LD50 in Rats

Greg Johnson

An animal experiment administers k dose levels x_i to n_i rats resulting in some sort of positive outcome y_i. Under exchangeability, the data points y_i are binomially distributed:

$$y_i|\theta_i \sim Bin(n_i, \theta_i)$$

We introduce the covariate x_i using a logit dose-response relation - the result is a logistic regression model:

$$\text{logit}(\theta_i) = \alpha + \beta x_i$$

This results in the likelihood:

$$p(\mathbf{y}|\alpha, \beta, \mathbf{n}, \mathbf{x}) = \prod_i p(y_i|\alpha, \beta, \mathbf{n}, \mathbf{x}) = \prod_i \binom{n_i}{y_i} \left[\text{logit}^{-1}(\alpha + \beta x_i)\right]^{y_i} \left[1 - \text{logit}^{-1}(\alpha + \beta x_i)\right]^{n_i - y_i}$$

Gaussian Prior and Grid Sampling

Based on prior research we parameterize our knowledge about the parameters α and β as jointly normal.

$$(\alpha, \beta) \sim N_2\left(\mu = \begin{pmatrix} 4 \\ 10 \end{pmatrix}, \Sigma = \begin{pmatrix} 4 & 10 \\ 10 & 100 \end{pmatrix}\right)$$

This gives us the posterior density:

$$p(\alpha, \beta|\mathbf{y}, \mathbf{n}, \mathbf{x}) \propto p(\alpha, \beta|\mathbf{n}, \mathbf{x}) p(\mathbf{y}|\alpha, \beta, \mathbf{n}, \mathbf{x}) = p(\alpha, \beta|\mathbf{n}, \mathbf{x}) \prod_i p(y_i|\alpha, \beta, n_i, x_i)$$

$$p(\alpha, \beta|\mathbf{y}, \mathbf{n}, \mathbf{x}) \propto N_2\left(\mu = \begin{pmatrix} 4 \\ 10 \end{pmatrix}, \Sigma = \begin{pmatrix} 4 & 10 \\ 10 & 100 \end{pmatrix}\right) \times \prod_i \left[\text{logit}^{-1}(\alpha + \beta x_i)\right]^{y_i} \left[1 - \text{logit}^{-1}(\alpha + \beta x_i)\right]^{n_i - y_i}$$

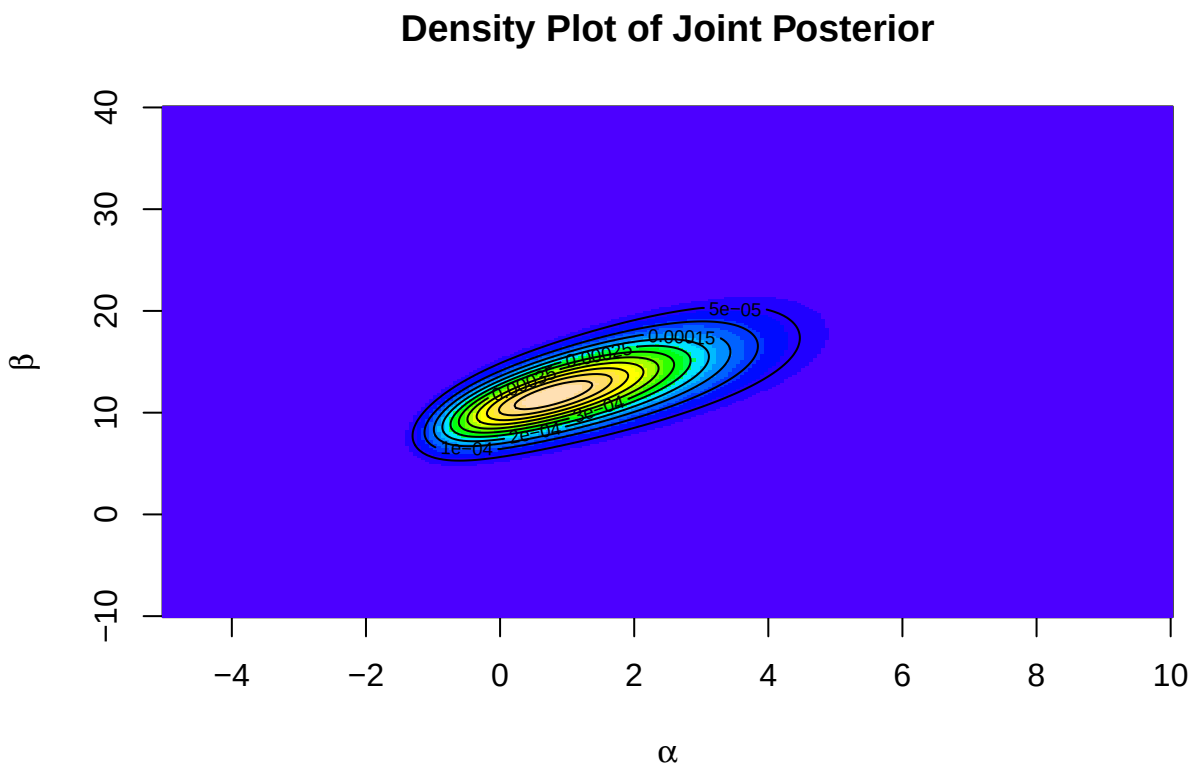
This bivariate normal prior is non-conjugate so instead of an analytic Monte Carlo approach, we can use a simple two-dimensional grid-sampling approach.

Approximate Joint Posterior

First we compute the unnormalized posterior density at a grid of values over the effect range of (α, β). We use the range $(\alpha, \beta) \in [-5, 10] \times [-10, 40]$.

```
# data and logistic function
x = c(-0.86, -0.3, -0.05, 0.73)
n = c(5, 5, 5, 5)
y = c(0, 1, 3, 5)
logitinv = function(r) {
    exp(r)/(1 + exp(r))
}
```

```r
# unnormalized posterior
unnorm_post = function(alpha, beta) {
    prior = dmvnorm(c(alpha, beta), mean = c(4, 10), sigma = matrix(c(4, 10,
        10, 100), 2))
    like = prod((logitinv(alpha + beta * x))^y * (1 - logitinv(alpha + beta *
        x))^(n - y))
    prior * like
}


# create alpha, beta grid set alpha,beta range
alphalim = c(-5, 10)
betalim = c(-10, 40)
### grid size
gsize = 250
agrid = seq(alphalim[1], alphalim[2], length.out = gsize)
astep = agrid[2] - agrid[1]
bgrid = seq(betalim[1], betalim[2], length.out = gsize)
bstep = bgrid[2] - bgrid[1]
post_grid = matrix(0, gsize, gsize)

# grid of unnormalized posterior
for (i in 1:gsize) {
    for (j in 1:gsize) {
        post_grid[i, j] = unnorm_post(agrid[j], bgrid[i])
    }
}

# normalize the grid
norm_post_grid = post_grid/sum(post_grid)

# marginal of alpha
alpha_post_grid = apply(norm_post_grid, 2, sum)
# take a look at it
image(agrid, (bgrid), norm_post_grid, col = topo.colors(20), xlab = expression(alpha),
    ylab = expression(beta), main = "Density Plot of Joint Posterior")
contour(agrid, bgrid, norm_post_grid, add = TRUE)
```

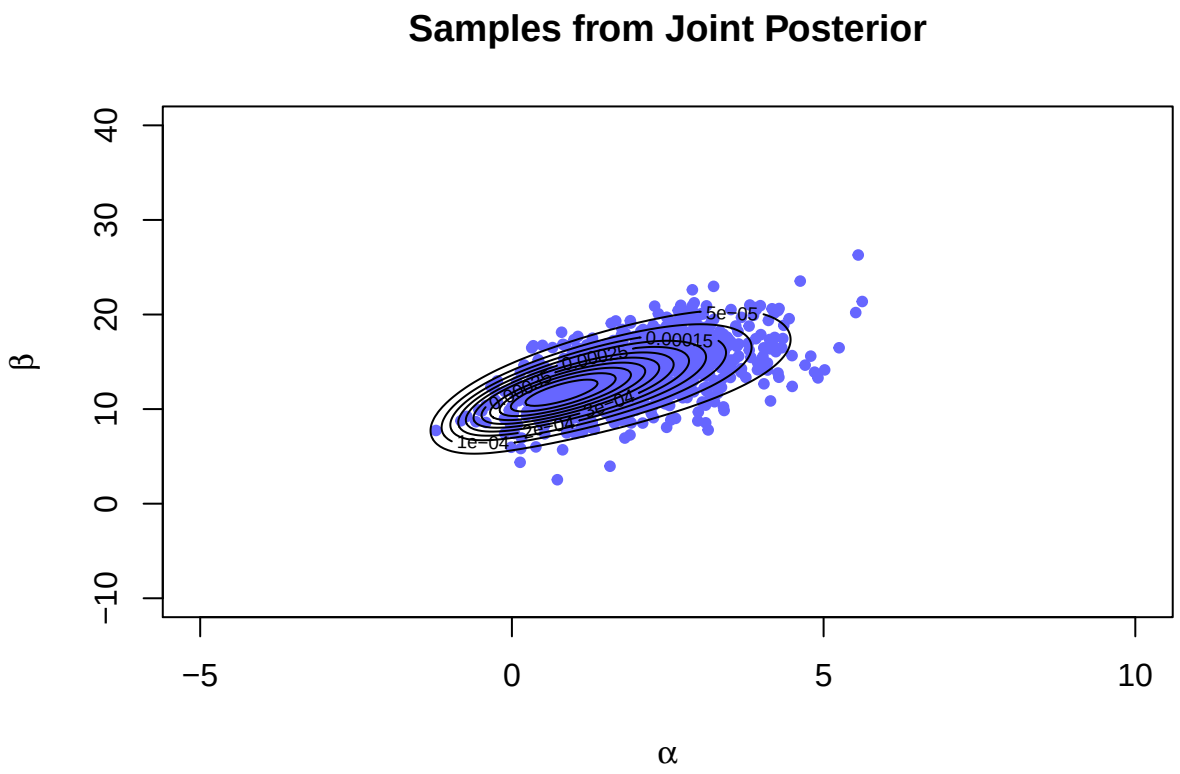


Grid Sample from Joint Posterior

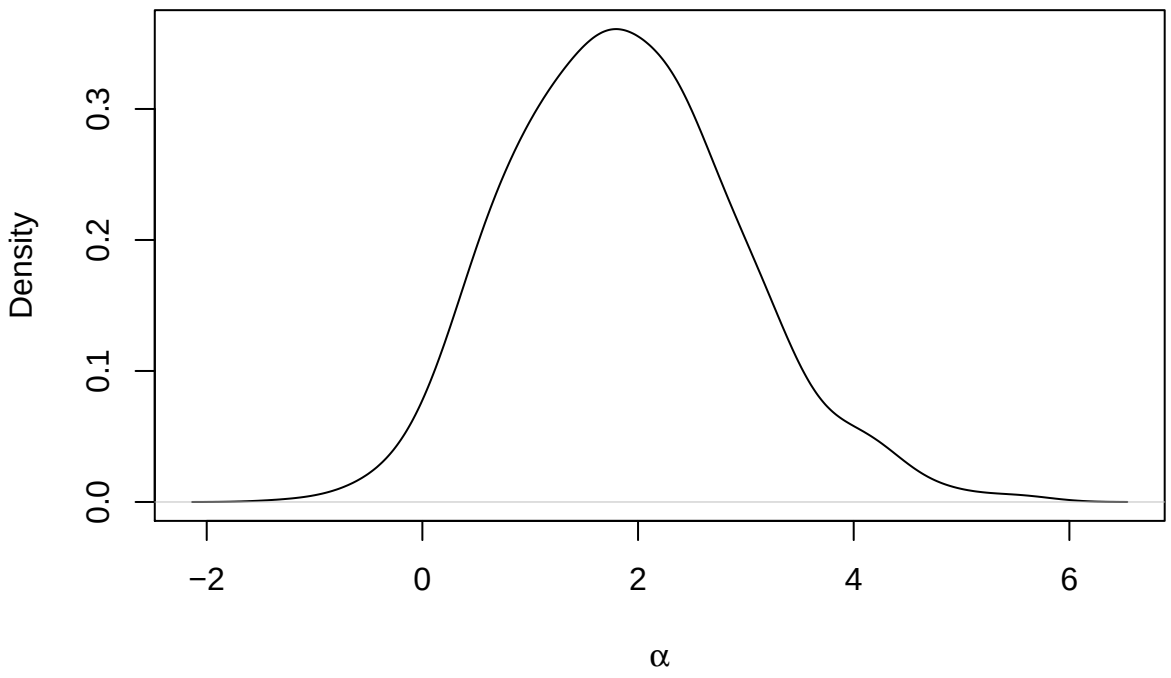
```
#number of samples
B=1000
#sample marginal of alpha
place = sample.int(gsize,B,prob=alpha_post_grid,replace=TRUE)
alpha_post_sample = agrid[place]

#sample from conditional of beta
beta_post_sample = place2 = numeric()
for(b in 1:B){
  place2[b] = sample.int(gsize,1,prob=norm_post_grid[place[b],],replace=TRUE)
  beta_post_sample[b] = bgrid[place2[b]]
}

#add random jitter to make continuous
alpha_post_sample = alpha_post_sample+runif(B,-astep/2,astep/2)
beta_post_sample = beta_post_sample + runif(B,-bstep/2,bstep/2)
```

Plot Joint and Marginal Posteriors

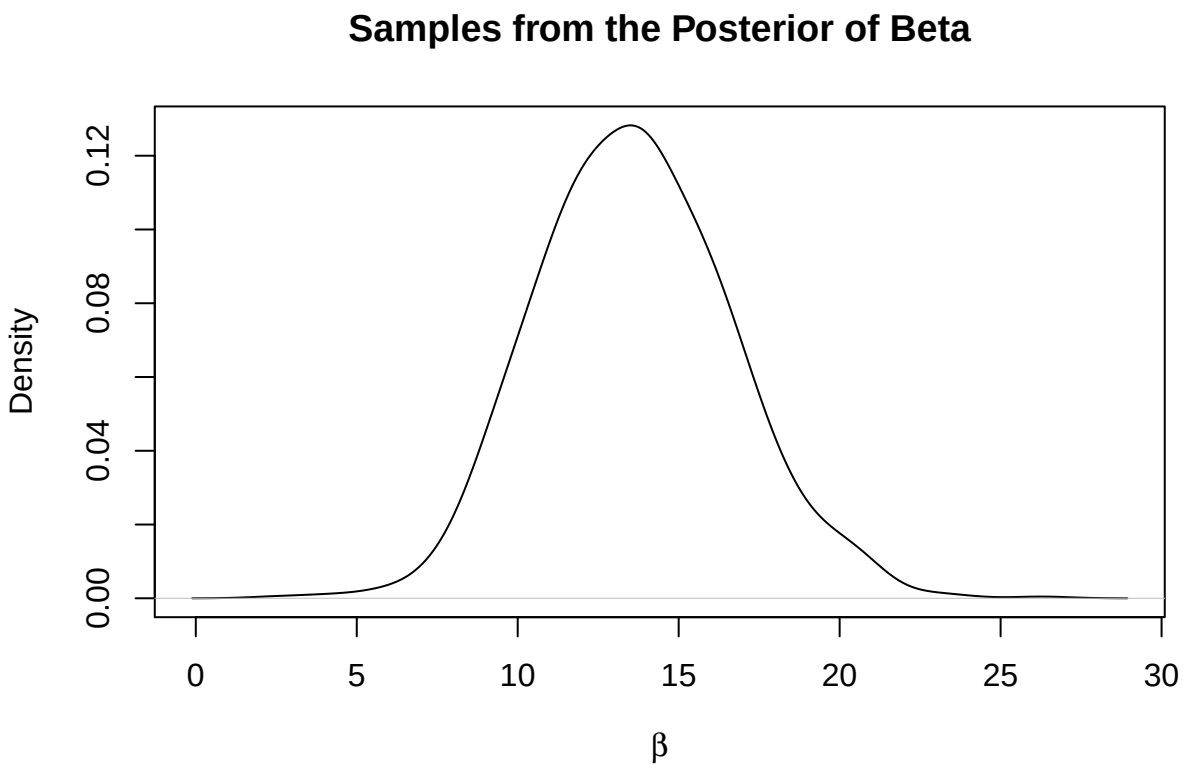
```
plot(alpha_post_sample, beta_post_sample, col = rgb(0.4, 0.4, 1, 1), pch = 20,
    ylim = c(-10, 40), xlim = c(-5, 10), main = "Samples from Joint Posterior",
    xlab = expression(alpha), ylab = expression(beta))
contour(agrid, bgrid, norm_post_grid, add = TRUE)
```




```
plot(density(alpha_post_sample, adjust = 1.3), main = "Samples from the Posterior of Alpha",
    xlab = expression(alpha))
```

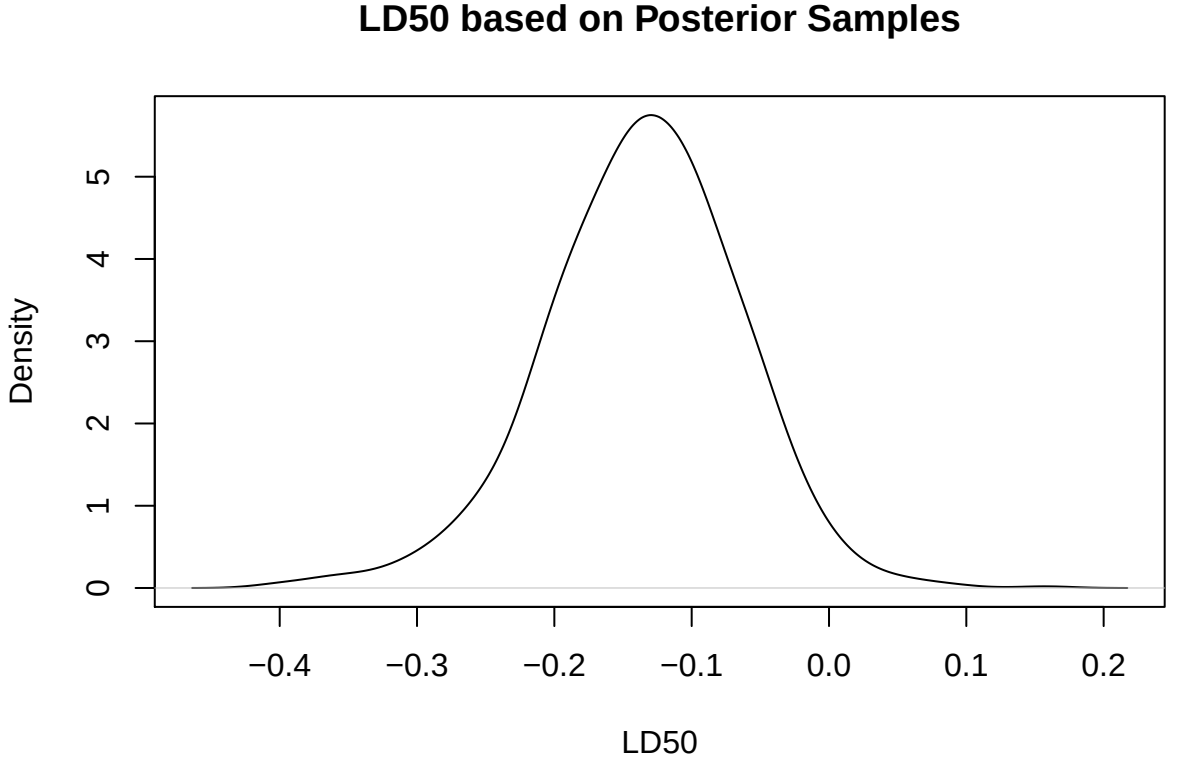



```
plot(density(beta_post_sample, adjust = 1.3), main = "Samples from the Posterior of Beta",
    xlab = expression(beta))
```



Posterior Distribution of LD50

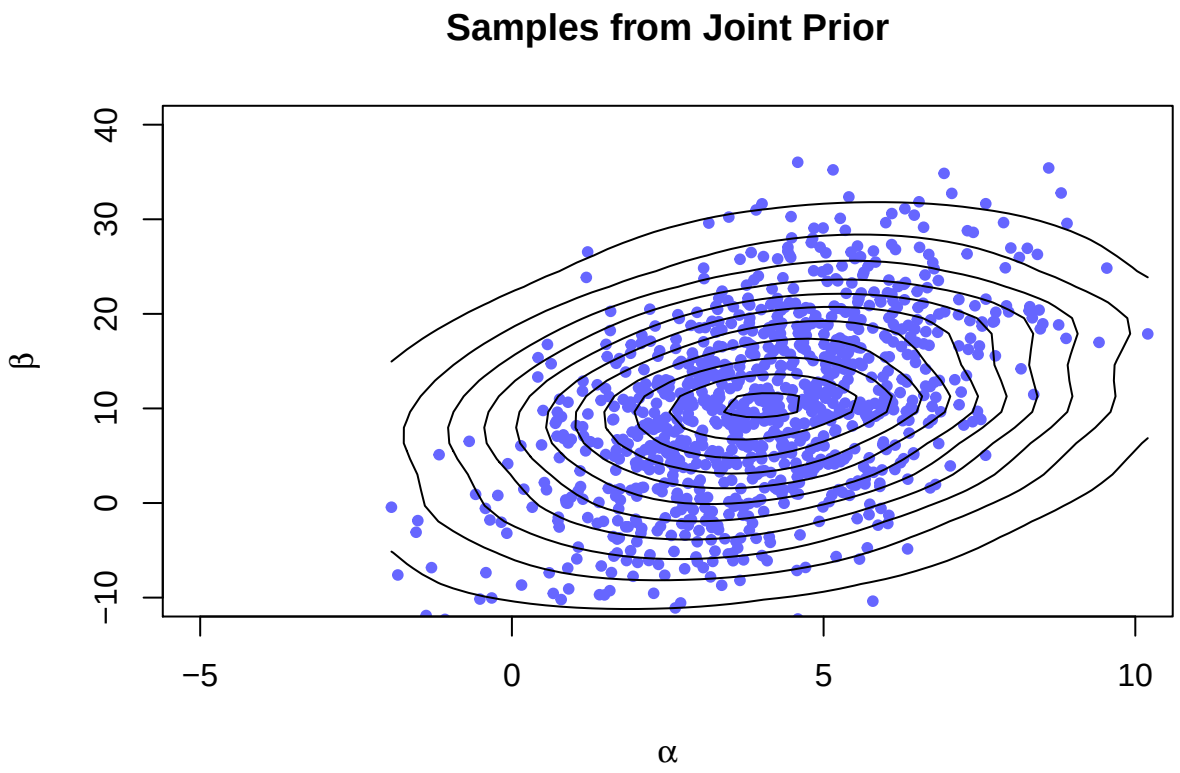
```
plot(density(-alpha_post_sample/beta_post_sample, adjust = 1.3), type = "l",
    main = "LD50 based on Posterior Samples", xlab = "LD50")
```



Let's take a look at the contour plot and scatterplot to see if our posterior looks like a compromise between the prior and the likelihood (which it should!).

Prior

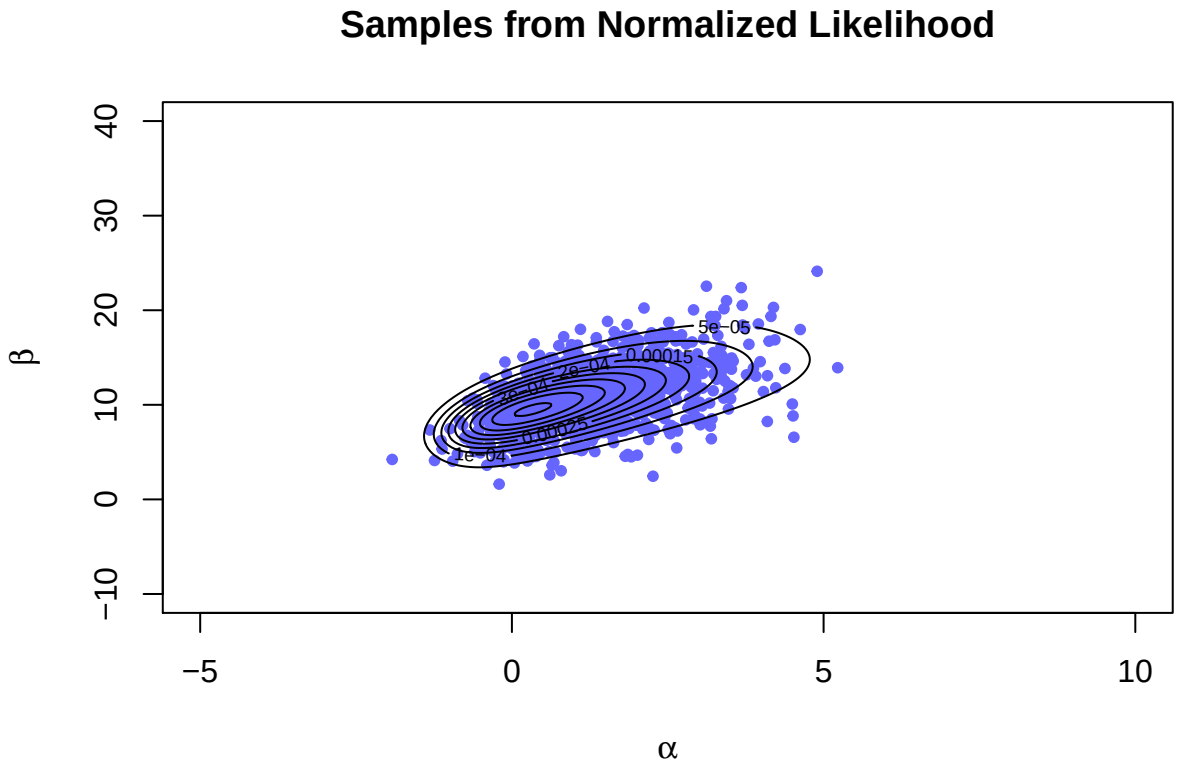
```
prior_sample = rmvnorm(1000, mean = c(4, 10), sigma = matrix(c(4, 10, 10, 100),
    2))
plot(prior_sample[, 1], prior_sample[, 2], col = rgb(0.4, 0.4, 1, 1), pch = 20,
    ylim = c(-10, 40), xlim = c(-5, 10), main = "Samples from Joint Prior",
    xlab = expression(paste(alpha)), ylab = expression(paste(beta)))
contour(kde2d(prior_sample[, 1], prior_sample[, 2], h = c(10, 10), n = 50),
    drawlabels = FALSE, nlevels = 11, add = TRUE)
```



Normalized Likelihood

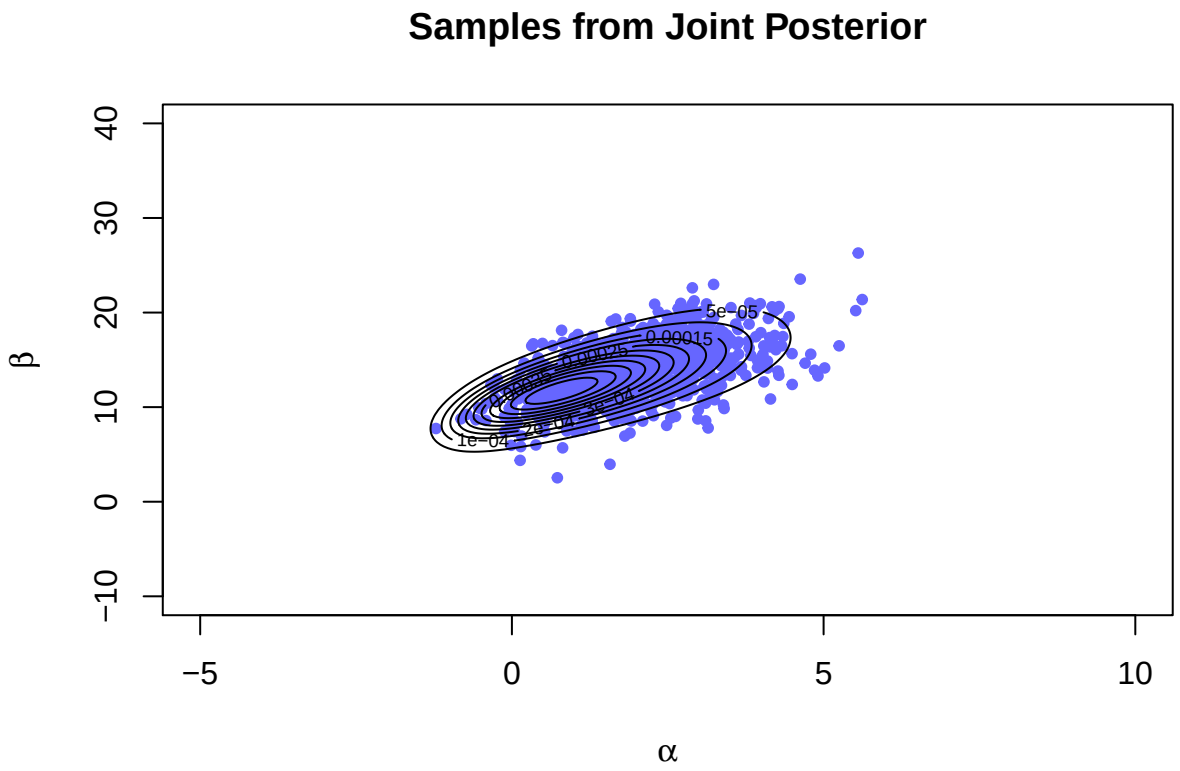
```
unnorm_like = function(alpha, beta) {
    prod((logitinv(alpha + beta * x))^y * (1 - logitinv(alpha + beta * x))^(n -
        y))
}


like_grid = matrix(0, gsize, gsize)

# grid of normalized likelihood
for (i in 1:gsize) {
    for (j in 1:gsize) {
        like_grid[i, j] = unnorm_like(agrid[j], bgrid[i])
    }
}
```

```r
# normalize the grid
norm_like_grid = like_grid/sum(like_grid)

# marginal of alpha
alpha_like_grid = apply(norm_like_grid, 2, sum)


# number of samples
B = 1000
# sample marginal of alpha
place = sample.int(gsize, B, prob = alpha_like_grid, replace = TRUE)
alpha_like_sample = agrid[place]

# sample from conditional of beta
beta_like_sample = place2 = numeric()
for (b in 1:B) {
    place2[b] = sample.int(gsize, 1, prob = norm_like_grid[place[b], ], replace = TRUE)
    beta_like_sample[b] = bgrid[place2[b]]
}

# add random jitter to make continuous
alpha_like_sample = alpha_like_sample + runif(B, -astep/2, astep/2)
beta_like_sample = beta_like_sample + runif(B, -bstep/2, bstep/2)

plot(alpha_like_sample, beta_like_sample, col = rgb(0.4, 0.4, 1, 1), pch = 20,
    ylim = c(-10, 40), xlim = c(-5, 10), main = "Samples from Normalized Likelihood",
    xlab = expression(alpha), ylab = expression(beta))
contour(agrid, bgrid, norm_like_grid, add = TRUE)
```



Joint Posterior

```
plot(alpha_post_sample, beta_post_sample, col = rgb(0.4, 0.4, 1, 1), pch = 20,
    ylim = c(-10, 40), xlim = c(-5, 10), main = "Samples from Joint Posterior",
    xlab = expression(alpha), ylab = expression(beta))
contour(agrid, bgrid, norm_post_grid, add = TRUE)
```



We can tell through the plots of samples of the prior, normalized likelihood, and posterior that the posterior is a compromise of the former two (more heavily weighted towards the data).

How has our chosen prior affected our analysis?

This prior has centroid at $(\alpha, \beta) = (4, 10)$. It is also bivariate normal so its density decreases exponentially as you move away from the centroid. Altogether, this means that by using this prior, we are fairly confident that the effect of dosage level on probability of dying is not zero, and in fact assumes a linear form that is very close to the values that we have chosen for the centorid of our prior. This makes a nice contrast with the previous uniform prior used in the book where we assume that we know nothing - all possible values of alpha and beta are equally likely. As a result, the posterior for the uniform prior is a little more spread out than ours but not by much - the likelihood has a fairly overwhelming effect on the prior. This suggests that so long as we have a lot of data, choice of prior is relatively robust to misspecification (so long as we aren't picking something close to a degenerate prior).

Improper Prior and the Metropolis Algorithm

Say our prior is improper:

$$p(\alpha, \beta) \propto 1$$

This yields the unnormalized joint posterior:

$$p(\alpha,\beta|y) \propto p(\alpha,\beta)p(y|\alpha,\beta,x) = \prod_{i=1}^{k} p(y_i|\alpha,\beta,x_i)$$

$$\propto \prod_{i=1}^{k} [logit^{-1}(\alpha+\beta x_i)]^{y_i}[1-logit^{-1}(\alpha+\beta x_i)]^{n_i-y_i}$$

Now that we have the unnormalized posterior density, we can use the Metropolis algorithm to sample from it! It doesn't need to be normalized because we are calculating a ratio of the posterior density evaluated at different points - so the constants cancel out.

Our process is:

1. Draw a starting points from an overdispersed starting distribution. I believe a uniform on the α by β grid $(-4, 10) \times (-10, 40)$ is a good starting point.
2. For 2000 iterations, we sample a proposal θ^* from a jumping distribution which must be symmetric. We will use the bivariate normal centered on the previous iteration: $J_t(\theta^*|\theta^{t-1}) = N_2(\theta^*|\theta^{t-1}, I_2)$.
3. Then we calculate the ratio of densities:

$$r = \frac{p(\theta^*|\mathbf{y})}{p(\theta^{t-1}|\mathbf{y})}$$

4. And finally we update the parameter with θ^* with probability $\min(r, 1)$. Otherwise we don't update.

```
#data
data = matrix(c(-0.86,-0.3,-0.05,0.73,5,5,5,5,0,1,3,5),nrow=4,ncol=3,byrow=FALSE)
colnames(data) = c("x","n","y")
x = data[,"x"]
n = data[,"n"]
y = data[,"y"]

#metropolis algorithm
metropolis = function(start,posterior,jump,niter){
  theta = start
  draws = matrix(NA,niter,2) #initialize draws with starting point
  accepted = rep(0,niter)

  for(j in 1:niter){
    proposal = jump(theta)
    r = posterior(proposal)/posterior(theta) #simplified since normal jumping distribution is symmetric
    if(runif(1,0,1)<r){
      theta = proposal
      accepted[j] = 1
    }
    draws[j,] = theta
  }
  print(mean(accepted))
  return(draws)
}


#run
start = numeric(2)
start[1] = runif(1,-4,10) #alpha
```

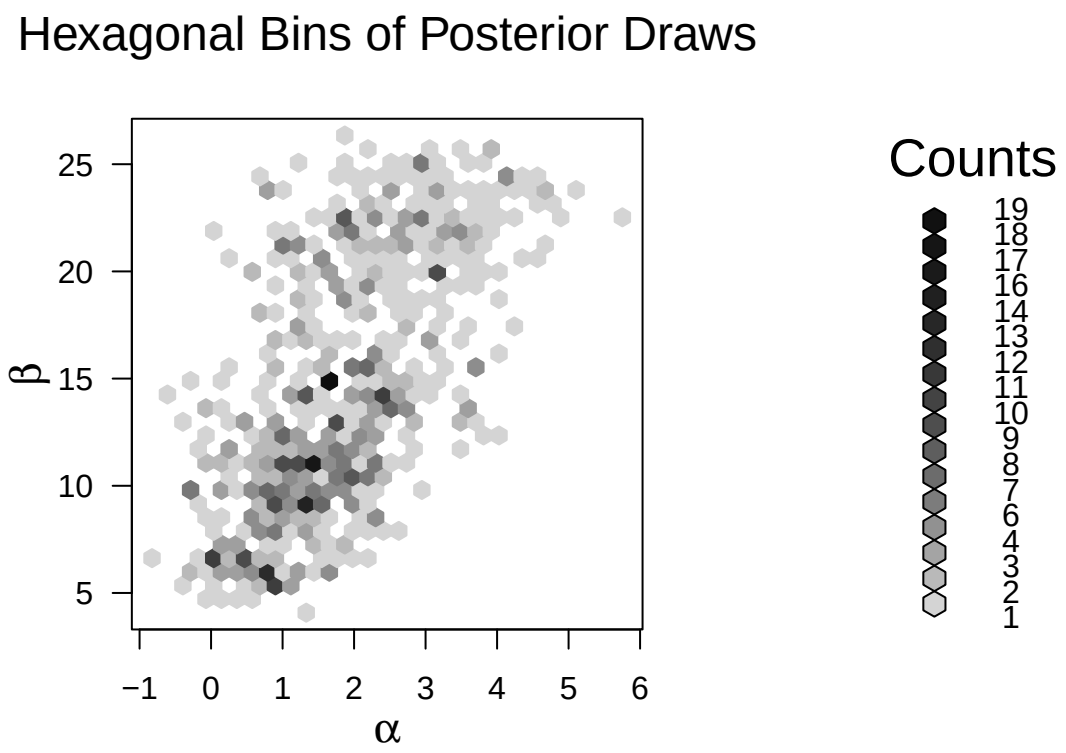
```
start[2] = runif(1,-10,40) #beta

posterior = function(theta){
  alpha = theta[1]
  beta = theta[2]
  loginv = function(x){exp(x)/(1+exp(x))}
  prod((loginv(alpha+beta*x))^y*(1-loginv(alpha+beta*x))^(n-y))
}

#bivariate normal as jumping distribution
jump = function(theta){
  return(rmvnorm(1,mean=theta,sigma=diag(length(theta))))
}

ndraws = 2000

post_draws = metropolis(start,posterior,jump,ndraws)
```

```
## [1] 0.6645
```

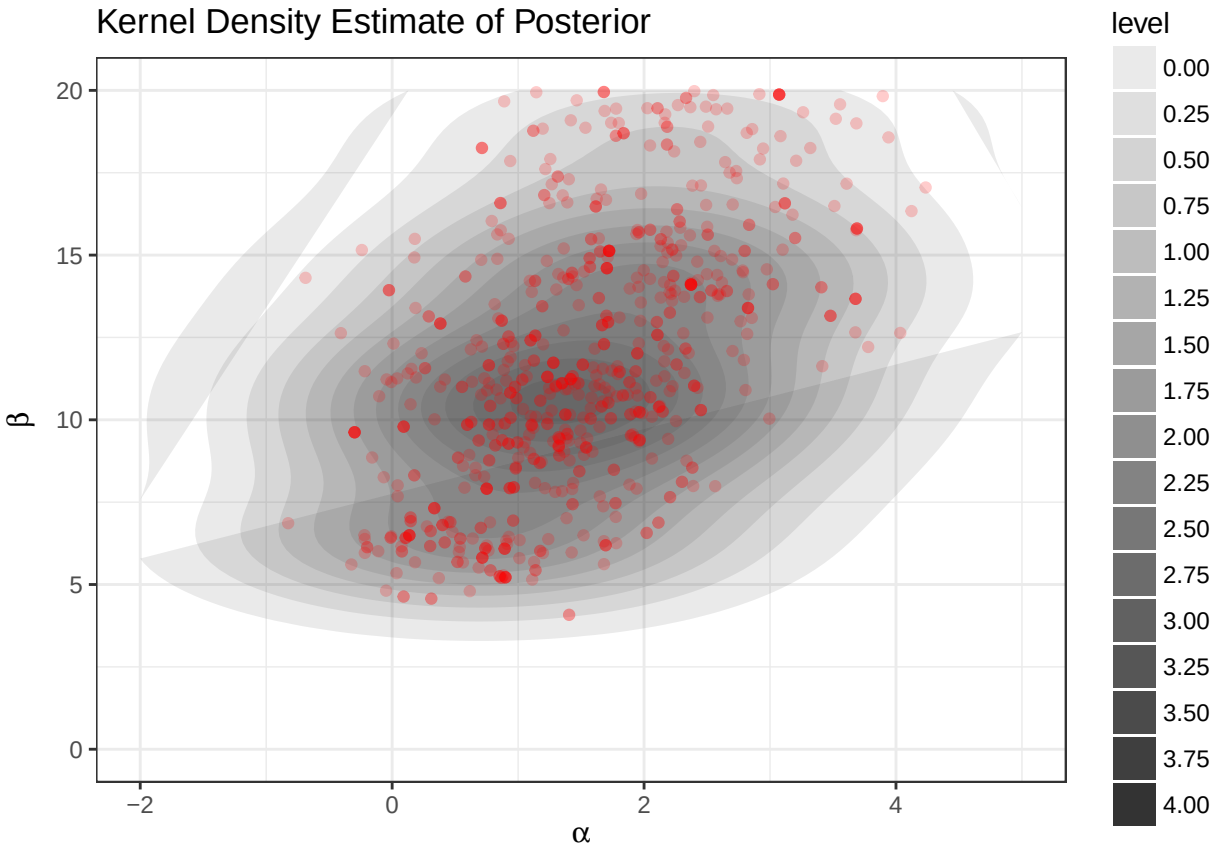
```
#conservative burn-in - 50% as suggested by textbook
burnin = 1:(nrow(post_draws)%/%2)
post_draws = post_draws[-burnin,]
```

Density Plot of Joint Posterior

```
# bivariate histogram - hexagonal binning
plot(hexbin(post_draws[, 1], post_draws[, 2]), xlab = expression(alpha), ylab = expression(beta),
    main = "Hexagonal Bins of Posterior Draws")
```

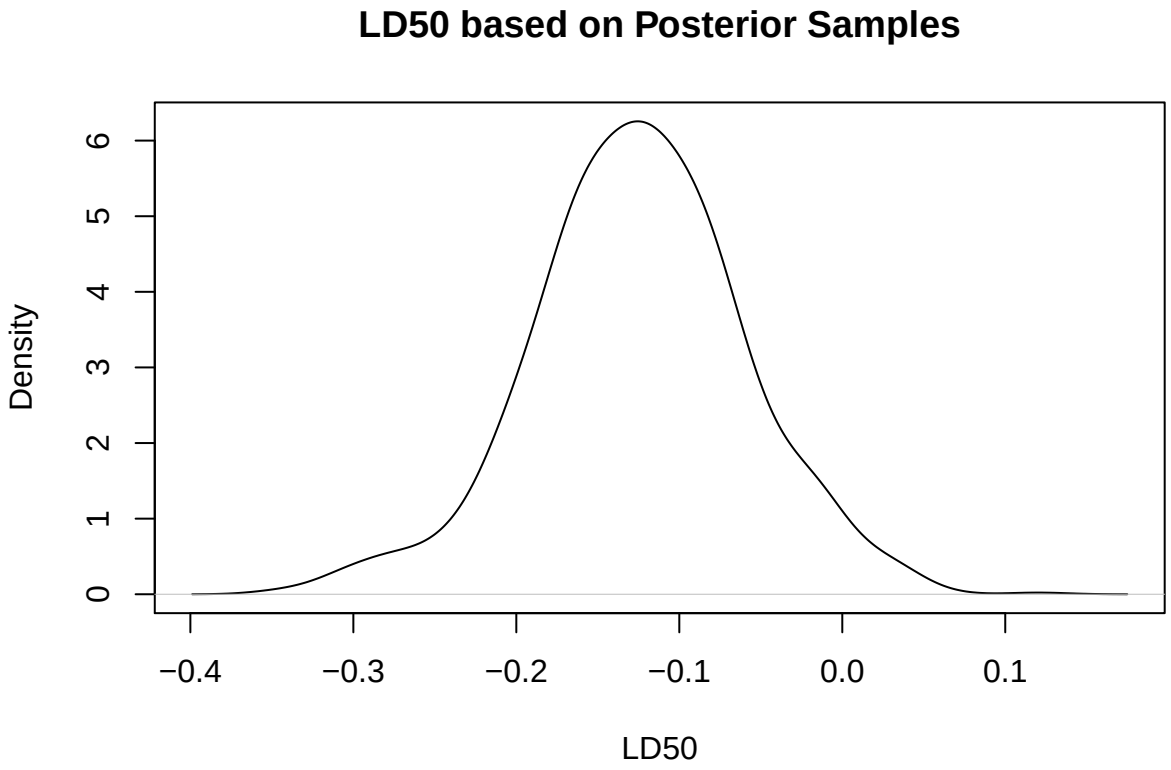


```
ggplot(data = as.data.frame(post_draws), aes(V1, V2)) + stat_density2d(aes(alpha = ..level..),
    geom = "polygon", h = 5) + scale_alpha_continuous(limits = c(0, 4), breaks = seq(0,
    4, by = 0.25)) + geom_point(colour = "red", alpha = 0.2) + theme_bw() +
    xlab(expression(alpha)) + ylab(expression(beta)) + ggtitle("Kernel Density Estimate of Posterior") +
    xlim(-2, 5) + ylim(0, 20)
```



Posterior Distribution of LD50

```
plot(density(-post_draws[, 1]/post_draws[, 2], adjust = 1.3), type = "l", main = "LD50 based on Posteri
    xlab = "LD50")
```


LD50 based on Posterior Samples
Density
0
1
2
3
4
5
6
−0.4
−0.3
−0.2
−0.1
0.0
0.1
LD50