251 Project | Mall Ratings

Derek Walton & Bryce Martin

Introduction

It's a Friday night in Provo and you want to go to a mall. But you realize there are two nearby. How do you decide which to visit? Both the Provo Towne Centre and University Place Mall seem to have a variety of stores and capabilities, making the decision even harder. Our research question thus follows: On average, which mall has better store ratings and which has more consistent ratings? Without having a specific store in mind, one way we can measure which mall is better is by exploring the average store ratings and the variance for the averages at each mall. That would be two parameters for each mall, totaling 4 parameters. This would tell us which mall, on average, is better, while also letting us know how consistent we can expect our experience across each mall to be.

Methods

As there isn't a known posterior distribution for our average store rating data, we use a Monte Carlo approximation to analyze our data. Our prior is an uninformative uniform distribution with parameters 1 to 5 because we wanted to ensure a prior that would least effect our posterior distribution. This gives equal weight to all possible values. For our likelihood, we assume a good approximation may be the beta distribution with alpha = 3 and beta = 1.5, as we imagine the data is more left skewed. We acknowledge that the beta distribution doesn't allow for the endpoints, so we slightly change the data on the endpoints (5 star to 4.999, 1 star to 1.001). Since we aren't predicting how any individual would rate the store, but instead the stores overall rating, we can confidently say that any given true store's rating is not equal to exactly 1 or 5 stars. This allows us to use the beta distribution for our likelihood.

For both malls, our models will be the same.

$$x_i = \text{Average rating for the ith store at a specific mall}$$

$$\text{Data} = x_1, x_2, ..., x_n$$

Prior: $Rating \sim Unif(1, 5)$

Prior: Variance $\sim IG(2.01, 1)$

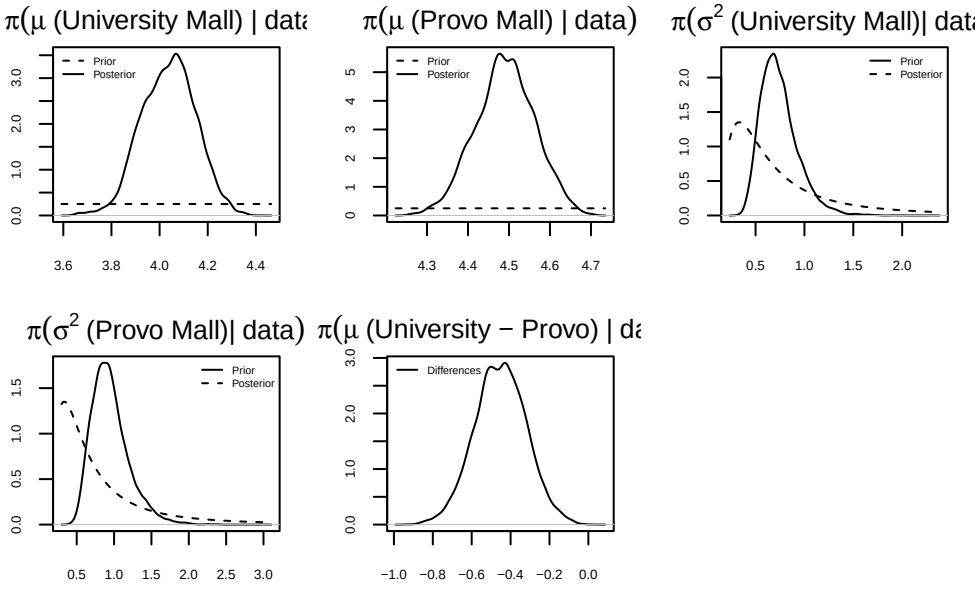
Likelihood: $f(Data|\mu_a, \sigma_a^2) \sim Beta(3, 1.5)$

We randomly selected 30 stores from each malls' website and used the ratings and number of ratings on Google Maps for each of the stores as our sample.

Data Prep

Because we're trying to estimate two parameters of interest, we will use Gibbs Sampling to estimate both the average rating as well as the variance of the ratings.

We are not able to derive the posterior/full conditional distributions for the parameters we are estimating since there is no known distribution for a normal prior/beta likelihood or a double inverse gamma distribution. However we are able to plot them as a 3D graph, and not that there doesn't appear to be any significant correlation between the two for either populations.



We are able to conclude that, since our 95% CIs of our average ratings don't overlap, they are statistically significant. The true average rating for University Mall is between 3.81 and 4.25, and the true average rating for Provo Towne Centre Mall is between 4.34 and 4.63.

However, we aren't able to conclude the same for our variance, since our 95% CIs of our average ratings overlap. The true average variation of ratings for University Mall is between .455 and 1.20, and the true average variation of ratings for Provo Towne Centre Mall is between 0.581 and 1.60.

Conclusions

It's decided. If you have no particular store destination in mind, then the better place is to spend your Friday night is at the Provo Towne Centre Mall. You maybe be concerned that their stores may not be as consistently well rated, however, there is no significant variance of ratings given our prior assumptions and data collected.

Before we collected our data, we assumed a very uninformative prior given that we didn't know how stores would be rated in general (hospitals tend to rate super low, as compared to Chick-fil-a…). Using the uniform distribution allowed our posterior density to be nearly completely dependent on our data. Because of the aforementioned problems with the Beta distribution being non-inclusive on the extremes, our data is admittedly very slightly doctored. However, our research of different distributions didn't come up with a better alternative, so it's the best that we got. The variance we were a little more sure about, but our data moved the variance for both populations to be higher than we had expected.

In future analyses, we can also explore the relationship between our variables and potentially other variables, such as total number of stores, total size of lot, and industry of individual stores.

Appendix (all code)

Part A: Regression

As an aside, we had considered seeing if there is a relationship between the store ratings and number of ratings, helping us know whether having more ratings indicates an overall lower or higher average. As the project would potentially get too long and would be less directly applicable to whether the ratings differed between the malls, we decided exclude this parameter from our analysis. However, for reference and/or fun, here is some of our code and analysis:

The beginning of our model for the relationship between rating and number of ratings is:

$$Y_i \sim \beta_0 + \beta_1 x_i + \epsilon_i$$

$$\text{Where } \epsilon_i \sim N(\mu_1, \sigma_1^2)$$

Given the output of the confidence intervals, we can not confidently say that the log(n) has any significant correlative effect on the ratings. Our 95% confidence interval of our beta value for log(N) is between -.56 and .70.

While the log(N) didn't have any initial significance in our linear regression, given some more data and perhaps some prior knowledge of how stores encourage people to rate, we may be able to discover if there is a more complicated underlying relationship between them all. You can find the code for some of the regression below with the rest of the code.

Part B: Code

```
## Loading all of the libraries
library(brms)
library(invgamma)
library(MASS)
library(plot3D)

##############################

## Loading in all of our data
N <- c(9663, 117, 76, 116, 15, 71, 115, 1, 144, 473, 327, 94, 90, 6, 178, 61, 480, 432, 513,
logN <- log(N)
Rating1 = c(4.6, 3.4, 3.8, 4.2, 3.5, 4.2, 3.2, 1.001, 3.9, 4.4, 4.8, 3.4, 4.6, 4.999, 4.4, 4
pop1 <- data.frame(Rating = Rating1, logN = logN, Population = "Pop1")

N <- c(55, 78, 103, 59, 1596, 54, 12, 26, 1, 310, 8, 1448, 1, 40, 1, 13, 51, 28, 25, 2388, 1
logN <- log(N)
Rating2 = c(4.5, 4.9, 4.3, 4.8, 4.2, 4.3, 4.999, 4.999, 4.999, 3.9, 4.999, 4.1, 4.999, 4.6,
pop2 <- data.frame(Rating = Rating2, logN = logN, Population = "Pop2" )

data <- rbind(pop1, pop2)

##############################

# ## Fit Bayesian linear regression model
```

```r
fit <- brm(
  formula = Rating ~ logN + (1 + logN | Population),  # Random intercept and slope for Popula
  data = data,
  family = gaussian(),  # Normal likelihood
  prior = c(
    prior(normal(4, 2), class = "Intercept"),
    prior(normal(0, 10), class = "b"),
    prior(cauchy(0, 20), class = "sd")  # Prior on group-level standard deviations
  ),
  iter = 4000,  # Number of iterations
  chains = 4,   # Number of MCMC chains
  seed = 123    # For reproducibility
)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 8.8e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.88 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 3.842 seconds (Warm-up)
Chain 1:                5.432 seconds (Sampling)
Chain 1:                9.274 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
```

```
Chain 2: Gradient evaluation took 2.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
Chain 2: Iteration: 1200 / 4000 [ 30%]  (Warmup)
Chain 2: Iteration: 1600 / 4000 [ 40%]  (Warmup)
Chain 2: Iteration: 2000 / 4000 [ 50%]  (Warmup)
Chain 2: Iteration: 2001 / 4000 [ 50%]  (Sampling)
Chain 2: Iteration: 2400 / 4000 [ 60%]  (Sampling)
Chain 2: Iteration: 2800 / 4000 [ 70%]  (Sampling)
Chain 2: Iteration: 3200 / 4000 [ 80%]  (Sampling)
Chain 2: Iteration: 3600 / 4000 [ 90%]  (Sampling)
Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 3.805 seconds (Warm-up)
Chain 2:                2.155 seconds (Sampling)
Chain 2:                5.96 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 3e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.3 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
Chain 3: Iteration: 1200 / 4000 [ 30%]  (Warmup)
Chain 3: Iteration: 1600 / 4000 [ 40%]  (Warmup)
Chain 3: Iteration: 2000 / 4000 [ 50%]  (Warmup)
Chain 3: Iteration: 2001 / 4000 [ 50%]  (Sampling)
Chain 3: Iteration: 2400 / 4000 [ 60%]  (Sampling)
Chain 3: Iteration: 2800 / 4000 [ 70%]  (Sampling)
Chain 3: Iteration: 3200 / 4000 [ 80%]  (Sampling)
Chain 3: Iteration: 3600 / 4000 [ 90%]  (Sampling)
Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
Chain 3:
```

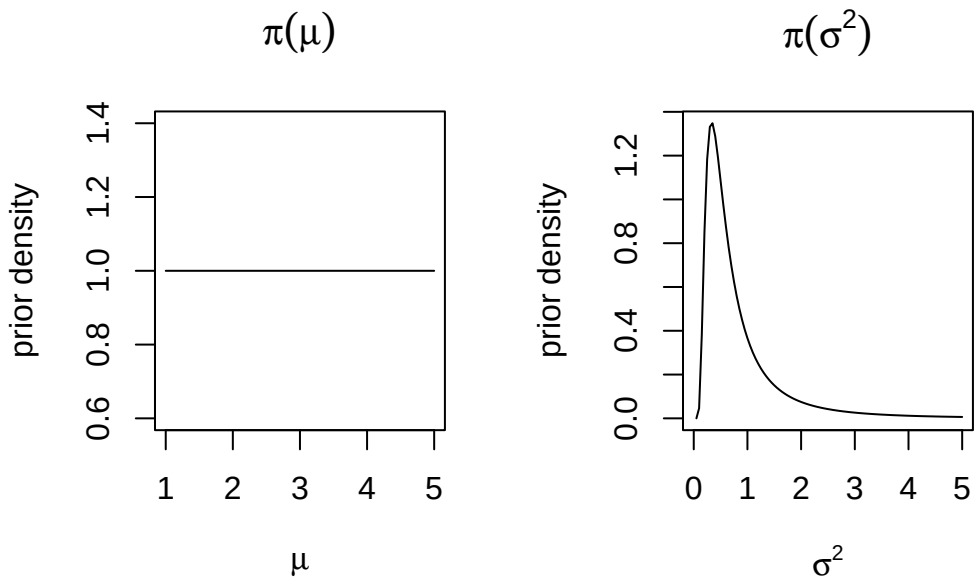
```
Chain 3:  Elapsed Time: 3.981 seconds (Warm-up)
Chain 3:                3.527 seconds (Sampling)
Chain 3:                7.508 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 2.7e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.27 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
Chain 4: Iteration: 1200 / 4000 [ 30%]  (Warmup)
Chain 4: Iteration: 1600 / 4000 [ 40%]  (Warmup)
Chain 4: Iteration: 2000 / 4000 [ 50%]  (Warmup)
Chain 4: Iteration: 2001 / 4000 [ 50%]  (Sampling)
Chain 4: Iteration: 2400 / 4000 [ 60%]  (Sampling)
Chain 4: Iteration: 2800 / 4000 [ 70%]  (Sampling)
Chain 4: Iteration: 3200 / 4000 [ 80%]  (Sampling)
Chain 4: Iteration: 3600 / 4000 [ 90%]  (Sampling)
Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 4.273 seconds (Warm-up)
Chain 4:                6.113 seconds (Sampling)
Chain 4:                10.386 seconds (Total)
Chain 4:
```

```
summary(fit)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: Rating ~ logN + (1 + logN | Population)
   Data: data (Number of observations: 60)
  Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
         total post-warmup draws = 8000

Multilevel Hyperparameters:
~Population (Number of levels: 2)
                    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
sd(Intercept)           3.76      3.59     0.33    13.33 1.00      881     2571
sd(logN)                0.65      0.73     0.02     2.70 1.01      509      378
cor(Intercept,logN)    -0.20      0.59    -0.99     0.92 1.00     1125      549

Regression Coefficients:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     3.67      1.56     0.50     7.02 1.01      967     2000
logN          0.08      0.29    -0.56     0.70 1.01      851     1100

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     0.80      0.08     0.67     0.97 1.00     2155     2956

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
############################

## Bayesian analysis

# PRIOR PARAMETERS
# Prior parameters for mu:
alpha <- 3
beta <- 1.5

# Prior parameters for sigma2:
gamma <- 2.01
phi <- 1

# Plot the prior distributions to make sure they seem reasonable
par(mfrow=c(1,2))

curve(1+4*dbeta(x, alpha, beta), xlim=c(1, 5), ylab="prior density", main=expression(pi(mu))

curve(dinvgamma(x, gamma, phi), xlim=c(0, 5), ylab="prior density", main=expression(pi(sigma
```




```
# COLLECT DATA
pop1 <- Rating1
pop2 <- Rating2
n <- length(pop1)

# POSTERIOR DISTRIBUTIONS: Must use Gibbs Sampling Algorithm to approximate
#Starting values
mu <- 1
sigma2 <- 1

# initializations for the Gibbs Sampling Algorithm
iters <- 10000
mu.1.save <- rep(0, iters)
mu.1.save[1] <- mu
sigma2.1.save <- rep(0, iters)
sigma2.1.save[1] <- sigma2

### Added for Metropolis RW
accept.mu <- 0
s.1.mu <- .32

#Gibbs Sampling Algorithm For Pop1 (University Mall)
for(t in 2:iters){
```

```
  # Use Metropolis RW to draw from the full conditional distribution
  mu.star <- rnorm(1, mu, s.1.mu)
  mu.adj <- (mu-1)/4
  mu.star.adj <-(mu.star-1)/4

  if (0 < mu.star.adj && mu.star.adj < 1){
    beta.norm = (alpha - mu.adj*alpha)/mu.adj
    beta.star = (alpha - mu.star.adj*alpha)/mu.star.adj

    log.r <- sum(dbeta((pop1-1)/4, alpha, beta.star, log=T)) +
             dunif(mu.star.adj, 0, 1, log=T) -
             sum(dbeta((pop1-1)/4, alpha, beta.norm, log=T)) -
             dunif(mu.adj, 0, 1, log=T)

    logu <- log(runif(1))
    if(logu < log.r){
        mu <- mu.star
        accept.mu <- accept.mu + 1
    }
  }

  mu.1.save[t] <- mu

  # full conditional of sigma2 (update the value of the parameters)
  gamma.p <- gamma + n/2
  phi.p <- phi + sum((pop1 - mu)^2 )/2

  #sample and save new value of sigma2
  sigma2 <- rinvgamma(1, gamma.p, phi.p)
  sigma2.1.save[t] <- sigma2

}

##Added to check acceptance rates
accept.mu/iters
```

```
[1] 0.3996
```

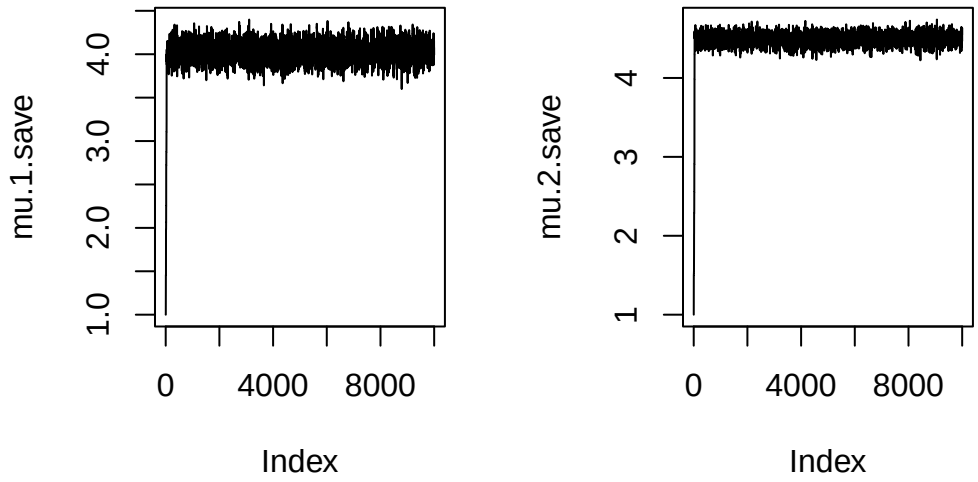
```
# Reinit for Pop2 Approximation
mu <- 1
sigma2 <- 1
```

```
mu.2.save <- rep(0, iters)
mu.2.save[1] <- mu
sigma2.2.save <- rep(0, iters)
sigma2.2.save[1] <- sigma2

accept.mu <- 0
s.2.mu <- .22

#Gibbs Sampling Algorithm For Pop2 (Provo Towne Centre Mall)
for(t in 2:iters){
  # Use Metropolis RW to draw from the full conditional distribution
  mu.star <- rnorm(1, mu, s.2.mu)
  mu.adj <- (mu-1)/4
  mu.star.adj <-(mu.star-1)/4

  if (0 < mu.star.adj && mu.star.adj < 1){
    beta.norm = (alpha - mu.adj*alpha)/mu.adj
    beta.star = (alpha - mu.star.adj*alpha)/mu.star.adj

    log.r <- sum(dbeta((pop2-1)/4, alpha, beta.star, log=T)) +
             dunif(mu.star.adj, 0, 1, log=T) -
             sum(dbeta((pop2-1)/4, alpha, beta.norm, log=T)) -
             dunif(mu.adj, 0, 1, log=T)

    logu <- log(runif(1))
    if(logu < log.r){
        mu <- mu.star
        accept.mu <- accept.mu + 1
    }
  }

  mu.2.save[t] <- mu

  # full conditional of sigma2 (update the value of the parameters)
  gamma.p <- gamma + n/2
  phi.p <- phi + sum((pop1 - mu)^2 )/2

  #sample and save new value of sigma2
  sigma2 <- rinvgamma(1, gamma.p, phi.p)
  sigma2.2.save[t] <- sigma2

}
```

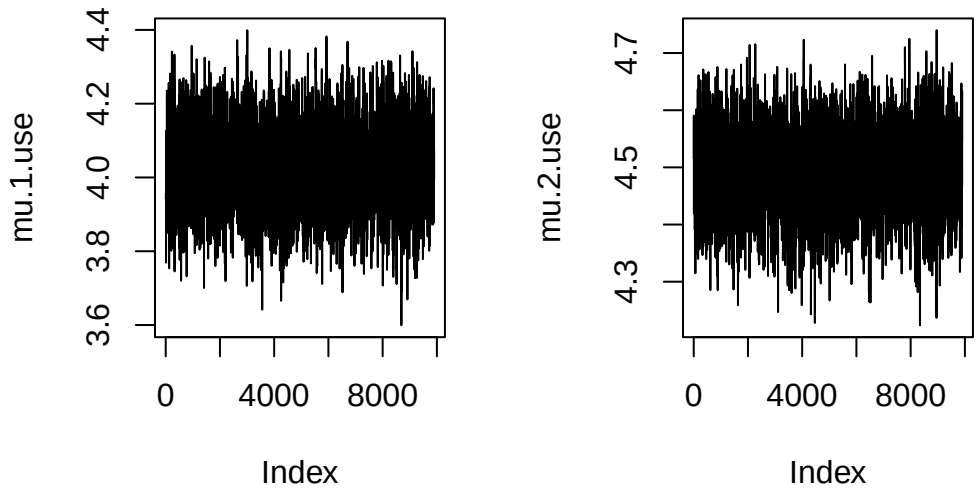
```
accept.mu/iters
```

```
[1] 0.3765
```

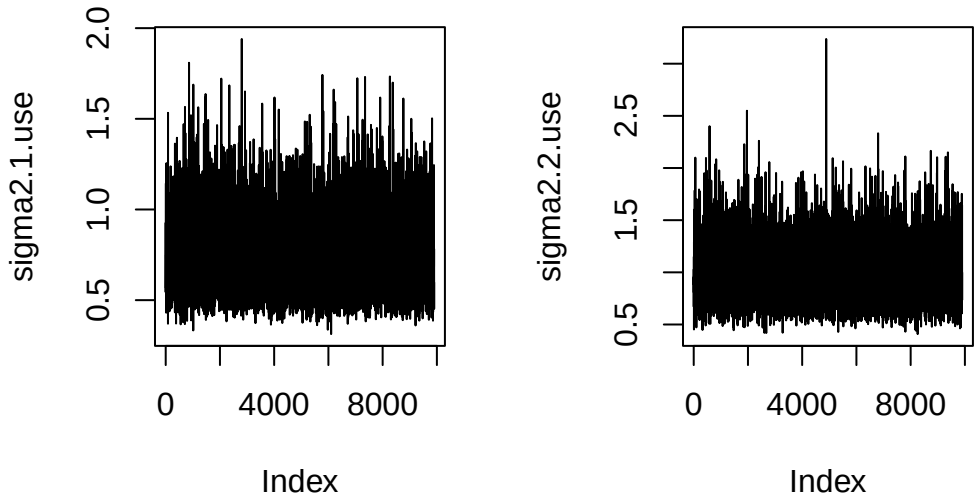
```
# Look at trace plots to determine burn-in
par(mfrow=c(1,2))
plot(mu.1.save, type='l')
plot(mu.2.save, type='l')
```



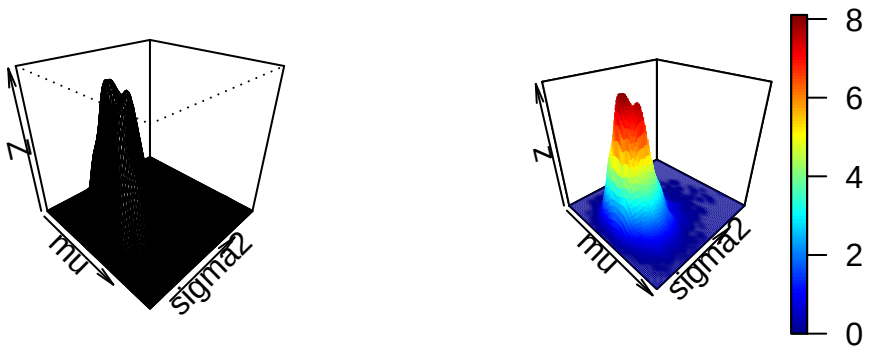

```
#throw out the first few values
burn <- 100
mu.1.use <- mu.1.save[-(1:burn)]
sigma2.1.use <- sigma2.1.save[-(1:burn)]
mu.2.use <- mu.2.save[-(1:burn)]
sigma2.2.use <- sigma2.2.save[-(1:burn)]

# Look at updated trace plots
par(mfrow=c(1,2))
plot(mu.1.use, type='l')
plot(mu.2.use, type='l')
```



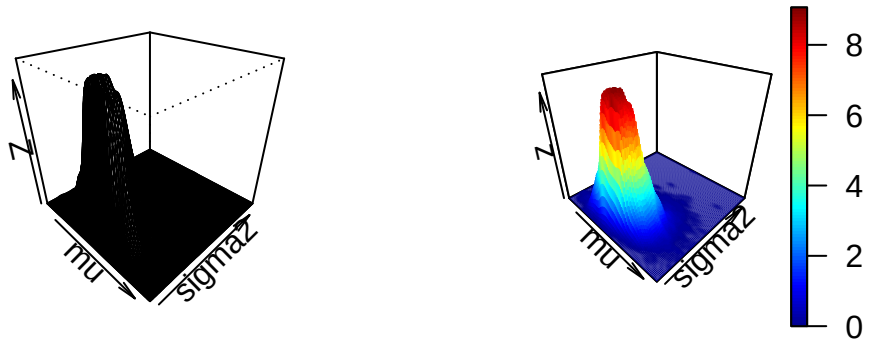

```
par(mfrow=c(1,2))
plot(sigma2.1.use, type='l')
plot(sigma2.2.use, type='l')
```



```
#SUMMARIZE THE POSTERIOR DISTRIBUTION(S)

# JOINT POSTERIOR distribution of mu and sigma2 (for both populations)
par(mfrow=c(1,2))
joint.dens <- kde2d(mu.1.use, sigma2.1.use, n=100)
persp(joint.dens, xlab="mu", ylab="sigma2", phi=30, theta=45)
hist3D(joint.dens$x, joint.dens$y, joint.dens$z, xlab="mu", ylab="sigma2", phi=30, theta=45)
```




```
joint.dens <- kde2d(mu.2.use, sigma2.2.use, n=100)
persp(joint.dens, xlab="mu", ylab="sigma2", phi=30, theta=45)
hist3D(joint.dens$x, joint.dens$y, joint.dens$z, xlab="mu", ylab="sigma2", phi=30, theta=45)
```




```r
# Adjusting parameters for tighter layout
par(mfrow=c(2,3),    # Fit 6 plots (2 rows, 3 columns)
    cex=0.7,         # Reduce text and symbol size
    cex.axis=0.6,    # Smaller axis labels
    cex.lab=0.7,     # Smaller axis titles
    mar=c(3, 3, 2, 1),  # Smaller plot margins (bottom, left, top, right)
    oma=c(1, 1, 1, 1))  # Smaller outer margins

# POSTERIOR distribution of mu
plot(density(mu.1.use), xlab=expression(mu), ylab="density",
     main=expression(pi(mu~"(University Mall) |"~data)))
curve(dunif(x, 1, 5), lty=2, add=T)
legend("topleft", c("Prior", "Posterior"), lty=c(2, 1), cex=0.5, bty="n")

plot(density(mu.2.use), xlab=expression(mu), ylab="density",
     main=expression(pi(mu~"(Provo Mall) |"~data)))
curve(dunif(x, 1, 5), lty=2, add=T)
legend("topleft", c("Prior", "Posterior"), lty=c(2, 1), cex=0.5, bty="n")

# POSTERIOR distribution of sigma2
plot(density(sigma2.1.use), xlab=expression(sigma^2),
     main=expression(pi(sigma^2~"(University Mall)|"~data)))
curve(dinvgamma(x, gamma, phi), add=T, lty=2)
```

```r
legend("topright", c("Prior", "Posterior"), lty=c(1, 2), cex=0.5, bty="n")

plot(density(sigma2.2.use), xlab=expression(sigma^2),
     main=expression(pi(sigma^2~"(Provo Mall)|"~data)))
curve(dinvgamma(x, gamma, phi), add=T, lty=2)
legend("topright", c("Prior", "Posterior"), lty=c(1, 2), cex=0.5, bty="n")

# POSTERIOR distribution of the difference (mu.1 - mu.2)
diffs <- mu.1.use - mu.2.use
plot(density(diffs), xlab=expression(mu), ylab="density",
     main=expression(pi(mu~"(University Mall - Provo) |"~data)))
legend("topleft", c("Differences"), lty=c(1), cex=0.5, bty="n")


#95% credible interval
print("Mean, variance, and 95% CI for average rating of University Mall")
```

```
[1] "Mean, variance, and 95% CI for average rating of University Mall"
```

```r
mean(mu.1.use)
```

```
[1] 4.034684
```

```r
var(mu.1.use)
```

```
[1] 0.01261505
```

```r
quantile(mu.1.use, c(.025, .975))
```

```
    2.5%    97.5%
3.817875 4.250732
```

```r
print("Mean, variance, and 95% CI for average rating of Provo Towne Centre Mall")
```

```
[1] "Mean, variance, and 95% CI for average rating of Provo Towne Centre Mall"
```

```r
mean(mu.2.use)
```

```
[1] 4.493511
```

```r
var(mu.2.use)
```

```
[1] 0.005416531
```

```r
quantile(mu.2.use, c(.025, .975))
```

```
    2.5%    97.5%
4.348952 4.631366
```

```r
# Given our data and prior knowledge, there is a 95% chance that
# the true average rating for University Mall is between 3.81 and 4.25 and
# the true average rating for Provo Towne Centre Mall is between 4.34 and 4.63


#posterior mean of the average variance in mall ratings
#95% credible interval for sigma2 for University Mall
print("Mean, variance, and 95% CI for variance of ratings at University Mall")
```

```
[1] "Mean, variance, and 95% CI for variance of ratings at University Mall"
```

```r
mean(sigma2.1.use)
```

```
[1] 0.7404992
```

```r
var(sigma2.1.use)
```

```
[1] 0.0362967
```

```r
quantile(sigma2.1.use, c(.025, .975))
```

```
     2.5%     97.5%
0.4525018 1.2019114
```

```
#95% credible interval for sigma2 for Provo Towne Centre Mall
print("Mean, variance, and 95% CI for variance of ratings at Provo Towne Centre Mall")
```

```
[1] "Mean, variance, and 95% CI for variance of ratings at Provo Towne Centre Mall"
```

```
mean(sigma2.2.use)
```

```
[1] 0.9590687
```

```
var(sigma2.2.use)
```

```
[1] 0.06543397
```

```
quantile(sigma2.2.use, c(.025, .975))
```

```
     2.5%     97.5%
0.5759825 1.5723806
```

```
#While the variation in the plots may look different, their 95% confidence intervals overlap
```