Normal Mixture Model Gibbs Sampler

Jordan Aron

October 15, 2017

$z_i \sim Bern(\frac{a}{a+b})$ where $a = (p * f(x_i|\mu_1, \sigma_1))$ and $b = ((1-p) * f(x_i|\mu_2, \sigma_2))$. 1000 $z_i's$ are drawn each iteration.

$p \sim beta(\sum z, n - \sum z)$

$\sigma_1 \sim inverse - gamma(\frac{\kappa_{n_1}}{2}, \frac{\kappa_{n_1}}{2}\sigma^2_{n_1})$ where $\kappa_{n_1} = \kappa_{0_1} + n_1$, κ_{0_1} is the prior sample size for population 1, and n_1 is the sample size of the data for population 1

$\sigma_2 \sim inverse - gamma(\frac{\kappa_{n_2}}{2}, \frac{\kappa_{n_2}}{2}\sigma^2_{n_2})$ where $\kappa_{n_2} = \kappa_{0_2} + n_2$, κ_{0_2} is the prior sample size for population 2, and n_2 is the sample size of the data for population 2

$\theta_1 \sim normal(\mu_{n_1}, \frac{\sigma_1^2}{\kappa_{n_1}})$ where $\mu_{n_1} = \frac{\kappa_{0_1}}{\kappa_{n_1}}\mu_{0_1} + \frac{n_1}{\kappa_{n_1}}\bar{y_1}$, μ_{0_1} is the prior mean for population 1, and $\bar{y_1}$ is the mean of the data for population 1

$\theta_2 \sim normal(\mu_{n_2}, \frac{\sigma_2^2}{\kappa_{n_2}})$ where $\mu_{n_2} = \frac{\kappa_{0_2}}{\kappa_{n_2}}\mu_{0_2} + \frac{n_2}{\kappa_{n_2}}\bar{y_2}$, μ_{0_2} is the prior mean for population 1, and $\bar{y_2}$ is the mean of the data for population 1

If the Metropolis-Hastings algorihtm is used to calculate p:

We are attempting to sample from $P(p|\sum z)$.

The prior for p is: $P(p) = \frac{sin(5*\pi*p)^2}{c}$ where c is some normalizing constant.

As the $z_i's$ are distributed the same as above, $P(\sum z|p) = \binom{n}{\sum z} * p^{\sum z} * (1-p)^{n-\sum z}$

The proposal distribution for p^* is the uniform distribution on (0,1)

We then calculate r. $r = \frac{P(p^*|\sum z)}{P(p^{(s)}|\sum z)} = \frac{P(\sum z|p^*)P(p^*)}{P(\sum z|p^{(s)})P(p^{(s)})} = \frac{((p^*)^{\sum z} * (1-(p^*))^{n-\sum z}) * (sin(5*\pi*p^*)^2)}{(p^{\sum z} * (1-p)^{n-\sum z}) * (sin(5*\pi*p)^2)}$. Note that both the c's and the $\binom{n}{\sum z}$ are cancelled out. We accept p^* as a new value with probability r. If rejected p is instead accepted.

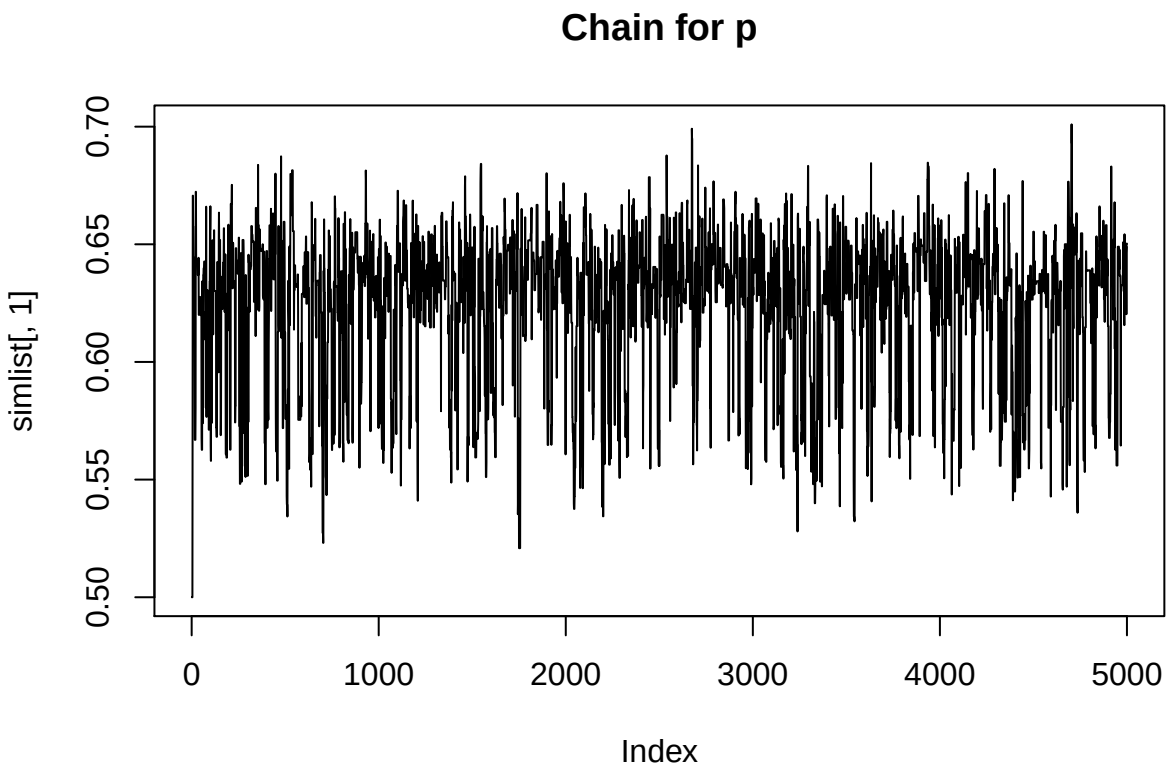
```
#initialize values
#balance of obs1 vs. obs2 determins mixing of the normal distribution
obs1 <- 300
obs2 <- 200
#total amount of observations
n <- obs1+obs2

#Set to one if you want to use a metropolis-hastings algorithm to calculate p
#Set to zero if you want to use gibbs sampler method
use.MH <- 1


#initialize values for population 1
pop1.trueMean <- 0
pop1.trueSD <- 10

#initialize values for population 2
pop2.trueMean <- 50
pop2.trueSD <- 10
```

```r
#initialize values for prior for population 1
pop1.prior.sampleSize <- 10
pop1.prior.trueMean <- 15
pop1.prior.trueSD <- 20

#initialize values for prior for population 2
pop2.prior.sampleSize <- 15
pop2.prior.trueMean <- 45
pop2.prior.trueSD <- 20


#Determines number of iterations for gibbs sampler
trials <-5000



#####Don't Alter Anything Below#####

#creates data, then binds together to create normal mixture model
samp1 <- rnorm(obs1,pop1.trueMean,pop1.trueSD)
samp2 <- rnorm(obs2,pop2.trueMean,pop2.trueSD)
data <- c(samp1,samp2)

#Boys.18 <- c(179,195.1,183.7,178.7,171.5,181.8,172.5,174.6,190.4,173.8,172.6,185.2,178.4,177.6,183.5,1

#Girls.18 <- c(169.6,166.8,157.1,181.1,158.4,165.6,166.7,156.5,168.1,165.3,163.7,173.7,169.2,170.1,164.

#data<-c(Boys.18,Girls.18)

#Wendys <- c(47,50,31,175,65,68,87,114,135,116,125,130,127,150,50,52,226,34,45,85,130,102,108,58,63,84,

#Hot <- c(177,233,166,154,173,99,161,240,244,251,212,156,127,166,113,87,223,225,259,261,395,409,398,326

#data <- c(Wendys,Hot)

n <- length(data)
p<-0.5
r<-1

#creates priors
prior.pop1 <- rnorm(pop1.prior.sampleSize, pop1.prior.trueMean, pop1.prior.trueSD)
prior.pop2 <- rnorm(pop2.prior.sampleSize, pop2.prior.trueMean, pop2.prior.trueSD)

#creates matrix where gibbs will be done
  simlist <- matrix(rep(0,5*trials), ncol = 5)
colnames(simlist) <- c("p", "SD of Pop1", "SD of Pop2", "Mean of Pop1", "Mean of Pop2")

#saves value for later usage
pop1.prior.mean <- mean(prior.pop1)
pop1.prior.variance <- var(prior.pop1)
pop2.prior.mean <- mean(prior.pop2)
pop2.prior.variance <- var(prior.pop2)
```

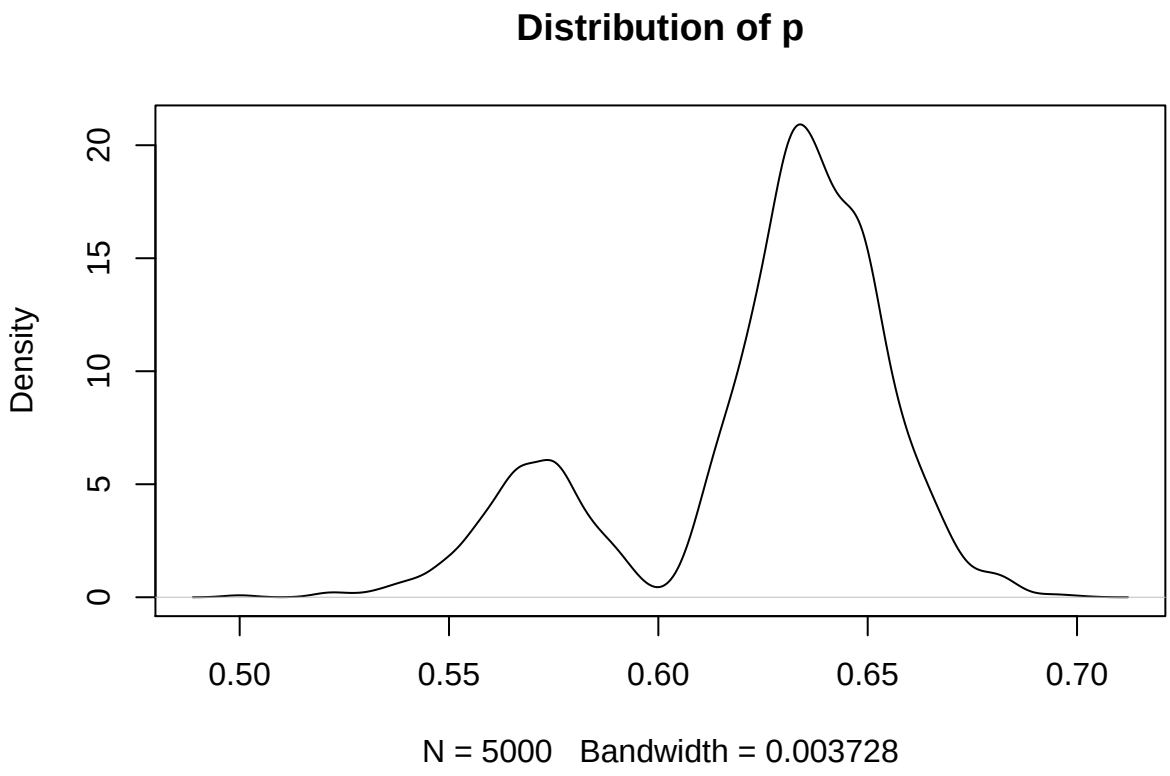
```
#initialize first entry in gibbs so it will run correctly
#p sigma1 sigma2 theta1 theta2
simlist[1,1] <- .5
simlist[1,2] <- sqrt(pop1.prior.variance)
simlist[1,3] <- sqrt(pop2.prior.variance)
simlist[1,4] <- pop1.prior.mean
simlist[1,5] <- pop2.prior.mean


#The actual gibbs sampler
#starts at 2 since we initialized values for when i=1
for(i in 2:trials){

  #max size will be z
  z <- numeric(n)
  a <- numeric(n)
  b <- numeric(n)


  j = 1
  #calculates a_j and b_j for each data_j
  #then calculates z_i's
  for (j in 1:n){
    a[j] = simlist[i-1,1]*dnorm(data[j],simlist[i-1,4],simlist[i-1,2])
    b[j] = (1-simlist[i-1,1])*dnorm(data[j],simlist[i-1,5],simlist[i-1,3])
    z[j] = rbinom(1,1,(a[j]/(a[j]+b[j])))
  }

  #calculates p using beta distribution for gibbs
  if (use.MH == 0){
    simlist[i,1] <- rbeta(1,sum(z), n-sum(z))
  } else {
    #metropolis hastings algorithm to calculate p
    p.star <- p + rnorm(1,0,.05)
    if (p.star < 0){
      p.star = 0
    }
    if (p.star > 1){
      p.star = 1
    }
    r <- ((p.star^sum(z))*((1-p.star)^(n-sum(z)))*(sin(5*pi*p.star)^2))/((p^sum(z))*((1-p)^(n-sum(z)))*
    if (runif(1) < r){
      p <- p.star
    }
    simlist[i,1] <- p
  }


  pop1 <- numeric(n)
  pop2 <- numeric(n)

  j = 1
  #dividing data into pop1 and pop2 as best as we can at the moment
```

```
  for (j in 1:n){
    pop1[j] = z[j]*data[j]
    pop2[j] = ((1-z[j])*-1)*data[j]
    pop2[j] = -1*pop2[j]
  }

  #gets rid of zeros in list
  pop1 <- pop1[pop1!=0]
  pop2 <- pop2[pop2!=0]

  #calculates length, used later
  n1 <- length(pop1)
  n2 <- length(pop2)

  #calculates SD for population 1
  #first calculated precision with gamma, then variance, finally SD
  pop1.Vn <- pop1.prior.sampleSize + n1
  pop1.sigmaN <- ((pop1.prior.sampleSize*pop1.prior.variance) + ((n1 - 1)*var(pop1)) + (((pop1.prior.sa
  prec1 <- rgamma(1,pop1.Vn/2,(pop1.Vn * pop1.sigmaN)/2)
  variance1 = 1/prec1
  simlist[i,2] <- sqrt(variance1)

  #similar to above except uses population 2 parameters
  pop2.Vn <- pop2.prior.sampleSize + n2
  pop2.sigmaN <- ((pop2.prior.sampleSize*pop2.prior.variance) + ((n2 - 1)*var(pop2)) + (((pop2.prior.sa
  prec2 <- rgamma(1,pop2.Vn/2,(pop2.Vn * pop2.sigmaN)/2)
  variance2 = 1/prec2
  simlist[i,3] <- sqrt(variance2)

  #calculates mean for population 1
  #uses normal distribution
  #LOOK HERE

  mu1 <- ((pop1.prior.sampleSize*pop1.prior.mean)+(n1*mean(pop1)))/(pop1.prior.sampleSize+n1)
  var1 <- simlist[i,2]/(sqrt(pop1.prior.sampleSize+n1))
  simlist[i,4] <- rnorm(1,mu1,var1)

  #same as above except uses population 2 parameters
  mu2 <- ((pop2.prior.sampleSize*pop2.prior.mean)+(n2*mean(pop2)))/(pop2.prior.sampleSize+n2)
  var2 <- simlist[i,3]/(sqrt(pop2.prior.sampleSize+n2))
  simlist[i,5] <- rnorm(1,mu2,var2)



}
```

```
#plots and gives means of parameters
plot(simlist[,1], main = "Chain for p", type = "l")
```



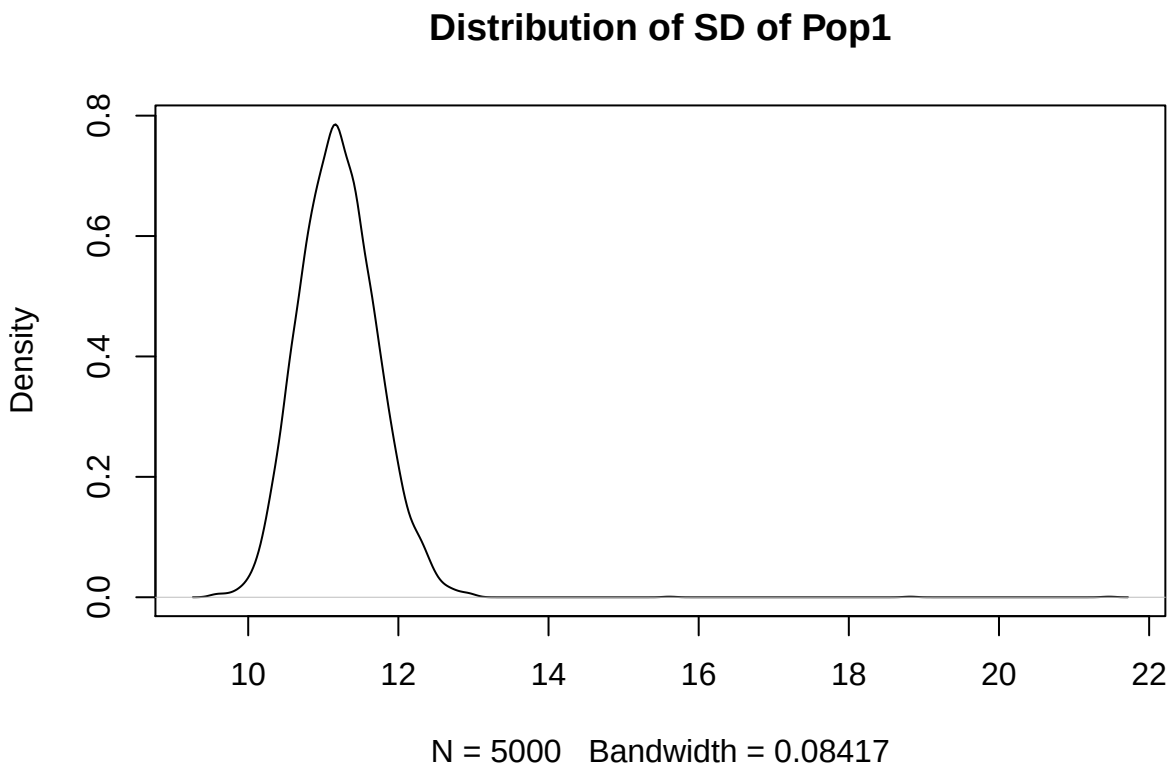

```
plot(density(simlist[,1]), main = "Distribution of p")
```




```
mean(simlist[,1])
```

```
## [1] 0.624364
```

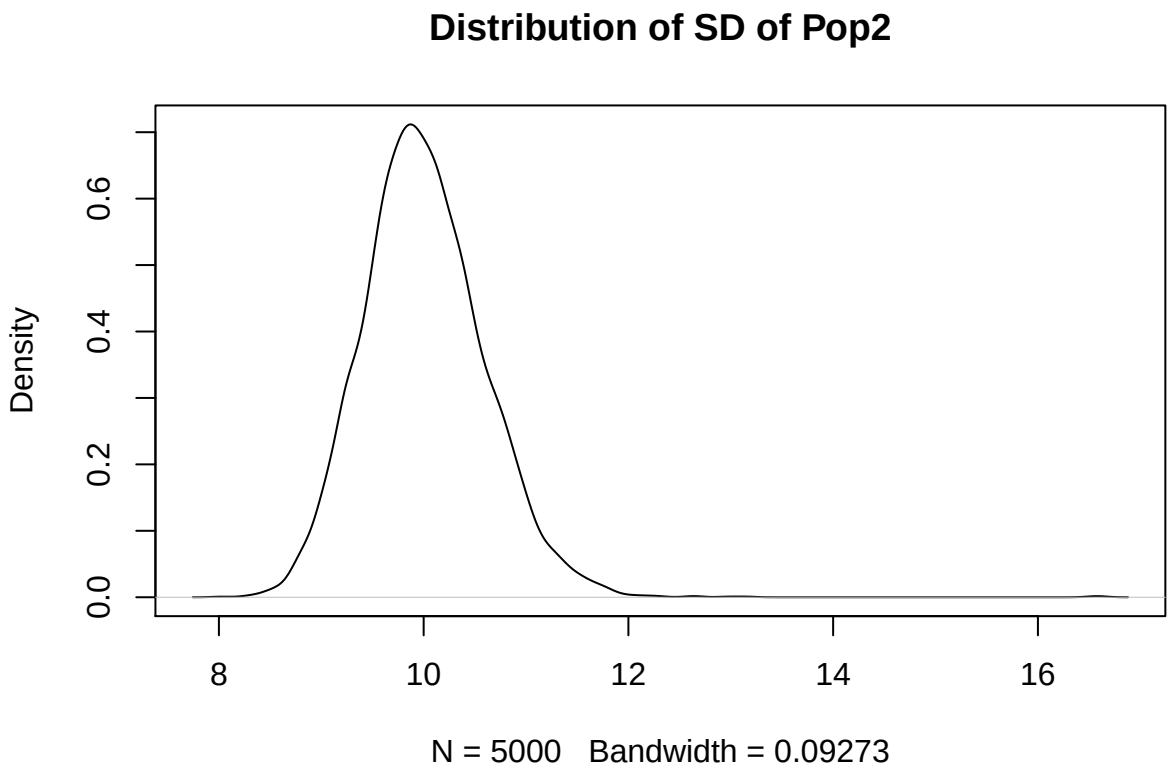
```
plot(density(simlist[,2]),main = "Distribution of SD of Pop1")
```




```
mean(simlist[,2])
```

```
## [1] 11.20203
```

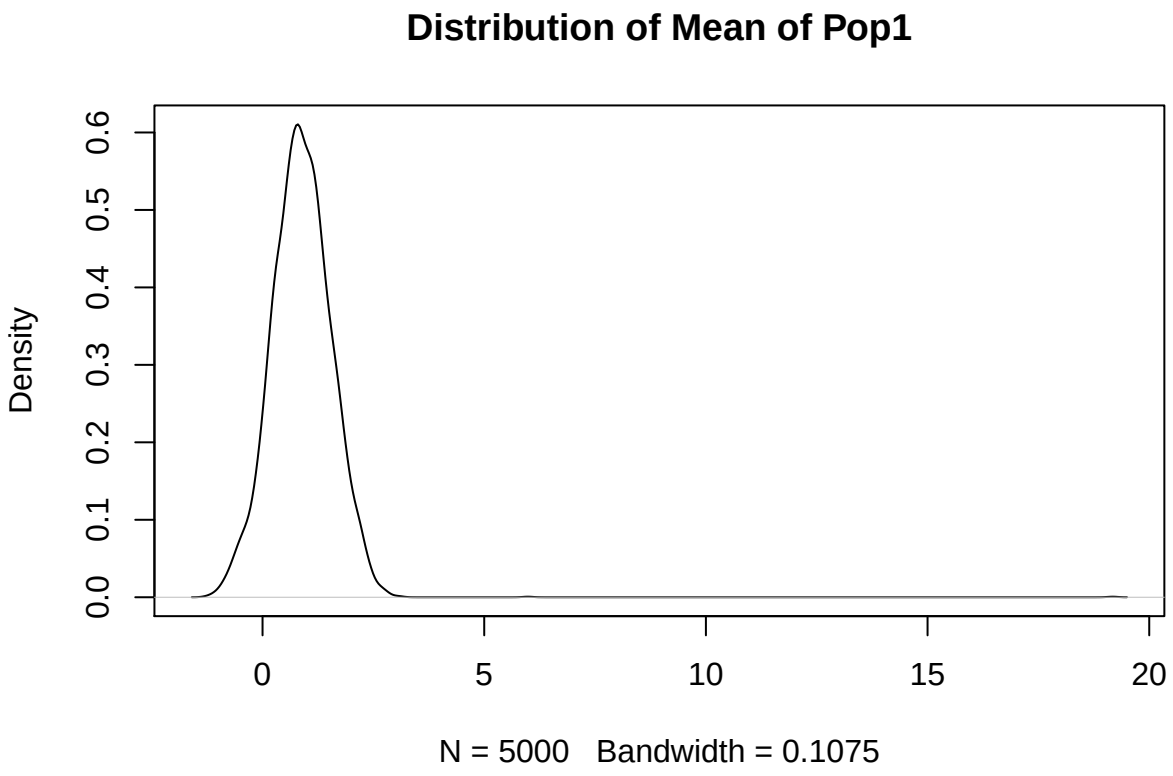
```
plot(density(simlist[,3]),main = "Distribution of SD of Pop2")
```




```
mean(simlist[,3])
```

```
## [1] 10.00705
```

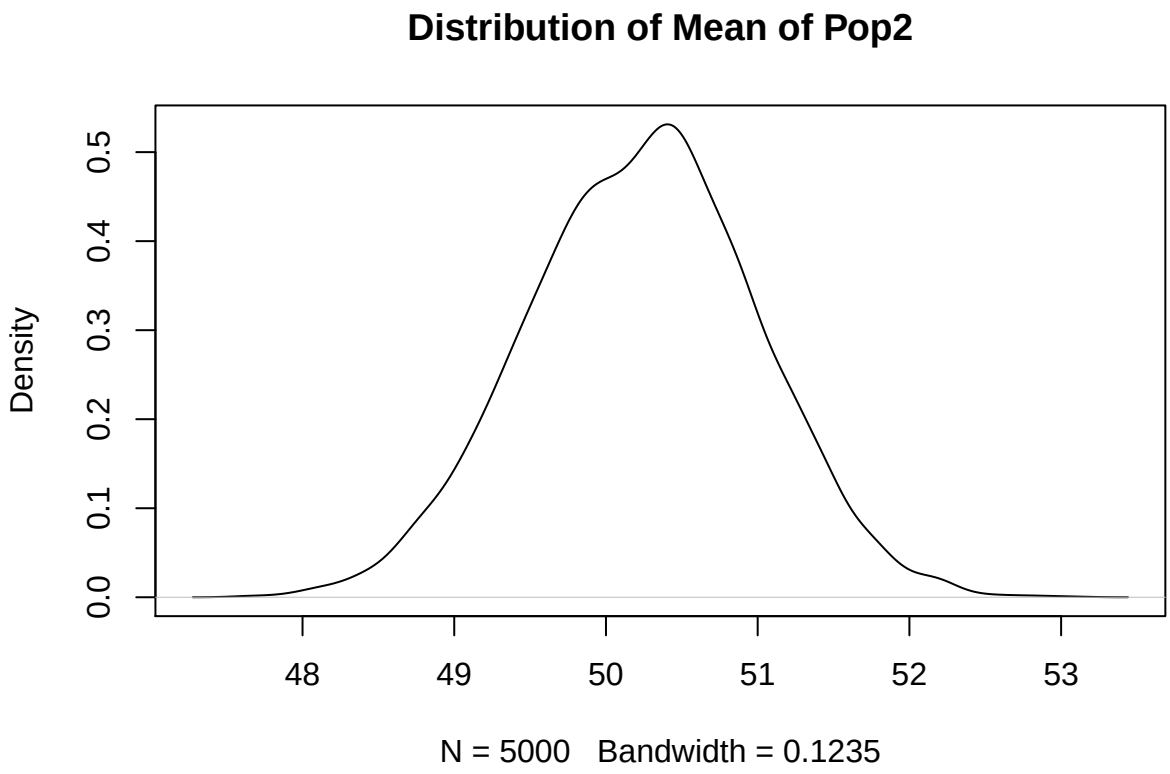
```
plot(density(simlist[,4]),main = "Distribution of Mean of Pop1")
```




```
mean(simlist[,4])
```

```
## [1] 0.8839151
```

```
plot(density(simlist[,5]),main = "Distribution of Mean of Pop2")
```




```
mean(simlist[,5])
```

```
## [1] 50.23737
```