RPM code

Gary Venter

R Code

```
setwd("~/OneDrive/R/Ratemaking Stan/RPM")
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library("loo")
library(readxl)
library(glmnet)

#Start with regression
y = as.matrix(read_excel("z_small.xlsx")[,6]) #logs of severity
x = read_excel("x_small.xlsx") # for lm regression need x as a data frame
reg_mod <- lm(y ~ ., data = x)
summary(reg_mod)

#Next lasso
x = as.matrix(read_excel("x_small.xlsx")) # for lasso need x as a matrix
fit1 = glmnet(x, y, standardize = FALSE)
plot(fit1, label=TRUE)
cvfit = cv.glmnet(x, y, standardize = FALSE)
mid = (cvfit$lambda.min*cvfit$lambda.1se)^0.5
c(cvfit$lambda.min,mid,cvfit$lambda.1se) #3 lambdas from cross validation
coef(cvfit, s=c(cvfit$lambda.min,mid,cvfit$lambda.1se)) #fits for 3 lambdas

#Now Bayesian lasso;  our program needs y to be a vector
x_10 = as.matrix(read_excel("x_small_2nd.xlsx")) #all 10 variables
#This one for 2nd diff dummies. Just x_small for 0, 1 duummies
y = as.vector(as.matrix(read_excel("z_small.xlsx")[,3]))/100 # $ loss/100
claims = as.vector(as.matrix(read_excel("z_small.xlsx")[,4]))
```

```
#Doing regression for $loss which is scaled to make parameters reasonable
x=x_10[,-c(2,3,7,8)] #excludes some variables. Start with x=x_10
U = ncol(x)
N = length(y)
c(N,U) #number of rows and columns passed to Stan

set.seed(4) #done so using same random seed each time. 4 is arbitrary
fitsevgam = stan(file = 'sevgam.stan', verbose = FALSE, chains = 7,
                 iter = 7000, warmup = 2000)
#more chains and iterations than needed for this, but runs fast

#Now calculate loo
log_LL <- extract_log_lik(fitsevgam)
loo_LLsevgam1 <- loo(log_LL)
loo_LLsevgam1

#Shows some of the parameter ranges
print(fitsevgam, pars=c("cn", "v", "s","logs", "alpha" ),
      probs=c(.05, 0.2, 0.5, 0.8, 0.95), digits_summary = 3)
plot(fitsevgam, pars = c("v", "s"))

#Outputs means by chain for every parameter, transformed parameter and LL
out <- get_posterior_mean(fitsevgam)
write.csv(out, file="out_sevgamultdiff.csv")
```

Stan Code

```
data {  //have to declare variables - compiled in C
  int N;            //number of observations
  int U;            //number of variables
  vector[N] y;      //the dollar losses in a column
  vector[N] claims; //exposures
  matrix[N,U] x;    //design matrix with U columns
}
```

```
parameters {  // all except v will get uniform prior, which is default
 real<lower=-5, upper=5> logcn;     //log constant term
 vector[U] v;                       //the parameters
 real<lower=-5, upper=-1> logs;     //log of s, related to lambda, not too high
 real logalpha;                     //log of gamma a parameter
}
transformed parameters {
 real cn;
 real alpha;
 real s;                        //shrinkage parameter, like lambda
 vector[N] beta;                //fitted means
 cn = exp(logcn); //for positive parameter, uniform on log is like ~ 1/X
 alpha = exp(logalpha);    //1/X prior gives more weight to lower values
 s = exp(logs);   //uniform prior can bias a positive parameter upward
 beta = exp(x*v)*cn/alpha; //vector of gamma b parameters, so mean/alpha
 for (j in 1:N) beta[j]=1/beta[j]/claims[j]; //b for $loss
} //actually in Stan, gamma mean is a/b
model { // gives priors for those not assumed uniform. Choose this one for Bayesian lass
   for (i in 1:U)  v[i] ~ double_exponential(0, s);
for (j in 1:N) y[j] ~ gamma(alpha, beta[j]);  //Says data is gamma distributed
}
generated quantities { //outputs log likelihood for loo
  vector[N] log_lik;
for (j in 1:N) log_lik[j] = gamma_lpdf(y[j] | alpha,beta[j]);
}
```