

Bayesian Data Analysis Assignment 2

Benjamin Cox, S1621312

Question 1

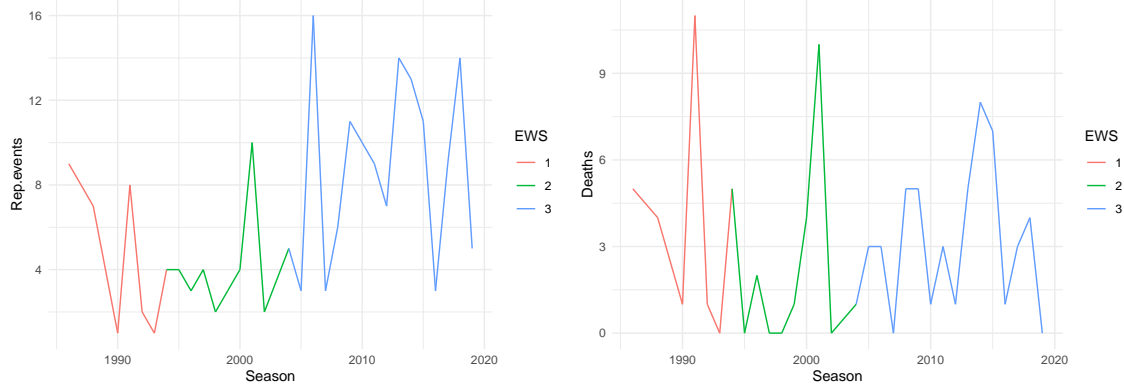


Figure 1: Plots illustrating the temporal evolution of avalanche related statistics. The EWS measure is 1 = No EADS, 2 = EADS present, 3 = EADS online daily.

From the above graphs we can see a positive trend in the number of avalanches and year, but no obvious trend in the number of deaths. We calculate the correlations between the number of deaths and the number of avalanches separated into EWS periods.

We obtain the following correlations (90% bootstrap intervals)

No EADS	EADS	EADS Online
0.807 (0.9325, 0.9986)	0.875 (0.1890, 0.9728)	0.602 (0.3842, 0.8147)

This shows that the events become less correlated after the general public obtained easy access to EADS. It is not likely that the introduction of EADS increased to correlation, so the observed increase in correlation for that period is likely due to noise (10 events in 2001 resulting in 10 deaths). However it may also be due to an increase in user confidence, which led to foolish behaviour.

We are now going to model the number of deaths in avalanches. We are using a Poisson model with a logarithmic (canonical) link function.

Our formulae are as follows:

$$\log(\lambda_i) = \beta_0 + \beta_1 \cdot \text{Rep.events}_i + \beta_2 \cdot \text{EADS1}_i + \beta_3 \cdot \text{EADS2}_i$$
$$\text{Deaths}_i \sim \text{Poisson}(\lambda_i)$$

We note that these parameters have a multiplicative effect on the rate, so it is fine to have an intercept on physical terms. We will remove the intercept later.

We place wide normal priors on all β_i and code up our model. The code is given in A.2, with a JAGS version given in A.3.

We run it and obtain the following posterior summaries. We have exponentiated our parameters prior to summarising to ease interpretation.

	(Intercept)	Rep.events	EADS1TRUE	EADS2TRUE
Min.	0.41	1.07	0.28	0.12
1st Qu.	1.08	1.17	0.66	0.32
Median	1.32	1.19	0.81	0.39
Mean	1.37	1.19	0.85	0.40
3rd Qu.	1.60	1.22	0.99	0.47
Max.	4.12	1.34	2.75	1.10

Table 1: Posterior summaries for the first Poisson model

From this we can make some initial conclusions. We see that the expected number of deaths increases by 1.19 times per avalanche (all other variables held constant). We also see that each EADS evolution decreases the expected number of deaths, by 0.85 and 0.40 times respectively. The latter is a rather large decrease, befitting of the drastic change in preparation tact that the EADS going online brought about.

We are interested in the posterior predictive distribution. We want to predict the probability of observing less than 15 deaths given 20 avalanches next year. We know that EADS will be online, so we have the appropriate data.

We obtain a probability of $P(D < 15|A = 20, EADS = 2) = 0.312$. This is roughly expected, as the number of avalanches increases deaths multiplicatively in expectation. In contrast the probability of less than 15 deaths for 10 avalanches is computationally indistinguishable from 1.

We are also interested in the probability of observing more than 1 death per avalanche in each stage of the EADS lifespan (not present, present, online). To do this we are going to have to make an assumption. We assume that the mean number of avalanches occurs.

A Code for Question 1

A.1 R

```
../Q1.R

1 library(data.table)
2 library(ggplot2)
3
4 library(rstan)
5 rstan_options(auto_write = TRUE)
6 #options(mc.cores = parallel::detectCores())
7 Sys.setenv(LOCAL_CPPFLAGS = '-march=corei7 -mtune=corei7')
8 options(mc.cores = 1)
9 library(rstanarm)
10 library(coda)
11 library(bayesplot)
12
13
14 #####
15 #a
16 avalanches <- fread(file = "data/Avalanches.csv")
17 avalanches <- avalanches[Rep.events > 0]
18 avalanches[, ']:= (EADS1 = (Season >= 1994 &
19                        Season <= 2003),
20                        EADS2 = (Season >= 2004))]]
21
22 avalanches[Season %in% c(1986, 1994, 2004)]
23
24 avalanches[, EWS := 1 + EADS1 + 2 * EADS2]
25 avalanches[, EWS := as.factor(EWS)]
26
27 base_plot <-
28   ggplot(data = as.data.frame(avalanches), aes(colour = EWS)) + theme_minimal()
29 base_plot + geom_line(aes(x = Season, y = Rep.events, group = F))
30 base_plot + geom_line(aes(x = Season, y = Deaths, group = F))
31 base_plot + geom_boxplot(aes(x = EWS, y = Deaths), colour = "black")
32
33
34 cor_boot <- function(data, index) {
```

```

35   dt_s <- data[index, ]
36   return(cor(dt_s))
37 }
38
39 cor(avalanches[(EADS1 == FALSE &
40               EADS2 == FALSE), .(Rep.events, Deaths)])
41 cor(avalanches[EADS1 == TRUE, .(Rep.events, Deaths)])
42 cor(avalanches[EADS2 == TRUE, .(Rep.events, Deaths)])
43
44 bs1 <- boot::boot(avalanches[(EADS1 == FALSE &
45                             EADS2 == FALSE),
46                       .(Rep.events, Deaths)]
47                 , cor_boot, R = 1e3)
48 bs2 <- boot::boot(avalanches[(EADS1 == TRUE),
49                             .(Rep.events, Deaths)]
50                 , cor_boot, R = 1e3)
51 bs3 <- boot::boot(avalanches[(EADS2 == TRUE),
52                             .(Rep.events, Deaths)]
53                 , cor_boot, R = 1e3)
54 boot::boot.ci(bs1,
55               index = 2,
56               type = "perc",
57               conf = 0.9)
58 boot::boot.ci(bs2,
59               index = 2,
60               type = "perc",
61               conf = 0.9)
62 boot::boot.ci(bs3,
63               index = 2,
64               type = "perc",
65               conf = 0.9)
66 #####
67 #b
68 to_model <- avalanches[, .(Deaths, Rep.events, EADS1, EADS2)]
69 model_mat <-
70   model.matrix(Deaths ~ ., data = to_model) #no intercept as cannot have deaths without avalanche
71
72 model_mat <- model_mat[,]
73 out_names = colnames(model_mat)
74 #no need to centre as discrete
75
76 #new data
77
78 X_new = matrix(c(1, 20, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1),
79               nrow = 4,
80               byrow = T)
81 # X_new = matrix(c(20, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1),
82 #               nrow = 4,
83 #               byrow = T)
84 N_new = nrow(X_new)
85 #check, should be similar
86 f_glm <-
87   glm(Deaths ~ ., data = to_model, family = poisson(link = "log"))
88
89
90 stan_poisson_glm <- stan_model(file = "stan/poisson_glm.stan")
91 stan_poisson_glm_data <-
92   list(
93     N = nrow(model_mat),
94     P = ncol(model_mat),
95     y = avalanches$Deaths,
96     X = model_mat,
97     n_params = c(0, 1e2),
98     N_new = N_new,
99     X_new = X_new
100   )
101
102
103 stan_poisson_glm_s <-
104   sampling(
105     stan_poisson_glm,
106     data = stan_poisson_glm_data,
107     chains = 7,
108     control = list(adapt_delta = 0.9),
109     iter = 3000,
110     init_r = 0.1
111   )
112

```

```

113 post_params <- extract(stan_poisson_glm_s, "lambda")[[1]]
114 colnames(post_params) <- out_names
115 exp_post_params <- exp(post_params)
116 apply(exp_post_params, 2, summary)
117
118
119 p_pred <- extract(stan_poisson_glm_s, "y_new")[[1]]
120 mean(p_pred[, 1] < 15)
121 mean(p_pred[, 2] > 1)
122 mean(p_pred[, 3] > 1)
123 mean(p_pred[, 4] > 1)
124
125 data_pred <- extract(stan_poisson_glm_s, "data_ppred")[[1]]
126 apply(data_pred, 2, summary)
127 #####
128 #dic is bad
129 #formulae taken from https://en.wikipedia.org/wiki/Deviance_information_criterion
130 plikrar <- function(x, data) {
131   sum(dpois(data, x, log = T))
132 }
133 sampling_rates <- extract(stan_poisson_glm_s, "rate")[[1]]
134 sr_like <-
135   apply(sampling_rates, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
136 sr_like_mean <-
137   mean(sr_like)#calculate mean log likelihood of samples
138 eap <-
139   colMeans(sampling_rates)#calculate posterior means of rates (not parameters)
140 p_mean_like <-
141   sum(dpois(avalanches$Deaths, eap, log = T))#calculate log likelihood of EAP
142 dbar <- -2 * sr_like_mean#expected deviance
143 pd <- dbar + 2 * p_mean_like#calculate penalty
144 dic <- pd + dbar#give dic
145 #####
146 #prior checking
147 # dp_av <- avalanches$Deaths/avalanches$Rep.events
148 # dp_av <- dp_av[!is.nan(dp_av)]
149 # m_deaths <- mean(dp_av)
150 # xm <- dp_av - m_deaths
151 # lnfactor <- 2/(xm)^2
152 # infactor <- dp_av / m_deaths
153 # beta_p <-
154 # mfc <- exp(xm * infactor)
155 # mfc_p <- plnorm(mfc, 0, 2)
156 avno <- avalanches$Rep.events
157 avde <- avalanches$Deaths
158 mede <- mean(avde)
159 psi <- avde / mede
160 beta <- log(psi) / (avno - mean(avno))
161 psi_p <- dlnorm(psi, 0, 2)
162 beta_p <- dnorm(beta, 0, (avno - mean(avno)) ^ (-2))
163 #####
164 stan_poisson_glm_exvar <-
165   stan_model(file = "stan/poisson_glm_exvar.stan")
166
167 model_mat <- model_mat[,-1]#messes with exvar
168 out_names = colnames(model_mat)
169
170 X_new = matrix(c(20, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1),
171               nrow = 4,
172               byrow = T)
173
174 ym <- data.frame(ym = as.factor(avalanches$Season))
175 yim <- model.matrix(~ . - 1, ym)
176
177 stan_poisson_glm_exvar_data <-
178   list(
179     N = nrow(model_mat),
180     P = ncol(model_mat),
181     y = avalanches$Deaths,
182     X = model_mat,
183     n_params = c(0, sqrt(10)),
184     N_new = N_new,
185     X_new = X_new,
186     yearindmat = yim,
187     N_years = ncol(yim)
188   )
189
190

```

```

191 stan_poisson_glm_exvar_s <-
192   sampling(
193     stan_poisson_glm_exvar,
194     data = stan_poisson_glm_exvar_data,
195     chains = 4,
196     control = list(adapt_delta = 0.999),
197     iter = 8000,
198     init_r = 1
199   )
200
201 post_params_exvar <-
202   extract(stan_poisson_glm_exvar_s, "lambda")[[1]]
203 colnames(post_params_exvar) <- out_names
204 apply(post_params_exvar, 2, summary)
205
206 dpp <- extract(stan_poisson_glm_exvar_s, "data_ppred")[[1]]
207 apply(dpp, 2, summary)
208 #####
209 plikrar <- function(x, data) {
210   sum(dpois(data, x, log = T))
211 }
212 sampling_rates_exv <- extract(stan_poisson_glm_exvar_s, "rate")[[1]]
213 sr_like_exv <-
214   apply(sampling_rates_exv, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
215 sr_like_mean_exv <-
216   mean(sr_like_exv)#calculate mean log likelihood of samples
217 eap_exv <-
218   colMeans(sampling_rates_exv)#calculate posterior means of rates (not parameters)
219 p_mean_like_exv <-
220   sum(dpois(avalanches$Deaths, eap_exv, log = T))#calculate log likelihood of EAP
221 dbar_exv <- -2 * sr_like_mean_exv#expected deviance
222 pd_exv <- dbar_exv + 2 * p_mean_like_exv#calculate penalty
223 dic_exv <- pd_exv + dbar_exv#give dic
224 #####

```

../jags/Q1jags.R

```

1 library(data.table)
2 library(ggplot2)
3
4 library(rjags)
5 library(coda)
6 library(bayesplot)
7
8
9 #####
10 #a
11 avalanches <- fread(file = "data/Avalanches.csv")
12 avalanches <- avalanches[Rep.events > 0]
13 avalanches[, ':= ' (EADS1 = (Season >= 1994 &
14   Season <= 2003),
15   EADS2 = (Season >= 2004))]
16
17 avalanches[Season %in% c(1986, 1994, 2004)]
18
19 avalanches[, EWS := 1 + EADS1 + 2 * EADS2]
20 avalanches[, EWS := as.factor(EWS)]
21
22 pglm_data <-
23   list(
24     n = nrow(avalanches),
25     rep = avalanches$Rep.events,
26     w1 = avalanches$EADS1,
27     w2 = avalanches$EADS2,
28     death = avalanches$Deaths
29   )
30
31 res.a <-
32   jags.model(
33     file = "jags/poisson.jags",
34     data = pglm_data,
35     n.chains = 4,
36     quiet = T
37   )
38 update(res.a, n.iter = 1e4)
39 res.b <-

```

```

40 coda.samples(
41   res.a,
42   variable.names = c("intercept", "beta_rep", "beta_w1", "beta_w2"),
43   n.iter = 1e4
44 )
45 summary(res.b)
46 dic.samples(model = res.a,
47   n.iter = 1e4,
48   type = 'pD')
49
50 res.a.ev <-
51   jags.model(
52     file = "jags/poisson_exvar.jags",
53     data = pglm_data,
54     n.chains = 4,
55     quiet = T
56   )
57 update(res.a, n.iter = 1e4)
58 res.b.ev <-
59   coda.samples(
60     res.a.ev,
61     variable.names = c("beta_rep", "beta_w1", "beta_w2"),
62     n.iter = 1e4
63   )
64 summary(res.b.ev)
65 dic.samples(model = res.a.ev,
66   n.iter = 1e4,
67   type = 'pD')

```

A.2 Stan

../stan/poisson_glm.stan

```

1 data {
2   int<lower=0> N;
3   int<lower=0> P;
4
5   int<lower=0> y[N];
6
7   matrix[N, P] X;
8
9   int<lower=0> N_new;
10  matrix[N_new, P] X_new;
11
12  vector[2] n_params;
13 }
14 transformed data{
15 }
16
17 parameters {
18   vector[P] lambda;
19 }
20
21 transformed parameters{
22   vector[N] log_rate = X * lambda;
23   vector[N_new] log_rate_new = X_new * lambda;
24   vector<lower=0>[N] rate = exp(log_rate);
25 }
26
27 model {
28   lambda ~ normal(n_params[1], n_params[2]);
29   y ~ poisson_log(log_rate);
30 }
31
32 generated quantities{
33   int<lower=0> y_new[N_new] = poisson_log_rng(log_rate_new);
34   int<lower=0> data_ppred[N] = poisson_log_rng(log_rate);
35 }

```

../stan/poisson_glm_exvar.stan

```
1 data {
2   int<lower=0> N;
3   int<lower=0> P;
4
5   int<lower=0> y[N];
6
7   matrix[N, P] X;
8
9   int<lower=0> N_new;
10  matrix[N_new, P] X_new;
11
12  vector[2] n_params;
13 }
14 transformed data{
15 }
16
17 parameters {
18   vector[P] lambda;
19   real<lower=0,upper=10> theta_hyp;
20   real theta;
21 }
22
23 transformed parameters{
24   vector[N] log_rate = X * lambda + theta;
25   vector[N_new] log_rate_new = X_new * lambda + theta;
26   vector<lower=0>[N] rate = exp(log_rate);
27 }
28
29 model {
30   theta_hyp ~ uniform(0, 10);
31   theta ~ normal(0, theta_hyp);
32   lambda ~ normal(n_params[1], n_params[2]);
33   y ~ poisson_log(log_rate);
34 }
35
36 generated quantities{
37   int<lower=0> y_new[N_new] = poisson_log_rng(log_rate_new);
38   int<lower=0> data_ppred[N] = poisson_log_rng(log_rate);
39 }
```

A.3 JAGS

../jags/poisson.jags

```
1 model {
2   #hyperparameters
3   p_mu <- 0
4   p_tau <- 0.01
5
6   #priors
7   intercept ~ dnorm(p_mu, p_tau)
8   beta_rep ~ dnorm(p_mu, p_tau)
9   beta_w1 ~ dnorm(p_mu, p_tau)
10  beta_w2 ~ dnorm(p_mu, p_tau)
11
12  #likelihood
13  for(i in 1:n){
14    log(mu[i]) <- intercept + beta_rep * rep[i] + beta_w1 * w1[i] + beta_w2 * w2[i]
15    death[i] ~ dpois(mu[i])
16  }
17 }
```

../jags/poisson_exvar.jags

```
1 model {
2   #hyperparameters
3   p_mu <- 0
4   p_tau <- 0.01
5
6   #priors
```

```

7   beta_rep ~ dnorm(p_mu, p_tau)
8   beta_w1 ~ dnorm(p_mu, p_tau)
9   beta_w2 ~ dnorm(p_mu, p_tau)
10  theta_hyp ~ dunif(0, 10)
11  theta ~ dnorm(0, 1 / pow(theta_hyp, 2))
12
13  #likelihood
14  for (i in 1:n) {
15    log(mu[i]) <- beta_rep * rep[i] + beta_w1 * w1[i] + beta_w2 * w2[i] + theta
16    death[i] ~ dpois(mu[i])
17  }
18 }

```

B Code for Question 2

B.1 R

../Q2.R

```

1  library(data.table)
2  library(ggplot2)
3  library(dplyr)
4
5  library(rstan)
6  rstan_options(auto_write = TRUE)
7  #options(mc.cores = parallel::detectCores())
8  Sys.setenv(LOCAL_CPPFLAGS = '-march=corei7 -mtune=corei7')
9  options(mc.cores = 1)
10 library(rstanarm)
11 library(coda)
12 library(bayesplot)
13
14 #####
15 #loading and eda
16 avalanches_prop <- fread(file = "data/Avalanches_part2.csv")
17 avalanches_prop[, Event_ID := NULL]
18 avalanches_prop[, Snow_meters := Snow_total / 100]
19 avalanches_prop[, Snow_fnights := Snow_days / 14]
20 avalanches_prop[, death_prop := Deaths / Hit]
21 avalanches_prop[, Geo_space := as.factor(Geo_space)]
22 avalanches_prop[, Rec.station := as.factor(Rec.station)]
23 cor(avalanches_prop[, .(Season, Snow_meters, Snow_fnights)])
24 #####
25 stan_binomial_glm_reff <-
26   stan_model(file = "stan/binomial_glm_randomeffects.stan")
27
28 submin <- function(x){
29   m <- min(x)
30   x <- x - m
31   attributes(x) <- list("scaled:submin" = m)
32   return(x)
33 }
34
35 cont_vars <- c("Snow_meters", "Snow_fnights")#variables to centre
36 avalanches_prop[, (cont_vars) := lapply(.SD, scale, scale = FALSE), .SDcols = cont_vars]#centre variables
37 tm_vars <- c("Season")
38 avalanches_prop[, (tm_vars) := lapply(.SD, submin), .SDcols = tm_vars]
39
40
41 X_fixedeff <-
42   model.matrix(death_prop ~ Season + Snow_meters + Snow_fnights - 1, data = avalanches_prop)
43 X_randomeff <-
44   model.matrix(death_prop ~ Geo_space - 1, data = avalanches_prop)
45 success <- avalanches_prop[, Deaths]
46 trials <- avalanches_prop[, Hit]
47
48
49 stan_binomial_glm_reff_data <-
50   list(
51     success = success,
52     trials = trials,
53     X_f = X_fixedeff,
54     X_r = X_randomeff,

```



```

55     N = length(success),
56     P_f = ncol(X_fixedeff),
57     P_r = ncol(X_ranomeff),
58     n_params = c(0, sqrt(10))
59   )
60
61 stan_binomial_glm_reff_s <-
62   sampling(
63     stan_binomial_glm_reff,
64     data = stan_binomial_glm_reff_data,
65     chains = 4,
66     control = list(adapt_delta = 0.9),
67     iter = 10000#,
68     #init_r = 0.1
69   )
70 reff_coda <- As.mcmc.list(stan_binomial_glm_reff_s, pars = c("beta_r", "beta_f"))
71 gelman.plot(reff_coda, ask = FALSE)
72
73 plot_diag_objects <- function(stanfit){
74   list(post = as.array(stanfit),
75         lp = log_posterior(stanfit),
76         np = nuts_params(stanfit))
77 }
78
79 plot_diag <- function(stanfit, pars){
80   ps <- vars(starts_with(pars))
81   post <- as.array(stanfit)
82   lp <- log_posterior(stanfit)
83   np <- nuts_params(stanfit)
84   p1 <- mcmc_parcoord(post, np = np, pars = ps)
85   p2 <- mcmc_pairs(post, np = np, pars = ps)
86   p3 <- mcmc_trace(post, pars = ps, np = np)
87   p4 <- mcmc_nuts_divergence(np, lp)
88   p5 <- mcmc_nuts_energy(np)
89   list(p1, p2, p3, p4, p5)
90 }
91
92 #mcmc_trace(stan_binomial_glm_reff_s, pars = vars(starts_with("beta")))
93
94 #####
95 #sans snow fortnights
96
97 X_f_nsf <- model.matrix(death_prop ~ Season + Snow_meters - 1, data = avalanches_prop)
98
99 stan_binomial_glm_reff_nsf_data <-
100   list(
101     success = success,
102     trials = trials,
103     X_f = X_f_nsf,
104     X_r = X_ranomeff,
105     N = length(success),
106     P_f = ncol(X_f_nsf),
107     P_r = ncol(X_ranomeff),
108     n_params = c(0, sqrt(10))
109   )
110
111 stan_binomial_glm_reff_nsf_s <-
112   sampling(
113     stan_binomial_glm_reff,
114     data = stan_binomial_glm_reff_nsf_data,
115     chains = 4,
116     control = list(adapt_delta = 0.9),
117     iter = 10000#,
118     #init_r = 0.1
119   )
120
121 c_data <- extract(stan_binomial_glm_reff_nsf_s, "data_prop")
122
123
124 #####
125 #hierarchical on station, sans snow fortnights
126 X_r_station <- model.matrix(death_prop ~ Rec.station - 1, data = avalanches_prop)
127
128 stan_binomial_glm_reff_station_data <-
129   list(
130     success = success,
131     trials = trials,
132     X_f = X_f_nsf,

```

```

133   X_r = X_r_station,
134   N = length(success),
135   P_f = ncol(X_f_nsf),
136   P_r = ncol(X_r_station),
137   n_params = c(0, sqrt(10))
138 )
139
140 stan_binomial_glm_reff_station_s <-
141   sampling(
142     stan_binomial_glm_reff,
143     data = stan_binomial_glm_reff_station_data,
144     chains = 4,
145     control = list(adapt_delta = 0.9),
146     iter = 10000#,
147     #init_r = 0.1
148   )

```

B.2 Stan

../stan/binomial_glm.stan

```

1  data {
2    int<lower=0> N;
3    int<lower=0> P;
4
5    int<lower=0> y[N];
6
7    matrix[N, P] X;
8
9    vector[2] n_params;
10 }
11
12 parameters {
13   vector[P] beta;
14 }
15
16 transformed parameters{
17   vector[N] lg_p = X * beta;
18 }
19
20 model {
21   beta ~ normal(n_params[1], n_params[2]);
22   y ~ binomial(1, inv_logit(lg_p));
23 }
24 generated quantities{
25   int data_ppred[N] = binomial_rng(1, inv_logit(lg_p));
26 }

```

../stan/binomial_glm_ranomeffects.stan

```

1  data {
2    int<lower=0> N;
3    int<lower=0> P_f;
4    int<lower=0> P_r;
5
6    int<lower=0> success[N];
7    int<lower=1> trials[N];
8
9    matrix[N, P_f] X_f;
10   matrix[N, P_r] X_r;
11
12   vector[2] n_params;
13 }
14
15 parameters {
16   vector[P_f] beta_f;
17   vector[P_r] sn_vec;
18   real<lower=0,upper=10> reff_sdv;
19 }
20
21 transformed parameters{
22   vector[P_r] beta_r = reff_sdv * sn_vec;

```

```

23   vector[N] lg_p = X_f * beta_f + X_r * beta_r;
24 }
25
26 model {
27   reff_sdv ~ uniform(0, 10);
28   sn_vec ~ std_normal(); //hence beta_r ~ normal(0, reff_sdv)
29   beta_f ~ normal(n_params[1], n_params[2]);
30   success ~ binomial(trials, inv_logit(lg_p));
31 }
32 generated quantities{
33   int data_ppred[N] = binomial_rng(trials, inv_logit(lg_p));
34   vector[N] data_prop = inv_logit(lg_p);
35 }

```