Bayesian Data Analysis Assignment 2

Benjamin Cox, S1621312

Question 1

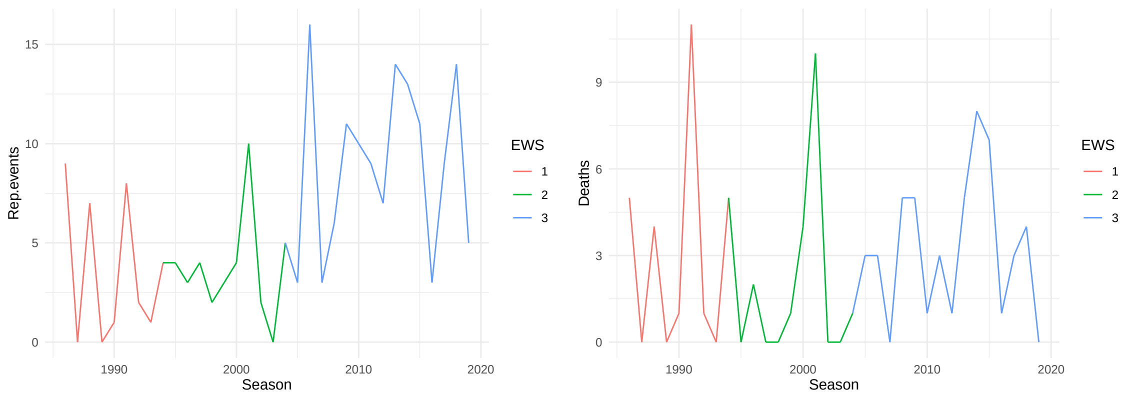

Figure 1: Plots illustrating the temporal evolution of avalanche related statistics. The EWS measure is 1 = No EADS, 2 = EADS extant, 3 = EADS online daily.

From the above graphs we can see a 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:

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

We note that these parameters have a multiplicative effect on the rate, so it is fine to have an intercept on physical terms. Note that we are using an offset, as we are calculating the rate of casualty per avalanche, so the rate should be the same per avalanche and hence the offset. We need to filter out years that have no avalanches, as they will break the 2nd equation. This is fine, as those years give no information anyway.

We could model without the offset and with a regression coefficient on the number of avalanches. However this would be nonsensical, as it would imply that there can be avalanche deaths without an avalanche even occurring. That model also had a larger DIC (and was overall worse in other measures such as WAIC and LOO) than the model that we are using.

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

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

	Intercept (β_0)	EADS1TRUE (β_1)	EADS2TRUE (β_2)
Min.	0.276	0.258	0.174
1st Qu.	0.668	0.635	0.388
Median	0.776	0.778	0.459
Mean	0.787	0.815	0.479
3rd Qu.	0.893	0.954	0.549
Max.	1.646	3.019	1.419

Table 1: Posterior summaries for the first Poisson model

From this we can make some initial conclusions. We see that the expected number of deaths per avalanche given no mitigation is 0.79. We also see that each EADS evolution decreases the expected number of deaths, by 0.82 and 0.48 times respectively (if all other variables are held constant). The latter is a rather large decrease, befitting of the drastic change in preparation tact that the EADS going online brought about.

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

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

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

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

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

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

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.

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

	EADS1TRUE (β_1)	EADS2TRUE (β_2)	theta (θ)
Min.	-1.322	-1.660	-1.101
1st Qu.	-0.491	-0.982	-0.357
Median	-0.298	-0.816	-0.206
Mean	-0.290	-0.809	-0.222
3rd Qu.	-0.096	-0.647	-0.069
Max.	0.788	0.111	0.434

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

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[, .(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)
model_mat <- model_mat[, ]
```

```r
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)
n_offset <- log(c(20, 1, 1, 1))
# X_new = matrix(c(20, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1),
#                nrow = 4,
#                byrow = T)
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 = 0.6),
    iter = 3000#,
    #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)

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)

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

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 = 4000,
    init_r = 0.05
  )

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(bound, 2, summary)

dpp <- extract(stan_poisson_glm_exvar_s, "data_ppred")[[1]]
apply(dpp, 2, summary)
#####
plikrar <- function(x, data) {
  sum(dpois(data, x, log = T))
}
sampling_rates_exv <- extract(stan_poisson_glm_exvar_s, "rate")[[1]]
```

```r
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
#####
```

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

```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)]

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

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"),
    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", "theta"),
               n.iter = 1e5)
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_w1 ~ dnorm(p_mu, p_tau)
  beta_w2 ~ dnorm(p_mu, p_tau)

  #likelihood
  for (i in 1:n) {
    log(mu[i]) <-
      intercept + 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_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_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

```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[, 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)
}

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,
```

```r
    control = list(adapt_delta = 0.9),
    iter = 10000#,
    #init_r = 0.1
  )
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))
  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

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 = 0.9),
    iter = 10000#,
    #init_r = 0.1
  )

c_data <- extract(stan_binomial_glm_reff_nsf_s, "data_prop")


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

B.2 Stan

../stan/binomial_glm.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;
  vector[P_r] sn_vec;
  real<lower=0,upper=10> reff_sdv;
}

transformed parameters{
  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 ~ 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);
}
```