Final Report

Xinyi Lin

5/14/2019

Problem 1

Question 1

First, we need to tidy data. As the day of year $x_{i,1}(t)$ and time t are two variables in given model, we need to transform `time` into some suitable formats. Normally, records are taken every 6 hour at 00:00:00, 06:00:00, 12:00:00 and 18:00:00. However, some records are taken at other timepoints. These records are ineffective as there are no record 6 hour before or after those timepoints to help train or test given models. Thus, we remove those observations. For categorical variable `nature`, as the model only use one coefficient for this variable, I assume the relationship among type of hurricans and responses is roughly monotone and changes it to numercial. As I tidied train data and test data in same way, the results will not be quiet influenced by tidy process. There are 10330 observations and 13 variables after tidied data.

Then, we randomely select 80% hurricanes and remove `id`. There are 7981 observations in the training dataset.

After that, we use componentwise M-H algorithm to develop an MCMC process. As there are too much parameters in this model, they will influence each other and it is hard to find suitable 'step length' a when using regular M-H algorithm.

As

$$Y_{i,j}(t+6) = \mu_{i,j}(t) + \rho_j Y_{i,j}(t) + \epsilon_{i,j}(t)$$

$$Y_i \sim MVN\Bigg(\begin{bmatrix} \mu_{i,1}(t) + \rho Y_{i,1}(t) \\ \mu_{i,2}(t) + \rho Y_{i,2}(t) \\ \mu_{i,3}(t) + \rho Y_{i,3}(t) \end{bmatrix}, \Sigma \Bigg)$$

$$f_{Y_i(t+6)}(Y_i|\rho_j, \beta, \Sigma) = \frac{exp[-\frac{1}{2}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu)]}{\sqrt{(2\pi)^k|\Sigma|}}$$

$$\pi(\theta) = \prod \frac{exp[-\frac{1}{2}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu)]}{\sqrt{(2\pi)^k|\Sigma|}} \times \pi_1(\beta)\pi_2(\rho_1)\pi_3(\rho_2)\pi_4(\rho_3)\pi_5(\Sigma^{-1})$$

where π are corresponding density functions.

$$\pi'(\theta) = \sum log[\frac{exp[-\frac{1}{2}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu)]}{\sqrt{(2\pi)^k|\Sigma|}}] + log[\pi_1(\beta)] + log[\pi_2(\rho_1)] + log[\pi_3(\rho_2)] + log[\pi_4(\rho_3)] + log[\pi_5(\Sigma^{-1})]$$

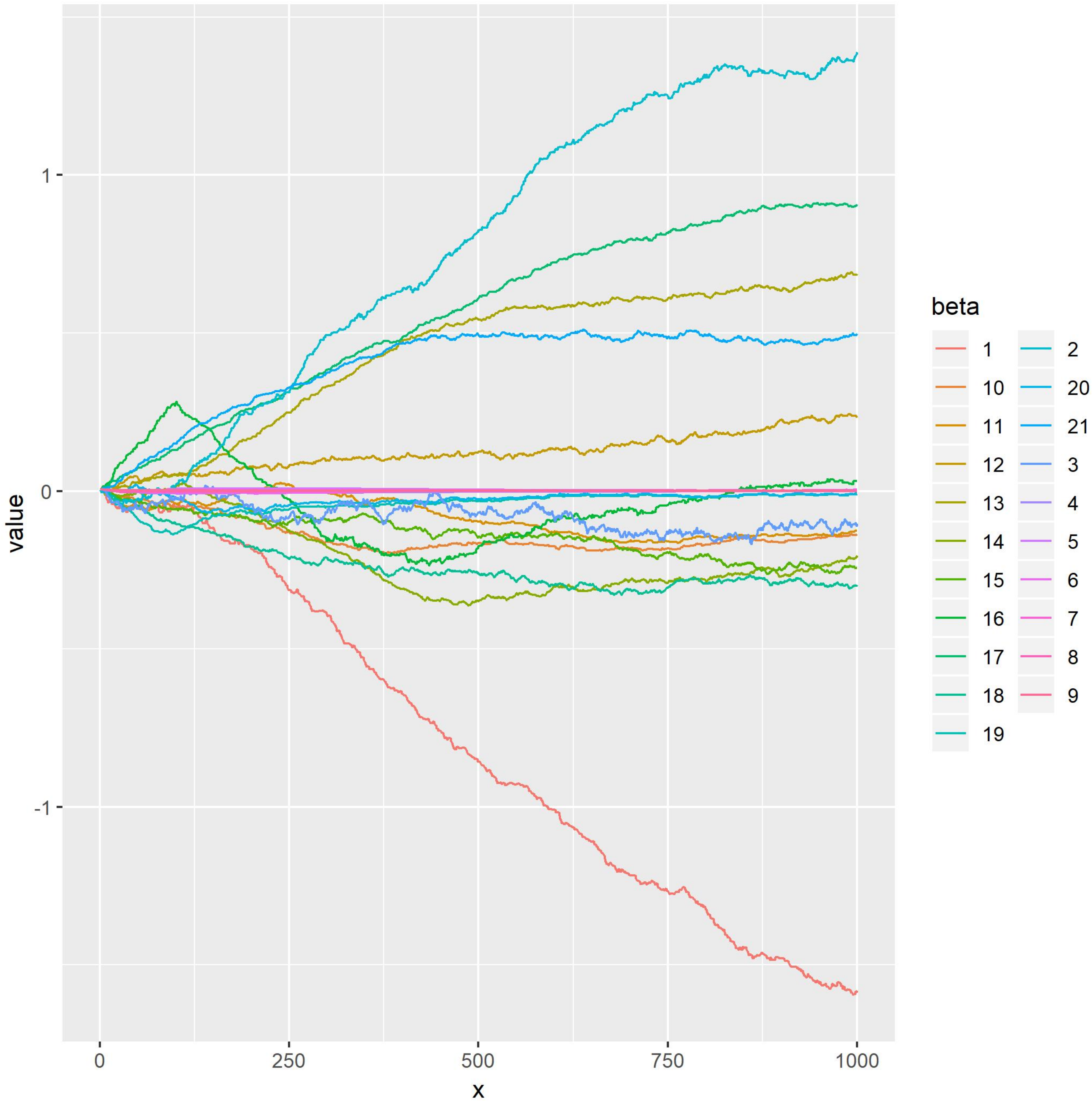
Let $\theta = (\beta, \rho_j, \Sigma)$, with probability

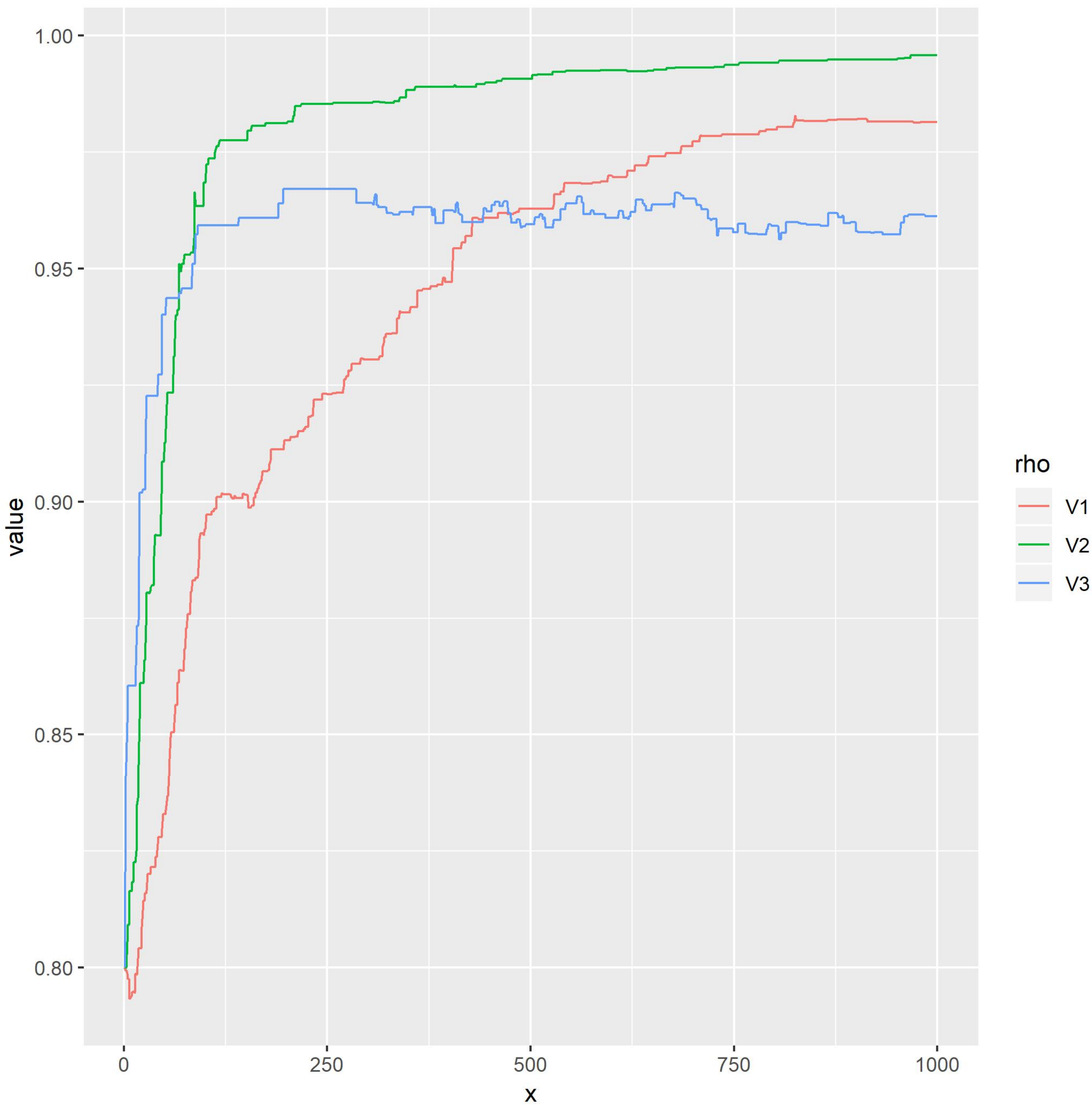
$$\alpha(\lambda|\theta^{(t)}) = min\left\{1, \frac{\pi(\lambda)q(\theta^{(t)}|\lambda)}{\pi(\theta^{(t)})q(\lambda|\theta^{(t)})}\right\} = min\left\{1, \frac{\pi(\lambda)}{\pi(\theta^{(t)})}\right\}$$

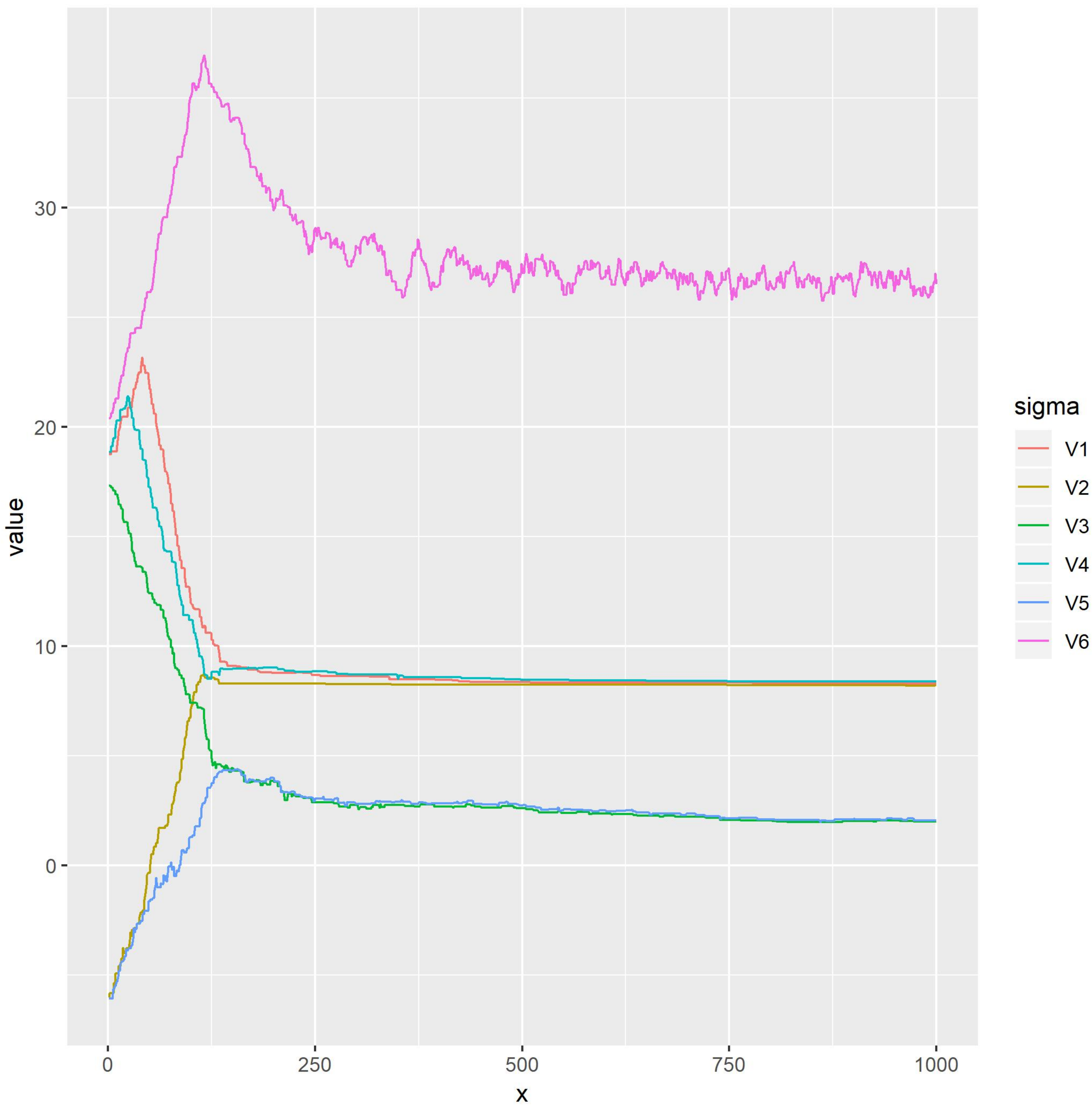
when using random walks.

Accept $\theta^{(t+1)} = \lambda$, else, set$\theta^{(t+1)} = \theta^{(t)}$.

The chain plots of parameters are shown below.




1.00
0.95
0.90
0.85
0.80
value
0
250
500
750
1000
x
rho
V1
V2
V3



I saved chain plots as jpeg files and printed them on output file to aviod runing sampling process every time I knit this file. Accordng to chain plots, we can find that only about half of betas converge, because the starting points of some beta are not good enough and their 'moving step length' a are too small. Chains of some betas look like straight lines because changes of each candidate value are small.

For rho, rho1 converged slow because bad starting point, rho2 and rho3 converged faster but the accept rate of rho2 is low, because of big a.

All sigmas converged, but some chains look like straight lines because changes of each candidate value are small.

As the efficiency of my code is too low, I have no time find out the best a and starting point for each parameter.

As most betas converged after 600 iterations, I average last 400 values as estimators of betas. I average last 500 values as estimators of rhos and sigmas for the same reason. Estimators of betas, sigmas and rhos are as following.

beta

-1.3410483554 1.2690682121 -0.1216046993 -0.0013797162 0.0015284044 0.0020540567 0.0016177516 -0.0005743277 0.0004422451 -0.1669843518 -0.1458111909 0.1786957651 0.6270347348 -0.2712040395 -0.2090274000 -0.0235848412 0.8415286601 -0.2960576751 -0.0150639019 -0.0123601083 0.4852047615

rho

0.9763370 0.9936699 0.9607596

sigma

8.317652 8.218036 2.160484 8.402449 2.261315 26.788717

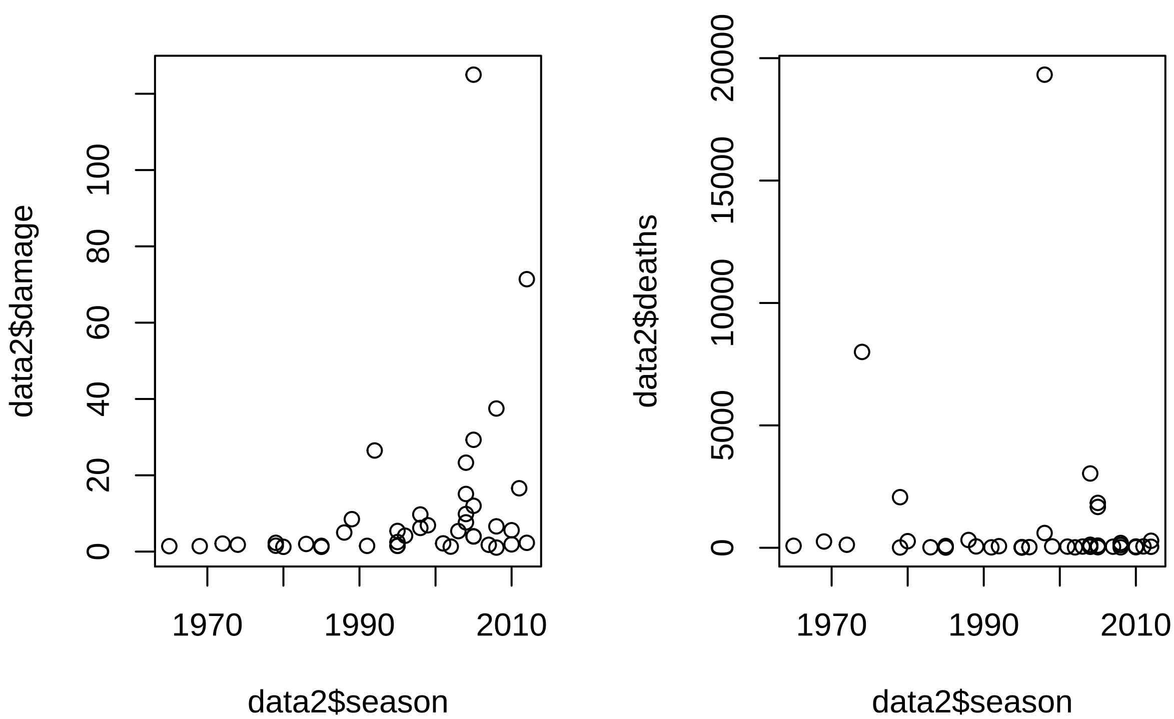
Question 2

Using test data to evaluate the model. There are 7981 observations in test data.

Mean square errors of latitude, longitude and wind speed are 8.5992694, 9.4798461, 34.9256526. As the mean of latitude, longitude and wind speed are 26.71699, -62.67803, 48.12899, this model does not predict results very well especially when predicting wind speed.

Problem 2

There are 26 unique year in the `season` variable, as relationships between season and death, damage seperately are roughly monotone, I treat season as numerical variable.



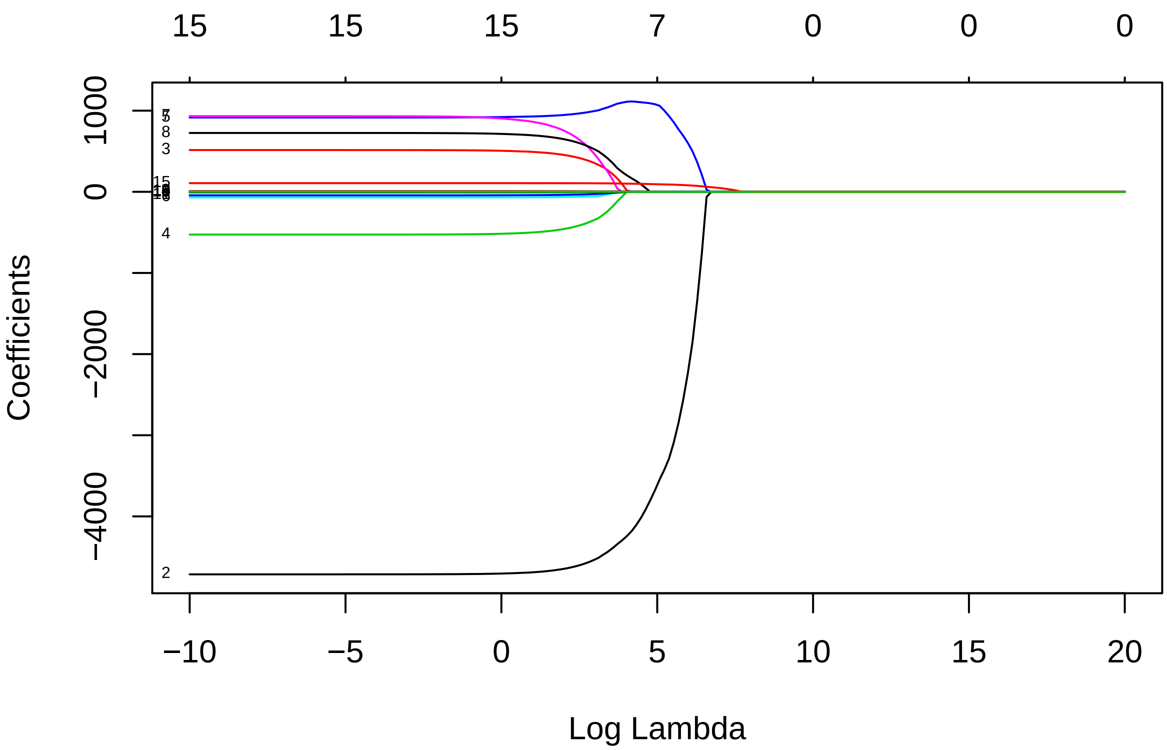
Question 1

For damage, the lasso regression does not perform well, so I use stepwise to select best model. And results are shown below.

```
##
## Call:
## lm(formula = damage ~ season + maxspeed + percent_usa, data = damage_data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.343  -8.408  -4.118   2.205  94.832
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.374e+03  5.093e+02  -2.698   0.0102 *
## season       6.770e-01  2.539e-01   2.667   0.0111 *
## maxspeed     2.202e-01  1.182e-01   1.863   0.0700 .
## percent_usa  1.391e-01  7.532e-02   1.847   0.0723 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.1 on 39 degrees of freedom
## Multiple R-squared:   0.22,  Adjusted R-squared:   0.16
## F-statistic: 3.666 on 3 and 39 DF,  p-value: 0.02026
```

According to stepwise results, `season`, `maxspeed` and `percent_usa` are associated with damage.

I use lasso regression to investigate which characteristics of the hurricanes are associated with deaths. I try to use cross validation to find out the best lambda.




```
## 17 x 4 sparse Matrix of class "dgCMatrix"
```

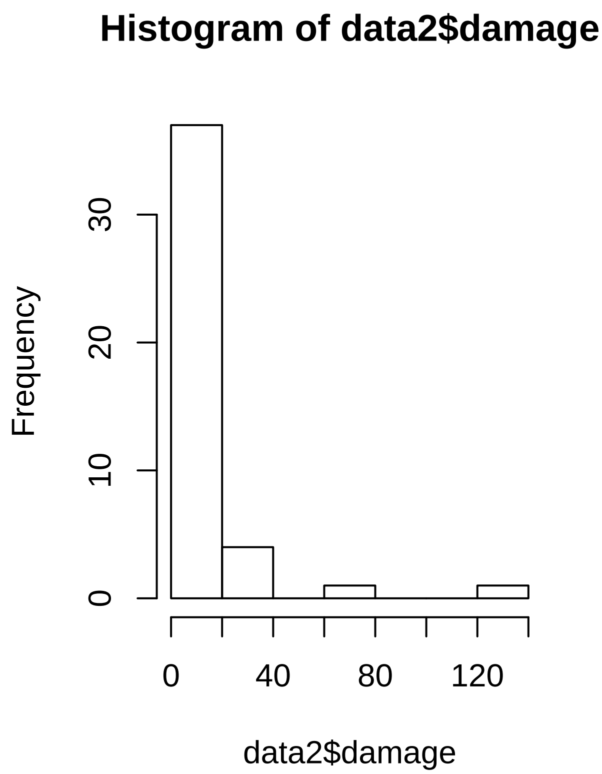
```
##                              1             2             3             4
## (Intercept)    -9.572032e+02 -3.219954e+02  1.462087e+01  2.172568e+02
## (Intercept)     .             .             .             .
## monthJuly      -3.932102e+03 -3.606340e+03 -3.363100e+03 -3.101078e+03
## monthJune       .             .             .             .
## monthNovember   .             .             .             .
## monthOctober    1.098912e+03  1.068998e+03  9.703512e+02  8.616810e+02
## monthSeptember  .             .             .             .
## natureNR        .             .             .             .
## natureTS        5.468387e+01  .             .             .
## maxspeed        4.541126e+00  2.704318e+00  8.809955e-01  1.290793e-02
## meanspeed       .             .             .             .
## maxpressure     .             .             .             .
## meanpressure    5.648366e-01  8.274959e-02  .             .
## hours           7.631274e-01  8.872830e-01  7.098641e-01  3.883862e-01
## total_pop      -2.939859e-04 -2.119758e-04 -1.607040e-04 -1.298056e-04
## percent_poor    9.704930e+01  9.306811e+01  9.049837e+01  8.809743e+01
## percent_usa     .             .             .             .
```

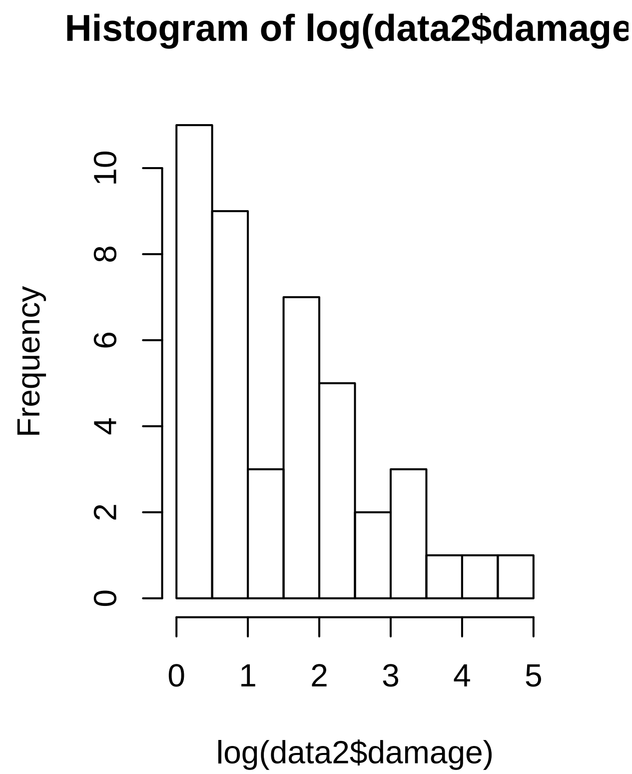
For deaths, if we use the lambda given by cross validation, no variable are significantly associated with damage. In order to select relatively significant variables, I let lambda be 100, 150, 200, 250 separately. And the results are shown above. According to results, we can find that `month`, `hours`, `total_pop`, and `percent_poor` are more significant than other variables and I conclude that they are relatively associated with deaths.

Question 2

For damage

As `damage` is numerical, I use linear regression as the model. Histograms of damage and log of damage are shown below.





According to histograms, damage is not normally distributed while the distribution of the log of damage is closer to normal, so I use log of damage as response and fit the following model.

```
##
## Call:
## lm(formula = log_damage ~ season + maxspeed + percent_usa, data = damage_data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.70669 -0.68323 -0.05057  0.48513  2.22928
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.125e+02  2.306e+01  -4.879 1.83e-05 ***
## season       5.580e-02  1.150e-02   4.854 1.99e-05 ***
## maxspeed     1.880e-02  5.351e-03   3.513  0.00114 **
## percent_usa  7.585e-03  3.410e-03   2.224  0.03199 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9102 on 39 degrees of freedom
## Multiple R-squared:  0.4634, Adjusted R-squared:  0.4221
## F-statistic: 11.22 on 3 and 39 DF,  p-value: 1.903e-05
```

According to the fitted model, with `season`, `maxspeed` and `percent_usa` increase, the average damage increase. The average damage increase 1.0573862, 1.0189778, 1.0076138 separately with one unit increase in `season`, `maxspeed` and `percent_usa`.

The mean square error is 0.7514624.As the mean of deaths is 1.532392, the mean square error is large and the model does not perform well.

For deaths

As `deaths` can only be integer, it can be regarded as following Possion distribution, so I fit following generalize linear model.

```
##
## Call:
## glm(formula = deaths ~ month + hours + total_pop + percent_poor,
##     family = poisson, data = data2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -31.608  -19.845  -14.133    0.252   91.246
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.472e+00  3.173e-02 140.940  < 2e-16 ***
## monthJuly      -5.198e+00  1.035e-01 -50.208  < 2e-16 ***
## monthJune      -3.370e-01  7.969e-02  -4.229 2.35e-05 ***
## monthNovember  -9.657e-01  1.466e-01  -6.588 4.46e-11 ***
## monthOctober    8.996e-01  2.700e-02  33.322  < 2e-16 ***
## monthSeptember  4.774e-01  2.322e-02  20.559  < 2e-16 ***
## hours           1.459e-03  5.818e-05  25.082  < 2e-16 ***
## total_pop       4.742e-07  1.314e-08  36.083  < 2e-16 ***
## percent_poor    3.784e-02  1.633e-04 231.764  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 156102  on 42  degrees of freedom
## Residual deviance:  29169  on 34  degrees of freedom
## AIC: 29462
##
## Number of Fisher Scoring iterations: 6
```

According to model results, with `hours`, `total_pop`, `percent_poor` increase, the average number of deaths increase. Comparing to August, the average number of deaths is larger in September and October and is lower in July, June and November. The avearage number of deaths increases 1.0014601, 1.0000005, 1.038565 with one unit increases in `hours`, `total_pop`, `percent_poor`.

The mean square error is 9.1901337. As the mean of deaths is 914.2558, the mean square error is small and the model performs well.

```r
knitr::opts_chunk$set(echo = FALSE)
library(tidyverse)
library(lubridate)
library(truncnorm)
library(mvtnorm)
library(matrixcalc)
library(glmnet)
# Problem 1
shift <- function(x, n=1){
  c(x[-(seq(n))], rep(NA, n))
}
data1 = read.csv("./hurrican356.csv") %>%
```

```
  janitor::clean_names() %>%
  mutate(year = season,
         date_hour = time) %>%
  separate(date_hour, into = c("date", "hour"), sep = " ") %>%
  filter(hour == "00:00:00)" | hour == "06:00:00)" | hour == "12:00:00)" | hour == "18:00:00)") %>%
  mutate(hour = str_replace(hour, ":00:00\\)", ""),
         hour = as.numeric(hour),
         date = str_replace(date, "\\(", ""),
         date = yday(date),
         nature = as.numeric(as.factor(nature))) %>%
  group_by(id) %>%
  mutate(delta1 = c(NA, diff(latitude)),
         delta2 = c(NA, diff(longitude)),
         delta3 = c(NA, diff(wind_kt)),
         latitude_p = shift(latitude),
         longitude_p = shift(longitude),
         windkt_p = shift(wind_kt)) %>%
  ungroup() %>%
  na.omit() %>%
  select(id, latitude, longitude, wind_kt, latitude_p, longitude_p, windkt_p, date, year, nature, delta

#head(data1)
#summary(data1)
set.seed(123)
id = unique(data1$id)
num_id = length(id)
train_id = sample(id, 0.8*num_id)

train_data = data1[which(data1$id %in% train_id),] %>%
  select(-id)
# Starting points
set.seed(111)
# rho: 1*3 vector
#rho = rtruncnorm(3, a=0, b=1, mean = 0.5, sd = 1/5)
rho = rep(0.8, 3)
# epsilon: 1*3 vector
sigma = bayesm::rwishart(3,diag(0.1,3))$IW
sigma = c(sigma[1,], sigma[2,c(2,3)], sigma[3,3])
#beta: 1*18 vector, where beta_kj is the [6*(j-1)+(k+1)]th number.
#beta = rmvnorm(1, rep(0,21), diag(1,21))
beta = rep(0.005,21)
# Density function.
# for each yi
logdy = function(obs, beta, rho, sigma){
  x = c(1,obs[7:12])
  y = obs[1:3]
  mu = beta %*% x + rho*obs[1:3]
  dy = dmvnorm(obs[4:6], mean = mu, sigma = sigma)
  return(log(dy))
}

#traintest = train_data[c(1:100),]
#betatest = rep(0.008,21)
```

```
#apply(traintest, 1, logdy, beta.=betatest)

logdensity = function(data=train_data, beta.=beta, rho.=rho, sigma.=sigma){
  beta_m = matrix(beta.,3)
  sigma_m = matrix(c(sigma.[c(1:3)], sigma.[2], sigma.[c(4,5)], sigma.[c(3,5)], sigma.[6]), 3)
  logdy = apply(data, 1, logdy, beta=beta_m, rho=rho., sigma=sigma_m)
  logdens = sum(logdy) + log(dmvnorm(beta., rep(0,21), diag(1,21))) + log(dtruncnorm(rho.[1], a=0, b=1,
  return(logdens)
}

#logdensity(train_data)
# Sampling process
regularMHstep = function(startpars, niter = 1000, rhoa, betaa, sigmaa){
  beta_m = matrix(NA, niter, 21)
  rho_m = matrix(NA, niter, 3)
  sigma_m = matrix(NA, niter, 6)
  beta_m[1,] = startpars$beta
  rho_m[1,] = startpars$rho
  sigma_m[1,] = startpars$sigma
  for (i in 2:niter) {   # correlated issue
    #posbeta = beta_m[i-1,] + runif(21,-1,1)*beta_a*ifelse((runif(21) < 0.1),1,0)
    #posrho = rho_m[i-1,] + runif(3,-1,1)*rho_a*ifelse((runif(3) < 0.1),1,0)
    #possigma = sigma_m[i-1,] + runif(6,-1,1)*sigma_a*ifelse((runif(6) < 0.1),1,0)
    posbeta = beta_m[i-1,] + runif(21,-1,1)*beta_a
    posrho = rho_m[i-1,] + runif(3,-1,1)*rho_a
    possigma = sigma_m[i-1,] + runif(6,-1,1)*sigma_a
    possigma_m = matrix(c(possigma[c(1:3)], possigma[2], possigma[c(4,5)], possigma[c(3,5)], possigma[6
    if (sum(ifelse(posrho<1, 0, 1))==0 & is.positive.definite(possigma_m)) {
      if (log(runif(1)) < logdensity(beta.=posbeta, rho.=posrho, sigma.=possigma) - logdensity(beta.=be
        beta_m[i,] = posbeta
        rho_m[i,] = posrho
        sigma_m[i,] = possigma
        }
      else{
        beta_m[i,] = beta_m[i-1,]
        rho_m[i,] = rho_m[i-1,]
        sigma_m[i,] = sigma_m[i-1,]
      }}
    else{
      beta_m[i,] = beta_m[i-1,]
      rho_m[i,] = rho_m[i-1,]
      sigma_m[i,] = sigma_m[i-1,]
      }
  }
  return(list(MHbeta = beta_m, MHrho = rho_m, MHsigma = sigma_m))
}
compMHstep = function(startpars, niter = 1000, rhoa, betaa, sigmaa){
  beta_m = matrix(NA, niter, 21)
  rho_m = matrix(NA, niter, 3)
  sigma_m = matrix(NA, niter, 6)
  beta_m[1,] = startpars$beta
  rho_m[1,] = startpars$rho
  sigma_m[1,] = startpars$sigma
```

```
  for (i in 2:niter) {
    posbeta = beta_m[i-1,]
    for (j in 1:21) {
      curbeta = posbeta
      posbeta[j] = posbeta[j] + runif(1,-1,1)*beta_a[j]
      if (log(runif(1)) >= logdensity(beta.=posbeta, rho.=rho_m[i-1,], sigma.=sigma_m[i-1,]) - logdensi
        posbeta[j] = curbeta[j]
      }
      }
    beta_m[i,] = posbeta
    posrho = rho_m[i-1,]
    for (j in 1:3) {
      currho = posrho
      posrho[j] = posrho[j] + runif(1,-1,1)*rho_a[j]
      if (sum(ifelse(posrho<1, 0, 1))==0) {
        if (log(runif(1)) >= logdensity(beta.=beta_m[i,], rho.=posrho, sigma.=sigma_m[i-1,]) - logdensi
          posrho[j] = rho_m[i-1,j]
        }
      }
      else{
        posrho[j] = rho_m[i-1,j]
      }
      }
    rho_m[i,] = posrho
    possigma = sigma_m[i-1,]
    for (j in 1:6) {
      cursigma = possigma
      possigma[j] = possigma[j] + runif(1,-1,1)*sigma_a[j]
      possigma_m = matrix(c(possigma[c(1:3)], possigma[2], possigma[c(4,5)], possigma[c(3,5)], possigma
      if (is.positive.definite(possigma_m)) {
        if (log(runif(1)) >= logdensity(beta.=beta_m[i,], rho.=rho_m[i,], sigma.=possigma) - logdensity
          possigma[j] = sigma_m[i-1,j]
        }
      }
      else{
        possigma[j] = sigma_m[i-1,j]
      }
    }
    sigma_m[i,] = possigma
  }
  return(list(MHbeta = beta_m, MHrho = rho_m, MHsigma = sigma_m))
}
# starting points of parameters
startpars = list(rho = rho, beta = beta, sigma = sigma)
rho_a = c(0.005,0.005,0.01)
sigma_a = rep(0.5, 6)
beta_a = c(rep(0.01, 3), rep(0.0005, 2), 0.001, rep(0.0001, 3), rep(0.005,6), 0.01, rep(0.005, 5))
set.seed(123)
MHresults = compMHstep(startpars, niter = 1000, rhoa = rho_a, betaa = beta_a, sigmaa = sigma_a)
# check accept rate
uni_beta = rep(NA,21)
for (i in 1:21){
  uni_beta[i] = length(unique(MHresults$MHbeta[,i]))
```

```
}
uni_beta

uni_sigma = rep(NA,6)
for (i in 1:6) {
  uni_sigma[i] = length(unique(MHresults$MHsigma[,i]))
}
uni_sigma

uni_rho = rep(NA,3)
for (i in 1:3) {
  uni_rho[i] = length(unique(MHresults$MHrho[,i]))
}
uni_rho
# print chain plots
niter = 1000
beta_results = as.data.frame(MHresults$MHbeta) %>%
  mutate(x = 1:niter) %>%
  gather(key = beta, value = value, V1:V21) %>%
  mutate(beta = str_replace(beta, "V", ""))
beta_plot = ggplot(beta_results, aes(x = x, y = value, color = beta)) +
  geom_line()
ggsave("beta_plot1000.jpeg", beta_plot)

rho_results = as.data.frame(MHresults$MHrho) %>%
  mutate(x = 1:niter) %>%
  gather(key = rho, value = value, V1:V3)
rho_plot = ggplot(rho_results, aes(x = x, y = value, color = rho)) +
  geom_line()
ggsave("rho_plot1000.jpeg", rho_plot)

sigma_results = as.data.frame(MHresults$MHsigma) %>%
  mutate(x = 1:niter) %>%
  gather(key = sigma, value = value, V1:V6)
sigma_plot = ggplot(sigma_results, aes(x = x, y = value, color = sigma)) +
  geom_line()
ggsave("sigma_plot1000.jpeg", sigma_plot)
knitr::include_graphics("./beta_plot1000.jpeg")
knitr::include_graphics("./rho_plot1000.jpeg")
knitr::include_graphics("./sigma_plot1000.jpeg")
# get test data
test_id = setdiff(id,train_id)
test_data = data1[which(data1$id %in% test_id),] %>%
  select(-id)

# estimated parameters
esbeta = c(-1.3410483554, 1.2690682121, -0.1216046993, -0.0013797162, 0.0015284044, 0.0020540567, 0.0016
esbeta_m = matrix(esbeta,3)
esrho = c(0.9763370, 0.9936699, 0.9607596)
essigma = c(8.317652, 8.218036, 2.160484, 8.402449, 2.261315, 26.788717)
essigma_m = matrix(c(essigma[c(1:3)], essigma[2], essigma[c(4,5)], essigma[c(3,5)], essigma[6]), 3)
# for each yi
predy = function(obs){
```

```r
  x = c(1,obs[7:12])
  mu = esbeta_m %*% x + esrho*obs[1:3]
  epsilon = rmvnorm(1, mean = rep(0,3), sigma = essigma_m)
  y = t(mu) + epsilon
  return(y)
}

set.seed(123)
pred_results = apply(test_data, 1, predy)
# evaluate mean square error
squ_latitude = (pred_results[1,] - test_data[,1])^2
err_latitude = sum(squ_latitude)/2349
squ_longitude = (pred_results[2,] - test_data[,2])^2
err_longitude = sum(squ_longitude)/2349
squ_wk = (pred_results[3,] - test_data[,3])^2
err_wk = sum(squ_wk)/2349

#mean(test_data$latitude)
#mean(test_data$longitude)
#mean(test_data$wind_kt)
# Problem 2
data2 = read_csv("./hurricanoutcome2.csv") %>%
  janitor::clean_names() %>%
  mutate(damage = str_replace(damage, "\\$", ""),
         damage = as.numeric(damage)) %>%
  select(-hurrican_id)
#summary(data2)
#length(unique(data2$season))
#length(unique(data2$month))
par(mfrow=c(1,2))
plot(data2$season, data2$damage)
plot(data2$season, data2$deaths)
damage_data2 = data2 %>%
  dplyr::select(-deaths)
damage_step <- step(lm(damage ~ .,  data = damage_data2),direction="both",trace = 0)
summary(damage_step)
# for deaths
deathsx = model.matrix(deaths~.,data2)[,-c(2,3)]
cv.lasso <- cv.glmnet(deathsx, data2$deaths, alpha = 1, lambda = exp(seq(-10, 20, length=200)))
#cv.lasso$lambda.min
#plot(cv.lasso)
plot(cv.lasso$glmnet.fit, xvar = "lambda", label=TRUE)
coef(cv.lasso, seq(100,250,length.out = 4))
par(mfrow=c(1,2))
hist(data2$damage)
hist(log(data2$damage))
damage_data2 = data2 %>%
  select(-deaths) %>%
  mutate(log_damage = log(damage)) %>%
  select(-damage)

damage.lm = lm(log_damage~ season + maxspeed + percent_usa, data = damage_data2)
summary(damage.lm)
```

```r
#mean(damage.lm$residuals^2)
#mean(damage_data2$log_damage)
deaths.glm <- glm(deaths ~ month + hours + total_pop + percent_poor, data = data2, poisson)

summary(deaths.glm)
#mean(data2$deaths)
```