Week 4 assignment: Poisson models

Will Stutz

February 5, 2016

Wasps in the family Cynipidae lay their eggs on plants which form galls around the developing larvae, providing nutrition until the larvae metamorphose and burrow out of the galls, emerging as adults. From any particular gall, there is variation in the number of host wasps that emerge.

Here, you will construct a Bayesian model for the number of emerging cynipid wasps, using features of the galls as explanatory variables. The data are available in the `cleaned_galls.csv` file. Your task is to estimate the parameters of your model, and then to do a posterior predictive check to evaluate overdispersion.

Problem 1: looking at the data

Load the data and explore how the features relate to the response variable.

```
# libraries
library(ggplot2)
library(tidyr)
library(rstan)
```

```
## rstan (Version 2.9.0, packaged: 2016-01-05 16:17:47 UTC, GitRev: 05c3d0058b6a)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

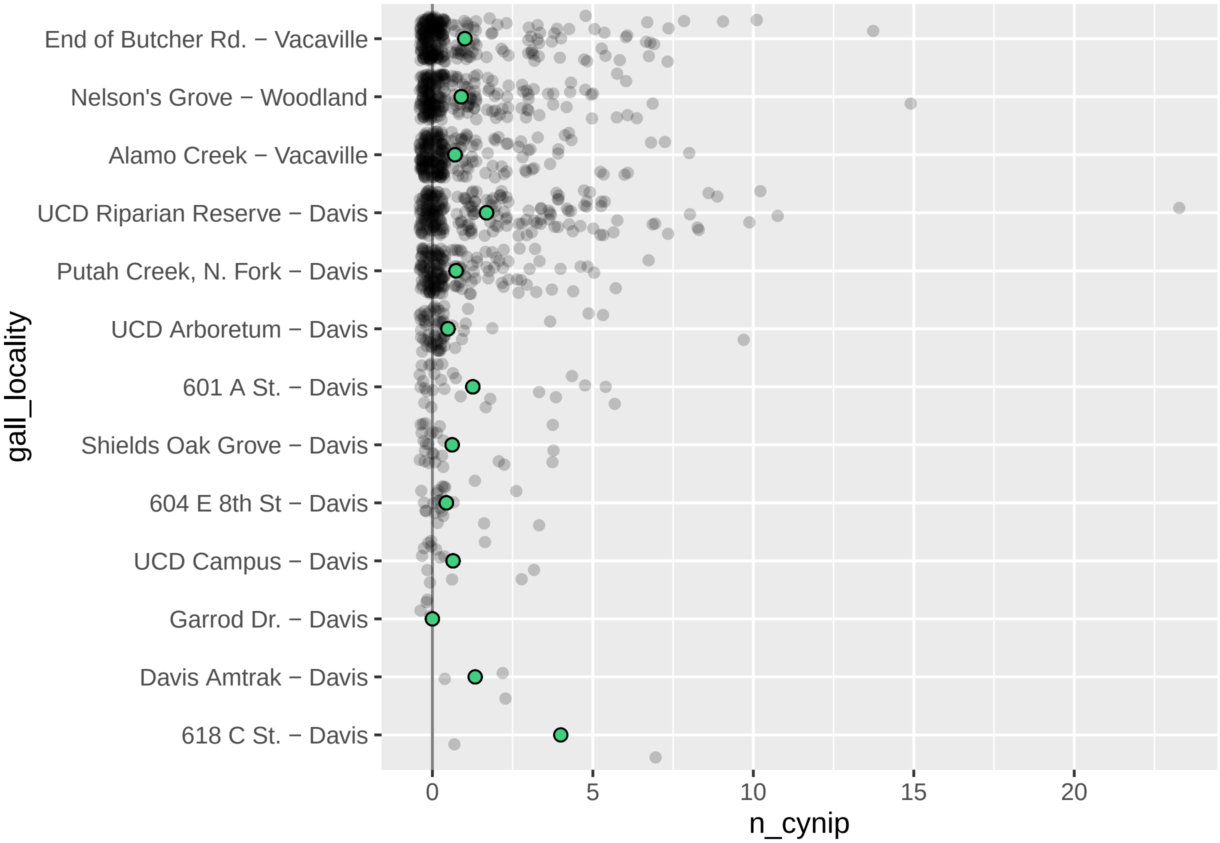
```
# load the data
dat <- read.csv("cleaned_galls.csv")

# what kind of data do we have
head(dat)
```

```
##   gall_ID gall_size                gall_locality n_cynip
## 1      10        20           UCD Campus - Davis       0
## 2     100        40          604 E 8th St - Davis       0
## 3    1000         5 Putah Creek, N. Fork - Davis       0
## 4    1002        25 Putah Creek, N. Fork - Davis       0
## 5    1005        10 Putah Creek, N. Fork - Davis       0
## 6    1006        15 Putah Creek, N. Fork - Davis       0
```

```
  # looks like n_cynip, gall size and locality
```

Let’s take a look at how the data are distributed across sites (i.e. sample sizes)

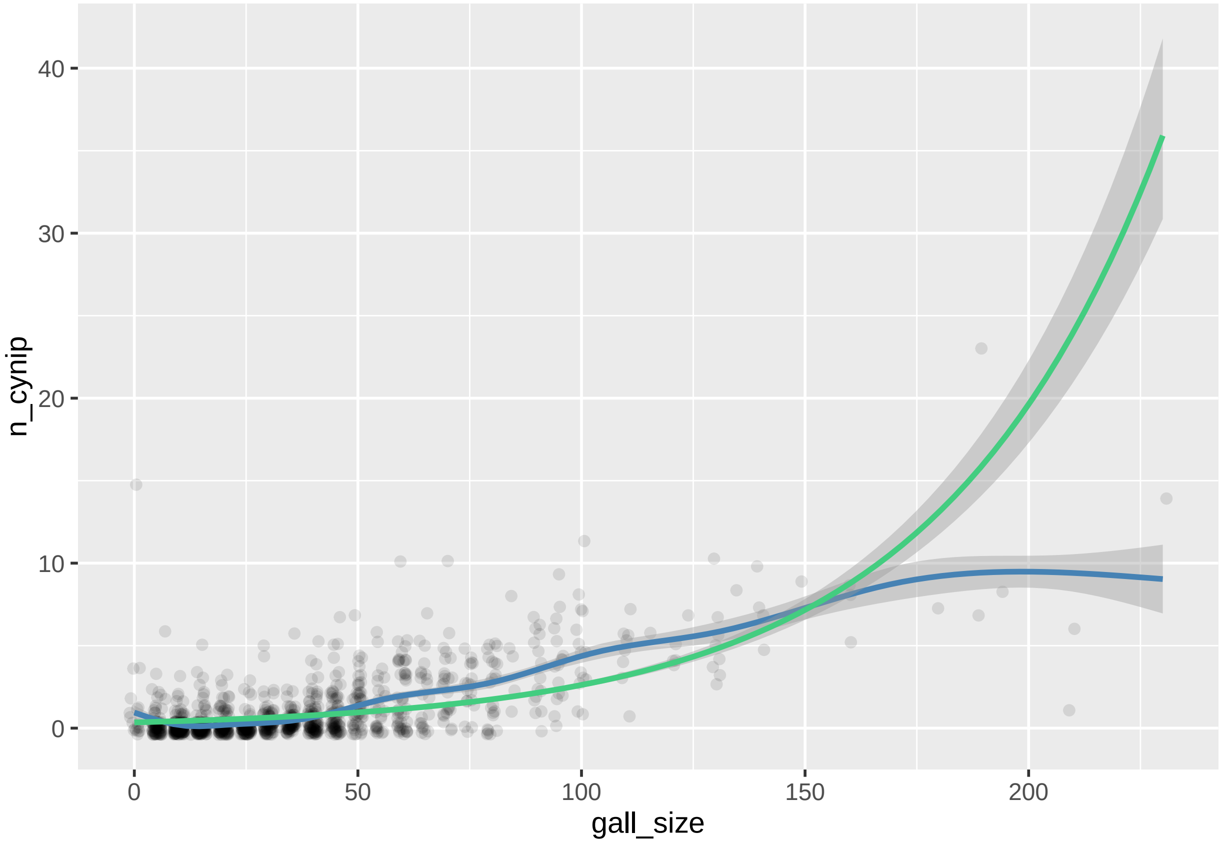
```
# order of localities by sample size
ordered_names <- sort(table(dat$gall_locality), decreasing = FALSE) %>%
  names()

# reorder localities in the data frame
dat$gall_locality <- factor(dat$gall_locality, levels = ordered_names)

# calculate mean number of wasps per sample location
loc_means <- data.frame(gall_locality = levels(dat$gall_locality),
  means = tapply(dat$n_cynip, dat$gall_locality,mean))
dat <- merge(dat, loc_means, by = "gall_locality")

# plot raw data with mean number of wasps
ggplot(data = dat, aes(n_cynip, gall_locality)) +
  geom_vline(xintercept = 0, color = "grey50") +
  geom_jitter(, alpha = 1/5) +
  geom_point(aes(means,gall_locality), pch = 21, fill = "seagreen3", size = 2)
```



It looks like a few sites have lots of galls while some have only a handful. Also, there are a lot of zeros in the data (all the points jittered around zero on the x-axis). There is also variation in the means across sites, so we’ll probably want to account for that in our model as well.

Let’s look at the relationship between gall size and the number of wasps

```r
# number of wasps as a function of gall size
ggplot(data = dat, aes(gall_size,n_cynip)) +
  geom_jitter(width = 3, alpha = 1/10) +
  stat_smooth(color = "steelblue") +
  stat_smooth(method = "glm",
      method.args = list(family = "poisson"), color = "seagreen3")
```

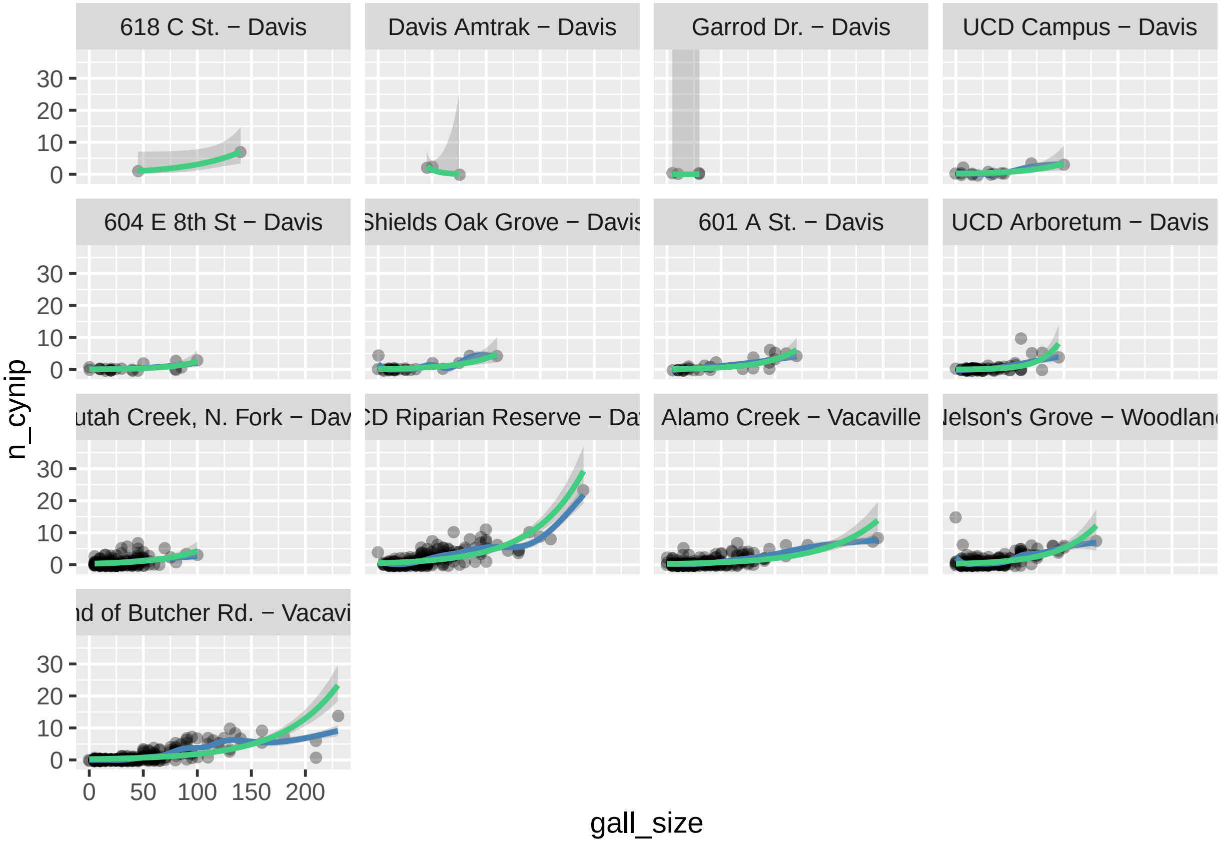


```r
  # number of emerging wasps increases with gall size
```

It would appear that the number of wasps levels off at higher gall sizes relative to standard Poisson (green) regression. Perhaps a quadratic term later on might help? Let's break it down by locality first:

```r
# break it down by locality
ggplot(data = dat, aes(gall_size,n_cynip)) +
  geom_jitter(alpha = 1/3) +
  stat_smooth(color = "steelblue") +
  stat_smooth(method = "glm",
    method.args = list(family = "poisson"), color = "seagreen3") +
  facet_wrap(~gall_locality)
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```



Hmm. It's possible that the overall pattern we saw before may have to do with different slope parameters in different populations, as there isn't too much difference between the Poisson regression (green) and the smooth curve(blue) within localities, except in the case of the Butcher Rd. site

Problem 2: model specification

What is your model? Write it in LaTeX.

Given the above, I want to fit a model that models the number of wasps as a Poisson distributed random variable that has different mean values for each population. Additionally, I'll fit a different gall size parameter for each locality, rather than a single gall size parameter applied across all localities

$$y \sim Poisson(\lambda)$$

$$log(\lambda) = X\beta$$

Here we are modeling the number of wasps y_i as a Poisson distributed random variable where beta has different intercept and gall size coefficients for each population (13 populations x 2 = 26 total parameters).

What is your Stan model statement?

```
data {
  int n; // number of galls
  int p; // number of parameters
  matrix[n, p] X;  // design matrix
  int y[n];  // the number of observed wasps emerging from each gall
}

parameters {
  vector[p] beta; // p length vector of parameters
}

model {
  beta ~ normal(0, 5);  // use a somewhat vague Normal prior centered at zero
  y ~ poisson_log(X * beta); // Poisson likelihood with log transformation
}
```

Note that I've given my intercept and my slope parameters all the same prior. We probably won't always do this, as it often makes sense to use different priors for these since they are fundamentally different parameters, but this prior is vague enough that it shouldn't matter too much

Also, keep in mind that centering the intercept priors at zero means that the prior is centered around a value of 1 on the raw data scale (hint: what does eˆ0 equal?) and that, while it's symmetric on the log scale, it's not on the raw-data scale

Problem 3: parameter estimation

First I want to center my gall size variable at zero so the site intercepts are not correlated with the site level gall size parameter. This also means my intercept parameters will now be estimates of mean wasp number for an average sized gall and not for gall size equals zero. I could instead center gall size within each population, but I won't do that here).

```r
# what is the average call size
mean(dat$gall_size)
```

```
## [1] 35.52998
```

```r
# center gall size(and scale to unit variance)
dat$gsize_adj <- scale(dat$gall_size, center = TRUE, scale = TRUE) %>%
  as.numeric  # the as.numeric() removes the attributes appended by scale()
```

Next I'll use R's glm() function to create my design matrix, which is easier than doing it by hand.

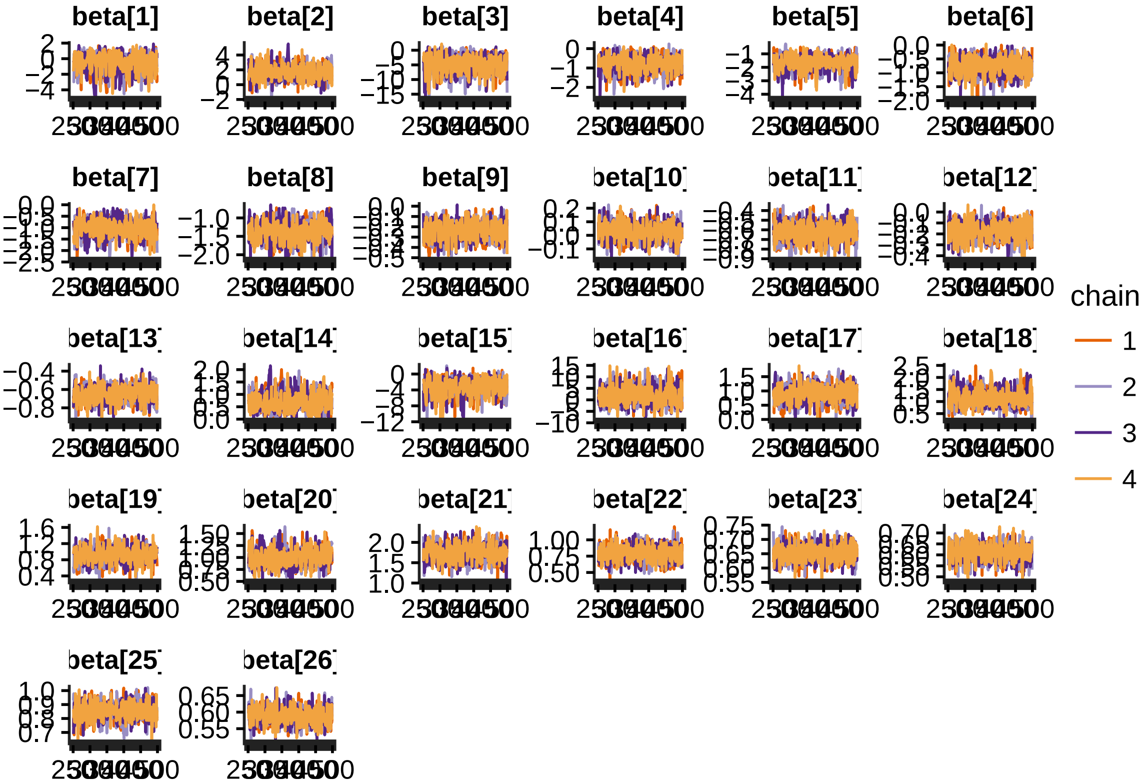
```r
# create the design matrix
X <- glm(n_cynip ~ 0 + gall_locality + gall_locality:gsize_adj, data = dat,
  family = "poisson") %>% model.matrix()
    # note suppression of global intercept

# create the data to input into Stan
stan_d <- list(n = nrow(dat),
               p = ncol(X),
               X = X,
                y = dat$n_cynip)

# tell Stan to be aware of all of my processors when fitting models
options(mc.cores = parallel::detectCores())

# fit the Stan model (note this takes awhile because p is big (26!)
gall_fit <- stan("poisson_glm_will.stan",
  data = stan_d,
  chains = 4,
  iter = 500,
  open_progress = FALSE)
```

Verify convergence using traceplots and the Rhat statistic:

```r
# check Rhat
gall_fit
```

```
## Inference for Stan model: poisson_glm_will.
## 4 chains, each with iter=500; warmup=250; thin=1;
## post-warmup draws per chain=250, total post-warmup draws=1000.
##
##              mean se_mean   sd    2.5%     25%     50%     75%   97.5%
## beta[1]     -0.71    0.06 1.27   -3.52   -1.50   -0.54    0.22    1.29
## beta[2]      1.72    0.04 1.00   -0.16    1.05    1.67    2.39    3.70
## beta[3]     -4.98    0.12 3.13  -11.80   -7.04   -4.56   -2.72    0.22
## beta[4]     -0.81    0.02 0.44   -1.75   -1.08   -0.78   -0.49   -0.04
## beta[5]     -1.68    0.02 0.56   -2.92   -1.98   -1.63   -1.27   -0.73
## beta[6]     -0.78    0.01 0.32   -1.44   -0.98   -0.78   -0.54   -0.17
## beta[7]     -1.03    0.02 0.39   -1.92   -1.28   -1.00   -0.74   -0.32
## beta[8]     -1.37    0.01 0.27   -1.93   -1.55   -1.37   -1.18   -0.84
## beta[9]     -0.24    0.00 0.09   -0.42   -0.31   -0.24   -0.17   -0.08
## beta[10]     0.03    0.00 0.07   -0.09   -0.01    0.03    0.07    0.16
## beta[11]    -0.62    0.00 0.10   -0.81   -0.68   -0.61   -0.55   -0.43
## beta[12]    -0.18    0.00 0.08   -0.34   -0.23   -0.18   -0.12   -0.03
## beta[13]    -0.65    0.00 0.09   -0.82   -0.71   -0.65   -0.59   -0.49
## beta[14]     0.75    0.02 0.39    0.11    0.47    0.70    1.00    1.60
## beta[15]    -3.32    0.10 2.20   -8.23   -4.58   -3.02   -1.69    0.14
## beta[16]     2.39    0.14 4.04   -4.91   -0.46    2.25    5.00   10.69
## beta[17]     0.90    0.01 0.31    0.27    0.70    0.91    1.11    1.52
## beta[18]     1.17    0.01 0.35    0.56    0.92    1.15    1.39    1.94
## beta[19]     0.92    0.01 0.19    0.53    0.80    0.93    1.05    1.31
## beta[20]     1.01    0.01 0.19    0.66    0.88    1.00    1.13    1.41
## beta[21]     1.75    0.01 0.21    1.36    1.62    1.74    1.89    2.15
## beta[22]     0.79    0.00 0.12    0.56    0.70    0.78    0.86    1.01
## beta[23]     0.65    0.00 0.03    0.59    0.63    0.65    0.67    0.71
## beta[24]     0.61    0.00 0.04    0.53    0.59    0.61    0.64    0.69
## beta[25]     0.85    0.00 0.06    0.72    0.81    0.85    0.90    0.97
## beta[26]     0.59    0.00 0.03    0.54    0.57    0.59    0.61    0.64
## lp__      -538.83    0.17 3.42 -545.68 -541.13 -538.62 -536.34 -532.89
##           n_eff Rhat
## beta[1]     409 1.00
## beta[2]     681 1.00
## beta[3]     738 1.00
## beta[4]    681 1.00
## beta[5]    545 1.00
## beta[6]    812 1.00
## beta[7]    632 1.01
## beta[8]    747 1.00
## beta[9]   1000 1.00
## beta[10]   685 1.00
## beta[11]  1000 1.00
## beta[12]  1000 1.00
## beta[13]   859 1.00
## beta[14]   365 1.01
## beta[15]   521 1.00
## beta[16]   829 1.00
## beta[17]   748 1.00
## beta[18]   588 1.00
## beta[19]   778 1.00
## beta[20]   609 1.01
## beta[21]   797 1.00
## beta[22]  1000 1.00
## beta[23]   731 1.00
## beta[24]   894 1.00
## beta[25]  1000 1.00
## beta[26]   867 1.00
## lp__       418 1.00
##
## Samples were drawn using NUTS(diag_e) at Mon Feb 15 13:43:47 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# traceplot
traceplot(gall_fit, pars = "beta")
```



Looks good

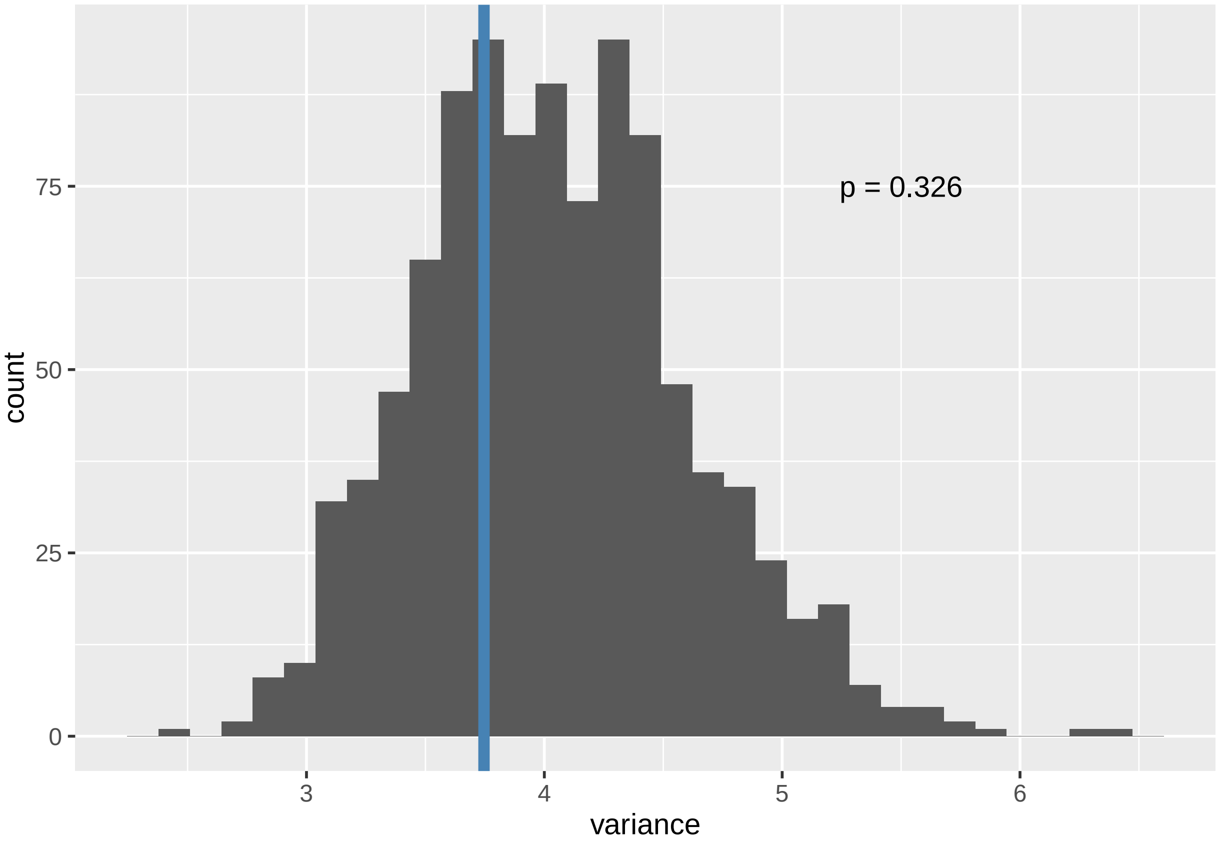
Problem 4: posterior predictive check

Does your model adequately capture the variance in the emergence data, or is there overdispersion?

In addition to checking the variance predicted by the model, I'm also going to check whether my model adequately captures the number of zeros in the data. If there are more zeros in the actual data than my model predicts, it might be an indication of zero-inflation (which could be do some extra effect I haven't accounted for)

```r
# extract the posteriors
posts <- extract(gall_fit)$beta

# write a function calculate the variance
calc_var <- function(X, beta){
  n <- nrow(X)  # number of data points
  lambda <- c(exp(X %*% beta)) # simulate lambda for the draw
  y <- rpois(n, lambda)
  var_y <- var(y)
}

# write a function to calculate the number of zeros
calc_zeros <- function(X, beta){
  n <- nrow(X)  # number of data points
  lambda <- c(exp(X %*% beta)) # simulate lambda for the draw
  y <- rpois(n, lambda)
  zero_y <- sum(y == 0)
}

# how many draws do we have
n_draws <- nrow(posts)

# create a data.frame to store our variances and zero counts
sims <- data.frame(variance = rep(NA, n_draws),
                   n_zeros = rep(NA, n_draws))

# simulate variances and zero counts for each draw
for(i in 1:n_draws){
  sims[i,"variance"] <- calc_var(X, beta = posts[i, ])
  sims[i,"n_zeros"] <- calc_zeros(X, beta = posts[i, ])
}

# calculate Bayesian P-values
var_p <- sum(sims$variance < var(dat$n_cynip))/n_draws
zeros_p <- sum(sims$n_zeros > sum(dat$n_cynip == 0))/n_draws

# now plot the distribution of simulated variances with the actual variance
ggplot(data = sims, aes(variance)) +
  geom_histogram() +
  geom_vline(xintercept = var(dat$n_cynip), color = "steelblue", size = 2) +
  annotate("text", x = 5.5, y = 75,
    label = paste0("p = ", var_p))
```

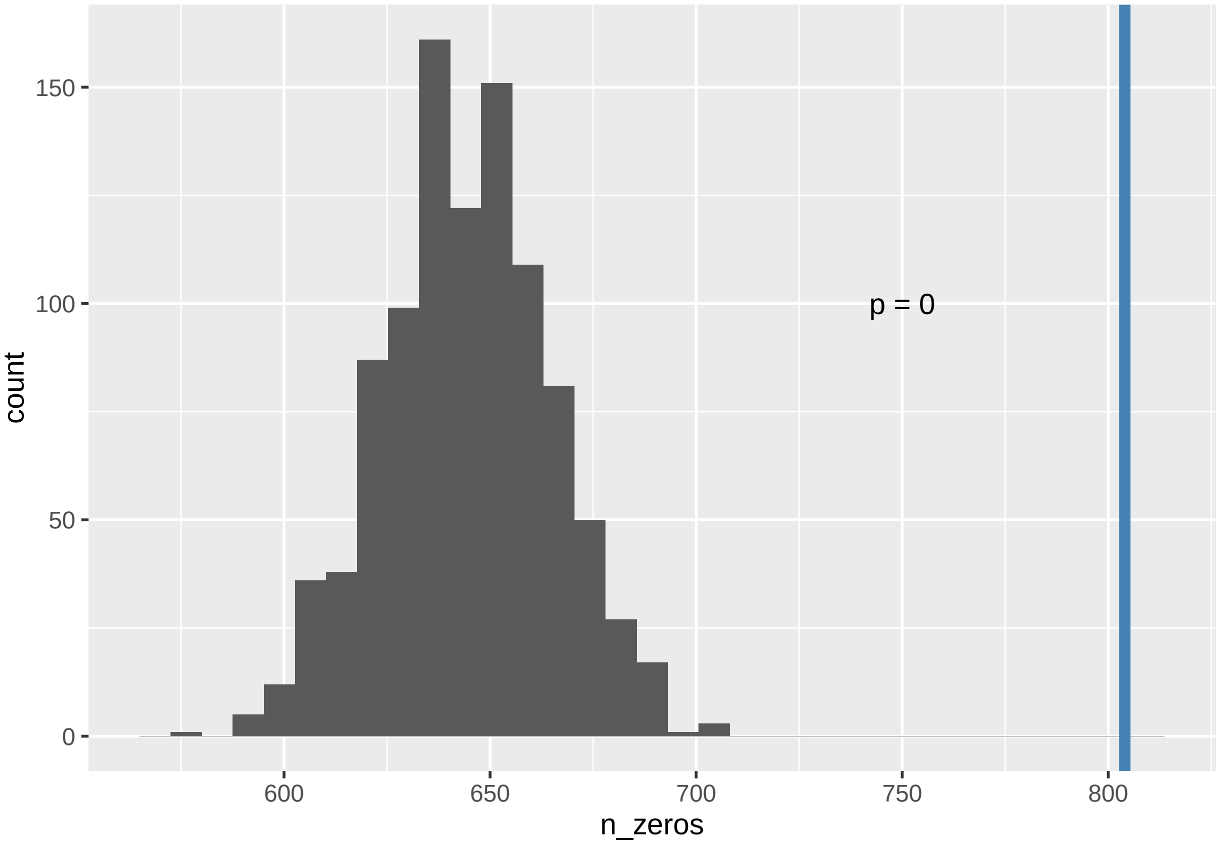
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Looks like our model captures the observed variance quite well, which would indicate that there is no overdispersion

```
# now plot the distribution of simulated number of zeros
ggplot(data = sims, aes(n_zeros)) +
  geom_histogram() +
  geom_vline(xintercept = sum(dat$n_cynip == 0), color = "steelblue", size = 2) +
  annotate("text", x = 750, y = 100,
    label = paste0("p = ", zeros_p))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



However, it looks like we are undershooting the number of zeros by 100-200, so there may be some other factor that is causing there to be zero emerging wasps in some galls beyond what is predicted by random Poisson variation.