Assignment6

Yung-Tang Chou

2019/5/22

Here is the rmarkdown file for the Multivariate Analysis Assignment 6.

< Question 1 >

The data is about counts of salamanders, modeling with the percentage of ground cover and age of tress. Before doing the modeling first let’s import needed libraries.

library(rstan)

## Loading required package: ggplot2

## Loading required package: StanHeaders

## rstan (Version 2.18.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling  
## options(mc.cores = parallel::detectCores()).  
## To avoid recompilation of unchanged Stan programs, we recommend calling  
## rstan\_options(auto\_write = TRUE)

## For improved execution time, we recommend calling  
## Sys.setenv(LOCAL\_CPPFLAGS = '-march=native')  
## although this causes Stan to throw an error on a few processors.

library(rethinking)

## Loading required package: parallel

## rethinking (Version 1.88)

library(loo)

## This is loo version 2.1.0.  
## \*\*NOTE: As of version 2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' argument or set options(mc.cores = NUM\_CORES) for an entire session. Visit mc-stan.org/loo/news for details on other changes.

## \*\*NOTE for Windows 10 users: loo may be very slow if 'mc.cores' is set in your .Rprofile file (see https://github.com/stan-dev/loo/issues/94).

##   
## Attaching package: 'loo'

## The following object is masked from 'package:rethinking':  
##   
## compare

## The following object is masked from 'package:rstan':  
##   
## loo

library(skimr)

##   
## Attaching package: 'skimr'

## The following object is masked from 'package:stats':  
##   
## filter

Let’s import the data. To improve the effectiveness of modeling, let’s do nomalization for dependent variables, PCTCOVER and FORESTAGE. Code is as below.

data("salamanders")  
d <- salamanders; rm(salamanders)  
d$PCTCOVER <- (d$PCTCOVER - mean(d$PCTCOVER)) / sd(d$PCTCOVER)  
d$FORESTAGE <- (d$FORESTAGE - mean(d$PCTCOVER)) / sd(d$FORESTAGE)

A basic summary statistics is shown below.

## Skim summary statistics  
## n obs: 47   
## n variables: 4   
##   
## -- Variable type:integer --------------------------------------------------------------------------  
## variable missing complete n mean sd p0 p25 p50 p75 p100 hist  
## SALAMAN 0 47 47 2.47 3.37 0 0 1 3 13 ▇▂▁▁▁▁▁▁  
## SITE 0 47 47 24 13.71 1 12.5 24 35.5 47 ▇▇▇▇▇▇▇▇  
##   
## -- Variable type:numeric --------------------------------------------------------------------------  
## variable missing complete n mean sd p0 p25 p50 p75 p100  
## FORESTAGE 0 47 47 0.87 1 -1.2e-20 0.15 0.33 1.38 3.49  
## PCTCOVER 0 47 47 2.2e-18 1 -1.65 -1.15 0.67 0.81 0.95  
## hist  
## ▇▁▂▁▁▁▁▁  
## ▃▁▁▁▁▁▁▇

Let’s start by building the stan model. In the first model we want to predict the relationship between the count of salamaders and percent ground cover. To model counts, Poisson Distribution will be the best choice;therefore in this question we will form a Poisson model. We are asked to use weakly informative priors to do the modeling. Hence, I intentionally choose a flatter prior. The model is shown below.

model.01.stan <- "  
data{  
 vector[47] FORESTAGE;  
 int SITE[47];  
 int SALAMAN[47];  
 vector[47] PCTCOVER;  
}  
parameters{  
 real a;  
 real bP;  
}  
model{  
 vector[47] lambda;  
 a ~ normal( 0 , 50 );  
 bP ~ normal( 0 , 20 );  
 for ( i in 1:47 ) {  
 lambda[i] = a + bP \* PCTCOVER[i];  
 lambda[i] = exp(lambda[i]);  
 }  
 SALAMAN ~ poisson( lambda );  
}  
generated quantities{  
 vector[47] log\_lik;  
 vector[47] lambda;  
 for ( i in 1:47 ) {  
 lambda[i] = a + bP \* PCTCOVER[i];  
 lambda[i] = exp(lambda[i]);  
 }  
 for ( i in 1:47 ) log\_lik[i] = poisson\_lpmf( SALAMAN[i] | lambda[i] );  
}  
"

Note that within the generated quantities block, the log\_lik is also calculated.

Now we are going to run the stan model. Before doing so, to prevent stan model from crashing, we form a specific dataset for the stan model. Code is shown below.

dat <- list(  
 N = NROW(d),  
 SITE = d$SITE,  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER,  
 FORESTAGE = d$FORESTAGE  
)

For the stan model, we will use “dat” generated above as the input data, and use four markov chains to do the modeling. The model code is shown below.

model.01 <- stan(model\_code = model.01.stan, data = dat, chains = 4)

##   
## SAMPLING FOR MODEL '6c5a5550c3d24272bf8bb6126a67f64d' 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: 0.079 seconds (Warm-up)  
## Chain 1: 0.085 seconds (Sampling)  
## Chain 1: 0.164 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL '6c5a5550c3d24272bf8bb6126a67f64d' 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: 0.088 seconds (Warm-up)  
## Chain 2: 0.099 seconds (Sampling)  
## Chain 2: 0.187 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL '6c5a5550c3d24272bf8bb6126a67f64d' 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: 0.09 seconds (Warm-up)  
## Chain 3: 0.088 seconds (Sampling)  
## Chain 3: 0.178 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL '6c5a5550c3d24272bf8bb6126a67f64d' 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: 0.09 seconds (Warm-up)  
## Chain 4: 0.084 seconds (Sampling)  
## Chain 4: 0.174 seconds (Total)  
## Chain 4:

The descriptions of the model is shown below, using print statement.

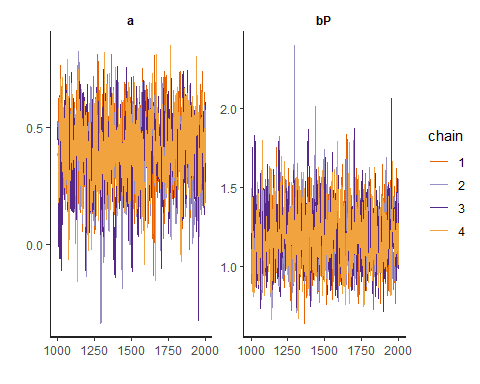
print(model.01, include=FALSE, pars=c('log\_lik','lambda'), probs = c(.1, .5, .9))

## Inference for Stan model: 6c5a5550c3d24272bf8bb6126a67f64d.  
## 4 chains, each with iter=2000; warmup=1000; thin=1;   
## post-warmup draws per chain=1000, total post-warmup draws=4000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a 0.40 0.01 0.16 0.18 0.41 0.59 969 1.01  
## bP 1.20 0.01 0.20 0.95 1.19 1.45 989 1.01  
## lp\_\_ 22.28 0.03 0.98 21.06 22.57 23.16 1244 1.00  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 22:21:41 2019.  
## 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).

Rhat = 1 means that the model can reach convergence in this Markov Chain. n\_eff is greater than a thousand, which implys a pretty solid model. From the description we can know that percentage of ground cover is positive related to the counts of salamanders, with mean slope being around 1.18 and mean intercept being roughly 0.41.

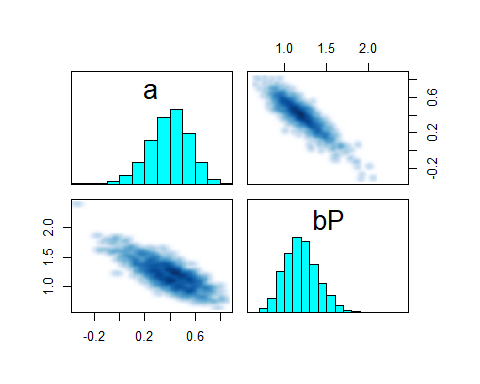
It would be a good idea to check whether the stan model is decent graphically. In order to do that we use traceplot to see if all the params lie in a specific range. The code is as below.

traceplot(model.01, pars = c('a','bP'))

 From the traceplot above we can easily tell that the model converges.

Besides, let’s see the distribution of the model params using pairplot. The code is as below.

pairs(model.01, pars = c('a','bP'))



From the plots above, we can tell the following things. 1. The stan model converges properly. The traceplot shows that all the values lie in a specific range, which is the desired outcome. 2. The pairplot shows that the distribution of param bP is right-skewed.

Now it’s time to evaluate the model. In this question we use try to use loo to examine the model’s effectiveness. Code is shown below.

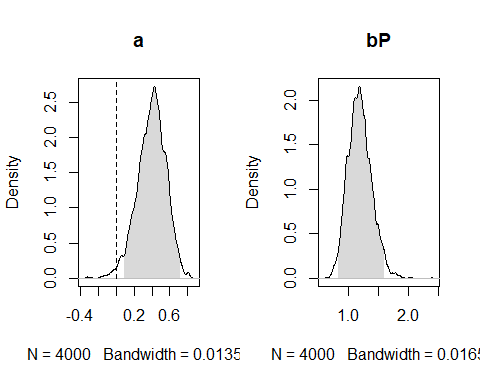
log\_lik\_01 <- extract\_log\_lik(model.01, merge\_chains = FALSE)  
r\_eff <- relative\_eff(exp(log\_lik\_01))  
loo\_01 <- loo(log\_lik\_01, r\_eff = r\_eff, cores = 2)  
loo\_01

##   
## Computed from 4000 by 47 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_loo -106.6 13.4  
## p\_loo 4.6 1.1  
## looic 213.2 26.7  
## ------  
## Monte Carlo SE of elpd\_loo is 0.1.  
##   
## All Pareto k estimates are good (k < 0.5).  
## See help('pareto-k-diagnostic') for details.

Monte Carlo SE of elpd\_loo is 0.1. All Pareto k estimates are good, with k < 0.5. This model is pretty decent in doing simulations, even though the prior is pretty flat.

Before ending this question let’s plot out the density plot to see the distribution. The codes and plots are shown below.

post <- extract.samples(model.01)  
par(mfrow= c(1,2))  
dens(post$a, show.HPDI = .95, show.zero = T, add = F, main = 'a')  
dens(post$bP, show.HPDI = .95, show.zero = T, add = F, main = 'bP')

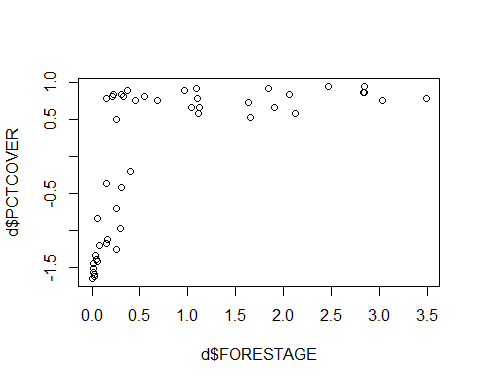


From the plot we can see that there is still a slim chance for the intercept a being negative number. It is impossible to have negative counts. This is what the poorly-used flat priors can confuse us. However, the slope will almost always be positive, counteracting the effects of negative intercept. This model is still good, but it can be better by having a more strongly informative priors.

< Question 2 > How can we improve the model by using the second variable, FORESTAGE? There may be three ways to include FORESTAGE into the model. 1) Using only FORESTAGE as the dependent variable. 2) Using both PCTCOVER and FORESTAGE, but without interactions. 3) Using both PCTCOVER and FORESTAGE with interactions.

Before trying the model, let’s see what we can get by simply plotting out the distribution. The code is shown below.

par(mfrow= c(1,1))  
plot(d$PCTCOVER ~ d$FORESTAGE)



Let’s try the three models mentioned above to test whether using the variable FORESTAGE can improve the model. In later models we will use the same mean and sd for all variables.

The first model is shown below.

model.02.stan <- "  
data{  
 vector[47] PCTCOVER;  
int SITE[47];  
int SALAMAN[47];  
vector[47] FORESTAGE;  
}  
parameters{  
real a;  
real bF;  
}  
model{  
vector[47] lambda;  
bF ~ normal( 0 , 20 );  
a ~ normal( 0 , 50 );  
for ( i in 1:47 ) {  
lambda[i] = a + bF \* FORESTAGE[i];  
lambda[i] = exp(lambda[i]);  
}  
SALAMAN ~ poisson( lambda );  
}  
generated quantities{  
vector[47] log\_lik;  
vector[47] lambda;  
for ( i in 1:47 ) {  
lambda[i] = a + bF \* FORESTAGE[i];  
lambda[i] = exp(lambda[i]);  
}  
for ( i in 1:47 ) log\_lik[i] = poisson\_lpmf( SALAMAN[i] | lambda[i] );  
}  
"

Similar to what we have done above, let’s use stan to build the model and print out the descriptions. The model code is shown below.

model.02 <- stan(model\_code = model.02.stan, data = dat, chains = 4)

##   
## SAMPLING FOR MODEL '59a1769b32c6d5fc65e136a3f5333056' 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: 0.079 seconds (Warm-up)  
## Chain 1: 0.081 seconds (Sampling)  
## Chain 1: 0.16 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL '59a1769b32c6d5fc65e136a3f5333056' 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: 0.08 seconds (Warm-up)  
## Chain 2: 0.074 seconds (Sampling)  
## Chain 2: 0.154 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL '59a1769b32c6d5fc65e136a3f5333056' 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: 0.08 seconds (Warm-up)  
## Chain 3: 0.086 seconds (Sampling)  
## Chain 3: 0.166 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL '59a1769b32c6d5fc65e136a3f5333056' 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: 0.076 seconds (Warm-up)  
## Chain 4: 0.076 seconds (Sampling)  
## Chain 4: 0.152 seconds (Total)  
## Chain 4:

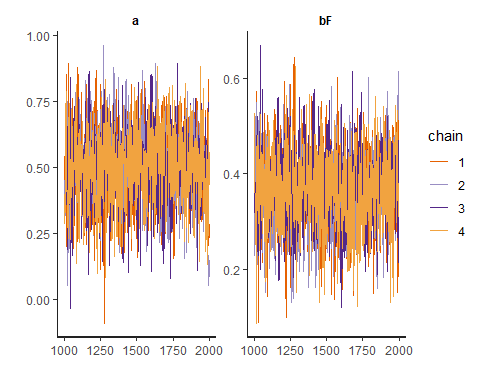
print(model.02, include=FALSE, pars=c('log\_lik','lambda'), probs = c(.1, .5, .9))

## Inference for Stan model: 59a1769b32c6d5fc65e136a3f5333056.  
## 4 chains, each with iter=2000; warmup=1000; thin=1;   
## post-warmup draws per chain=1000, total post-warmup draws=4000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a 0.50 0.00 0.14 0.32 0.50 0.68 1254 1  
## bF 0.37 0.00 0.08 0.26 0.37 0.47 1142 1  
## lp\_\_ -2.46 0.03 1.03 -3.82 -2.13 -1.52 1524 1  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 22:22:44 2019.  
## 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).

Rhat = 1 means that the model can reach convergence in this Markov Chain. n\_eff is greater than a thousand, which implys a pretty solid model. From the description we can know that percentage of ground cover is positive related to the counts of salamanders, with mean slope being around 0.37 and mean intercept being roughly 0.50.

Again, it would be a good idea to check whether the stan model is decent graphically. In order to do that we use traceplot to see if all the params lie in a specific range. The code is as below.

traceplot(model.02, pars = c('a','bF'))

 From the traceplot above we can easily tell that the model converges.

For later model comparion, let’s again use loo. The code is shown below.

log\_lik\_02 <- extract\_log\_lik(model.02, merge\_chains = FALSE)  
r\_eff <- relative\_eff(exp(log\_lik\_02))  
loo\_02 <- loo(log\_lik\_02, r\_eff = r\_eff, cores = 2)

For the second and third model, we will replicate the same procedure again and again. Code is shown below with only some intepretations.

For model two, with both PCTCOVER and FORESTAGE but without interactions.

model.03.stan <- "  
data{  
int SITE[47];  
int SALAMAN[47];  
vector[47] FORESTAGE;  
vector[47] PCTCOVER;  
}  
parameters{  
real a;  
real bF;  
real bP;  
}  
model{  
vector[47] lambda;  
bP ~ normal( 0 , 20 );  
bF ~ normal( 0 , 20 );  
a ~ normal( 0 , 50 );  
for ( i in 1:47 ) {  
lambda[i] = a + bP \* PCTCOVER[i] + bF \* FORESTAGE[i];  
lambda[i] = exp(lambda[i]);  
}  
SALAMAN ~ poisson( lambda );  
}  
generated quantities{  
vector[47] log\_lik;  
vector[47] lambda;  
for ( i in 1:47 ) {  
lambda[i] = a + bP \* PCTCOVER[i] + bF \* FORESTAGE[i];  
lambda[i] = exp(lambda[i]);  
}  
for ( i in 1:47 ) log\_lik[i] = poisson\_lpmf( SALAMAN[i] | lambda[i] );  
}  
"

The model is built through the code below, with descriptions provided.

model.03 <- stan(model\_code = model.03.stan, data = dat, chains = 4)

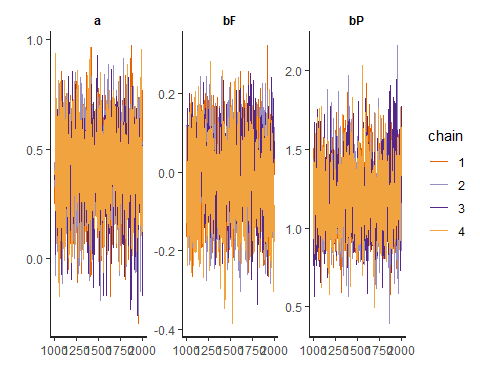
##   
## SAMPLING FOR MODEL '3b9e35bdb36e05b5c934e56cea80bb0d' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 0 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## 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: 0.141 seconds (Warm-up)  
## Chain 1: 0.138 seconds (Sampling)  
## Chain 1: 0.279 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL '3b9e35bdb36e05b5c934e56cea80bb0d' 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: 0.132 seconds (Warm-up)  
## Chain 2: 0.124 seconds (Sampling)  
## Chain 2: 0.256 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL '3b9e35bdb36e05b5c934e56cea80bb0d' 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: 0.128 seconds (Warm-up)  
## Chain 3: 0.13 seconds (Sampling)  
## Chain 3: 0.258 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL '3b9e35bdb36e05b5c934e56cea80bb0d' 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: 0.229 seconds (Warm-up)  
## Chain 4: 0.212 seconds (Sampling)  
## Chain 4: 0.441 seconds (Total)  
## Chain 4:

print(model.03, include=FALSE, pars=c('log\_lik','lambda'), probs = c(.1, .5, .9))

## Inference for Stan model: 3b9e35bdb36e05b5c934e56cea80bb0d.  
## 4 chains, each with iter=2000; warmup=1000; thin=1;   
## post-warmup draws per chain=1000, total post-warmup draws=4000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a 0.41 0.00 0.19 0.16 0.41 0.64 2131 1  
## bF -0.01 0.00 0.10 -0.14 -0.01 0.12 2117 1  
## bP 1.20 0.01 0.22 0.93 1.18 1.49 1914 1  
## lp\_\_ 21.63 0.04 1.31 19.88 21.96 22.91 1344 1  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 22:23:48 2019.  
## 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).

With two independent variables, we can see that the mean slope of variable FORESTAGE will be roughly 0, and the mean slope of variable PCTCOVER being 1.19.

traceplot(model.03, pars = c('a','bF','bP'))

 Traceplot above also indicates that the model converges.

For future model comparison, loo is calculated below.

log\_lik\_03 <- extract\_log\_lik(model.03, merge\_chains = FALSE)  
r\_eff <- relative\_eff(exp(log\_lik\_03))  
loo\_03 <- loo(log\_lik\_03, r\_eff = r\_eff, cores = 2)

## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

For the last model, which includes interaction effects between PCTCOVER and FORESTAGE, the code is shown below.

model.04.stan <- "  
data{  
int SITE[47];  
int SALAMAN[47];  
vector[47] FORESTAGE;  
vector[47] PCTCOVER;  
}  
parameters{  
real a;  
real bF;  
real bP;  
real bPF;  
}  
model{  
vector[47] lambda;  
bPF ~ normal( 0 , 20 );  
bP ~ normal( 0 , 20 );  
bF ~ normal( 0 , 20 );  
a ~ normal( 0 , 50 );  
for ( i in 1:47 ) {  
lambda[i] = a + bP \* PCTCOVER[i] + bF \* FORESTAGE[i] + bPF \* PCTCOVER[i] \* FORESTAGE[i];  
lambda[i] = exp(lambda[i]);  
}  
SALAMAN ~ poisson( lambda );  
}  
generated quantities{  
vector[47] log\_lik;  
vector[47] lambda;  
for ( i in 1:47 ) {  
lambda[i] = a + bP \* PCTCOVER[i] + bF \* FORESTAGE[i] + bPF \* PCTCOVER[i] \* FORESTAGE[i];  
lambda[i] = exp(lambda[i]);  
}  
for ( i in 1:47 ) log\_lik[i] = poisson\_lpmf( SALAMAN[i] | lambda[i] );  
}  
"

The descriptions is shown below.

model.04 <- stan(model\_code = model.04.stan, data = dat, chains = 4)

##   
## SAMPLING FOR MODEL '7f9855f89a8fd0d2d5b137e53f058f90' 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: 0.69 seconds (Warm-up)  
## Chain 1: 0.67 seconds (Sampling)  
## Chain 1: 1.36 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL '7f9855f89a8fd0d2d5b137e53f058f90' 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: 0.681 seconds (Warm-up)  
## Chain 2: 0.88 seconds (Sampling)  
## Chain 2: 1.561 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL '7f9855f89a8fd0d2d5b137e53f058f90' 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: 0.596 seconds (Warm-up)  
## Chain 3: 0.748 seconds (Sampling)  
## Chain 3: 1.344 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL '7f9855f89a8fd0d2d5b137e53f058f90' 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: 0.595 seconds (Warm-up)  
## Chain 4: 0.438 seconds (Sampling)  
## Chain 4: 1.033 seconds (Total)  
## Chain 4:

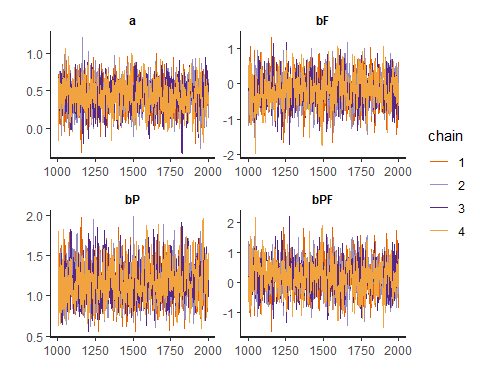
print(model.04, include=FALSE, pars=c('log\_lik','lambda'), probs = c(.1, .5, .9))

## Inference for Stan model: 7f9855f89a8fd0d2d5b137e53f058f90.  
## 4 chains, each with iter=2000; warmup=1000; thin=1;   
## post-warmup draws per chain=1000, total post-warmup draws=4000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a 0.43 0.01 0.20 0.18 0.44 0.68 1292 1.00  
## bF -0.18 0.01 0.44 -0.74 -0.18 0.38 1040 1.01  
## bP 1.17 0.01 0.23 0.88 1.15 1.48 1341 1.00  
## bPF 0.20 0.02 0.52 -0.46 0.21 0.85 1060 1.01  
## lp\_\_ 21.32 0.04 1.42 19.42 21.65 22.79 1497 1.00  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 22:25:00 2019.  
## 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).

With interacton effects included, now the mean slope of bF turn out to be negative, which means that FORESTAGE itself is negatively related to the counts of salamanders. PCTCOVER itself still remains positive related to the counts of salamanders. For the interaction effects, it turns out to be positive related to the counts of salamanders.

The traceplot below indicates that the model actually reach convergence.

traceplot(model.04, pars = c('a','bF','bP','bPF'))



For comparison reasons, let’s compute the loo value of the interaction models.

log\_lik\_04 <- extract\_log\_lik(model.04, merge\_chains = FALSE)  
r\_eff <- relative\_eff(exp(log\_lik\_04))  
loo\_04 <- loo(log\_lik\_04, r\_eff = r\_eff, cores = 2)

## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

Now it’s time to do model comparison. We will try both loo and waic to determine which model is the best, and in the meantime try to identify whether adding variable FORESTAGE can do good to modeling.

loo::compare(loo\_01, loo\_02, loo\_03, loo\_04)

## elpd\_diff se\_diff elpd\_loo p\_loo looic   
## loo\_01 0.0 0.0 -106.6 4.6 213.2  
## loo\_03 -2.7 0.8 -109.3 8.2 218.6  
## loo\_04 -4.5 1.3 -111.1 10.5 222.2  
## loo\_02 -26.3 11.4 -132.9 7.5 265.9

From loo, the first model with only PCTCOVER turns out to the best model, which is somewhat surprising. The multivariate model with both PCTCOVER and FORESTAGE comes behind, and the model consisting only variable FORESTAGE turns out to be the worst one.

rethinking::compare(model.01, model.02, model.03, model.04)

## WAIC pWAIC dWAIC weight SE dSE  
## model.01 213.1372 4.568690 0.000000 9.243790e-01 26.40936 NA  
## model.03 218.5193 8.156455 5.382094 6.268206e-02 27.44553 1.661861  
## model.04 221.6750 10.288303 8.537767 1.293891e-02 28.15627 2.428381  
## model.02 265.7878 7.470634 52.650598 3.411284e-12 35.66207 22.739936

From waic, we can reach the same conclusions as using loo.

In addition, there’s little difference between model.01 and model.03, which implys that the multivarite model using both PCTCOVER and FORESTAGE is pretty similar to the univariate model using only PCTCOVER. Even the model with interaction effects does not have a greater performance. How is this possible?

The model containing only the PCTCOVER gets most of the weight, in this case being 90%. From the results stemming from modeling, we can conclude that there is not much evidence that forestage matters.