Chapter 7

Beyond the Normal. Other Bivariate Posteriors (I)

7.1 Introduction

Beyond the normal distribution, few multiparameter sampling models allow simple explicit calculation of posterior distributions. To illustrate that practical analysis is possible for some problems using simple simulation techniques, we present here an example of nonconjugate models. This is an example of a nonconjugate model for a bioassay experiment, drawn from literature on applied bayesian statistics. This case was analyzed in Gelman et al's book (1995, p. 82), and we only add the programming to it, along with some interpretations. The model is a two-parameter example from the broad class of generalized linear models to be considered more thoroughly later in the course.

Required reading for this topic, in addition to this lesson, is chapter 4 of Hoff's book and the discussions on grid in chapter 5.

7.2 Bioassay experiment

7.2.1 The scientific problem and the data

In the development of drugs and other chemical compounds, acute toxicity tests or bioassay experiments are commonly performed on animals. Such experiments proceed by administering various dose levels of the compound to batches of animals. The animals' responses are typically characterized by a dichotomous outcome: for example, death or not, tumor or not. An experiment of this kind gives rise to data of the form

$$(x_i, n_i, y_i) \qquad i = 1, ..., k$$

where x_i represents the *ith* of k dose levels (measured on a logarithmic scale) given to n_i animals, of which y_i subsequently respond with positive outcome. An example is the data given in Racine, A., Grieve, A.P., Fluhler, H. and Smith, A.F. M. (1986) Bayesian Methods in Practice: experiences in the pharmaceutical industry. *Applied Statistics* 35, 93-150.

$Dose, x_i$ ($\log(g/ml)$)	$Number\ \ of$ *animals*	$Number\ \ of$ $deaths, y_i$
−0.863	5	0
−0.296	5	1
−0.053	5	3
0.727	5	5

7.2.2 Modeling the dose-response relation

We model the outcomes of the five animals within each group as exchangeable, independent and with equal probabilities, which implies that the data points y_i are binomially distributed:

$$y_i \mid \theta \sim Bin(n_i; \theta_i)$$

where θ_i is the probability of death for animals given dose x_i. For this study, it is assumed that the outcomes in the four groups are independent of each other, given the parameters $\theta_1, \theta_2, \theta_3, \theta_4$. An example of a situation in which independence and the binomial model would be inappropriate is if the deaths were caused by a contagious disease.

The simplest analysis would treat the four parameters θ_1 as exchangeable in their prior distribution, perhaps using a noninformative density such as

$$p(\theta_1, \theta_2, \theta_3, \theta_4) \propto 1$$

in which case the parameters θ_i would have independent beta posterior distributions. The exchangeable prior model for the θ_i parameters has a serious flaw, however; we know the dose level x_i for each group i, and one would expect the probability to death to vary systematically as a function of dose.

The simplest model of the dose-response-relation of θ_i to x_i is

$$\theta_i = \alpha + \beta x_i.$$

But this model has the flaw that for very low or high doses, x_i approaches infinity (log scale of x), whereas θ_i, being a probability, must lie between 0 and 1. We solve this by taking a logistic transformation of the θ_i and specify the model as

$$logit(\theta_i) = log(\theta_i/(1-\theta_i)$$

$$logit(\theta_i) = \alpha + \beta x_i$$

This is called a *logistic regression model*. The logistic transformation is

$$logit^{-1}(\theta) = exp(\theta_i)/1 + exp(\theta),$$

which has the inverse transform

$$logit^{-1}(\theta) = exp(\theta_i)/1 + exp(\theta).$$

7.2.3 The likelihood

The likelihood for each of the experiments i, in terms of the parameters α and β is

$$p(y_i \mid \alpha, \beta, n_i, x_i) \propto \left[logit^{-1}(\alpha + \beta x_i)\right]^{y_i} \left[1 - logit^{-1}(\alpha + \beta x_i)\right]^{n_i - y_i}$$

The overall model for all the experiments is characterized by the parameters α and β, whose joint posterior distribution is

$$p(\alpha, \beta \mid y, n, x) \propto p(\alpha, \beta \mid n, x) p(y \mid \alpha, \beta, n, x) \quad (7.1)$$

$$\propto p(\alpha, \beta) \prod_{i=1}^{k} p(y_i \mid \alpha, \beta, n_i, x_i) \quad (7.2)$$

We consider the sample sizes n_i and dose levels x_i as fixed for this analysis and suppress the conditioning on (n, x) in subsequent notation.

The following function in R defines the likelihood. You make it active to be ready for use later by just copy pasting it into R to compile.

```
 ########################
 # Functions needed to construct the likelihood
 ############################

logistic <- function(th)
{ exp(th)/(1+exp(th))
}

likl.82 <- function(a,b)
{
x <- c(-0.863,-0.296,-0.053,0.727)
n <- c(5,5,5,5)
y <- c(0,1,3,5)

f <- 1
for(i in 1:4){
  lxi <- logistic(a+b*x[i])
    f <- f*lxi^y[i] * (1-lxi)^(n[i]-y[i])
  }
    f
  }
```

7.2.4 The prior distribution

For prior distribution for (α,β) we choose one that is independent and locally uniform in the two parameters; that is, $p(\alpha,\beta) \propto 1$. In practice, we might use a uniform prior distribution if we really have no prior knowledge about the parameters, or if we want to present a simple analysis of this experiment alone. If the analysis using a noninformative prior distribution is insufficiently precise, we may consider using other sources of substantive information (for example, from other bioassay experiments) to construct an informative prior distribution.

7.2.5 The posterior distribution

Because of the prior we chose, the posterior distribution is the same as the likelihood. So we make R know that, by creating another function that we will copy paste into R to compile for future use.

```
###################################
#  Posterior distribution
###################################

 post.82 <- function(a,b)
 {
 likl.82(a,b)
 }
```

7.2.6 Obtaining a contour plot of the joint posterior density

A rough estimate of the parameters

We will compute the joint posterior distribution at a grid of points(α,β). But first we get a rough estimate of where to look doing a regression of the logit of the observed proportions $(y_i/n + i)$ with respect to the x_i. The authors did that

and found approximately $(\hat{\alpha}, \hat{\beta}) = (0.1, 2.9)$ with standard errors of 0.3 and 0.5 for α and β respectively. Notice that because logit of 0 and logit of 1 both give infinity, we changed a little the values of y_1 and y_4. This does not hurt, since all we want is to be able to obtain some idea of where to draw our grid of points for *alpha* and β.

```
 ###################
 # Getting an idea of where to look at in the grid
 #################

 x=c(-0.863, -0.296,-0.053,0.727)  #dose
 y=c(0.5,1,3,4.5)   # number of dead, slightly changed
 yn5=y/5   # proportion of dead
 logity=log(yn5/(1-yn5))   # logit transform
 lm(logity~x)     # run regression to get initial alpha and beta

Call:                            # Result of the regression
lm(formula = logity ~ x)

Coefficients:
(Intercept)            x
     0.1045       2.8841  #slope is 2.88 and intercept is 0.1045
```

Getting the grid ready

We are now ready to compute the posterior density at a grid of points (α, β). In our first try at a contour plot, we computed the unnormalized density given above, based on a uniform prior distribution, on a 200×200 grid on the range $[-1, 1] \times [1, 5]$ that is, the estimated mean plus or minus more than two standard errors in each direction- and then used R to calculate contour lines of equal posterior density. The contour lines on the first plot ran off the page, indicating that the initial normal regression analysis was very crude indeed. We recomputed the posterior density in a much wider range: $(\alpha, \beta) \in [-5, 10] \times [-10, 40]$.

```
  ##############################
  # Obtaining the unnormalized posterior on the grid
  ##############################

   agrid<- seq(-5,10,length=200)
 bgrid <- seq(-10,40,length=200)
 p <- matrix(0,nrow=200,ncol=200) # initialize matrix for posterior
                                                          # eval's over agrid x bgrid

 #evaluate posterior on grid
 for(i in 1:200) {
    for(j in 1:200) {
     p[i,j] <- post.82(agrid[i],bgrid[j])
  }
     }

 contour(agrid,bgrid,p,xlab="ALPHA", ylab="BETA")  #contour

image(agrid, bgrid,p,xlab="ALPHA",ylab="BETA")   #image
```

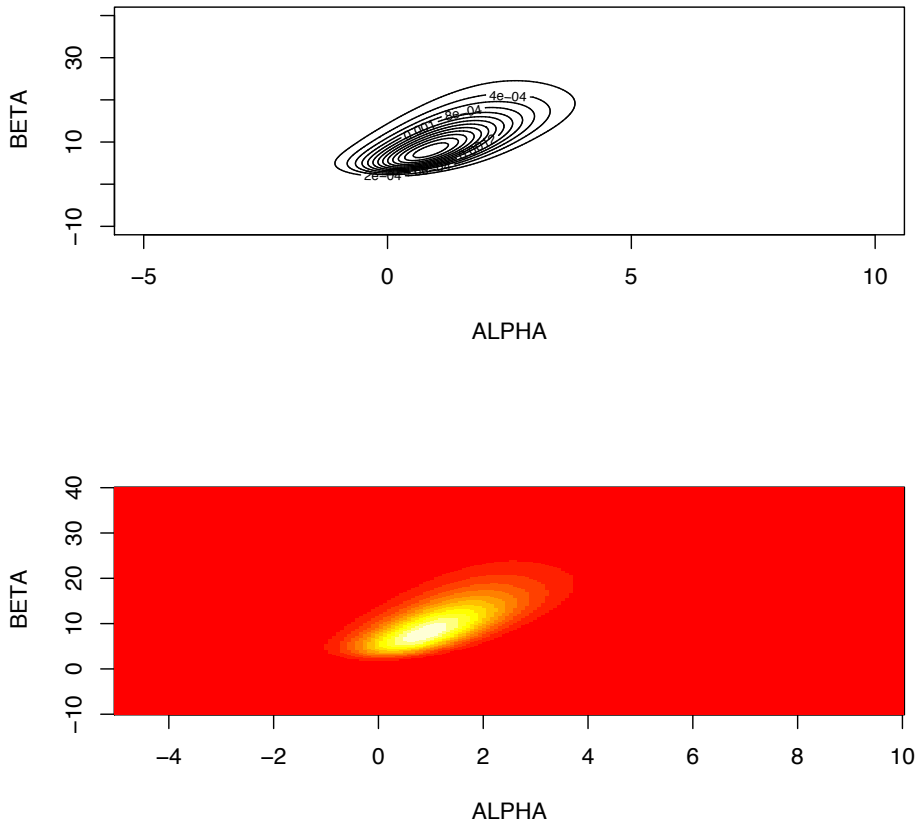


fig:gelmanp82}

Figure 7.1: Contour plot and contour image of the posterior distribution

We interpret this contour plot as the inner lines having higher density, (as in the map of a mountain, where inner lines mean higher altitude). In the color image, colors become darker as density becomes smaller and smaller. It gives us regions of similar density.

7.2.7 Sampling from the joint posterior distribution

Having computed the unnormalized posterior density at a grid of values that cover the effective range of (α, β) we can normalize by approximating the distribution as a step function over the grid and setting the total probability of the grid to 1. We sample 1000 random draws (α^l, β^l), from the posterior distribution using the following procedure:

(a) Compute the marginal posterior distribution of α by numerically summing over β in the step-function distribution computed on the grid of the figure just created (the contour plot).

(b) For $l = 1, ..., 1000$:

Draw α^l from $p(\alpha \mid y)$ using the inverse cdf method.

Draw β^l from the conditional distribution $p(\beta \mid \alpha, y)$, given the just sampled value of α (actually, using the nearest value on the computed grid), again using the inverse cdf method.

The 1000 draws (α^l, β^l) can then be displayed on a scatterplot. The scale of this plot must be set so that all the 1000 draws will fit in the graph.

```
 # First create the functions that we will need to generate the distributions

#################################################################
#     Function init.rpost.82
 # requires: agrid, bgrid, p, p.agrid
 # sets up  marginal cdf for alpha:                 cdfa
 #                 cond cdf for beta   | alpha:            cdfba
 # note: computing cdfba-inf for p(beta | alpha) would be too cumbersome,
 # would need it on a whole grid of alphas !
 #################################################################
 init.rpost.82 <- function()
 {
 da <- agrid[2]-agrid[1]              #step size on a grid
 k <- sum(p.agrid*da)      # probability in that interval
 cat("computing cdfa.inv....\n")
 cdfa <<- rep(0,200)   # initialize marginal cdf for alpha
 cdfa[1] <<- p.agrid[1]*da/k

 for ( i in 2:200)          #compute marginal cdf for alpha
   cdfa[i] <<- cdfa[i-1] + da*p.agrid[i]/k

      cdfba <<- matrix(0, nrow=200,ncol=200)   # initialize cdf for beta given alpha
         db <- bgrid[2] - bgrid[1]   #step size on bgrid
         k <- apply(p, 1, sum)*db     #vector of standardization constants
          cdfba[,1] <<- db*p[,1]/k
         for ( j in 2:200)
             cdfba[,j] <<- cdfba[, j-1]+db*p[,j]/k



           NULL
     }

  ################################################################
  #  Function rpost.82
  #
  #simulating a sample from the posterior using inverse cdf method
  # note:need to call init.rpost to setup cdfa & cdfba
  # see the Appendix to this lesson for an explanation of the inverse cdf method
  ################################################################
   rpost.82 <- function(n)
   {
    # Now generate p(a) and p(b |a)
   u1<-runif(n)
   u2<- runif(n)    #uniform r.v.
   a<- rep(0,n)   #initialization
   b<-rep(0,n)     #initialization
```

```
for(j in 1:n){
    k <- (1:200)[cdfa>=u1[j]][1]  # k= first index st cdfa[k]>u1
    a[j] <-agrid[k]                          #a[j] =cdf-inv(u1[j])
    l<-(1:200)[cdfba[k,] >= u2[j]][1]     # l= first index st cdf[k]>u1
    b[j]<- bgrid[l]                          #b[j]=cdfab-inv(u2[j])
    }
    return(cbind(a,b))  # return a (n,2) matrix of a,b vectors.

  }

    ##########################
    #  Main commands
    ##########################

    #marginal in alpha
      p.agrid <<- apply(p,1,sum)

     #marginal in beta
      p.bgrid <<- apply(p,2,sum)

    #Generate a posterior sample
   init.rpost.82()
   th <- rpost.82(1000)    # simulate 200 draws from posterior
   plot(th[,1],th[,2],xlab="ALPHA", ylab="BETA",xlim=c(-5,10),ylim=c(-10,40), pch=".")
```

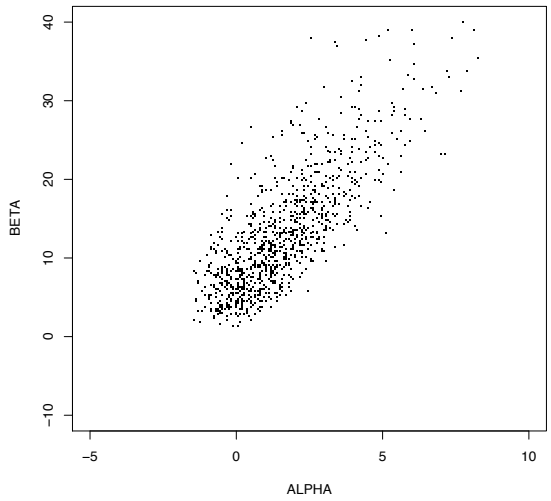


Figure 7.2: Scatterplot of 1000 points from the posterior distribution {fig:rpost82}

7.2.8 The posterior distribution of the LD50

The LD50 is a parameter of common interest in bioassay studies. It is defined as the dose level at which the probability of death is 50%. In the logistic model we are using, a 50% survival rate means

$$LD50 : E\left(\frac{y_i}{n_i}\right) = logit^{-1}(\alpha + \beta x_i) = 0.5;$$

thus $\alpha + \beta x_i = logit(0.5) = 0$, and the $LD50$ is $x_i = -\frac{\alpha}{\beta}$. Computing the posterior distribution of any summaries in the Bayesian approach is straightforward. Given what we have done so far, simulating the posterior distribution of the $LD50$ is trivial: we just compute $-\frac{\alpha}{\beta}$ for the 1000 draws of (α, β).

```
############################
# Simulating LD95
############################

      alpha = th[,1]
      beta = th[,2]

        lambda <- (log(0.95/0.05)-alpha)/beta
       hist(lambda, col=0,xlab="LD95",ylab="P(LD95 | x)", prob=T )
```

7.3 Appendix A: Sampling using the inverse cumulative distribution function

The cumulative distribution function, or cdf, F, of a one-dimensional distribution, $p(v)$, is defined by

$$F(v) = Prob(c \leq v_* = \int_{-\infty}^{v_*} p(v)dv$$

with summation replacing integral when the random variable is discrete.

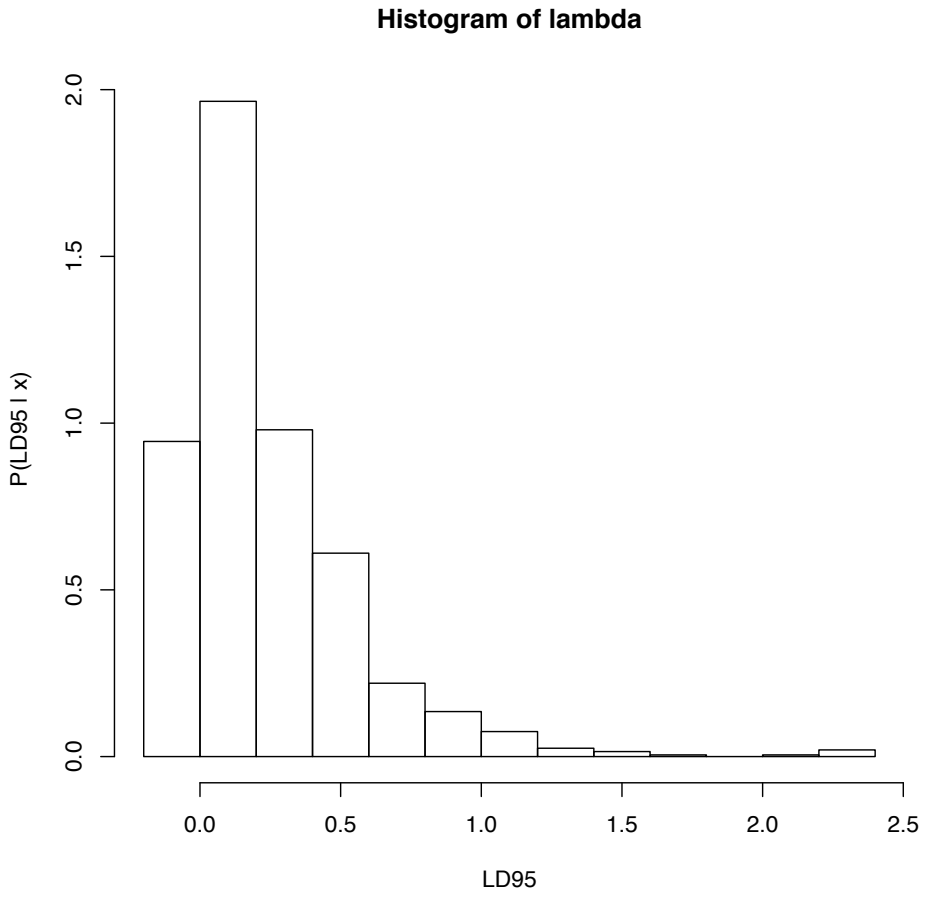
The inverse cdf can be used to obtain random samples from the distribution p, as follows. First draw a random value, U, from the uniform distribution on $[0, 1]$, runif(1). Then let $v = F^{-1}(U)$. The function F is not necessarily one-to-one–certainly not if the distribution is discrete-but $v = F^{-1}(U)$ is unique with probability 1. The value v will be a random draw from p, and is easy to compute as long as $F^{-1}(U)$ is simple.

For a continuous example, suppose v has an exponential distribution with parameter λ; then its CDF is $F(v) = 1 - \exp(-\lambda v)$ and the value of v for which $U = F(v)$ is $v = -log(1 - U)/\lambda$. Of course $(1 - U)$ also has the uniform distribution on $[0, 1]$, so we can obtain random draws from the exponential distribution as $-(\log U)/\lambda$.

7.4 Appendix B: The program all together

```
 ##########################
# Example p. 84 GCSR 1995
##########################

# likelihood in Example p. 82
```



{fig:ld95}

Figure 7.3: Histogram of the draws from the posterior distribution of the LD50

```
logistic <- function(th)
{ exp(th)/(1+exp(th))
}

likl.82 <- function(a,b)
{
x <- c(-0.863,-0.296,-0.053,0.727)
n <- c(5,5,5,5)
y <- c(0,1,3,5)

f <- 1
for(i in 1:4){
  lxi <- logistic(a+b*x[i])
    f <- f*lxi^y[i] * (1-lxi)^(n[i]-y[i])
  }
    f
  }
```

```
 #Posterior in example p82

 post.82 <- function(a,b)
 {
 likl.82(a,b)
 }

 agrid<- seq(-5,10,length=200)
 bgrid <- seq(-10,40,length=200)
 p <- matrix(0,nrow=200,ncol=200) # initialize matrix for posterior
                                                                  # eval's over agrid x bgrid

 #evaluate posterior on grid
 for(i in 1:200) {
    for(j in 1:200) {
     p[i,j] <- post.82(agrid[i],bgrid[j])
  }
     }

#alternatively --note that post.82 can take a whole vector t a time..
#                          use that only if you clearly understand what's happening
for(i in 1:200) {
    p[i,] <- post.82(agrid[i],bgrid)
  }

contour(agrid,bgrid,p,xlab="ALPHA", ylab="BETA")
image(agrid, bgrid,p,xlab="ALPHA",ylab="BETA")


# First create the functions that we will need to generate the distributions

#################################################################
#     Function init.rpost.82
 # requires: agrid, bgrid, p, p.agrid
 # sets up  marginal cdf for alpha:                cdfa
 #                 cond cdf for beta   | alpha:            cdfba
 # note: computing cdfba-inf for p(beta | alpha) would be too cumbersome,
 # would need it on a whole grid of alphas !
 #################################################################
 init.rpost.82 <- function()
 {
 da <- agrid[2]-agrid[1]              #step size on a grid
 k <- sum(p.agrid*da)        # probability in that interval
 cat("computing cdfa.inv....\n")
 cdfa <<- rep(0,200)   # initialize marginal cdf for alpha
 cdfa[1] <<- p.agrid[1]*da/k

 for ( i in 2:200)          #compute marginal cdf for alpha
   cdfa[i] <<- cdfa[i-1] + da*p.agrid[i]/k
```

```
     cdfba <<- matrix(0, nrow=200,ncol=200)   # initialize cdf for beta given alpha
        db <- bgrid[2] - bgrid[1]   #step size on bgrid
        k <- apply(p, 1, sum)*db     #vector of standardization constants
         cdfba[,1] <<- db*p[,1]/k
        for ( j in 2:200)
            cdfba[,j] <<- cdfba[, j-1]+db*p[,j]/k



          NULL
    }

 ################################################################
 #  Function rpost.82
 #
  #simulating a sample from the posterior using inverse cdf method
  # note:need to call init.rpost to setup cdfa & cdfba
  # see GCSR p 22/23 & p 84 for an explanation of the inverse cdf method
 ################################################################
  rpost.82 <- function(n)
  {
   # Now generate p(a) and p(b |a)
  u1<-runif(n)
  u2<- runif(n)    #uniform r.v.
  a<- rep(0,n)   #initialization
  b<-rep(0,n)     #initialization
  for(j in 1:n){
      k <- (1:200)[cdfa>=u1[j]][1]  # k= first index st cdfa[k]>u1
      a[j] <-agrid[k]                          #a[j] =cdf-inv(u1[j])
      l<-(1:200)[cdfba[k,] >= u2[j]][1]     # l= first index st cdf[k]>u1
      b[j]<- bgrid[l]                          #b[j]=cdfab-inv(u2[j])
      }
      return(cbind(a,b))  # return a (n,2) matrix of a,b vectors.

    }

      ##########################
      #  Main commands
      ##########################

      #marginal in alpha
        p.agrid <<- apply(p,1,sum)  # note global assignment by by <<-

      #Generate a posterior sample
     init.rpost.82()
     th <- rpost.82(200)    # simulate 200 draws from posterior
     plot(th[,1],th[,2],xlab="ALPHA", ylab="BETA",xlim=c(-5,10),ylim=c(-10,40), pch=".")

 ############################
 # Simulating LD95
 ############################
```

```
alpha = th[,1]
beta = th[,2]

  lambda <- (log(0.95/0.05)-alpha)/beta
 hist(lambda, col=0,xlab="LD95",ylab="P(LD95 | x)", prob=T )
```