Homework Chapter 10

Emily Maloney

March 17, 2019

```
library(tidyverse)
library(brms)
library(rethinking)
library(MASS)
```

Chapter 10 Homework

Easy Problems

10E1

```
prob <- 0.35
odds <- prob/(1 - prob)
logodds <- log(odds)
```

The log-odds of an event with 0.35 probability is -0.6190392.

10E2

```
logodds <- 3.2
odds <- exp(logodds)
prob <- odds/(1+odds)
```

The probability of an event with log-odds 3.2 is 0.9608343.

10E3

```
logodds <- 1.7
odds <- exp(logodds)
```

If a coefficient in a logistic regression has a value of 1.7, this implies that the proportional increase in odds of the outcome is 5.4739474.

10E4

Poisson regressions sometimes require the use of an offset because there can be different amounts of exposure across cases. Adding the logarithm of the exposure to the linear model effectively accounts for this. An example might be if you're modeling the number of times a student speaks in class by gender, you should include an offset term that accounts for attendance rate in class.

Medium Problems

10M1

When the data is aggregated in a binomial model, the likelihood needs to account for the fact that the ordering of the aggregated trials could happen in a number of different ways. To do this, the likelihood for

the aggregated binomial has a coefficient indiciating the multiplicity of the aggregation. It does not for the disaggregated binomial, because there is only one way to order a single trial.

10M2

```
exp(1.7)
```

```
## [1] 5.473947
```

A coefficient in a Poisson regression with a value of 1.7 indicates that the odds of the outcome are 5.47 times greater.

10M3

The logit link is appropriate for the binomial generalized linear model because the logit function effectively limits the outcome to be between 0 and 1, which means that the model's estimate of the probability will remain within appropriate bounds. You can't have a negative probability of an event occuring or a probability of an event occuring that is greater than one. The logit link ensures that estimates will fall within these bounds.

10M4

The log link is appropriate for the Poisson generalized linear model because the log function bounds the lower limit at 0. Because the Poisson distribution is used with a count outcome with an unknown maximum, ensuring that the model does not consider negative counts but doesn't limit the upper bound is required.

10M5

Using a logit link for the mean of a Poisson generalized linear model would imply that you had a theoretically-motivated reason to believe that the upper bound of the count needs to be limited at some value.

10M6

The constraints for which the binomial distribution has maximum entropy are that there are only two unordered events and that the expected value is constant. The constraints for which the Poisson distribution has maximum entropy are also that there are only two unordered events and that the expected value is constant. The constraints are the same because the Poisson distribution is a special case of the binomial distribution in which probability of the event is small and number of trials is large or unknown.

Hard Problems

10H1

```
data("chimpanzees")
d <- chimpanzees

#map approximation
h10.1 <- map(
          alist(
            pulled_left ~ dbinom(1, p),
            logit(p) <- a[actor] + (bp + bpC*condition)*prosoc_left,
            a[actor] ~ dnorm(0, 10),
            bp ~ dnorm(0, 10),
            bpC ~ dnorm(0, 10)
```

```
        ),
        data = d
)


#using MCMC
h10.2 <-
  brm(data = d, family = binomial,
      pulled_left | trials(1) ~ 0 + factor(actor) + prosoc_left + condition:prosoc_left ,
      prior(normal(0, 10), class = b),
      iter = 2500, warmup = 500, chains = 2, cores = 2,
      control = list(adapt_delta = 0.9))
```

```
## Only 2 levels detected so that family 'bernoulli' might be a more efficient choice.
```

```
## Compiling the C++ model
```

```
## Start sampling
```

```
precis(h10.1, depth = 2)
```

```
##       Mean StdDev  5.5% 94.5%
## a[1] -0.73   0.27 -1.16 -0.30
## a[2]  6.67   3.61  0.90 12.45
## a[3] -1.03   0.28 -1.48 -0.59
## a[4] -1.03   0.28 -1.48 -0.59
## a[5] -0.73   0.27 -1.16 -0.30
## a[6]  0.21   0.27 -0.21  0.64
## a[7]  1.75   0.38  1.14  2.37
## bp    0.82   0.26  0.40  1.24
## bpC  -0.13   0.30 -0.61  0.34
```

```
fixef(h10.2) %>% round(digits = 2)
```

```
##                       Estimate Est.Error  Q2.5 Q97.5
## factoractor1             -0.74      0.27 -1.28 -0.22
## factoractor2             11.03      5.39  3.94 24.46
## factoractor3             -1.05      0.28 -1.62 -0.53
## factoractor4             -1.04      0.28 -1.59 -0.51
## factoractor5             -0.73      0.27 -1.26 -0.21
## factoractor6              0.22      0.27 -0.30  0.74
## factoractor7              1.81      0.40  1.06  2.64
## prosoc_left               0.83      0.26  0.31  1.35
## prosoc_left:condition    -0.13      0.30 -0.72  0.45
```

The primary difference between the estimates from the quadratic approximation and the model that used MCMC to produce the estimates is the intercept for actor 2. In the quadratic approximation, the intercept is 11.05 while in the MCMC model, the intercept is 10.96. In the data actor 2 pulled the left lever every time, skewing the posterior distribution. Since the quadratic approximation assumes that the posterior distributions are multivariate Gaussian, it underestimates the intercept for this actor. MCMC does not have this assumption, meaning that it can capture the skew of actor 2's posterior distribution more effectively. This results in an increased intercept in comparison to the quadratic approximation estimate.

The fact that most of the other estimates are very similar to each other in the model estimated using the quadratic approximation and the MCMC model indicates that these other posterior distributions are not skewed and meet the multivariate Gaussian assumption required by the quadratic approximation.

10H2

```
#simpler models
h10.3 <-
  brm(data = d, family = binomial,
      pulled_left | trials(1) ~ 1,
      prior(normal(0, 10), class = Intercept))
```

```
## Only 2 levels detected so that family 'bernoulli' might be a more efficient choice.
```

```
## Compiling the C++ model
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'c99755203d00b2c8fecf6f5cf3db74fd' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.445 seconds (Warm-up)
## Chain 1:                1.336 seconds (Sampling)
## Chain 1:                2.781 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'c99755203d00b2c8fecf6f5cf3db74fd' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.382 seconds (Warm-up)
## Chain 2:                1.243 seconds (Sampling)
## Chain 2:                2.625 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'c99755203d00b2c8fecf6f5cf3db74fd' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.251 seconds (Warm-up)
## Chain 3:                1.264 seconds (Sampling)
## Chain 3:                2.515 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'c99755203d00b2c8fecf6f5cf3db74fd' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
```

```
## Chain 4:  Elapsed Time: 1.464 seconds (Warm-up)
## Chain 4:                1.318 seconds (Sampling)
## Chain 4:                2.782 seconds (Total)
## Chain 4:
```

```
h10.4 <-
  brm(data = d, family = binomial,
      pulled_left | trials(1) ~ 1 + prosoc_left,
      prior = c(prior(normal(0, 10), class = Intercept),
                prior(normal(0, 10), class = b)))
```

```
## Only 2 levels detected so that family 'bernoulli' might be a more efficient choice.
```

```
## Compiling the C++ model
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.812 seconds (Warm-up)
## Chain 1:                2.699 seconds (Sampling)
## Chain 1:                4.511 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.885 seconds (Warm-up)
## Chain 2:                2.565 seconds (Sampling)
## Chain 2:                4.45 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.771 seconds (Warm-up)
## Chain 3:                1.64 seconds (Sampling)
## Chain 3:                3.411 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.925 seconds (Warm-up)
## Chain 4:                2.739 seconds (Sampling)
## Chain 4:                4.664 seconds (Total)
## Chain 4:
```

```
h10.5 <-
  update(h10.4,
         newdata = d,
         formula = pulled_left | trials(1) ~ 1 + prosoc_left + condition:prosoc_left)
```

```
## Only 2 levels detected so that family 'bernoulli' might be a more efficient choice.
## Start sampling
```

```
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.682 seconds (Warm-up)
## Chain 1:                2.447 seconds (Sampling)
## Chain 1:                5.129 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.001 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 2.482 seconds (Warm-up)
## Chain 2:                2.255 seconds (Sampling)
## Chain 2:                4.737 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 2.159 seconds (Warm-up)
## Chain 3:                2.27 seconds (Sampling)
## Chain 3:                4.429 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '52c42f84b157fd37b125c1c23331427f' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 2.42 seconds (Warm-up)
## Chain 4:                2.569 seconds (Sampling)
## Chain 4:                4.989 seconds (Total)
## Chain 4:
```

```
waic(h10.2, h10.3, h10.4, h10.5)
```

```
##                   WAIC    SE
## h10.2          529.66 19.92
## h10.3          687.86  7.05
## h10.4          680.41  9.33
## h10.5          682.41  9.49
## h10.2 - h10.3 -158.20 19.86
## h10.2 - h10.4 -150.75 19.16
## h10.2 - h10.5 -152.75 19.11
## h10.3 - h10.4    7.45  6.17
## h10.3 - h10.5    5.44  6.34
## h10.4 - h10.5   -2.00  0.89
```

The model including the unique intercepts for each actor (h10.2) has the lowest WAIC by far - more than 150 points lower than the second best model (h10.4). The other three models are fairly similar to each other, with the model including an interaction between condition and prosocial_left (h10.5) having a slightly higher WAIC than the h10.4, which does not have an interaction term. This indicates that the benefit from including the interaction term is not signficant enough to counteract the penalty from adding another term to the model. Unsurprisingly, the model with only an intercept has the highest WAIC of all of the models.

10H3

a)

```
data("eagles")
e <- eagles

e <- e %>% mutate(P = ifelse(P == "L", 1, 0),
                  A = ifelse(A == "A", 1, 0),
                  V = ifelse(V == "L", 1, 0))

h10.6 <- map(
         alist(
           y ~ dbinom(n, p),
           logit(p) <- a + bp*P + bv*V + ba*A,
           a ~ dnorm(0, 10),
           bp ~ dnorm(0, 5),
           bv ~ dnorm(0, 5),
           ba ~ dnorm(0, 5)
         ),
         data = e,
         method="Nelder-Mead" , control=list(maxit=1e4)
)

h10.7 <-
```

```
  brm(data = e, family = binomial,
     y | trials(n) ~ 1 + P + V + A,
      c(prior(normal(0, 10), class = Intercept),
                prior(normal(0, 5), class = b)),
      iter = 2500, warmup = 500, chains = 2, cores = 2)
```

```
## Compiling the C++ model
```

```
## Start sampling
```

```
precis(h10.6, depth = 2)
```

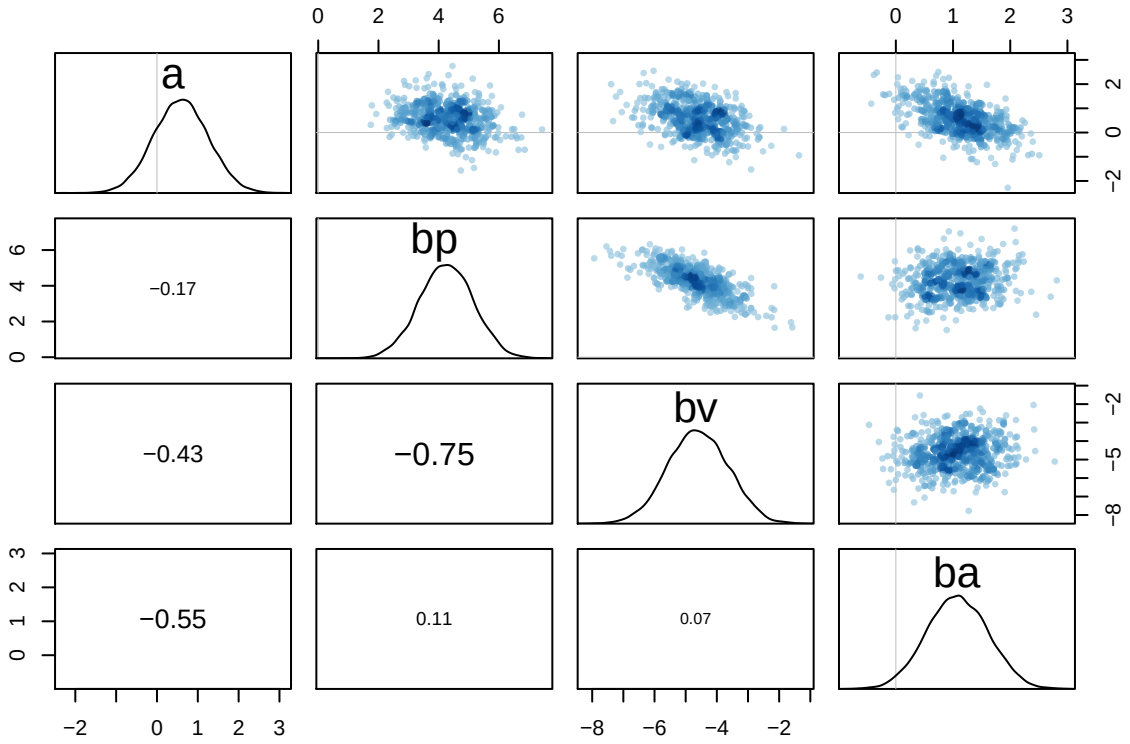
```
##     Mean StdDev  5.5% 94.5%
## a   0.59   0.66 -0.47  1.65
## bp  4.24   0.90  2.81  5.68
## bv -4.59   0.96 -6.13 -3.06
## ba  1.08   0.53  0.23  1.93
```

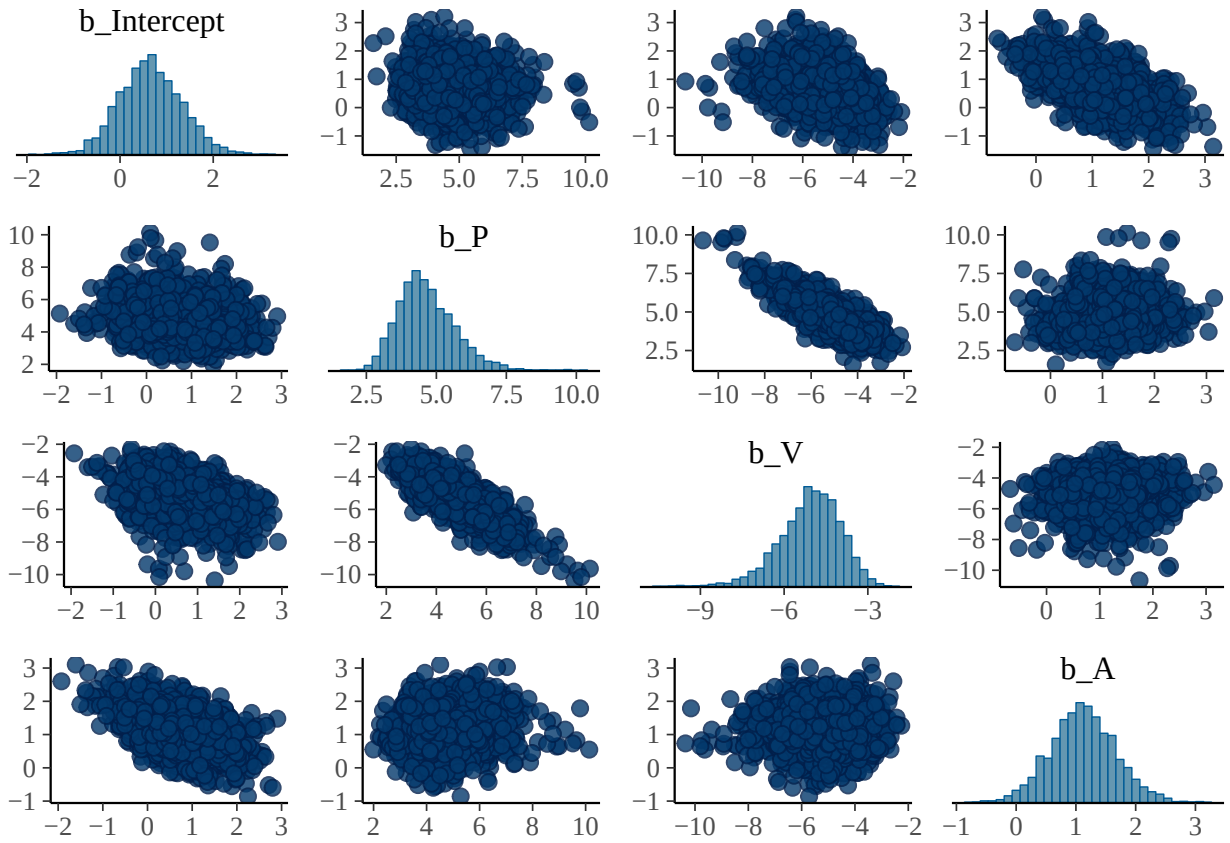
```
fixef(h10.7) %>% round(digits = 2)
```

```
##           Estimate Est.Error  Q2.5 Q97.5
## Intercept     0.66      0.67 -0.61  2.01
## P             4.66      1.05  2.97  7.01
## V            -5.07      1.10 -7.48 -3.22
## A             1.12      0.55  0.05  2.23
```

The estimates for the coefficients produced by each model are similar, but not close enough to not warrant looking into the posterior distributions of the coefficients. In particular, the estimate for the coefficients for P is about 0.5 points greater in the MCMC model than the map model, and similarly the coefficient for V is about 0.5 points lower in the MCMC model than the estimate generated by the map model.

```
pairs(h10.6)
pairs(h10.7)
```

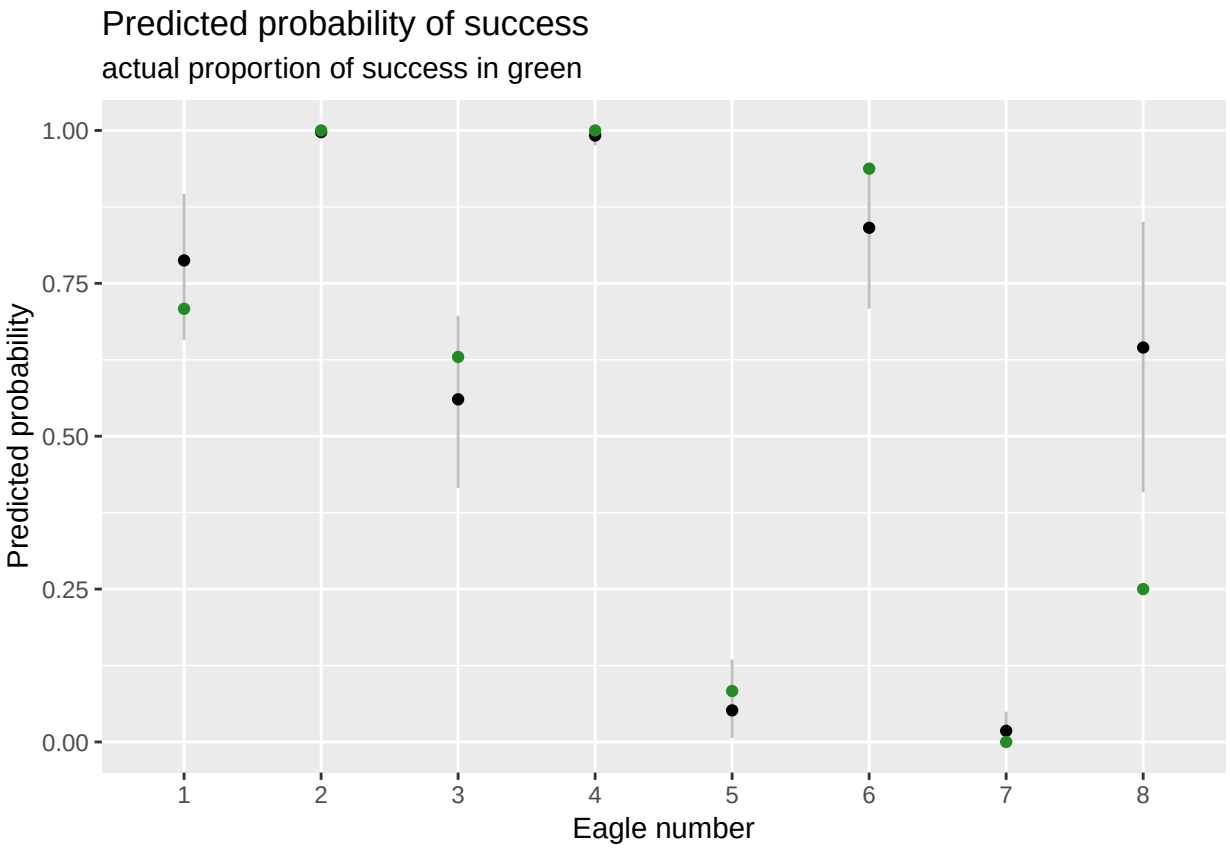

a
bp
bv
ba
−0.17
−0.43
−0.75
−0.55
0.11
0.07



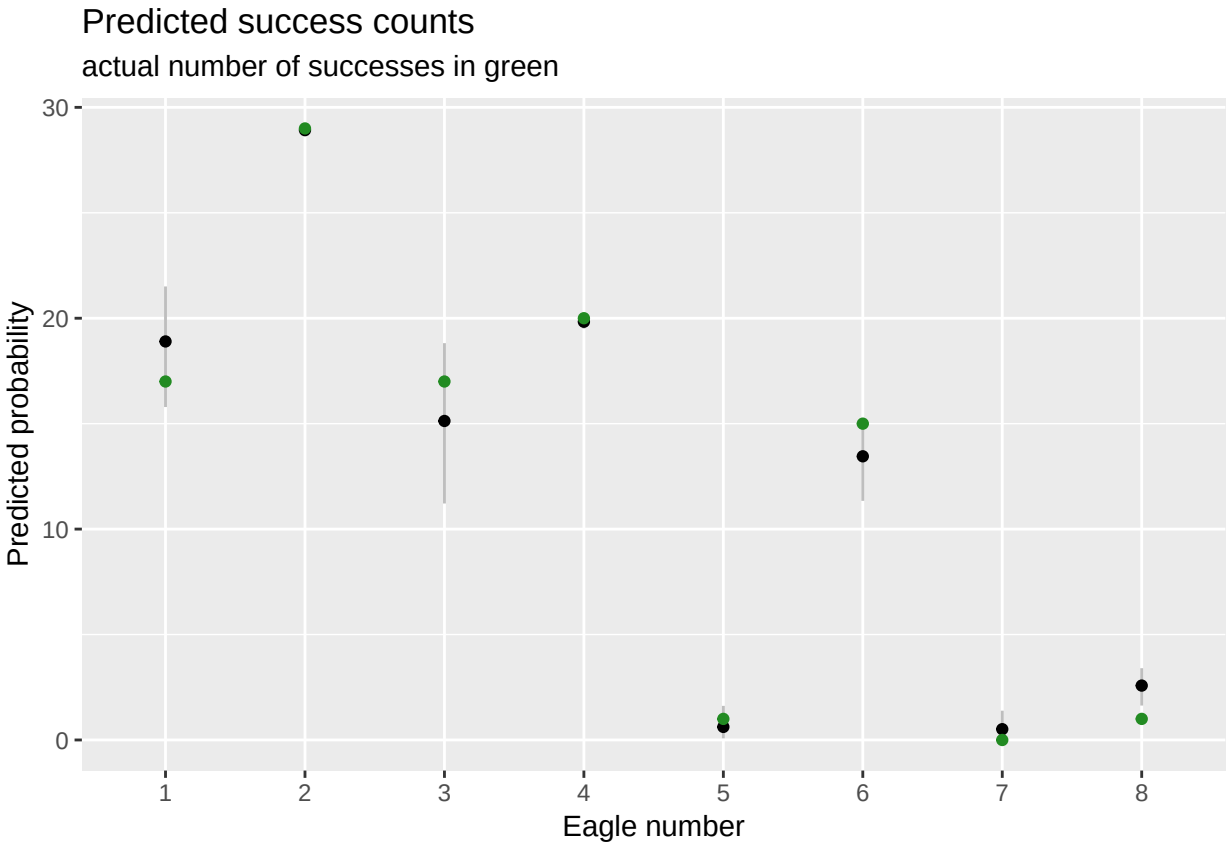
Looking at the pairs plot for the model estimated using MCMC (h10.7), the posterior distributions for b_P and b_V exhibit right skew and left skew, respectively. This indicates that using the quadratic approximation is not okay, because the assumption of multivariate Gaussian posterior distributions is not met.

b)

```
ftd <-
  fitted(h10.7, probs = c(0.055, 0.945)) %>%
  as_tibble() %>%
  bind_cols(h10.7$data)

ftd <- ftd %>% mutate(prob = Estimate/n,
                      probl = Q5.5/n,
                      probh = Q94.5/n)

ftd <- mutate(ftd, id = rownames(ftd)) %>% mutate(actual = e$y,
                                                  actualp = actual/n)

ggplot(data = ftd, mapping = aes(x = id, y = prob)) +
        geom_linerange(mapping = aes(ymin = probl, ymax = probh), color = "gray") +
        geom_point() +
        geom_point(mapping = aes(x = id, y = actualp), color = "forest green") +
        labs(title = "Predicted probability of success",
        subtitle = "actual proportion of success in green",
        x = "Eagle number", y = "Predicted probability")
```




```
ggplot(data = ftd, mapping = aes(x = id, y = Estimate)) +
        geom_linerange(mapping = aes(ymin = Q5.5, ymax = Q94.5), color = "gray") +
        geom_point() +
        geom_point(mapping = aes(x = id, y = actual), color = "forest green") +
        labs(title = "Predicted success counts",
        subtitle = "actual number of successes in green",
        x = "Eagle number", y = "Predicted probability")
```



The plot of the predicted probability of success indicates how likely it is for each eagle to succeed on any given trial. This information can be easily compared across the eagles, such that both eagle 2 and 4 have a predicted probability of 1, even though the counts in the data were 29 and 20, respectively, because the probability estimate took into account the number of attempts each eagle had. This plot shows how effective each eagle is at pirating other eagles' food.

The plot of predicted success counts shows how many successful trials we predict, given the number of trials they completed. This means that the predicted success count for eagle 2 and 4 are different, even though they have the same probability of success, because each has a different number of trials. Similarly, the estimate for the probability of success for eagle 8 is fairly high, but the predicted count is much lower, considering that this eagle only attempted to pirate 4 times. This plot shows how many times we would expect each eagle to pirate another eagle's food successfully.

c)

```
h10.8 <-
  update(h10.7,
         newdata = e,
         formula = y | trials(n) ~ 1 + P + V + A + P:A)
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL '556aefb70e0e86922dd47d7598e4903d' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
```

```
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 1: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 1: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 1: Iteration:  501 / 2500 [ 20%]  (Sampling)
## Chain 1: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 1: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 1: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 1: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 1: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 1: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 1: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 1: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.124 seconds (Warm-up)
## Chain 1:                0.506 seconds (Sampling)
## Chain 1:                0.63 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '556aefb70e0e86922dd47d7598e4903d' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 2: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 2: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 2: Iteration:  501 / 2500 [ 20%]  (Sampling)
## Chain 2: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 2: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 2: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 2: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 2: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 2: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 2: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 2: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.128 seconds (Warm-up)
## Chain 2:                0.474 seconds (Sampling)
## Chain 2:                0.602 seconds (Total)
## Chain 2:
```

```
waic(h10.7, h10.8)
```

```
##                WAIC   SE
## h10.7         30.32 8.14
## h10.8         20.67 6.02
## h10.7 - h10.8  9.66 3.26
```

The WAIC in the model with the interaction (h10.8) is lower than the WAIC for the model without the interaction (h10.7) by about 10 points, indicating that this model performs better.

10H4

```
data("salamanders")
s <- salamanders

h10.9 <-
  map( alist( SALAMAN ~ dpois( lambda ),
              log(lambda) <- a + bp*PCTCOVER,
              a ~ dnorm(0,10),
              bp ~ dnorm(0,5) ), data=s )

h10.10 <-
  brm(data = s, family = poisson,
     SALAMAN ~ 1 + PCTCOVER,
      c(prior(normal(0, 10), class = Intercept),
                prior(normal(0, 5), class = b)),
      iter = 2500, warmup = 500, chains = 2, cores = 2)
```

```
## Compiling the C++ model
```

```
## Start sampling
```

```
precis(h10.9, depth = 2)
```

```
##     Mean StdDev  5.5% 94.5%
## a  -3.23      0 -3.23 -3.23
## bp  5.90      0  5.90  5.90
```

```
fixef(h10.10) %>% round(digits = 2)
```

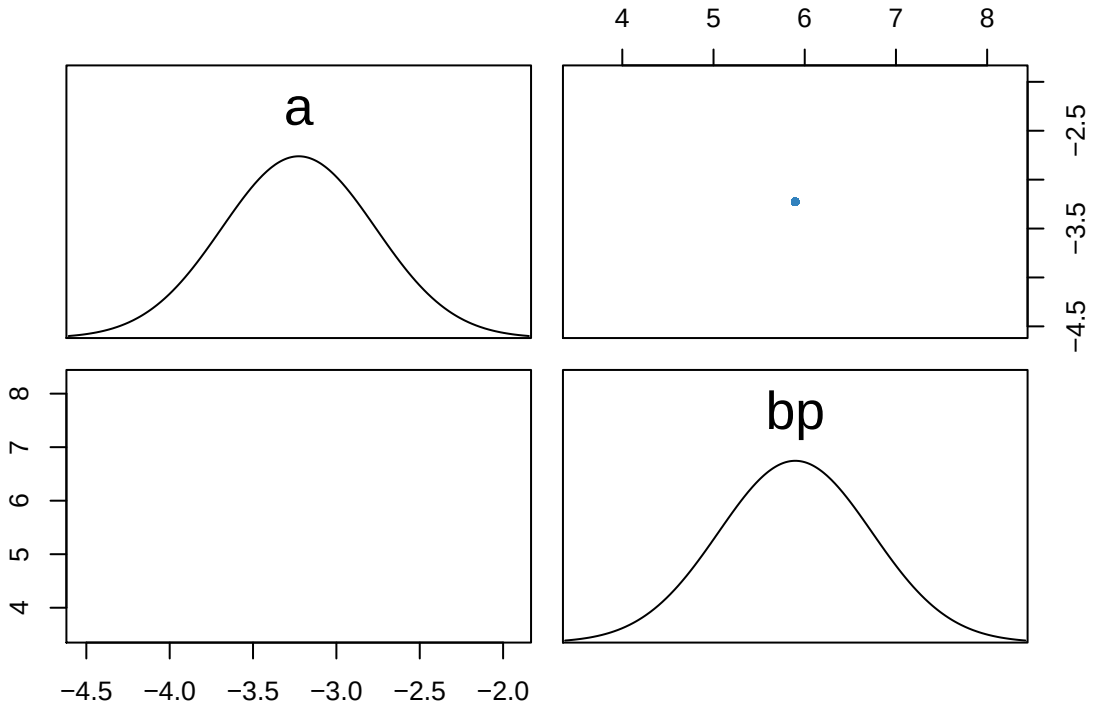
```
##           Estimate Est.Error  Q2.5 Q97.5
## Intercept    -1.55      0.48 -2.55 -0.67
## PCTCOVER      0.03      0.01  0.02  0.04
```

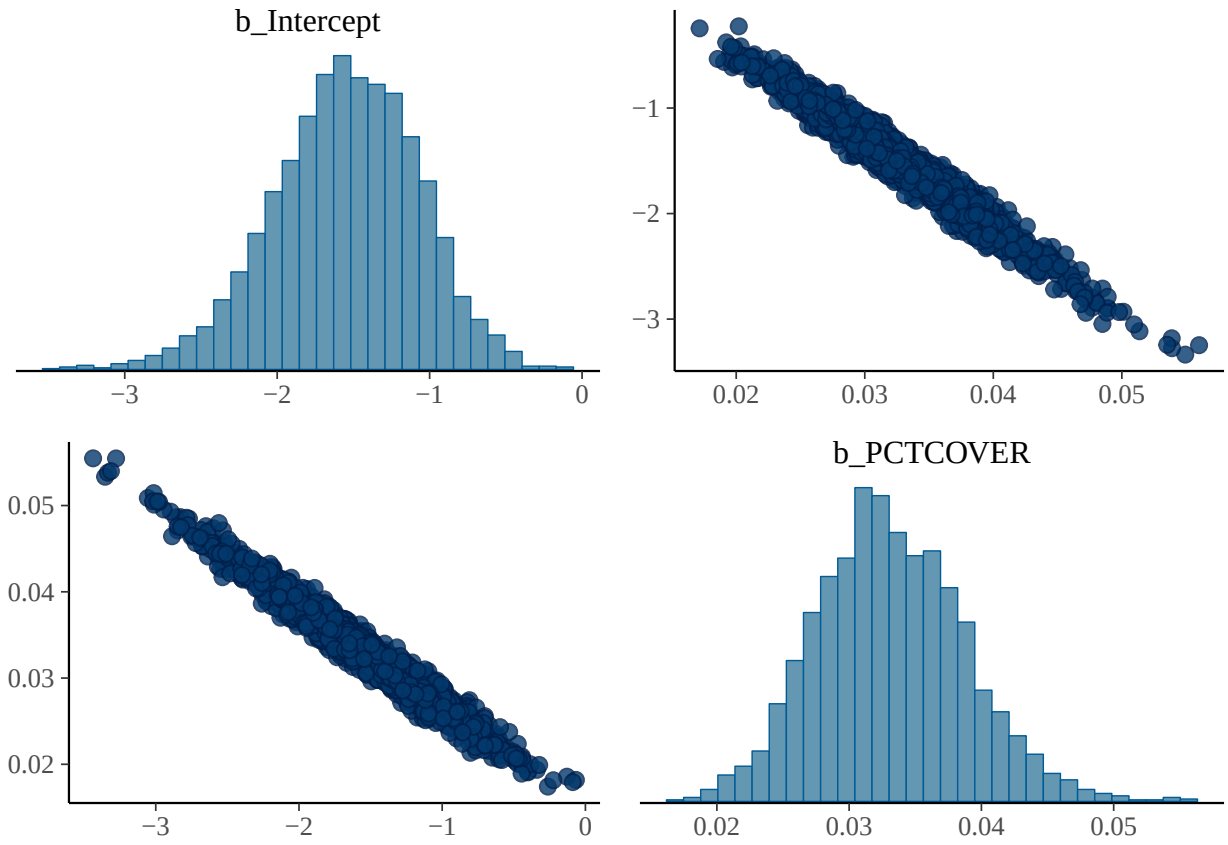
The estimates generated by the map model and the MCMC model are different, indicating that we should check the posterior distributions to see if the multivariate Gaussian assumption is not met.

```
pairs(h10.9)
```

```
## Warning in cor(x, y): the standard deviation is zero
```

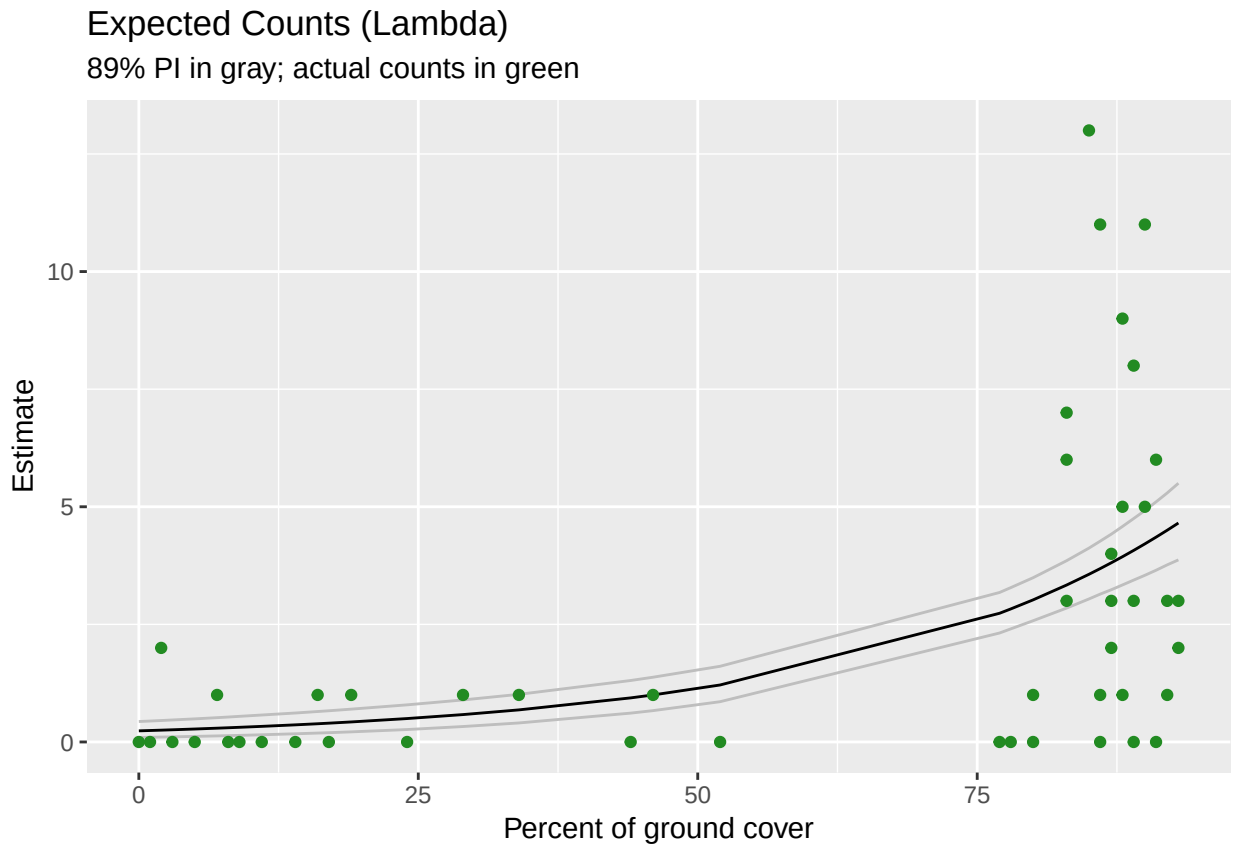
```
pairs(h10.10)
```


a
bp
4
5
6
7
8
−2.5
−3.5
−4.5
4
5
6
7
8
−4.5
−4.0
−3.5
−3.0
−2.5
−2.0



Indeed, the posterior distributions for both the intercept and percent cover both exhibit skew. Therefore, we should use the MCMC model and not the map model.

```
ftd <-
  fitted(h10.10, probs = c(0.055, 0.945)) %>%
  as_tibble() %>%
  bind_cols(h10.10$data)

ggplot(data = ftd, mapping = aes(x = PCTCOVER, y = Estimate)) + geom_line(color = "black") +
       geom_line(mapping = aes(x = PCTCOVER, y = Q94.5), color = "gray") +
       geom_line(mapping = aes(x = PCTCOVER, y = Q5.5), color = "gray") +
       geom_point(mapping = aes(x = PCTCOVER, y = SALAMAN), color = "forestgreen") +
       labs(title = "Expected Counts (Lambda)",
       subtitle = "89% PI in gray; actual counts in green",
       x = "Percent of ground cover", y = "Estimate")
```



The model does a good job estimating expected counts when the percent of ground cover is less than 50%, but fails to capture the range of data points when percent of ground cover is high, with much of the data falling both above and below the 89% interval. This appears to be because there is less of a consistent relationship between percent of ground cover and expected count when the percent of ground cover is high. Perhaps including the forestage variable will improve the model fit.

b)

```
h10.11 <-
  update(h10.10,
         newdata = s,
         formula = SALAMAN ~ 1 + FORESTAGE)
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'bc36538c452ea1ea4e3486603b979792' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 1: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 1: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 1: Iteration:  501 / 2500 [ 20%]  (Sampling)
```

```
## Chain 1: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 1: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 1: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 1: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 1: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 1: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 1: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 1: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.972 seconds (Warm-up)
## Chain 1:                2.396 seconds (Sampling)
## Chain 1:                3.368 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bc36538c452ea1ea4e3486603b979792' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 2: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 2: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 2: Iteration:  501 / 2500 [ 20%]  (Sampling)
## Chain 2: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 2: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 2: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 2: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 2: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 2: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 2: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 2: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.085 seconds (Warm-up)
## Chain 2:                36.64 seconds (Sampling)
## Chain 2:                36.725 seconds (Total)
## Chain 2:

## Warning: There were 1896 transitions after warmup that exceeded the maximum treedepth. Increase max_
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
h10.12 <- update(h10.10,
                 newdata = s,
                 formula = SALAMAN ~ 1 + PCTCOVER + FORESTAGE)
```

```
## Start sampling

##
## SAMPLING FOR MODEL 'bc36538c452ea1ea4e3486603b979792' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
```

```
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 1: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 1: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 1: Iteration:  501 / 2500 [ 20%]  (Sampling)
## Chain 1: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 1: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 1: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 1: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 1: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 1: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 1: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 1: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.766 seconds (Warm-up)
## Chain 1:                1.185 seconds (Sampling)
## Chain 1:                1.951 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bc36538c452ea1ea4e3486603b979792' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 2: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 2: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 2: Iteration:  501 / 2500 [ 20%]  (Sampling)
## Chain 2: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 2: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 2: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 2: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 2: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 2: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 2: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 2: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.05 seconds (Warm-up)
## Chain 2:                0.333 seconds (Sampling)
## Chain 2:                0.383 seconds (Total)
## Chain 2:

## Warning: There were 2 transitions after warmup that exceeded the maximum treedepth. Increase max_tre
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
h10.13 <- update(h10.10,
                 newdata = s,
                 formula = SALAMAN ~ 1 + PCTCOVER + FORESTAGE + PCTCOVER:FORESTAGE)
```

```
## Start sampling
##
## SAMPLING FOR MODEL 'bc36538c452ea1ea4e3486603b979792' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
```

```
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 1: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 1: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 1: Iteration:  501 / 2500 [ 20%]  (Sampling)
## Chain 1: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 1: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 1: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 1: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 1: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 1: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 1: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 1: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.064 seconds (Warm-up)
## Chain 1:                0.346 seconds (Sampling)
## Chain 1:                0.41 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bc36538c452ea1ea4e3486603b979792' NOW (CHAIN 2).
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
```

```
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
```

```
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
```

```
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2500 [  0%]  (Warmup)
## Chain 2: Iteration:  250 / 2500 [ 10%]  (Warmup)
## Chain 2: Iteration:  500 / 2500 [ 20%]  (Warmup)
## Chain 2: Iteration:  501 / 2500 [ 20%]  (Sampling)
## Chain 2: Iteration:  750 / 2500 [ 30%]  (Sampling)
## Chain 2: Iteration: 1000 / 2500 [ 40%]  (Sampling)
## Chain 2: Iteration: 1250 / 2500 [ 50%]  (Sampling)
## Chain 2: Iteration: 1500 / 2500 [ 60%]  (Sampling)
## Chain 2: Iteration: 1750 / 2500 [ 70%]  (Sampling)
## Chain 2: Iteration: 2000 / 2500 [ 80%]  (Sampling)
## Chain 2: Iteration: 2250 / 2500 [ 90%]  (Sampling)
## Chain 2: Iteration: 2500 / 2500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.088 seconds (Warm-up)
## Chain 2:                0.309 seconds (Sampling)
## Chain 2:                0.397 seconds (Total)
## Chain 2:
```

```
## Warning: There were 1054 divergent transitions after warmup. Increasing adapt_delta above 0.8 may hel
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 7 transitions after warmup that exceeded the maximum treedepth. Increase max_tree
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
waic(h10.10, h10.11, h10.12)
```

```
##                           WAIC           SE
## h10.10           2.135500e+02 2.671000e+01
## h10.11           2.006192e+05 1.876209e+05
## h10.12           2.338769e+23 1.287303e+23
## h10.10 - h10.11 -2.004056e+05 1.876221e+05
## h10.10 - h10.12 -2.338769e+23 1.287303e+23
## h10.11 - h10.12 -2.338769e+23 1.287303e+23
```

I estimated models in which expected count is predicted by just forest age (h10.13), by forest age and percent cover together (h10.12), and by forest age, precent cover, and an interaction between the two (h10.13). The model with the interaction term returns an infinite WAIC, so I did not include it in the model comparision, only evaluating the other two against the model with just percent cover (h10.10). Interestingly, that initial model has the lowest WAIC, and the model with both forest age and percent cover has the highest WAIC. This indicates that these two variables are likely explaining the same part of the variance in expected count, so including both in the model does add enough explanatory power to overcome the penalty for including another term.