Modeling Options for the Wheat Streak Mosaic Virus Study

K. Flagg, B. Fenton, and D. Anderson

BA Consulting, LLP

Fall 2015

Contents

1 Introduction

Bayesian inference has become widely used in various scientific disciplines in recent decades. However, Bayesian data analysis is rarely utilized in the Consulting Seminar because few clients are familiar with the methodology. We hope to illustrate that, while Bayesian data analysis may be less familiar than its likelihood or frequentist counterparts, it can provide results when the other approaches cannot. Additionally, it is our goal to provide easy-to-understand implementations of Bayesian inference. This report will demonstrate a comparison of likelihoodist and Bayesian analyses.

2 Objectives

Researchers are interested in the effect of the timing of a nitrogen treatment on the rate of Wheat Streak Mosaic Virus infection for several varieties of winter wheat. It is also of interest to see whether the strength of the effect differs between different varieties, or between environments where the virus is present in high levels versus low level. In this report, we will demonstrate two versions of the analysis of variance (ANOVA) to determine which variables and interaction are the most important in explaining variation in infection rate.

3 Design

The experiment was performed in six blocks. Each block contained five rows, with one of five varieties of wheat randomly assigned to each row. Each row was split into six plots, and each plot was assigned a unique combination of nitrogen application time (early, mid, and late) and inoculation status (inoculated or not). Infected plants were transplanted into the inoculated plots so that the inoculated plots were known to have an elevated virus presence. Non-inoculated plots were assumed to have the virus present in a naturally-occurring level.

In total, 180 plots were observed. From each plot, 30 leaves were collected and the infection status was recorded for each. These data were aggregated to the plot level so that the response variable was the number of infected plants out of 30 for each plot.

4 Model and Simulation

The statistical model is

$$\begin{aligned}
y_j &\sim \mathrm{Bin}(30, \pi_j), \text{ for } j = 1, 2, \ldots, 180; \\
\mathrm{logit}(\pi_j) &= \mu + \beta^{(1)}_{b[j]} + \beta^{(2)}_{v[j]} + \beta^{(3)}_{b[j],v[j]} + \beta^{(4)}_{i[j]} + \beta^{(5)}_{n[j]} \\
&\quad + \beta^{(6)}_{b[j],i[j]} + \beta^{(7)}_{b[j],n[j]} + \beta^{(8)}_{i[j],n[j]} + \beta^{(9)}_{i[j],v[j]} + \beta^{(10)}_{n[j],v[j]} \qquad (1) \\
&\quad + \beta^{(11)}_{b[j],i[j],n[j]} + \beta^{(12)}_{b[j],i[j],v[j]} + \beta^{(13)}_{b[j],n[j],v[j]} + \beta^{(14)}_{i[j],n[j],v[j]}; \\
\beta^{(k)}_{(\cdot)} &\sim \mathrm{N}(0, \sigma^2_k), \text{ for } k = 1, 2, \ldots, 14;
\end{aligned}$$

where $b = 1, 2, \ldots, 6$ indexes the blocks; $v = 1, 2, \ldots, 5$ indexes the varieties; $i = 1$ indicates non-inoculated status and $i = 2$ indicates inoculated status; $n = 1, 2, 3$ indicate early, mid, and late nitrogen application, respectively.

In the above model, each $\beta^{(k)}_{(\cdot)}$ is a member of a batch of coefficients sharing a common Normal distribution with a scale parameter σ_k. The analysis of variance is less concerned with the values of particular coefficients than with the contribution of each batch to the total variation of the data; the blocks, varieties, and nitrogen application times can be thought of as random samples from populations.

To understand the model, it is helpful to think about the process that would generate the data, and then simulate data from that process. There is some overall infection rate for the population of all winter wheat plants. Plants in inoculated plots will have higher infection rates than plants in plots that were not inoculated. Characteristics like variety, timing of the nitrogen application, and environmental conditions (for which block is a proxy) cause variation about the overall mean. If a certain characteristic, such as the variety, has a large influence on the infection rate, then the coefficients in batches associated with variety will have a large amount of variation and a high standard deviation.

Infection is an uncommon event, so an overall infection rate of 5% could be expected. For the simulation, we set $\mu = -3$ because $\text{logit}(0.05) \approx -3$. Inoculated plots will have higher infection rates than non-inoculated plots, so $\beta^{(4)}_2 = 0.5$ and $\beta^{(4)}_1 = -0.5$ were used to give baseline infection rates of about 8% and 3% in inoculated and non-inoculated plots, respectively.

For each batch, values of the coefficients were generated via random draws from normal distributions centered at zero. Standard deviations were chosen for each effect to reflect our beliefs about the importance of that effect on the infection rate. Variety, nitrogen timing, and interactions involving inoculation should have high variation, so the values $\sigma_2 = 0.3$, $\sigma_5 = 0.4$, $\sigma_8 = 0.03$, $\sigma_9 = 0.03$, and $\sigma_{14} = 0.001$ were used. Other batches were given relatively small standard deviations.

We have provided code so that all details and steps of the process are available and reproducible. It is possible and fairly rudimentary to adjust the parameters to try to adjust the simulated data to ones liking. The overall infected leaf counts can be seen in Figure 1. A more detailed visualization of these counts at the plot level is provided in Figure 2.

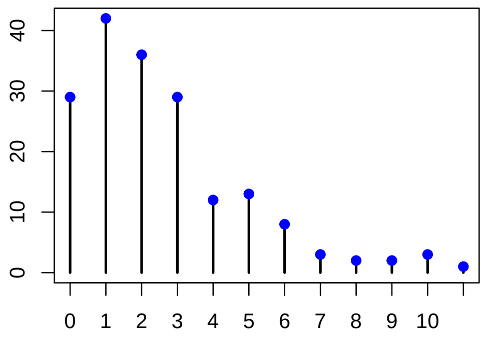

Figure 1: The number of the 180 simulated plots that had 0 leaves, 1 leaf, ..., 11 leaves infected out of the 30 collected from each plot.

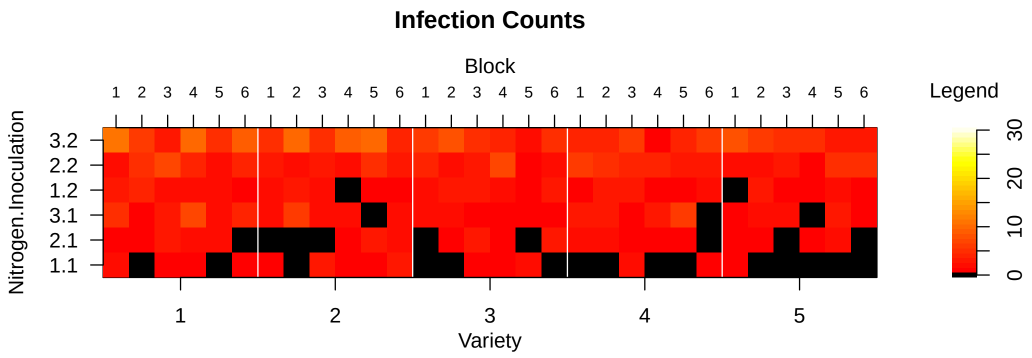

Figure 2: Heatmap of the number of infected leaves out of 30 from each plot. The data have been arranged by variety, block, inoculation status, and nitrogen application time so that patterns can be seen easily.

5 Generalized Linear Mixed Model

The most common way to analyze data from a split-plot experiment is with a linear mixed model. In this situation, the response variable is a Binomial count, so a generalized linear mixed model is used. This approach uses nested batches of coefficients to model the structure of the experimental design. These coefficients are considered to be varying members of a population (commonly called *random effects*), and so a scale parameter is estimated for each batch. The parameters are estimated by maximizing the likelihood function. The `glmer` function from the R package `lme4` will be used.

5.1 Background

The likelihood function provides a relative measure of how well a possible value of a parameter θ fits with the observed data y, and so it can be used to identify the value of θ that is then used as the estimate most consistent with y. The likelihood function takes its form from the probability model for y, denoted $p(y|\theta)$. However, likelihoods are discussed after the data have been collected so we consider y to be fixed and we examine how $p(y|\theta)$ varies as a function of θ.

Since $p(y|\theta)$ is the probability of observing y for a given θ, a θ where $p(y|\theta)$ is large is more consistent with the obeserved data than a θ value that makes $p(y|\theta)$ small. Then a reasonable approach to estimating θ is to find the value of θ that maximizes $p(y|\theta)$ for the observed y.

Note that the likelihood function depends on the particular data that were observed. The data y' could very well have been collected instead of y, and then the likelihood function would be $p(y'|\theta)$, which might be maximized by a different θ value than the θ value the maximizes $p(y|\theta)$. If we consider how this maximum likelihood estimator (MLE) of θ will vary between different samples, we can construct a probability distribution. For large samples, MLEs are unbiased and follow Normal distributions that are well-behaved and have small standard errors. Additionally, the use of MLEs is philosophically supported by the Likelihood Principle, which states that all of the information in a sample relevant to estimating the model parameters is contained in the likelihood function.

5.2 Fitting the Model in GLMER

Model (1) will be adapted to be fit with `glmer`. Since specific levels of nitrogen and inoculation were pre-assigned to plots, they are treated as constant (typically known as *fixed effects*). Standard deviations will not be estimated for the blocks associated with these variables. On the other hand, block and variety levels can be considered as members of a larger population. As a result they will be treated as varying and their standard deviations will be estimated. Our R code to run `glmer` appears in the appendix.

5.3 Results

Using the GLMM approach with standard statisitical software yields results that can be confusing and extremely difficult to interpret. Tables 1 and 2 show the most relevant parts of the the default output from the `anova` and `summary` functions. It is inappropriate to compute an F-statistic for the Binomial setting as the model does not include a residual error term to provide a denominator. The output does not provide p-values as they would be incorrect. Instead, we would use simulation to generate p-values (an approach that mimics the Bayesian method). The reported standard deviations of zero for the varying terms are not believable. Two batches have larger standard deviations; a plot of the varying coefficients within those two batches (Figure 3) shows that all confidence intervals include zero. This suggests that there is very little variability, but we would not conclude that all of the coefficients are in fact 0.

The output does provide the information that nitrogen application time, inoculation, the variety by nitrogen timing interaction, and the block by variety by nitrogen timing interaction contribute to the variation in the observed number of infected leaves; however, the presentation is convoluted.

	Df	Sum Sq	Mean Sq	F value
nit	2	55.28	27.64	27.64
ino	1	95.56	95.56	95.56
nit:ino	2	0.42	0.21	0.21

Table 1: ANOVA table for the constant terms.

Batch	Std. Dev.
blk	0.0000
vty:ino	0.0000
blk:ino	0.0000
vty:nit	0.1293
blk:nit	0.0000
vty:blk	0.0000
vty:nit:ino	0.0000
blk:nit:ino	0.0000
blk:vty:ino	0.0000
blk:vty:nit	0.1913

Table 2: Estimated standard deviations of the varying terms.

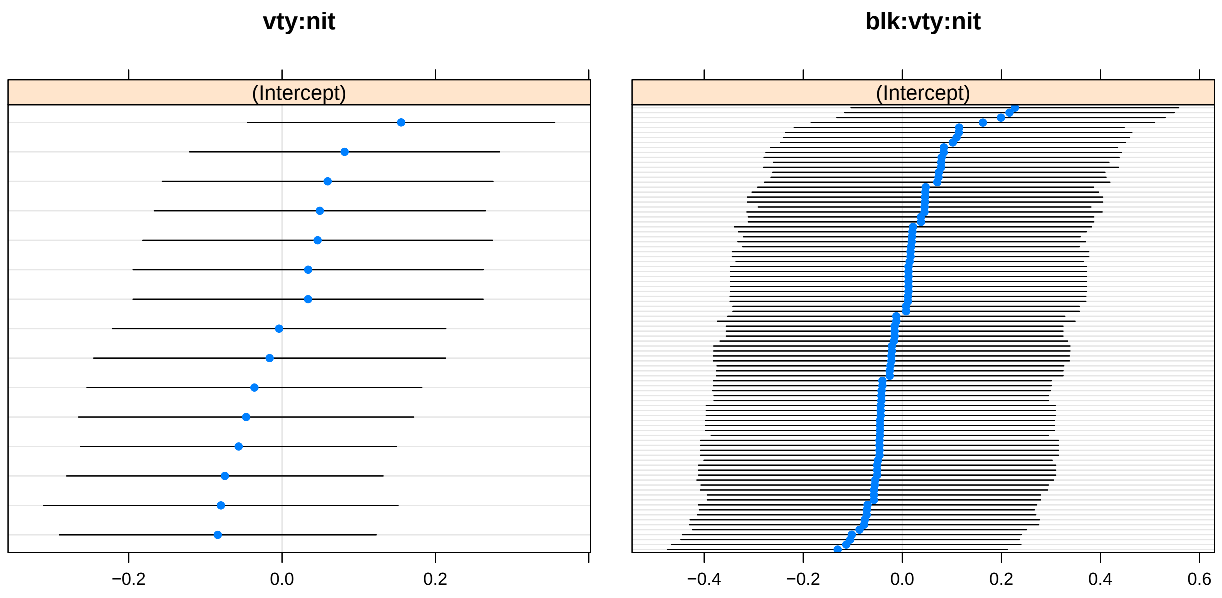

Figure 3: 95% confidence intervals for the individual coefficients in the two batches with large standard deviations.

5.4 Additional Issues with Maximum Likelihood Estimation

While the strict maximum likelihood approach can be powerful and elegant in its simplicity, it has some major shortcomings. It is possible for two different study designs with different models to have the same likelihood function, so the likelihood function discards the information about how the data were collected. Also, even if the sample is sufficiently large and was collected in such a way that it should be representative, it is still possible to end up with a sample that is not representative of the population. Either of these flaws could be tempered by including information about prior beliefs or domain-specific knowledge relevant to the model, but the likelihoodist approach does not provide a way to do so. In some cases this may not matter, but in many others the relevant information which is left unused by the likelihood approach can greatly improve both the ability to make inferences and the quality of those inferences.

6 Bayesian Analysis

A Bayesian model incorporates a prior distribution that describes existing knowledge or beliefs about the parameters. The prior distribution contributes additional information not contained in the likelihood. This information is then conditioned upon the likelihood, and the knowledge about the parameters is updated into a posterior distribution.

6.1 Background

Probability theory provides the basic property of conditional probability known as Bayes' rule,

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)} = \frac{p(\theta)p(y|\theta)}{p(y)}$$

where $p(y) = \sum_\theta p(\theta)p(y|\theta)$ is summed over all possible values of θ. A Bayesian analysis starts from Bayes' rule. It involves finding a posterior distribution for θ, denoted $p(\theta|y)$, from the likelihood $p(y|\theta)$ and a prior distribution of θ, $p(\theta)$. The posterior is found using the proportionality relationship,

$$p(\theta|y) \propto p(\theta) \times p(y|\theta).$$

Often, analysts assume that all values of θ are approximately equally probable, and so the prior distribution is constant: $p(\theta) = c$. Then the posterior distribution is proportional to the likelihood,

$$p(\theta|y) \propto c \times p(y|\theta) \propto p(y|\theta).$$

In cases like this, the value of θ that is most probable in posterior is the same as the maximum likelihood estimate, so it is common for Bayesian inference to appear similar to likelihood-based inference. It is important to remember, however, that a likelihood is not a probabilty statement about θ. A Bayesian posterior distribution is used to make probability statements about θ, and a likelihood-based *estimator* has a probability distribution, but the likelihood itself can only be used to compare possible θ values in the context of one set of observed data.

6.2 Advantages to Bayesian Data Analysis

There are many advantages to performing a Bayesian analysis. One such advantage is the ability to provide meaningful results even when there are few available observations with which to carry out inferences about a given parameter. Another is a relative robustness to unusual samples; with carefully-chosen priors a Bayesian analysis can succeed where a likelihoodist approach would utterly fail, especially in situations where the data produce an MLE value at the edge of the parameter space. For example, if this experiment had been performed on only one plot and no infected leaves were observed, the MLE would lead to the conclusion that on average no leaves would be infected upon replication. This result would run counter to both common sense and the scientific consensus motivating the experiment in the first place.

Furthermore, a Bayesian analysis will produce a distribution with which inferences can be made about a given parameter. A likelihoodist approach applied to the same data would only furnish point and variance estimates. With the availability of posterior distributions comes a more complete picture regarding the parameter itself. This additional information provides the ability to perform the same calculations that could be performed with other distributions, including computing complicated functions of the parameters that would provide additional insight into the research question(s). Bayesian posterior intervals also have an interpretation which usually aligns more closely with intuition than the interpretation of classical confidence intervals, since because Bayesian posterior intervals are direct statements of probability.

Finally, it can be very difficult in some instances to fit a generalized linear mixed model in R. While an analogous Bayesian model might require more lines of code to set up, doing so can often be a more logical and straightforward process. And once the model has been set up, there is again a much larger set of ways in which it can be used.

6.3 Priors

As described in Section 6.1, the use of Bayesian analysis requires a prior distribution to be chosen for each parameter. There may be little known about the distribution of each parameter, but any small amount of preexisting knowledge can guide the choice of prior. Regardless, it is desirable for the data to have as large an impact on the outcome of the analysis as is reasonable; a well-chosen prior should serve to permit unusual outcomes and prohibit impossible outcomes.

A natural choice for many who implement a Bayesian analysis is to use what is called a noninformative prior. A noninformative prior is "noninformative" in the sense that ideally it should not contain any subjective information about the parameter. If there is subjective information available about the parameter, whether it takes the form of a belief or prior knowledge, an analysis can be improved by including this information in the form of an informative prior. A compromise between these two extremes is to utilize a weakly informative prior (the approach taken in this report's model), which will include enough subjective information to constrain posterior distributions to what is plausible.

6.4 The Bayesian Model

The model here uses the details from Model (1), with a few additional pieces. In order to implement a Bayesian fit of this model, it is necessary to define priors for the model's parameters. Each batch of coefficients was treated as varying and given a Normal distribution. These Normal distributions were centered at 0, since adjustments to the overall mean were being modeled. The standard deviations of these Normal distributions were assigned Half Cauchy distributions per [3], with the same scale parameter for each. The value chosen for this scale parameter was 3 since it is apparent in (4) that this value best constrains all of the parameters (in particular σ_5 or `sigma_n`) to values which are considered reasonable in the context of this experiment. A scale parameter value of 3 keeps the SD values for the Normal priors close to 3 or less, and a difference of 3 translates to a difference of 18 on the original scale of the data. Scale parameters 2 and especially 1 start to excessively limit the variability of well-behaved parameters like σ_5 (`sigma_n`) and σ_{14} (`sigma_vni`).

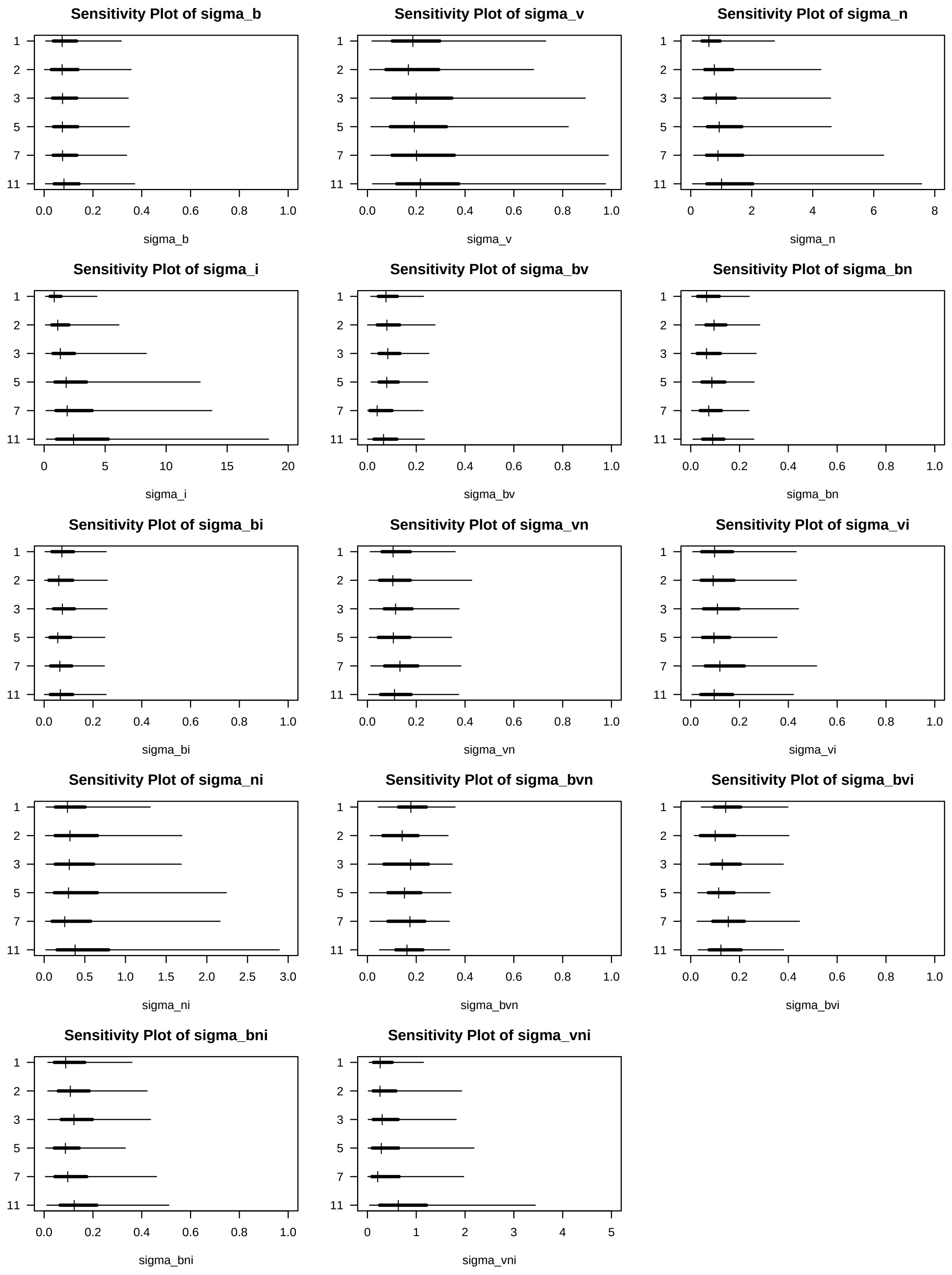

Figure 4: 95% and 50% posterior intervals resulting from prior distributions with various scale parameters.

6.5 STAN

It is possible to fit a Bayesian model by writing the code in a programming language such as R, and for some applications the level of detail and customizeability afforded by this tack is necessary. However, going this route can be a large undertaking, both in terms of time spent coding and computing resources required.

Software packages designed to fit Bayesian models can streamline the process of specifying a model and obtaining draws from its posterior distribution. These draws are typically obtained via some form of the Markov Chain Monte Carlo (MCMC) method. The MCMC method takes draws from a specified targed distribution via one or more Markov Chains. The precise details of this process are beyond the scope of this report (for more details see [4]). It should suffice to say that given enough iterations the MCMC method will converge to its target distribution with mathematical certainty. MCMC methods are utilized because they are much more computationally efficient than naive sampling methods.

For this analysis, the software package STAN was used (our implementation appears in the appendix). STAN implements the Hamiltonian Monte Carlo (HMC) method, which often has a large performance advantage over other MCMC variants. STAN also translates its models to C++ code, which it then compiles – the results of this run much faster than would, for example, an implementation of the same code in R. There are other popular software packages such as BUGS, JAGS and BayesGlm which may be utilized depending on the problem at hand.

7 Posterior Inferences

Bayesian inferences are made by summarizing the posterior distribution. This can be done with probabilities, intervals, expectations, or any other summaries that are helpful for answering the research questions. With a large number of coefficients, perhaps the most concise way to make inference is to summarize the contribution of each batch to the overall variation of the data. With a Bayesian posterior disribution, it is easy to describe this directly in terms of the variances or standard deviations of parameters, rather than the F-ratios of the traditional ANOVA table.

7.1 Bayesian ANOVA

The posterior distributions of scale parameters can be used to construct an analysis of variance display that is much more straightforward to understand than the ANOVA table that usually accompanies a likelihoodist or frequentist analysis of variance. Figure 5 presents a table of posterior summaries as described by Gelman [1]. The batches are grouped by level of nesting, and 50% and 95% posterior intervals for the standard deviations are displayed.

Comparisons can be made at a glance. There was little overall variation among varieties, but Inoculation and nitrogen application time are the largest contributors to the total variation, with respective posterior median standard deviations of 1.34 and 0.84. A large amount of uncertainty is associated with the coefficients, which is manifested in the wide 95% posterior interval reaching beyond 8. This is not surprising, since there are only two inoculation coefficients providing information about the variation in that batch.

There was little variability among varieties (posterior median SD of 0.20), but variety contributed more variation than block (posterior median SD of 0.08).

The inoculation by nitrogen and the inoculation by nitrogen by variety batches, with posterior median standard deviations of 0.31 and 0.30, are the most interesting interaction batches. There is evidence that the effects of nitrogen application time are different between inoculated and non-inoculated plots, and that the size of the differences changes from variety to variety.

The other interaction terms showed much less variation; however, the intervals indicate a good chance that the amount of variation is not 0.

The motivation for summarizing the standard deviations of the coefficients is that these values give direct measurements of how much variation the associated predictors describe on the logit scale. The standard deviations presented here are the scale parameters for the distributions of the effects. These describe the population of all possible blocks, varieties, nitrogen application times, etc, and so they are called the *superpopulation* standard deviations. If the only levels of these variables that are of interest are the specific ones used in this study, it would be more meaningful to examine the posterior distribution of the *finite-population* standard deviations. These can be found by computing the sample standard deviation of each batch in each posterior draw of coefficients.

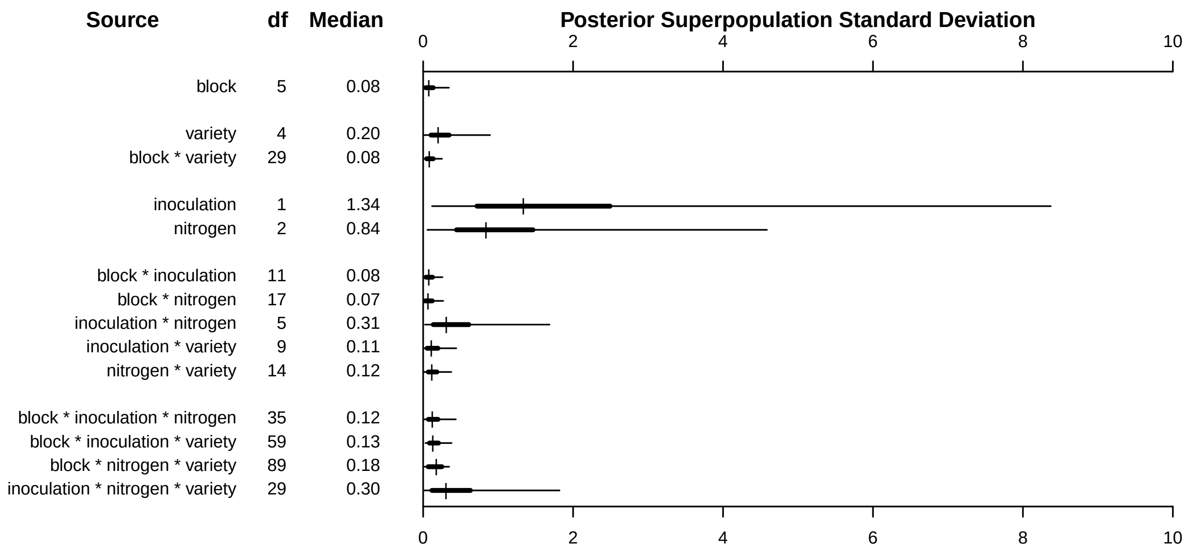

Figure 5: Visual presentation of an analysis of variance for the Bayesian model. The line segments in the right panel show 50% and 95% posterior intervals for the standard deviation of each batch of coefficients.

7.2 MCMC Convergence

Draws from Markov chain Monte Carlo depend on the previous draws, so in order for inferences to be valid we need to know that the chains are sampling from a common distribution. A useful approach to assessing convergence to the posterior distribution is by examining sample path plots of the draws, commonly known as "traceplots." In Figure 6, it is apparent in the first 250 draws that the chains were in different areas of the parameters space and had not settled into a common distribution. It was necessary to run the chains longer until agreement between the chains occurred.

This was achieved by the final 250 draws. Although the chains occaisionaly jumped to outlying areas of the parameter space, they returned to the common path. There are other convergence diagnostics available as well; see [4] for more details.

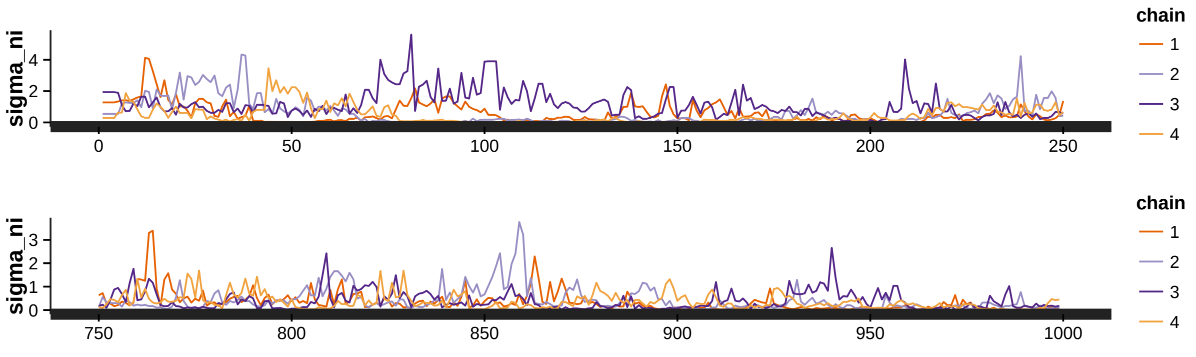

Figure 6: Top: Traceplot of the first 250 draws from each chain. Bottom: Traceplot of the last 250 draws from each chain.

8 Posterior Predictive Check

One major advantage to taking a Bayesian approach to data analysis is the inherent flexibility of Bayesian methods. Once draws have been obtained from a posterior distribution, those draws can be used to predict new data sets which will hopefully be representative of the population from which the original data were drawn. If this is true, then the original data should not seem extraordinary in relation to the posterior predictions. By extension, inferences made based on the predicted values should be close to those made based on the original data. More specifically, checking inferences about aspects of the data which were not explicitly modeled can indicate whether or not the model performs well. Such checks, known as posterior predictive checks, should ideally compare measurements which are related to the researchers' objectives.

There are two particular features of the original data which are important to capture in the modeling process: the relatively large number of zeros and the maximum infected count. A model incorporating either of these features separately would be relatively straightforward to fit, but capturing both simultaneously takes more consideration. Since the extreme values of the data are of primary concern here, the mean is not a useful metric. In order to ascertain whether or not the model did in fact capture these aspects of the data, posterior predictive checks for the number of predicted zeros (Figure 7, left) and predicted maximum (Figure 7, center) were performed. The relationship between predicted counts and the original counts can be visualized using a scatterplot (Figure 7, right).

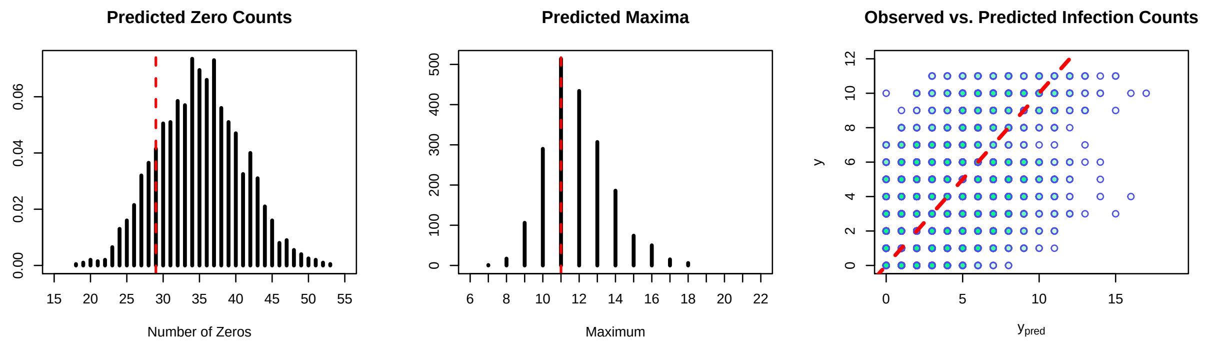

Figure 7: Posterior predictive checks to compare the model's ability to capture zero counts (left) and maxima (center). The posterior predictions of all 180 counts for 100 simulations vs. the original counts are shown on the right.

In Figure 7 (left), the vertical dashed line represents the number of zeros in the original data. The observed zero count is plausible under the distribution of posterior predictive zero counts, suggesting that the model has adequately captured this characteristic of the data. Likewise, in figure 4 (center) it is apparent that the model was able to reproduce the maximum value of infected leaf counts in the data. The scatterplot in Figure 7 (right) shows a tendency for this model to slightly overpredict infected leaf counts, but examining the shading of the dots shows that an overwhelming majority of posterior predictive counts resemble the original data.

Posterior predictive checks need not only be used to make graphical comparisons. Meaningful aspects of the data can be compared to values predicted by the model using numerical summaries. In this study, plots with infection counts greater than 3 are of practical interest since it would represent an infection rate of more than 10%. In the posterior predictions generated, the probability of an infected leaf count greater than 3 for a plot is 0.273, which is fairly close to the observed proportion of 0.244. A calculation of absolute error produces an average infected leaf count difference of 1.51 with a 95% posterior predictive interval of $(1.33, 1.72)$.

9 Model Discussion

The `glmer` fit required one line of code, while the STAN fit was much more involved. This reality would seem to favor `glmer`, but only if the fact that more information as well as more meaningful information was produced by the STAN fit is ignored. This study is attempting to quantify sources of variation contributing to infected leaf counts, and the `glmer` estimates for many of these were 0. While many of the 95% posterior intervals produced by the STAN fit contain 0, the differing interpretation of Bayesian intervals means that there is a good chance that these values are not equal to 0.

Another point in favor of the STAN fit is that we could easily add an arbitrary amount of complexity to the model. `glmer` simply does not offer as many options, and utilizing the highest level of complexity `glmer` can handle involves specifying the model in terms of increasingly convoluted syntax. This is an inherent consequence of the nature of each: STAN is a full-blown modeling language, while `glmer` is an expanded version of the S models used by R's `lm` function.

One complication that can occur when working with counts is separation. This is when one variable or a combination of factors is perfectly associated with the response. For example, if a certain treatment results in an infection count of zero, then knowing that a leaf received that treatment determines that the leaf will not get infected, regardless of other variables. This causes difficulty in estimating the effect of that treatment on the infection rate, and, in a maximum likelihood estimation, results in a large standard error because the effect cannot be precisely determined.

Figure 8 provides a visual illustration of the separation problem. The left panel shows a small probability on the vertical axis and its corresponding logit value on the horizontal axis. The right panel zooms in on the lower left corner. The dotted lines show how a small amount of uncertainty on the probability scale translates to a relatively larger amount of uncertainty on the logit scale.

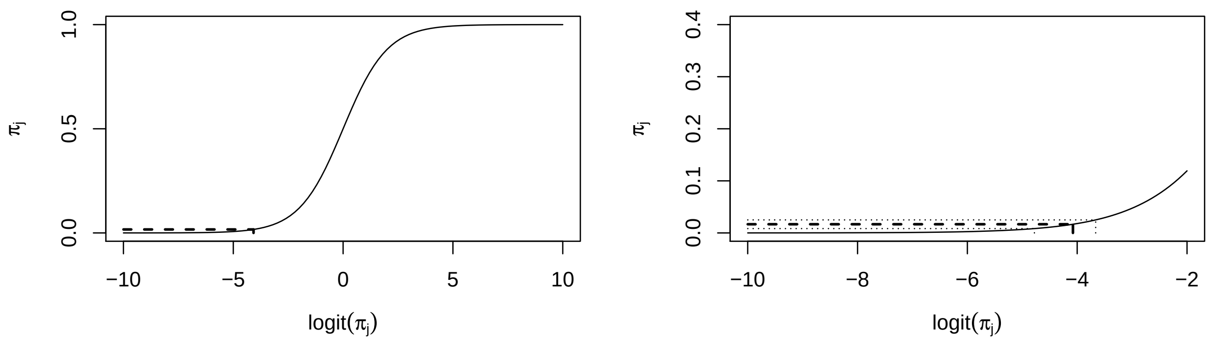

Figure 8: Illustration of separation.

The problem gets worse if the infected count is 0. In that case, the best estimate of the infection probability is 0, and $\text{logit}(0) = -\infty$. This will cause computational problems because the software will try to improve the model fit by decreasing the effect estimate toward $-\infty$. The software may fail to find a solution, and if it does report a solution, the standard error will be large because the solution cannot be found precisely.

Proper Bayesian modeling does a better job of handling separation than likelihoodist modeling, since the priors will restrict parameters to reasonable values. Whether or not this actually occurs hinges on the selection of suitable priors, so again careful thought must go into decisions about which priors to use.

10 Conclusion

This report has demonstrated the analysis of variance in a maximum likelihood setting and in a Bayesian setting. Hopefully the advantages a Bayesian model has over its maximum likelihood counterpart are clear: the Bayesian model can include information about parameters beyond that provided by the data and is easy to interpret after it has been fit. We also hope the implementation is comprehensible enough that it can be used as a starting point for further exploration of Bayesian Statistics.

11 Appendix: R and Stan Code

11.1 Data Simulation

```r
## These functions will be used thoughout.
logit <- function(x){return(log(x/(1-x)))}
expit <- function(x){return(exp(x)/(1+exp(x)))}

## Here, we set up a data frame to store our data: factors of each
##   variable for each plot.
design <- data.frame('block' = as.numeric(gl(6, 30)),
                     'row' = as.numeric(gl(30, 6)),
                     'plot' = 1:180,
                     'variety' = numeric(180),
                     'inoc' = numeric(180),
                     'nitrogen' = numeric(180))
set.seed(6823)
## Each column is a permutation of variety indices
permute.v <- matrix(replicate(6, sample(5, 5, replace = FALSE)), ncol = 6)
for(i in 1:180){
  design$variety[i] <- permute.v[(design$row[i]-1)%%5+1, design$block[i]]
}
ni <- matrix(c(1, 1, 1, 2, 2, 2, 1:3, 1:3), ncol = 2)
## Each column is a permutation of inoc:nitrogen indices
permute.ni <- matrix(replicate(30, sample(6, 6, replace = FALSE)), ncol = 30)
for(i in 1:30){
  design[((i-1)*6+1):(i*6), c('inoc', 'nitrogen')] <- ni[permute.ni[,i],]
}


#### Start the process of simulating data.
## For the binomial setting we have
##    y ~ Bin(30,pi) where
##    logit(pi) = Xb and
##    X is the model matrix and b is a vector of coefficients


# Determine a value of mu to be used for the intercept (first row of b vector).
#   We found that logit(0.05) is approximately -3.
mu <- -3

# It is good practice to set seed when simulating data as it will be
#   reproducible.
set.seed(93)

## We can set values for the main coefficients or use random draws.
##   These are also rows of the b vector.
# There are six blocks.
b.eff <- rnorm(6,0,0.01)
# There are five varieties.
v.eff <- rnorm(5,0,0.3)
# There are three nitrogen levels.
n.eff <- sort(rnorm(3,0,0.4))
# There are two inoculation levels.
i.eff <- c(-0.5,0.5)


## We can do the same for all the two-way interactions.
##   These are more rows of the b vector.
# There are 30 combinations of block and variety.
bv.eff <- rnorm(30,0,0.00001)
# There are 18 combinations of block and nitrogen.
bn.eff <- rnorm(18,0,0.00001)
# There are 12 combinations of block and inoculation.
bi.eff <- rnorm(12,0,0.00001)
# There are 15 combinations of variety and nitrogen.
vn.eff <- rnorm(15,0,0.01)
# There are 10 combinations of variety and inoculation.
vi.eff <- rnorm(10,0,0.03)
# There are 6 combinations of nitrogen and inoculation.
ni.eff <- rnorm(6,0,0.03)

## We can also establish effects for three-way interactions.
##   These are also in the b vector.
# There are 90 combinations of block, variety, and nitrogen.
bvn.eff <- rnorm(90,0,0.00001)
# There are 60 combinations of block, variety, and inoculation.
bvi.eff <- rnorm(60,0,0.00001)
# There are 36 combinations of block, nitrogen, and inoculation.
bni.eff <- rnorm(36,0,0.00001)
# There are 30 combinations of variety, nitrogen, and inoculation.
vni.eff <- rnorm(30,0,0.001)

## It was not of interest to include a four-way interaction.


#### We can now update our "design" matrix with the levels of each
####   interaction.
# head(design) # reminder of the layout of design
## Find the levels for each two-way interaction assigned to a plot.
bv.int <- as.numeric(interaction(design[,1],design[,4]))
bn.int <- as.numeric(interaction(design[,1],design[,6]))
bi.int <- as.numeric(interaction(design[,1],design[,5]))
vn.int <- as.numeric(interaction(design[,4],design[,6]))
vi.int <- as.numeric(interaction(design[,4],design[,5]))
ni.int <- as.numeric(interaction(design[,6],design[,5]))
## Find the levels for each three-way interaction assigned to a plot.
bvn.int <- as.numeric(interaction(design[,1],design[,4],design[,6]))
bvi.int <- as.numeric(interaction(design[,4],design[,6]))
bni.int <- as.numeric(interaction(design[,4],design[,5]))
vni.int <- as.numeric(interaction(design[,6],design[,5]))


## Once the levels have been established for each interaction (of
##   each plot) we can include them as columns in our data set.
##   We have opted to call the data set 'od' for original data.
od <- data.frame(
  "blk" = factor(design[,1]),
  "vty" = factor(design[,4]),
  "nit" = factor(design[,6]),
  "ino" = factor(design[,5]),
  "bv.int" = factor(bv.int),
  "bn.int" = factor(bn.int),
  "bi.int" = factor(bi.int),
  "vn.int" = factor(vn.int),
  "vi.int" = factor(vi.int),
```

```r
  "ni.int" = factor(ni.int),
  "bvn.int" = factor(bvn.int),
  "bvi.int" = factor(bvi.int),
  "bni.int" = factor(bni.int),
  "vni.int" = factor(vni.int)
)


## Using the levels of each factor for a plot, we can find the probabilities
##   associated with the simulated data.
# We can start on the logit scale: logit(pi) = Xb.
logit.pi.vec <- mu + b.eff[od[,1]] + v.eff[od[,2]] + n.eff[od[,3]] +
  i.eff[od[,4]] + bv.eff[od[,5]] + bn.eff[od[,6]] + bi.eff[od[,7]] +
  vn.eff[od[,8]] + vi.eff[od[,9]] + ni.eff[od[,10]] + bvn.eff[od[,11]] +
  bvi.eff[od[,12]] + bni.eff[od[,13]] + vni.eff[od[,14]]

# Now, transform those back to the pi (probability) scale.
pi.vec <- apply(cbind(logit.pi.vec), 1, expit) # pi = expit(X * beta)

# We can use the probabilities to generate y's: y_i ~ Bin(30,pi_i). These will
#   be the counts of infected leaves for a given plot.
set.seed(324)
y.vec <- apply(cbind(pi.vec), 1, function(p) rbinom(1,30,p))


## Lastly, we can attach the counts of infected leaves and uninfected leaves
##   for each plot to our data set.
od$"infected" <- y.vec
od$"uninfected" <- 30 - y.vec

## If desired, it is possible to write the data set to a csv or table for
##   future use.
# write.csv(od,"od.csv",row.names = FALSE)
```

11.2 Running GLMER

```r
## Fit the model using glmer.
glm.mod <- glmer(formula = cbind(infected, uninfected) ~ (nit + ino)^2 +
                   (1|blk/vty) + (1|blk:nit) + (1|blk:ino) + (1|vty:nit) + (1|vty:ino) +
                   (1|vty:nit:ino) + (1|blk:nit:ino) + (1|blk:vty:nit) + (1|blk:vty:ino),
                 data = od, family = binomial(link = "logit"),
                 control = glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2000)))
```

11.3 STAN Model Code

```stan
data{
  // Number of observations, number of levels of each factor, number of
  // Combinations of factors
  int num_obs;
  int m; // Number of Binomial trials
  real<lower=0> cauchysd; // Scale for cauchy priors
  int num_b;
  int num_v;
  int num_n;
  int num_i;
  int num_bv;
  int num_bn;
  int num_bi;
  int num_vn;
  int num_vi;
  int num_ni;
  int num_bvn;
  int num_bvi;
  int num_bni;
  int num_vni;

  // Send in data (actually levels)
  int y[num_obs];
  int blk[num_obs];
  int vty[num_obs];
  int nit[num_obs];
  int ino[num_obs];
  int bv_int[num_obs];
  int bn_int[num_obs];
  int bi_int[num_obs];
  int vn_int[num_obs];
  int vi_int[num_obs];
  int ni_int[num_obs];
  int bvn_int[num_obs];
  int bvi_int[num_obs];
  int bni_int[num_obs];
  int vni_int[num_obs];
}

parameters{
  real mu; // Overall mean
  vector[num_b] b;
  vector[num_v] v;
  vector[num_bv] b_v;
  vector[num_i] i;
  vector[num_n] n;
  vector[num_bi] b_i;
  vector[num_bn] b_n;
  vector[num_ni] n_i;
  vector[num_vi] v_i;
  vector[num_vn] v_n;
  vector[num_bni] b_n_i;
  vector[num_bvi] b_v_i;
  vector[num_bvn] b_v_n;
  vector[num_vni] v_n_i;

  // Standard deviations of the batches
  real <lower=0> sigma_b;    // Batch 1
  real <lower=0> sigma_v;    // Batch 2
  real <lower=0> sigma_bv;   // Batch 3
  real <lower=0> sigma_i;    // Batch 4
  real <lower=0> sigma_n;    // Batch 5
  real <lower=0> sigma_bi;   // Batch 6
  real <lower=0> sigma_bn;   // Batch 7
  real <lower=0> sigma_ni;   // Batch 8
  real <lower=0> sigma_vi;   // Batch 9
  real <lower=0> sigma_vn;   // Batch 10
  real <lower=0> sigma_bni;  // Batch 11
  real <lower=0> sigma_bvi;  // Batch 12
  real <lower=0> sigma_bvn;  // Batch 13
  real <lower=0> sigma_vni;  // Batch 14
}

transformed parameters{
  vector[num_obs] l_p;
  vector[num_obs] p;

  // logit(p) = Xb
  for (o in 1:num_obs) {
    l_p[o] <- mu + b[blk[o]] + v[vty[o]] + n[nit[o]] + i[ino[o]] +
    b_v[bv_int[o]] + b_n[bn_int[o]] + b_i[bi_int[o]] + v_n[vn_int[o]] +
    v_i[vi_int[o]] + n_i[ni_int[o]] + b_v_n[bvn_int[o]] +
    b_v_i[bvi_int[o]] + b_n_i[bni_int[o]] + v_n_i[vni_int[o]];
    p[o] <- inv_logit(l_p[o]);
  }
}

model{
  // likelihood
  y ~ binomial(m,p);

  // Priors on effects
  mu ~ normal(0, 10);           //batch of 1
  b ~ normal(0, sigma_b);       //batch of 6
  v ~ normal(0, sigma_v);       //batch of 5
  n ~ normal(0, sigma_n);       //batch of 3
  i ~ normal(0, sigma_i);       //batch of 2
  b_v ~ normal(0, sigma_bv);    //batch of 30
  b_n ~ normal(0, sigma_bn);    //batch of 18
  b_i ~ normal(0, sigma_bi);    //batch of 12
  v_n ~ normal(0, sigma_vn);    // batch of 15
  v_i ~ normal(0, sigma_vi);    //batch of 10
  n_i ~ normal(0, sigma_ni);    //batch of 6
  b_v_n ~ normal(0, sigma_bvn); //batch of 90
  b_v_i ~ normal(0, sigma_bvi); //batch of 60
  b_n_i ~ normal(0, sigma_bni); //batch of 36
  v_n_i ~ normal(0, sigma_vni); // batch of 30

  // Priors on standard deviations
  sigma_b ~ cauchy(0, cauchysd);
  sigma_v ~ cauchy(0, cauchysd);
  sigma_n ~ cauchy(0, cauchysd);
  sigma_i ~ cauchy(0, cauchysd);
  sigma_bv ~ cauchy(0, cauchysd);
  sigma_bn ~ cauchy(0, cauchysd);
  sigma_bi ~ cauchy(0, cauchysd);
  sigma_vn ~ cauchy(0, cauchysd);
  sigma_vi ~ cauchy(0, cauchysd);
  sigma_ni ~ cauchy(0, cauchysd);
  sigma_bvn ~ cauchy(0, cauchysd);
  sigma_bvi ~ cauchy(0, cauchysd);
  sigma_bni ~ cauchy(0, cauchysd);
  sigma_vni ~ cauchy(0, cauchysd);
}
```

11.4 Running STAN

```
# List of data to send to Stan from our data (od).
# The names on the left are the variable names used in the Stan model.
# The names on the right are the values being sent.
od.stan.data <- list(num_obs = 180
                     m = 30,
                     cauchysd = od$infected,
                     num_b = 6,
                     num_v = 5,
                     num_n = 3,
                     num_i = 2,
                     num_bv = 6*5,
                     num_bn = 6*3,
                     num_bi = 6*2,
                     num_vn = 5*3,
                     num_vi = 5*2,
                     num_ni = 3*2,
                     num_bvn = 6*5*3,
                     num_bvi = 6*5*2,
                     num_bni = 6*3*2,
                     num_vni = 5*3*2,
                     y = od$infected,
                     blk = as.numeric(od$blk),
                     vty = as.numeric(od$vty),
                     nit = as.numeric(od$nit),
                     ino = as.numeric(od$ino),
                     bv_int = as.numeric(od$bv.int),
                     bn_int = as.numeric(d$bn.int),
                     bi_int = as.numeric(od$bi.int),
                     vn_int = as.numeric(od$vn.int),
                     vi_int = as.numeric(od$vi.int),
                     ni_int = as.numeric(od$ni.int),
                     bvn_int = as.numeric(od$bvn.int),
                     bvi_int = as.numeric(od$bvi.int),
                     bni_int = as.numeric(od$bni.int),
                     vni_int = as.numeric(od$vni.int))

# Compile the Stan model
od.cauchy  <- stan_model(file = 'cauchy.stan', model_name = 'cauchyprior')

# Run the sampler with 4 Markov chains for 1000 iterations each
od.stan <- sampling(od.cauchy, chains = 4, iter = 1000,
                    data = od.stan.data, sample_file = 'trial')
```

11.5 Bayesian ANOVA Display

```
# Gelman ANOVA plot for superpop sds
batches <- c('block',
             NA,
             'variety',
             'block * variety',
             NA,
             'inoculation',
             'nitrogen',
             NA,
             'block * inoculation',
```

```r
              'block * nitrogen',
              'inoculation * nitrogen',
              'inoculation * variety',
              'nitrogen * variety',
              NA,
              'block * inoculation * nitrogen',
              'block * inoculation * variety',
              'block * nitrogen * variety',
              'inoculation * nitrogen * variety')
superpop.sds <- c("sigma_b", NA,
                  "sigma_v",
                  "sigma_bv", NA,
                  "sigma_i", "sigma_n", NA,
                  "sigma_bi", "sigma_bn", "sigma_ni",
                  "sigma_vi", "sigma_vn", NA,
                  "sigma_bni", "sigma_bvi",
                  "sigma_bvn", "sigma_vni")
dfs <- c(n.b, NA, n.v, n.bv, NA, n.i, n.n, NA, n.bi, n.bn, n.ni,
          n.vi, n.vn, NA, n.bni, n.bvi, n.bvn, n.vni) - 1

anova.superpop <- function(stan.fit, sources, sds, dfs, xlim = c(0, 1),
                           inner = 0.5, outer = 0.95,
                           width = c(7, 1, 2, 20), digits = 2){
# Creates a caterpillar plot comparing posterior intervals for each of
# several parameters for several values of one hyperparameter.
#  stan.fit  A stan_fit object
#  sources   Character vector of batches being analyzed
#  sds       Character vector of the scale parameters being analyzed
#  dfs       Numeric vector of degrees of freedom for each batch
#  inner     Confidence lever for outer interval
#  outer     Confidence level for outer interval
#  width     Numeric vector of widths to send to layout()
#  digits    Number of decimal places to print medians to
  med <- numeric(length(sources))
  meds <- character(length(sources))
  for(j in 1:length(sources)){
    if(!is.na(sources[j])){
      current <- unlist(extract(stan.fit, pars = sds[j]))
      med[j] <- median(current)
      meds[j] <- formatC(med[j], digits = digits, flag = '#', format = 'f')

    }
    else{
      med[j] <- NA
    }
  }
  layout(matrix(1:4, nrow = 1), width = width)
  # Sources
  par(mar = c(4.1, 1.1, 5.1, 1.1))
  plot(NULL, xlim = c(0, 1), ylim = c(length(sources), 1), main = 'Source',
       xlab = '', ylab = '', xaxt = 'n', yaxt = 'n', frame.plot = FALSE)
  text(x = 1, y = 1:length(sources), labels = sources, pos = 2, offset = 0)
  # dfs
  par(mar = c(4.1, 0, 5.1, 1.1))
  plot(NULL, xlim = c(0, 1), ylim = c(length(sources), 1), main = 'df',
       xlab = '', ylab = '', xaxt = 'n', yaxt = 'n', frame.plot = FALSE)
  text(x = 1, y = 1:length(sources), labels = dfs, pos = 2, offset = 0)
  # Medians
  par(mar = c(4.1, 0, 5.1, 0.5))
  plot(NULL, xlim = c(0, 1), ylim = c(length(sources), 1), main = 'Median',
       xlab = '', ylab = '', xaxt = 'n', yaxt = 'n', frame.plot = FALSE)
  text(x = 1, y = 1:length(sources), pos = 2, offset = 0,
       labels = meds)
  # SDs
  par(mar = c(4.1, 0, 5.1, 1.1))
  plot(NULL, xlim = xlim, ylim = c(length(sds), 1), frame.plot = FALSE,
       main = 'Posterior Superpopulation Standard Deviation',
       xlab = '', ylab = '', yaxt = 'n')
  axis(3)
  for(j in 1:length(sources)){
    if(!is.na(sources[j])){
      current <- unlist(extract(stan.fit, pars = sds[j]))
      segments(x0 = quantile(current, c((1-outer)/2, (1-inner)/2)),
               x1 = quantile(current, 1-c((1-outer)/2, (1-inner)/2)),
               y0 = j, lwd = c(1, 3))
    }
  }
  points(x = med, y = 1:length(sources), pch = '|')
  abline(v = 0)
}

anova.superpop(od.stan, batches, superpop.sds, dfs, xlim = c(0, 10))
```

12 References

[1] Gelman, A. (2005). Analysis of variance–why it is more important than ever. *The Annals of Statistics*, 33(1), 1-53.

[2] Gelman, A., and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*, Cambridge University Press.

[3] Gelman et. al. (2008). A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models, *The Annals of Applied Statistics*, 2(4), 1360-1383.

[4] Gelman et. al. (2013). *Bayesian Data Analysis*, 3rd ed. CRC Press.

[5] Ramsey, F. and Schafer, D. (2012). *The Statistical Sleuth: A Course in Methods of Data Analysis*, 3rd ed. Cengage Learning.