

Bayesian Data Analysis Assignment 2

Benjamin Cox, S1621312

Question 1

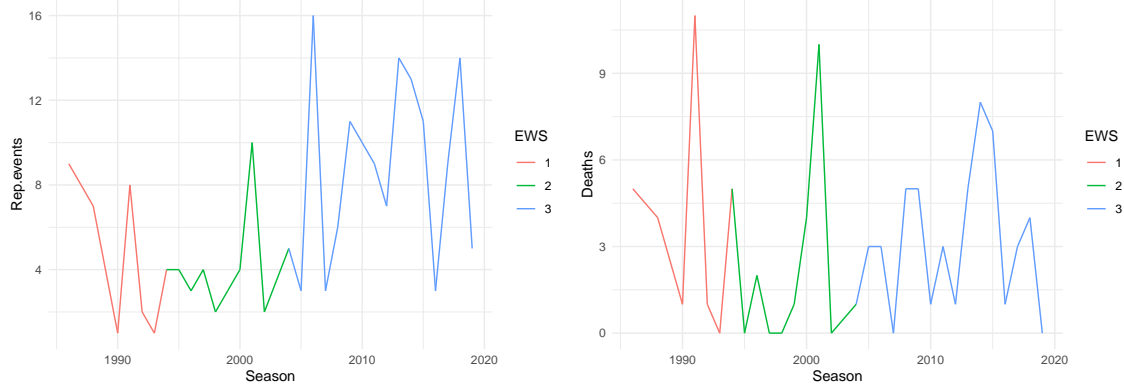


Figure 1: Plots illustrating the temporal evolution of avalanche related statistics. The EWS measure is 1 = No EADS, 2 = EADS present, 3 = EADS online daily.

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:

$$\begin{aligned}\lambda_i &= \text{Rep.events}_i \cdot \exp(\beta_0 + \beta_1 \cdot \text{EADS1}_i + \beta_2 \cdot \text{EADS2}_i) \\ \log(\lambda_i) &= \beta_0 + \beta_1 \cdot \text{EADS1}_i + \beta_2 \cdot \text{EADS2}_i + \ln(\text{Rep.events}_i) \\ \text{Deaths}_i &\sim \text{Poisson}(\lambda_i)\end{aligned}$$

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. Note that we are using an offset, as we are calculating the rate of casualty per avalanche, so the rate should be the same per avalanche and hence the offset.

We could model without the offset and with a regression coefficient on the number of avalanches. However this nonsensical, as it would imply that there can be avalanche deaths without an avalanche even occurring.

That model also had quite a bit larger DIC (and was overall worse in other measures such as WAIC and LOO) than the model that we are using.

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)	EADS1TRUE	EADS2TRUE
Min.	0.30	0.26	0.16
1st Qu.	0.66	0.64	0.39
Median	0.77	0.78	0.47
Mean	0.78	0.82	0.48
3rd Qu.	0.89	0.96	0.55
Max.	1.64	2.44	1.21

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 per avalanche given no mitigation is 0.78. We also see that each EADS evolution decreases the expected number of deaths, by 0.82 and 0.48 times respectively (if all other variables are held constant). 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 the EADS will still be online, so we have the appropriate data.

We obtain a probability of $P(D < 15 | A = 20, EADS = 2) = 0.987$.

We are also interested in the probability of observing more than 1 death in mean per avalanche in each stage of the EADS lifespan (not present, present, online). For this we need to calculate

$$P\left(\frac{\lambda}{\text{Rep.events}} > 1 \mid \text{EADS} = x\right).$$

Given our offsetting this is rather simple, as this simplifies to

$$P(\exp(\beta_0 + \beta_1 \cdot \text{EADS1} + \beta_2 \cdot \text{EADS2}) > 1 | \text{EADS} = x),$$

of which we have posterior samples.

We calculate these probabilities for all values of the EADS and obtain

No EADS	EADS	EADS online
0.104	0.005	0

Table 2: Probabilities of multiple fatalities per avalanche given the various states of the EADS

After this we are told that on average the number of avalanches per year is between 5 and 15, and that they consider that for an extreme number of events that the number of casualties could be 4 times greater (or lesser) than the average number of casualties.

From this we work out that the mean number of avalanches is 10 with standard deviation 5. We also want to give the multiplier high mass between 0.25 and 4.

Suggested is a log-normal prior with mean 0 and standard deviation 2 on

$$\phi = \exp((x - \mu_x) \cdot \beta_{\text{Rep.events}}),$$

the multiplier. This implies a normal prior with mean 0 and standard deviation 2 for $(x - \mu_x) \cdot \beta_{\text{Rep.events}}$, or $\beta_{\text{Rep.events}} \sim N(\mu = 0, \sigma^2 = 4(x - \mu_x)^2)$. There could be problems with this, as it is possible for $(x - \mu_x)$ to be 0.

The mean and standard deviation parameters for a lognormal distribution are typically given as the mean and standard deviation of the underlying normal distribution. Hence we calculate the true mean and SD as

$$\mu_\phi = \exp\left(0 + \frac{2^2}{2}\right) = e^2 \approx 7.39, \quad \sigma_\phi^2 = (\exp(2^2) - 1) \exp(2 \cdot 0 + 2^2) = e^8 - e^4 \approx 2925, \implies \sigma \approx 54.$$

This is clearly not appropriate for the multiplier, as the mean is too high, and the standard deviation even moreso.

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, EADS1, EADS2)]
69 model_mat <-
70   model.matrix(Deaths ~ ., data = to_model) #no intercept as cannot have deaths without avalanche
71 d_offset <- log(avalanches$Rep.events)
```

```

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, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1),
79               nrow = 4,
80               byrow = T)
81 n_offset <- log(c(20, 1, 1, 1))
82 # X_new = matrix(c(20, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1),
83 #               nrow = 4,
84 #               byrow = T)
85 N_new = nrow(X_new)
86 #check, should be similar
87 f_glm <-
88   glm(Deaths ~ ., data = to_model, family = poisson(link = "log"))
89
90
91 stan_poisson_glm <- stan_model(file = "stan/poisson_glm.stan")
92 stan_poisson_glm_data <-
93   list(
94     N = nrow(model_mat),
95     P = ncol(model_mat),
96     y = avalanches$Deaths,
97     X = model_mat,
98     n_params = c(0, 1e2),
99     N_new = N_new,
100    X_new = X_new,
101    offset = d_offset,
102    offset_new = n_offset
103  )
104
105
106 stan_poisson_glm_s <-
107   sampling(
108     stan_poisson_glm,
109     data = stan_poisson_glm_data,
110     chains = 7,
111     control = list(adapt_delta = 0.9),
112     iter = 3000,
113     init_r = 0.1
114   )
115
116 post_params <- extract(stan_poisson_glm_s, "lambda")[[1]]
117 colnames(post_params) <- out_names
118 exp_post_params <- exp(post_params)
119 apply(exp_post_params, 2, summary)
120
121 news_1 <- mean(exp(post_params[, 1]) > 1)
122 news_2 <- mean(exp(post_params[, 1] + post_params[, 2]) > 1)
123 news_3 <- mean(exp(post_params[, 1] + post_params[, 3]) > 1)
124
125
126 p_pred <- extract(stan_poisson_glm_s, "y_new")[[1]]
127 mean(p_pred[, 1] < 15)
128 mean(p_pred[, 2] > 1)
129 mean(p_pred[, 3] > 1)
130 mean(p_pred[, 4] > 1)
131
132 data_pred <- extract(stan_poisson_glm_s, "data_ppred")[[1]]
133 apply(data_pred, 2, summary)
134 #####
135 #dic is bad
136 #formulae taken from https://en.wikipedia.org/wiki/Deviance_information_criterion
137 plikrar <- function(x, data) {
138   sum(dpois(data, x, log = T))
139 }
140 sampling_rates <- extract(stan_poisson_glm_s, "rate")[[1]]
141 sr_like <-
142   apply(sampling_rates, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
143 sr_like_mean <-
144   mean(sr_like)#calculate mean log likelihood of samples
145 eap <-
146   colMeans(sampling_rates)#calculate posterior means of rates (not parameters)
147 p_mean_like <-
148   sum(dpois(avalanches$Deaths, eap, log = T))#calculate log likelihood of EAP
149 dbar <- -2 * sr_like_mean#expected deviance

```

```

150 pd <- dbar + 2 * p_mean_like#calculate penalty
151 dic <- pd + dbar#give dic
152 #####
153 #prior checking
154 # dp_av <- avalanches$Deaths/avalanches$Rep.events
155 # dp_av <- dp_av[!is.nan(dp_av)]
156 # m_deaths <- mean(dp_av)
157 # xm <- dp_av - m_deaths
158 # lnfactor <- 2/(xm)^2
159 # lnfactor <- dp_av / m_deaths
160 # beta_p <-
161 # mfc <- exp(xm * lnfactor)
162 # mfc_p <- plnorm(mfc, 0, 2)
163 avno <- avalanches$Rep.events
164 avde <- avalanches$Deaths
165 mede <- mean(avde)
166 psi <- avde / mede
167 beta <- log(psi) / (avno - mean(avno))
168 psi_p <- dlnorm(psi, 0, 2)
169 beta_p <- dnorm(beta, 0, (avno - mean(avno)) ^ (-2))
170 #####
171 stan_poisson_glm_exvar <-
172   stan_model(file = "stan/poisson_glm_exvar.stan")
173
174 model_mat <- model_mat[,-1]#messes with exvar
175 out_names = colnames(model_mat)
176
177 X_new = matrix(c(0, 1, 0, 0, 1, 0, 0, 1),
178               nrow = 4,
179               byrow = T)
180
181 n_offset <- log(c(20, 1, 1, 1))
182
183 ym <- data.frame(ym = as.factor(avalanches$Season))
184 yim <- model.matrix( ~ . - 1, ym)
185
186 stan_poisson_glm_exvar_data <-
187   list(
188     N = nrow(model_mat),
189     P = ncol(model_mat),
190     y = avalanches$Deaths,
191     X = model_mat,
192     n_params = c(0, sqrt(10)),
193     N_new = N_new,
194     X_new = X_new,
195     yearindmat = yim,
196     N_years = ncol(yim),
197     offset = d_offset,
198     offset_new = n_offset
199   )
200
201
202 stan_poisson_glm_exvar_s <-
203   sampling(
204     stan_poisson_glm_exvar,
205     data = stan_poisson_glm_exvar_data,
206     chains = 4,
207     control = list(adapt_delta = 0.999),
208     iter = 8000,
209     init_r = 1
210   )
211
212 post_params_exvar <-
213   extract(stan_poisson_glm_exvar_s, "lambda")[[1]]
214 colnames(post_params_exvar) <- out_names
215 apply(post_params_exvar, 2, summary)
216
217 dpp <- extract(stan_poisson_glm_exvar_s, "data_ppred")[[1]]
218 apply(dpp, 2, summary)
219 #####
220 plikrar <- function(x, data) {
221   sum(dpois(data, x, log = T))
222 }
223 sampling_rates_exv <- extract(stan_poisson_glm_exvar_s, "rate")[[1]]
224 sr_like_exv <-
225   apply(sampling_rates_exv, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
226 sr_like_mean_exv <-
227   mean(sr_like_exv)#calculate mean log likelihood of samples

```

```

228 eap_exv <-
229   colMeans(sampling_rates_exv)#calculate posterior means of rates (not parameters)
230 p_mean_like_exv <-
231   sum(dpois(avalanches$Deaths, eap_exv, log = T))#calculate log likelihood of EAP
232 dbar_exv <- -2 * sr_like_mean_exv#expected deviance
233 pd_exv <- dbar_exv + 2 * p_mean_like_exv#calculate penalty
234 dic_exv <- pd_exv + dbar_exv#give dic
235 #####

```

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  vector[N] offset;
15  vector[N_new] offset_new;
16 }
17 transformed data{
18 }
19
20 parameters {
21   vector[P] lambda;
22 }
23
24 transformed parameters{
25   vector[N] log_rate = X * lambda + offset;
26   vector[N_new] log_rate_new = X_new * lambda + offset_new;
27   vector<lower=0>[N] rate = exp(log_rate);
28 }
29
30 model {
31   lambda ~ normal(n_params[1], n_params[2]);
32   y ~ poisson_log(log_rate);
33 }
34
35 generated quantities{
36   int<lower=0> y_new[N_new] = poisson_log_rng(log_rate_new);
37   int<lower=0> data_ppred[N] = poisson_log_rng(log_rate);
38 }

```

../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  vector[N] offset;
15  vector[N_new] offset_new;
16 }
17 transformed data{
18 }

```

```

19
20 parameters {
21   vector[P] lambda;
22   real<lower=0,upper=10> theta_hyp;
23   real theta;
24 }
25
26 transformed parameters{
27   vector[N] log_rate = X * lambda + theta + offset;
28   vector[N_new] log_rate_new = X_new * lambda + theta + offset_new;
29   vector<lower=0>[N] rate = exp(log_rate);
30 }
31
32 model {
33   theta_hyp ~ uniform(0, 10);
34   theta ~ normal(0, theta_hyp);
35   lambda ~ normal(n_params[1], n_params[2]);
36   y ~ poisson_log(log_rate);
37 }
38
39 generated quantities{
40   int<lower=0> y_new[N_new] = poisson_log_rng(log_rate_new);
41   int<lower=0> data_ppred[N] = poisson_log_rng(log_rate);
42 }

```

A.3 JAGS

../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     w1 = avalanches$EADS1,
26     w2 = avalanches$EADS2,
27     death = avalanches$Deaths,
28     offset = log(avalanches$Rep.events)
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_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_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')

```

../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_w1 ~ dnorm(p_mu, p_tau)
9   beta_w2 ~ dnorm(p_mu, p_tau)
10
11  #likelihood
12  for(i in 1:n){
13    log(mu[i]) <- intercept + beta_w1 * w1[i] + beta_w2 * w2[i] + offset[i]
14    death[i] ~ dpois(mu[i])
15  }
16 }

```

../jags/poisson_exvar.jags

```

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

```

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_fights := 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_fights)])
24 #####
25 stan_binomial_glm_reff <-
26   stan_model(file = "stan/binomial_glm_ranomeffects.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_fights")#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_fights - 1, data = avalanches_prop)
43 X_ranomeff <-
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_ranomeff,
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_randomeff,
105     N = length(success),
106     P_f = ncol(X_f_nsf),
107     P_r = ncol(X_randomeff),
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 }
```