

# Bayesian Data Analysis Assignment 2

Benjamin Cox, S1621312

## Question 1

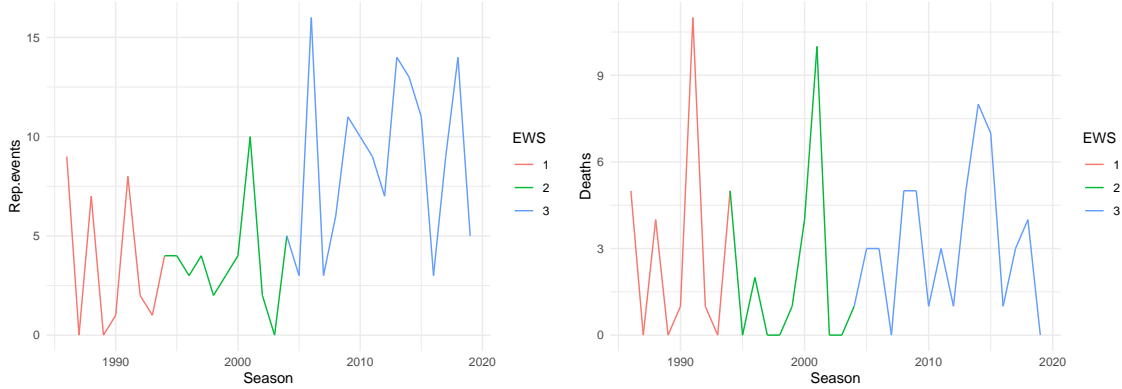


Figure 1: Plots illustrating the temporal evolution of avalanche related statistics. The EWS measure is 1 = No EADS, 2 = EADS extant, 3 = EADS online daily.

From the above graphs we can see a broadly 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 (0.6397, 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:

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

We could model with an offset and without a regression coefficient on the number of avalanches. That model would assume a constant rate per avalanche, which this model does not. We note that this model allows for deaths without an avalanche occurring.

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

We are going to run 7 parallel chains with initial values drawn from a Uniform(-0.1, 0.1) distribution. We are going to run each chain for 3000 iterations and discard the first 1500 (HMC/NUTS converges faster than Gibbs so the length is fine).

After running we check BGR statistics and find that they have all converged to 1. We also check NUTS specific diagnostics (divergences, energies) and find them satisfactory as well (no divergences, good energy mixing). Therefore we proceed with our analysis.

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

	(Intercept) ( $\beta_0$ )	Rep.events ( $\beta_3$ )	EADS1TRUE ( $\beta_1$ )	EADS2TRUE ( $\beta_2$ )
Min.	0.35	1.09	0.22	0.12
1st Qu.	0.86	1.19	0.71	0.32
Median	1.05	1.22	0.88	0.39
Mean	1.08	1.22	0.92	0.41
3rd Qu.	1.26	1.24	1.08	0.48
Max.	2.62	1.38	3.01	1.32

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 year given no mitigation (ie all other covariates 0) is 1.08. We also see that each EADS evolution decreases the expected number of deaths, by 0.92 and 0.41 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 also see that each avalanche increases the number of expected deaths 1.22 times. This means that avalanches get exponentially more dangerous the more that there are, which seems somewhat strange.

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  $P(\text{Deaths} < 15 | \text{Rep.events} = 20, \text{EADS} = 2) = 0.185$  with a 95% bootstrap interval of (0.1838, 0.1864). This is rather low, but this is expected given that large number of avalanches (and that they get more dangerous the more there are.)

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
0.105	0.005	0 (machine precision)

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.

We are now going to expand our model to include a term to capture randomness not accounted for by the other components. We are going to design the model as follows

$$\begin{aligned} \theta_{hyp} &\sim \text{Uniform}(0, 10), \\ \theta &\sim \text{Normal}(0, \theta_{hyp}), \\ \lambda_i &= \exp(\beta_1 \cdot \text{EADS1}_i + \beta_2 \cdot \text{EADS2}_i + \beta_3 \cdot \text{Rep.events}_i + \theta) \\ \log(\lambda_i) &= \beta_1 \cdot \text{EADS1}_i + \beta_2 \cdot \text{EADS2}_i + \beta_3 \cdot \text{Rep.events}_i + \theta \\ \text{Deaths}_i &\sim \text{Poisson}(\lambda_i) \end{aligned}$$

This model is a lot more computationally complex than the other model and requires us to make some tweaks to the sampling and model code in order to make it converge well. It runs significantly slower than the previous model, but we do get convergence. We have re-parametrised and de-centred the model so that it is mathematically equivalent, but we are dealing with standard normals and multiples thereof, rather than working with normals with variable  $\sigma$ .

To make this model run well we must remove the intercept term. This is because  $\theta$  and the intercept term serve the same purpose; capture the latent effect. Therefore the intercept term must be removed, as  $\theta + \beta_0$  should be constant, but this does not constrain either of them, thus without removing the intercept we do not get convergence. We note that  $\beta_0$  had a normal prior with mean 0, so  $\theta$  should well compensate for it.

We are going to run 4 parallel chains with initial values drawn from a Uniform(-0.05, 0.05) distribution. We are going to run each chain for 8000 iterations and discard the first 4000 (HMC/NUTS converges faster than Gibbs so the length is fine). We are going to increase the maximum tree depth to 15 (from 10) and increase the adaptation acceptance probability to 0.99 (from 0.8). These will help us to deal with the implied distributional shape given by the uniform-normal combination. It will significantly slow sampling, but this is required for convergence.

After running we check BGR statistics and find that they have all converged to 1. We also check NUTS specific diagnostics (divergences, energies) and find them satisfactory as well (no divergences, good energy mixing). Therefore we proceed with our analysis.

This is one of the few times I have seen JAGS converge better than Stan, as the NUTS sampler finds it somewhat tricky to deal with the implied distribution space given by the normal-uniform combination alongside the others. We have to run for more iterations and with a smaller stepping than we would like, so it takes significantly longer to run. A single chain of this model takes over 3 times as long as all of the chains of the previous model. Given all of this it should give us a lot better predictions right?

Well, no.

We obtain the following table for our posterior values

Observe that these are mostly the same as the estimates that we got above, with the exception that we have  $\theta$  rather than  $\beta_0$ . However  $\theta$  has different distributional properties not captured in this table that make it somewhat better for this task.

Now we are going to compare the two models and make some recommendations. Comparing the posterior predictives for the data they give identical results:

	Rep.events ( $\beta_3$ )	EADS1TRUE ( $\beta_1$ )	EADS2TRUE ( $\beta_2$ )	theta ( $\theta$ )
Min.	1.09	0.26	0.12	0.37
1st Qu.	1.19	0.73	0.32	0.89
Median	1.22	0.89	0.40	1.02
Mean	1.22	0.93	0.42	1.06
3rd Qu.	1.24	1.08	0.49	1.21
Max.	1.36	2.99	1.34	3.35

Table 3: Posterior summaries for the second Poisson model, which attempts to encapsulate the extra variability

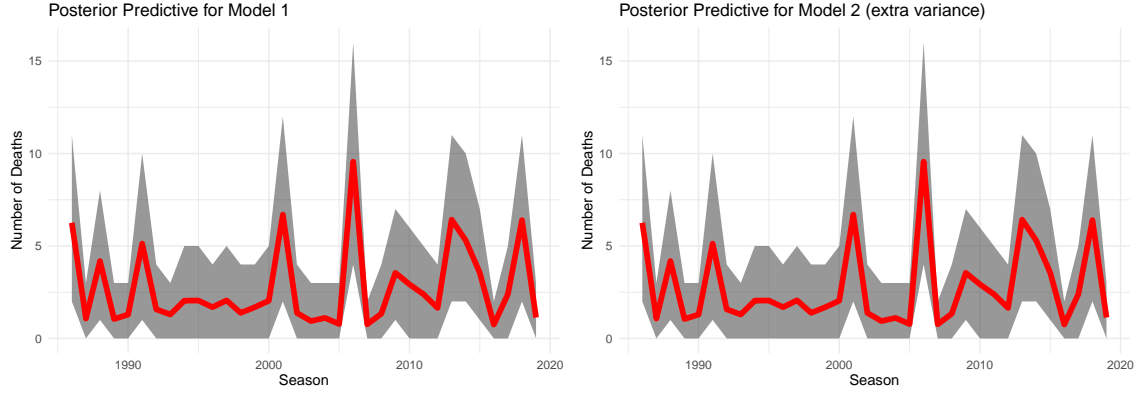


Figure 2: Posterior predictive plots for the data. Note that they are identical. the red line indicates the predictive mean, and the bands indicate the 90% credible interval.

Comparing the parameter summaries tells a similar story:

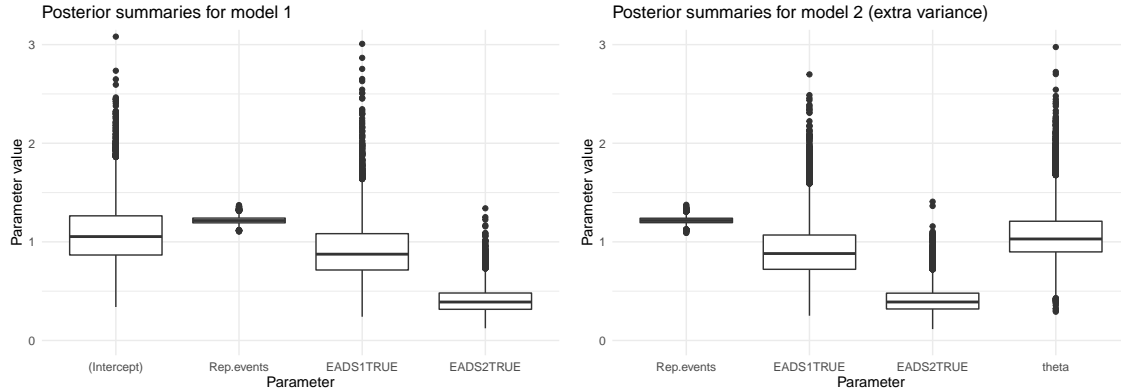


Figure 3: Posterior summaries for the parameters for each model. The parameters have been exponentiated to ease interpretation.

We see that there is more variability in theta than in the intercept, but both models will lead to the same conclusions as they are very similar in terms of distribution.

Calculating the Deviance Information Criterion for both models we get a DIC of 141.9 for the first model and a DIC of 141.6 for the second. Based on this we would weakly prefer the second model, as it has a smaller DIC.

However I would prefer the first model. The DICs are very similar in size, and given the stochastic nature overlap distributionally quite a bit. However the first model is both more interpretable and more stable.

The first model converges better and samples faster. Both lead to the same conclusions, so I don't see much reason to choose the second.

However I propose a third model. I believe that a model of the first form with the number of avalanches as an offset rather than as a covariate makes the most sense. This is because it would mean that each avalanche is not inherently more dangerous than the last. It would also ease interpretation further, as the calculated rates would be deaths per avalanche. Furthermore it would eliminate the predictions of deaths without avalanches occurring, which is a problem with the two previous models.

However it would not account for some years having more avalanches and thus being more dangerous than other years. I believe that this model makes more sense (and believed that it was the model we were being asked to work on prior to corresponding with the lecturer), but the biggest weakness is what I just mentioned. This model is more classical Poisson, but does not allow for some more advanced deductions.

## Question 2

We have data on each avalanche reported in the Piedmont region between 2014 and 2019. The data contains the year reported (Season), the number of people involved (Hit), the number of fatalities (Deaths), and the nearest recording station (Rec.station). Furthermore for each recording station the total amount of snow (Snow\_total, cm) and the snow permanence (Snow\_days, days) are recorded for each year. The recording stations are divided into 3 geographical regions (Geo.space).

We transform Snow\_days into Snow\_fnights by dividing by 14. This gives us the permanence in fort-nights. We transform Snow\_total into Snow\_meters by dividing by 100. This gives us the amount of snow in meters. We do not round, as this would lose information.

We centre both Snow\_fnights and Snow\_meters, as they are continuous variables, so centering them will not aid in interpretation and in sampler convergence.

We subtract the minimum of Season from Season, meaning that we interpret Season as the number of years since 2014. This will aid in interpretation and in sampler convergence.

None of these transformations will change correlations or variance, as they are additive.

We obtain the following correlations between our three variables.

Pearson Correlation	Season	Snow_meters	Snow_fnights
Season	1.00	-0.10	-0.10
Snow_meters	-0.10	1.00	0.83
Snow_fnights	-0.10	0.83	1.00

From this we can deduce a weak negative correlation between Season and both Snow\_meters and Snow\_fnights, likely due to climate change. Furthermore we see that Snow\_meters and Snow\_fnights are extremely highly correlated, as expected (more snow means more time for snow to melt). This could prove to be a problem, so we will address this later.

We are interested in fitting a random effects model for the proportion of fatalities in an avalanche. We assume fixed effects for Season, Snow\_meters, and Snow\_fnights, with a random effect on Geo.space. This is characterised by the following equations:

$$\begin{aligned}
\text{logit}(p_i) &= \beta_1 \cdot \text{Season} + \beta_2 \cdot \text{Snow\_meters} + \beta_3 \cdot \text{Snow\_fnight} + R_{\text{Geo.space}_i} \\
R_i &\equiv R_{\text{Geo.space}_i} \sim \text{Normal}(0, R_{hyp}) \\
R_{hyp} &\sim \text{Uniform}(0, 10) \\
\beta_i &\sim \text{Normal}(0, 10) \\
\text{Deaths}_i &\sim \text{Binomial}(\text{Hit}_i, p_i)
\end{aligned}$$

Note that there is no intercept as the random effects on the geographical location serve the same purpose. If we included an intercept we would not get repeatable results as the random effects would be offset by the intercept, thus both the intercept and random effects would converge to different values in each run (but

would sum to the same distribution). We would include an intercept if some events had no random effect associated, but that is not the case here.

We are going to run 4 parallel chains with initial values drawn from a Uniform(-0.1, 0.1) distribution. We are going to run each chain for 10000 iterations and discard the first 5000 (HMC/NUTS converges faster than Gibbs so the length is fine). We are going to increase the adaptation acceptance probability to 0.95 (from 0.8). These will help us to deal with the implied distributional shape given by the uniform-normal combination. It will significantly slow sampling, but this is required for convergence.

After running we check BGR statistics and find that they have all converged to 1. We also check NUTS specific diagnostics (divergences, energies) and find them satisfactory as well (no divergences, good energy mixing). Therefore we proceed with our analysis.

We obtain the following posterior summaries for our parameters:

	Season ( $\beta_1$ )	Snow_meters ( $\beta_2$ )	Snow_fights ( $\beta_3$ )	Geo_space1 ( $R_1$ )	Geo_space2 ( $R_2$ )	Geo_space3 ( $R_3$ )
Min.	-0.69	-1.01	-0.71	-2.12	-1.70	-3.89
1st Qu.	-0.27	-0.33	-0.04	-0.09	-0.26	-0.87
Median	-0.19	-0.19	0.08	0.10	-0.05	-0.40
Mean	-0.18	-0.18	0.08	0.18	-0.07	-0.52
3rd Qu.	-0.10	-0.04	0.19	0.45	0.11	-0.07
Max.	0.43	0.89	0.80	2.83	1.36	1.24

Table 4: Posterior summaries for the first binomial random effects model, which has the effects on geographical area

These parameters have an effect on the logit scale, meaning that they affect the log-odds of deaths:survived. For example a snow depth of 1 meter above the mean subtracts 0.18 from the expected log-odds (all other variables held constant). This means that survival becomes more likely the more snow that has occurred.

Looking at the regions we see that region 1 is more dangerous than region 2, which is more dangerous than region 3 (if all other variables are the same).

Some of these estimates seem strange, as we would intuitively think that more snow is more deadly. The seasonal trend is expected, and the geographical trend is interesting. However the high correlation between the amount of snow and its permanence seems to be affecting the model, as realistically both of them should have the same sign due to their correlation. Therefore we propose a second model without the Snow\_fights term.

$$\text{logit}(p_i) = \beta_1 \cdot \text{Season} + \beta_2 \cdot \text{Snow\_meters} + R_{\text{Geo.space}_i}$$

We perform the same adjustments to the sampler that we did for this model, and check the same statistics.

We obtain the following posterior summaries for our parameters:

	Season ( $\beta_1$ )	Snow_meters ( $\beta_2$ )	Geo_space1 ( $R_1$ )	Geo_space2 ( $R_2$ )	Geo_space3 ( $R_3$ )
Min.	-0.87	-0.56	-2.04	-1.51	-3.12
1st Qu.	-0.26	-0.17	-0.10	-0.24	-0.93
Median	-0.18	-0.09	0.13	-0.03	-0.57
Mean	-0.18	-0.10	0.20	-0.05	-0.61
3rd Qu.	-0.09	-0.02	0.50	0.14	-0.23
Max.	0.35	0.29	2.91	1.73	0.72

Table 5: Posterior summaries for the second binomial random effects model, which has the effects on geographical area. This model has been adjusted to compensate for collinearity between Snow\_meters and Snow\_fights.

## 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[, ']:= (EADS1 = (Season >= 1994 &
18                        Season <= 2003),
19                        EADS2 = (Season >= 2004))]]
20
21 avalanches[Season %in% c(1986, 1994, 2004)]
22
23 avalanches[, EWS := 1 + EADS1 + 2 * EADS2]
24 avalanches[, EWS := as.factor(EWS)]
25
26 base_plot <-
27   ggplot(data = as.data.frame(avalanches), aes(colour = EWS)) + theme_minimal()
28 base_plot + geom_line(aes(x = Season, y = Rep.events, group = F))
29 base_plot + geom_line(aes(x = Season, y = Deaths, group = F))
30 base_plot + geom_boxplot(aes(x = EWS, y = Deaths), colour = "black")
31
32 #avalanches <- avalanches[Rep.events > 0]
33 cor_boot <- function(data, index) {
34   dt_s <- data[index, ]
35   return(cor(dt_s))
36 }
37
38 cor(avalanches[(EADS1 == FALSE &
39                 EADS2 == FALSE), .(Rep.events, Deaths)])
40 cor(avalanches[EADS1 == TRUE, .(Rep.events, Deaths)])
41 cor(avalanches[EADS2 == TRUE, .(Rep.events, Deaths)])
42
43 bs1 <- boot::boot(avalanches[(EADS1 == FALSE &
44                               EADS2 == FALSE),
45                     .(Rep.events, Deaths)]
46                 , cor_boot, R = 1e3)
47 bs2 <- boot::boot(avalanches[(EADS1 == TRUE),
48                               .(Rep.events, Deaths)]
49                 , cor_boot, R = 1e3)
50 bs3 <- boot::boot(avalanches[(EADS2 == TRUE),
51                               .(Rep.events, Deaths)]
52                 , cor_boot, R = 1e3)
53 boot::boot.ci(bs1,
54               index = 2,
55               type = "perc",
56               conf = 0.9)
57 boot::boot.ci(bs2,
58               index = 2,
59               type = "perc",
60               conf = 0.9)
61 boot::boot.ci(bs3,
62               index = 2,
63               type = "perc",
64               conf = 0.9)
65 #####
66 #b
67 to_model <- avalanches[, .(Rep.events, Deaths, EADS1, EADS2)]
68 model_mat <-
69   model.matrix(Deaths ~ ., data = to_model)#no intercept as cannot have deaths without avalanche
70 #d_offset <- log(avalanches$Rep.events)
71 d_offset <- rep(0, nrow(avalanches))
```

```

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
82 X_new = matrix(c(1, 20, 0, 1,
83                 1, 1, 0, 0,
84                 1, 1, 1, 0,
85                 1, 1, 0, 1),
86               nrow = 4,
87               byrow = T)
88 #n_offset <- log(c(20, 1, 1, 1))
89 n_offset <- rep(0, nrow(X_new))
90
91 N_new = nrow(X_new)
92 #check, should be similar
93 f_glm <-
94   glm(Deaths ~ ., data = to_model, family = poisson(link = "log"))
95
96
97 stan_poisson_glm <- stan_model(file = "stan/poisson_glm.stan")
98 stan_poisson_glm_data <-
99   list(
100     N = nrow(model_mat),
101     P = ncol(model_mat),
102     y = avalanches$Deaths,
103     X = model_mat,
104     n_params = c(0, 1e2),
105     N_new = N_new,
106     X_new = X_new,
107     offset = d_offset,
108     offset_new = n_offset
109   )
110
111
112 stan_poisson_glm_s <-
113   sampling(
114     stan_poisson_glm,
115     data = stan_poisson_glm_data,
116     chains = 7,
117     control = list(adapt_delta = 0.8),
118     iter = 1e5,
119     init_r = 0.1
120   )
121
122 post_params <- extract(stan_poisson_glm_s, "lambda")[[1]]
123 colnames(post_params) <- out_names
124 exp_post_params <- exp(post_params)
125 apply(exp_post_params, 2, summary)
126 apply(post_params, 2, summary)
127
128 news_1 <- mean(exp(post_params[, 1]) > 1)
129 news_2 <- mean(exp(post_params[, 1] + post_params[, 2]) > 1)
130 news_3 <- mean(exp(post_params[, 1] + post_params[, 3]) > 1)
131
132
133 p_pred <- extract(stan_poisson_glm_s, "y_new")[[1]]
134 mean(p_pred[, 1] < 15)
135 mean(p_pred[, 2] > 1)
136 mean(p_pred[, 3] > 1)
137 mean(p_pred[, 4] > 1)
138
139 pp1 <- p_pred[,1] < 15
140
141 mean_boot <- function(data, index) {
142   dt_s <- data[index]
143   return(mean(dt_s))
144 }
145
146 bs4 <- boot::boot(pp1, mean_boot, R = 1e3)
147 boot::boot.ci(bs4, type = "perc", conf = 0.95)
148
149 data_pred <- extract(stan_poisson_glm_s, "data_ppred")[[1]]

```



```

150 apply(data_pred, 2, summary)
151
152 dpp_m1_plotdf <-
153   data.frame(
154     mean = apply(data_pred, 2, mean),
155     lq = apply(data_pred, 2, quantile, 0.05),
156     uq = apply(data_pred, 2, quantile, 0.95),
157     Season = avalanches$Season
158   )
159 #####
160 #dic is bad
161 #formulae taken from https://en.wikipedia.org/wiki/Deviance_information_criterion
162 plikrar <- function(x, data) {
163   sum(dpois(data, x, log = T))
164 }
165 sampling_rates <- extract(stan_poisson_glm_s, "rate")[[1]]
166 sr_like <-
167   apply(sampling_rates, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
168 sr_like_mean <-
169   mean(sr_like)#calculate mean log likelihood of samples
170 eap <-
171   colMeans(sampling_rates)#calculate posterior means of rates (not parameters)
172 p_mean_like <-
173   sum(dpois(avalanches$Deaths, eap, log = T))#calculate log likelihood of EAP
174 dbar <- -2 * sr_like_mean#expected deviance
175 pd <- dbar + 2 * p_mean_like#calculate penalty
176 dic <- pd + dbar#give dic
177 #####
178 #prior checking
179 # dp_av <- avalanches$Deaths/avalanches$Rep.events
180 # dp_av <- dp_av[!is.nan(dp_av)]
181 # m_deaths <- mean(dp_av)
182 # xm <- dp_av - m_deaths
183 # lnfactor <- 2/(xm)^2
184 # infactor <- dp_av / m_deaths
185 # beta_p <-
186 # mfc <- exp(xm * infactor)
187 # mfc_p <- plnorm(mfc, 0, 2)
188 avno <- avalanches$Rep.events
189 avde <- avalanches$Deaths
190 mede <- mean(avde)
191 psi <- avde / mede
192 beta <- log(psi) / (avno - mean(avno))
193 psi_p <- dlnorm(psi, 0, 2)
194 beta_p <- dnorm(beta, 0, (avno - mean(avno)) ^ (-2))
195 #####
196 stan_poisson_glm_exvar <-
197   stan_model(file = "stan/poisson_glm_exvar.stan")
198
199 model_mat <- model_mat[,-1]#messes with exvar
200 out_names = colnames(model_mat)
201
202 # X_new = matrix(c(0, 1, 0, 0, 1, 0, 0, 1),
203 #               nrow = 4,
204 #               byrow = T)
205
206 X_new = matrix(c(20, 0, 1,
207                 1, 0, 0,
208                 1, 1, 0,
209                 1, 0, 1),
210               nrow = 4,
211               byrow = T)
212
213 #n_offset <- log(c(20, 1, 1, 1))
214
215 ym <- data.frame(ym = as.factor(avalanches$Season))
216 yim <- model.matrix(~ . - 1, ym)
217
218 stan_poisson_glm_exvar_data <-
219   list(
220     N = nrow(model_mat),
221     P = ncol(model_mat),
222     y = avalanches$Deaths,
223     X = model_mat,
224     n_params = c(0, sqrt(10)),
225     N_new = N_new,
226     X_new = X_new,
227     yearindmat = yim,

```

```

228     N_years = ncol(yim),
229     offset = d_offset,
230     offset_new = n_offset
231   )
232
233
234   stan_poisson_glm_exvar_s <-
235     sampling(
236       stan_poisson_glm_exvar,
237       data = stan_poisson_glm_exvar_data,
238       chains = 4,
239       control = list(adapt_delta = 0.99, max_treedepth = 15),
240       iter = 8000,
241       init_r = 0.05,
242       pars = c("lambda", "theta", "data_ppred", "rate")
243     )
244
245   post_params_exvar <-
246     extract(stan_poisson_glm_exvar_s, c("lambda"))[[1]]
247   post_params_theta <- extract(stan_poisson_glm_exvar_s, "theta")[[1]]
248   colnames(post_params_exvar) <- out_names
249   names(post_params_theta) <- "theta"
250
251   bound <- cbind(post_params_exvar, post_params_theta)
252   colnames(bound) <- c(out_names, "theta")
253   apply(exp(bound), 2, summary)
254
255   dpp <- extract(stan_poisson_glm_exvar_s, "data_ppred")[[1]]
256   apply(dpp, 2, summary)
257
258   dpp_m2_plotdf <-
259     data.frame(
260       mean = apply(dpp, 2, mean),
261       lq = apply(dpp, 2, quantile, 0.05),
262       uq = apply(dpp, 2, quantile, 0.95),
263       Season = avalanches$Season
264     )
265   #####
266   plikrar <- function(x, data) {
267     sum(dpois(data, x, log = T))
268   }
269   sampling_rates_exv <- extract(stan_poisson_glm_exvar_s, "rate")[[1]]
270   sr_like_exv <-
271     apply(sampling_rates_exv, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
272   sr_like_mean_exv <-
273     mean(sr_like_exv)#calculate mean log likelihood of samples
274   eap_exv <-
275     colMeans(sampling_rates_exv)#calculate posterior means of rates (not parameters)
276   p_mean_like_exv <-
277     sum(dpois(avalanches$Deaths, eap_exv, log = T))#calculate log likelihood of EAP
278   dbar_exv <- -2 * sr_like_mean_exv#expected deviance
279   pd_exv <- dbar_exv + 2 * p_mean_like_exv#calculate penalty
280   dic_exv <- pd_exv + dbar_exv#give dic
281   #####
282   ggplot(data = dpp_m1_plotdf, aes(x = Season)) + theme_minimal() +
283     geom_ribbon(aes(ymin = lq, ymax = uq), alpha = 0.5) + labs(title = "Posterior Predictive for Model 1", y =
284       ↪ "Number of Deaths") +
285     geom_line(aes(y = mean), size = 2, colour = "red")
286
287   ggplot(data = dpp_m2_plotdf, aes(x = Season)) + theme_minimal() +
288     geom_ribbon(aes(ymin = lq, ymax = uq), alpha = 0.5) + labs(title = "Posterior Predictive for Model 2 (extra
289       ↪ variance)", y = "Number of Deaths") +
290     geom_line(aes(y = mean), size = 2, colour = "red")
291
292   pp_mod_1 <- as.data.frame(exp_post_params)
293   pp_mod_1_long <- reshape2::melt(pp_mod_1)
294   pp_mod_2 <- as.data.frame(exp(bound))
295   pp_mod_2_long <- reshape2::melt(pp_mod_2)
296
297   ggplot(data = pp_mod_1_long, aes(x = variable, y = value)) + theme_minimal() +
298     geom_boxplot() + labs(title = "Posterior summaries for model 1", y = "Parameter value", x = "Parameter") +
299     ↪ coord_cartesian(ylim = c(0, 3))
300   ggplot(data = pp_mod_2_long, aes(x = variable, y = value)) + theme_minimal() +
301     geom_boxplot() + labs(title = "Posterior summaries for model 2 (extra variance)", y = "Parameter value", x =
302     ↪ "Parameter") + coord_cartesian(ylim = c(0, 3))

```

## 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   real theta_raw;
25   vector[P] lambda_raw;
26 }
27
28 transformed parameters{
29   vector[P] lambda = n_params[1] + n_params[2] * lambda_raw;
30   real theta = theta_hyp* theta_raw;
31   vector[N] log_rate = X * lambda + theta + offset;
```

```

32 vector[N_new] log_rate_new = X_new * lambda + theta + offset_new;
33 vector<lower=0>[N] rate = exp(log_rate);
34 }
35
36 model {
37   theta_hyp ~ uniform(0, 10);
38   lambda_raw ~ std_normal(); //implies lambda ~ normal(n_params[1], n_params[2])
39   theta_raw ~ std_normal(); //implies theta ~ normal(0, theta_hyp)
40   //lambda ~ normal(n_params[1], n_params[2]);
41   y ~ poisson_log(log_rate);
42 }
43
44 generated quantities{
45   int<lower=0> y_new[N_new] = poisson_log_rng(log_rate_new);
46   int<lower=0> data_ppred[N] = poisson_log_rng(log_rate);
47 }

```

### 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 #a
10 avalanches <- fread(file = "data/Avalanches.csv")
11 #avalanches <- avalanches[Rep.events > 0]
12 avalanches[, ']:= (EADS1 = (Season >= 1994 &
13   Season <= 2003),
14   EADS2 = (Season >= 2004))
15
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 d_offset <- rep(0, nrow(avalanches))
23
24 pglm_data <-
25   list(
26     n = nrow(avalanches),
27     w1 = avalanches$EADS1,
28     w2 = avalanches$EADS2,
29     rep = avalanches$Rep.events,
30     death = avalanches$Deaths,
31     offset = d_offset
32   )
33
34 res.a <-
35   jags.model(
36     file = "jags/poisson.jags",
37     data = pglm_data,
38     n.chains = 4,
39     quiet = T
40   )
41 update(res.a, n.iter = 1e4)
42 res.b <-
43   coda.samples(
44     res.a,
45     variable.names = c("intercept", "beta_w1", "beta_w2", "beta_rep"),
46     n.iter = 1e4
47   )
48 summary(res.b)
49 dic.samples(model = res.a,
50   n.iter = 1e4,
51   type = 'pD')
52
53 sm <- rbindlist(lapply(res.b, as.data.frame))

```

```

54
55 news_1_j <- mean(exp(sm$intercept) > 1)
56 news_2_j <- mean(exp(sm$beta_w1 + sm$intercept) > 1)
57 news_3_j <- mean(exp(sm$beta_w2 + sm$intercept) > 1)
58
59 res.a.ev <-
60   jags.model(
61     file = "jags/poisson_exvar.jags",
62     data = pglm_data,
63     n.chains = 4,
64     quiet = T
65   )
66 update(res.a, n.iter = 1e4)
67 res.b.ev <-
68   coda.samples(
69     res.a.ev,
70     variable.names = c("beta_w1", "beta_w2", "beta_rep", "theta"),
71     n.iter = 1e4
72   )
73 summary(res.b.ev)
74 dic.samples(model = res.a.ev,
75             n.iter = 1e4,
76             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_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]) <-
15      intercept + beta_rep * rep[i] + beta_w1 * w1[i] + beta_w2 * w2[i] + offset[i]
16    death[i] ~ dpois(mu[i])
17  }
18 }

```

../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 + offset[i]
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_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 = 0.95),
67     iter = 1e4,
68     init_r = 0.1
69   )
70
71 post_params_rand <-
```

```

72   extract(stan_binomial_glm_reff_s, c("beta_r"))[[1]]
73 post_params_fixed <-
74   extract(stan_binomial_glm_reff_s, c("beta_f"))[[1]]
75 post_params <- cbind(post_params_fixed, post_params_rand)
76 colnames(post_params) <-
77   c(colnames(X_fixedeff), colnames(X_randomeff))
78 ilogit_post_params <- plogis(post_params)
79 apply(ilogit_post_params, 2, summary)
80 apply(post_params, 2, summary)
81
82 dpp_rand <- extract(stan_binomial_glm_reff_s, "data_ppred")[[1]]
83 dpp_prop <- apply(dpp_rand, 1, "/", avalanches_prop$Hit)
84 apply(dpp_prop, 1, summary)
85
86 reff_coda <-
87   As.mcmc.list(stan_binomial_glm_reff_s, pars = c("beta_r", "beta_f"))
88   gelman.plot(reff_coda, ask = FALSE)
89
90 plot_diag_objects <- function(stanfit) {
91   list(
92     post = as.array(stanfit),
93     lp = log_posterior(stanfit),
94     np = nuts_params(stanfit)
95   )
96 }
97
98 plot_diag <- function(stanfit, pars) {
99   ps <- vars(starts_with(pars))
100  post <- as.array(stanfit)
101  lp <- log_posterior(stanfit)
102  np <- nuts_params(stanfit)
103  p1 <- mcmc_parcoord(post, np = np, pars = ps)
104  p2 <- mcmc_pairs(post, np = np, pars = ps)
105  p3 <- mcmc_trace(post, pars = ps, np = np)
106  p4 <- mcmc_nuts_divergence(np, lp)
107  p5 <- mcmc_nuts_energy(np)
108  list(p1, p2, p3, p4, p5)
109 }
110
111 #mcmc_trace(stan_binomial_glm_reff_s, pars = vars(starts_with("beta")))
112
113 #####
114 #sans snow fortnights
115
116 X_f_nsf <-
117   model.matrix(death_prop ~ Season + Snow_meters - 1, data = avalanches_prop)
118
119 stan_binomial_glm_reff_nsf_data <-
120   list(
121     success = success,
122     trials = trials,
123     X_f = X_f_nsf,
124     X_r = X_randomeff,
125     N = length(success),
126     P_f = ncol(X_f_nsf),
127     P_r = ncol(X_randomeff),
128     n_params = c(0, sqrt(10))
129   )
130
131 stan_binomial_glm_reff_nsf_s <-
132   sampling(
133     stan_binomial_glm_reff,
134     data = stan_binomial_glm_reff_nsf_data,
135     chains = 4,
136     control = list(adapt_delta = 0.95),
137     iter = 10000,
138     init_r = 0.1
139   )
140
141 post_params_rand_ns <-
142   extract(stan_binomial_glm_reff_nsf_s, c("beta_r"))[[1]]
143 post_params_fixed_ns <-
144   extract(stan_binomial_glm_reff_nsf_s, c("beta_f"))[[1]]
145 post_params_ns <- cbind(post_params_fixed_ns, post_params_rand_ns)
146 colnames(post_params_ns) <-
147   c(colnames(X_f_nsf), colnames(X_randomeff))
148 ilogit_post_params_ns <- plogis(post_params_ns)
149 apply(ilogit_post_params_ns, 2, summary)

```

```

150 apply(post_params_ns, 2, summary)
151
152 dpp_rand <- extract(stan_binomial_glm_reff_nsf_s, "data_ppred")[[1]]
153 dpp_prop <- apply(dpp_rand, 1, "/", avalanches_prop$Hit)
154 apply(dpp_prop, 1, summary)
155
156 #####
157 #hierarchical on station, sans snow fortnights
158 X_r_station <-
159   model.matrix(death_prop ~ Rec.station - 1, data = avalanches_prop)
160
161 stan_binomial_glm_reff_station_data <-
162   list(
163     success = success,
164     trials = trials,
165     X_f = X_f_nsf,
166     X_r = X_r_station,
167     N = length(success),
168     P_f = ncol(X_f_nsf),
169     P_r = ncol(X_r_station),
170     n_params = c(0, sqrt(10))
171   )
172
173 stan_binomial_glm_reff_station_s <-
174   sampling(
175     stan_binomial_glm_reff,
176     data = stan_binomial_glm_reff_station_data,
177     chains = 4,
178     control = list(adapt_delta = 0.9),
179     iter = 10000#,
180     #init_r = 0.1
181   )

```

## 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_raw;
17   vector[P_r] sn_vec;
18   real<lower=0,upper=10> reff_sdv;
19 }
20
21 transformed parameters{
22   vector[P_f] beta_f = n_params[2] * beta_f_raw + n_params[1];
23   vector[P_r] beta_r = reff_sdv * sn_vec;
24   vector[N] lg_p = X_f * beta_f + X_r * beta_r;
25 }
26
27 model {
28   reff_sdv ~ uniform(0, 10);
29   sn_vec ~ std_normal(); //hence beta_r ~ normal(0, reff_sdv)
30   beta_f_raw ~ std_normal(); //hence beta_f ~ normal(n_params[1], n_params[2])
31   //beta_f ~ normal(n_params[1], n_params[2]);
32   success ~ binomial(trials, inv_logit(lg_p));
33 }
34 generated quantities{
35   int data_ppred[N] = binomial_rng(trials, inv_logit(lg_p));
36   vector[N] data_prop = inv_logit(lg_p);
37 }

```

## B.3 JAGS

../jags/Q2jags.R

```

1  library(data.table)
2  library(ggplot2)
3
4  library(rjags)
5  library(coda)
6  library(bayesplot)
7
8  #####
9  #loading and eda
10 avalanches_prop <- fread(file = "data/Avalanches_part2.csv")
11 avalanches_prop[, Event_ID := NULL]
12 avalanches_prop[, Snow_meters := Snow_total / 100]
13 avalanches_prop[, Snow_fights := Snow_days / 14]
14 avalanches_prop[, death_prop := Deaths / Hit]
15 avalanches_prop[, Geo_space := as.factor(Geo_space)]
16 avalanches_prop[, Rec.station := as.factor(Rec.station)]
17 cor(avalanches_prop[, .(Season, Snow_meters, Snow_fights)])
18 #####
19
20 submin <- function(x) {
21   m <- min(x)
22   x <- x - m
23   attributes(x) <- list("scaled:submin" = m)
24   return(x)
25 }
26
27 cont_vars <- c("Snow_meters", "Snow_fights")#variables to centre
28 avalanches_prop[, (cont_vars) := lapply(.SD, scale, scale = FALSE), .SDcols = cont_vars]#centre variables
29 tm_vars <- c("Season")
30 avalanches_prop[, (tm_vars) := lapply(.SD, submin), .SDcols = tm_vars]
31
32 snow <- avalanches_prop$Snow_meters
33 fight <- avalanches_prop$Snow_fights
34 season <- avalanches_prop$Season
35 n_eff <- length(unique(avalanches_prop$Geo_space))
36 eff <- as.integer(avalanches_prop$Geo_space)
37 n <- nrow(avalanches_prop)
38 deaths <- as.integer(avalanches_prop$Deaths)

```

```

39 hit <- as.integer(avalanches_prop$Hit)
40
41 bglm_data <-
42   list(
43     n = n,
44     snow = snow,
45     fnight = fnight,
46     season = season,
47     n_eff = n_eff,
48     eff = eff,
49     deaths = deaths,
50     hit = hit
51   )
52
53 res.a <-
54   jags.model(
55     file = "jags/binom_reff.jags",
56     data = bglm_data,
57     n.chains = 4,
58     quiet = T
59   )
60 update(res.a, n.iter = 1e4)
61 res.b <-
62   coda.samples(
63     res.a,
64     variable.names = c("beta_snow", "beta_season", "beta_fnight", "reff"),
65     n.iter = 1e4
66   )
67
68 summary(res.b)
69 #####
70 snow <- avalanches_prop$Snow_meters
71 season <- avalanches_prop$Season
72 n_eff <- length(unique(avalanches_prop$Geo_space))
73 eff <- as.integer(avalanches_prop$Geo_space)
74 n <- nrow(avalanches_prop)
75 deaths <- as.integer(avalanches_prop$Deaths)
76 hit <- as.integer(avalanches_prop$Hit)
77
78 bglm_data_nf <-
79   list(
80     n = n,
81     snow = snow,
82     season = season,
83     n_eff = n_eff,
84     eff = eff,
85     deaths = deaths,
86     hit = hit
87   )
88
89 res.a_nf <-
90   jags.model(
91     file = "jags/binom_reff_nofn.jags",
92     data = bglm_data_nf,
93     n.chains = 4,
94     quiet = T
95   )
96 update(res.a_nf, n.iter = 1e4)
97 res.b_nf <-
98   coda.samples(
99     res.a_nf,
100    variable.names = c("beta_snow", "beta_season", "reff"),
101    n.iter = 1e4
102  )
103
104 summary(res.b_nf)
105 #####

```

../jags/binom\_reff.jags

```

1 model {
2   #hyperparameters
3   p_mu <- 0
4   p_tau <- 0.1
5
6   #priors

```

```

7  #beta_0 ~ dnorm(p_mu, p_tau)
8  beta_snow ~ dnorm(p_mu, p_tau)
9  beta_season ~ dnorm(p_mu, p_tau)
10 beta_fnight ~ dnorm(p_mu, p_tau)
11
12 reff_hyp ~ dunif(0, 10)
13 for (i in 1:n_eff) {
14   reff[i] ~ dnorm(0, 1 / pow(reff_hyp, 2))
15 }
16
17 #likelihood
18 for (i in 1:n) {
19   logit(p[i]) <-
20     beta_snow * snow[i] + beta_season * season[i] + beta_fnight * fnight[i] + reff[eff[i]]
21   deaths[i] ~ dbinom(p[i], hit[i])
22 }
23 }

```