Assign5

Ryan Huang

5/8/2019

# Question 1

# Load data and convert categorical variables to dummy variables  
data(eagles)  
d = eagles; rm(eagles)  
d$P\_dummy = ifelse(d$P == "L", 1, 0)  
d$A\_dummy = ifelse(d$A == "A", 1, 0)  
d$V\_dummy = ifelse(d$V == "L", 1, 0)

m1.1 = "  
data{  
 int N;  
 int Y[N];  
 int n[N];  
 vector[N] P;  
 vector[N] V;  
 vector[N] A;  
}  
parameters{  
 real alpha;  
 real bP;  
 real bV;  
 real bA;  
}  
model{  
 vector[N] p;  
 alpha ~ normal(0, 10);  
 bP ~ normal(0, 5);  
 bV ~ normal(0, 5);  
 bA ~ normal(0, 5);  
  
 for (i in 1:N){  
 p[i] = inv\_logit(alpha + bP \* P[i] + bV \* V[i] + bA \* A[i]);  
 }  
 Y ~ binomial(n,p);  
}  
generated quantities{  
 vector[N] log\_lik;  
 {  
 vector[N] p;  
 for (i in 1:N){  
 p[i] = inv\_logit(alpha + bP \* P[i] + bV \* V[i] + bA \* A[i]);  
 log\_lik[i] = binomial\_logit\_lpmf(Y[i] | n[i], p[i]);  
 }  
 }  
}"  
  
dat1.1 = list(  
 N = nrow(d),  
 Y = d$y,  
 n = d$n,  
 P = d$P\_dummy,  
 V = d$V\_dummy,  
 A = d$A\_dummy  
)

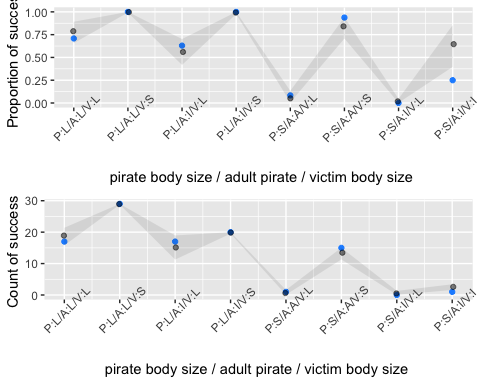
fit1.1 = stan(model\_code = m1.1, data = dat1.1, cores = 4)

print(fit1.1)

## Inference for Stan model: 5b519071d30c93b7c072d18088c53348.  
## 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 2.5% 25% 50% 75% 97.5% n\_eff  
## alpha 0.66 0.02 0.69 -0.62 0.21 0.65 1.10 2.05 1812  
## bP 4.66 0.02 0.99 3.01 3.96 4.55 5.24 6.89 1658  
## bV -5.08 0.03 1.08 -7.49 -5.74 -4.99 -4.34 -3.22 1572  
## bA 1.12 0.01 0.54 0.08 0.75 1.12 1.49 2.21 1897  
## log\_lik[1] -1.77 0.00 0.05 -1.91 -1.79 -1.75 -1.74 -1.73 2340  
## log\_lik[2] -9.11 0.00 0.02 -9.17 -9.11 -9.10 -9.09 -9.09 2308  
## log\_lik[3] -1.88 0.00 0.04 -1.98 -1.89 -1.86 -1.85 -1.85 2551  
## log\_lik[4] -6.31 0.00 0.05 -6.45 -6.32 -6.29 -6.28 -6.27 2129  
## log\_lik[5] -6.10 0.00 0.22 -6.70 -6.19 -6.04 -5.94 -5.85 2412  
## log\_lik[6] -3.81 0.01 0.30 -4.53 -3.98 -3.76 -3.59 -3.39 3050  
## log\_lik[7] -19.66 0.00 0.22 -20.25 -19.75 -19.59 -19.50 -19.43 2244  
## log\_lik[8] -2.25 0.01 0.23 -2.65 -2.41 -2.26 -2.09 -1.80 1844  
## lp\_\_ -48.12 0.04 1.46 -51.81 -48.81 -47.78 -47.06 -46.28 1370  
## Rhat  
## alpha 1  
## bP 1  
## bV 1  
## bA 1  
## log\_lik[1] 1  
## log\_lik[2] 1  
## log\_lik[3] 1  
## log\_lik[4] 1  
## log\_lik[5] 1  
## log\_lik[6] 1  
## log\_lik[7] 1  
## log\_lik[8] 1  
## lp\_\_ 1  
##   
## Samples were drawn using NUTS(diag\_e) at Sat May 11 15:09:10 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).

# Question b

# Plot the posterior predictions  
inv\_logit = function(x){  
 exp(x)/(1+exp(x))  
}  
  
post\_1.1 = as.data.frame(fit1.1)  
pred\_1.1 = function(P, V, A){  
 a = with(post\_1.1, alpha + bP \* P + bV \* V + bA \* A)  
 p = inv\_logit(a)  
 return(p)  
}  
  
mean1.1 = pmap(list(d$P\_dummy, d$V\_dummy, d$A\_dummy), pred\_1.1) %>%  
 purrr::map(mean) %>%   
 unlist()  
pi1.1 = pmap(list(d$P\_dummy, d$V\_dummy, d$A\_dummy), pred\_1.1) %>%  
 purrr::map(PI) %>%   
 unlist()  
  
test\_result1.1\_p =   
 tibble(  
 actual\_p = d$y / d$n,  
 test\_p\_PI\_l = pi1.1[seq(from = 1, to = length(pi1.1), by = 2)],  
 test\_p\_mean = mean1.1,  
 test\_p\_PI\_h = pi1.1[seq(from = 2, to = length(pi1.1), by = 2)],  
 index = 1:8  
 ) %>%   
 ggplot() +  
 geom\_point(aes(x=index, y=actual\_p), color = 'dodgerblue') +   
 geom\_ribbon(aes(x = index,  
 ymin = test\_p\_PI\_l,  
 ymax = test\_p\_PI\_h),  
 alpha = .1) +  
 geom\_jitter(aes(x=index, y=test\_p\_mean), alpha=0.5, width = 0.02, height = 0) +   
 lims(y = c(0, 1)) +  
 scale\_x\_continuous(labels = c("1" = "P:L/A:L/V:L", "2" = "P:L/A:L/V:S", "3" = "P:L/A:I/V:L", "4" = "P:L/A:I/V:S", "5" = "P:S/A:A/V:L", "6" = "P:S/A:A/V:S", "7" = "P:S/A:I/V:L", "8" = "P:S/A:I/V:I"), breaks = c(1:8)) +  
 labs(x = "pirate body size / adult pirate / victim body size ", y = "Proportion of success") +  
 theme(axis.text.x = element\_text(angle = 45))  
  
  
test\_result1.1\_y =   
 tibble(  
 actual\_y = d$y,  
 test\_y\_PI\_l = pi1.1[seq(from = 1, to = length(pi1.1), by = 2)] \* d$n,  
 test\_y\_mean = mean1.1 \* d$n,  
 test\_y\_PI\_h = pi1.1[seq(from = 2, to = length(pi1.1), by = 2)] \* d$n,  
 index = 1:8  
 ) %>%   
ggplot() +  
 geom\_point(aes(x=index, y=actual\_y), color = 'dodgerblue') +   
 geom\_ribbon(aes(x = index,  
 ymin = test\_y\_PI\_l,  
 ymax = test\_y\_PI\_h),  
 alpha = .1) +  
 geom\_jitter(aes(x=index, y=test\_y\_mean), alpha=0.5, width = 0.02, height = 0) +   
 scale\_x\_continuous(labels = c("1" = "P:L/A:L/V:L", "2" = "P:L/A:L/V:S", "3" = "P:L/A:I/V:L", "4" = "P:L/A:I/V:S", "5" = "P:S/A:A/V:L", "6" = "P:S/A:A/V:S", "7" = "P:S/A:I/V:L", "8" = "P:S/A:I/V:I"), breaks = c(1:8)) +  
 labs(x = "pirate body size / adult pirate / victim body size ", y = "Count of success ") +  
 theme(axis.text.x = element\_text(angle = 45))  
  
grid.arrange(test\_result1.1\_p, test\_result1.1\_y)

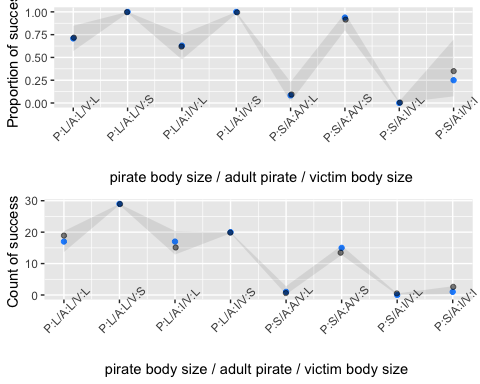
 Comparing the above 2 plots, we can tell that it’s more accurate when we are predicting the counts of success. The reason is that the Ns in each row of the training data are different. When we have a small N, a small change y will result in a significantly different proportion.

# Question c

m1.2 = "  
data{  
 int N;  
 int Y[N];  
 int n[N];  
 vector[N] P;  
 vector[N] V;  
 vector[N] A;  
}  
parameters{  
 real alpha;  
 real bP;  
 real bV;  
 real bA;  
 real bPA;  
}  
model{  
 vector[N] p;  
 alpha ~ normal(0, 10);  
 bP ~ normal(0, 5);  
 bV ~ normal(0, 5);  
 bA ~ normal(0, 5);  
 bPA ~ normal(0,5);  
  
 for (i in 1:N){  
 p[i] = inv\_logit(alpha + bP \* P[i] + bV \* V[i] + bA \* A[i] + bPA \* P[i] \* A[i]);  
 }  
 Y ~ binomial(n,p);  
}  
generated quantities{  
 vector[N] log\_lik;  
 {  
 vector[N] p;  
 for (i in 1:N){  
 p[i] = inv\_logit(alpha + bP \* P[i] + bV \* V[i] + bA \* A[i]+ bPA \* P[i] \* A[i]);  
 log\_lik[i] = binomial\_logit\_lpmf(Y[i] | n[i], p[i]);  
 }  
 }  
}"  
  
dat1.2 = dat1.1

fit1.2 = stan(model\_code = m1.2, data = dat1.2, cores = 4)

post\_1.2 = as.data.frame(fit1.2)  
pred\_1.2 = function(P, V, A){  
 a = with(post\_1.2, alpha + bP \* P + bV \* V + bA \* A + + bPA \* P \* A)  
 p = inv\_logit(a)  
 return(p)  
}  
  
mean1.2 = pmap(list(d$P\_dummy, d$V\_dummy, d$A\_dummy), pred\_1.2) %>%  
 purrr::map(mean) %>%   
 unlist()  
pi1.2 = pmap(list(d$P\_dummy, d$V\_dummy, d$A\_dummy), pred\_1.2) %>%  
 purrr::map(PI) %>%   
 unlist()  
  
test\_result1.2\_p =   
 tibble(  
 actual\_p = d$y / d$n,  
 test\_p\_PI\_l = pi1.2[seq(from = 1, to = length(pi1.1), by = 2)],  
 test\_p\_mean = mean1.2,  
 test\_p\_PI\_h = pi1.2[seq(from = 2, to = length(pi1.1), by = 2)],  
 index = 1:8  
 ) %>%   
 ggplot() +  
 geom\_point(aes(x=index, y=actual\_p), color = 'dodgerblue') +   
 geom\_ribbon(aes(x = index,  
 ymin = test\_p\_PI\_l,  
 ymax = test\_p\_PI\_h),  
 alpha = .1) +  
 geom\_jitter(aes(x=index, y=test\_p\_mean), alpha=0.5, width = 0.02, height = 0) +   
 lims(y = c(0, 1)) +  
 scale\_x\_continuous(labels = c("1" = "P:L/A:L/V:L", "2" = "P:L/A:L/V:S", "3" = "P:L/A:I/V:L", "4" = "P:L/A:I/V:S", "5" = "P:S/A:A/V:L", "6" = "P:S/A:A/V:S", "7" = "P:S/A:I/V:L", "8" = "P:S/A:I/V:I"), breaks = c(1:8)) +  
 labs(x = "pirate body size / adult pirate / victim body size ", y = "Proportion of success") +  
 theme(axis.text.x = element\_text(angle = 45))  
  
  
test\_result1.2\_y =   
 tibble(  
 actual\_y = d$y,  
 test\_y\_PI\_l = pi1.2[seq(from = 1, to = length(pi1.1), by = 2)] \* d$n,  
 test\_y\_mean = mean1.1 \* d$n,  
 test\_y\_PI\_h = pi1.2[seq(from = 2, to = length(pi1.1), by = 2)] \* d$n,  
 index = 1:8  
 ) %>%   
ggplot() +  
 geom\_point(aes(x=index, y=actual\_y), color = 'dodgerblue') +   
 geom\_ribbon(aes(x = index,  
 ymin = test\_y\_PI\_l,  
 ymax = test\_y\_PI\_h),  
 alpha = .1) +  
 geom\_jitter(aes(x=index, y=test\_y\_mean), alpha=0.5, width = 0.02, height = 0) +   
 scale\_x\_continuous(labels = c("1" = "P:L/A:L/V:L", "2" = "P:L/A:L/V:S", "3" = "P:L/A:I/V:L", "4" = "P:L/A:I/V:S", "5" = "P:S/A:A/V:L", "6" = "P:S/A:A/V:S", "7" = "P:S/A:I/V:L", "8" = "P:S/A:I/V:I"), breaks = c(1:8)) +  
 labs(x = "pirate body size / adult pirate / victim body size ", y = "Count of success ") +  
 theme(axis.text.x = element\_text(angle = 45))  
  
grid.arrange(test\_result1.2\_p, test\_result1.2\_y)

 Model Comparison

# extract log likelihood from the fitted model and use WAIC.  
log\_lik\_1.1 = extract\_log\_lik(fit1.1, merge\_chains = FALSE)  
log\_lik\_1.2 = extract\_log\_lik(fit1.2, merge\_chains = FALSE)  
  
(waic\_1.1 = waic(log\_lik\_1.1))

##   
## Computed from 4000 by 8 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_waic -51.0 16.9  
## p\_waic 0.2 0.1  
## waic 102.0 33.8

(waic\_1.2 = waic(log\_lik\_1.2))

##   
## Computed from 4000 by 8 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_waic -50.5 16.9  
## p\_waic 0.3 0.2  
## waic 100.9 33.7

mod\_comp <- loo::compare(waic\_1.1, waic\_1.2)  
mod\_comp

## elpd\_diff se   
## 0.6 0.6

From the above comparison, the 2 models have very similar waic.