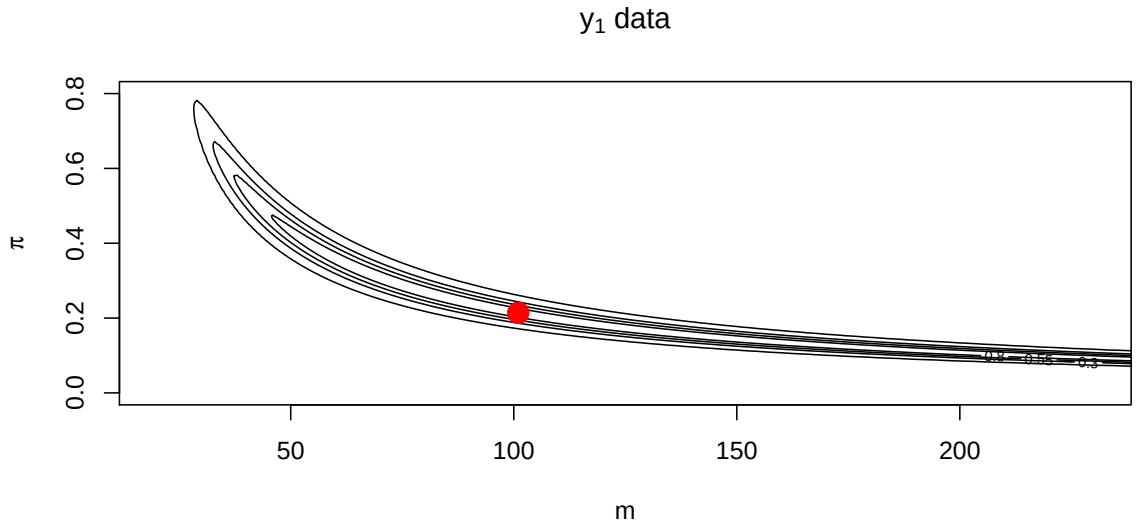
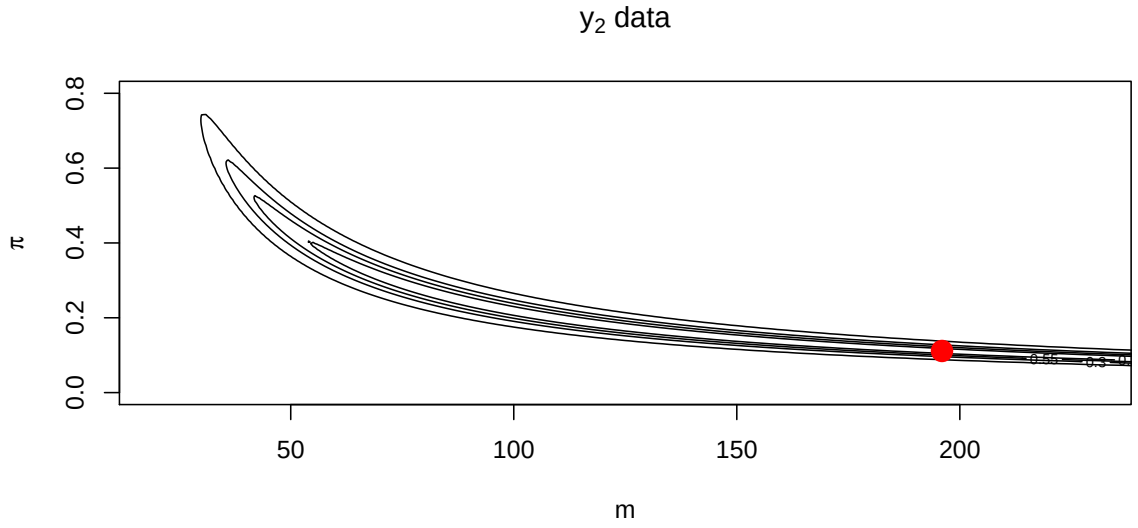
Bayes Midterm 2
Leslie Gains-Germain

1. (a) The following example was inspired by Claire and Jordan's consulting project. Suppose there are five reservations in Montana. The number of children eligible to receive dental sealants on the reservation is unknown because it's hard to keep track of this population. A success is observed when a child comes to a dental clinic to have dental sealants applied. The number of children on each of the five reservations who had dental sealants applied in 2014 was observed. The probability of having the sealants applied and the number of eligible children is assumed to be the same for all five reservations.

   (b) The MLE occurs near $m = 101, \pi = 0.21$.

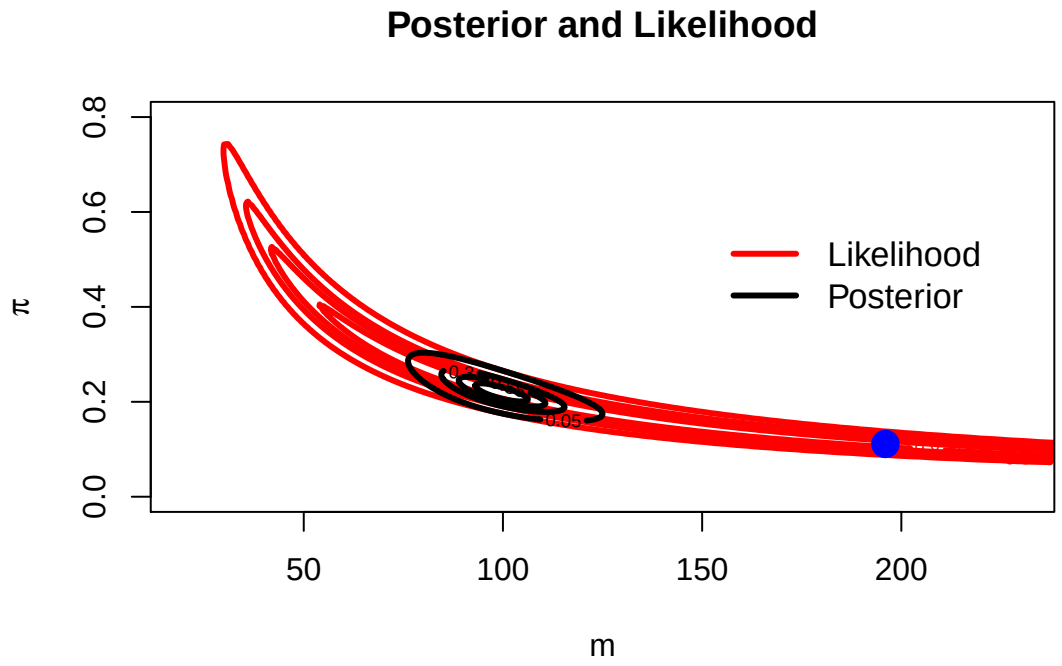


   (c) The MLE occurs near $m = 196, \pi = 0.11$.



(d)

(e) i. The joint posterior distribution of (m, π) is shown below.



ii. My work for deriving the complete conditionals for m and π is shown below.

$$\begin{aligned} p(\pi|y,m) &= \frac{p(\pi,y,m)}{p(y,m)} \\ &\propto p(\pi,y,m) \\ &\propto p(y|\pi,m)p(m,\pi) \\ &\propto p(y|\pi,m)p(\pi) \\ p(m|y,\pi) &= \frac{p(\pi,y,m)}{p(y,\pi)} \\ &\propto p(\pi,y,m) \\ &\propto p(y|\pi,m)p(m,\pi) \\ &\propto p(y|\pi,m)p(m) \end{aligned}$$

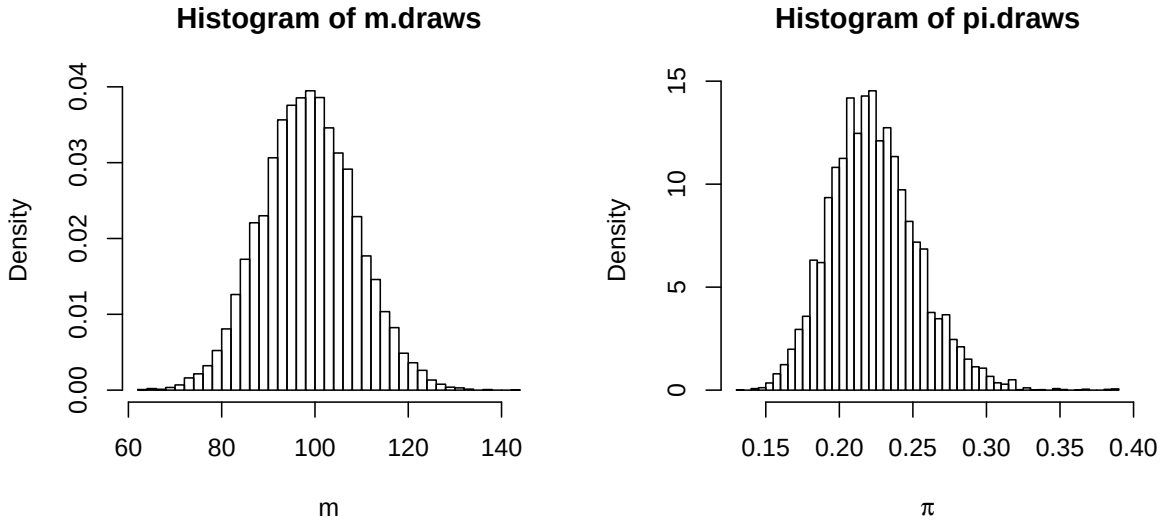
iii. My code for the Gibbs sampler is shown below.

```
#function to calculate complete conditional for pi on log scale
log.pi.cc.fun <- function(m.pi, y.vec){
  loglik.fun(m.pi, y.vec)
}

#check function
#log.pi.cc.fun(c(40, 0.5), y2.data)
```

```r
#function to calculate complete conditional for m on log scale
log.m.cc.fun <- function(m.pi, y.vec){
  log.post.fun(m.pi, y.vec)
}

#check function
#log.m.cc.fun(c(40, 0.5), y2.data)

nchain <- 3
nsim <- 10000
m.pi.mat <- array(NA, dim=c(nsim, 2, nchain))

#keep track of acceptance ratios
jump.mat <- matrix(NA, nrow=nsim-1, ncol=2)

#specify starting values for each chain
m.pi.mat[1, 1:2, 1] <- c(100, 0.4)
m.pi.mat[1, 1:2, 2] <- c(50, 0.2)
m.pi.mat[1, 1:2, 3] <- c(150, 0.4)

#define standard deviations for normal proposal distributions
sd.scale <- c(10, 5)

set.seed(230923)

for (j in 1:nchain) {
  for (i in 2:nsim) {
    #set pi equal to the starting value
    pi <- m.pi.mat[i-1, 2, j]

    #now draw m from complete conditional with Metropolis Hastings Algorithm
    m.cur <- m.pi.mat[i-1, 1, j]
    m.cand <- rpois(1, lambda=m.cur)

    log.r.num.m <- log.m.cc.fun(c(m.cand, pi), y.vec = y2.data) +
                    dpois(m.cur, lambda=m.cand, log = TRUE)

    log.r.denom.m <- log.m.cc.fun(c(m.cur, pi), y.vec = y2.data) +
                      dpois(m.cand, lambda=m.cur, log = TRUE)

    log.r.m <- log.r.num.m - log.r.denom.m

    p.accept.m <- min(1, exp(log.r.m))

    u.vec <- runif(2)

    ifelse(u.vec[1] <= p.accept.m, m.pi.mat[i, 1, j] <- m.cand,
           m.pi.mat[i, 1, j] <- m.cur)

    jump.mat[i-1, 1] <- ifelse(u.vec[1] <= p.accept.m, 1, 0)

    #now fix m at its value in the ith iteration
    m <- m.pi.mat[i, 1, j]
```

```
    #draw pi from complete conditional with metropolis hastings algorithm
    pi.cur <- m.pi.mat[i-1, 2, j]
    pi.cand <- rbeta(1, 2, sd.scale[2])

    log.r.num.pi <- log.pi.cc.fun(c(m, pi.cand), y.vec = y2.data) +
                    dbeta(pi.cur, 2, sd.scale[2], log = TRUE)

    log.r.denom.pi <- log.pi.cc.fun(c(m, pi.cur), y.vec = y2.data) +
                      dbeta(pi.cand, 2, sd.scale[2], log = TRUE)

    log.r.pi <- log.r.num.pi - log.r.denom.pi

    p.accept.pi <- min(1, exp(log.r.pi))

    u.vec <- runif(2)

    ifelse(u.vec[2] <= p.accept.pi, m.pi.mat[i, 2, j] <- pi.cand,
           m.pi.mat[i, 2, j] <- pi.cur)

    jump.mat[i-1, 2] <- ifelse(u.vec[2] <= p.accept.pi, 1, 0)
  }
}
```

(f) The marginal posterior distributions for m and π are shown below. The posterior probability that $m > 100$ is 0.443 and the posterior probability that $\pi < 0.3$ is 0.989. Since m and π are independent, the joint posterior probability that $m > 100$ and $\pi < 0.3$ is $0.443 * 0.989 = 0.438$.



(g) disads

(h) The hierarchical version of the model puts priors on λ and priors on parameters of a $Beta(\alpha, \beta)$ distribution. I chose to put priors on the prior mean, $\eta = \frac{\alpha}{\alpha+\beta}$ and the somewhat approximate prior standard deviation, $\sigma = \sqrt{\frac{1}{\alpha+\beta}}$, kind of like what we did in

the Beta-Binomial example with the stomach cancer data.

$$
\begin{aligned}
y_i &\sim Bin(m, \pi) \\
\pi &\sim Beta(\frac{\eta}{\sigma^2}, \frac{1-\eta}{\sigma^2}) \\
\sigma &\sim Uniform(0, 100) \\
\eta &\sim Uniform(0, 1) \\
m &\sim Poisson(\lambda)\lambda \qquad\qquad \sim Uniform(0, 500)
\end{aligned}
$$

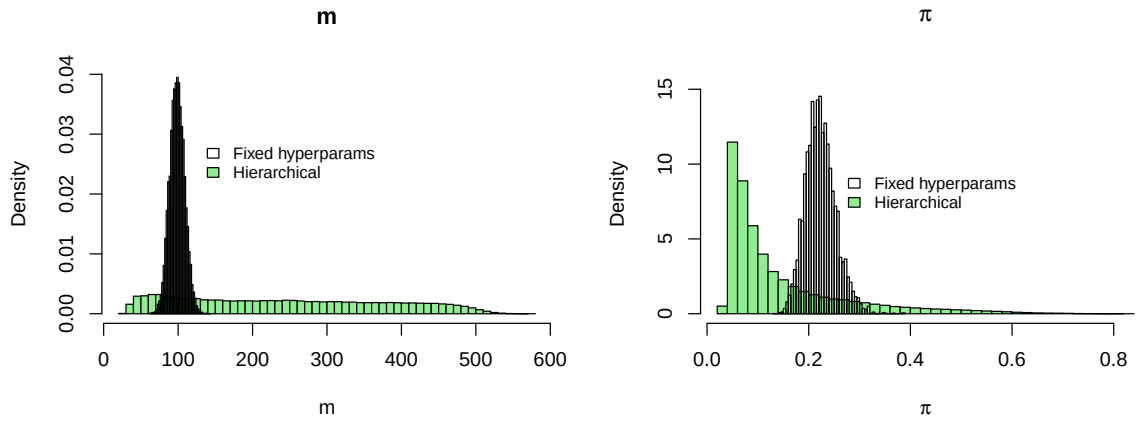
(i) The JAGs code I used to obtain draws from the above model is shown below.

```
##write model file first
cat("
model
{
for(i in 1:N)
{
y[i] ~ dbin(pi, m)
}

pi ~ dbeta(eta/sigma^2, (1-eta)/sigma^2)
eta ~ dunif(0, 1)
sigma ~ dunif(0, 100)

m ~ dpois(lambda)
lambda ~ dunif(0, 500)
}",
file="model1.jags")
```

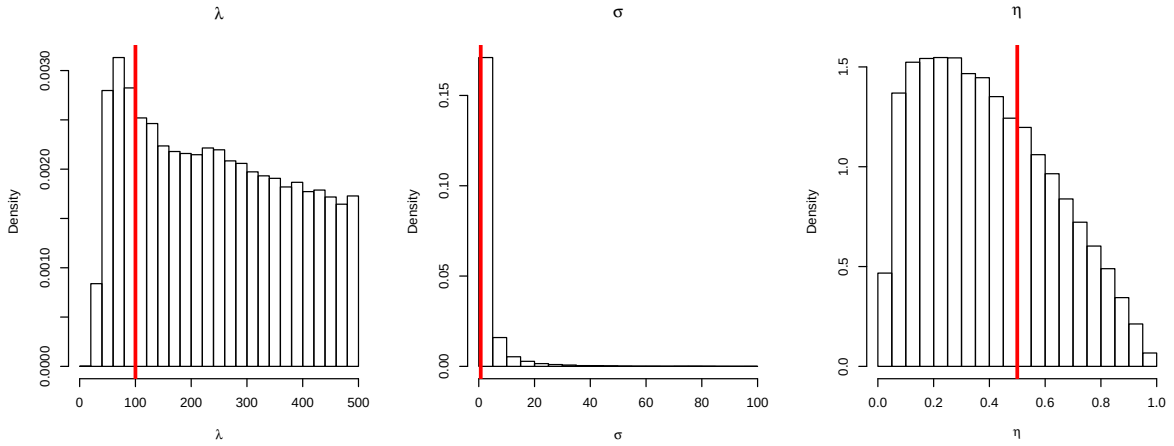
```
##jags call
library(R2jags)
set.seed(52)


dental.data <- list(N=length(y2.data), y=y2.data)

inits <- list(list(pi=0.4, eta = .00003, m = 100, lambda = 500,
                   sigma = 1.20),
              list(pi=0.2, eta = .00003, m = 50, lambda = 150,
                   sigma = 2.24),
              list(pi=0.4, eta = .00003, m = 150, lambda = 300,
                   sigma = 1.58))
n.chain <- 3

#warmup
warmup.model1 <- jags.model("model1.jags", data=dental.data, n.chains=n.chain, inits= inits, n.adapt

#parameters to save
params <- c("pi", "eta", "sigma", "m", "lambda")

n.iter=50000
```

```
#running the model for real
model1 <- coda.samples(warmup.model1, params, n.iter=n.iter)
```

(j) The plots below compare the posterior distributions for π and m obtained in part (f) to those from the hierarchical model. There is much more variability in the posterior draws for m and π in the hierarchical model compared to the model with fixed hyperparameters. I think this makes sense because when the hyperparameter values are approximated, the model does not incorporate uncertainty in these parameters. With such a small sample size, the additional source of uncertainty accounted for in the hierarchical model makes a big difference in the results.



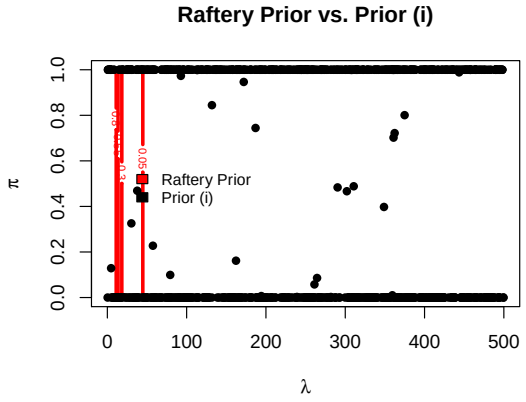
The posterior distributions of the hyperparameters are shown below, with the fixed values used in (e).



(k) i. When I first looked at this prior, I thought he was trying to give higher density to values of μ and π that yield smaller numbers of successes. But, on second thought, I think he was just trying to specify a vague prior over the entire range of possible expected observations. I do think he was trying to get at a non-informative prior.

ii. When I look at the plot of the Raftery Prior, it looks like small values of μ with any value of π are most probable. When I generated draws from the priors I specified

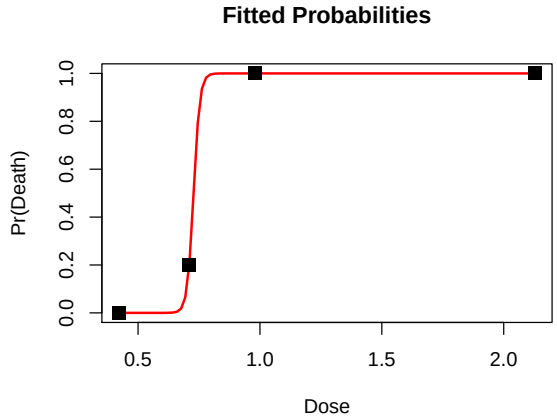
in (i), however, it looks like values of μ between 0 and 500 are equally likely, with π either very close to 0 or very close to 1. It seems like the raftery prior makes lower expected number of observations most probable by making lower values of λ, and thus m, more probable. The prior used in i makes values of λ between 0 and 500 equally likely, and values of π close to 0 or 1 are most likely across all values of λ. I included the code for this part in the appendix.



2. (a) The simple logistic regression model for estimating the dose-death relationship is as follows. I treat dose as a continuous predictor variable. The empirical probabilities of death at each dose as well as the fitted probabilities from the glm() model are shown on the plot below. The dose at which half of the frogs are estimated to die is $\frac{0.5+56.97}{78.29} = 0.734$ grams per ml.

$$y_{ij} \sim Bin(5, \pi_j)$$
$$logit(\pi_j) = \alpha + \beta x_j$$

```
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
```



(b) The model I will use to do the analysis in a Bayesian framework is shown below.

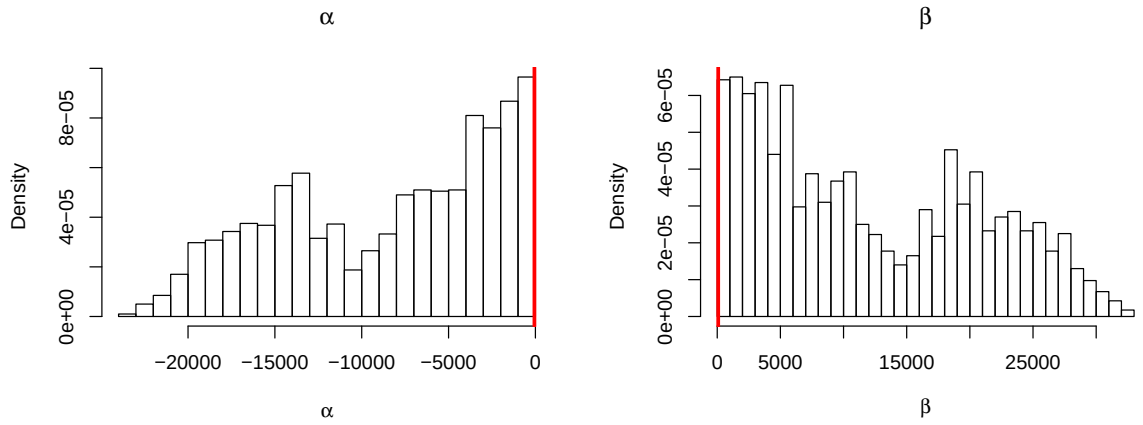
$$\begin{aligned} y_{ij} &\sim Bern(logit^{-1}(\alpha + \beta x_j)) \\ p(\alpha) &\propto 1 \\ p(\beta) &\propto 1 \end{aligned}$$

(c) The model code is shown below.

```
data {
  int<lower=0> N;
  vector[N] x;
  int<lower=0,upper=1> y[N];
}
parameters {
  real alpha;
  real beta;
}
model {
  y ~ bernoulli_logit(alpha + beta * x);
}
```

```
require(rstan)
set.seed(23)
data <- with(frog.data, list(y = death, x = dose, N = length(death)))

model2 <- stan_model(file = "~/Documents/Stat532/exams/exam2/model2.stan",
                     model_name = "model2")
samp2 <- sampling(model2, chains = 4, iter = 2000, data = data)
```

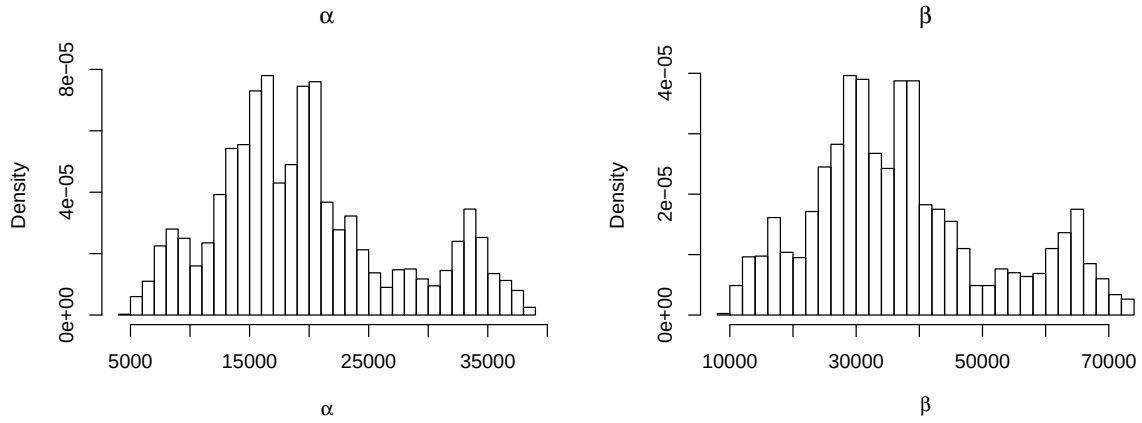
(d) The posterior draws for the y-intercept (α) and the dose effect (β) are shown below. The red line indicates the estimate from the above generalized linear model.



(e) Discuss and explain

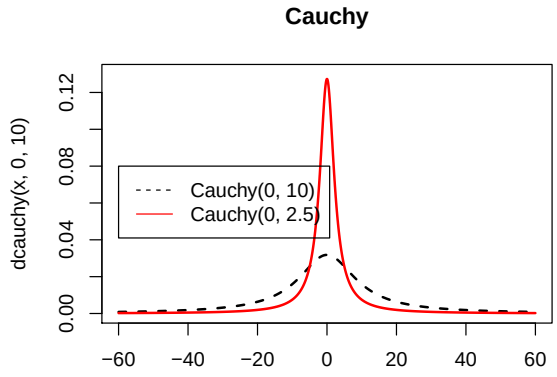
(f) The plots of the posterior draws for α and β are shown below for the model with standardized dose as the predictor variable. The posterior draws for α have changed, but this is expected because α is now the log odds of death at the average dose. Standardizing

dose does not appear to change inference about the effect of dose on the log odds of survival. Also, the efficiency of the sampler has not changed noticeably.



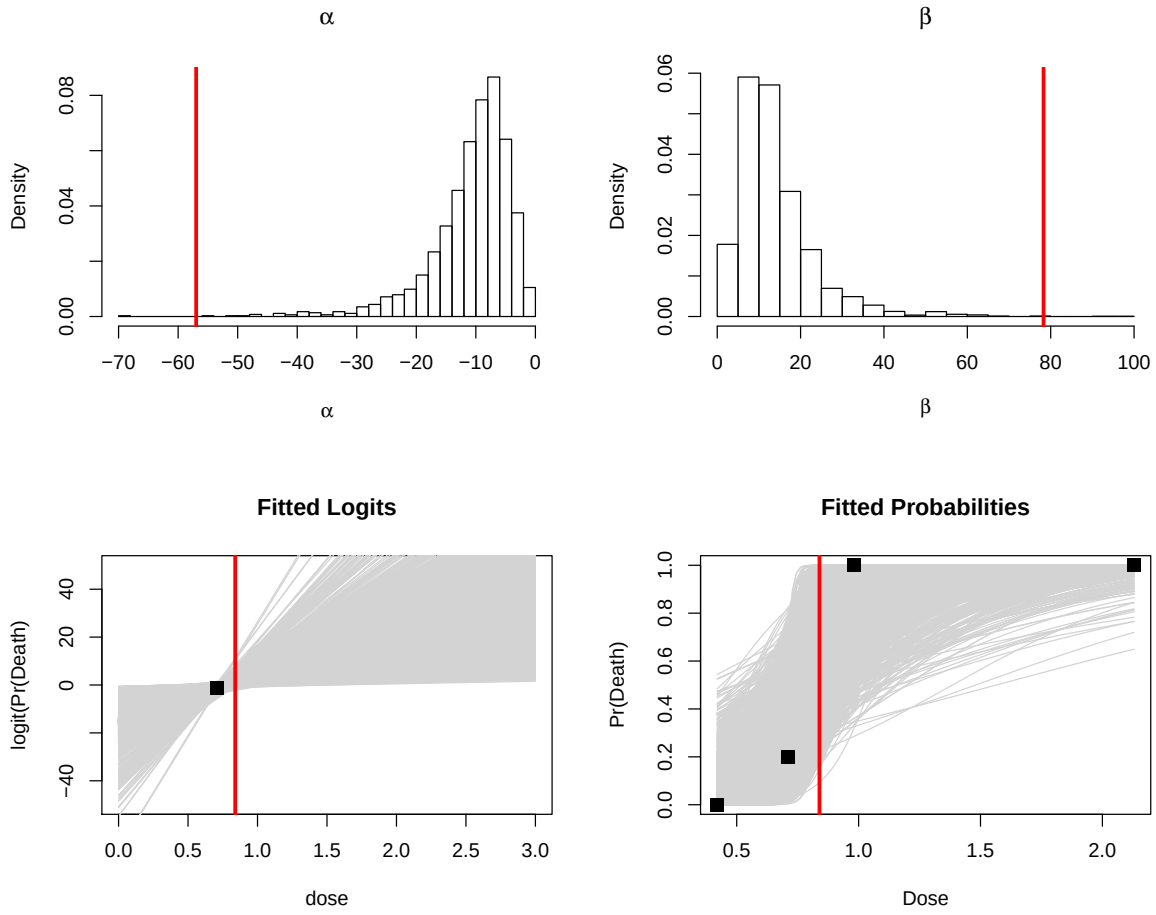
(g) The cauchy priors are shown below. For the intercept, the $Cauchy(0, 10)$ prior allows for the possibility of any value over the whole real line, but the probable values for the intercept are between -30 and 30. This is a huge spread on the logit scale, and if we transform back to the original scale, this prior allows for the probability of death at the average dose to be between $logit^{-1}(-30) = 9.4 * 10^{-14}$ and $logit^{-1}(30) \approx 1$! With such a large spread on the original scale, this prior is weakly informative because it incorporates the natural constraints that the probability of death must be between 0 and 1.

For the slope, the probable prior values are between -20 and 20 on the logit scale. This means that for a one standardized unit increase of dose, the odds of death can change by a multiplicative factor between the values of $e^{-10} = 4.54 * 10^{-5}$ and $e^{10} = 22026.5$. With the prior density spread out over such a large range over values, very little prior is incorporated about the dose effect. It does incorporate the natural constraint that the change in the odds must be greater than 0. For this parameter, I think we could do a better job at incorporating the prior knowledge that $\beta > 0$ (it's a lethal toxin). I come up with a better weakly informative prior in part (i).

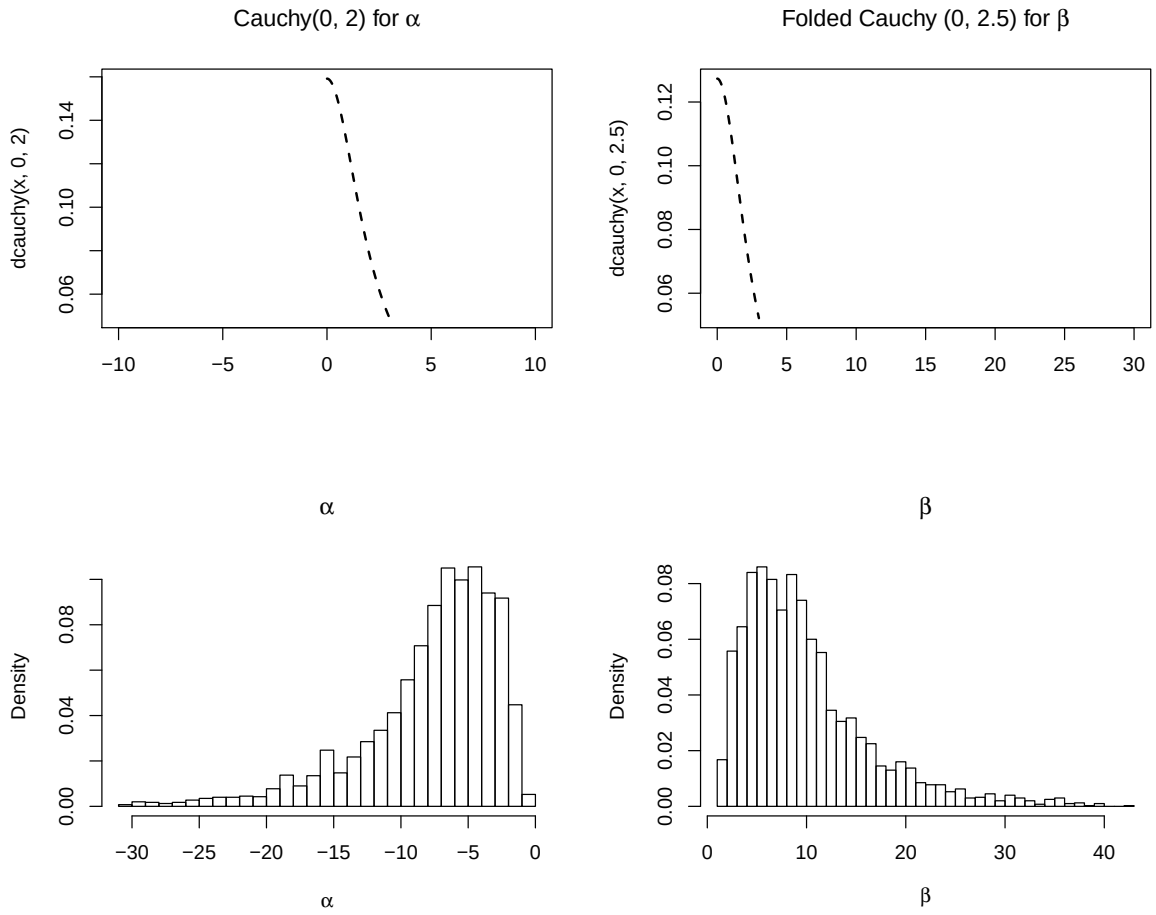


(h) The posterior distributions of α and β are shown below. The regression lines displaying the fitted logits, and the fitted probability curves are shown for all 4000 posterior draws are also shown.

talk about differences...

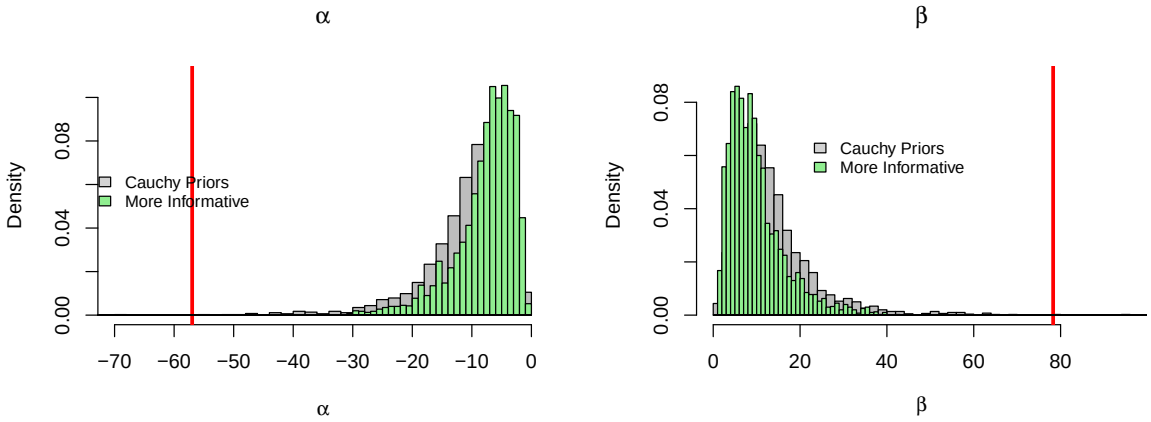


(i) I'll use the Cauchy prior for α again with a smaller scale parameter. I'll use the Cauchy(0, 2) prior for α, which makes the more likely values for α (the probability of death at the average dose) to be between about 0.01 and 0.99. This is still not very informative, but it is more informative than the Cauchy(0, 10). For β, I'll use a folded Cauchy prior. The problem does state that this drug is known to be lethal to frogs, so we should incorporate into our prior the knowledge that β is greater than 0. I'll use a folded Cauchy(0, 2.5). I'm still reflecting little knowledge in the magnitude of β, but this prior is more informative in the sense that it does not allow for negative values of β. The results are shown below for this model.



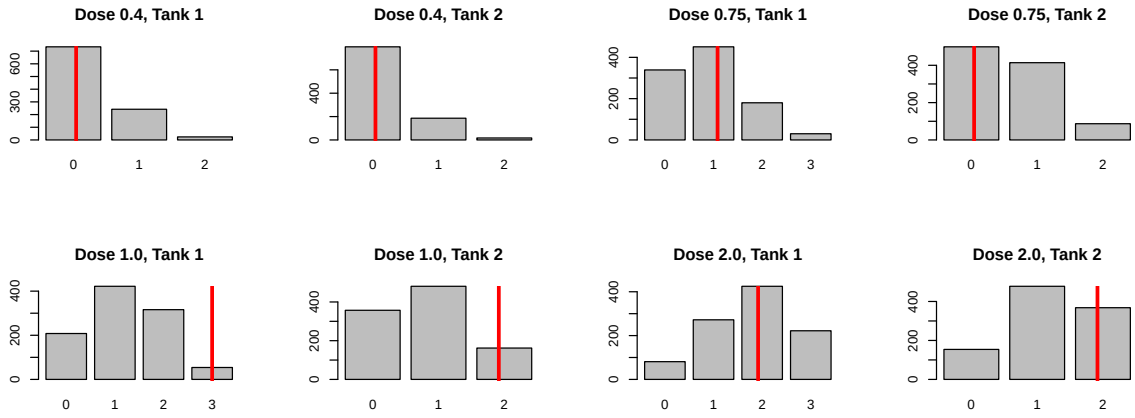
(j) In the histograms below, I compare the results for the models from (g) and (i). You can see the posterior draws of α and β are drawn closer to 0 in the model from part (i).

I think the analysis in part (i) is most appropriate. The results from the improprer priors in part (c) do not make sense, not to mention I get a warning message when I run the model. The results from the Cauchy priors in part (g) are reasonable, but I think the prior I chose in part (i) is more appropriate because I chose priors that incorporated reasonable constraints in the context of the problem but were still vague enough to reflect little prior knowledge. The Cauchy priors in part (g) were more diffuse than they needed to be to reflect little prior knowlege in this context. I guess this is just another situation where it's smarter to think carefully about your priors instead of just choosing default non-informative priors.



(k) I do not expect the assumption of independence to be met because I would expect the frogs in the same tank to have more similar death rates than frogs in different tanks. There could be other sources of dependence in the design that we are not told about such as age or health of the frogs. For this problem, I'll focus on investigating whether the death rates are more similar for frogs in the same tank.

To implement the posterior predictive check, I will generate a large number of posterior predictive samples. For each posterior predictive sample, I'll assume that the first three frogs within a dose came from tank 1, and the last two frogs within that dose came from tank 2. Then, for each posterior predictive sample, I'll calculate the number of deaths within each tank for each dose. I'll display the histograms showing the distribution of deaths within each tank at each dose for all posterior predictive samples. I'll then compare to what was observed in the original dataset. My posterior predictive displays will look something like the following. I simulated these posterior predictions for the purpose of example, and I also show the code here.



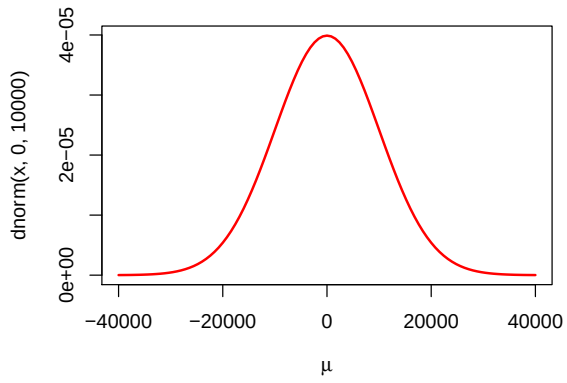
Since the model generates posterior predictions under the assumption of independent observations, the posterior predictions represent what we would expect to see if frog deaths occurred independently within tanks. If the observed data showed more tanks with all frogs dying or no frogs dying than predicted by the model, I would suspect

dependence among frogs within a tank. For example, suppose the number of deaths were most often 1 death per tank in the posterior predictive samples. Then, if we observed either all or none of the frogs dying within a tank in the original sample, I would suspect that one frog dying in the tank is affecting the probability of death for another frog in that tank. An example of this would be the barplot shown for Dose 1.0, Tank 1 in the above figure. Biologically, I think it seems reasonable that if one frog dies within a tank and is not cleaned out right away, it would decrease the overall health of the other frog(s) in the tank and increase their probability of death.

(l) A fully Bayesian hierarchical model would be as follows. This model incorporates an adjustment for each tank ($k = 1, 2$), and it accounts for uncertainty in the parameters for the population of tank adjustments.

$$\begin{aligned} y_{ijk} &\sim Bern(logit^{-1}(\alpha + \beta x_j + \gamma_k)) \\ p(\alpha) &\propto 1 \\ p(\beta) &\propto 1 \\ \gamma_k &\sim N(0, \sigma^2) \\ p(\sigma^2) &\propto 1 \end{aligned}$$

3. (a) Take for example the $N(0, 10000)$ prior that people sometimes use as a default 'uninformative prior' in canned software packages when they are forced to choose a proper prior. The relative density does appear to be higher for values of the parameter near 0 (see below). But, the difference in densities is very small because the prior is spread out over such a large range of values (note the peak density is around 0.00004). As a result, the chance of drawing a value between 0 and 10 isn't that much different than the chance of drawing a value between 20000 and 20010. This prior is uninformative in the sense that a very wide range of prior values are probable, and the difference in densities over this very wide range of values is small because the prior is so diffuse.



(b) The posterior is sometimes sensitive to priors that are commonly regarded as non-informative. Take for example the $InvGam(0.01, 0.01)$ prior that is commonly used as a non-informative prior for variance parameters. Recent work has shown that applying this prior to a group level variance parameter in a hiearchical model can lead to very

different posterior distributions than if a different vague, diffuse prior is used (Gelman 2006). This goes to show that a sensitivity analysis of the posterior to the priors should be performed even if the priors used are commonly known as non-informative.

4. The proposal distribution is a computational tool used to obtain draws from the posterior distribution. We need to get the spread and center of the proposal distribution just right to maximize efficiency of the Metropolis Hastings algorithm. So, we often look at the likelihood to get a ballpark idea of what the spread of the proposal distribution should be. The key difference between the proposal distribution and the prior distribution is that the proposal distribution does not contribute to the model at all, it is simply used in computation. If we try to use the likelihood to inform the prior distribution, we'd be using an empirical Bayes approach. If you allow the data to inform the prior, you're not doing a true Bayesian analysis.

5. The data won't always be representative of the population, and with few data it is hard to quantify the amount of variability in the population. As a result, classical inference is often wrong if few data available and are not representative of the population. The ability to incorporate prior knowledge in a Bayesian analysis can help us see when the data collected are not representative of the truth, and it can help us fit a better, more informed model. I think of the prior as a piece of known information going into the fitted model. As I saw in problems 1 and 2, the prior can really help to inform results and lead to reasonable posteriors even when the few data available don't seem to provide much information at all.

References

Gelman, Andrew. "Prior Distributions for Variance Parameters in Hierarchical Models(Comment on Article by Browne and Draper)." International Society for Bayesian Analysis 1.3 (2006): 515-34. Print.

R Code Appendix

```
set.seed(908)
pi.draw <- numeric(1000)
mu.draw <- numeric(1000)
lambda.draw <- numeric(1000)
for(i in 1:1000){
  lambda.draw[i] <- runif(1, 0, 500)
  m.draw <- rpois(1, lambda.draw)

  sigma.draw <- runif(1, 0, 100)
  eta.draw <- runif(1, 0, 1)
  pi.draw[i] <- rbeta(1, eta.draw/sigma.draw^2, (1-eta.draw)/sigma.draw^2)
}

lambda.vals <- seq(0, 100, length=1000)
pi.vals <- seq(0, 1, length=1000)
mu.vals <- lambda.vals*pi.vals
mu.pi.vals <- expand.grid(mu.vals, pi.vals)
mu.pi.vals <- as.matrix(mu.pi.vals)
prior.vals.fun <- function(mu.pi){
  1/mu.pi[1]
```

```
}

prior.vals <- apply(mu.pi.vals, 1, prior.vals.fun)

prior.mat <- matrix(prior.vals, nrow=length(mu.vals), ncol=length(pi.vals))

contour(lambda.vals, pi.vals, prior.mat, levels=seq(0.05, 0.95, 0.25),
        xlab=expression(lambda), ylab=expression(pi), main="Raftery Prior vs. Prior (i)",
        lwd=3, col="red", xlim = c(0, 500))
points(lambda.draw, pi.draw, col="black", pch=16)

legend(20, 0.6, bty="n", legend=c("Raftery Prior", "Prior (i)"),
       fill=c("red", "black"), cex=0.8)
```