Bayes and STAT 550 Component

Paul Harmon

3/25/2019

Introduction

This document contains my solutions to the STAT 532/STAT 550 component of the PhD Comprehensive Exam. In accordance with the requirement for reproducibility, this document was generated as an R-Markdown file with all relevant code included either in line (when necessary) or in an appendix at the end of the document.

Zero-Truncated Binomial

a. Write the PMF for ZT Binomial:

As described in Thomas and Gart (1971), the PMF for the ZT Binomial distribution is given below:

$$f_X(x) = P(X_T = x) = \binom{n}{x} \frac{(p)^x(1-p)^{n-x}}{(1-(1-p)^n)}$$

b. **Derive the MLE for n=2 trial case**:

Assume there are m iid observations from a ZT Binomial with $n = 2$ trials. We can calculate the MLE in closed form as long as there are fewer than 3 trials.

Note that in this case, the sample space takes on only two discrete values $\mathcal{X} \in [1, 2]$. The Likelihood is given below, for m independent observations:

$$L(p|X_T) = \prod_{i=1}^{m} \binom{2}{x_i} \frac{p^{x_i}(1-p)^{(2-x_i)}}{(1-(1-p)^2)}$$

Taking the log, we can obtain the log-likelihood:

$$log(L(p|X_T)) = \sum_{i=1}^{m} log(\binom{2}{x_i}) + x_i log(p) + (2-x_i)log(1-p) - log(1-(1-p)^2)$$

Next, we take the partial derivative of the log-likelihood with respect to the parameter of interest, p. I also do some simplifying algebra in this step:

$$= \sum_{i=1}^{m} \frac{x_i}{p} - \frac{2-x_i}{1-p} - \frac{2-2p}{2p-p^2}$$

Carring out the sum through each piece, and setting the whole thing equal to zero, we can solve for the closed-form version of the MLE.

$$= \frac{\sum_{i=1}^{m} x_i}{p} - \frac{2m - \sum_{i=1}^{m} x_i}{1-p} - \frac{2m-2pm}{2p-p^2} \stackrel{set}{=} 0$$

From here, we can multiply everything by the common term $p(1-p)(2p-p^2)$ to get rid of the pesky fractions we need to remove. Doing, so, we obtain:

$$0 = \sum_{i=1}^{m} x_i((1-p)(2p-p^2)) - (p(2p-p^2))(2m - \sum_{i=1}^{m} x_i) - (p(1-p))(2m-2pm)$$

Simplifying, and collecting like terms, we have a form organized by powers of p. I'm dropping the indicators on the sum notation for readability here:

$$p^3(2\Sigma x_i + 2m - 2) + p^2(-2\Sigma x_i - 4m + 4) + p(2\Sigma x_i + 2) = 0$$

We can induce a quadratic equation by dividing both sides by common term p:

$$p^2(2\Sigma x_i + 2m - 2) + p(-2\Sigma x_i - 4m + 4) + (2\Sigma x_i + 2) = 0$$

The solution is then obtained via the quadratic equation, with the a, b, and c terms as functions of the observed data, sum, and known constants:

$$\hat{p}_{MLE} = \frac{(2\Sigma x_i + 4m - 4) \pm \sqrt{(-2\Sigma x_i - 4m + 4)^2 - 4 * (-2\Sigma x_i - 4m + 4)(2\Sigma x_i + 2)}}{4 * (\Sigma x_i + m - 1)}$$

It is strange that this would turn into a quadratic problem - I have gone over the math several times and have not been able to determine where I made a mistake. However, my sense of things is that the positive root would be the ML estimator based on the fact that the positive root produces a larger value than the negative root.

The second derivative appears to always be negative, implying that the maximum occurs at the MLE, as noted below. Note that each piece is a negative number, since the x's are constrained to live in the set $\mathcal{X} \in [0, 1]$.

$$\frac{\partial^2}{\partial p} = \frac{-k}{p^2} - \frac{(2-k)}{(1-p)^2} + \frac{2(p-1)^2}{p^2(p-2)} < 0$$

I'm dubious about the above result. It feels like something should have simplified in the above math, or perhaps I am not clever enough to see it. Interestingly, Thomas and Gart(1971) as well as several other statisticians (i.e. Fisher(1934)) seem to avoid use of the MLE in favor of other estimators; however, given the age of most of those papers, it may have been for the computational reasons discussed in the next question.

c. **The MLE does not exist in closed form for greater than 3 trials.Describe a procedure for finding the MLE**:

Given this, I would utilize a numerical approximation method that allows the MLE to be found iteratively. Specifically, I would consider using a Newton-Raphson method.

For such a method, the algorithm would start as follows. An initial guess for $\hat{p}$ would be made. Given that choice, the second step would be to update that estimate by calculating two quantities, H^{-1} and $g_{\hat{p}_i}$. Their definitions are as follows:

$$H^{-1} = \frac{\partial^2(l(p|x))}{\partial p^2}|_{p=\hat{p}_i}$$
$$g_{\hat{p}_i} = \frac{\partial l(p|x)}{\partial p}|_{p=\hat{p}_i}$$

The updating equation, which would calculate the next iteration's estimate of the parameter p, is:

$$\hat{p}_{i+1} = \hat{p}_i - H^{-1}_{\hat{p}_i} g_{\hat{\theta}_i}$$

This process would be repeated until convergence, i.e. when the difference between the i^{th} and $i + 1^{th}$ iterations are within some specified tolerance.

The downside to this method is that in certain cases, the likelihood function may not have an easy-to-find global maximum, or the algorithm may have the potential to get stuck in a local optimum. Therefore, I would start the algorithm with different starting values and check for consistency in my solution. Note that Newton-Raphson is just one of several different numerical methods that could be used to tackle this problem.

d. **Describe a way to construct uncertainty:**

Typically, I would take advantage of the asymptotic properties of MLEs as my primary way to assess the uncertainty around the parameter estimate $\hat{p}$. There are several ways to build confidence intervals in this manner.

For large values of m, the asymptotic distribution of the MLE $\hat{p}$ can be leveraged to generate a confidence interval using the Observed Information:

$$I(p_0) = E|_{p_0}(\frac{\partial^2 logLik(p|X)}{\partial p^2})$$

In theory, we might use this quantity to approximate the standard error of the $\hat{p}$ by $\frac{1}{\sqrt{I(\hat{p})}}$. However, there's an interesting caveat in here that this is derived via asymptotic results. Agresti et al(2002) note that these asymptotics may not kick in until m is relatively large, which is not of much interest to us when comparing small m values. Additionally, given the difficulty in calculating MLE estimates for n>3 (no closed form), calculating Information could be tedious as well in certain cases.

Therefore, I think the better approach is to take a bootstrap approach. The general idea here is that, given some sample, we generate the estimate for the MLE of the truncated binomial. Call the sample here X_m. Now, we can resample from the observed data (with replacement), and we will likely get a (slightly) different sample. We can re-calculate the ML estimate for this new sample and compare it to the one that we got in the first iteration.

By doing this many times (perhaps 100 or 1000 times), we can get a sense for the sampling distribution of our estimate, and we can build a confidence interval by taking quantiles of that distribution. For simple quantification of uncertainty, it may simply suffice to take a standard deviation of that vector of estimated values.

e. **Simulation Study:**

The design of my simulation study is as follows. The data were simulated under a true truncated normal distribution using the R package `rztbinom` from the `actuar` package. Given that the true data are distributed from a truncated binomial, the binomial estimator should not be an appropriate model for the data, but the truncated binomial estimator should provide reasonable results.

After simulating data, the data are used to generate a binomial MLE of the following form:

$$\hat{y}_{bin} = \frac{\sum x_i}{nm}$$

.

Then that estimator is compared to the MLE for the truncated binomial distribution as calculated above. Although I expressed some doubt as to whether or not that estimator is correct, it seems to be generating reasonable results (something between 0 and 1 at least).

I created a shiny app that is given at the following link: https://paulharmon.shinyapps.io/CompSimulationStudy/. It can be accessed on your cell phone QR reader by scanning the QR code below. The app is not particularly fancy, but it does allow a user to assess different combinations of m and p with sliders. Images in the next question are taken directly from the app.

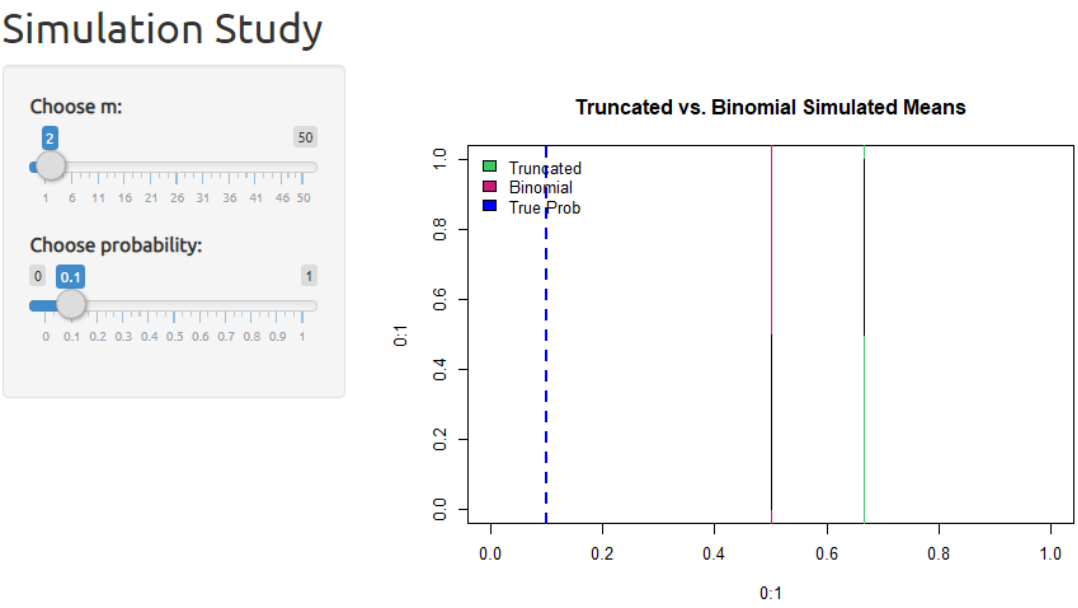

Figure 1: When p and m are both large, .

Since I use a bootstrap method to assess the variability around the Zero-truncated MLE, I do the same with the binomial distribution. Bootstrap CIs are shown instead of histograms because in early versions of the app, they looked better than histograms of bootstrapped values. In both cases, the standard deviation of the sampling distribution was quite small, so I multiplied both values by a constant so that they can be seen in the app. My justification for this is that I am less interested in the actual value of the variance and more interested in the behavior of the variation of each estimator - multiplying by the same constant allows me to make such comparisons.

Code for the shiny application is given at the end of this document.

f. **Summarize the findings from the simulation study**:

The simulation study found that for some combinations of m and p, you could get away with using a binomial estimator without a great deal of difficulty. For others, the binomial model did not provide a reasonable approximation; the Binomial MLE either had a great deal of variation associated with it or it was farther away from the true probability. My recommendation is that the zero-truncated version should be used if possible, if the observations of 0 are not observable.

Consider the case where p and m are both small. In this case, estimates both have relatively small variance, and the binomial actually seems to outperform the zero-truncated binomial. Figure 1 illustrates this effect.

Moreover, when m is small and p is near 0.5, the binomial does substantially worse than the zero-truncated version, and it has much more uncertainty associated with it. Figure 2 shows this effect.

Finally, when both p and m are large, the binomial actually appears to do a better job of approximating the true probability. However, in repeated samples, the variability in the binomial MLE seems to have blown up relative to the Zero-truncated version, as shown in Figure 3. Remember, the way the app illustrates uncertainty, the main takeaway is that the uncertainty around the estimate for the binomial MLE is larger than that of the Z-truncated binomial, so the fact that it extends past 1 is less of a concern.

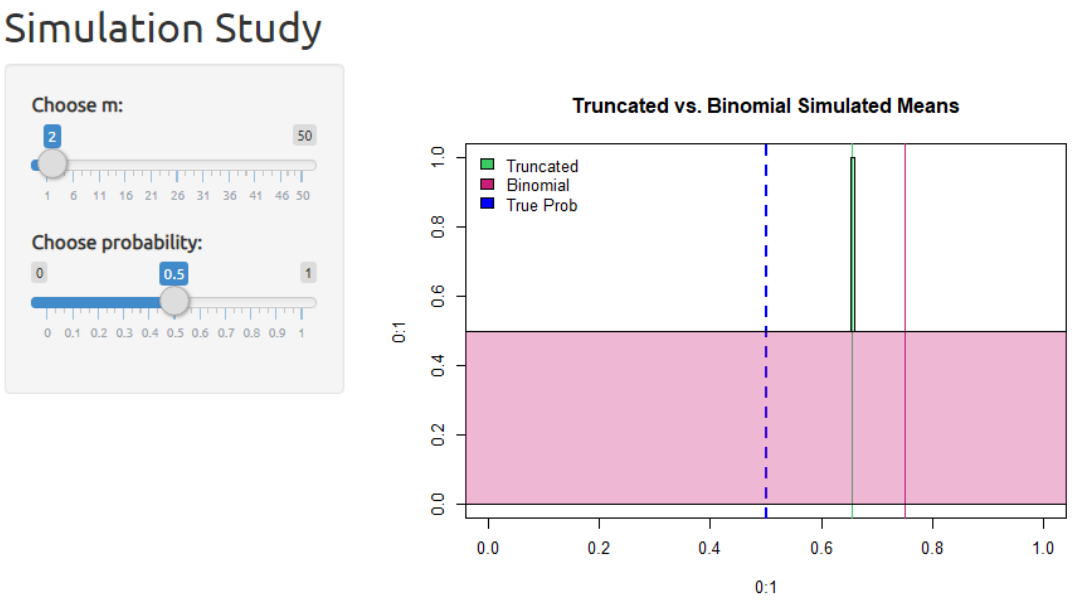

Figure 2: When p and m are both large, .

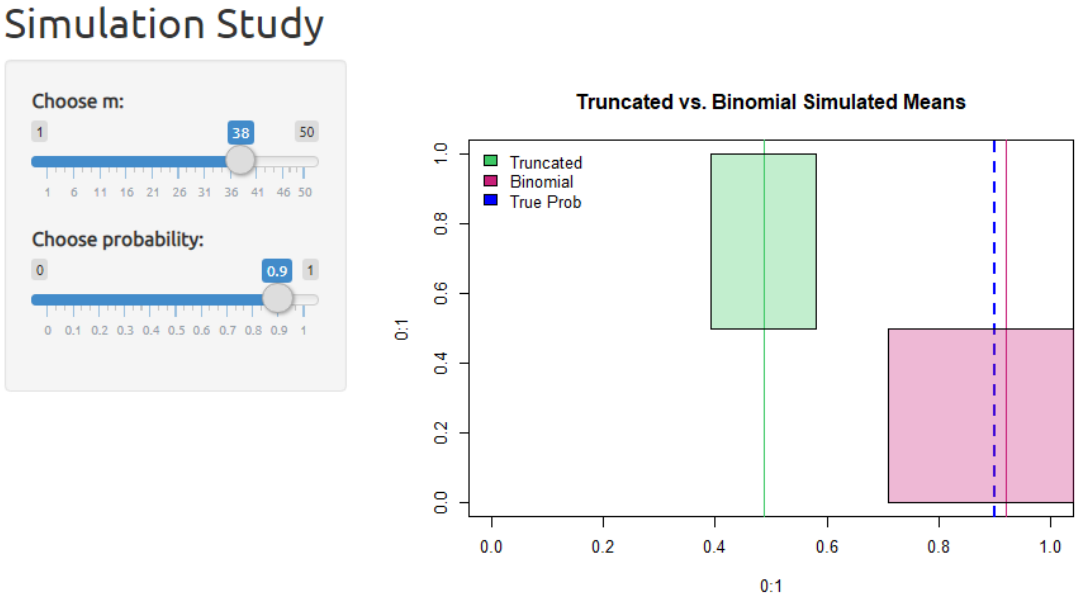

Figure 3: When p and m are both large, .

Bayesian Estimation of Zero-Truncated Binomial

a. **Philosophy for choosing priors**

Using Bayesian methods offer several benefits over traditional frequentist methods. For one, basing inferences off of posterior distributions gives one the ability to talk about credible intervals without having to consider long-run probability, and thinking of modeling the parameters given the observed data has a number of benefits from a philosophical point of view. Moreover, the ability to use a carefully chosen prior distribution to inform a model is something that makes Bayesian statistics a tool that is inherently well-designed in iterative scientific studies: One can use prior knowledge/results to inform current research.

Given this, there are several different methods for choosing prior distributions. Choosing uninformative priors (including Jeffrey's Priors) is beneficial if one seeks to gain advantages associated with posterior-based inference, without adding bias to their model. I think this is a good idea if there is little previous research/theory that you can leverage to justify more informative prior distributions.

Choices of prior distributions that are informative have value in certain cases. It may make sense to put strong shrinkage priors onto factor loadings in a factor analysis to induce sparsity if the problem is high-dimensional. In other cases, if previous models have been fit in different studies and theory is well-established, relatively informative prior distributions make a great deal of sense to use. In this sense, we are able to actually take advantage of the shrinkage effects of a Bayesian model that will pull the posterior parameter distributions away from the (noisy) data and towards the values specified by the prior. In this sense, a well-chosen prior provides regularization to the model that can increase parsimony and reduce overall model variance - this is the main benefit of a reasonable, informative prior.

In short, I believe that the specific problem should drive the choice of prior. While it pains me to say this, computational complexity is typically not a sufficient reason to move away from a prior if there are good, scientific reasons that the more complex model should be used. In some cases, a researcher might be tempted to fit a problem into a structure where it can be solved with easy conjugate priors. In such a case, this is bad practice because the prior should be chosen because it best solves the problem at hand, not because it is easy to use. That being said, if they are appropriate, conjugate priors are certainly not unreasonable in some instances.

b. **State and defend a prior for each p_i:**

An uninformative prior would be to assign the same uniform(0,1) to each of the species. Alternatively, a prior distribution could be included so that we take advantage of what appears to be two pretty distinct subpopulations (i.e. Juniper and Alpine Fir tend to cover at a 50 percent probability, and Sagebrush varieties cover at a much lower rate), as shown in Figure 4.

Unfortunately, I am not an expert on alpine flora, and I do not have the benefit of previous studies to inform the prior knowledge. Without getting into the world of Empirical Bayes, and using the data to inform my prior specification, I think it is smarter to use an uninformative prior here rather than guessing at a specification. Therefore, I chose to use the same prior distribution for each of the species, a $P(p_i) \sim Beta(1,1)$ that can be construed as a $Uniform(0,1)$ distribution.

```
##Code for Bayesian Analysis

## 'JUNCOM' - Common Juniper
## 'ABILAS' - Subalpine Fir
## 'ARTFRI' - Fringed Sagebrush
## 'ARTNOV' - Black Sagebrush
shosh.binary <- shoshveg %>% dplyr::select('JUNCOM', 'ABILAS', 'ARTFRI', 'ARTNOV') %>%
  melt() %>% mutate(present = as.numeric(value > 0), plot = rep(1:150,4)) %>%
  dplyr::select(species = variable, present, plot)
```

```
## No id variables; using all as measure variables
```

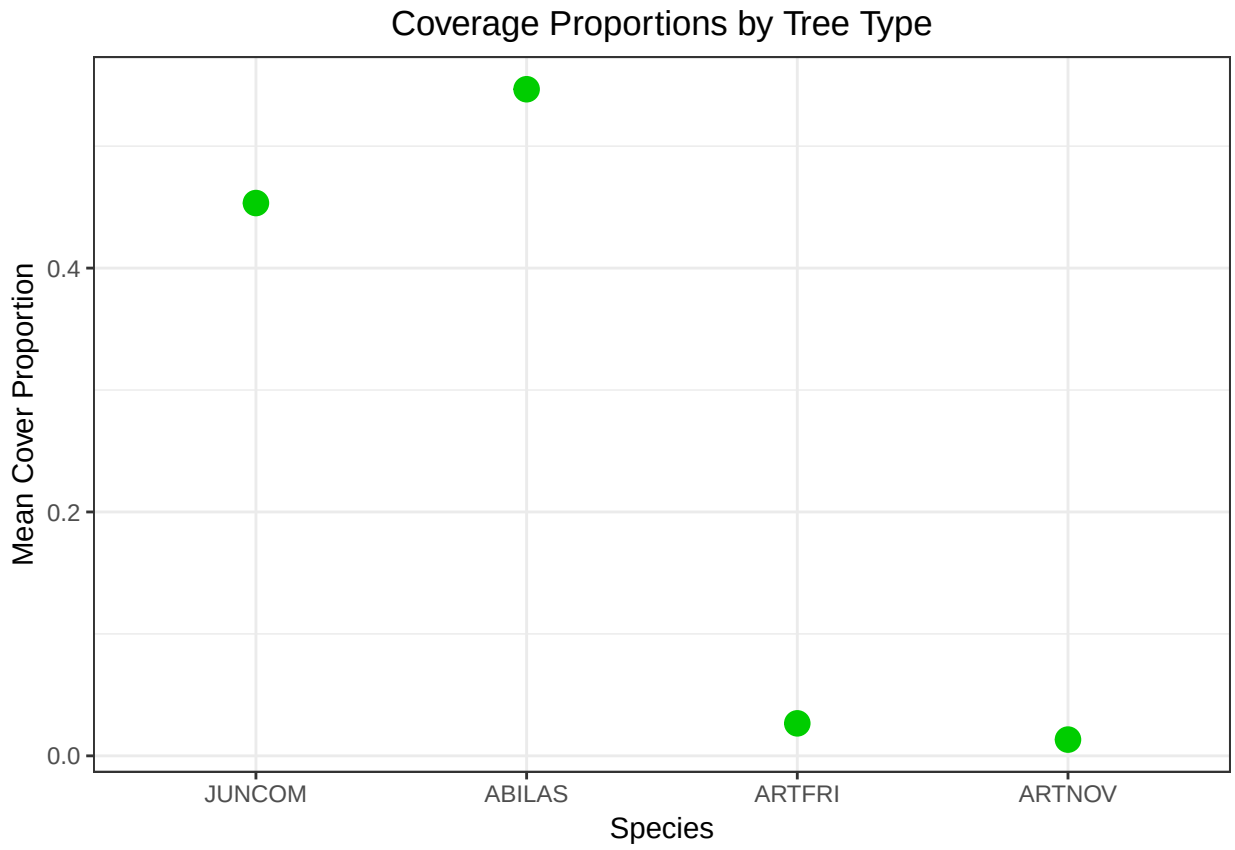

Figure 4: Cover percentage by species.

```
#calculates the mean coverage
shosh.binary %>% group_by(species) %>% summarise(plot = mean(present))%>%
  ggplot(., aes(species, plot)) +
   geom_point(size = 4, color = "green3")  +
   theme_bw() +
   theme(plot.title = element_text(hjust = 0.5)) +
   xlab("Species") + ylab("Mean Cover Proportion")+
     ggtitle("Coverage Proportions by Tree Type")
```

c. **Choose sampling model and compute posterior**:

Given the choice of the non-informative prior, I choose a binomial sampling model. While I realize that this is not a technically appropriate sampling model for the data (since they are zero-truncated), it should reasonably approximate the data in most instances. However, since the sampling model is allowed to produce values of 0, the posterior distribution will allow for 0-values to occur as well.

A binomial sampling model is used under the assumption of exhangeablility, meaning that the order in which the data were sampled does not affect the coverage proportions observed.

With the prior distribution as specified:

$$P(p) \sim Beta(1, 1)$$

The sampling model takes the form:

$$f(y|p) \sim Binomial(n, p)$$

Then, the beta-binomial prior/likelihood combination is conjugate, so the posterior distribution takes the following form, as described by Hoff (2009) *see page 37*:

$$P(p|y) \sim Beta(1 + y, 1 + n - y)$$

The posterior, then, is easy to generate using the `rbinom` function with the proper parameters. The posterior can be summarized in several different ways, including a 95% credible interval, a Maximum a Posteriori (MAP estimate), and, of course, the mean of the posterior. Rather than taking samples directly from the posterior distribution, predictions for new observations can made using a posterior predictive distribution.

```
#generates the posterior distributions for each species
betaBin <- function(alpha = 1, beta = 1, data){
   #alpha = 1
   #beta = 1
   N <- length(data$species)
   y <- sum(data$present)
   set.seed(123)
   #parameters of the posterior
   alpha.post <- y + alpha
   beta.post <- N + beta - y
   posterior_sample <- rbeta(100, alpha.post, beta.post)
 #these are our posterior intervals
   Credible.Int <- qbeta(c(0.025, 0.975), alpha.post, beta.post)

   #posterior mode: MAP estimate
   MAP <- (alpha.post -1)/(alpha.post + beta.post - 2)
return(list(posterior.sample = posterior_sample, alpha = alpha.post, beta = beta.post,
            credible_int = Credible.Int, MAP = MAP))}

#filters the data
jc <- filter(shosh.binary, species %in% 'JUNCOM')
ab <- filter(shosh.binary, species %in% 'ABILAS')
at <- filter(shosh.binary, species %in% 'ARTFRI')
an <- filter(shosh.binary, species %in% 'ARTNOV')

#calculates posterior distributions for each group
JCmod <- betaBin(1,1,data = jc)
ABmod <- betaBin(1,1,data = ab)
ATmod <- betaBin(1,1,data = at)
ANmod <- betaBin(1,1,data = an)
```

d. **Summarize your results in non-technical terms**:

Assuming my cousin is a biology major (without much in the way of statistical background), the results of the model are still quite interesting:

The measured site in Wyoming contains several plants that cover the ground at varying frequencies. In samples gathered by researchers, the pine trees (Common Juniper and Subalpine Fir) covered a greater proportion of the ground, on average, than did the shrub brush species (Fringed Sagebrush and Black Sagebrush).

The beta-binomial model is designed to estimate coverage proportion for each of the four types of vegetation found in the area. The model incorporates two key components, a prior belief about each type of tree's probability of being present in an area, and a sampling model that predicts how many trees might be present in an area. The sampling model used by this model is flawed in that it allows for the prediction of 0 trees in an area; however, the way that the data were collected, plants had to have occurred at least once to be in the dataset. Thus, an improvement on this model could be made by utilizing a zero-truncated sampling model that precludes this possibility.

Nevertheless, the beta-binomial allows us to utilize *a priori* information about each species to inform the model and produce a posterior distribution - the probability distribution that you can think of as an estimate for distribution of cover percentages for a given tree species. This model gives credible intervals for each type of tree. Because I do not know much about tree cover (outside of what this data set tells me), I chose to use a prior distribution that does not add a great deal of information to the model. Unfortunately, that choice leads to results that are somewhat un-exciting, as shown in Figure 5 below. The blue-shaded region indicates the 95% credible interval. We can interpret these intervals as the area that we would expect the true cover proportion to occur with 95% probability. The green dots refer to the original observed proportions and the white rectangles are the MAP estimates - basically, these are our best single point estimates for a Bayesian model of this type.

Since the prior distribution did not add much additional information to the model, the posterior estimates do not differ much from what we observed in the data, as we can see in Figure 5.

```
## produces a similar plot to before with MAP and mean of posterior
#calculates the mean coverage
shosh.binary %>% group_by(species) %>% summarise(plot = mean(present))%>%
  ggplot(., aes(species, plot)) + geom_point(size = 4, color = "green3")  +

   geom_point(aes(species, c(JCmod$MAP, ABmod$MAP,ATmod$MAP,ANmod$MAP)),
              shape = 'square',color = "white",alpha = .9) +
   theme_bw() +   theme(plot.title = element_text(hjust = 0.5)) +
   xlab("Species") + ylab("Mean Cover Proportion")+
   ggtitle("Coverage Proportions by Tree Type") +
   geom_rect(aes(xmin = .95,xmax = 1.05,
         ymin = JCmod$credible_int[1],ymax = JCmod$credible_int[2]),
         fill = "blue", alpha = .05)+
   geom_rect(aes(xmin = 1.95,xmax = 2.05,
         ymin = ABmod$credible_int[1],ymax = ABmod$credible_int[2]),
         fill = "blue", alpha = .05)+
   geom_rect(aes(xmin = 2.95, xmax = 3.05,
         ymin = ATmod$credible_int[1],ymax = ATmod$credible_int[2]),
         fill = "blue", alpha = .05) +
   geom_rect(aes(xmin = 3.95, xmax = 4.05,
            ymin = ANmod$credible_int[1],ymax = ANmod$credible_int[2]),
             fill = "blue", alpha = .05)
```

If we had some additional information that we could have leveraged to build informative priors, we might have been able to build a model that had more radical differences between the estimated proportions and those that we had observed. For example, if we had estimated that the prior mean coverage for all the species types were $p = .84$, we might have something along the following lines. You can see in the Figure 6 where the new MAP estimators are being pulled away from their data values towards 0.84.

```
#adding priors with expected value 5/(6)
JCmod <- betaBin(5,1,data = jc)
ABmod <- betaBin(5,1,data = ab)
ATmod <- betaBin(5,1,data = at)
ANmod <- betaBin(5,1,data = an)

#creates new plot
```

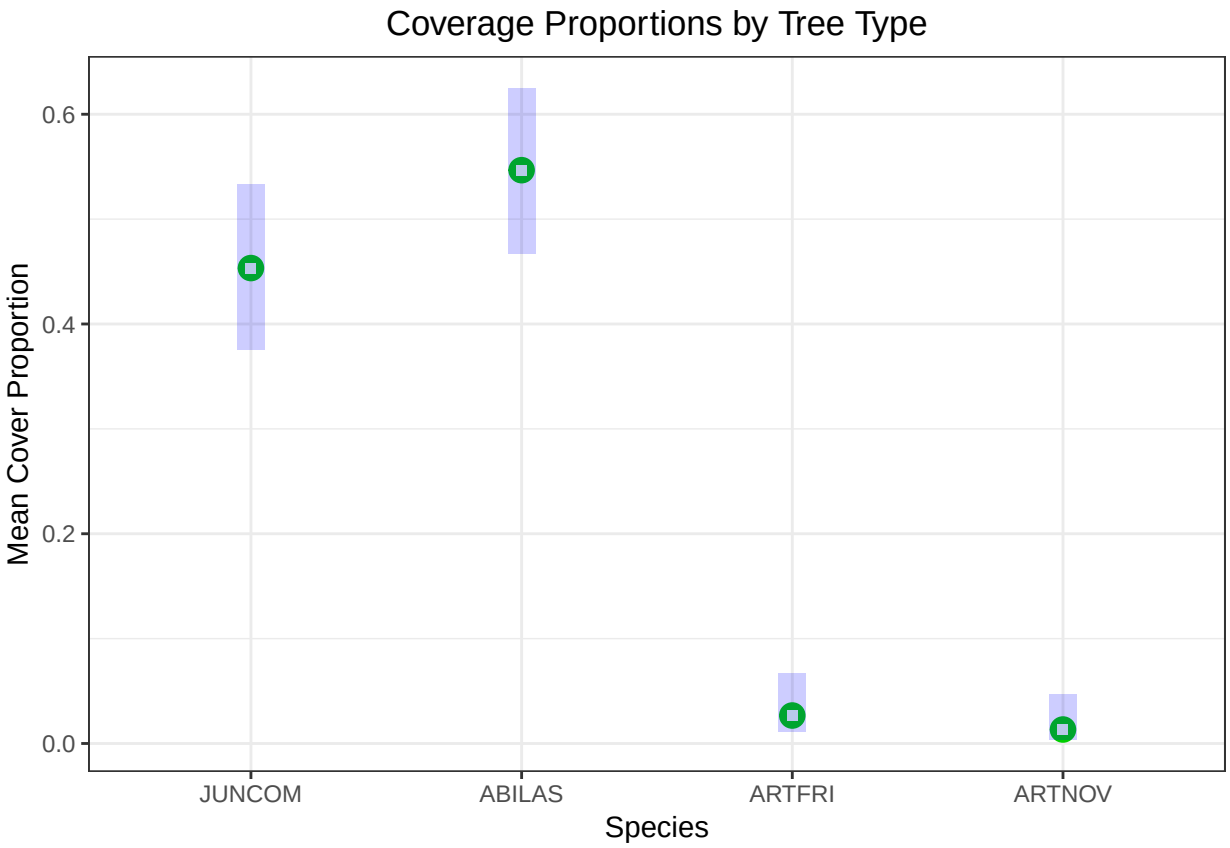

Figure 5: Plot of observed data, credible interval, and the MAP estimator for each species.

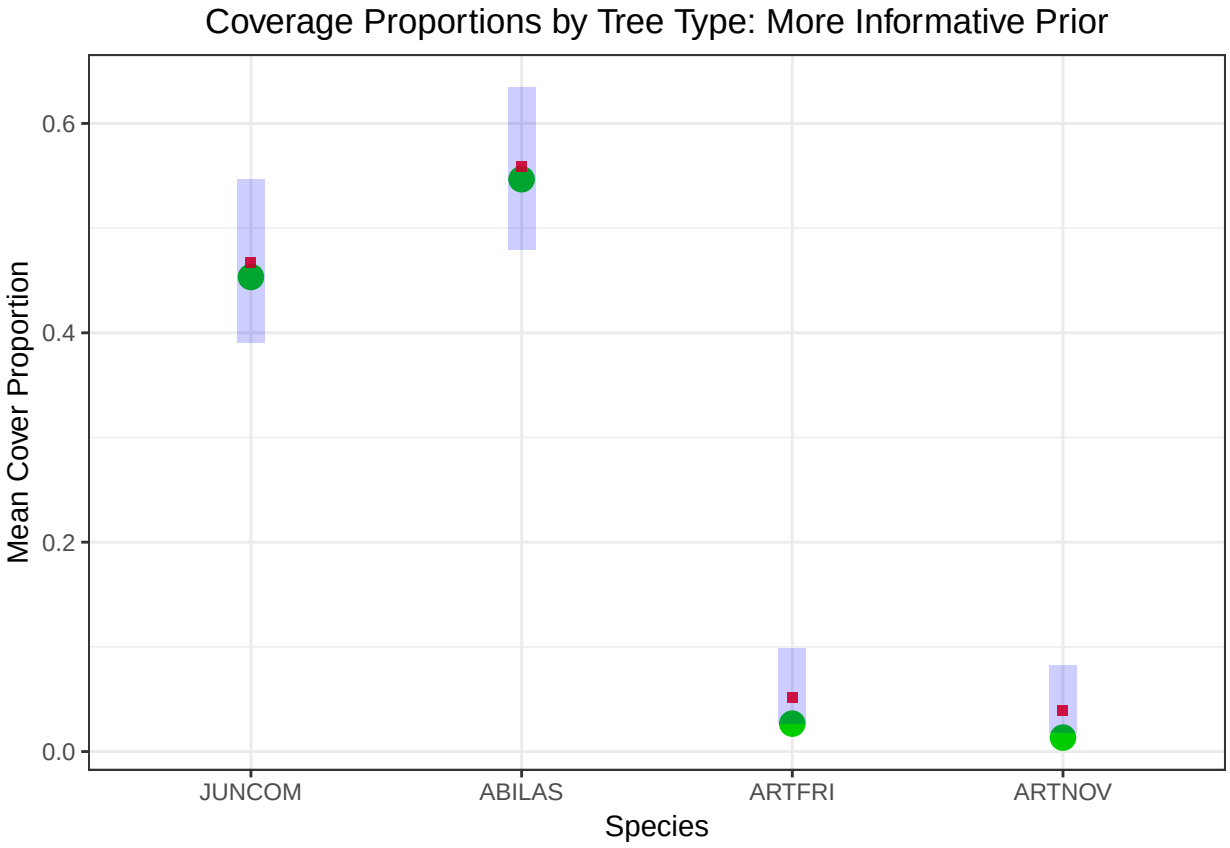

Figure 6: The plot of coverages with more informative priors.

```
shosh.binary %>% group_by(species) %>% summarise(plot = mean(present))%>%
  ggplot(., aes(species, plot)) +  geom_point(size = 4, color = "green3")  +
   geom_point(aes(species, c(JCmod$MAP, ABmod$MAP,ATmod$MAP,ANmod$MAP)),
              shape ='square',color = "red",alpha = .9) + theme_bw() +

  theme(plot.title = element_text(hjust = 0.5)) + xlab("Species") +
   ylab("Mean Cover Proportion")+
   ggtitle("Coverage Proportions by Tree Type: More Informative Prior") +
   geom_rect(aes(xmin = .95,xmax = 1.05,
                 ymin = JCmod$credible_int[1],ymax = JCmod$credible_int[2]),
             fill = "blue", alpha = .05)+
   geom_rect(aes(xmin = 1.95,xmax = 2.05,
                 ymin = ABmod$credible_int[1],ymax = ABmod$credible_int[2]),
             fill = "blue", alpha = .05)+
   geom_rect(aes(xmin = 2.95, xmax = 3.05, ymin = ATmod$credible_int[1],
                 ymax = ATmod$credible_int[2]),fill = "blue", alpha = .05) +
   geom_rect(aes(xmin = 3.95, xmax = 4.05,
                 ymin = ANmod$credible_int[1],ymax = ANmod$credible_int[2]),
             fill = "blue", alpha = .05)
```

e. **If covariate information are available, how would it be implemented?**

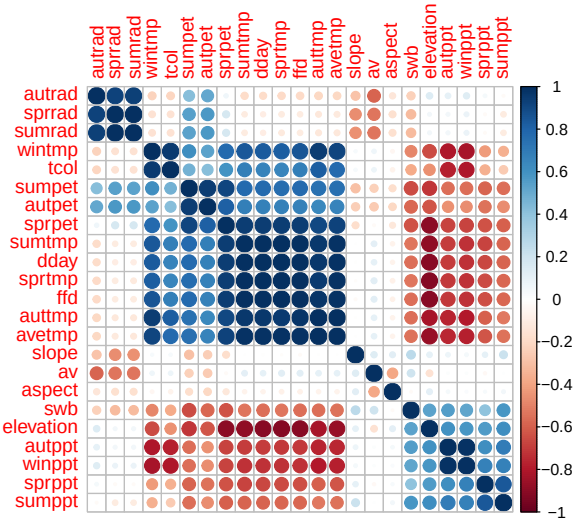

Figure 7: Shared information in many of the covariates means that we might want to use Bayes Factors for model comparison.

Note below that covariate information is available at each site. We have 24 covariates available for each of the 150 sites, including seemingly germane features such as Elevation, Slope, temperature-related variables during the year, and other features relating to the climate of each site.

IF we wanted to take advantage of including these features in the model, our framework would change from estimating a single-parameter model to a more complex multiple regression framework. Consider the classical linear model framework for a binary-regression problem:

$$logit(Y) = \mathbf{X}\beta$$

The logit function is the $logit(Y) = log(\frac{Y}{1-Y})$ and β is a vector of parameters associated with the effect of each covariate on the response.

Using the zero-truncated binomial model, you can pose the problem as a binary regression problem win a probit regression framework. In essence, you would model a latent variable that. The biggest difference is that instead of a mean only model, where the parameter p is the main one of interest, you now have several new parameters for the effects of each covariate β_i and the covariance of those features.

Computationally, this makes the problem more interesting because generating the full conditional distributions of each parameter requires stepping through several distributions which will most likely not have a closed form. Moreover, even if multivariate normal distributions were used as a prior, the posterior would be decidedly non-normal (due to the effect of the link function, either a logit or probit link). This would necessitate the use of a Metropolis-Hastings algorithm to sample the posterior distribution, taking into the account a wide range of possible priors on the model parameters β, Σ, or p.

As a last note, it is likely that many of the features are correlated or at least contain similar information, shown in the correlation plot in Figure 7. Obviously this is a problem in either Bayesian regression models or traditional frequentist regression settings. Careful consideration of features should be taken even before the model is fit (i.e. note that 'winpet' is 0 for all values) and models should be compared using **Bayes Factors** to determine the optimal regression model that takes advantage of additional covariates.

```
suppressMessages(library(corrplot))
data(shoshsite)
corrplot(cor(dplyr::select(shoshsite, - winpet)), order = 'hclust')
```

References

- Agresti, A., Min, Y., Unconditional small-sample confidence intervals for the odds ratio, Biostatistics, Volume 3, Issue 3, 1 September 2002, Pages 379–386, https://doi.org/10.1093/biostatistics/3.3.379
- Fisher, R. A. (1934), THE EFFECT OF METHODS OF ASCERTAINMENT UPON THE ESTIMATION OF FREQUENCIES. Annals of Eugenics, 6: 13-25. doi:10.1111/j.1469-1809.1934.tb02105.
- Hoff, P. A First Course in Bayesian Statistical Methods. Springer, 2009.
- Thomas, D., & Gart, J. (1971). Small Sample Performance of Some Estimators of the Truncated Binomial Distribution. Journal of the American Statistical Association, 66(333), 169-177. doi:10.2307/2284868

R Code: Shiny App

```
library(shiny);library(shinythemes);library(actuar)

# Define UI for application that draws a histogram
ui <- fluidPage(
    #shiny Theme
    theme = shinytheme("united"),
    # Application title
    titlePanel("Simulation Study"),

    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            sliderInput("mval",
                        "Choose m:",
                        min = 1,
                        max = 50,
                        value = 5)
        ,
        sliderInput("pval",
                    "Choose probability:",
                    min = 0,
                    max = 1,
                    value = .5,
                    step = .1)

    ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("distPlot")
        )
    )
)

# Define server logic required to draw a histogram
server <- function(input, output) {
```

```
output$distPlot <- renderPlot({
    #simulate the data using the actuar package's data
    m <- input$mval #number of trials
    p <- input$pval #probablility of success on each trial

    sim.dat <- rztbinom(m, size = 2, prob = p)
    #sim.dat <- rbinom(100, m, p)
    #True Mean
    TM <- m*p/(1-(1-p)^m)

    #bootstrap_function:
    boot_bin <- function(n.boot, data){
        #initializes a matrix to store bootstrapped values
        boot_mean <- rep(0, n.boot)
        for (j in 1:n.boot){
            #samples indices from observed data (a vector)
            bs.index <- sample(1:length(data),n.boot,replace = TRUE)
            bs <- data[bs.index]
            #calculates the estimate we need
            boot_mean[j] <- mean(bs)/(2*m)
        }

        return(boot_mean)}

    #sample mean
    BIN.mle <- mean(sim.dat)/(2)
    #gives us the sum of the x_i divided by the number of reps, i.e. xbar
    #create a bootstrapped estimate of the 95% CI
    BIN.sd.boot <- 1000*sd(boot_bin(500, data = sim.dat))
    BIN.alpha <- rgb(200,30,120,alpha = 80, maxColorValue = 255)
    #mean(sim.dat) gives the same output

    #zero-truncated binomial estimator (still iffy on this one!)
    x <- sim.dat
    ZBIN.mle <- (-(2*sum(x) +4*m - 4) + sqrt((-2*sum(x)-4*m + 4)^2 -
                                              4*(-2*sum(x) -
                                                   4*m + 4)*(2*sum(x) + 2)))/
       (4*(sum(x) + m - 1))

    boot_Zbin <- function(n.boot, data){
        #initializes a matrix to store bootstrapped values
        boot_mean <- rep(0, n.boot)
        for (j in 1:n.boot){
            #samples indices from observed data (a vector)
            bs.index <- sample(1:length(data),n.boot,replace = TRUE)
            x <- data[bs.index]
            #calculates the estimate we need
            boot_mean[j] <-  (-(2*sum(x) +4*m - 4) +
                                  sqrt((-2*sum(x)-4*m + 4)^2 -
                                           4*(-2*sum(x) - 4*m + 4)*(2*sum(x) + 2)))/
               (4*(sum(x) + m - 1))

        }
```

```
            return(boot_mean)}
        ZBIN.var <- 1000*sd(boot_Zbin(500, data = sim.dat)) #var of bootstrapped version
        Z.alpha <- rgb(60,200,100,alpha = 80, maxColorValue = 255)

        plot(0:1, 0:1, type = "n",
             main = "Truncated vs. Binomial Simulated Means",
             xlab = "Probability", ylab = "", yaxt = 'n')
        abline(v = p, col = "blue", lwd = 2, lty = 2)
        abline(v = ZBIN.mle, col = rgb(60,200,100, maxColorValue = 255)) #adds ZT bin
        rect(xleft = ZBIN.mle-ZBIN.var, xright = ZBIN.mle + ZBIN.var,
             ybottom = .5, ytop = 1, col = Z.alpha)
        abline(v = BIN.mle, col = rgb(200,30,120, maxColorValue = 255)) #adds MLE in
        rect(xleft = BIN.mle-BIN.sd.boot, xright = BIN.mle + BIN.sd.boot,
             ybottom = 0, ytop = .5, col = BIN.alpha)
        legend('topleft', fill = c(rgb(60,200,100, maxColorValue = 255),
                                   rgb(200,30,120, maxColorValue = 255),'blue'),
               legend = c('Truncated','Binomial','True Prob'), bty = 'n')

    })
}

# Run the application
shinyApp(ui = ui, server = server)
```