Bayesian Data Analysis Assignment 2

Benjamin Cox, S1621312

Question 1

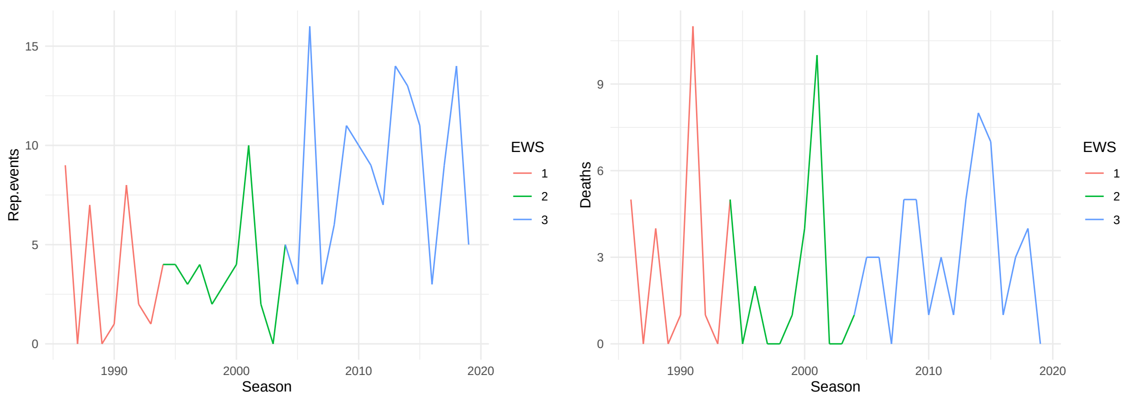

Figure 1: Plots illustrating the temporal evolution of avalanche related statistics. The EWS measure is 1 = No EADS, 2 = EADS extant, 3 = EADS online daily.

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:

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

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 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

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

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

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

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

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

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 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

	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.

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:

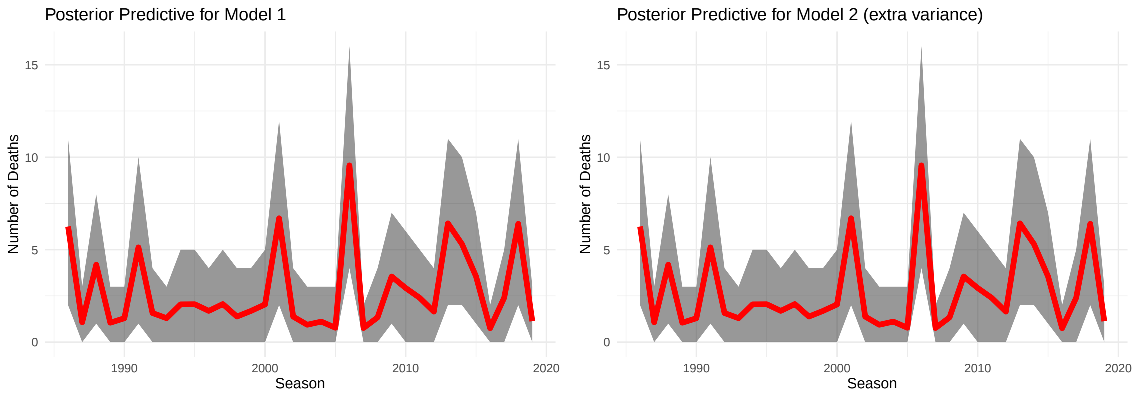

Figure 2: Posterior predictive plots for the data. Note that they are identical. the red line indicates the predictive mean, and the bands indicate the 90% credible interval.

Comparing the parameter summaries tells a similar story:

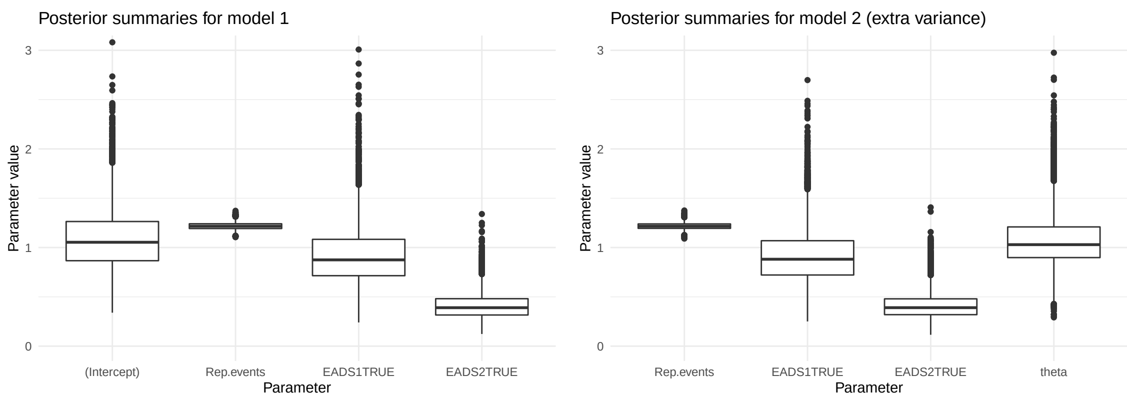

Figure 3: Posterior summaries for the parameters for each model. The parameters have been exponentiated to ease interpretation.

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 fortnights. 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.

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_fnights (β_3)	Geo_space1 (R_1)	Geo_space2 (R_2)	Geo_space3 (R
Min.	-0.69	-1.01	-0.71	-2.12	-1.70	-3.
1st Qu.	-0.27	-0.33	-0.04	-0.09	-0.26	-0.
Median	-0.19	-0.19	0.08	0.10	-0.05	-0.
Mean	-0.18	-0.18	0.08	0.18	-0.07	-0.
3rd Qu.	-0.10	-0.04	0.19	0.45	0.11	-0.
Max.	0.43	0.89	0.80	2.83	1.36	1.

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_fnights 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.

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:

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.

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:

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

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 and obtain the following summary statistics:

	Season	Snow_meters	Rec.station1	Rec.station2	Rec.station3	Rec.station4	Rec.station
Min.	-0.77	-0.97	-1.62	-16.32	-6.27	-6.44	-2.
1st Qu.	-0.30	-0.23	0.77	-1.22	-1.12	-1.64	-0.0
Median	-0.21	-0.11	1.55	-0.37	-0.50	-1.03	0.3
Mean	-0.21	-0.12	1.78	-0.59	-0.60	-1.12	0.3
3rd Qu.	-0.12	0.00	2.50	0.23	0.00	-0.48	0.7
Max.	0.31	0.56	14.07	6.94	3.05	1.49	3.3

	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.

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:

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

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.

A Code for Question 1

A.1 R

../Q1.R

```
library(data.table)
library(ggplot2)

library(rstan)
rstan_options(auto_write = TRUE)
#options(mc.cores = parallel::detectCores())
Sys.setenv(LOCAL_CPPFLAGS = '-march=corei7 -mtune=corei7')
options(mc.cores = 1)
library(rstanarm)
library(coda)
library(bayesplot)


#####
#a
avalanches <- fread(file = "data/Avalanches.csv")
avalanches[, ':=' (EADS1 = (Season >= 1994 &
                             Season <= 2003),
                   EADS2 = (Season >= 2004))]

avalanches[Season %in% c(1986, 1994, 2004)]

avalanches[, EWS := 1 + EADS1 + 2 * EADS2]
avalanches[, EWS := as.factor(EWS)]

base_plot <-
  ggplot(data = as.data.frame(avalanches), aes(colour = EWS)) + theme_minimal()
base_plot + geom_line(aes(x = Season, y = Rep.events, group = F))
base_plot + geom_line(aes(x = Season, y = Deaths, group = F))
base_plot + geom_boxplot(aes(x = EWS, y = Deaths), colour = "black")

#avalanches <- avalanches[Rep.events > 0]
cor_boot <- function(data, index) {
  dt_s <- data[index, ]
  return(cor(dt_s))
}

cor(avalanches[(EADS1 == FALSE &
                  EADS2 == FALSE), .(Rep.events, Deaths)])
cor(avalanches[EADS1 == TRUE, .(Rep.events, Deaths)])
cor(avalanches[EADS2 == TRUE, .(Rep.events, Deaths)])

bs1 <- boot::boot(avalanches[(EADS1 == FALSE &
                                EADS2 == FALSE),
                               .(Rep.events, Deaths)]
                  , cor_boot, R = 1e3)
bs2 <- boot::boot(avalanches[(EADS1 == TRUE),
                               .(Rep.events, Deaths)]
                  , cor_boot, R = 1e3)
bs3 <- boot::boot(avalanches[(EADS2 == TRUE),
                               .(Rep.events, Deaths)]
                  , cor_boot, R = 1e3)
boot::boot.ci(bs1,
              index = 2,
              type = "perc",
              conf = 0.9)
boot::boot.ci(bs2,
              index = 2,
              type = "perc",
```

```
              conf = 0.9)
boot::boot.ci(bs3,
              index = 2,
              type = "perc",
              conf = 0.9)
#####
#b
to_model <- avalanches[, .(Rep.events, Deaths, EADS1, EADS2)]
model_mat <-
  model.matrix(Deaths ~ ., data = to_model)#no intercept as cannot have deaths without avalanche
#d_offset <- log(avalanches$Rep.events)
d_offset <- rep(0, nrow(avalanches))
model_mat <- model_mat[,]
out_names = colnames(model_mat)
#no need to centre as discrete

#new data

# X_new = matrix(c(1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1),
#                 nrow = 4,
#                 byrow = T)

X_new = matrix(c(1, 20, 0, 1,
                 1, 1, 0, 0,
                 1, 1, 1, 0,
                 1, 1, 0, 1),
               nrow = 4,
               byrow = T)
#n_offset <- log(c(20, 1, 1, 1))
n_offset <- rep(0, nrow(X_new))

N_new = nrow(X_new)
#check, should be similar
f_glm <-
  glm(Deaths ~ ., data = to_model, family = poisson(link = "log"))


stan_poisson_glm <- stan_model(file = "stan/poisson_glm.stan")
stan_poisson_glm_data <-
  list(
    N = nrow(model_mat),
    P = ncol(model_mat),
    y = avalanches$Deaths,
    X = model_mat,
    n_params = c(0, 1e2),
    N_new = N_new,
    X_new = X_new,
    offset = d_offset,
    offset_new = n_offset
  )


stan_poisson_glm_s <-
  sampling(
    stan_poisson_glm,
    data = stan_poisson_glm_data,
    chains = 7,
    control = list(adapt_delta = 0.8),
    iter = 1e5,
    init_r = 0.1
  )

post_params <- extract(stan_poisson_glm_s, "lambda")[[1]]
colnames(post_params) <- out_names
exp_post_params <- exp(post_params)
apply(exp_post_params, 2, summary)
```

```r
apply(post_params, 2, summary)

news_1 <- mean(exp(post_params[, 1]) > 1)
news_2 <- mean(exp(post_params[, 1] + post_params[, 2]) > 1)
news_3 <- mean(exp(post_params[, 1] + post_params[, 3]) > 1)


p_pred <- extract(stan_poisson_glm_s, "y_new")[[1]]
mean(p_pred[, 1] < 15)
mean(p_pred[, 2] > 1)
mean(p_pred[, 3] > 1)
mean(p_pred[, 4] > 1)

pp1 <- p_pred[,1] < 15

mean_boot <- function(data, index) {
  dt_s <- data[index]
  return(mean(dt_s))
}

bs4 <- boot::boot(pp1, mean_boot, R = 1e3)
boot::boot.ci(bs4, type = "perc", conf = 0.95)

data_pred <- extract(stan_poisson_glm_s, "data_ppred")[[1]]
apply(data_pred, 2, summary)

dpp_m1_plotdf <-
  data.frame(
    mean = apply(data_pred, 2, mean),
    lq = apply(data_pred, 2, quantile, 0.05),
    uq = apply(data_pred, 2, quantile, 0.95),
    Season = avalanches$Season
  )
#####
#dic is bad
#formulae taken from https://en.wikipedia.org/wiki/Deviance_information_criterion
plikrar <- function(x, data) {
  sum(dpois(data, x, log = T))
}
sampling_rates <- extract(stan_poisson_glm_s, "rate")[[1]]
sr_like <-
  apply(sampling_rates, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
sr_like_mean <-
  mean(sr_like)#calculate mean log likelihood of samples
eap <-
  colMeans(sampling_rates)#calculate posterior means of rates (not parameters)
p_mean_like <-
  sum(dpois(avalanches$Deaths, eap, log = T))#calculate log likelihood of EAP
dbar <- -2 * sr_like_mean#expected deviance
pd <- dbar + 2 * p_mean_like#calculate penalty
dic <- pd + dbar#give dic
#####
#prior checking
# dp_av <- avalanches$Deaths/avalanches$Rep.events
# dp_av <- dp_av[!is.nan(dp_av)]
# m_deaths <- mean(dp_av)
# xm <- dp_av - m_deaths
# lnfactor <- 2/(xm)^2
# inffactor <- dp_av / m_deaths
# beta_p <-
# mfc <- exp(xm * inffactor)
# mfc_p <- plnorm(mfc, 0, 2)
avno <- avalanches$Rep.events
avde <- avalanches$Deaths
mede <- mean(avde)
psi <- avde / mede
```

```
beta <- log(psi) / (avno - mean(avno))
psi_p <- dlnorm(psi, 0, 2)
beta_p <- dnorm(beta, 0, (avno - mean(avno)) ^ (-2))
#####
stan_poisson_glm_exvar <-
  stan_model(file = "stan/poisson_glm_exvar.stan")

model_mat <- model_mat[,-1]#messes with exvar
out_names = colnames(model_mat)

# X_new = matrix(c(0, 1, 0, 0, 1, 0, 0, 1),
#                 nrow = 4,
#                 byrow = T)

X_new = matrix(c(20, 0, 1,
                 1, 0, 0,
                 1, 1, 0,
                 1, 0, 1),
               nrow = 4,
               byrow = T)

#n_offset <- log(c(20, 1, 1, 1))

ym <- data.frame(ym = as.factor(avalanches$Season))
yim <- model.matrix( ~ . - 1, ym)

stan_poisson_glm_exvar_data <-
  list(
    N = nrow(model_mat),
    P = ncol(model_mat),
    y = avalanches$Deaths,
    X = model_mat,
    n_params = c(0, sqrt(10)),
    N_new = N_new,
    X_new = X_new,
    yearindmat = yim,
    N_years = ncol(yim),
    offset = d_offset,
    offset_new = n_offset
  )


stan_poisson_glm_exvar_s <-
  sampling(
    stan_poisson_glm_exvar,
    data = stan_poisson_glm_exvar_data,
    chains = 4,
    control = list(adapt_delta = 0.99, max_treedepth = 15),
    iter = 8000,
    init_r = 0.05,
    pars = c("lambda", "theta", "data_ppred", "rate")
  )

post_params_exvar <-
  extract(stan_poisson_glm_exvar_s, c("lambda"))[[1]]
post_params_theta <- extract(stan_poisson_glm_exvar_s, "theta")[[1]]
colnames(post_params_exvar) <- out_names
names(post_params_theta) <- "theta"

bound <- cbind(post_params_exvar, post_params_theta)
colnames(bound) <- c(out_names, "theta")
apply(exp(bound), 2, summary)

dpp <- extract(stan_poisson_glm_exvar_s, "data_ppred")[[1]]
apply(dpp, 2, summary)
```

```r
dpp_m2_plotdf <-
  data.frame(
    mean = apply(dpp, 2, mean),
    lq = apply(dpp, 2, quantile, 0.05),
    uq = apply(dpp, 2, quantile, 0.95),
    Season = avalanches$Season
  )
#####
plikrar <- function(x, data) {
  sum(dpois(data, x, log = T))
}
sampling_rates_exv <- extract(stan_poisson_glm_exvar_s, "rate")[[1]]
sr_like_exv <-
  apply(sampling_rates_exv, 1, plikrar, avalanches$Deaths)#calculate log likelihoods of each sampling
sr_like_mean_exv <-
  mean(sr_like_exv)#calculate mean log likelihood of samples
eap_exv <-
  colMeans(sampling_rates_exv)#calculate posterior means of rates (not parameters)
p_mean_like_exv <-
  sum(dpois(avalanches$Deaths, eap_exv, log = T))#calculate log likelihood of EAP
dbar_exv <- -2 * sr_like_mean_exv#expected deviance
pd_exv <- dbar_exv + 2 * p_mean_like_exv#calculate penalty
dic_exv <- pd_exv + dbar_exv#give dic
#####
ggplot(data = dpp_m1_plotdf, aes(x = Season)) + theme_minimal() +
  geom_ribbon(aes(ymin = lq, ymax = uq), alpha = 0.5) + labs(title = "Posterior Predictive for Model 1", y = "Number of Deaths") +
  geom_line(aes(y = mean), size = 2, colour = "red")

ggplot(data = dpp_m2_plotdf, aes(x = Season)) + theme_minimal() +
  geom_ribbon(aes(ymin = lq, ymax = uq), alpha = 0.5) + labs(title = "Posterior Predictive for Model 2 (extra variance)", y = "Number of Deaths") +
  geom_line(aes(y = mean), size = 2, colour = "red")

pp_mod_1 <- as.data.frame(exp_post_params)
pp_mod_1_long <- reshape2::melt(pp_mod_1)
pp_mod_2 <- as.data.frame(exp(bound))
pp_mod_2_long <- reshape2::melt(pp_mod_2)

ggplot(data = pp_mod_1_long, aes(x = variable, y = value)) + theme_minimal() +
  geom_boxplot() + labs(title = "Posterior summaries for model 1", y = "Parameter value", x = "Parameter") + coord_cartesian(ylim = c(0, 3))
ggplot(data = pp_mod_2_long, aes(x = variable, y = value)) + theme_minimal() +
  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

```stan
data {
  int<lower=0> N;
  int<lower=0> P;

  int<lower=0> y[N];

  matrix[N, P] X;

  int<lower=0> N_new;
  matrix[N_new, P] X_new;

```

```
  vector[2] n_params;

  vector[N] offset;
  vector[N_new] offset_new;
}
transformed data{
}

parameters {
  vector[P] lambda;
}

transformed parameters{
  vector[N] log_rate = X * lambda + offset;
  vector[N_new] log_rate_new = X_new * lambda + offset_new;
  vector<lower=0>[N] rate = exp(log_rate);
}

model {
  lambda ~ normal(n_params[1], n_params[2]);
  y ~ poisson_log(log_rate);
}

generated quantities{
  int<lower=0> y_new[N_new] = poisson_log_rng(log_rate_new);
  int<lower=0> data_ppred[N] = poisson_log_rng(log_rate);
}
```

../stan/poisson_glm_exvar.stan

```
data {
  int<lower=0> N;
  int<lower=0> P;

  int<lower=0> y[N];

  matrix[N, P] X;

  int<lower=0> N_new;
  matrix[N_new, P] X_new;

  vector[2] n_params;

  vector[N] offset;
  vector[N_new] offset_new;
}
transformed data{
}

parameters {
  //vector[P] lambda;
  real<lower=0,upper=10> theta_hyp;
  //real theta;
  real theta_raw;
  vector[P] lambda_raw;
}

transformed parameters{
  vector[P] lambda = n_params[1] + n_params[2] * lambda_raw;
  real theta = theta_hyp* theta_raw;
  vector[N] log_rate = X * lambda + theta + offset;
  vector[N_new] log_rate_new = X_new * lambda + theta + offset_new;
  vector<lower=0>[N] rate = exp(log_rate);
}
```

```

model {
  theta_hyp ~ uniform(0, 10);
  lambda_raw ~ std_normal(); //implies lambda ~ normal(n_params[1], n_params[2])
  theta_raw ~ std_normal(); // implies theta ~ normal(0, theta_hyp)
  //lambda ~ normal(n_params[1], n_params[2]);
  y ~ poisson_log(log_rate);
}

generated quantities{
  int<lower=0> y_new[N_new] = poisson_log_rng(log_rate_new);
  int<lower=0> data_ppred[N] = poisson_log_rng(log_rate);
}
```

A.3 JAGS

../jags/Q1jags.R

```
library(data.table)
library(ggplot2)

library(rjags)
library(coda)
library(bayesplot)


#####
#a
avalanches <- fread(file = "data/Avalanches.csv")
#avalanches <- avalanches[Rep.events > 0]
avalanches[, ':=' (EADS1 = (Season >= 1994 &
                              Season <= 2003),
                   EADS2 = (Season >= 2004))]

avalanches[Season %in% c(1986, 1994, 2004)]

avalanches[, EWS := 1 + EADS1 + 2 * EADS2]
avalanches[, EWS := as.factor(EWS)]

d_offset <- rep(0, nrow(avalanches))

pglm_data <-
  list(
    n = nrow(avalanches),
    w1 = avalanches$EADS1,
    w2 = avalanches$EADS2,
    rep = avalanches$Rep.events,
    death = avalanches$Deaths,
    offset = d_offset
  )

res.a <-
  jags.model(
    file = "jags/poisson.jags",
    data = pglm_data,
    n.chains = 4,
    quiet = T
  )
update(res.a, n.iter = 1e4)
res.b <-
  coda.samples(
    res.a,
```

```
    variable.names = c("intercept", "beta_w1", "beta_w2", "beta_rep"),
    n.iter = 1e4
  )
summary(res.b)
dic.samples(model = res.a,
            n.iter = 1e4,
            type = 'pD')

sm <- rbindlist(lapply(res.b, as.data.frame))

news_1_j <- mean(exp(sm$intercept) > 1)
news_2_j <- mean(exp(sm$beta_w1 + sm$intercept) > 1)
news_3_j <- mean(exp(sm$beta_w2 + sm$intercept) > 1)

res.a.ev <-
  jags.model(
    file = "jags/poisson_exvar.jags",
    data = pglm_data,
    n.chains = 4,
    quiet = T
  )
update(res.a, n.iter = 1e4)
res.b.ev <-
  coda.samples(
    res.a.ev,
    variable.names = c("beta_w1", "beta_w2", "beta_rep", "theta"),
    n.iter = 1e4
  )
summary(res.b.ev)
dic.samples(model = res.a.ev,
            n.iter = 1e4,
            type = 'pD')
```

../jags/poisson.jags

```
model {
  #hyperparameters
  p_mu <- 0
  p_tau <- 0.01

  #priors
  intercept ~ dnorm(p_mu, p_tau)
  beta_rep ~ dnorm(p_mu, p_tau)
  beta_w1 ~ dnorm(p_mu, p_tau)
  beta_w2 ~ dnorm(p_mu, p_tau)

  #likelihood
  for (i in 1:n) {
    log(mu[i]) <-
      intercept + beta_rep * rep[i] + beta_w1 * w1[i] + beta_w2 * w2[i] + offset[i]
    death[i] ~ dpois(mu[i])
  }
}
```

../jags/poisson_exvar.jags

```
model {
  #hyperparameters
  p_mu <- 0
  p_tau <- 0.01

  #priors
  beta_rep ~ dnorm(p_mu, p_tau)
```

```
  beta_w1 ~ dnorm(p_mu, p_tau)
  beta_w2 ~ dnorm(p_mu, p_tau)
  theta_hyp ~ dunif(0, 10)
  theta ~ dnorm(0, 1 / pow(theta_hyp, 2))

  #likelihood
  for (i in 1:n) {
    log(mu[i]) <- beta_rep * rep[i] + beta_w1 * w1[i] + beta_w2 * w2[i] + theta + offset[i]
    death[i] ~ dpois(mu[i])
  }
}
```

B Code for Question 2

B.1 R

../Q2.R

```
library(data.table)
library(ggplot2)
library(dplyr)

library(rstan)
rstan_options(auto_write = TRUE)
#options(mc.cores = parallel::detectCores())
Sys.setenv(LOCAL_CPPFLAGS = '-march=corei7 -mtune=corei7')
options(mc.cores = 1)
library(rstanarm)
library(coda)
library(bayesplot)

#####
#loading and eda
avalanches_prop <- fread(file = "data/Avalanches_part2.csv")
#avalanches_prop[, Event_ID := NULL]
avalanches_prop[, Snow_meters := Snow_total / 100]
avalanches_prop[, Snow_fnights := Snow_days / 14]
avalanches_prop[, Year := Season]
avalanches_prop[, death_prop := Deaths / Hit]
avalanches_prop[, Geo_space := as.factor(Geo_space)]
avalanches_prop[, Rec.station := as.factor(Rec.station)]
cor(avalanches_prop[, .(Season, Snow_meters, Snow_fnights)])
#####
stan_binomial_glm_reff <-
  stan_model(file = "stan/binomial_glm_randomeffects.stan")

submin <- function(x) {
  m <- min(x)
  x <- x - m
  attributes(x) <- list("scaled:submin" = m)
  return(x)
}

probcomp_geq <- function(x, value){
  mean(x >= value)
}

probcomp_leq <- function(x, value){
  mean(x <= value)
}

```

```r
cont_vars <- c("Snow_meters", "Snow_fnights")#variables to centre
avalanches_prop[, (cont_vars) := lapply(.SD, scale, scale = FALSE), .SDcols = cont_vars]#centre variables
tm_vars <- c("Season")
avalanches_prop[, (tm_vars) := lapply(.SD, submin), .SDcols = tm_vars]


X_fixedeff <-
  model.matrix(death_prop ~ Season + Snow_meters + Snow_fnights - 1, data = avalanches_prop)
X_randomeff <-
  model.matrix(death_prop ~ Geo_space - 1, data = avalanches_prop)
success <- avalanches_prop[, Deaths]
trials <- avalanches_prop[, Hit]


stan_binomial_glm_reff_data <-
  list(
    success = success,
    trials = trials,
    X_f = X_fixedeff,
    X_r = X_randomeff,
    N = length(success),
    P_f = ncol(X_fixedeff),
    P_r = ncol(X_randomeff),
    n_params = c(0, sqrt(10))
  )

stan_binomial_glm_reff_s <-
  sampling(
    stan_binomial_glm_reff,
    data = stan_binomial_glm_reff_data,
    chains = 4,
    control = list(adapt_delta = 0.95),
    iter = 1e4,
    init_r = 0.1
  )

post_params_rand <-
  extract(stan_binomial_glm_reff_s, c("beta_r"))[[1]]
post_params_fixed <-
  extract(stan_binomial_glm_reff_s, c("beta_f"))[[1]]
post_params <- cbind(post_params_fixed, post_params_rand)
colnames(post_params) <-
  c(colnames(X_fixedeff), colnames(X_randomeff))
ilogit_post_params <- plogis(post_params)
apply(ilogit_post_params, 2, summary)
apply(post_params, 2, summary)

dpp_rand <- extract(stan_binomial_glm_reff_s, "data_ppred")[[1]]
dpp_prop <- apply(dpp_rand, 1, "/", avalanches_prop$Hit)
apply(dpp_prop, 1, summary)

reff_coda <-
  As.mcmc.list(stan_binomial_glm_reff_s, pars = c("beta_r", "beta_f"))
gelman.plot(reff_coda, ask = FALSE)

plot_diag_objects <- function(stanfit) {
  list(
    post = as.array(stanfit),
    lp = log_posterior(stanfit),
    np = nuts_params(stanfit)
  )
}

plot_diag <- function(stanfit, pars) {
  ps <- vars(starts_with(pars))
```

```r
  post <- as.array(stanfit)
  lp <- log_posterior(stanfit)
  np <- nuts_params(stanfit)
  p1 <- mcmc_parcoord(post, np = np, pars = ps)
  p2 <- mcmc_pairs(post, np = np, pars = ps)
  p3 <- mcmc_trace(post, pars = ps, np = np)
  p4 <- mcmc_nuts_divergence(np, lp)
  p5 <- mcmc_nuts_energy(np)
  list(p1, p2, p3, p4, p5)
}

#mcmc_trace(stan_binomial_glm_reff_s, pars = vars(starts_with("beta")))

#####
#sans snow fortnights
varofint <- avalanches_prop[(Rec.station %in% c(1, 8, 10)) & (Year %in% c(2015, 2018))]
ids <- unique(varofint, by = c("Rec.station", "Year"))$Event_ID
index <- which(avalanches_prop$Event_ID %in% ids)

X_f_nsf <-
  model.matrix(death_prop ~ Season + Snow_meters - 1, data = avalanches_prop)

stan_binomial_glm_reff_nsf_data <-
  list(
    success = success,
    trials = trials,
    X_f = X_f_nsf,
    X_r = X_randomeff,
    N = length(success),
    P_f = ncol(X_f_nsf),
    P_r = ncol(X_randomeff),
    n_params = c(0, sqrt(10))
  )

stan_binomial_glm_reff_nsf_s <-
  sampling(
    stan_binomial_glm_reff,
    data = stan_binomial_glm_reff_nsf_data,
    chains = 4,
    control = list(adapt_delta = 0.95),
    iter = 10000,
    init_r = 0.1
  )



post_params_rand_ns <-
  extract(stan_binomial_glm_reff_nsf_s, c("beta_r"))[[1]]
post_params_fixed_ns <-
  extract(stan_binomial_glm_reff_nsf_s, c("beta_f"))[[1]]
post_params_ns <- cbind(post_params_fixed_ns, post_params_rand_ns)
colnames(post_params_ns) <-
  c(colnames(X_f_nsf), colnames(X_randomeff))
ilogit_post_params_ns <- plogis(post_params_ns)
apply(ilogit_post_params_ns, 2, summary)
apply(post_params_ns, 2, summary)

dpp_rand_nf <- extract(stan_binomial_glm_reff_nsf_s, "data_prop")[[1]]
apply(dpp_rand_nf, 2, summary)
dpp_ofint <- dpp_rand_nf[,index]
apply(dpp_ofint, 2, mean)
apply(dpp_ofint, 2, quantile, c(0.025, 0.975))
apply(dpp_ofint > 0.6, 2, mean)

#####
#hierarchical on station, sans snow fortnights
```

```r
X_r_station <-
  model.matrix(death_prop ~ Rec.station - 1, data = avalanches_prop)

stan_binomial_glm_reff_station_data <-
  list(
    success = success,
    trials = trials,
    X_f = X_f_nsf,
    X_r = X_r_station,
    N = length(success),
    P_f = ncol(X_f_nsf),
    P_r = ncol(X_r_station),
    n_params = c(0, sqrt(10))
  )

stan_binomial_glm_reff_station_s <-
  sampling(
    stan_binomial_glm_reff,
    data = stan_binomial_glm_reff_station_data,
    chains = 4,
    control = list(adapt_delta = 0.9),
    iter = 10000#,
    #init_r = 0.1
  )

post_params_rand_ns_stat <-
  extract(stan_binomial_glm_reff_station_s, c("beta_r"))[[1]]
post_params_fixed_ns_stat <-
  extract(stan_binomial_glm_reff_station_s, c("beta_f"))[[1]]
post_params_ns_stat <- cbind(post_params_fixed_ns_stat, post_params_rand_ns_stat)
colnames(post_params_ns_stat) <-
  c(colnames(X_f_nsf), colnames(X_r_station))
ilogit_post_params_ns_stat <- plogis(post_params_ns_stat)
apply(ilogit_post_params_ns_stat, 2, summary)
apply(post_params_ns_stat, 2, summary)

dpp_rand_ns_stat <- extract(stan_binomial_glm_reff_station_s, "data_prop")[[1]]
apply(dpp_rand_ns_stat, 2, summary)
dpp_ofintns_stat <- dpp_rand_ns_stat[,index]
apply(dpp_ofintns_stat, 2, mean)
apply(dpp_ofintns_stat, 2, quantile, c(0.025, 0.975))
apply(dpp_ofintns_stat > 0.6, 2, mean)
```

B.2 Stan

../stan/binomial_glm.stan

```stan
data {
  int<lower=0> N;
  int<lower=0> P;

  int<lower=0> y[N];

  matrix[N, P] X;

  vector[2] n_params;
}

parameters {
  vector[P] beta;
}

```

```
transformed parameters{
  vector[N] lg_p = X * beta;
}

model {
  beta ~ normal(n_params[1], n_params[2]);
  y ~ binomial(1, inv_logit(lg_p));
}
generated quantities{
  int data_ppred[N] = binomial_rng(1, inv_logit(lg_p));
}
```

../stan/binomial_glm_randomeffects.stan

```
data {
  int<lower=0> N;
  int<lower=0> P_f;
  int<lower=0> P_r;

  int<lower=0> success[N];
  int<lower=1> trials[N];

  matrix[N, P_f] X_f;
  matrix[N, P_r] X_r;

  vector[2] n_params;
}

parameters {
  vector[P_f] beta_f_raw;
  vector[P_r] sn_vec;
  real<lower=0,upper=10> reff_sdv;
}

transformed parameters{
  vector[P_f] beta_f = n_params[2] * beta_f_raw + n_params[1];
  vector[P_r] beta_r = reff_sdv * sn_vec;
  vector[N] lg_p = X_f * beta_f + X_r * beta_r;
}

model {
  reff_sdv ~ uniform(0, 10);
  sn_vec ~ std_normal(); //hence beta_r ~ normal(0, reff_sdv)
  beta_f_raw ~ std_normal(); //hence beta_f ~ normal(n_params[1], n_params[2])
  //beta_f ~ normal(n_params[1], n_params[2]);
  success ~ binomial(trials, inv_logit(lg_p));
}
generated quantities{
  int data_ppred[N] = binomial_rng(trials, inv_logit(lg_p));
  vector[N] data_prop = inv_logit(lg_p);
}
```

B.3 JAGS

../jags/Q2jags.R

```
library(data.table)
library(ggplot2)

library(rjags)
library(coda)
```

```r
library(bayesplot)

#####
#loading and eda
avalanches_prop <- fread(file = "data/Avalanches_part2.csv")
avalanches_prop[, Event_ID := NULL]
avalanches_prop[, Snow_meters := Snow_total / 100]
avalanches_prop[, Snow_fnights := Snow_days / 14]
avalanches_prop[, death_prop := Deaths / Hit]
avalanches_prop[, Geo_space := as.factor(Geo_space)]
avalanches_prop[, Rec.station := as.factor(Rec.station)]
cor(avalanches_prop[, .(Season, Snow_meters, Snow_fnights)])
#####

submin <- function(x) {
  m <- min(x)
  x <- x - m
  attributes(x) <- list("scaled:submin" = m)
  return(x)
}

cont_vars <- c("Snow_meters", "Snow_fnights")#variables to centre
avalanches_prop[, (cont_vars) := lapply(.SD, scale, scale = FALSE), .SDcols = cont_vars]#centre
↪  variables
tm_vars <- c("Season")
avalanches_prop[, (tm_vars) := lapply(.SD, submin), .SDcols = tm_vars]

snow <- avalanches_prop$Snow_meters
fnight <- avalanches_prop$Snow_fnights
season <- avalanches_prop$Season
n_eff <- length(unique(avalanches_prop$Geo_space))
eff <- as.integer(avalanches_prop$Geo_space)
n <- nrow(avalanches_prop)
deaths <- as.integer(avalanches_prop$Deaths)
hit <- as.integer(avalanches_prop$Hit)

bglm_data <-
  list(
    n = n,
    snow = snow,
    fnight = fnight,
    season = season,
    n_eff = n_eff,
    eff = eff,
    deaths = deaths,
    hit = hit
  )

res.a <-
  jags.model(
    file = "jags/binom_reff.jags",
    data = bglm_data,
    n.chains = 4,
    quiet = T
  )
update(res.a, n.iter = 1e4)
res.b <-
  coda.samples(
    res.a,
    variable.names = c("beta_snow", "beta_season", "beta_fnight", "reff"),
    n.iter = 1e4
  )

summary(res.b)
#####
snow <- avalanches_prop$Snow_meters
```

```r
season <- avalanches_prop$Season
n_eff <- length(unique(avalanches_prop$Geo_space))
eff <- as.integer(avalanches_prop$Geo_space)
n <- nrow(avalanches_prop)
deaths <- as.integer(avalanches_prop$Deaths)
hit <- as.integer(avalanches_prop$Hit)

bglm_data_nf <-
  list(
    n = n,
    snow = snow,
    season = season,
    n_eff = n_eff,
    eff = eff,
    deaths = deaths,
    hit = hit
  )

res.a_nf <-
  jags.model(
    file = "jags/binom_reff_nofn.jags",
    data = bglm_data_nf,
    n.chains = 4,
    quiet = T
  )
update(res.a_nf, n.iter = 1e4)
res.b_nf <-
  coda.samples(
    res.a_nf,
    variable.names = c("beta_snow", "beta_season", "reff"),
    n.iter = 1e4
  )

summary(res.b_nf)

dic.samples(model = res.a_nf,
            n.iter = 1e4,
            type = 'pD')
#####
snow <- avalanches_prop$Snow_meters
season <- avalanches_prop$Season
#n_eff <- length(unique(avalanches_prop$Geo_space))
#eff <- as.integer(avalanches_prop$Geo_space)
eff_stat <- as.integer(avalanches_prop$Rec.station)
n_eff_stat <- length(unique(eff_stat))
n <- nrow(avalanches_prop)
deaths <- as.integer(avalanches_prop$Deaths)
hit <- as.integer(avalanches_prop$Hit)

bglm_data_nf_stat <-
  list(
    n = n,
    snow = snow,
    season = season,
    n_eff = n_eff_stat,
    eff = eff_stat,
    deaths = deaths,
    hit = hit
  )

res.a_nf_stat <-
  jags.model(
    file = "jags/binom_reff_nofn.jags",
    data = bglm_data_nf_stat,
    n.chains = 4,
    quiet = T
```

```r
  )
update(res.a_nf_stat, n.iter = 1e4)
res.b_nf_stat <-
  coda.samples(
    res.a_nf_stat,
    variable.names = c("beta_snow", "beta_season", "reff"),
    n.iter = 1e4
  )

summary(res.b_nf_stat)

dic.samples(model = res.a_nf_stat,
            n.iter = 1e4,
            type = 'pD')
#####
snow <- avalanches_prop$Snow_meters
season <- avalanches_prop$Season
#n_eff <- length(unique(avalanches_prop$Geo_space))
#eff <- as.integer(avalanches_prop$Geo_space)
stations <- as.integer(avalanches_prop$Rec.station)
geos <- as.integer(avalanches_prop$Geo_space)
n_station <- length(unique(stations))
n_geo <- length(unique(geos))
n <- nrow(avalanches_prop)
deaths <- as.integer(avalanches_prop$Deaths)
hit <- as.integer(avalanches_prop$Hit)

bglm_data_nf_statgeo <-
  list(
    n = n,
    snow = snow,
    season = season,
    geos = geos,
    stations = stations,
    n_station = n_station,
    n_geo = n_geo,
    deaths = deaths,
    hit = hit
  )

res.a_nf_statgeo <-
  jags.model(
    file = "jags/binom_doublereff.jags",
    data = bglm_data_nf_statgeo,
    n.chains = 4,
    quiet = T
  )
update(res.a_nf_statgeo, n.iter = 1e4)
res.b_nf_statgeo <-
  coda.samples(
    res.a_nf_statgeo,
    variable.names = c("beta_snow", "beta_season", "r_eff_geo", "r_eff_statgeo"),
    n.iter = 1e4
  )

summary(res.b_nf_statgeo)
```

../jags/binom_reff.jags

```
model {
  #hyperparameters
  p_mu <- 0
  p_tau <- 0.1

```

```
  #priors
  #beta_0 ~ dnorm(p_mu, p_tau)
  beta_snow ~ dnorm(p_mu, p_tau)
  beta_season ~ dnorm(p_mu, p_tau)
  beta_fnight ~ dnorm(p_mu, p_tau)

  reff_hyp ~ dunif(0, 10)
  for (i in 1:n_eff) {
    reff[i] ~ dnorm(0, 1 / pow(reff_hyp, 2))
  }

  #likelihood
  for (i in 1:n) {
    logit(p[i]) <-
      beta_snow * snow[i] + beta_season * season[i] + beta_fnight * fnight[i] + reff[eff[i]]
    deaths[i] ~ dbinom(p[i], hit[i])
  }
}
```

../jags/binom_reff_nofn.jags

```
model {
  #hyperparameters
  p_mu <- 0
  p_tau <- 0.1

  #priors
  #beta_0 ~ dnorm(p_mu, p_tau)
  beta_snow ~ dnorm(p_mu, p_tau)
  beta_season ~ dnorm(p_mu, p_tau)

  reff_hyp ~ dunif(0, 10)
  for (i in 1:n_eff) {
    reff[i] ~ dnorm(0, 1 / pow(reff_hyp, 2))
  }

  #likelihood
  for (i in 1:n) {
    logit(p[i]) <-
      beta_snow * snow[i] + beta_season * season[i] + reff[eff[i]]
    deaths[i] ~ dbinom(p[i], hit[i])
  }
}
```

../jags/binom_doublereff.jags

```
model {
  #hyperparameters
  p_mu <- 0
  p_tau <- 0.1

  #priors
  #beta_0 ~ dnorm(p_mu, p_tau)
  beta_snow ~ dnorm(p_mu, p_tau)
  beta_season ~ dnorm(p_mu, p_tau)

  sigma_1 ~ dgamma(1, 0.1)
  sigma_2 ~ dgamma(1, 0.1)

  for(i in 1:n_geo){
    r_eff_geo[i] ~ dnorm(0, sigma_1)
  }

```

```
  for(i in 1:n_station){
    r_eff_statgeo[i] ~ dnorm(r_eff_geo[geos[i]], sigma_2)
  }

  #likelihood
  for (i in 1:n) {
    logit(p[i]) <-
      beta_snow * snow[i] + beta_season * season[i] + r_eff_statgeo[stations[i]]
    deaths[i] ~ dbinom(p[i], hit[i])
  }
}
```