Assign6

Ryan Huang

5/21/2019

# Question 1

Model the relationship between density and percent cover, using a log- link

data(salamanders, package = 'rethinking')  
d = salamanders  
# d$PCTCOVER = d$PCTCOVER / 100  
d$PCTCOVER\_log = ifelse(d$PCTCOVER==0, 0, log(d$PCTCOVER))

Stan Model

m1.1='  
data {  
 int N;  
 int SALAMAN[N];   
 real PCTCOVER[N];   
}  
parameters {  
 real a;  
 real bp;  
}  
model {  
 vector[N] lambda;  
 a ~ normal(0,100);  
 bp ~ normal(0,10);  
  
 for(i in 1:N) lambda[i] = a + bp \* (PCTCOVER[i]);  
 SALAMAN ~ poisson\_log(lambda);  
}  
  
generated quantities {  
 vector[N] log\_lik;  
 vector[N] lambda;  
 {  
 for(i in 1:N) {  
 lambda[i] = a + bp \* (PCTCOVER[i]);  
 log\_lik[i] = poisson\_log\_lpmf(SALAMAN[i] | lambda[i]);  
 }  
 }  
}  
'

Data

dat1.1 = list(  
 N = nrow(d),  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log  
)

Model fitting

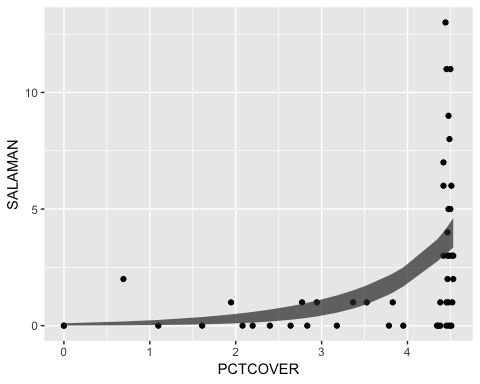
fit1.1 = stan(model\_code = m1.1, data = dat1.1, cores = 4)

Prediction

print(fit1.1, include = F, pars = 'log\_lik', probs = c(.1, .5, .9))

## Inference for Stan model: 2cefd2c9c0f80e7b6263698607e28862.  
## 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 -3.71 0.04 0.95 -4.94 -3.68 -2.52 542 1.01  
## bp 1.12 0.01 0.22 0.85 1.11 1.40 545 1.01  
## lambda[1] 1.27 0.00 0.09 1.15 1.28 1.39 3241 1.00  
## lambda[2] 1.29 0.00 0.09 1.17 1.29 1.40 3159 1.00  
## lambda[3] 1.34 0.00 0.10 1.21 1.34 1.46 2715 1.00  
## lambda[4] 1.31 0.00 0.10 1.19 1.32 1.43 2955 1.00  
## lambda[5] 1.32 0.00 0.10 1.20 1.33 1.44 2845 1.00  
## lambda[6] 1.25 0.00 0.09 1.13 1.25 1.36 3387 1.00  
## lambda[7] 1.25 0.00 0.09 1.13 1.25 1.36 3387 1.00  
## lambda[8] 1.35 0.00 0.10 1.22 1.35 1.47 2567 1.00  
## lambda[9] 1.31 0.00 0.10 1.19 1.32 1.43 2955 1.00  
## lambda[10] 1.34 0.00 0.10 1.21 1.34 1.46 2715 1.00  
## lambda[11] 1.30 0.00 0.10 1.18 1.30 1.42 3067 1.00  
## lambda[12] 1.25 0.00 0.09 1.13 1.25 1.36 3387 1.00  
## lambda[13] 1.30 0.00 0.10 1.18 1.30 1.42 3067 1.00  
## lambda[14] 1.32 0.00 0.10 1.20 1.33 1.44 2845 1.00  
## lambda[15] 1.36 0.00 0.10 1.23 1.36 1.48 2429 1.00  
## lambda[16] 1.37 0.00 0.10 1.24 1.38 1.50 2301 1.00  
## lambda[17] -2.93 0.03 0.80 -3.97 -2.91 -1.93 542 1.01  
## lambda[18] 1.30 0.00 0.10 1.18 1.30 1.42 3067 1.00  
## lambda[19] 1.37 0.00 0.10 1.24 1.38 1.50 2301 1.00  
## lambda[20] -1.53 0.02 0.53 -2.22 -1.50 -0.86 545 1.01  
## lambda[21] -0.60 0.02 0.35 -1.06 -0.58 -0.16 556 1.01  
## lambda[22] -0.41 0.01 0.32 -0.82 -0.39 -0.01 561 1.01  
## lambda[23] 0.07 0.01 0.23 -0.24 0.08 0.36 591 1.01  
## lambda[24] 0.25 0.01 0.20 -0.02 0.25 0.50 616 1.01  
## lambda[25] 0.58 0.01 0.14 0.40 0.59 0.77 740 1.01  
## lambda[26] 1.21 0.00 0.09 1.09 1.21 1.32 3522 1.00  
## lambda[27] 1.29 0.00 0.09 1.17 1.29 1.40 3159 1.00  
## lambda[28] 1.31 0.00 0.10 1.19 1.32 1.43 2955 1.00  
## lambda[29] 1.36 0.00 0.10 1.23 1.36 1.48 2429 1.00  
## lambda[30] -3.71 0.04 0.95 -4.94 -3.68 -2.52 542 1.01  
## lambda[31] -3.71 0.04 0.95 -4.94 -3.68 -2.52 542 1.01  
## lambda[32] -2.48 0.03 0.71 -3.40 -2.45 -1.58 543 1.01  
## lambda[33] -1.91 0.03 0.60 -2.69 -1.88 -1.14 544 1.01  
## lambda[34] -1.38 0.02 0.50 -2.03 -1.36 -0.75 546 1.01  
## lambda[35] -1.25 0.02 0.48 -1.87 -1.23 -0.65 547 1.01  
## lambda[36] -1.02 0.02 0.43 -1.59 -1.00 -0.48 549 1.01  
## lambda[37] -0.75 0.02 0.38 -1.25 -0.73 -0.27 553 1.01  
## lambda[38] -0.53 0.01 0.34 -0.98 -0.52 -0.10 558 1.01  
## lambda[39] -0.15 0.01 0.27 -0.50 -0.13 0.19 573 1.01  
## lambda[40] 0.53 0.01 0.15 0.34 0.54 0.73 710 1.01  
## lambda[41] 0.72 0.00 0.13 0.56 0.73 0.88 874 1.01  
## lambda[42] 1.16 0.00 0.09 1.05 1.16 1.28 3510 1.00  
## lambda[43] 1.18 0.00 0.09 1.06 1.18 1.29 3533 1.00  
## lambda[44] 1.21 0.00 0.09 1.09 1.21 1.32 3522 1.00  
## lambda[45] 1.29 0.00 0.09 1.17 1.29 1.40 3159 1.00  
## lambda[46] 1.32 0.00 0.10 1.20 1.33 1.44 2845 1.00  
## lambda[47] 1.35 0.00 0.10 1.22 1.35 1.47 2567 1.00  
## lp\_\_ 17.48 0.04 0.99 16.16 17.79 18.37 746 1.01  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 21:50:34 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).

post1.1 <- as.data.frame(fit1.1)  
  
for( i in (1:nrow(d))){  
 if(i == 1){  
 pred = tibble(mean = exp(post1.1[,i+49]) %>% mean,  
 l\_PI = exp(post1.1[,i+49]) %>% PI%>% .[1],  
 h\_PI = exp(post1.1[,i+49]) %>% PI%>% .[2])  
 }   
 else{  
 pred = tibble(mean = exp(post1.1[,i+49]) %>% mean,  
 l\_PI = exp(post1.1[,i+49]) %>% PI%>% .[1],  
 h\_PI = exp(post1.1[,i+49]) %>% PI%>% .[2]) %>%   
 rbind(pred, .)  
 }  
}  
pred = pred %>% mutate(  
 SALAMAN = d$SALAMAN,  
 site = d$SITE,  
 PCTCOVER = d$PCTCOVER\_log  
)  
  
fig1.1 = pred %>%   
 ggplot() +   
 geom\_point(aes(x=PCTCOVER, y=SALAMAN)) +   
 geom\_ribbon(aes(x=PCTCOVER,  
 ymin = l\_PI,  
 ymax = h\_PI),  
 alpha = 0.7)  
fig1.1

 From the above plot we can tell that the prediction result is poor. We can’t infer the amount of salamanders by only including the percentage of PCTCOVER.

# Question 2

Improve the model by using other predictors, FORESTAGE

Model 2.1: lambda = a + bp \* PCTCOVER + bf \* FORESTAGE (w/o interaction)

m2.1='  
data {  
 int N;  
 int SALAMAN[N];   
 real PCTCOVER[N];   
 int FORESTAGE[N];  
}  
parameters {  
 real a;  
 real bp;  
 real bf;  
}  
model {  
 vector[N] lambda;  
 a ~ normal(0,100);  
 bp ~ normal(0,10);  
 bf ~ normal(0,10);  
  
 for(i in 1:N) lambda[i] = a + bp \* (PCTCOVER[i]) + bf \* (FORESTAGE[i]);  
 SALAMAN ~ poisson\_log(lambda);  
}  
  
generated quantities {  
 vector[N] log\_lik;  
 vector[N] lambda;  
 {  
 for(i in 1:N) {  
 lambda[i] = a + bp \* (PCTCOVER[i]) + bf \* (FORESTAGE[i]);  
 log\_lik[i] = poisson\_log\_lpmf(SALAMAN[i] | lambda[i]);  
 }  
 }  
}  
'

Model 2.2: Add category variable cover = ifelse(PCTCOVER > 70, 1, 0) w/ interaction between PCTCOVER and COVER

m2.2='  
data {  
 int N;  
 int SALAMAN[N];   
 real PCTCOVER[N];   
 int FORESTAGE[N];  
 int COVER[N];  
}  
parameters {  
 real a;  
 real bp;  
 real bf;  
 real bc;  
 real bpc;  
}  
model {  
 vector[N] lambda;  
 a ~ normal(0,100);  
 bp ~ normal(0,10);  
 bf ~ normal(0,10);  
 bc ~ normal(0,10);  
 bpc ~ normal(0,10);  
  
 for(i in 1:N) lambda[i] = a + bp \* (PCTCOVER[i]) + bf \* (FORESTAGE[i]) + bc \* (COVER[i]) + bpc \* COVER[i] \* PCTCOVER[i];  
 SALAMAN ~ poisson\_log(lambda);  
}  
  
generated quantities {  
 vector[N] log\_lik;  
 vector[N] lambda;  
 {  
 for(i in 1:N) {  
 lambda[i] = a + bp \* (PCTCOVER[i]) + bf \* (FORESTAGE[i]) + bc \* (COVER[i]) + bpc \* COVER[i] \* PCTCOVER[i];  
 log\_lik[i] = poisson\_log\_lpmf(SALAMAN[i] | lambda[i]);  
 }  
 }  
}  
'

Model 2.3: no age, only cover, cover rate and cover \* cover rate

m2.3='  
data {  
 int N;  
 int SALAMAN[N];   
 real PCTCOVER[N];   
 int COVER[N];  
}  
parameters {  
 real a;  
 real bp;  
 real bc;  
 real bpc;  
}  
model {  
 vector[N] lambda;  
 a ~ normal(0,100);  
 bp ~ normal(0,10);  
 bc ~ normal(0,10);  
 bpc ~ normal(0,10);  
  
 for(i in 1:N) lambda[i] = a + bp \* (PCTCOVER[i]) + bc \* (COVER[i]) + bpc \* COVER[i] \* PCTCOVER[i];  
 SALAMAN ~ poisson\_log(lambda);  
}  
  
generated quantities {  
 vector[N] log\_lik;  
 vector[N] lambda;  
 {  
 for(i in 1:N) {  
 lambda[i] = a + bp \* (PCTCOVER[i]) + bc \* (COVER[i]) + bpc \* COVER[i] \* PCTCOVER[i];  
 log\_lik[i] = poisson\_log\_lpmf(SALAMAN[i] | lambda[i]);  
 }  
 }  
}'

Model 2.5: cover, cover rate, cover \* cover rate,

m2.5='  
data {  
 int N;  
 int SALAMAN[N];   
 real PCTCOVER[N];   
 int FORESTAGE[N];  
 int COVER[N];  
}  
parameters {  
 real a;  
 real bp;  
 real bf;  
 real bc;  
 real bpc;  
 real bfc;  
 real bpf;  
}  
model {  
 vector[N] lambda;  
 a ~ normal(0,100);  
 bp ~ normal(0,10);  
 bf ~ normal(0,10);  
 bc ~ normal(0,10);  
 bpc ~ normal(0,10);  
 bfc ~ normal(0,10);  
 bpf ~ normal(0,10);  
  
 for(i in 1:N) lambda[i] = a + bp \* (PCTCOVER[i]) + bf \* (FORESTAGE[i]) + bc \* (COVER[i]) + bpc \* COVER[i] \* PCTCOVER[i] + bpf \* PCTCOVER[i] \* FORESTAGE[i] + bfc \* FORESTAGE[i] \* COVER[i];  
 SALAMAN ~ poisson\_log(lambda);  
}  
  
generated quantities {  
 vector[N] log\_lik;  
 vector[N] lambda;  
 {  
 for(i in 1:N) {  
 lambda[i] = a + bp \* (PCTCOVER[i]) + bf \* (FORESTAGE[i]) + bc \* (COVER[i]) + bpc \* COVER[i] \* PCTCOVER[i] + bpf \* PCTCOVER[i] \* FORESTAGE[i] + bfc \* FORESTAGE[i] \* COVER[i];  
 log\_lik[i] = poisson\_log\_lpmf(SALAMAN[i] | lambda[i]);  
 }  
 }  
}  
'

Data2.1

dat2.1 = list(  
 N = nrow(d),  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 FORESTAGE = d$FORESTAGE  
)

Data2.2

dat2.2 = list(  
 N = nrow(d),  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 FORESTAGE = d$FORESTAGE,  
 COVER = ifelse(d$PCTCOVER>70,1,0)  
)

Data2.3

dat2.3 = list(  
 N = nrow(d),  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 COVER = ifelse(d$PCTCOVER>70,1,0)  
)

Data2.4 Don’t take log on PCTCOVER

dat2.4 = list(  
 N = nrow(d),  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER,  
 COVER = ifelse(d$PCTCOVER>70,1,0)  
)

Data2.5

dat2.5 = list(  
 N = nrow(d),  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 FORESTAGE = d$FORESTAGE,  
 COVER = ifelse(d$PCTCOVER>70,1,0)  
)

Model fitting

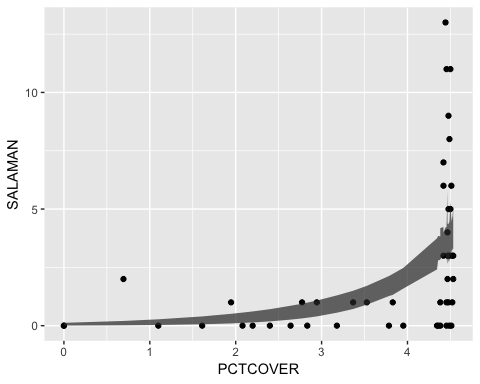
fit2.1 = stan(model\_code = m2.1, data = dat2.1, cores = 4, iter = 4000)  
fit2.2 = stan(model\_code = m2.2, data = dat2.2, cores = 4, iter = 4000)  
fit2.3 = stan(model\_code = m2.3, data = dat2.3, cores = 4, iter = 4000)  
fit2.4 = stan(model\_code = m2.3, data = dat2.4, cores = 4, iter = 4000)  
fit2.5 = stan(model\_code = m2.5, data = dat2.5, cores = 4, iter = 4000)

Prediction of model 2.1

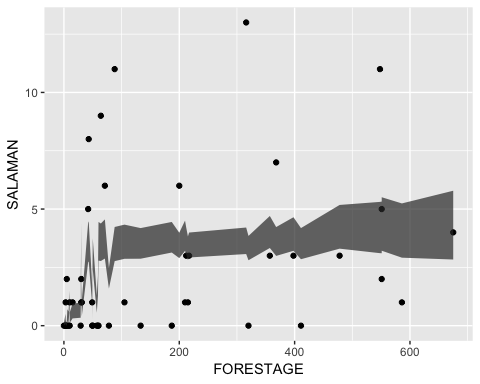
print(fit2.1, include = F, pars = 'log\_lik', probs = c(.1, .5, .9))

## Inference for Stan model: 5051948fbe23a792d97e429f2af4ae76.  
## 4 chains, each with iter=4000; warmup=2000; thin=1;   
## post-warmup draws per chain=2000, total post-warmup draws=8000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a -3.57 0.03 0.96 -4.79 -3.53 -2.40 765 1  
## bp 1.07 0.01 0.23 0.80 1.06 1.37 783 1  
## bf 0.00 0.00 0.00 0.00 0.00 0.00 3994 1  
## lambda[1] 1.28 0.00 0.10 1.16 1.28 1.41 7462 1  
## lambda[2] 1.23 0.00 0.13 1.06 1.24 1.40 3456 1  
## lambda[3] 1.40 0.00 0.17 1.19 1.41 1.62 6491 1  
## lambda[4] 1.25 0.00 0.14 1.07 1.26 1.43 3158 1  
## lambda[5] 1.26 0.00 0.15 1.07 1.26 1.45 3039 1  
## lambda[6] 1.27 0.00 0.11 1.13 1.27 1.41 7706 1  
## lambda[7] 1.23 0.00 0.10 1.10 1.23 1.35 5400 1  
## lambda[8] 1.29 0.00 0.14 1.11 1.29 1.47 2899 1  
## lambda[9] 1.25 0.00 0.15 1.05 1.25 1.43 3124 1  
## lambda[10] 1.40 0.00 0.17 1.19 1.41 1.62 6471 1  
## lambda[11] 1.40 0.00 0.22 1.11 1.40 1.68 5556 1  
## lambda[12] 1.23 0.00 0.10 1.11 1.23 1.36 5755 1  
## lambda[13] 1.28 0.00 0.10 1.15 1.28 1.41 4633 1  
## lambda[14] 1.35 0.00 0.12 1.20 1.35 1.50 7306 1  
## lambda[15] 1.38 0.00 0.11 1.24 1.38 1.52 6288 1  
## lambda[16] 1.42 0.00 0.14 1.24 1.42 1.60 6814 1  
## lambda[17] -2.83 0.03 0.81 -3.85 -2.79 -1.85 765 1  
## lambda[18] 1.23 0.00 0.15 1.03 1.24 1.42 3206 1  
## lambda[19] 1.44 0.00 0.17 1.22 1.44 1.65 6525 1  
## lambda[20] -1.48 0.02 0.53 -2.15 -1.46 -0.83 774 1  
## lambda[21] -0.59 0.01 0.35 -1.04 -0.58 -0.16 809 1  
## lambda[22] -0.40 0.01 0.32 -0.80 -0.39 -0.01 820 1  
## lambda[23] 0.05 0.01 0.24 -0.25 0.06 0.34 957 1  
## lambda[24] 0.23 0.01 0.20 -0.03 0.23 0.48 995 1  
## lambda[25] 0.55 0.00 0.16 0.34 0.55 0.75 1616 1  
## lambda[26] 1.19 0.00 0.10 1.07 1.19 1.31 6377 1  
## lambda[27] 1.36 0.00 0.18 1.13 1.37 1.60 5959 1  
## lambda[28] 1.26 0.00 0.13 1.10 1.27 1.42 3272 1  
## lambda[29] 1.34 0.00 0.11 1.20 1.34 1.47 3482 1  
## lambda[30] -3.57 0.03 0.96 -4.79 -3.53 -2.40 765 1  
## lambda[31] -3.57 0.03 0.96 -4.79 -3.53 -2.40 765 1  
## lambda[32] -2.39 0.03 0.72 -3.30 -2.36 -1.52 766 1  
## lambda[33] -1.84 0.02 0.60 -2.61 -1.82 -1.11 769 1  
## lambda[34] -1.34 0.02 0.50 -1.97 -1.31 -0.72 776 1  
## lambda[35] -1.21 0.02 0.47 -1.81 -1.19 -0.63 779 1  
## lambda[36] -1.00 0.02 0.43 -1.54 -0.98 -0.46 787 1  
## lambda[37] -0.72 0.01 0.38 -1.21 -0.71 -0.25 787 1  
## lambda[38] -0.52 0.01 0.34 -0.95 -0.51 -0.10 808 1  
## lambda[39] -0.14 0.01 0.27 -0.49 -0.13 0.19 843 1  
## lambda[40] 0.51 0.00 0.16 0.30 0.51 0.71 1327 1  
## lambda[41] 0.69 0.00 0.14 0.51 0.69 0.86 1845 1  
## lambda[42] 1.11 0.00 0.14 0.93 1.11 1.28 4545 1  
## lambda[43] 1.19 0.00 0.10 1.07 1.19 1.32 7192 1  
## lambda[44] 1.24 0.00 0.12 1.09 1.24 1.40 6732 1  
## lambda[45] 1.25 0.00 0.12 1.09 1.25 1.39 3718 1  
## lambda[46] 1.26 0.00 0.14 1.08 1.27 1.44 3059 1  
## lambda[47] 1.32 0.00 0.11 1.18 1.32 1.46 3397 1  
## lp\_\_ 17.12 0.03 1.26 15.46 17.48 18.33 1504 1  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 21:52:19 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).

post2.1 <- as.data.frame(fit2.1)  
  
for( i in (1:nrow(d))){  
 if(i == 1){  
 pred2 = tibble(mean = exp(post2.1[,i+50]) %>% mean,  
 l\_PI = exp(post2.1[,i+50]) %>% PI%>% .[1],  
 h\_PI = exp(post2.1[,i+50]) %>% PI%>% .[2])  
 }   
 else{  
 pred2 = tibble(mean = exp(post2.1[,i+50]) %>% mean,  
 l\_PI = exp(post2.1[,i+50]) %>% PI%>% .[1],  
 h\_PI = exp(post2.1[,i+50]) %>% PI%>% .[2]) %>%   
 rbind(pred2, .)  
 }  
}  
pred2 = pred2 %>% mutate(  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 FORESTAGE = d$FORESTAGE  
)  
# Plot PCTCOVER  
fig2.1.1 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=PCTCOVER, y=SALAMAN)) +   
 geom\_ribbon(aes(x=PCTCOVER,  
 ymin = l\_PI,  
 ymax = h\_PI),  
 alpha = 0.7)  
fig2.1.1



# Plot FORESTAGE  
fig2.1.2 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=FORESTAGE, y=SALAMAN)) +   
 geom\_ribbon(aes(x=FORESTAGE,  
 ymin = l\_PI,  
 ymax = h\_PI),  
 alpha = 0.7)  
fig2.1.2

 Prediction of model 2.2

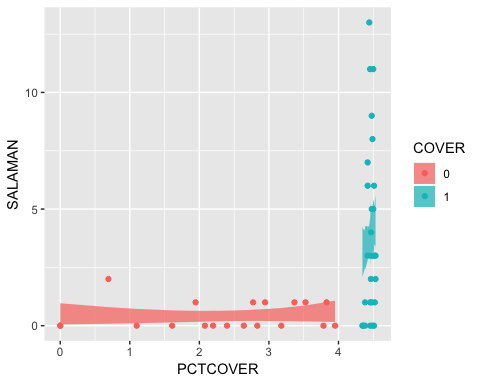
print(fit2.2, include = F, pars = 'log\_lik', probs = c(.1, .5, .9))

## Inference for Stan model: b2be0988f40a2bf9fc67058b7a364cdb.  
## 4 chains, each with iter=4000; warmup=2000; thin=1;   
## post-warmup draws per chain=2000, total post-warmup draws=8000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a -1.34 0.02 0.93 -2.56 -1.23 -0.25 1757 1  
## bp 0.14 0.01 0.33 -0.27 0.12 0.57 1832 1  
## bf 0.00 0.00 0.00 0.00 0.00 0.00 7287 1  
## bc -6.69 0.14 6.55 -15.14 -6.57 1.67 2043 1  
## bpc 1.96 0.03 1.49 0.05 1.94 3.88 1967 1  
## lambda[1] 1.29 0.00 0.11 1.15 1.29 1.43 6210 1  
## lambda[2] 1.33 0.00 0.13 1.15 1.33 1.49 8447 1  
## lambda[3] 1.40 0.00 0.17 1.18 1.41 1.62 6842 1  
## lambda[4] 1.37 0.00 0.14 1.18 1.38 1.56 7337 1  
## lambda[5] 1.40 0.00 0.16 1.19 1.40 1.60 6554 1  
## lambda[6] 1.24 0.00 0.13 1.07 1.24 1.41 4343 1  
## lambda[7] 1.25 0.00 0.12 1.09 1.25 1.40 4719 1  
## lambda[8] 1.44 0.00 0.16 1.24 1.45 1.65 5003 1  
## lambda[9] 1.38 0.00 0.15 1.18 1.38 1.57 7314 1  
## lambda[10] 1.40 0.00 0.17 1.18 1.41 1.62 6853 1  
## lambda[11] 1.32 0.00 0.23 1.03 1.33 1.61 7092 1  
## lambda[12] 1.25 0.00 0.12 1.09 1.25 1.40 4604 1  
## lambda[13] 1.34 0.00 0.10 1.21 1.35 1.47 8549 1  
## lambda[14] 1.38 0.00 0.12 1.23 1.39 1.54 6925 1  
## lambda[15] 1.46 0.00 0.13 1.29 1.46 1.62 4101 1  
## lambda[16] 1.47 0.00 0.16 1.26 1.48 1.68 4500 1  
## lambda[17] -1.24 0.02 0.72 -2.20 -1.16 -0.40 1829 1  
## lambda[18] 1.35 0.00 0.16 1.15 1.36 1.55 8008 1  
## lambda[19] 1.47 0.00 0.19 1.23 1.48 1.71 5105 1  
## lambda[20] -1.07 0.01 0.43 -1.63 -1.04 -0.56 2995 1  
## lambda[21] -0.96 0.01 0.38 -1.45 -0.94 -0.49 5642 1  
## lambda[22] -0.94 0.01 0.40 -1.45 -0.91 -0.45 5629 1  
## lambda[23] -0.88 0.01 0.46 -1.49 -0.84 -0.32 4772 1  
## lambda[24] -0.86 0.01 0.49 -1.51 -0.82 -0.26 4365 1  
## lambda[25] -0.81 0.01 0.56 -1.57 -0.77 -0.13 3713 1  
## lambda[26] 1.17 0.00 0.16 0.97 1.17 1.37 3223 1  
## lambda[27] 1.30 0.00 0.19 1.06 1.31 1.54 6870 1  
## lambda[28] 1.37 0.00 0.13 1.20 1.38 1.54 7397 1  
## lambda[29] 1.46 0.00 0.13 1.29 1.46 1.63 3827 1  
## lambda[30] -1.34 0.02 0.93 -2.56 -1.23 -0.25 1757 1  
## lambda[31] -1.34 0.02 0.93 -2.56 -1.23 -0.25 1757 1  
## lambda[32] -1.19 0.01 0.61 -2.00 -1.12 -0.47 1934 1  
## lambda[33] -1.12 0.01 0.49 -1.76 -1.07 -0.54 2339 1  
## lambda[34] -1.05 0.01 0.41 -1.59 -1.03 -0.56 3329 1  
## lambda[35] -1.04 0.01 0.40 -1.55 -1.01 -0.55 3719 1  
## lambda[36] -1.01 0.01 0.38 -1.50 -0.98 -0.54 4534 1  
## lambda[37] -0.98 0.01 0.38 -1.47 -0.96 -0.51 5357 1  
## lambda[38] -0.95 0.01 0.39 -1.45 -0.93 -0.48 5678 1  
## lambda[39] -0.90 0.01 0.43 -1.46 -0.87 -0.38 5236 1  
## lambda[40] -0.82 0.01 0.55 -1.56 -0.78 -0.15 3792 1  
## lambda[41] -0.80 0.01 0.60 -1.59 -0.75 -0.08 3519 1  
## lambda[42] 1.09 0.00 0.22 0.81 1.10 1.38 3280 1  
## lambda[43] 1.11 0.00 0.19 0.86 1.12 1.36 2782 1  
## lambda[44] 1.16 0.00 0.18 0.93 1.16 1.39 3295 1  
## lambda[45] 1.32 0.00 0.12 1.17 1.33 1.47 8490 1  
## lambda[46] 1.40 0.00 0.15 1.20 1.40 1.59 6528 1  
## lambda[47] 1.44 0.00 0.13 1.28 1.44 1.60 4469 1  
## lp\_\_ 21.34 0.03 1.61 19.18 21.68 23.06 2204 1  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 21:53:55 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).

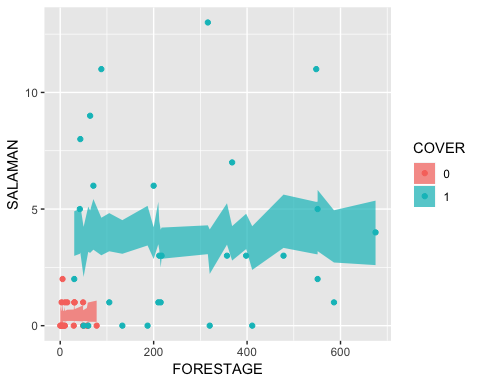
post2.2 <- as.data.frame(fit2.2)  
colnames(post2.2)

## [1] "a" "bp" "bf" "bc" "bpc"   
## [6] "log\_lik[1]" "log\_lik[2]" "log\_lik[3]" "log\_lik[4]" "log\_lik[5]"   
## [11] "log\_lik[6]" "log\_lik[7]" "log\_lik[8]" "log\_lik[9]" "log\_lik[10]"  
## [16] "log\_lik[11]" "log\_lik[12]" "log\_lik[13]" "log\_lik[14]" "log\_lik[15]"  
## [21] "log\_lik[16]" "log\_lik[17]" "log\_lik[18]" "log\_lik[19]" "log\_lik[20]"  
## [26] "log\_lik[21]" "log\_lik[22]" "log\_lik[23]" "log\_lik[24]" "log\_lik[25]"  
## [31] "log\_lik[26]" "log\_lik[27]" "log\_lik[28]" "log\_lik[29]" "log\_lik[30]"  
## [36] "log\_lik[31]" "log\_lik[32]" "log\_lik[33]" "log\_lik[34]" "log\_lik[35]"  
## [41] "log\_lik[36]" "log\_lik[37]" "log\_lik[38]" "log\_lik[39]" "log\_lik[40]"  
## [46] "log\_lik[41]" "log\_lik[42]" "log\_lik[43]" "log\_lik[44]" "log\_lik[45]"  
## [51] "log\_lik[46]" "log\_lik[47]" "lambda[1]" "lambda[2]" "lambda[3]"   
## [56] "lambda[4]" "lambda[5]" "lambda[6]" "lambda[7]" "lambda[8]"   
## [61] "lambda[9]" "lambda[10]" "lambda[11]" "lambda[12]" "lambda[13]"   
## [66] "lambda[14]" "lambda[15]" "lambda[16]" "lambda[17]" "lambda[18]"   
## [71] "lambda[19]" "lambda[20]" "lambda[21]" "lambda[22]" "lambda[23]"   
## [76] "lambda[24]" "lambda[25]" "lambda[26]" "lambda[27]" "lambda[28]"   
## [81] "lambda[29]" "lambda[30]" "lambda[31]" "lambda[32]" "lambda[33]"   
## [86] "lambda[34]" "lambda[35]" "lambda[36]" "lambda[37]" "lambda[38]"   
## [91] "lambda[39]" "lambda[40]" "lambda[41]" "lambda[42]" "lambda[43]"   
## [96] "lambda[44]" "lambda[45]" "lambda[46]" "lambda[47]" "lp\_\_"

for( i in (1:nrow(d))){  
 if(i == 1){  
 pred2 = tibble(mean = exp(post2.2[,i+52]) %>% mean,  
 l\_PI = exp(post2.2[,i+52]) %>% PI%>% .[1],  
 h\_PI = exp(post2.2[,i+52]) %>% PI%>% .[2])  
 }   
 else{  
 pred2 = tibble(mean = exp(post2.2[,i+52]) %>% mean,  
 l\_PI = exp(post2.2[,i+52]) %>% PI%>% .[1],  
 h\_PI = exp(post2.2[,i+52]) %>% PI%>% .[2]) %>%   
 rbind(pred2, .)  
 }  
}  
pred2 = pred2 %>% mutate(  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 FORESTAGE = d$FORESTAGE,  
 COVER = as.factor(ifelse(d$PCTCOVER>70,1,0))  
)  
# Plot PCTCOVER  
fig2.2.1 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=PCTCOVER, y=SALAMAN, color = COVER)) +   
 geom\_ribbon(aes(x=PCTCOVER,  
 ymin = l\_PI,  
 ymax = h\_PI,  
 group = COVER,fill = COVER),  
 alpha = 0.7)  
fig2.2.1



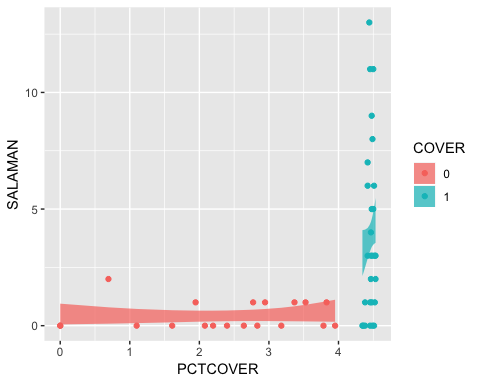
# Plot FORESTAGE  
fig2.2.2 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=FORESTAGE, y=SALAMAN, color = COVER)) +   
 geom\_ribbon(aes(x=FORESTAGE,  
 ymin = l\_PI,  
 ymax = h\_PI,  
 group = COVER, fill=COVER),  
 alpha = 0.7)  
fig2.2.2

 Prediction of model 2.3

print(fit2.3, include = F, pars = 'log\_lik', probs = c(.1, .5, .9))

## Inference for Stan model: bafae7f14b58dab126ec9bf9f6cd90fd.  
## 4 chains, each with iter=4000; warmup=2000; thin=1;   
## post-warmup draws per chain=2000, total post-warmup draws=8000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a -1.33 0.02 0.90 -2.49 -1.25 -0.26 2474 1  
## bp 0.14 0.01 0.32 -0.26 0.12 0.55 2448 1  
## bc -6.65 0.13 6.59 -14.98 -6.63 1.74 2520 1  
## bpc 1.95 0.03 1.49 0.03 1.95 3.83 2476 1  
## lambda[1] 1.30 0.00 0.10 1.16 1.30 1.43 6304 1  
## lambda[2] 1.32 0.00 0.10 1.20 1.32 1.45 7334 1  
## lambda[3] 1.42 0.00 0.11 1.28 1.42 1.55 5660 1  
## lambda[4] 1.37 0.00 0.10 1.25 1.37 1.49 7765 1  
## lambda[5] 1.39 0.00 0.10 1.26 1.39 1.52 6921 1  
## lambda[6] 1.25 0.00 0.12 1.09 1.25 1.40 4654 1  
## lambda[7] 1.25 0.00 0.12 1.09 1.25 1.40 4654 1  
## lambda[8] 1.44 0.00 0.12 1.29 1.44 1.59 4694 1  
## lambda[9] 1.37 0.00 0.10 1.25 1.37 1.49 7765 1  
## lambda[10] 1.42 0.00 0.11 1.28 1.42 1.55 5660 1  
## lambda[11] 1.35 0.00 0.10 1.22 1.35 1.47 7972 1  
## lambda[12] 1.25 0.00 0.12 1.09 1.25 1.40 4654 1  
## lambda[13] 1.35 0.00 0.10 1.22 1.35 1.47 7972 1  
## lambda[14] 1.39 0.00 0.10 1.26 1.39 1.52 6921 1  
## lambda[15] 1.46 0.00 0.13 1.30 1.46 1.62 4147 1  
## lambda[16] 1.48 0.00 0.14 1.31 1.49 1.66 3773 1  
## lambda[17] -1.24 0.01 0.70 -2.16 -1.17 -0.40 2595 1  
## lambda[18] 1.35 0.00 0.10 1.22 1.35 1.47 7972 1  
## lambda[19] 1.48 0.00 0.14 1.31 1.49 1.66 3773 1  
## lambda[20] -1.07 0.01 0.43 -1.63 -1.03 -0.55 3792 1  
## lambda[21] -0.96 0.01 0.39 -1.47 -0.93 -0.48 5919 1  
## lambda[22] -0.93 0.01 0.40 -1.47 -0.90 -0.44 5879 1  
## lambda[23] -0.88 0.01 0.46 -1.49 -0.84 -0.31 5140 1  
## lambda[24] -0.85 0.01 0.49 -1.51 -0.82 -0.25 4655 1  
## lambda[25] -0.81 0.01 0.56 -1.55 -0.78 -0.11 3997 1  
## lambda[26] 1.17 0.00 0.16 0.97 1.17 1.37 3498 1  
## lambda[27] 1.32 0.00 0.10 1.20 1.32 1.45 7334 1  
## lambda[28] 1.37 0.00 0.10 1.25 1.37 1.49 7765 1  
## lambda[29] 1.46 0.00 0.13 1.30 1.46 1.62 4147 1  
## lambda[30] -1.33 0.02 0.90 -2.49 -1.25 -0.26 2474 1  
## lambda[31] -1.33 0.02 0.90 -2.49 -1.25 -0.26 2474 1  
## lambda[32] -1.18 0.01 0.60 -1.95 -1.13 -0.46 2746 1  
## lambda[33] -1.11 0.01 0.49 -1.75 -1.07 -0.53 3179 1  
## lambda[34] -1.05 0.01 0.41 -1.59 -1.01 -0.55 4174 1  
## lambda[35] -1.03 0.01 0.40 -1.56 -1.00 -0.54 4573 1  
## lambda[36] -1.01 0.01 0.39 -1.52 -0.98 -0.54 5192 1  
## lambda[37] -0.97 0.01 0.38 -1.48 -0.94 -0.51 5771 1  
## lambda[38] -0.95 0.01 0.39 -1.47 -0.92 -0.47 5933 1  
## lambda[39] -0.90 0.01 0.43 -1.47 -0.87 -0.37 5522 1  
## lambda[40] -0.82 0.01 0.55 -1.54 -0.78 -0.13 4070 1  
## lambda[41] -0.80 0.01 0.59 -1.57 -0.76 -0.06 3829 1  
## lambda[42] 1.09 0.00 0.21 0.83 1.09 1.35 3071 1  
## lambda[43] 1.12 0.00 0.19 0.88 1.12 1.35 3173 1  
## lambda[44] 1.17 0.00 0.16 0.97 1.17 1.37 3498 1  
## lambda[45] 1.32 0.00 0.10 1.20 1.32 1.45 7334 1  
## lambda[46] 1.39 0.00 0.10 1.26 1.39 1.52 6921 1  
## lambda[47] 1.44 0.00 0.12 1.29 1.44 1.59 4694 1  
## lp\_\_ 21.85 0.03 1.44 19.89 22.19 23.34 2248 1  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 21:55:14 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).

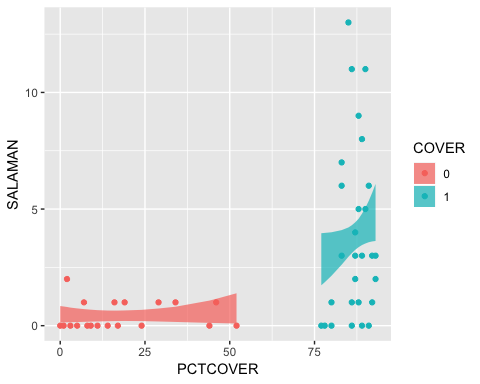
post2.3 <- as.data.frame(fit2.3)  
  
for( i in (1:nrow(d))){  
 if(i == 1){  
 pred2 = tibble(mean = exp(post2.3[,i+51]) %>% mean,  
 l\_PI = exp(post2.3[,i+51]) %>% PI%>% .[1],  
 h\_PI = exp(post2.3[,i+51]) %>% PI%>% .[2])  
 }   
 else{  
 pred2 = tibble(mean = exp(post2.3[,i+51]) %>% mean,  
 l\_PI = exp(post2.3[,i+51]) %>% PI%>% .[1],  
 h\_PI = exp(post2.3[,i+51]) %>% PI%>% .[2]) %>%   
 rbind(pred2, .)  
 }  
}  
pred2 = pred2 %>% mutate(  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 FORESTAGE = d$FORESTAGE,  
 COVER = as.factor(ifelse(d$PCTCOVER>70,1,0))  
)  
# Plot PCTCOVER  
fig2.3.1 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=PCTCOVER, y=SALAMAN, color = COVER)) +   
 geom\_ribbon(aes(x=PCTCOVER,  
 ymin = l\_PI,  
 ymax = h\_PI,  
 group = COVER,fill = COVER),  
 alpha = 0.7)  
fig2.3.1

 Prediction of model 2.4

print(fit2.4, include = F, pars = 'log\_lik', probs = c(.1, .5, .9))

## Inference for Stan model: bafae7f14b58dab126ec9bf9f6cd90fd.  
## 4 chains, each with iter=4000; warmup=2000; thin=1;   
## post-warmup draws per chain=2000, total post-warmup draws=8000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a -1.03 0.01 0.57 -1.79 -1.00 -0.33 2552 1  
## bp 0.00 0.00 0.02 -0.03 0.00 0.03 2286 1  
## bc -0.80 0.04 2.09 -3.48 -0.77 1.86 2510 1  
## bpc 0.03 0.00 0.03 -0.01 0.03 0.08 1949 1  
## lambda[1] 1.26 0.00 0.11 1.12 1.26 1.41 6029 1  
## lambda[2] 1.30 0.00 0.10 1.17 1.30 1.43 7726 1  
## lambda[3] 1.44 0.00 0.11 1.30 1.45 1.59 6004 1  
## lambda[4] 1.37 0.00 0.10 1.24 1.37 1.50 9049 1  
## lambda[5] 1.41 0.00 0.10 1.27 1.41 1.54 7648 1  
## lambda[6] 1.19 0.00 0.14 1.01 1.19 1.37 4291 1  
## lambda[7] 1.19 0.00 0.14 1.01 1.19 1.37 4291 1  
## lambda[8] 1.48 0.00 0.13 1.31 1.48 1.64 4936 1  
## lambda[9] 1.37 0.00 0.10 1.24 1.37 1.50 9049 1  
## lambda[10] 1.44 0.00 0.11 1.30 1.45 1.59 6004 1  
## lambda[11] 1.33 0.00 0.10 1.21 1.34 1.46 9011 1  
## lambda[12] 1.19 0.00 0.14 1.01 1.19 1.37 4291 1  
## lambda[13] 1.33 0.00 0.10 1.21 1.34 1.46 9011 1  
## lambda[14] 1.41 0.00 0.10 1.27 1.41 1.54 7648 1  
## lambda[15] 1.52 0.00 0.14 1.33 1.52 1.70 4295 1  
## lambda[16] 1.55 0.00 0.16 1.34 1.56 1.76 3900 1  
## lambda[17] -1.03 0.01 0.53 -1.74 -0.99 -0.37 2637 1  
## lambda[18] 1.33 0.00 0.10 1.21 1.34 1.46 9011 1  
## lambda[19] 1.55 0.00 0.16 1.34 1.56 1.76 3900 1  
## lambda[20] -1.01 0.01 0.46 -1.63 -0.98 -0.46 2962 1  
## lambda[21] -0.99 0.01 0.38 -1.49 -0.96 -0.53 4427 1  
## lambda[22] -0.98 0.01 0.37 -1.48 -0.96 -0.53 4931 1  
## lambda[23] -0.96 0.01 0.45 -1.55 -0.92 -0.42 4443 1  
## lambda[24] -0.94 0.01 0.52 -1.64 -0.89 -0.32 3832 1  
## lambda[25] -0.91 0.01 0.74 -1.88 -0.84 -0.04 3044 1  
## lambda[26] 1.08 0.00 0.20 0.83 1.08 1.33 3463 1  
## lambda[27] 1.30 0.00 0.10 1.17 1.30 1.43 7726 1  
## lambda[28] 1.37 0.00 0.10 1.24 1.37 1.50 9049 1  
## lambda[29] 1.52 0.00 0.14 1.33 1.52 1.70 4295 1  
## lambda[30] -1.03 0.01 0.57 -1.79 -1.00 -0.33 2552 1  
## lambda[31] -1.03 0.01 0.55 -1.77 -1.00 -0.35 2592 1  
## lambda[32] -1.02 0.01 0.52 -1.72 -0.99 -0.39 2683 1  
## lambda[33] -1.02 0.01 0.49 -1.67 -0.99 -0.43 2798 1  
## lambda[34] -1.01 0.01 0.44 -1.60 -0.98 -0.47 3081 1  
## lambda[35] -1.01 0.01 0.43 -1.58 -0.97 -0.48 3229 1  
## lambda[36] -1.00 0.01 0.41 -1.55 -0.97 -0.50 3519 1  
## lambda[37] -1.00 0.01 0.39 -1.51 -0.97 -0.52 4036 1  
## lambda[38] -0.99 0.01 0.38 -1.48 -0.96 -0.53 4618 1  
## lambda[39] -0.97 0.01 0.40 -1.49 -0.94 -0.49 5060 1  
## lambda[40] -0.92 0.01 0.70 -1.83 -0.85 -0.09 3127 1  
## lambda[41] -0.90 0.02 0.87 -2.02 -0.81 0.13 2862 1  
## lambda[42] 0.97 0.00 0.26 0.64 0.97 1.30 3191 1  
## lambda[43] 1.01 0.00 0.24 0.70 1.01 1.31 3255 1  
## lambda[44] 1.08 0.00 0.20 0.83 1.08 1.33 3463 1  
## lambda[45] 1.30 0.00 0.10 1.17 1.30 1.43 7726 1  
## lambda[46] 1.41 0.00 0.10 1.27 1.41 1.54 7648 1  
## lambda[47] 1.48 0.00 0.13 1.31 1.48 1.64 4936 1  
## lp\_\_ 22.10 0.03 1.46 20.18 22.44 23.60 2421 1  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 21:55:25 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).

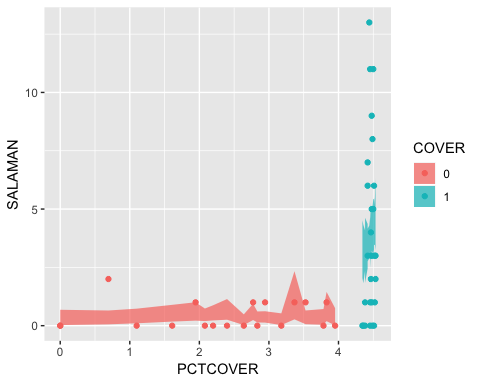
post2.4 <- as.data.frame(fit2.4)  
  
for( i in (1:nrow(d))){  
 if(i == 1){  
 pred2 = tibble(mean = exp(post2.4[,i+51]) %>% mean,  
 l\_PI = exp(post2.4[,i+51]) %>% PI%>% .[1],  
 h\_PI = exp(post2.4[,i+51]) %>% PI%>% .[2])  
 }   
 else{  
 pred2 = tibble(mean = exp(post2.4[,i+51]) %>% mean,  
 l\_PI = exp(post2.4[,i+51]) %>% PI%>% .[1],  
 h\_PI = exp(post2.4[,i+51]) %>% PI%>% .[2]) %>%   
 rbind(pred2, .)  
 }  
}  
pred2 = pred2 %>% mutate(  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER,  
 FORESTAGE = d$FORESTAGE,  
 COVER = as.factor(ifelse(d$PCTCOVER>70,1,0))  
)  
# Plot PCTCOVER  
fig2.4.1 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=PCTCOVER, y=SALAMAN, color = COVER)) +   
 geom\_ribbon(aes(x=PCTCOVER,  
 ymin = l\_PI,  
 ymax = h\_PI,  
 group = COVER,fill = COVER),  
 alpha = 0.7)  
fig2.4.1

 Prediction 2.5

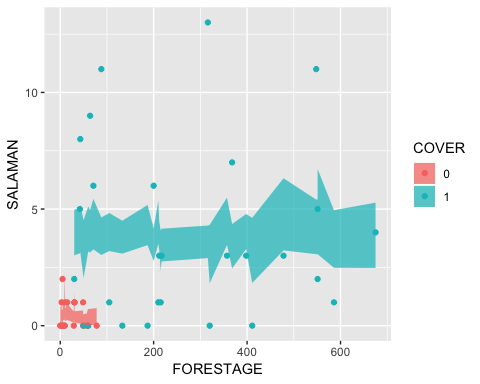
print(fit2.5, include = F, pars = 'log\_lik', probs = c(.1, .5, .9))

## Inference for Stan model: 4033843e9815942970b9057a3f1288ab.  
## 4 chains, each with iter=4000; warmup=2000; thin=1;   
## post-warmup draws per chain=2000, total post-warmup draws=8000.  
##   
## mean se\_mean sd 10% 50% 90% n\_eff Rhat  
## a -1.70 0.02 1.03 -3.09 -1.60 -0.47 3057 1  
## bp 0.57 0.01 0.48 -0.03 0.55 1.18 2627 1  
## bf -0.05 0.00 0.04 -0.10 -0.04 0.01 3077 1  
## bc -4.99 0.15 8.27 -15.59 -4.90 5.60 3113 1  
## bpc 1.24 0.03 1.88 -1.20 1.21 3.66 3022 1  
## bfc 0.03 0.00 0.03 0.00 0.03 0.07 3525 1  
## bpf 0.00 0.00 0.01 -0.01 0.00 0.01 3379 1  
## lambda[1] 1.26 0.00 0.12 1.11 1.27 1.42 6230 1  
## lambda[2] 1.33 0.00 0.13 1.15 1.33 1.49 7480 1  
## lambda[3] 1.41 0.00 0.18 1.18 1.42 1.64 7535 1  
## lambda[4] 1.37 0.00 0.14 1.19 1.38 1.55 7347 1  
## lambda[5] 1.40 0.00 0.16 1.20 1.40 1.60 6957 1  
## lambda[6] 1.19 0.00 0.18 0.95 1.20 1.41 5608 1  
## lambda[7] 1.23 0.00 0.13 1.06 1.23 1.39 6535 1  
## lambda[8] 1.44 0.00 0.16 1.24 1.44 1.64 6060 1  
## lambda[9] 1.38 0.00 0.15 1.18 1.38 1.57 7378 1  
## lambda[10] 1.41 0.00 0.18 1.18 1.42 1.64 7533 1  
## lambda[11] 1.29 0.00 0.24 0.98 1.30 1.59 7342 1  
## lambda[12] 1.22 0.00 0.13 1.06 1.23 1.39 6474 1  
## lambda[13] 1.34 0.00 0.10 1.21 1.34 1.47 7268 1  
## lambda[14] 1.38 0.00 0.12 1.23 1.39 1.53 7923 1  
## lambda[15] 1.48 0.00 0.14 1.29 1.48 1.66 6775 1  
## lambda[16] 1.52 0.00 0.21 1.24 1.52 1.78 5954 1  
## lambda[17] -1.53 0.01 0.78 -2.55 -1.44 -0.61 3189 1  
## lambda[18] 1.36 0.00 0.15 1.15 1.36 1.55 7631 1  
## lambda[19] 1.52 0.00 0.25 1.20 1.53 1.83 5714 1  
## lambda[20] -0.72 0.01 0.48 -1.35 -0.69 -0.13 5443 1  
## lambda[21] -0.70 0.01 0.43 -1.26 -0.66 -0.17 5553 1  
## lambda[22] -1.19 0.01 0.49 -1.83 -1.15 -0.60 4813 1  
## lambda[23] -0.15 0.01 0.68 -1.04 -0.11 0.68 3430 1  
## lambda[24] -1.45 0.01 0.75 -2.42 -1.35 -0.58 4712 1  
## lambda[25] -0.58 0.01 0.63 -1.40 -0.54 0.20 4311 1  
## lambda[26] 1.14 0.00 0.17 0.91 1.14 1.35 6114 1  
## lambda[27] 1.26 0.00 0.22 0.98 1.26 1.53 6477 1  
## lambda[28] 1.37 0.00 0.13 1.21 1.37 1.53 7292 1  
## lambda[29] 1.47 0.00 0.13 1.30 1.47 1.64 6503 1  
## lambda[30] -1.70 0.02 1.03 -3.09 -1.60 -0.47 3057 1  
## lambda[31] -1.88 0.02 1.05 -3.28 -1.78 -0.63 2915 1  
## lambda[32] -1.21 0.01 0.63 -2.04 -1.14 -0.46 3739 1  
## lambda[33] -0.87 0.01 0.52 -1.56 -0.83 -0.23 4868 1  
## lambda[34] -0.92 0.01 0.41 -1.47 -0.89 -0.41 5651 1  
## lambda[35] -0.77 0.01 0.43 -1.34 -0.74 -0.25 5993 1  
## lambda[36] -0.58 0.01 0.47 -1.19 -0.54 0.01 5370 1  
## lambda[37] -2.07 0.02 0.94 -3.31 -1.99 -0.96 3281 1  
## lambda[38] -1.18 0.01 0.47 -1.80 -1.15 -0.62 4736 1  
## lambda[39] -1.98 0.02 0.97 -3.28 -1.87 -0.86 3752 1  
## lambda[40] -1.61 0.01 0.94 -2.84 -1.49 -0.53 4698 1  
## lambda[41] -2.15 0.02 1.36 -3.92 -2.00 -0.54 4353 1  
## lambda[42] 1.12 0.00 0.25 0.80 1.12 1.43 4144 1  
## lambda[43] 1.03 0.00 0.27 0.69 1.04 1.38 5555 1  
## lambda[44] 1.07 0.00 0.29 0.70 1.08 1.44 5126 1  
## lambda[45] 1.32 0.00 0.12 1.16 1.32 1.47 7371 1  
## lambda[46] 1.40 0.00 0.15 1.21 1.40 1.59 6946 1  
## lambda[47] 1.44 0.00 0.13 1.28 1.44 1.60 6586 1  
## lp\_\_ 21.17 0.03 1.88 18.68 21.49 23.27 2952 1  
##   
## Samples were drawn using NUTS(diag\_e) at Wed May 22 21:58:00 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).

post2.5 <- as.data.frame(fit2.5)  
  
for( i in (1:nrow(d))){  
 if(i == 1){  
 pred2 = tibble(mean = exp(post2.5[,i+54]) %>% mean,  
 l\_PI = exp(post2.5[,i+54]) %>% PI%>% .[1],  
 h\_PI = exp(post2.5[,i+54]) %>% PI%>% .[2])  
 }   
 else{  
 pred2 = tibble(mean = exp(post2.5[,i+54]) %>% mean,  
 l\_PI = exp(post2.5[,i+54]) %>% PI%>% .[1],  
 h\_PI = exp(post2.5[,i+54]) %>% PI%>% .[2]) %>%   
 rbind(pred2, .)  
 }  
}  
pred2 = pred2 %>% mutate(  
 SALAMAN = d$SALAMAN,  
 PCTCOVER = d$PCTCOVER\_log,  
 FORESTAGE = d$FORESTAGE,  
 COVER = as.factor(ifelse(d$PCTCOVER>70,1,0))  
)  
# Plot PCTCOVER  
fig2.5.1 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=PCTCOVER, y=SALAMAN, color = COVER)) +   
 geom\_ribbon(aes(x=PCTCOVER,  
 ymin = l\_PI,  
 ymax = h\_PI,  
 group = COVER,fill = COVER),  
 alpha = 0.7)  
fig2.5.1



# Plot FORESTAGE  
fig2.5.2 = pred2 %>%   
 ggplot() +   
 geom\_point(aes(x=FORESTAGE, y=SALAMAN, color = COVER)) +   
 geom\_ribbon(aes(x=FORESTAGE,  
 ymin = l\_PI,  
 ymax = h\_PI,  
 group = COVER, fill=COVER),  
 alpha = 0.7)  
fig2.5.2

 # Model Comparison

fit\_list <- list(fit1.1, fit2.1, fit2.2, fit2.3, fit2.4, fit2.5)  
# extract log likelihoods  
ll\_list <- lapply(fit\_list, extract\_log\_lik)  
# exponentiate  
exp\_ll\_list <- lapply(ll\_list, exp)  
  
waic\_list <- list()   
for(i in 1:6) {  
waic\_list[[i]] <- waic(ll\_list[[i]], r\_eff = rel\_n\_eff\_list[[i]], cores = 4)  
}

## Warning: 4 (8.5%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

## Warning: 6 (12.8%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.  
  
## Warning: 6 (12.8%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

## Warning: 4 (8.5%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

## Warning: 6 (12.8%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

## Warning: 10 (21.3%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

names(waic\_list) <- c('fit1.1', 'fit2.1', 'fit2.2', 'fit2.3', 'fit2.4', 'fit2.5')  
loo::compare(x = waic\_list)

## elpd\_diff elpd\_waic se\_elpd\_waic p\_waic se\_p\_waic waic se\_waic  
## fit2.3 0.0 -108.2 13.2 7.4 1.8 216.5 26.5   
## fit2.4 -0.5 -108.7 13.4 8.1 1.5 217.5 26.8   
## fit2.2 -2.3 -110.5 13.7 10.5 2.4 221.1 27.4   
## fit2.5 -4.1 -112.3 14.2 13.1 2.8 224.6 28.5   
## fit1.1 -4.3 -112.5 14.6 6.5 2.5 225.0 29.2   
## fit2.1 -6.4 -114.6 14.9 9.6 2.9 229.2 29.7

Since model2.3 and model2.3 (include cover, cover % and interaction) are the 2 best models, we can tell that the covering percentage is more significant and we can tell that age of the forest isn’t an important variable.  
The impacts of covering percentage will be different when covering rate is above or below 70%.