Bayes: Homework 5
Leslie Gains-Germain

1. The data are counts of mortalities due to stomach cancer ($\mathbf{y}$) out of a total number at risk males ($\mathbf{n}$) in the largest cities in Missouri. The counts are assumed to follow a $Beta\!-\!Binomial(\alpha, \beta)$ model. This model is used because it allows the probability of death to vary across cities.

   First, we write a function that calculates the log likelihood function for parameters η and K given the data. η represents the mean of the Beta-Binomial model, and K represents the precision. α and β are the specified parameters of the Beta-Binomial model, and they are functions of η and K:

$$\beta = K(1-\eta)$$
$$\alpha = K * \eta$$

   The likelihood function is as follows, shown here both as a function of α and β, and as a function of η and K.

$$L(\alpha, \beta|\mathbf{y}, \mathbf{n}) = \prod_{j=1}^{j=20} \binom{n_j}{y_j} B(\alpha + y_j, n_j + \beta - y_j) \frac{1}{B(\alpha, \beta)}$$
$$L(\eta, K|\mathbf{y}, \mathbf{n}) = \prod_{j=1}^{j=20} \binom{n_j}{y_j} B(K*\eta + y_j, n_j + K(1-\eta) - y_j) \frac{1}{B(K*\eta, K(1-\eta))}$$

   The log likelihood function is as follows:

$$logL(\alpha, \beta|\mathbf{y}, \mathbf{n}) = \sum_{j=1}^{j=20} log(\binom{n_j}{y_j}) + log(B(\alpha + y_j, n_j + \beta - y_j) - log(B(\alpha, \beta))$$
$$logL(\eta, K|\mathbf{y}, \mathbf{n}) = \sum_{j=1}^{j=20} log(\binom{n_j}{y_j}) + log(B(K*\eta + y_j, n_j + K(1-\eta) - y_j)) - log(B(K*\eta, K(1-\eta)))$$

   The following function evaluates the log likelihood at specified values of η and K given the data.

```
#We can assume the counts come from a Beta-Binomial model with mean eta
# and precision K  (See Appendix page 584 in Gelman et al
##    for parameterization)
##     n=nj,  beta=K(1-eta), alpha=K*eta

  BB.loglik.fun <- function(etaK.vec, n.vec, y.vec) {
     ll.out <- sum(lchoose(n.vec, y.vec) +
        lbeta((etaK.vec[2]*etaK.vec[1] + y.vec),(etaK.vec[2]*(1-etaK.vec[1])+n.vec-y.vec))-
        lbeta(etaK.vec[2]*etaK.vec[1], etaK.vec[2]*(1-etaK.vec[1])))
     return(ll.out)
        }
```

Next, the data are shown. Recall that these are death counts along with the total at risk males for the largest cities in Missouri.

```
  #### Applied problem  (Tsutakawa et al. 1985) - Estimate rate of death from
  ## stomach cancer for at risk males between ages 45-64 for the largest cities in
  ## Missouri.  Here are the mortality rates for 20 of the cities
  ##  (number at risk (nj), number of cancer deaths (yj))

  nj.vec <- c(1083,855,3461,657,1208,1025, 527, 1668, 583, 582, 917, 857, 680, 917, 53637,
              874, 395, 581, 588, 383)
  yj.vec <- c(0,0,2,0,1,1,0,2,1,3,0,1,1,1,54,0,0,1,3,0)
```

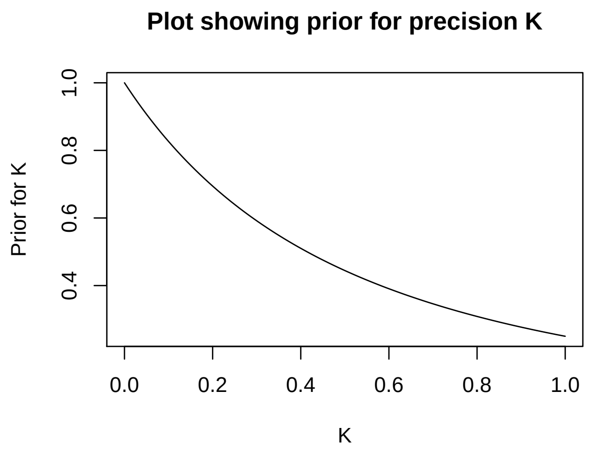
Next, we write a function to find prior probabilities for specified values of η and K. She assumes that η and K are independent, and that the joint prior is simply a product of the marginal priors. The prior distributions for η and K that she assigns are as follows:

$$\begin{aligned} p(\eta) &= \frac{1}{\eta(1-\eta)} \\ p(K) &= \frac{1}{(1+K)^2} \end{aligned}$$

```
  ## We need to assign priors for eta and K - let's use the vague prior
  ##  porportional to  g(eta,K) propto (1/(eta*(1-eta)))*(1/(1+K)^2)

   BB.prior.fun <- function(etaK.vec) {
      (1/(etaK.vec[1]*(1-etaK.vec[1])))*(1/((1+etaK.vec[2])^2))
     }
```

The prior distribution for η is the improper $Beta(0, 0)$ prior which is a non-informative prior with the constraint that η is a value between 0 and 1. The prior distribution for K is shown in the plot below. Since K is the precision for the Beta-Binomial model, the prior gives higher weight to lower values of precision (and thus higher values of the variance).



Next she writes a function to find unnormalized posterior probabilities on the log scale for specified values of η and K given the data. The `BB.logpost1` function simply adds the prior and the likelihood on the log scale (equivalent to multiplying them on the original scale).

$$p(\eta, K|\mathbf{y}, \mathbf{n}) = L(\eta, K|\mathbf{y}, \mathbf{n})p(\eta, K)$$

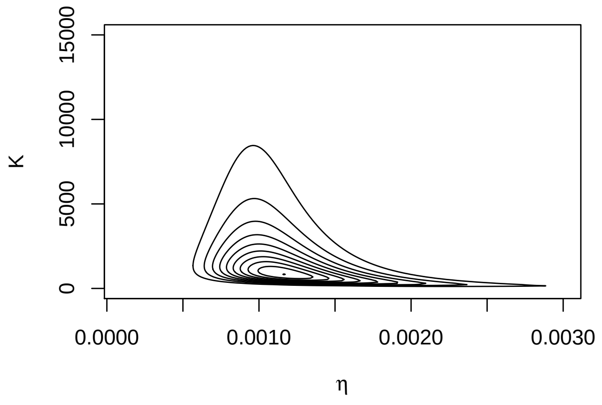
$$logp(\eta, K|\mathbf{y}, \mathbf{n}) = logL(\eta, K|\mathbf{y}, \mathbf{n}) + logp(\eta, K)$$

```
## We could combine them into one function to find the posterior

BB.logpost1<- function(etaK.vec, n.vec, y.vec) {
      ll <- sum(lbeta(etaK.vec[2]*etaK.vec[1] + y.vec, etaK.vec[2]*(1-etaK.vec[1])+n.vec-y.vec)-
              lbeta(etaK.vec[2]*etaK.vec[1], etaK.vec[2]*(1-etaK.vec[1])))
      lprior <- -log(etaK.vec[1]) - log(1-etaK.vec[1]) - 2*log(1+etaK.vec[2])
      lpost.out <- ll + lprior
      return(lpost.out)
     }
```

Below, she makes a grid of possible values for η and K, and she stores them in a matrix called `etaK.grid`. Then, for every combination of η and K, she evaluates the `BB.logpost1` function to find the unnormalized posterior probability on the log scale for that combination. Then, she transforms back to the original scale by exponentiating. Subtracting `lp.mat-max(lp.mat)` before exponentiating makes computation easier because it increases the values of the posterior probabilities (if this step is not included, the posterior probabilities are on the order of $3 * 10^{-261}$). She then displays the unnormalized posterior probabilities in a contour plot for each combination of η and K.

```
    eta.vec <- seq(0.0001,0.003,length=200)
     K.vec <- seq(1,15000, length=200)
        etaK.grid <- expand.grid(eta.vec, K.vec)
        lp.grid <- apply(etaK.grid, 1, BB.logpost1, n.vec=nj.vec, y.vec=yj.vec)
        lp.mat <- matrix(lp.grid, nrow=200, ncol=200)
        p.mat <- exp(lp.mat - max(lp.mat))
        contour(eta.vec, K.vec, p.mat, ylab="K", xlab=expression(eta))
```



Next, she defines a transformation of parameters η and K, $\theta_1 = logit(\eta)$ and $\theta_2 = log(K)$. The function in the code below finds unnormalized posterior probabilities on the log scale for given values of θ_1 and θ_2. The function does this by first solving for η and K and then calculating the log likelihood. The prior is given on the scale of θ_1 and θ_2, and I assume it was found via transformation.

$$p(\theta_1, \theta_2) = \frac{\theta_2}{(1 + e^{\theta_2})^2}$$

$$logp(\theta_1, \theta_2) = \theta_2 - 2log(1 + e^{\theta_2})$$

Since θ_1 is a function of η only and θ_2 is a function of K only, we can assume that θ_1 and θ_2 are independent (because we assumed η and K were independent). The marginal priors for θ_1 and θ_2 are as follows:

$$p(\theta_1) \propto 1$$
$$p(\theta_2) = \frac{\theta_2}{(1+e^{\theta_2})^2}$$

In the `BB.logpost` function, she adds the log likelihood and the log of the prior for a specified (θ_1, θ_2) pair, and the output of this function are unnormalized posterior probabilities (on the log scale) for given values of θ_1 and θ_2.

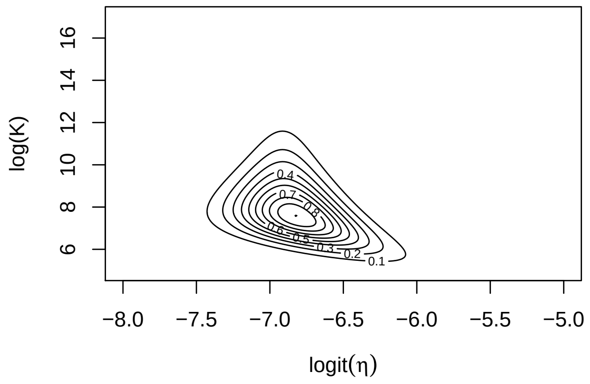
```
#### Now, let's transform the parameters to the real line
## theta1 <- logit(eta)
## theta2 <- log(K)

     BB.logpost <- function(theta.vec, n.vec, y.vec) {
       eta <- exp(theta.vec[1])/(1+exp(theta.vec[1]))
       K <- exp(theta.vec[2])
       ll <- sum(lbeta(K*eta + y.vec, K*(1-eta) + n.vec - y.vec) -
             lbeta(K*eta, K*(1-eta)))
       #log.prior <- -log(eta) - log(1-eta) - 2*log(1+K)
       #log.Jacob <- (theta.vec[1]+theta.vec[2]) - 2*log(1+exp(theta.vec[1]))
       log.rest <- theta.vec[2] - 2*log(1+exp(theta.vec[2]))
       #trans.log.post <- ll + log.prior + log.Jacob
       trans.log.post <- ll + log.rest
       return(trans.log.post)
     }
```

Next, she defines a grid of reasonable values for θ_1 and θ_2. Then she finds the unnormalized posterior probability for each combination of θ_1 and θ_2 in the grid. These posterior values are displayed in the contour plot below. Again, she subtracts the maximum of the log posterior from each log posterior value before exponentiating so that the resulting values are on a scale that easier to deal with computationally.

```
  theta1.vec <- seq(-8,-5,length=200)
     theta2.vec <- seq(5,17, length=200)
     theta.grid <- expand.grid(theta1.vec, theta2.vec)
     logpost.grid <- apply(theta.grid, 1, BB.logpost, n.vec=nj.vec, y.vec=yj.vec)
```

```
        logpost.mat <- matrix(logpost.grid, nrow=200, ncol=200)
        post.mat <- exp(logpost.mat - max(logpost.mat))
        contour(theta1.vec, theta2.vec, post.mat, ylab="log(K)",
              xlab=expression(logit(eta)))
```



Next she finds the values of θ_1 and θ_2 that have the largest posterior value. It turns out that this maximum occurs at the mode which is approximately $(\theta_1, \theta_2) = (-6.82, 7.58)$, which she finds with the `optim` function. I assume she finds the mode to give us a "best guess" for θ_1 and θ_2. She also finds the inverse of the negative hessian matrix which is supposed to be the variance covariance matrix for the estimates for θ_1 and θ_2. **Note***: I know what $Var(\hat{\theta}_1)$ means in Frequentist land, but what does the variance of an estimator mean in Bayesian land? I'm not sure what this matrix is telling us.**

```
## $par
## [1] -6.82  7.58
##
## $value
## [1] -571
##
## $counts
## function gradient
##       63       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
```

```
## $hessian
##        [,1]   [,2]
## [1,] -15.97 -1.759
## [2,]  -1.76 -0.935
## [1] -6.82  7.58
##        [,1]   [,2]
## [1,]  0.079 -0.149
## [2,] -0.149  1.348
```

She decides that we are going to use rejection sampling to draw from this posterior distribution instead of using a multivariate normal approximation. In rejection sampling, we have to come up with a proposal distribution that we can draw from. In the following comments, she says that there is a multivariate t-distribution in the `LearnBayes` package that would be a good proposal distribution, so we download this package and look at the help file for the `dmt` function used to draw from the multivariate t distribution.

```
    ## Using this information we could use a multi-variate normal approximation for
    ##  the posterior distribution,  though we would rather sample from it directly.
    ##  We can also use this information to come
        ##  up with a proposal distribution for a rejection sampling algorithm.  If
        ##  we use the normal it will probably be too skinny in the tails and may
        ##  not bound our posterior, so let's try a t-distribution on very few d.f.
        ##    to get fatter tails.
    ##  There is a nice multivariate t function in the LearnBayes package
    ##   install.packages("LearnBayes")
       library(LearnBayes)
       help(dmt)

        #### NOW let's use REJECTION SAMPLING to obtain samples from the joint
        #### posterior distribution of logit(eta) and log(K)

        ### STEP 1:  Find a proposal density of a simple form that, when multiplied
        # by an appropriate constant covers the posterior density of interest
```

We then set about finding the correct c for rejection sampling. She finds this constant in the below code. To find the constant, she finds the maximum difference between the proposal distribution and the target distribution. She writes a function called `post.prop.diff` to find the difference between the log of the proposal distribution and the log of the target posterior distribution, and then she finds the maximum of this function. To maximize this difference, she uses the `optim` function, and it seems like the maximum found does depend on the starting values that she specifies. For some starting values, she finds the maximum to be -570.07 occuring at $(-6.887, 7.507)$ and for other starting values she finds the maximum to be -569.3335, occuring at $(-6.8899, 12.46)$. I think she uses the -569.3335 maximum in the acceptance ratio (see below) because it is slightly larger than the other maximum found.

```
    #We essentially want to maximize log(g(theta|y)) - log(p(theta))
        #  Let's use optim() again to do this, by first writing a function to maximize

        post.prop.diff <- function(theta, n.vec, y.vec, t.params) {
                post.part <- BB.logpost(theta, n.vec=n.vec, y.vec=y.vec)
                proposal.part <- dmt(theta, mean=t.params$m, S=t.params$var,
                             df=t.params$df, log=TRUE)
                d <- post.part - proposal.part
                return(d)
                }

        t.params.set <- list(m=c(-6.8, 7.6), var=2*Var, df=4)

        d.out <- optim(c(-7, 7), post.prop.diff, n.vec=nj.vec, y.vec=yj.vec,
                t.params=t.params.set, control=list(fnscale=-10))

        d.out$par   #-6.8899, 12.46 So, max value of d occurs at (-6.88, 12.46)
```

```
## [1] -6.89  7.51
```

```
                    # -6.886769  7.507773  (for some starting values)

  post.prop.diff(c(-5, 7), n.vec=nj.vec, y.vec=yj.vec, t.params=t.params.set)
```

```
## [1] -611
```

```
        d.max <- post.prop.diff(c(-6.8899, 12.46), n.vec=nj.vec, y.vec=yj.vec,
             t.params=t.params.set) #-569.3335
        dmax <- post.prop.diff(c(-6.8868, 7.5077),n.vec=nj.vec, y.vec=yj.vec,
            t.params=t.params.set) #-570.07
```

Here she begins the rejection sampling. First she takes 1000 draws from the proposal multivariate t-distribution.

```
  ##### NOW, finally we obtain a sample using rejection sampling...we will do
  ###   it in vector form, rather than a loop

        #1. Draw 1000 draws from the proposal distribution (a multivariate-t)
          n.draws <- 10000
          prop.draws <- rmt(n.draws, mean=t.params.set$m, S=t.params.set$var,
                            df=t.params.set$df)
```

Then, she evaluates the posterior and proposal distributions at all of the draws from above and calculates the acceptance ratio. The acceptance ratio is as follows. Recall that the rejection constant, c, was calculated on the log scale.

$$r = \frac{t(\theta)}{c * g(\theta)} = e^{log(t(\theta)) - c - log(g(\theta))}$$

Remember, $t(\theta)$ is the posterior distribution that we are trying to obtain draws from and $g(\theta)$ is the multivariate t-distribution that is used as the proposal distribution. To decide if a draw from the proposal distribution is accepted, she draws from a $Unif(0,1)$ distribution. If the uniform draw is less than the acceptance probability, that draw is accepted and recorded as a draw from the target distribution (which is the posterior distribution in this case). These draws are stored in the vector `theta.draws`.

```
      #2. Evaluate the posterior and the proposal distribution at all the values
    log.post <- apply(prop.draws, 1, BB.logpost, n.vec=nj.vec, y.vec=yj.vec)
          log.g.theta <- dmt(prop.draws, mean=t.params.set$m, S=t.params.set$var,
                             df=t.params.set$df, log=TRUE)


      #3. Calculate the ratio
    accept.probs <- exp(log.post - log.g.theta - d.max)

      #4. Pick off the draws that we accept
    theta.draws <- prop.draws[runif(n.draws) <= accept.probs,]
```

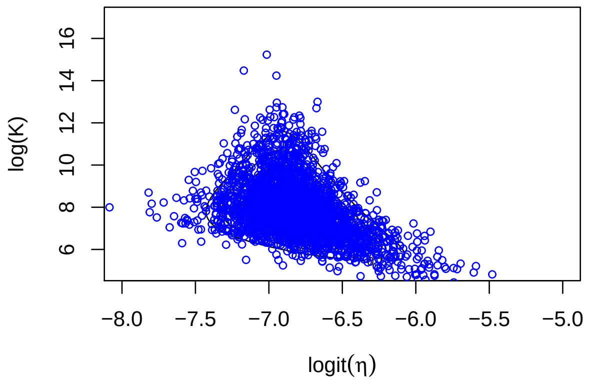
Then she calculates the proportion of draws that were accepted.

```
    #What percent did we accept?
           length(theta.draws[,1])/n.draws #only 28% inefficient, but easy to find?

## [1] 0.251
```

Then, she plots the draws from the posterior distribution found from the rejection sampling procedure.

```
            contour(theta1.vec, theta2.vec, post.mat, ylab="log(K)",
                    xlab=expression(logit(eta)))
            points(theta.draws[,1], theta.draws[,2], pch=1, col=4, cex=0.75)
```

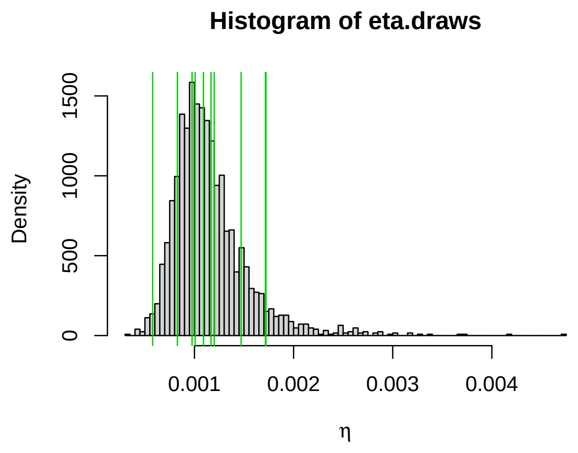


Lastly, she transforms back to η so that she can make inference about the mean of the Beta-Binomial distribution. The green lines on the plot below are the proportion of deaths for the 20 largest cities in Missouri (the original data). She can now use this distribution to find a best guess for η (such as the median), and find posterior intervals. *****Note: Where did you get the (0.0006410274, 0.0020984802) interval? I assume this is the 95% posterior interval found from the analytical posterior distribution, but how did you find it?**

```
        ##Inference about the mean, eta?
           eta.draws <- exp(theta.draws[,1])/(1+exp(theta.draws[,1]))
           hist(eta.draws, nclass=100, col="lightgray", freq=FALSE,
                        xlab=expression(eta))
           median(eta.draws) #0.001081160

## [1] 0.00108

           abline(v=yj.vec/nj.vec, col=3) # Add the observed proportions
```



```
          quantile(eta.draws, c(.025, .975)) #0.0006410274 0.0020984802

## 2.5% 97.5%
## 0.000648 0.002130

                             #may need more draws to better capture the right-hand
                             #  tail of the distn
```

2. Originally, I did have some questions about this R script, but I went to your office and got them answered. Here, I'll try to explain what I learned from this exploration. When going through this R tutorial, I focused on the two different methods used to sample from a posterior distribution.

   First, we use a grid approximation to approximate the posterior distribution. Using the first method of drawing from this distribution, we simply randomly draw a α, β pair. Then, we make the draw more continuous by addding jitter. We do this by adding a random draw from a uniform distribution over the step size of the grid to each α and β.

   The second method makes use of the equality $p(\alpha, \beta|y) = p(\beta|\alpha, y)p(\alpha|y)$. We take the posterior distribution found using the grid approximation method, and we find the marginal posterior distribution for α by summing over all β's on the grid. Then, we randomly draw an

α, and given this α we randomly draw a β within this row of the posterior probability matrix.

I think I could have written my own code to implement the first method, but the code for the second method was a little more involved. It is nice that I now have an example of how to do it. When I first started going through this, I thought of these two methods as very different ways to sample from a posterior distribution, but somewhere along the way I had this great realization that they are the same! They both rely on the grid approximation method to find the posterior probabilities at each combination of α and β on the grid, and they are both ways to arrive at a randomly drawn α, β pair. If the parameter space was 3 dimensional, it seems like it would be pretty easy to extend either of these methods to draw a random combination of the 3 parameters.