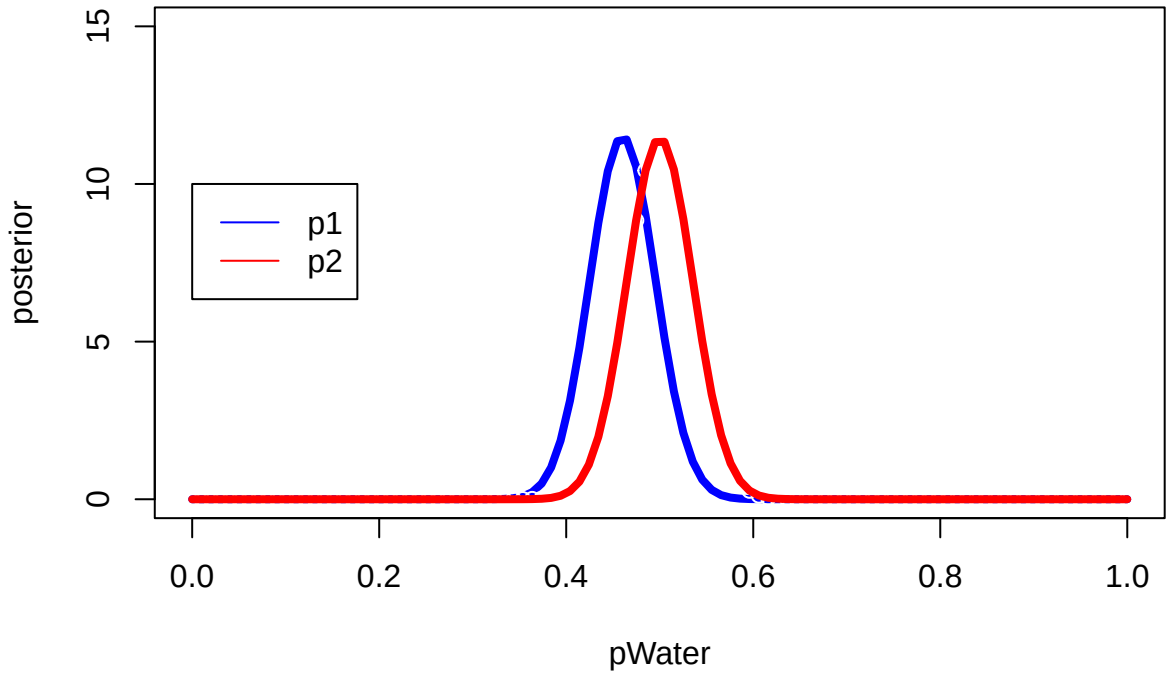



```
# Compute Probability of p2>p1

sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
```

```
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p2 > p1 = ", sum)
```

```
## [1] "posterior prob p2 > p1 =  0.816540573067689"
```

```
# Compute marginal posterior densities
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)

marg_post_1 <- function(p1){
  for (i in 1:nGridPoints){
    P <- sum(postM[floor(p1 * 100), ])* 1/100
  }
  P
}
marg_post_2 <- function(p2){
  for (i in 1:nGridPoints){
    P <- sum(postM[ , floor(p2 * 100)])* 1/100
  }
  P
}

marg_post_1_grid <- pGrid
marg_post_2_grid <- pGrid

for (i in 1:nGridPoints){
  marg_post_1_grid[i] <- marg_post_1(p1 = pGrid[i])
}
marg_post_1_grid <- marg_post_1_grid / (sum(marg_post_1_grid)* (1/nGridPoints))
for (i in 1:nGridPoints){
  marg_post_2_grid[i] <- marg_post_2(p2 = pGrid[i])
}
marg_post_2_grid <- marg_post_2_grid / (sum(marg_post_2_grid) * (1/nGridPoints))

# Plot marginal distributions
plot(pGrid, marg_post_1_grid, type="l", lwd=4,
     xlab = "pWater", ylab = "posterior",
     ylim = c(0,15), main = "Marginal Posterior Distributions", col="blue")
points(pGrid, marg_post_2_grid, ylim = c(0,15), col="white")
lines(pGrid, marg_post_2_grid,  lwd=4, ylim = c(0,15), col="red")
legend(0,10, legend = c("p1", "p2"),
       col = c("blue","red"), lty=c(1,1) )
```




```
# Compute means and standard deviations

mean_1_ar <- 0
mean_2_ar <- 0
i2_1_ar <- 0
i2_2_ar <- 0
for (i in 1:nGridPoints) {
  mean_1_ar <- mean_1_ar + marg_post_1_grid[i] * i * (1/nGridPoints)^2
  i2_1_ar <- i2_1_ar + marg_post_1_grid[i] * i^2 * (1/nGridPoints)^3
  mean_2_ar <- mean_2_ar + marg_post_2_grid[i] * i * (1/nGridPoints)^2
  i2_2_ar <- i2_2_ar + marg_post_2_grid[i] * i^2 * (1/nGridPoints)^3
}

sd_1_ar <- sqrt(i2_1_ar - mean_1_ar^2)
sd_2_ar <- sqrt(i2_2_ar - mean_2_ar^2)
paste("mean of p1", mean_1_ar)
```

```
## [1] "mean of p1 0.466054003715474"
```

```
paste("mean of p2", mean_2_ar)
```

```
## [1] "mean of p2 0.50521574222169"
```

```
paste("sd of p1", sd_1_ar)
```

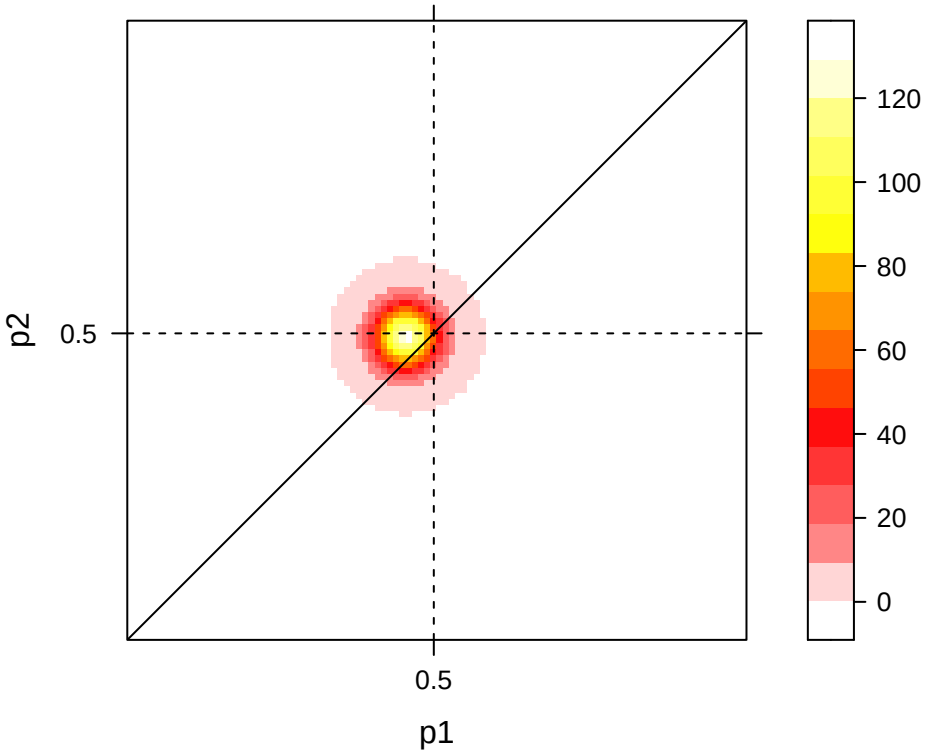
```
## [1] "sd of p1 0.0345520236870336"
```

```
paste("sd of p2", sd_2_ar)
```

```
## [1] "sd of p2 0.034714795413329"
```

B

```
# estimate using all data and prior a
#define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 / nGridPoints

# data
data = read.csv("~/Desktop/PS2_data.csv")
data1_ar = data$ball_1[1:200]
data2_ar = data$ball_2[1:200]
nWater_1_ar = sum(data1_ar)
nWater_2_ar = sum(data2_ar)
n1_ar = length(data1_ar)
n2_ar = length(data2_ar)

# Define prior matrix
priorM = bnd(mu1 = 0.5, mu2 = 0.5, s1 = 1, s2 = 1, rho = 0.25)

# 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_ar,n1_ar,p1) * dbinom(nWater_2_ar,n2_ar,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )

# Plot heat map

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)})
```

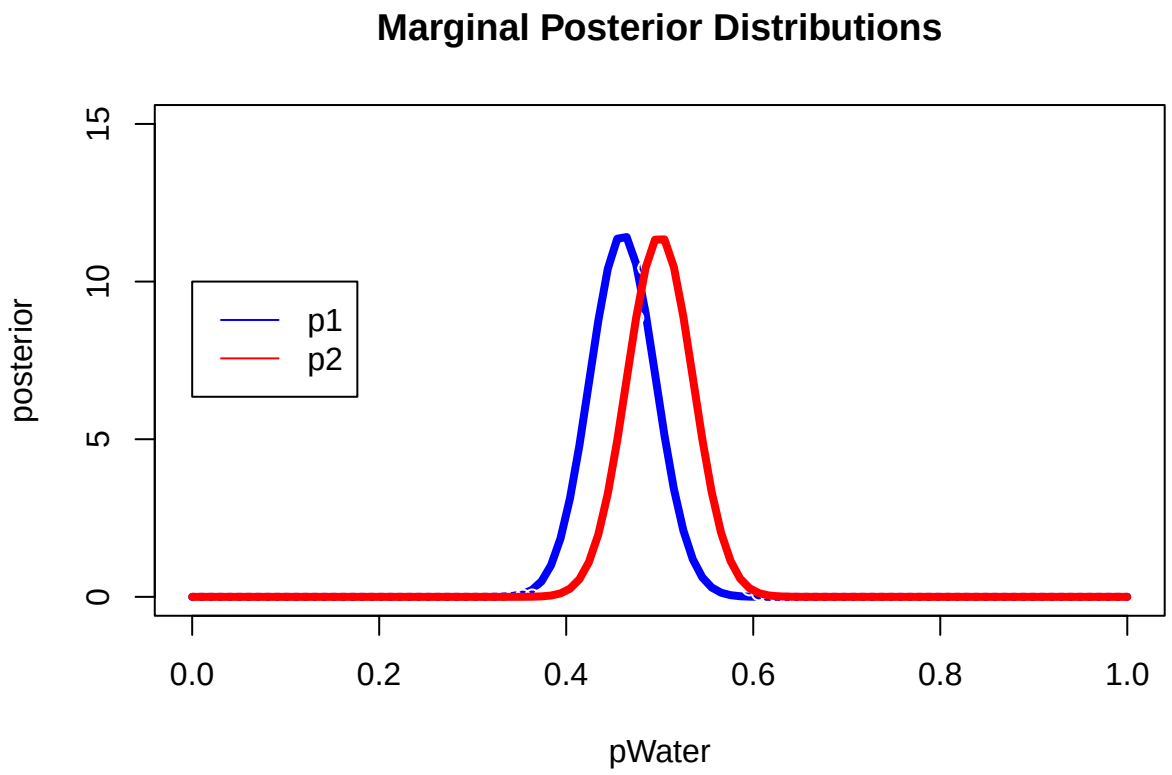



```
# Compute Probability of p2>p1

sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p2 > p1 = ", sum)
```

```
## [1] "posterior prob p2 > p1 =  0.816503722137692"
```

```
# Compute marginal posterior densities
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)

marg_post_1 <- function(p1){
  for (i in 1:nGridPoints){
    P <- sum(postM[floor(p1 * 100), ])* 1/100
  }
  P
}
marg_post_2 <- function(p2){
  for (i in 1:nGridPoints){
    P <- sum(postM[ , floor(p2 * 100)])* 1/100
  }
  P
}

marg_post_1_grid <- pGrid
marg_post_2_grid <- pGrid
```

```r
for (i in 1:nGridPoints){
  marg_post_1_grid[i] <- marg_post_1(p1 = pGrid[i])
}
marg_post_1_grid <- marg_post_1_grid / (sum(marg_post_1_grid)* (1/nGridPoints))
for (i in 1:nGridPoints){
  marg_post_2_grid[i] <- marg_post_2(p2 = pGrid[i])
}
marg_post_2_grid <- marg_post_2_grid / (sum(marg_post_2_grid) * (1/nGridPoints))

# Plot marginal distributions
plot(pGrid, marg_post_1_grid, type="l", lwd=4,
     xlab = "pWater", ylab = "posterior",
     ylim = c(0,15), main = "Marginal Posterior Distributions", col="blue")
points(pGrid, marg_post_2_grid, ylim = c(0,15), col="white")
lines(pGrid, marg_post_2_grid,  lwd=4, ylim = c(0,15), col="red")
legend(0,10, legend = c("p1", "p2"),
       col = c("blue","red"), lty=c(1,1) )
```




```r
# Compute means and standard deviations

mean_1_ar <- 0
mean_2_ar <- 0
i2_1_ar <- 0
i2_2_ar <- 0
for (i in 1:nGridPoints) {
  mean_1_ar <- mean_1_ar + marg_post_1_grid[i] * i * (1/nGridPoints)^2
  i2_1_ar <- i2_1_ar + marg_post_1_grid[i] * i^2 * (1/nGridPoints)^3
  mean_2_ar <- mean_2_ar + marg_post_2_grid[i] * i * (1/nGridPoints)^2
  i2_2_ar <- i2_2_ar + marg_post_2_grid[i] * i^2 * (1/nGridPoints)^3
```

```
}

sd_1_ar <- sqrt(i2_1_ar - mean_1_ar^2)
sd_2_ar <- sqrt(i2_2_ar - mean_2_ar^2)
paste("mean of p1", mean_1_ar)
```

```
## [1] "mean of p1 0.466053547428367"
```

```
paste("mean of p2", mean_2_ar)
```

```
## [1] "mean of p2 0.505199587157216"
```

```
paste("sd of p1", sd_1_ar)
```

```
## [1] "sd of p1 0.0345506640177531"
```

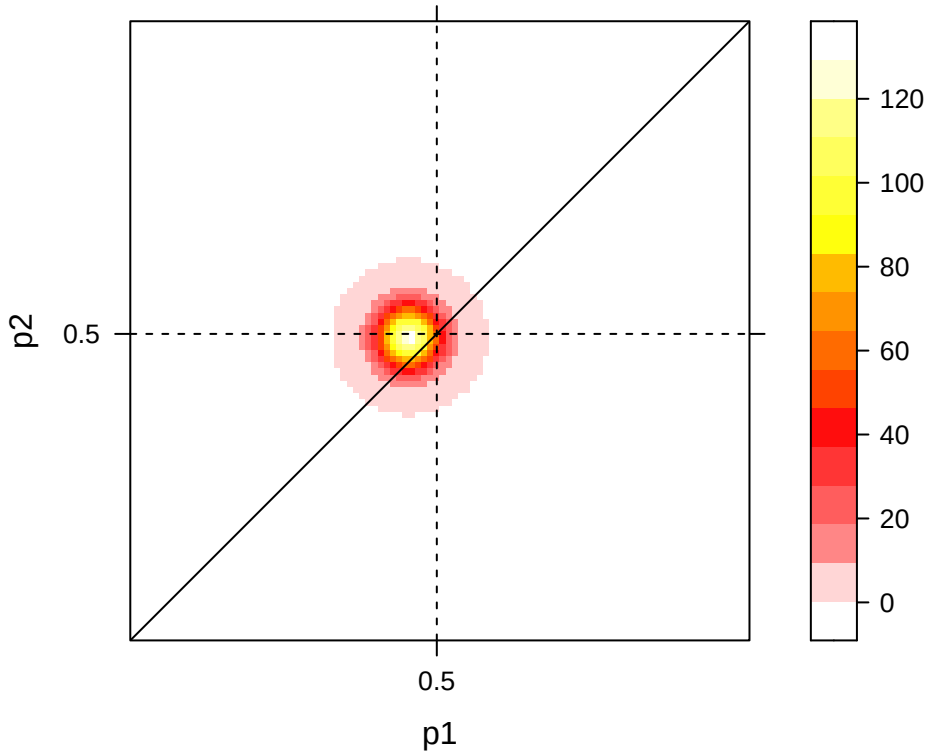
```
paste("sd of p2", sd_2_ar)
```

```
## [1] "sd of p2 0.0347133785341847"
```

C

```
# estimate using all data and prior a
#define grid size
nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)
gridSize = 1 / nGridPoints

# data
data = read.csv("~/Desktop/PS2_data.csv")
data1_ar = data$ball_1[1:200]
data2_ar = data$ball_2[1:200]
nWater_1_ar = sum(data1_ar)
nWater_2_ar = sum(data2_ar)
n1_ar = length(data1_ar)
n2_ar = length(data2_ar)

# Define prior matrix
priorM = bnd(mu1 = 0.5, mu2 = 0.5, s1 = 1, s2 = 1, rho = 0.5)

# 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_ar,n1_ar,p1) * dbinom(nWater_2_ar,n2_ar,p2) * priorM[row,col]
  }
}
postM = postM / ( sum(postM) * gridSize ^ 2 )
```

```
# Plot heat map

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)})
```

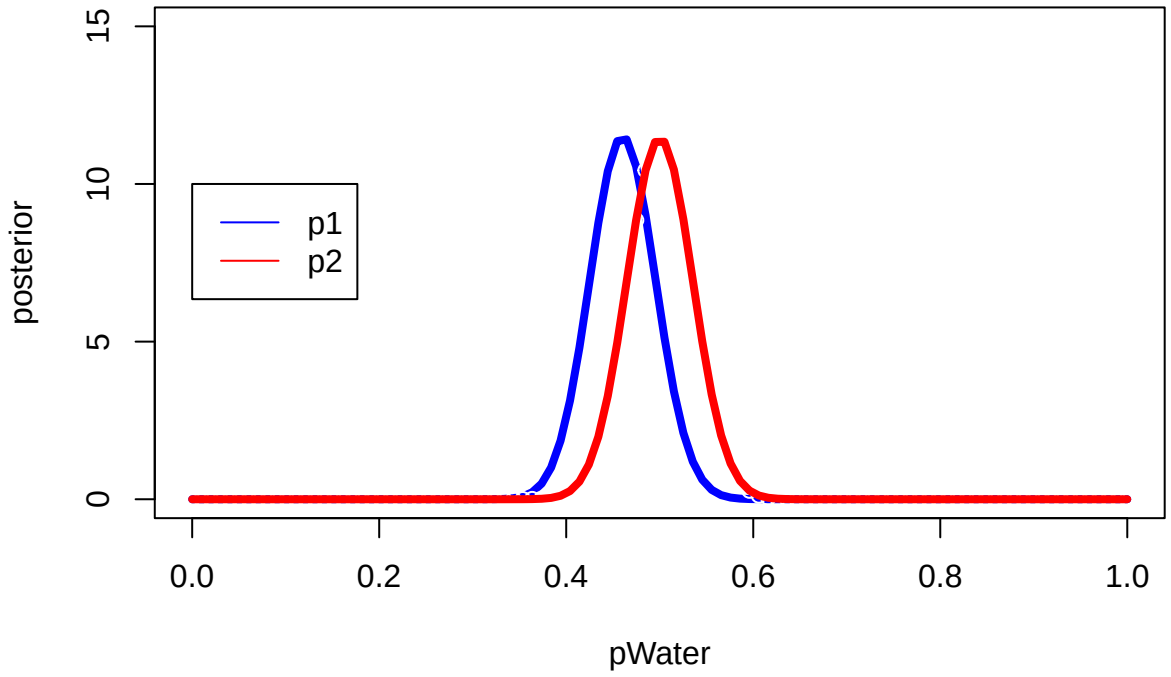



```
# Compute Probability of p2>p1

sum = 0
for (row in 1:nGridPoints) {
  for (col in row:nGridPoints) {
    sum = sum + (postM[row,col] * gridSize ^ 2)
  }
}
paste("posterior prob p2 > p1 = ", sum)
```

```
## [1] "posterior prob p2 > p1 =  0.816430053887279"
```

```
# Compute marginal posterior densities
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)

marg_post_1 <- function(p1){
```

```
  for (i in 1:nGridPoints){
    P <- sum(postM[floor(p1 * 100), ])* 1/100
  }
  P
}
marg_post_2 <- function(p2){
  for (i in 1:nGridPoints){
    P <- sum(postM[ , floor(p2 * 100)])* 1/100
  }
  P
}

marg_post_1_grid <- pGrid
marg_post_2_grid <- pGrid

for (i in 1:nGridPoints){
  marg_post_1_grid[i] <- marg_post_1(p1 = pGrid[i])
}
marg_post_1_grid <- marg_post_1_grid / (sum(marg_post_1_grid)* (1/nGridPoints))
for (i in 1:nGridPoints){
  marg_post_2_grid[i] <- marg_post_2(p2 = pGrid[i])
}
marg_post_2_grid <- marg_post_2_grid / (sum(marg_post_2_grid) * (1/nGridPoints))

# Plot marginal distributions
plot(pGrid, marg_post_1_grid, type="l", lwd=4,
     xlab = "pWater", ylab = "posterior",
     ylim = c(0,15), main = "Marginal Posterior Distributions", col="blue")
points(pGrid, marg_post_2_grid, ylim = c(0,15), col="white")
lines(pGrid, marg_post_2_grid,  lwd=4, ylim = c(0,15), col="red")
legend(0,10, legend = c("p1", "p2"),
       col = c("blue","red"), lty=c(1,1) )
```




```
# Compute means and standard deviations

mean_1_ar <- 0
mean_2_ar <- 0
i2_1_ar <- 0
i2_2_ar <- 0
for (i in 1:nGridPoints) {
  mean_1_ar <- mean_1_ar + marg_post_1_grid[i] * i * (1/nGridPoints)^2
  i2_1_ar <- i2_1_ar + marg_post_1_grid[i] * i^2 * (1/nGridPoints)^3
  mean_2_ar <- mean_2_ar + marg_post_2_grid[i] * i * (1/nGridPoints)^2
  i2_2_ar <- i2_2_ar + marg_post_2_grid[i] * i^2 * (1/nGridPoints)^3
}

sd_1_ar <- sqrt(i2_1_ar - mean_1_ar^2)
sd_2_ar <- sqrt(i2_2_ar - mean_2_ar^2)
paste("mean of p1", mean_1_ar)
```

```
## [1] "mean of p1 0.466063481189188"
```

```
paste("mean of p2", mean_2_ar)
```

```
## [1] "mean of p2 0.505178324521156"
```

```
paste("sd of p1", sd_1_ar)
```

```
## [1] "sd of p1 0.0345453918054662"
```

```
paste("sd of p2", sd_2_ar)
```

```
## [1] "sd of p2 0.0347078342292279"
```

Step 3

The results from parts a,b, and c are extremely similar and pretty much identical. Therefore, we don't really have to concern ourselves with whether our priors are uncorrelated or not, since it won't affect the results if we have a lot of data.

PART 3

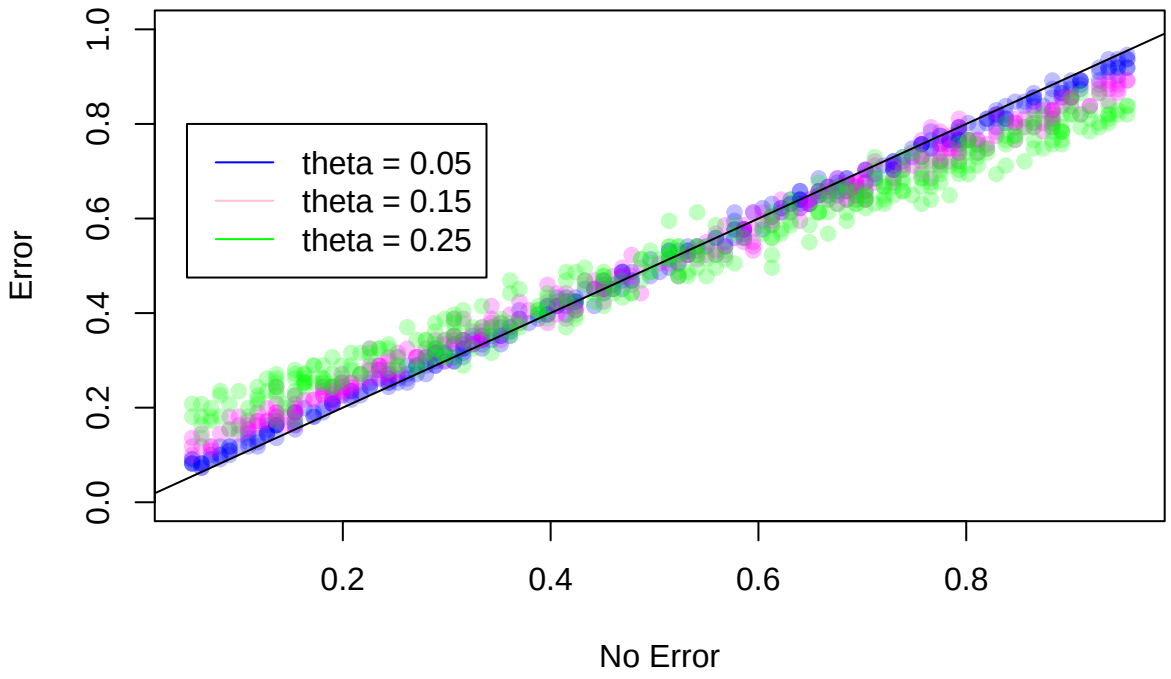
Step 1

```
# set random seed
set.seed(123)

nGridPoints = 100
pGrid = seq(from = 0, to = 1,length.out = nGridPoints)

E_05 <- (0)
E_15 <- c(0)
E_25 <- c(0)
NE_05 <- c(0)
NE_15 <- c(0)
NE_25 <- c(0)

get_theta <- function(){
  val <- runif(1)
  if (val < 0.33){
    theta <- 0.05
  } else if (val > 0.66){
    theta <- 0.25
  } else {
    theta <- 0.15
  }
  theta
}

for (i in 1:1000) {
  pTrue <- runif(1)
  theta <- get_theta()

  # build dataNoError
  dataNoError <- numeric(100)
  for (j in 1:100) {
    checker <- runif(1)
    if (checker < pTrue){
      dataNoError[j] <- 1
    }
  }
  # build dataError
  dataError <- dataNoError
  for (j in 1:(theta * 100)){
    checker <- runif(1)
    if (checker < 0.5){
```

```
      dataError[j] <- 1
    } else {
      dataError[j] <- 0
    }
  }

  # define prior: beta(5,5) grid version
  prior = dbeta(x = pGrid, shape1 = 5, shape2 = 5)

  #get NoError posterior
  postNE <- numeric(100)
  nWaterNE <- sum(dataNoError)
  for (k in 1:nGridPoints) {
    p1 = pGrid[k]
    postNE[k] = dbinom(nWaterNE,nGridPoints,p1) * prior[k]
  }
  postNE = postNE / ( sum(postNE) * gridSize )

  #get Error posterior
  postE <- numeric(100)
  nWaterE <- sum(dataError)
  for (k in 1:nGridPoints) {
    p1 = pGrid[k]
    postE[k] = dbinom(nWaterE,nGridPoints,p1) * prior[k]
  }
  postE = postE / ( sum(postE) * gridSize )

  # compute means
  meanNE <- 0
  meanE <- 0
  for (l in 1:nGridPoints) {
    meanNE <- meanNE + postNE[l] * l * (1/nGridPoints)^2
    meanE <- meanE + postE[l] * l * (1/nGridPoints)^2
  }

  meanNE <- round(meanNE, 3)
  meanE <- round(meanE, 3)
  if (theta == 0.05){
    NE_05 <- c(NE_05, meanNE)
    E_05 <- c(E_05, meanE)
  } else if (theta == 0.15){
    NE_15 <- c(NE_15, meanNE)
    E_15 <- c(E_15, meanE)
  } else if (theta == 0.25){
    NE_25 <- c(NE_25, meanNE)
    E_25 <- c(E_25, meanE)
  }

}

NE_05 <- NE_05[2:length(NE_05)]
NE_15 <- NE_15[2:length(NE_15)]
NE_25 <- NE_25[2:length(NE_25)]
```

```
E_05 <- E_05[2:length(E_05)]
E_15 <- E_15[2:length(E_15)]
E_25 <- E_25[2:length(E_25)]
```

```
plot(NE_05, E_05,
     xlab = "No Error", ylab = "Error", type="p",
     ylim = c(0,1), col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.25), pch=19)
points(NE_15, E_15, col=rgb(red=1.0, green=0.0, blue=1.0, alpha=0.25), pch=19)
points(NE_25, E_25, col=rgb(red=0.0, green=1.0, blue=0.0, alpha=0.25), pch=19)
lines(c(0,1),c(0,1))
legend(.05,0.8, legend = c("theta = 0.05", "theta = 0.15", "theta = 0.25"),
       col = c("blue","pink", "green"), lty=c(1,1) )
```



Step 2

We observe a trend where higher theta_true makes the line you would fit to that set of points more horizontal. This is because adding more "error" points to a given dataset pulls the mean of its posterior toward 0.05. Therefore, if theta_true =1, the corresponding points would make a horizontal line.