

Bayesian Data Analysis Assignment 2

Benjamin Cox, S1621312

Question 1

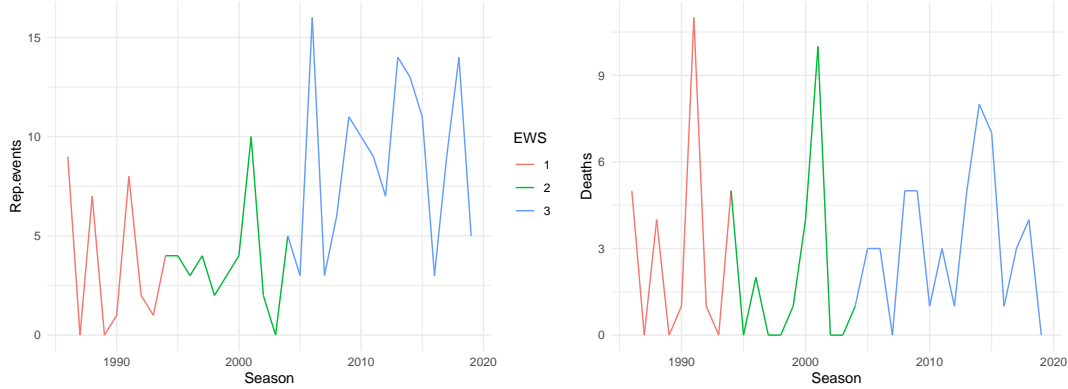


Figure 1: Plots illustrating the temporal evolution of avalanche related statistics. The EWS measure is 1 = No EADS, 2 = EADS extant, 3 = EADS online daily.

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 β_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) (β_0)	Rep.events (β_3)	EADS1TRUE (β_1)	EADS2TRUE (β_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 calculating the probabilities that the expected number of deaths is greater than one in each epoch of the EADS.

To do this we use our posterior estimates for λ in each epoch and compute the probability. We obtain the following average rates per avalanche and probabilities of aforementioned rate being greater than 1.

	No EADS	EADS	EADS online
Mean Rate per Avalanche	0.90	0.58	0.32
$P(\lambda > 1)$	0.31	0.014	0.00008

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 σ .

To make this model run well we must remove the intercept term. This is because θ 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 β_0 had a normal prior with mean 0, so θ should well compensate for it.

We are going to run 4 parallel chains with initial values drawn from a $\text{Uniform}(-0.05, 0.05)$ 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). Generally autocorrelation is not as much a problem with HMC as it is with Gibbs sampling, and we receive no warnings relating to it (we would if the ACF did not decay quickly enough). 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 θ rather than β_0 . However θ 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	$\exp(\beta_3)$	EADS1TRUE	$\exp(\beta_1)$	EADS2TRUE	$\exp(\beta_2)$	theta	$\exp(\theta)$
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.

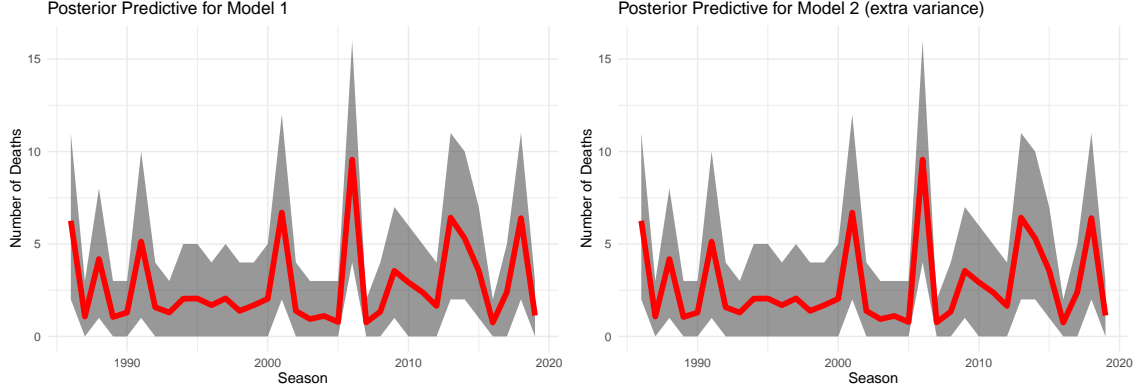


Figure 2: Posterior predictive plots for the data. Note that they are identical. the red line indicates the predictive mean, and the bands indicate the 90% credible interval.

Comparing the parameter summaries tells a similar story:

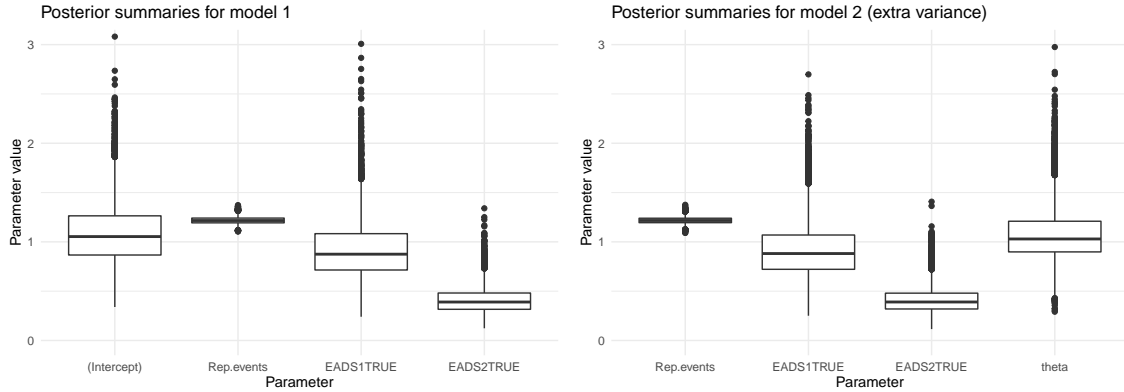


Figure 3: Posterior summaries for the parameters for each model. The parameters have been exponentiated to ease interpretation.

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 mean-centre both Snow_fnights and Snow_meters, as they are continuous variables, so centering them will both 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. Our model has the following DAG associated:

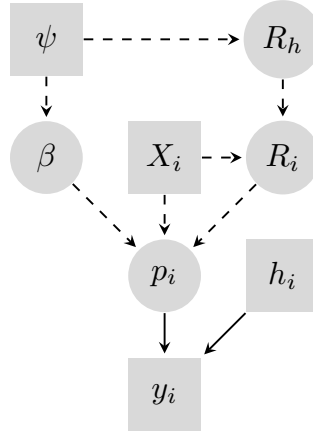


Figure 4: DAG of the random effects model described above. Note that the random effect is associated with the geographical region and hence takes information from the data. We denote stochastic nodes with circles and data nodes with squares.

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. We have implemented this model in both Stan and JAGS, but we will use the Stan samples for this analysis (both of them agree on the summaries anyway.) The Stan code is in B.2 with the main R script in B.1, whereas the JAGS code and R script is in B.3. All the code for this question is contained in these categories.

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 (β_1)	Snow_meters (β_2)	Snow_fights (β_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 (β_1)	Snow_meters (β_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_fnights.

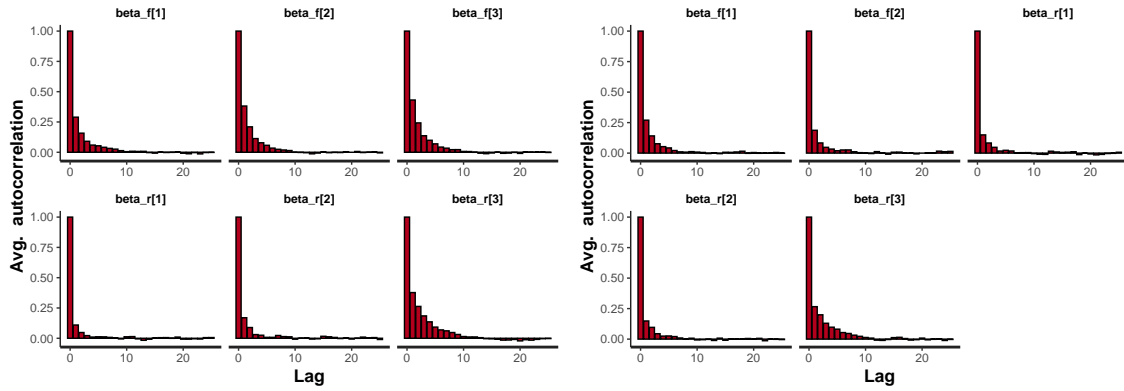


Figure 5: Average autocorrelations for our variables of interest. The left plot is before removing the permanence covariate and the right plot is after removal. Note the decrease in autocorrelation. The variables in the ACF graphs correspond to those in the corresponding above tables when read left to right, top to bottom.

Looking at the tables we come to very similar conclusions. Of particular note is that β_2 has mean equal to the sum of the means of the previous β_2 and β_3 . This is due to the high correlation between these variables.

This means that the random effects are more prominent. Notable is that the 3rd region seems more dangerous under this model.

We are now interested in the posterior distribution for the proportion of deaths expected at stations 1, 8, and 10 for the 2015 and 2018 seasons. We obtain the following means and 95% credible intervals:

	Station 1: 2015	2018	Station 8: 2015	2018	Station 10: 2015	2018
Mean	0.45	0.33	0.51	0.31	0.37	0.23
Interval	(0.24, 0.69)	(0.12, 0.63)	(0.34, 0.65)	(0.14, 0.51)	(0.19, 0.54)	(0.09, 0.41)

Table 6: Posterior summaries for the predicted proportion of casualties near the given recording stations in the given years. Data from the years has been used to construct these statistics.

We are also interested in comparing the probabilities of having a proportion of deaths greater than 60% between the stations. We obtain the following table:

After the success of this model we might be interested in fitting a model with more granular random effects. Therefore we propose a model with a random effect on the station, not on the geographical area. We will design the model with the following formula:

Year	Station 1	Station 8	Station 10
2015	0.10	0.08	0.003
2018	0.04	0.003	0.00

Table 7: Posterior probabilities of having a proportion of deaths greater than 60% near these stations in the years of interest.

$$\begin{aligned}
\text{logit}(p_i) &= \beta_1 \cdot \text{Season} + \beta_2 \cdot \text{Snow_meters} + \beta_3 \cdot \text{Snow_fnight} + R_{\text{Station}_i} \\
R_i &\equiv R_{\text{Station}_i} \sim \text{Normal}(0, R_{hyp}) \\
R_{hyp} &\sim \text{Uniform}(0, 10) \\
\beta_i &\sim \text{Normal}(0, 10) \\
\text{Deaths}_i &\sim \text{Binomial}(\text{Hit}_i, p_i)
\end{aligned}$$

This model is very similar to our previous model, but we expect more finely tuned results. We will also be able to assess the relative danger of stations (somewhat).

We run the sampler with the same settings as above, checking BGR and autocorrelation, and obtain the following summary statistics:

	Season	Snow_meters	Rec.station1	Rec.station2	Rec.station3	Rec.station4	Rec.station5
Min.	-0.77	-0.97	-1.62	-16.32	-6.27	-6.44	-2.74
1st Qu.	-0.30	-0.23	0.77	-1.22	-1.12	-1.64	-0.05
Median	-0.21	-0.11	1.55	-0.37	-0.50	-1.03	0.33
Mean	-0.21	-0.12	1.78	-0.59	-0.60	-1.12	0.36
3rd Qu.	-0.12	0.00	2.50	0.23	0.00	-0.48	0.75
Max.	0.31	0.56	14.07	6.94	3.05	1.49	3.33

	Rec.station6	Rec.station7	Rec.station8	Rec.station9	Rec.station10	Rec.station11
Min.	-6.73	-5.21	-1.92	-4.96	-4.33	-3.63
1st Qu.	-1.25	0.05	-0.01	-1.60	-0.69	-0.74
Median	-0.62	0.72	0.33	-1.04	-0.16	-0.29
Mean	-0.72	0.96	0.35	-1.11	-0.20	-0.32
3rd Qu.	-0.09	1.65	0.70	-0.53	0.31	0.10
Max.	2.63	18.10	3.29	1.70	3.54	2.58

Table 8: Posterior summaries for the third binomial random effects model, which has the effects on the recording station. This model has been adjusted to compensate for collinearity between Snow_meters and Snow_fnights.

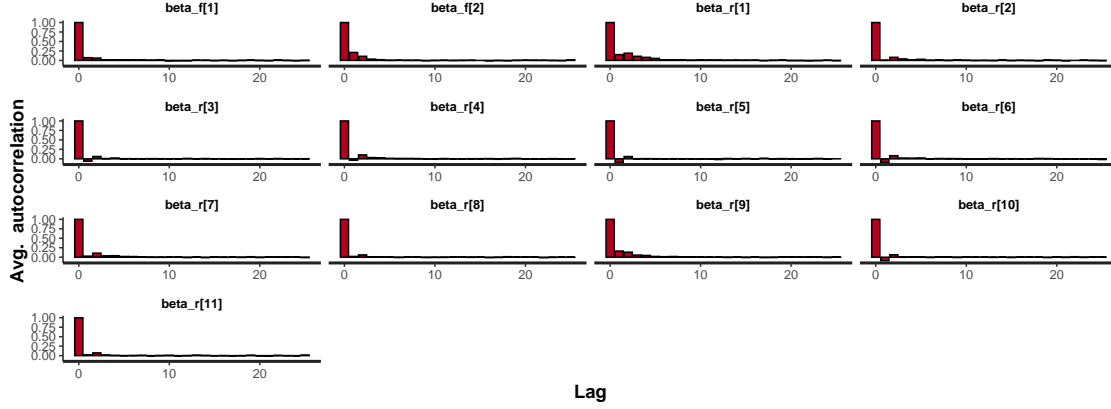


Figure 6: Average autocorrelations for our variables of interest. This plot explores the autocorrelation in the sampling for the station level model. We note the extremely rapid decay in autocorrelation for all variables of interest, indicating good sampling. The variables in the ACF graphs correspond to those in the above table when read left to right, top to bottom.

We also obtain the following means and 95% credible intervals for the proportion of casualties near the given stations in the given years.

	Station 1: 2015	2018	Station 8: 2015	2018	Station 10: 2015	2018
Mean	0.72	0.61	0.58	0.37	0.47	0.30
Interval	(0.34, 0.99)	(0.18, 0.98)	(0.12, 0.68)	(0.14, 0.51)	(0.15, 0.79)	(0.06, 0.66)

Table 9: Posterior summaries for the predicted proportion of casualties near the given recording stations in the given years. Data from the years has been used to construct these statistics. This model has the random effect on the station, not geographical area as previous.

These have much wider credible intervals, reflecting the relative lack of data that we have for each station. This model does not take into account geographical similarity between stations which lead to similar casualty proportions. We also often have only one point of data per station per year, leading to the estimates being overfit to the data. We have 43 data points and 13 parameters.

I would be inclined to say that we are overfitting on the station level, thus we should incorporate variability from the geographical level as well. This will add more parameters, but they will be more constrained on each other rather than biased on the data.

Comparing the two models above using the DIC we obtain a DIC of 91.04 for the model with effects on geographical area and a DIC of 85.21 for the model with effects of the nearest station. This is due to the better fit: the penalty terms are 4.171 and 8.854 respectively, showing the effect of the increased number of parameters. Based on this we would prefer the station level model. However both of them lead to similar conclusions.

We could have a random effect on the recording station drawn from the geographical area drawn from an overarching distribution. We propose the following model:

$$\begin{aligned}
\sigma_1, \sigma_2 &\sim \Gamma(1, 0.1), \\
\mu_{\text{Geo.space}_i} &\sim \text{Normal}(0, \sigma_2^{-2}) \\
R_{(\text{Rep.station, Geo.space})_i} &\sim \text{Normal}(\mu_{\text{Geo.space}_i}, \sigma_1^{-2}), \\
\text{logit}(p_i) &= \beta_1 \cdot \text{Season} + \beta_2 \cdot \text{Snow_meters} + R_{(\text{Rep.station, Geo.space})_i}, \\
\text{Deaths}_i &\sim \text{Binomial}(\text{Hit}_i, p_i),
\end{aligned}$$

which we have encoded in JAGS in `binom_doublereff.jags`. This model works rather well. It has a DIC of 86.82, which is slightly higher than that of the station level model, but more than compensates for it

in the information that we can glean from it, as we can observe both geographical and station level effects, rather than just station level. The model has the following DAG associated with it:

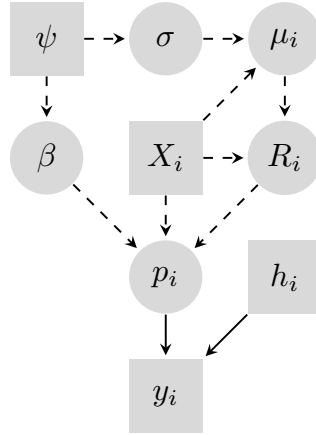


Figure 7: DAG of the random effects model described above. Note that the random effect is associated with the geographical region and station and hence takes information from the data twice.

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 = 1e4,
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$Season
158   )
159
160 lr_data <- extract(stan_poisson_glm_s, "log_rate")[[1]]
161 r_data <- exp(lr_data)
162 r_data_pe <- apply(r_data, 1, '/', avalanches$Rep.events)
163
164 r_data_b <- r_data_pe[avalanches$Season < 1994]
165 r_data_do <- r_data_pe[avalanches$EADS1 == TRUE]
166 r_data_o <- r_data_pe[avalanches$EADS2 == TRUE]
167
168 r_data_b <- unlist(r_data_b)
169 r_data_b <- r_data_b[!is.infinite(r_data_b)]
170 mean(r_data_b >= 1)
171 mean(r_data_b)
172
173 r_data_do <- unlist(r_data_do)
174 r_data_do <- r_data_do[!is.infinite(r_data_do)]
175 mean(r_data_do >= 1)
176 mean(r_data_do)
177
178 r_data_o <- unlist(r_data_o)
179 r_data_o <- r_data_o[!is.infinite(r_data_o)]
180 mean(r_data_o >= 1)
181 mean(r_data_o)
182 #####
183 #dic is bad
184 #formulae taken from https://en.wikipedia.org/wiki/Deviance_information_criterion
185 plikrar <- function(x, data) {
186   sum(dpois(data, x, log = T))
187 }
188 sampling_rates <- extract(stan_poisson_glm_s, "rate")[[1]]
189 sr_like <-
190   apply(sampling_rates, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
191 sr_like_mean <-
192   mean(sr_like)#calculate mean log likelihood of samples
193 eap <-
194   colMeans(sampling_rates)#calculate posterior means of rates (not parameters)
195 p_mean_like <-
196   sum(dpois(avalanches$Deaths, eap, log = T))#calculate log likelihood of EAP
197 dbar <- -2 * sr_like_mean#expected deviance
198 pd <- dbar + 2 * p_mean_like#calculate penalty
199 dic <- pd + dbar#give dic
200 #####
201 #prior checking
202 # dp_av <- avalanches$Deaths/avalanches$Rep.events
203 # dp_av <- dp_av[!is.nan(dp_av)]
204 # m_deaths <- mean(dp_av)
205 # xm <- dp_av - m_deaths
206 # lnfactor <- 2/(xm)^2
207 # infactor <- dp_av / m_deaths
208 # beta_p <-
209 # mfc <- exp(xm * infactor)
210 # mfc_p <- plnorm(mfc, 0, 2)
211 avno <- avalanches$Rep.events
212 avde <- avalanches$Deaths
213 mede <- mean(avde)
214 psi <- avde / mede
215 beta <- log(psi) / (avno - mean(avno))
216 psi_p <- dlnorm(psi, 0, 2)
217 beta_p <- dnorm(beta, 0, (avno - mean(avno)) ^ (-2))
218 #####
219 stan_poisson_glm_exvar <-
220   stan_model(file = "stan/poisson_glm_exvar.stan")
221
222 model_mat <- model_mat[,-1]#messes with exvar
223 out_names = colnames(model_mat)
224
225 # X_new = matrix(c(0, 1, 0, 0, 1, 0, 0, 1),
226 #                 nrow = 4,
227 #                 byrow = T)

```

```

228
229 X_new = matrix(c(20, 0, 1,
230                 1, 0, 0,
231                 1, 1, 0,
232                 1, 0, 1),
233                nrow = 4,
234                byrow = T)
235
236 #n_offset <- log(c(20, 1, 1, 1))
237
238 ym <- data.frame(ym = as.factor(avalanches$Season))
239 yim <- model.matrix( ~ . - 1, ym)
240
241 stan_poisson_glm_exvar_data <-
242   list(
243     N = nrow(model_mat),
244     P = ncol(model_mat),
245     y = avalanches$Deaths,
246     X = model_mat,
247     n_params = c(0, sqrt(10)),
248     N_new = N_new,
249     X_new = X_new,
250     yearindmat = yim,
251     N_years = ncol(yim),
252     offset = d_offset,
253     offset_new = n_offset
254   )
255
256
257 stan_poisson_glm_exvar_s <-
258   sampling(
259     stan_poisson_glm_exvar,
260     data = stan_poisson_glm_exvar_data,
261     chains = 4,
262     control = list(adapt_delta = 0.99, max_treedepth = 15),
263     iter = 8000,
264     init_r = 0.05,
265     pars = c("lambda", "theta", "data_ppred", "rate")
266   )
267
268 post_params_exvar <-
269   extract(stan_poisson_glm_exvar_s, c("lambda"))[[1]]
270 post_params_theta <- extract(stan_poisson_glm_exvar_s, "theta")[[1]]
271 colnames(post_params_exvar) <- out_names
272 names(post_params_theta) <- "theta"
273
274 bound <- cbind(post_params_exvar, post_params_theta)
275 colnames(bound) <- c(out_names, "theta")
276 apply(exp(bound), 2, summary)
277
278 dpp <- extract(stan_poisson_glm_exvar_s, "data_ppred")[[1]]
279 apply(dpp, 2, summary)
280
281 dpp_m2_plotdf <-
282   data.frame(
283     mean = apply(dpp, 2, mean),
284     lq = apply(dpp, 2, quantile, 0.05),
285     uq = apply(dpp, 2, quantile, 0.95),
286     Season = avalanches$Season
287   )
288 #####
289 plikrar <- function(x, data) {
290   sum(dpois(data, x, log = T))
291 }
292 sampling_rates_exv <- extract(stan_poisson_glm_exvar_s, "rate")[[1]]
293 sr_like_exv <-
294   apply(sampling_rates_exv, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
295 sr_like_mean_exv <-
296   mean(sr_like_exv)#calculate mean log likelihood of samples
297 eap_exv <-
298   colMeans(sampling_rates_exv)#calculate posterior means of rates (not parameters)
299 p_mean_like_exv <-
300   sum(dpois(avalanches$Deaths, eap_exv, log = T))#calculate log likelihood of EAP
301 dbar_exv <- -2 * sr_like_mean_exv#expected deviance
302 pd_exv <- dbar_exv + 2 * p_mean_like_exv#calculate penalty
303 dic_exv <- pd_exv + dbar_exv#give dic
304 #####
305 ggplot(data = dpp_m1_plotdf, aes(x = Season)) + theme_minimal() +

```

```

306 geom_ribbon(aes(ymin = lq, ymax = uq), alpha = 0.5) + labs(title = "Posterior Predictive for Model 1", y =
    ↪ "Number of Deaths") +
307 geom_line(aes(y = mean), size = 2, colour = "red")
308
309 ggplot(data = dpp_m2_plotdf, aes(x = Season)) + theme_minimal() +
310 geom_ribbon(aes(ymin = lq, ymax = uq), alpha = 0.5) + labs(title = "Posterior Predictive for Model 2 (extra
    ↪ variance)", y = "Number of Deaths") +
311 geom_line(aes(y = mean), size = 2, colour = "red")
312
313 pp_mod_1 <- as.data.frame(exp_post_params)
314 pp_mod_1_long <- reshape2::melt(pp_mod_1)
315 pp_mod_2 <- as.data.frame(exp(bound))
316 pp_mod_2_long <- reshape2::melt(pp_mod_2)
317
318 ggplot(data = pp_mod_1_long, aes(x = variable, y = value)) + theme_minimal() +
319 geom_boxplot() + labs(title = "Posterior summaries for model 1", y = "Parameter value", x = "Parameter") +
    ↪ coord_cartesian(ylim = c(0, 3))
320 ggplot(data = pp_mod_2_long, aes(x = variable, y = value)) + theme_minimal() +
321 geom_boxplot() + labs(title = "Posterior summaries for model 2 (extra variance)", y = "Parameter value", x =
    ↪ "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 #####
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 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[, Year := Season]
21 avalanches_prop[, death_prop := Deaths / Hit]
22 avalanches_prop[, Geo_space := as.factor(Geo_space)]
23 avalanches_prop[, Rec.station := as.factor(Rec.station)]
24 cor(avalanches_prop[, .(Season, Snow_meters, Snow_fights)])
25 #####
26 stan_binomial_glm_reff <-
27   stan_model(file = "stan/binomial_glm_ranomeffects.stan")
28
29 submin <- function(x) {
30   m <- min(x)
31   x <- x - m
32   attributes(x) <- list("scaled:submin" = m)
33   return(x)
34 }
35
36 probcomp_geq <- function(x, value){
37   mean(x >= value)
38 }
39
40 probcomp_leq <- function(x, value){
41   mean(x <= value)
42 }
43
44 cont_vars <- c("Snow_meters", "Snow_fights")#variables to centre
45 avalanches_prop[, (cont_vars) := lapply(.SD, scale, scale = FALSE), .SDcols = cont_vars]#centre variables
46 tm_vars <- c("Season")
```

```

47 avalanches_prop[, (tm_vars) := lapply(.SD, submin), .SDcols = tm_vars]
48
49
50 X_fixedeff <-
51   model.matrix(death_prop ~ Season + Snow_meters + Snow_fnnights - 1, data = avalanches_prop)
52 X_randomeff <-
53   model.matrix(death_prop ~ Geo_space - 1, data = avalanches_prop)
54 success <- avalanches_prop[, Deaths]
55 trials <- avalanches_prop[, Hit]
56
57
58 stan_binomial_glm_reff_data <-
59   list(
60     success = success,
61     trials = trials,
62     X_f = X_fixedeff,
63     X_r = X_randomeff,
64     N = length(success),
65     P_f = ncol(X_fixedeff),
66     P_r = ncol(X_randomeff),
67     n_params = c(0, sqrt(10))
68   )
69
70 stan_binomial_glm_reff_s <-
71   sampling(
72     stan_binomial_glm_reff,
73     data = stan_binomial_glm_reff_data,
74     chains = 4,
75     control = list(adapt_delta = 0.95),
76     iter = 1e4,
77     init_r = 0.1
78   )
79
80 post_params_rand <-
81   extract(stan_binomial_glm_reff_s, c("beta_r"))[[1]]
82 post_params_fixed <-
83   extract(stan_binomial_glm_reff_s, c("beta_f"))[[1]]
84 post_params <- cbind(post_params_fixed, post_params_rand)
85 colnames(post_params) <-
86   c(colnames(X_fixedeff), colnames(X_randomeff))
87 ilogit_post_params <- plogis(post_params)
88 apply(ilogit_post_params, 2, summary)
89 apply(post_params, 2, summary)
90
91 dpp_rand <- extract(stan_binomial_glm_reff_s, "data_ppred")[[1]]
92 dpp_prop <- apply(dpp_rand, 1, "/", avalanches_prop$Hit)
93 apply(dpp_prop, 1, summary)
94
95 reff_coda <-
96   As.mcmc.list(stan_binomial_glm_reff_s, pars = c("beta_r", "beta_f"))
97 gelman.plot(reff_coda, ask = FALSE)
98
99 plot_diag_objects <- function(stanfit) {
100   list(
101     post = as.array(stanfit),
102     lp = log_posterior(stanfit),
103     np = nuts_params(stanfit)
104   )
105 }
106
107 plot_diag <- function(stanfit, pars) {
108   ps <- vars(starts_with(pars))
109   post <- as.array(stanfit)
110   lp <- log_posterior(stanfit)
111   np <- nuts_params(stanfit)
112   p1 <- mcmc_parcoord(post, np = np, pars = ps)
113   p2 <- mcmc_pairs(post, np = np, pars = ps)
114   p3 <- mcmc_trace(post, pars = ps, np = np)
115   p4 <- mcmc_nuts_divergence(np, lp)
116   p5 <- mcmc_nuts_energy(np)
117   list(p1, p2, p3, p4, p5)
118 }
119
120 #mcmc_trace(stan_binomial_glm_reff_s, pars = vars(starts_with("beta")))
121
122 #####
123 #sans snow fortnights
124 varofint <- avalanches_prop[(Rec.station %in% c(1, 8, 10)) & (Year %in% c(2015, 2018))]

```

```

125 ids <- unique(varofint, by = c("Rec.station", "Year"))$Event_ID
126 index <- which(avalanches_prop$Event_ID %in% ids)
127
128 X_f_nsf <-
129   model.matrix(death_prop ~ Season + Snow_meters - 1, data = avalanches_prop)
130
131 stan_binomial_glm_reff_nsf_data <-
132   list(
133     success = success,
134     trials = trials,
135     X_f = X_f_nsf,
136     X_r = X_randomeff,
137     N = length(success),
138     P_f = ncol(X_f_nsf),
139     P_r = ncol(X_randomeff),
140     n_params = c(0, sqrt(10))
141   )
142
143 stan_binomial_glm_reff_nsf_s <-
144   sampling(
145     stan_binomial_glm_reff,
146     data = stan_binomial_glm_reff_nsf_data,
147     chains = 4,
148     control = list(adapt_delta = 0.95),
149     iter = 10000,
150     init_r = 0.1
151   )
152
153
154
155 post_params_rand_ns <-
156   extract(stan_binomial_glm_reff_nsf_s, c("beta_r"))[[1]]
157 post_params_fixed_ns <-
158   extract(stan_binomial_glm_reff_nsf_s, c("beta_f"))[[1]]
159 post_params_ns <- cbind(post_params_fixed_ns, post_params_rand_ns)
160 colnames(post_params_ns) <-
161   c(colnames(X_f_nsf), colnames(X_randomeff))
162 ilogit_post_params_ns <- plogis(post_params_ns)
163 apply(ilogit_post_params_ns, 2, summary)
164 apply(post_params_ns, 2, summary)
165
166 dpp_rand_nf <- extract(stan_binomial_glm_reff_nsf_s, "data_prop")[[1]]
167 apply(dpp_rand_nf, 2, summary)
168 dpp_ofint <- dpp_rand_nf[, index]
169 apply(dpp_ofint, 2, mean)
170 apply(dpp_ofint, 2, quantile, c(0.025, 0.975))
171 apply(dpp_ofint > 0.6, 2, mean)
172
173 #####
174 #hierarchical on station, sans snow fortnights
175 X_r_station <-
176   model.matrix(death_prop ~ Rec.station - 1, data = avalanches_prop)
177
178 stan_binomial_glm_reff_station_data <-
179   list(
180     success = success,
181     trials = trials,
182     X_f = X_f_nsf,
183     X_r = X_r_station,
184     N = length(success),
185     P_f = ncol(X_f_nsf),
186     P_r = ncol(X_r_station),
187     n_params = c(0, sqrt(10))
188   )
189
190 stan_binomial_glm_reff_station_s <-
191   sampling(
192     stan_binomial_glm_reff,
193     data = stan_binomial_glm_reff_station_data,
194     chains = 4,
195     control = list(adapt_delta = 0.9),
196     iter = 10000#,
197     #init_r = 0.1
198   )
199
200 post_params_rand_ns_stat <-
201   extract(stan_binomial_glm_reff_station_s, c("beta_r"))[[1]]
202 post_params_fixed_ns_stat <-

```

```

203   extract(stan_binomial_glm_reff_station_s, c("beta_f"))[[1]]
204 post_params_ns_stat <- cbind(post_params_fixed_ns_stat, post_params_rand_ns_stat)
205 colnames(post_params_ns_stat) <-
206   c(colnames(X_f_nsf), colnames(X_r_station))
207 ilogit_post_params_ns_stat <- plogis(post_params_ns_stat)
208 apply(ilogit_post_params_ns_stat, 2, summary)
209 apply(post_params_ns_stat, 2, summary)
210
211 dpp_rand_ns_stat <- extract(stan_binomial_glm_reff_station_s, "data_prop")[[1]]
212 apply(dpp_rand_ns_stat, 2, summary)
213 dpp_ofintns_stat <- dpp_rand_ns_stat[,index]
214 apply(dpp_ofintns_stat, 2, mean)
215 apply(dpp_ofintns_stat, 2, quantile, c(0.025, 0.975))
216 apply(dpp_ofintns_stat > 0.6, 2, mean)

```

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_fnights := 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_fnights)])
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_fnights")#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$Snow_meters
33 fnight <- avalanches_prop$Snow_fnights
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
106   dic.samples(model = res.a_nf,
107     n.iter = 1e4,
108     type = 'pD')
109   #####
110   snow <- avalanches_prop$Snow_meters
111   season <- avalanches_prop$Season
112   #n_eff <- length(unique(avalanches_prop$Geo_space))
113   #eff <- as.integer(avalanches_prop$Geo_space)
114   eff_stat <- as.integer(avalanches_prop$Rec.station)
115   n_eff_stat <- length(unique(eff_stat))
116   n <- nrow(avalanches_prop)
117   deaths <- as.integer(avalanches_prop$Deaths)
118   hit <- as.integer(avalanches_prop$Hit)
119
120   bglm_data_nf_stat <-
121     list(
122       n = n,
123       snow = snow,
124       season = season,
125       n_eff = n_eff_stat,
126       eff = eff_stat,
127       deaths = deaths,
128       hit = hit
129     )
130
131   res.a_nf_stat <-
132     jags.model(
133       file = "jags/binom_reff_nofn.jags",
134       data = bglm_data_nf_stat,

```

```

135     n.chains = 4,
136     quiet = T
137   )
138   update(res.a_nf_stat, n.iter = 1e4)
139   res.b_nf_stat <-
140     coda.samples(
141       res.a_nf_stat,
142       variable.names = c("beta_snow", "beta_season", "reff"),
143       n.iter = 1e4
144     )
145
146   summary(res.b_nf_stat)
147
148   dic.samples(model = res.a_nf_stat,
149     n.iter = 1e4,
150     type = 'pD')
151   #####
152   snow <- avalanches_prop$Snow_meters
153   season <- avalanches_prop$Season
154   #n_eff <- length(unique(avalanches_prop$Geo_space))
155   #eff <- as.integer(avalanches_prop$Geo_space)
156   stations <- as.integer(avalanches_prop$Rec.station)
157   geos <- as.integer(avalanches_prop$Geo_space)
158   n_station <- length(unique(stations))
159   n_geo <- length(unique(geos))
160   n <- nrow(avalanches_prop)
161   deaths <- as.integer(avalanches_prop$Deaths)
162   hit <- as.integer(avalanches_prop$Hit)
163
164   bglm_data_nf_statgeo <-
165     list(
166       n = n,
167       snow = snow,
168       season = season,
169       geos = geos,
170       stations = stations,
171       n_station = n_station,
172       n_geo = n_geo,
173       deaths = deaths,
174       hit = hit
175     )
176
177   res.a_nf_statgeo <-
178     jags.model(
179       file = "jags/binom_doublereff.jags",
180       data = bglm_data_nf_statgeo,
181       n.chains = 4,
182       quiet = T
183     )
184   update(res.a_nf_statgeo, n.iter = 1e4)
185   res.b_nf_statgeo <-
186     coda.samples(
187       res.a_nf_statgeo,
188       variable.names = c("beta_snow", "beta_season", "r_eff_geo", "r_eff_statgeo"),
189       n.iter = 1e4
190     )
191
192   summary(res.b_nf_statgeo)
193
194   dic.samples(model = res.a_nf_statgeo,
195     n.iter = 1e4,
196     type = 'pD')

```

../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 }

```

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

```

../jags/binom_doublereff.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
11   sigma_1 ~ dgamma(1, 0.1)
12   sigma_2 ~ dgamma(1, 0.1)
13
14   for(i in 1:n_geo){
15     r_eff_geo[i] ~ dnorm(0, sigma_1)
16   }
17
18   for(i in 1:n_station){
19     r_eff_statgeo[i] ~ dnorm(r_eff_geo[geos[i]], sigma_2)
20   }
21
22   #likelihood
23   for (i in 1:n) {
24     logit(p[i]) <-
25       beta_snow * snow[i] + beta_season * season[i] + r_eff_statgeo[stations[i]]
26     deaths[i] ~ dbinom(p[i], hit[i])
27   }
28 }

```