NBA Contract Modeling

Matthew Perkins

4/25/2022

```
library(tidyverse)
library(data.table)
library(rjags)
library(parallel)
library(arm) # contains bayesglm function
library(greta)
library(coda)
```

loading the data with reduced number of variables

```
train = fread('allSignificantPlayersRed.csv')
train_rest = fread('lowGamesPlayersRed.csv')
train = rbind(train, train_rest)
head(train)
```

```
##        fullName height weight signAge totalValue      aav signSeason draftRound
## 1:   A.J. Price     74    181      28  0.1007087 1.007087    2013-14          2
## 2:   A.J. Price     74    181      27  0.1063384 1.063384    2013-14          2
## 3: Aaron Brooks     72    161      32  0.2116955 2.116955    2016-17          1
## 4: Aaron Brooks     72    161      31  0.2700000 2.700000    2015-16          1
## 5: Aaron Brooks     72    161      30  0.2250000 2.250000    2014-15          1
## 6: Aaron Brooks     72    161      29  0.1145685 1.145685    2013-14          1
##    draftOverall draftYear undrafted termFirstYear termSecondYear signYear
## 1:           52      2009     FALSE          2014           2014     2014
## 2:           52      2009     FALSE          2014           2014     2014
## 3:           26      2007     FALSE          2017           2017     2017
## 4:           26      2007     FALSE          2016           2016     2016
## 5:           26      2007     FALSE          2015           2015     2015
## 6:           26      2007     FALSE          2014           2014     2014
##    latestContract totalGames totalGamesStarted     WAFG rangeFG     WAFG%
## 1:              0        129                23 1.832558     2.1 0.3775969
## 2:              1        129                23 1.832558     2.1 0.3775969
## 3:              0        216                21 3.028704     2.3 0.4091944
## 4:              0        223                33 3.413004     1.5 0.4083543
## 5:              0        207                53 3.468116     1.5 0.4222367
## 6:              0        184                44 3.216304     1.0 0.4076413
##    rangeFG%      WA3P range3P     WA3P% range3P%     WA2P range2P     WA2P%
## 1:    0.074 0.7782946     1.0 0.3145271    0.077 1.088372     1.1 0.4372946
## 2:    0.074 0.7782946     1.0 0.3145271    0.077 1.088372     1.1 0.4372946
## 3:    0.020 1.0995370     0.8 0.3738056    0.030 1.931019     1.6 0.4327500
## 4:    0.020 1.2807175     0.5 0.3777175    0.030 2.163229     0.9 0.4282780
```

```
## 5:    0.052 1.2768116     0.6 0.3834155    0.014 2.191304     0.9 0.4488841
## 6:    0.078 1.1527174     0.4 0.3541087    0.090 2.063587     0.7 0.4449348
##    range2P%    WAeFG% rangeeFG%      WAFT rangeFT     WAFT% rangeFT%     WAORB
## 1:    0.158 0.4548450     0.064 0.6023256     0.9 0.6219380    0.800 0.2790698
## 2:    0.158 0.4548450     0.064 0.6023256     0.9 0.6219380    0.800 0.2790698
## 3:    0.018 0.4837222     0.024 1.0574074     1.3 0.8016667    0.067 0.3379630
## 4:    0.031 0.4840224     0.024 1.2659193     1.1 0.8255067    0.108 0.4336323
## 5:    0.100 0.5006473     0.048 1.3352657     1.0 0.8308744    0.105 0.4183575
## 6:    0.100 0.4817935     0.098 1.3733696     1.3 0.8476033    0.117 0.3885870
##    rangeORB    WADRB rangeDRB    WAAST rangeAST     WASTL rangeSTL      WABLK
## 1:      0.4 1.147287      1.3 2.381395      3.1 0.4356589      0.6 0.04418605
## 2:      0.4 1.147287      1.3 2.381395      3.1 0.4356589      0.6 0.04418605
## 3:      0.1 1.231481      0.8 2.617130      1.3 0.5138889      0.3 0.13796296
## 4:      0.3 1.379372      0.4 3.014350      0.6 0.6071749      0.3 0.16905830
## 5:      0.4 1.418841      0.3 2.943961      1.0 0.6743961      0.1 0.20000000
## 6:      0.4 1.203804      0.3 3.136413      1.7 0.6391304      0.1 0.16793478
##    rangeBLK     WATOV rangeTOV      WAPF rangePF    WAPTS rangePTS    WAPER
## 1:      0.1 0.7899225      0.8 0.8565891     1.1 5.079845      6.1 11.50698
## 2:      0.1 0.7899225      0.8 0.8565891     1.1 5.079845      6.1 11.50698
## 3:      0.1 1.4055556      0.9 1.9013889     0.9 8.176389      6.6 12.09491
## 4:      0.1 1.5865471      0.7 2.0793722     0.4 9.368161      4.5 12.98206
## 5:      0.0 1.6420290      0.6 2.0676329     0.5 9.543478      4.5 13.07343
## 6:      0.1 1.5456522      0.4 1.9103261     0.2 8.997826      3.6 12.49076
##    rangePER     WATS% rangeTS%    WA3PAr range3PAr     WAFTr rangeFTr   WAORB%
## 1:      2.7 0.4780233    0.047 0.4902016     0.028 0.1441705    0.158 1.876744
## 2:      2.7 0.4780233    0.047 0.4902016     0.028 0.1441705    0.158 1.876744
## 3:      4.9 0.5130972    0.040 0.3997546     0.044 0.1643287    0.080 2.052315
## 4:      2.6 0.5164574    0.040 0.4006099     0.044 0.1733543    0.077 2.318386
## 5:      2.6 0.5338116    0.037 0.4088019     0.044 0.1836667    0.049 2.214976
## 6:      1.3 0.5193587    0.066 0.4181196     0.025 0.1883370    0.075 2.222283
##    rangeORB%   WADRB% rangeDRB%   WAAST% rangeAST%   WASTL% rangeSTL%    WABLK%
## 1:       1.5 9.043411       2.0 24.48372       4.9 1.331008       1.4 0.1906977
## 2:       1.5 9.043411       2.0 24.48372       4.9 1.331008       1.4 0.1906977
## 3:       0.4 7.138889       1.2 23.72176       5.3 1.437963       0.1 0.7222222
## 4:       1.2 7.241704       0.8 24.46637       2.7 1.501345       0.2 0.6309417
## 5:       1.6 7.324155       1.2 22.29952       6.2 1.534783       0.1 0.6512077
## 6:       1.6 6.564674       2.7 23.56902      10.9 1.475000       0.3 0.5293478
##    rangeBLK%   WATOV% rangeTOV%   WAUSG% rangeUSG%     WAOWS rangeOWS     WADWS
## 1:       0.3 13.37907       1.8 19.02636       5.5 0.4883721      1.1 0.7007752
## 2:       0.3 13.37907       1.8 19.02636       5.5 0.4883721      1.1 0.7007752
## 3:       0.3 15.36852       3.0 22.58380       5.8 0.6490741      1.9 0.9435185
## 4:       0.1 14.97399       1.6 22.89731       4.5 1.1067265      1.5 1.0587444
## 5:       0.2 15.62271       1.6 21.66812       6.9 1.3304348      0.9 0.9072464
## 6:       0.6 15.16793       3.3 21.54022       7.8 0.9635870      0.6 0.4804348
##    rangeDWS    WAWS/48 rangeWS/48     WAOBPM rangeOBPM     WADBPM rangeDBPM
## 1:      1.2 0.05600000      0.096 -0.8139535       2.2 -0.8682171       2.0
## 2:      1.2 0.05600000      0.096 -0.8139535       2.2 -0.8682171       2.0
## 3:      1.0 0.04910185      0.067 -1.0236111       2.7 -1.1939815       0.2
## 4:      0.8 0.06452915      0.043 -0.3565022       1.6 -1.0650224       0.5
## 5:      1.5 0.06668116      0.042 -0.1555556       1.1 -1.0724638       0.6
## 6:      0.9 0.05085326      0.027 -0.4048913       0.6 -1.5820652       1.9
##         WAVORP rangeVORP     WAMP rangeMP
## 1: 0.233333333       0.6 15.05736    18.9
## 2: 0.233333333       0.6 15.05736    18.9
```

```
## 3: 0.005555556       0.9 18.02731     9.2
## 4: 0.251121076       0.7 20.41300     6.9
## 5: 0.337198068       0.5 21.43768     4.2
## 6: 0.060326087       0.7 20.85761     3.0
```

```
train[,undrafted:=ifelse(undrafted, 1, 0)]
train[,signSeason:=NULL]
test = train[signYear==2021,]
train = train[(signYear!=2021),]
print(nrow(test))
```

```
## [1] 194
```

```
print(nrow(train))
```

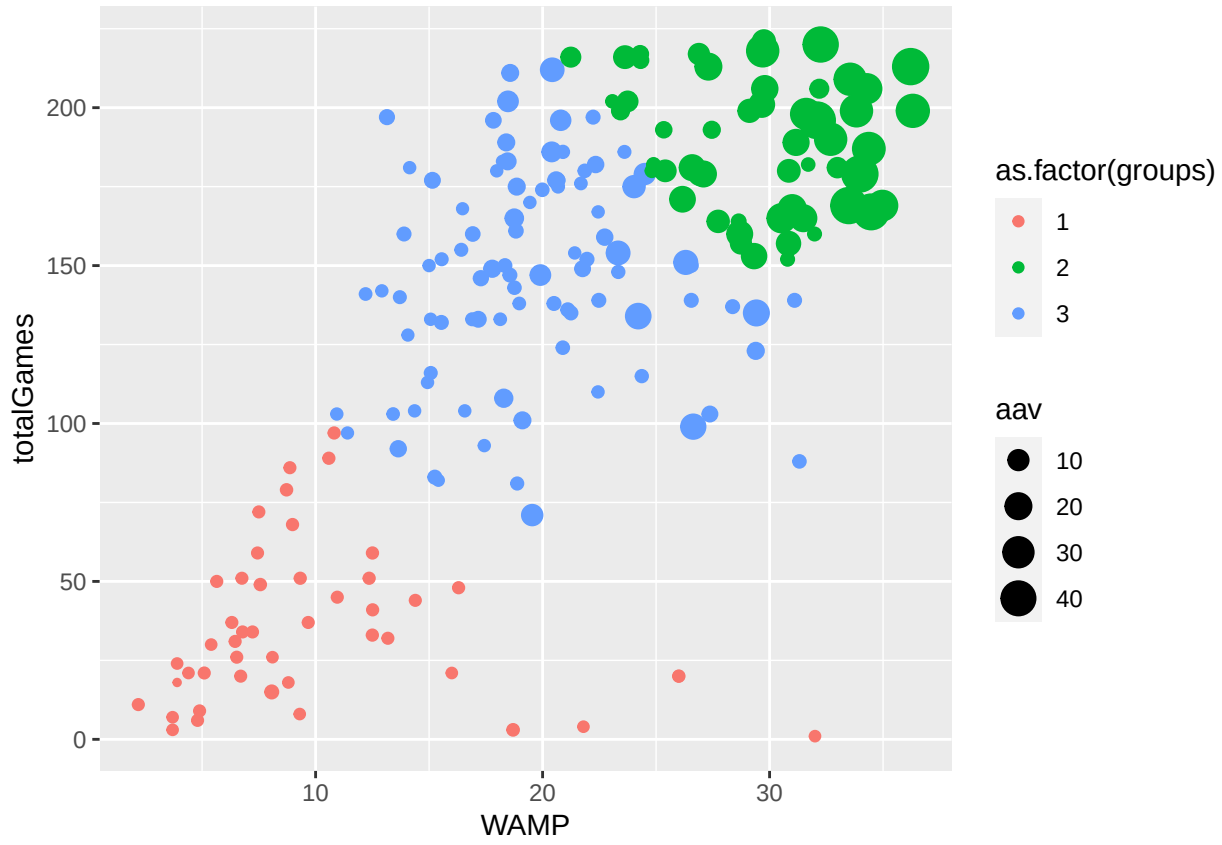
```
## [1] 1218
```

there were 194 test points and 1218 rows for training.

split the data into three groups based on the combination of total games and WAMP. I chose these because of their strong paired correlation with aav and because they are part of the predictors. It would be weird to base the groupings on aav since that would imply one already knows aav.

```
interMPG = train$WAMP*train$totalGames
lower_box = quantile(interMPG, .20)
upper_box = quantile(interMPG, .65)
train$interMPG = interMPG
train[,groups:=ifelse((interMPG<..lower_box), 1,
                      ifelse(interMPG>..upper_box, 2, 3))]
train$interMPG=NULL
ggplot(train, aes(x=WAMP, y = totalGames, size=aav, color=as.factor(groups)))+
geom_point()
```


```
# create the first group which is about the players in the upper quartile
train1 = train[groups==2,]
interMPG = test$WAMP*test$totalGames
test$interMPG = interMPG
test[,groups:=ifelse((interMPG<..lower_box), 1,
                     ifelse(interMPG>..upper_box, 2, 3))]
test$interMPG=NULL
test1 = test[groups==2, ]
ggplot(test, aes(x=WAMP, y = totalGames, size=aav, color=as.factor(groups)))+
geom_point()
```



Now prep the data for greta

```
# function that take the train and test data and creates vectors and
# matrices of the trained test data but with scaled predictors
make_train_test_data = function(data, test_data, target, remove_cols=c(), scale=TRUE){
  data = as.data.table(data)
  test_data=as.data.table(test_data)
  y = as.matrix(data[,..target])[,1]
  yTest = as.matrix(test_data[,..target])[,1]
  x = data[,-remove_cols, with=FALSE]
  xTest = test_data[,-remove_cols, with=FALSE]
  if(scale){
      x = scale(x)
      xTest = scale(xTest, attr(x, "scaled:center"), attr(x, "scaled:scale"))
  }
  return(list(y=y,yTest=yTest, x=x, xTest=xTest))
}

First_group = make_train_test_data(data=train1,
                                   test_data=test1, target='aav',
                                   remove_cols=c('totalValue', 'aav',
                                                 'fullName','termSecondYear',
                                                 'termFirstYear', 'groups'),
                                   )
```

the three models will be $Y_i \ Normal(\sum_{j=1}^{p} x_{ij} * \beta_j + \alpha, \sigma^2)$, with priors for beta being the laplace (double exponential) priors, that is $\beta_j \sim laplace(0,\ 0.01)$. The Y_i are normalized using the *bestNormalize* package and the x_{ij} will be scaled. The intercept, α, will have a prior of $\alpha \sim Normal(0, 100)$. Finally, the standard deviation will have a prior of $\sigma \sim InvGamma(.01, .01)$.

```
# define spike slab prior like that from documentation
# https://rdrr.io/cran/greta/f/inst/examples/linear_spike_and_slab.Rmd
nvar = ncol(First_group$x)
spike_and_slab <- function (spikiness = 0.1, slabiness = 10, center=0, dim = NULL) {
  stopifnot(spikiness < 1)
  slab <- normal(center, slabiness, dim = dim)
  spike <- laplace(center, spikiness, dim = dim)
  spike * slab
}
min_max_transform = function(data, method=list(), predict=FALSE, inverse=FALSE, range=c(0,1), hard_max=
  if(inverse){
    return((data-method$range[1])*(method$max_data-method$min_data)/(method$range[2]-method$range[1])+m
  }
  if(predict){
    return((method$range[2]-method$range[1])*(data-method$min_data)/(method$max_data-method$min_data) +
  }
  if(is.null(hard_min)){

    min_data=min(data)
  }else{
    min_data=hard_min
  }
  if(is.null(hard_max)){
  max_data=max(data)
  }else{
    max_data=hard_max
  }
  data_transformed = (range[2]-range[1])*(data-min_data)/(max_data-min_data)+range[1]
  return(list(transformed=data_transformed,
              min_data=min_data,
              max_data=max_data,
              range=range))

}
library(bestNormalize)
```

```
## Warning: package 'bestNormalize' was built under R version 4.0.5
```

```
##
## Attaching package: 'bestNormalize'
```

```
## The following object is masked from 'package:MASS':
##
##     boxcox
```

```
y_norm =bestNormalize(First_group$y)
y_normalized = y_norm$x.t
# set priors for the coefficients, intercept, and standard deviation
sigma = inverse_gamma(.01, .01)
```

```
## i Initialising python and checking dependencies, this may take a moment.
```

```
## v Initialising python and checking dependencies ... done!
```

```
##
```

```
int = normal(0, 10)
coef = laplace(0, 0.01, dim=nvar)
# define the expected value
greta_x = as_data(First_group$x)
greta_y = as_data(y_normalized)
mu = int + greta_x %*% coef
# indicate likelihood distribution with turncated normal
max_trunc = predict(y_norm, newdata=.5*112)
min_trunc = predict(y_norm, newdata=.125)
distribution(greta_y) = normal(mu, sigma, truncation=c(min_trunc,
                                                       max_trunc))
yTest_normalized = predict(y_norm, newdata=First_group$yTest)
# set up the expected values for the test data set
greta_y_test = as_data(yTest_normalized)
greta_x_test = as_data(First_group$xTest)
mup = int + greta_x_test %*% coef

# make model
m = model( coef, int, sigma)
```

```
draws = greta::mcmc(m, n_samples = 30000, n_cores=4, chains=2,
                    warmup=10000, verbose=FALSE)
```

```
effectiveSize(draws)
```

```
##  coef[1,1]  coef[2,1]  coef[3,1]  coef[4,1]  coef[5,1]  coef[6,1]  coef[7,1]
##  13510.447  17977.850   9468.987  14274.947  16186.747   8645.942  14689.909
##  coef[8,1]  coef[9,1] coef[10,1] coef[11,1] coef[12,1] coef[13,1] coef[14,1]
##  10205.833  13293.187  18851.006  11409.919   9829.178  16298.421  14371.776
## coef[15,1] coef[16,1] coef[17,1] coef[18,1] coef[19,1] coef[20,1] coef[21,1]
##  13463.939  12024.618  12070.569  11994.898  11703.681  10060.224  13492.648
## coef[22,1] coef[23,1] coef[24,1] coef[25,1] coef[26,1] coef[27,1] coef[28,1]
##  12560.240  14958.130  11975.124  13930.150  11792.216  12945.314  14150.723
## coef[29,1] coef[30,1] coef[31,1] coef[32,1] coef[33,1] coef[34,1] coef[35,1]
##  14162.921  14730.445  19305.104  12025.522  13125.113  14879.279  12133.622
## coef[36,1] coef[37,1] coef[38,1] coef[39,1] coef[40,1] coef[41,1] coef[42,1]
##  10813.887  14664.685  15020.678  14845.834  15311.540  10883.270  18461.099
## coef[43,1] coef[44,1] coef[45,1] coef[46,1] coef[47,1] coef[48,1] coef[49,1]
##  14906.366   9385.682  13607.303  12167.613  14094.402  12477.620  14321.662
## coef[50,1] coef[51,1] coef[52,1] coef[53,1] coef[54,1] coef[55,1] coef[56,1]
```

```
##  11407.386  16673.569  14905.843  14496.462  16014.570  15602.974  15725.004
## coef[57,1] coef[58,1] coef[59,1] coef[60,1] coef[61,1] coef[62,1] coef[63,1]
##  16095.492  12800.323  13123.973  13705.554  12282.178  16581.151  20088.319
## coef[64,1] coef[65,1] coef[66,1] coef[67,1] coef[68,1] coef[69,1] coef[70,1]
##  14137.040  17311.597  10782.889  11617.759  10554.971  14888.462  12347.142
## coef[71,1] coef[72,1] coef[73,1] coef[74,1] coef[75,1] coef[76,1] coef[77,1]
##  12208.676   8889.426  14845.098   9219.657  13694.493  10720.473  14568.978
## coef[78,1] coef[79,1] coef[80,1] coef[81,1]        int      sigma
##  10233.655  13774.973  10530.213  11151.674  18453.300  20328.763
```

```
gelman.diag(draws)
```

```
## Potential scale reduction factors:
##
##            Point est. Upper C.I.
## coef[1,1]           1       1.00
## coef[2,1]           1       1.00
## coef[3,1]           1       1.00
## coef[4,1]           1       1.00
## coef[5,1]           1       1.00
## coef[6,1]           1       1.00
## coef[7,1]           1       1.00
## coef[8,1]           1       1.00
## coef[9,1]           1       1.00
## coef[10,1]          1       1.00
## coef[11,1]          1       1.00
## coef[12,1]          1       1.00
## coef[13,1]          1       1.00
## coef[14,1]          1       1.01
## coef[15,1]          1       1.00
## coef[16,1]          1       1.00
## coef[17,1]          1       1.00
## coef[18,1]          1       1.00
## coef[19,1]          1       1.00
## coef[20,1]          1       1.00
## coef[21,1]          1       1.00
## coef[22,1]          1       1.00
## coef[23,1]          1       1.00
## coef[24,1]          1       1.00
## coef[25,1]          1       1.00
## coef[26,1]          1       1.00
## coef[27,1]          1       1.00
## coef[28,1]          1       1.00
## coef[29,1]          1       1.00
## coef[30,1]          1       1.00
## coef[31,1]          1       1.00
## coef[32,1]          1       1.00
## coef[33,1]          1       1.00
## coef[34,1]          1       1.00
## coef[35,1]          1       1.00
## coef[36,1]          1       1.00
## coef[37,1]          1       1.00
## coef[38,1]          1       1.00
## coef[39,1]          1       1.00
```

```
## coef[40,1]          1       1.00
## coef[41,1]          1       1.00
## coef[42,1]          1       1.00
## coef[43,1]          1       1.00
## coef[44,1]          1       1.00
## coef[45,1]          1       1.00
## coef[46,1]          1       1.00
## coef[47,1]          1       1.00
## coef[48,1]          1       1.00
## coef[49,1]          1       1.00
## coef[50,1]          1       1.00
## coef[51,1]          1       1.00
## coef[52,1]          1       1.00
## coef[53,1]          1       1.00
## coef[54,1]          1       1.00
## coef[55,1]          1       1.01
## coef[56,1]          1       1.00
## coef[57,1]          1       1.00
## coef[58,1]          1       1.00
## coef[59,1]          1       1.01
## coef[60,1]          1       1.00
## coef[61,1]          1       1.00
## coef[62,1]          1       1.00
## coef[63,1]          1       1.00
## coef[64,1]          1       1.00
## coef[65,1]          1       1.00
## coef[66,1]          1       1.01
## coef[67,1]          1       1.00
## coef[68,1]          1       1.00
## coef[69,1]          1       1.00
## coef[70,1]          1       1.00
## coef[71,1]          1       1.00
## coef[72,1]          1       1.00
## coef[73,1]          1       1.00
## coef[74,1]          1       1.00
## coef[75,1]          1       1.00
## coef[76,1]          1       1.01
## coef[77,1]          1       1.00
## coef[78,1]          1       1.00
## coef[79,1]          1       1.00
## coef[80,1]          1       1.00
## coef[81,1]          1       1.00
## int                 1       1.00
## sigma               1       1.00
##
## Multivariate psrf
##
## 1.01
```

```
geweke.diag(draws)
```

```
## $`11`
##
## Fraction in 1st window = 0.1
```

```
## Fraction in 2nd window = 0.5
##
##  coef[1,1]  coef[2,1]  coef[3,1]  coef[4,1]  coef[5,1]  coef[6,1]  coef[7,1]
##   0.982153  -0.732819   1.877030  -0.936197   1.300723   2.188288  -1.613929
##  coef[8,1]  coef[9,1] coef[10,1] coef[11,1] coef[12,1] coef[13,1] coef[14,1]
##  -0.098527  -0.331599   0.152291  -1.350800   0.042810   2.580055  -0.994805
## coef[15,1] coef[16,1] coef[17,1] coef[18,1] coef[19,1] coef[20,1] coef[21,1]
##   0.972360  -0.298673   0.431418  -0.590307   0.885120   0.341238  -0.427908
## coef[22,1] coef[23,1] coef[24,1] coef[25,1] coef[26,1] coef[27,1] coef[28,1]
##  -0.386431   0.598806  -0.001125   0.542471  -0.281206   0.997860   0.018461
## coef[29,1] coef[30,1] coef[31,1] coef[32,1] coef[33,1] coef[34,1] coef[35,1]
##   1.160834  -0.681716   0.955225   0.246482  -2.012560   0.178081  -2.588574
## coef[36,1] coef[37,1] coef[38,1] coef[39,1] coef[40,1] coef[41,1] coef[42,1]
##   0.458653   0.635355  -0.177801  -0.953543  -0.358860   2.041369   0.073127
## coef[43,1] coef[44,1] coef[45,1] coef[46,1] coef[47,1] coef[48,1] coef[49,1]
##  -0.465890  -1.042254  -1.450448   0.546874  -0.034081   0.606252  -1.681242
## coef[50,1] coef[51,1] coef[52,1] coef[53,1] coef[54,1] coef[55,1] coef[56,1]
##  -0.853707  -1.759283  -1.541936  -0.284200   1.025676  -0.779885  -0.104998
## coef[57,1] coef[58,1] coef[59,1] coef[60,1] coef[61,1] coef[62,1] coef[63,1]
##   1.142441   0.584284   0.129255  -0.137742  -1.806785   0.057208  -0.419274
## coef[64,1] coef[65,1] coef[66,1] coef[67,1] coef[68,1] coef[69,1] coef[70,1]
##  -0.451839  -0.286197   0.521400  -0.424013   0.667154   0.812078  -0.526448
## coef[71,1] coef[72,1] coef[73,1] coef[74,1] coef[75,1] coef[76,1] coef[77,1]
##   0.640389  -0.331382   0.662287  -0.589068  -0.221834  -0.105475   2.625018
## coef[78,1] coef[79,1] coef[80,1] coef[81,1]        int      sigma
##  -0.698625  -0.589046   1.560853  -0.441509  -0.374192   0.445203
##
##
## $`12`
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  coef[1,1]  coef[2,1]  coef[3,1]  coef[4,1]  coef[5,1]  coef[6,1]  coef[7,1]
##   -0.26428    0.17920   -0.62498    0.60556    1.24319   -0.41731   -0.97589
##  coef[8,1]  coef[9,1] coef[10,1] coef[11,1] coef[12,1] coef[13,1] coef[14,1]
##    0.72752   -0.83225   -0.63182    1.11104    1.63170   -0.53352    0.47065
## coef[15,1] coef[16,1] coef[17,1] coef[18,1] coef[19,1] coef[20,1] coef[21,1]
##   -1.24743   -1.05809   -0.29390    0.57184   -0.13889    0.84688   -1.09852
## coef[22,1] coef[23,1] coef[24,1] coef[25,1] coef[26,1] coef[27,1] coef[28,1]
##   -1.45348    1.98307   -0.35332   -1.01046    0.43811    0.71131    0.09944
## coef[29,1] coef[30,1] coef[31,1] coef[32,1] coef[33,1] coef[34,1] coef[35,1]
##    1.66050    0.08045    0.40665    0.64335    0.55410    0.08258    1.21956
## coef[36,1] coef[37,1] coef[38,1] coef[39,1] coef[40,1] coef[41,1] coef[42,1]
##    0.28672   -1.31994   -0.76152    0.67862   -0.93370    1.09731    0.38251
## coef[43,1] coef[44,1] coef[45,1] coef[46,1] coef[47,1] coef[48,1] coef[49,1]
##    0.61526   -1.54762   -0.97529    0.68829   -0.54449   -0.71959   -1.10466
## coef[50,1] coef[51,1] coef[52,1] coef[53,1] coef[54,1] coef[55,1] coef[56,1]
##    1.18065    0.63804    1.36208   -0.94855   -0.63019   -0.95816   -1.07332
## coef[57,1] coef[58,1] coef[59,1] coef[60,1] coef[61,1] coef[62,1] coef[63,1]
##    0.93651    0.02611   -1.97879    0.46265   -0.24671    0.63480   -0.23875
## coef[64,1] coef[65,1] coef[66,1] coef[67,1] coef[68,1] coef[69,1] coef[70,1]
##    0.80922    0.49110    1.85649   -0.01995   -0.38669   -0.70639    0.39030
## coef[71,1] coef[72,1] coef[73,1] coef[74,1] coef[75,1] coef[76,1] coef[77,1]
```

```
##   -0.64912   -0.06953    0.20500    0.56576   -0.37447    0.06614   -1.87084
## coef[78,1] coef[79,1] coef[80,1] coef[81,1]        int      sigma
##   -0.33529   -0.06566   -1.25073    0.83206   -1.71431   -0.62030
```

```
# sample of 30 trace plots
library(bayesplot)
```

```
## This is bayesplot version 1.9.0
```

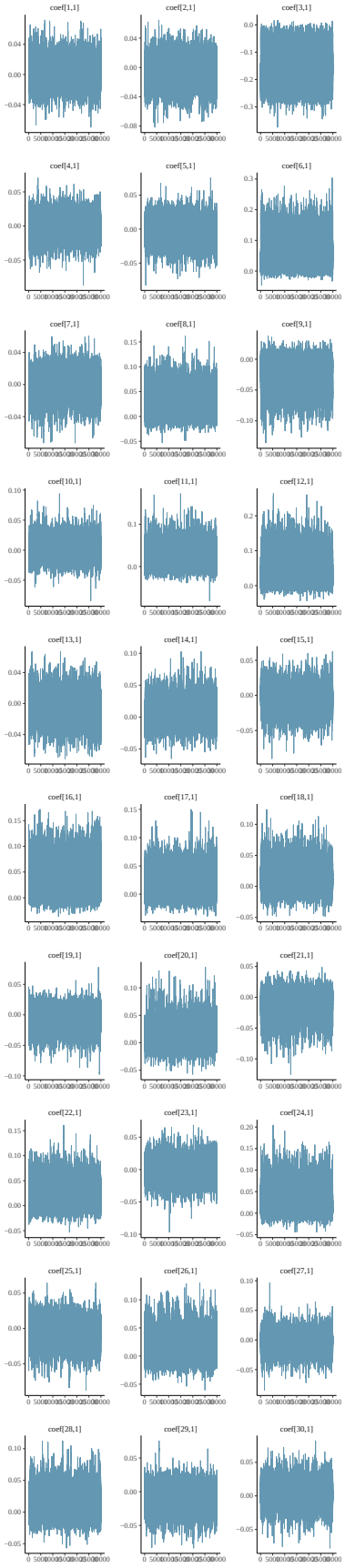
```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

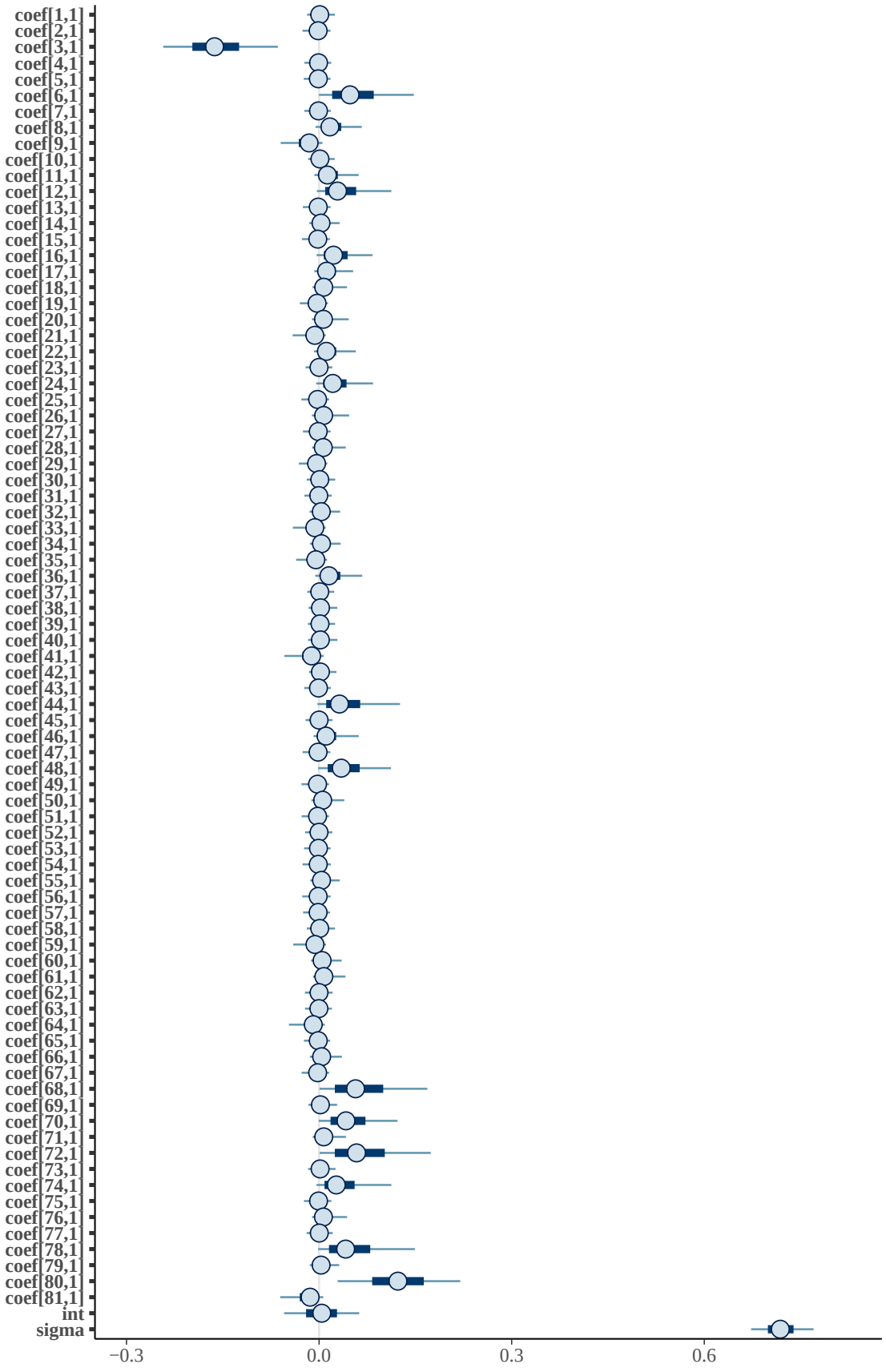
```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

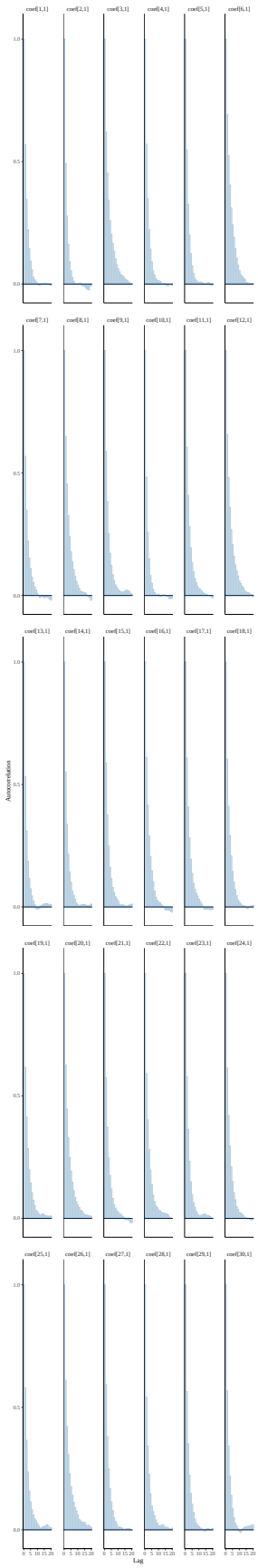
```
mcmc_trace(draws[[1]][,1:30], facet_args = list(ncol=3))
```


coef[1,1]
coef[2,1]
coef[3,1]
coef[4,1]
coef[5,1]
coef[6,1]
coef[7,1]
coef[8,1]
coef[9,1]
coef[10,1]
coef[11,1]
coef[12,1]
coef[13,1]
coef[14,1]
coef[15,1]
coef[16,1]
coef[17,1]
coef[18,1]
coef[19,1]
coef[20,1]
coef[21,1]
coef[22,1]
coef[23,1]
coef[24,1]
coef[25,1]
coef[26,1]
coef[27,1]
coef[28,1]
coef[29,1]
coef[30,1]

```
mcmc_intervals(draws)
```


coef[1,1]
coef[2,1]
coef[3,1]
coef[4,1]
coef[5,1]
coef[6,1]
coef[7,1]
coef[8,1]
coef[9,1]
coef[10,1]
coef[11,1]
coef[12,1]
coef[13,1]
coef[14,1]
coef[15,1]
coef[16,1]
coef[17,1]
coef[18,1]
coef[19,1]
coef[20,1]
coef[21,1]
coef[22,1]
coef[23,1]
coef[24,1]
coef[25,1]
coef[26,1]
coef[27,1]
coef[28,1]
coef[29,1]
coef[30,1]
coef[31,1]
coef[32,1]
coef[33,1]
coef[34,1]
coef[35,1]
coef[36,1]
coef[37,1]
coef[38,1]
coef[39,1]
coef[40,1]
coef[41,1]
coef[42,1]
coef[43,1]
coef[44,1]
coef[45,1]
coef[46,1]
coef[47,1]
coef[48,1]
coef[49,1]
coef[50,1]
coef[51,1]
coef[52,1]
coef[53,1]
coef[54,1]
coef[55,1]
coef[56,1]
coef[57,1]
coef[58,1]
coef[59,1]
coef[60,1]
coef[61,1]
coef[62,1]
coef[63,1]
coef[64,1]
coef[65,1]
coef[66,1]
coef[67,1]
coef[68,1]
coef[69,1]
coef[70,1]
coef[71,1]
coef[72,1]
coef[73,1]
coef[74,1]
coef[75,1]
coef[76,1]
coef[77,1]
coef[78,1]
coef[79,1]
coef[80,1]
coef[81,1]
int
sigma
−0.3
0.0
0.3
0.6

```
mcmc_acf_bar(draws[[1]][,1:30])
```


coef[1,1]
coef[2,1]
coef[3,1]
coef[4,1]
coef[5,1]
coef[6,1]
coef[7,1]
coef[8,1]
coef[9,1]
coef[10,1]
coef[11,1]
coef[12,1]
coef[13,1]
coef[14,1]
coef[15,1]
coef[16,1]
coef[17,1]
coef[18,1]
coef[19,1]
coef[20,1]
coef[21,1]
coef[22,1]
coef[23,1]
coef[24,1]
coef[25,1]
coef[26,1]
coef[27,1]
coef[28,1]
coef[29,1]
coef[30,1]
Autocorrelation
Lag

calculating DIC and doing predicitions

```
results = summary(draws)
results
```

```
##
## Iterations = 1:30000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 30000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                  Mean      SD  Naive SE Time-series SE
## coef[1,1]   1.835e-03 0.01316 5.372e-05      1.140e-04
## coef[2,1]  -2.602e-03 0.01335 5.449e-05      9.956e-05
## coef[3,1]  -1.595e-01 0.05380 2.196e-04      5.528e-04
## coef[4,1]  -1.273e-03 0.01262 5.151e-05      1.057e-04
## coef[5,1]  -2.012e-03 0.01266 5.169e-05      9.957e-05
## coef[6,1]   5.726e-02 0.04659 1.902e-04      5.011e-04
## coef[7,1]  -1.530e-03 0.01251 5.106e-05      1.033e-04
## coef[8,1]   2.191e-02 0.02292 9.358e-05      2.269e-04
## coef[9,1]  -1.991e-02 0.02074 8.467e-05      1.799e-04
## coef[10,1]  2.274e-03 0.01260 5.143e-05      9.176e-05
## coef[11,1]  1.831e-02 0.02201 8.986e-05      2.071e-04
## coef[12,1]  3.821e-02 0.03760 1.535e-04      3.793e-04
## coef[13,1] -2.293e-03 0.01321 5.392e-05      1.035e-04
## coef[14,1]  5.261e-03 0.01461 5.963e-05      1.219e-04
## coef[15,1] -3.255e-03 0.01329 5.426e-05      1.145e-04
## coef[16,1]  2.888e-02 0.02812 1.148e-04      2.564e-04
## coef[17,1]  1.595e-02 0.01932 7.887e-05      1.759e-04
## coef[18,1]  1.085e-02 0.01685 6.878e-05      1.538e-04
## coef[19,1] -5.119e-03 0.01355 5.532e-05      1.253e-04
## coef[20,1]  1.097e-02 0.01838 7.502e-05      1.832e-04
## coef[21,1] -1.006e-02 0.01630 6.654e-05      1.403e-04
## coef[22,1]  1.645e-02 0.02068 8.443e-05      1.845e-04
## coef[23,1] -1.669e-04 0.01258 5.137e-05      1.030e-04
## coef[24,1]  2.789e-02 0.02825 1.153e-04      2.581e-04
## coef[25,1] -3.834e-03 0.01321 5.395e-05      1.120e-04
## coef[26,1]  1.127e-02 0.01835 7.490e-05      1.702e-04
## coef[27,1] -2.255e-03 0.01311 5.351e-05      1.154e-04
## coef[28,1]  9.974e-03 0.01641 6.701e-05      1.380e-04
## coef[29,1] -6.109e-03 0.01373 5.605e-05      1.154e-04
## coef[30,1]  1.882e-03 0.01360 5.550e-05      1.121e-04
## coef[31,1] -9.710e-04 0.01295 5.289e-05      9.343e-05
## coef[32,1]  5.711e-03 0.01482 6.049e-05      1.351e-04
## coef[33,1] -9.991e-03 0.01607 6.561e-05      1.403e-04
## coef[34,1]  6.066e-03 0.01470 6.001e-05      1.206e-04
## coef[35,1] -7.766e-03 0.01504 6.140e-05      1.365e-04
## coef[36,1]  2.108e-02 0.02352 9.603e-05      2.271e-04
## coef[37,1]  1.771e-03 0.01266 5.166e-05      1.045e-04
## coef[38,1]  3.791e-03 0.01379 5.629e-05      1.125e-04
## coef[39,1]  2.432e-03 0.01297 5.294e-05      1.066e-04
```

```
## coef[40,1]  3.568e-03 0.01387 5.662e-05      1.122e-04
## coef[41,1] -1.600e-02 0.01941 7.924e-05      1.864e-04
## coef[42,1]  3.674e-03 0.01309 5.344e-05      9.635e-05
## coef[43,1] -1.586e-03 0.01265 5.165e-05      1.037e-04
## coef[44,1]  4.280e-02 0.04175 1.704e-04      4.322e-04
## coef[45,1]  1.223e-05 0.01281 5.230e-05      1.103e-04
## coef[46,1]  1.673e-02 0.02281 9.313e-05      2.080e-04
## coef[47,1] -2.594e-03 0.01311 5.351e-05      1.104e-04
## coef[48,1]  4.199e-02 0.03622 1.479e-04      3.243e-04
## coef[49,1] -3.905e-03 0.01315 5.369e-05      1.099e-04
## coef[50,1]  8.755e-03 0.01599 6.527e-05      1.499e-04
## coef[51,1] -3.887e-03 0.01312 5.358e-05      1.022e-04
## coef[52,1] -3.290e-04 0.01293 5.280e-05      1.060e-04
## coef[53,1] -1.719e-03 0.01260 5.144e-05      1.048e-04
## coef[54,1] -2.256e-03 0.01347 5.499e-05      1.064e-04
## coef[55,1]  6.071e-03 0.01410 5.756e-05      1.131e-04
## coef[56,1] -2.593e-03 0.01360 5.552e-05      1.085e-04
## coef[57,1] -2.561e-03 0.01263 5.157e-05      9.958e-05
## coef[58,1]  1.735e-03 0.01339 5.466e-05      1.184e-04
## coef[59,1] -9.614e-03 0.01586 6.475e-05      1.386e-04
## coef[60,1]  7.259e-03 0.01485 6.064e-05      1.269e-04
## coef[61,1]  1.074e-02 0.01586 6.475e-05      1.432e-04
## coef[62,1] -2.281e-04 0.01307 5.335e-05      1.015e-04
## coef[63,1] -6.039e-04 0.01259 5.142e-05      8.892e-05
## coef[64,1] -1.286e-02 0.01759 7.182e-05      1.482e-04
## coef[65,1] -2.122e-03 0.01249 5.099e-05      9.502e-05
## coef[66,1]  6.687e-03 0.01543 6.301e-05      1.486e-04
## coef[67,1] -3.694e-03 0.01313 5.361e-05      1.219e-04
## coef[68,1]  6.722e-02 0.05359 2.188e-04      5.216e-04
## coef[69,1]  3.594e-03 0.01375 5.613e-05      1.127e-04
## coef[70,1]  4.855e-02 0.03870 1.580e-04      3.485e-04
## coef[71,1]  1.059e-02 0.01629 6.649e-05      1.474e-04
## coef[72,1]  6.884e-02 0.05503 2.247e-04      5.838e-04
## coef[73,1]  2.788e-03 0.01306 5.334e-05      1.073e-04
## coef[74,1]  3.675e-02 0.03761 1.535e-04      3.922e-04
## coef[75,1] -1.444e-03 0.01289 5.264e-05      1.102e-04
## coef[76,1]  1.039e-02 0.01710 6.979e-05      1.653e-04
## coef[77,1]  5.920e-04 0.01230 5.022e-05      1.019e-04
## coef[78,1]  5.339e-02 0.04866 1.987e-04      4.811e-04
## coef[79,1]  5.238e-03 0.01409 5.751e-05      1.200e-04
## coef[80,1]  1.236e-01 0.05750 2.348e-04      5.608e-04
## coef[81,1] -1.892e-02 0.02157 8.808e-05      2.044e-04
## int         3.928e-03 0.03573 1.459e-04      2.630e-04
## sigma       7.196e-01 0.02963 1.210e-04      2.078e-04
##
## 2. Quantiles for each variable:
##
##                 2.5%        25%        50%        75%     97.5%
## coef[1,1]  -0.024390 -5.220e-03  9.888e-04  0.0083549  0.031145
## coef[2,1]  -0.033016 -9.094e-03 -1.437e-03  0.0044714  0.023124
## coef[3,1]  -0.255856 -1.974e-01 -1.627e-01 -0.1245774 -0.043776
## coef[4,1]  -0.028842 -7.621e-03 -8.138e-04  0.0053986  0.024634
## coef[5,1]  -0.030219 -8.409e-03 -1.180e-03  0.0046875  0.023332
## coef[6,1]  -0.003524  2.035e-02  4.813e-02  0.0851500  0.167932
```

```
## coef[7,1]  -0.029128 -7.857e-03 -9.082e-04  0.0051332  0.023662
## coef[8,1]  -0.009238  4.801e-03  1.688e-02  0.0344878  0.077694
## coef[9,1]  -0.070267 -3.165e-02 -1.557e-02 -0.0043280  0.008941
## coef[10,1] -0.022407 -4.545e-03  1.326e-03  0.0085107  0.030608
## coef[11,1] -0.011227  2.538e-03  1.296e-02  0.0292796  0.073519
## coef[12,1] -0.007181  9.509e-03  2.875e-02  0.0577159  0.132977
## coef[13,1] -0.032159 -8.815e-03 -1.298e-03  0.0047570  0.023695
## coef[14,1] -0.020524 -2.965e-03  2.978e-03  0.0121320  0.039852
## coef[15,1] -0.033424 -9.817e-03 -2.026e-03  0.0040420  0.021931
## coef[16,1] -0.007495  7.361e-03  2.251e-02  0.0444619  0.097578
## coef[17,1] -0.011494  2.129e-03  1.166e-02  0.0260244  0.063677
## coef[18,1] -0.014526 -1.585e-04  7.215e-03  0.0189674  0.052906
## coef[19,1] -0.037363 -1.164e-02 -3.153e-03  0.0026891  0.018761
## coef[20,1] -0.015777 -5.053e-04  6.814e-03  0.0190105  0.057899
## coef[21,1] -0.050266 -1.784e-02 -6.643e-03  0.0003974  0.014447
## coef[22,1] -0.011768  1.831e-03  1.140e-02  0.0266418  0.068576
## coef[23,1] -0.026803 -6.718e-03 -6.620e-05  0.0063217  0.026455
## coef[24,1] -0.008111  6.432e-03  2.130e-02  0.0427803  0.098309
## coef[25,1] -0.034570 -1.034e-02 -2.323e-03  0.0035607  0.020804
## coef[26,1] -0.015833 -3.367e-04  7.093e-03  0.0194822  0.057779
## coef[27,1] -0.031816 -8.702e-03 -1.307e-03  0.0046962  0.023497
## coef[28,1] -0.015209 -4.778e-04  6.547e-03  0.0176895  0.051104
## coef[29,1] -0.038970 -1.288e-02 -3.909e-03  0.0021022  0.017348
## coef[30,1] -0.025450 -5.181e-03  1.077e-03  0.0084784  0.031708
## coef[31,1] -0.029181 -7.455e-03 -5.083e-04  0.0057071  0.025239
## coef[32,1] -0.020122 -2.716e-03  3.423e-03  0.0127192  0.040908
## coef[33,1] -0.049764 -1.777e-02 -6.640e-03  0.0003564  0.014836
## coef[34,1] -0.019558 -2.502e-03  3.723e-03  0.0129630  0.041848
## coef[35,1] -0.043879 -1.507e-02 -5.044e-03  0.0014145  0.017024
## coef[36,1] -0.009891  3.784e-03  1.539e-02  0.0331040  0.079906
## coef[37,1] -0.023519 -4.977e-03  1.064e-03  0.0081986  0.029922
## coef[38,1] -0.021715 -3.774e-03  2.207e-03  0.0103973  0.035874
## coef[39,1] -0.022581 -4.588e-03  1.431e-03  0.0088356  0.031860
## coef[40,1] -0.022198 -4.038e-03  1.984e-03  0.0102591  0.035965
## coef[41,1] -0.063807 -2.603e-02 -1.164e-02 -0.0021036  0.011302
## coef[42,1] -0.020639 -3.586e-03  2.094e-03  0.0101059  0.033561
## coef[43,1] -0.029717 -7.939e-03 -9.170e-04  0.0050631  0.023661
## coef[44,1] -0.006288  1.110e-02  3.171e-02  0.0640153  0.148288
## coef[45,1] -0.027031 -6.463e-03  2.568e-05  0.0064890  0.027033
## coef[46,1] -0.012849  1.313e-03  1.079e-02  0.0265743  0.076300
## coef[47,1] -0.032437 -8.998e-03 -1.540e-03  0.0044800  0.023018
## coef[48,1] -0.005073  1.349e-02  3.451e-02  0.0632490  0.128552
## coef[49,1] -0.034169 -1.043e-02 -2.403e-03  0.0034636  0.020556
## coef[50,1] -0.016721 -1.052e-03  5.588e-03  0.0163706  0.048338
## coef[51,1] -0.034345 -1.027e-02 -2.312e-03  0.0034343  0.020243
## coef[52,1] -0.028064 -6.845e-03 -1.014e-04  0.0063811  0.026764
## coef[53,1] -0.029432 -8.097e-03 -9.284e-04  0.0050878  0.023488
## coef[54,1] -0.032548 -8.799e-03 -1.266e-03  0.0048336  0.024240
## coef[55,1] -0.018256 -2.101e-03  3.780e-03  0.0128663  0.039663
## coef[56,1] -0.033457 -9.164e-03 -1.404e-03  0.0046255  0.023587
## coef[57,1] -0.031282 -8.910e-03 -1.492e-03  0.0042995  0.022174
## coef[58,1] -0.024992 -5.244e-03  9.713e-04  0.0081814  0.031388
## coef[59,1] -0.048970 -1.727e-02 -6.361e-03  0.0005262  0.014956
## coef[60,1] -0.017123 -1.819e-03  4.675e-03  0.0144797  0.043226
```

```
## coef[61,1] -0.013466  1.569e-04  7.465e-03  0.0188768  0.049563
## coef[62,1] -0.028301 -6.727e-03 -8.787e-05  0.0064804  0.027117
## coef[63,1] -0.028041 -6.916e-03 -2.800e-04  0.0060014  0.025578
## coef[64,1] -0.056015 -2.189e-02 -9.062e-03 -0.0007744  0.013229
## coef[65,1] -0.030020 -8.378e-03 -1.245e-03  0.0046447  0.022308
## coef[66,1] -0.019288 -2.299e-03  3.980e-03  0.0138744  0.044475
## coef[67,1] -0.034052 -1.004e-02 -2.229e-03  0.0035165  0.020890
## coef[68,1] -0.002893  2.456e-02  5.682e-02  0.0998994  0.194040
## coef[69,1] -0.021719 -3.925e-03  2.054e-03  0.0101852  0.035391
## coef[70,1] -0.003710  1.782e-02  4.188e-02  0.0722291  0.137940
## coef[71,1] -0.014017 -8.362e-06  7.268e-03  0.0185380  0.050880
## coef[72,1] -0.002665  2.450e-02  5.836e-02  0.1023380  0.199326
## coef[73,1] -0.022325 -4.230e-03  1.582e-03  0.0090427  0.032717
## coef[74,1] -0.007850  8.431e-03  2.680e-02  0.0553344  0.133157
## coef[75,1] -0.029776 -7.875e-03 -7.911e-04  0.0053271  0.024713
## coef[76,1] -0.015430 -5.180e-04  6.758e-03  0.0183352  0.053436
## coef[77,1] -0.024748 -5.815e-03  3.212e-04  0.0068400  0.026974
## coef[78,1] -0.004889  1.546e-02  4.138e-02  0.0797074  0.172867
## coef[79,1] -0.019830 -2.717e-03  3.130e-03  0.0119218  0.038840
## coef[80,1]  0.016125  8.271e-02  1.228e-01  0.1630958  0.238412
## coef[81,1] -0.072376 -3.008e-02 -1.396e-02 -0.0031564  0.010216
## int        -0.066334 -2.027e-02  4.173e-03  0.0279093  0.074229
## sigma       0.664630  6.991e-01  7.184e-01  0.7390100  0.780725
```

```
intercept = results$statistics['int', 'Mean']
shape = results$statistics['sigma' , 'Mean']
```

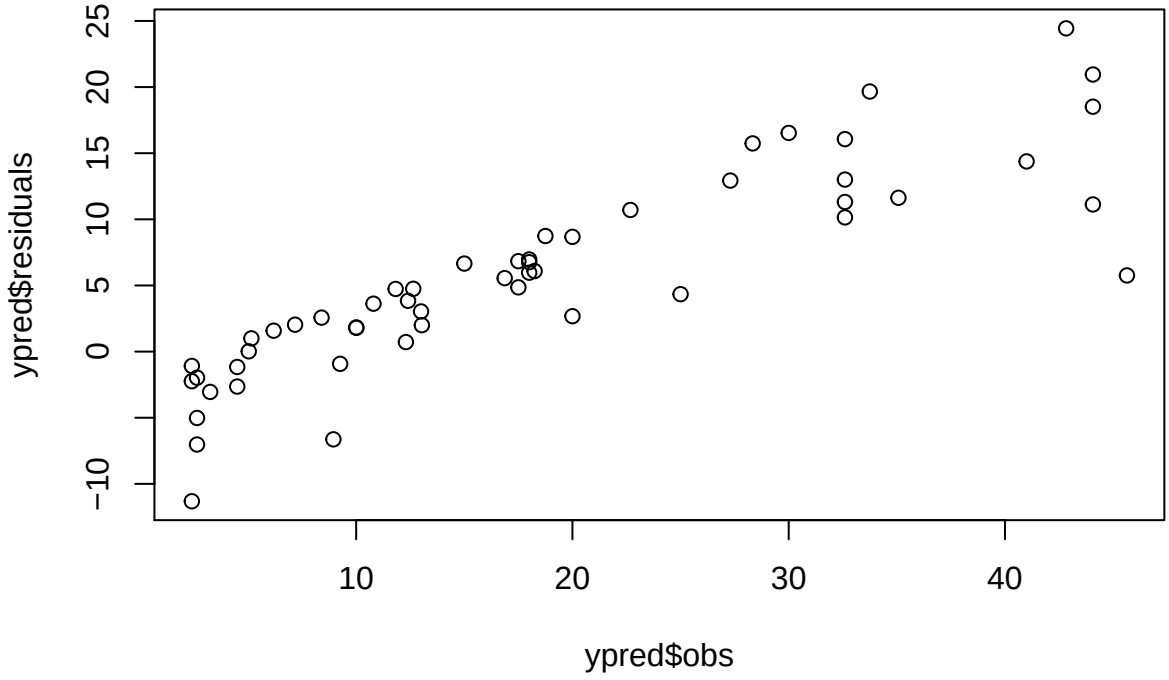
```
# get table of just quanitles for coefficients
coefs = results$quantiles[1:(nrow(results$quantiles)-2), ]
coefs = data.table(coefs, keep.rownames = TRUE)
coefs[,significant:=ifelse((0 < `97.5%`)&(0>`2.5%`), 0, 1)]
vars_keep = (coefs[significant==1,rn])
vars_keep_bool = coefs$significant==1
vars_names = colnames(First_group$x)[vars_keep_bool]
vars_names
```

```
## [1] "signAge" "WAMP"
```

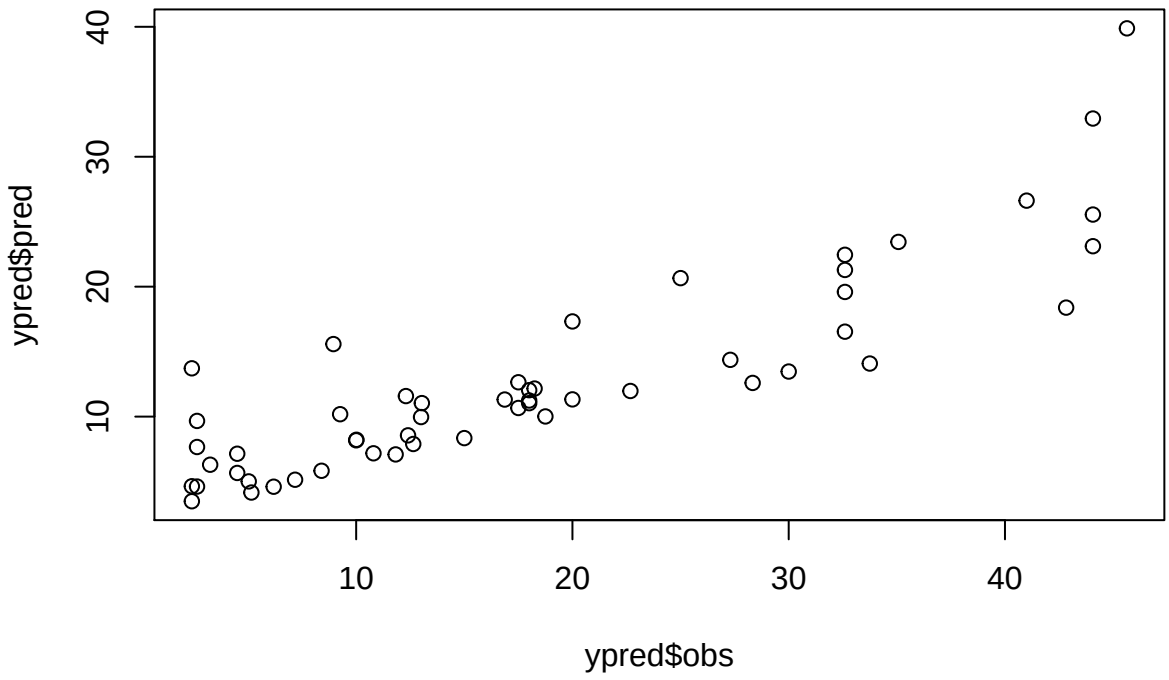
```
ypred = calculate(mup, values=draws, trace_batch_size = 250, nsim=100)
ypred = colMeans(ypred$mup)
ypred = predict(y_norm, ypred, inverse=TRUE)
ypred = data.frame(pred=ypred, obs=First_group$yTest)
caret::postResample(pred=ypred, obs=First_group$yTest)
```

```
##     RMSE Rsquared      MAE
##       NA 0.762942       NA
```

```
# make residuals
ypred$residuals = (ypred$obs-ypred$pred)
plot(ypred$obs, ypred$residuals)
```




```
plot(ypred$obs, ypred$pred)
```




```
test2 = test[groups==1, ]
train2 = train[groups==1,]
Second_group = make_train_test_data(data=train2,
                                    test_data=test2, target='aav',
                                    remove_cols=c('totalValue', 'aav',
                                                  'fullName','termSecondYear',
                                                  'termFirstYear', 'groups'),
                                    )
y_norm =bestNormalize(Second_group$y)
y_normalized = y_norm$x.t
# set priors for the coefficients, intercept, the multiplicative constants
# in the beta
sigma = inverse_gamma(.01, .01)
int = normal(0, 10)
coef = laplace(0, 0.01, dim=nvar)
# define the expected value
greta_x = as_data(Second_group$x)
greta_y = as_data(y_normalized)
mu = int + greta_x %*% coef
# indicated likelihood distribution
max_trunc = predict(y_norm, newdata=.5*112)
min_trunc = predict(y_norm, newdata=.125)
distribution(greta_y) = normal(mu, sigma, truncation=c(min_trunc,
                                                       max_trunc))
yTest_normalized = predict(y_norm, newdata=Second_group$yTest)
greta_y_test = as_data(yTest_normalized)
```

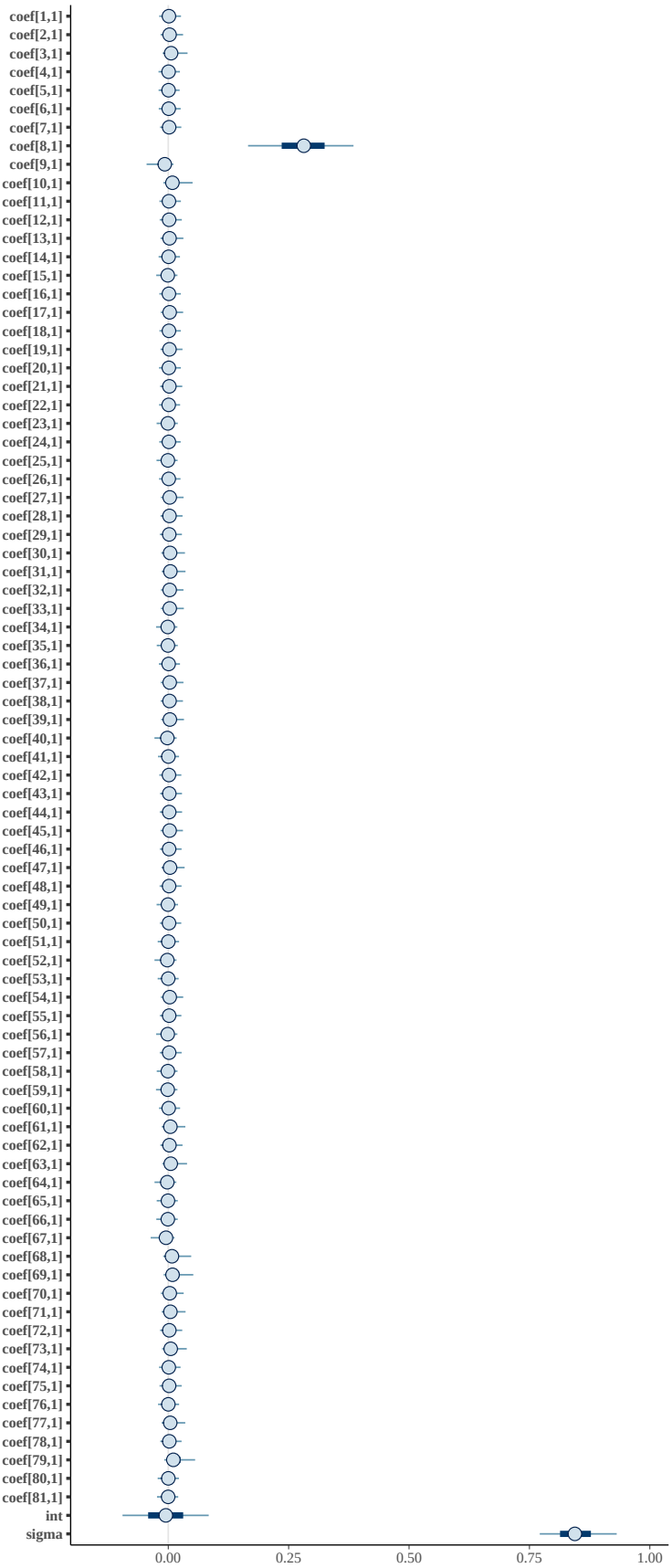
```
greta_x_test = as_data(Second_group$xTest)
mup = int + greta_x_test %*% coef

# make model
m = model( coef, int, sigma)
```

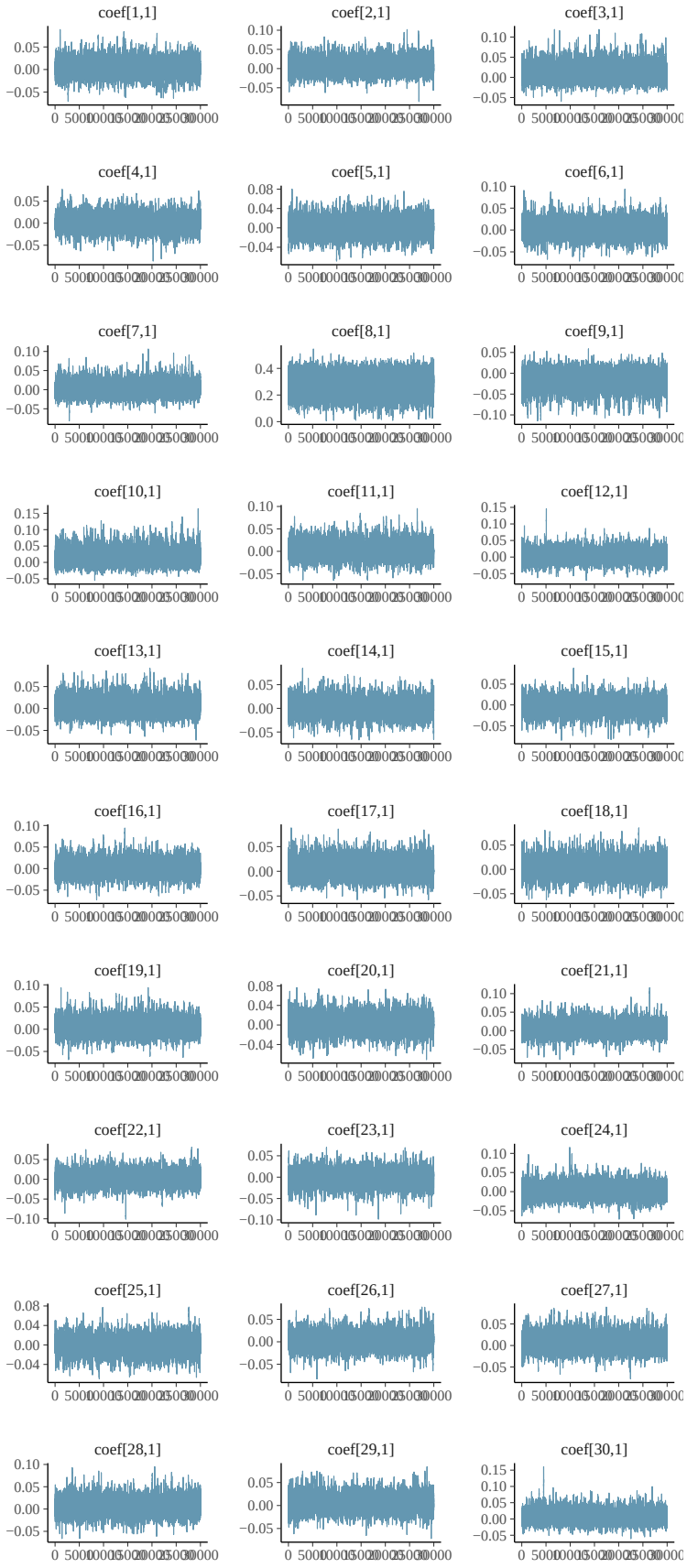
```
draws = greta::mcmc(m, n_samples = 30000, n_cores=4, chains=2,
                    warmup=10000, verbose=FALSE)
effectiveSize(draws)
```

```
##  coef[1,1]  coef[2,1]  coef[3,1]  coef[4,1]  coef[5,1]  coef[6,1]  coef[7,1]
##  11786.067  12311.654  11186.574  11052.632  14540.144  10714.368  13805.788
##  coef[8,1]  coef[9,1] coef[10,1] coef[11,1] coef[12,1] coef[13,1] coef[14,1]
##  11743.735   9644.888  10001.339  11931.969  10132.681  10411.603  12618.098
## coef[15,1] coef[16,1] coef[17,1] coef[18,1] coef[19,1] coef[20,1] coef[21,1]
##  13259.093  11633.063  13936.106  11899.928  11724.219  10855.622  10760.466
## coef[22,1] coef[23,1] coef[24,1] coef[25,1] coef[26,1] coef[27,1] coef[28,1]
##  10339.848  13740.288  13229.654  10507.982  10872.873  11819.555  11562.007
## coef[29,1] coef[30,1] coef[31,1] coef[32,1] coef[33,1] coef[34,1] coef[35,1]
##  10094.555  10032.606  10989.644   9601.088  11617.097  11719.086  12872.772
## coef[36,1] coef[37,1] coef[38,1] coef[39,1] coef[40,1] coef[41,1] coef[42,1]
##  11875.303   8760.259  12148.986  11662.305  12365.689  13524.857  11090.815
## coef[43,1] coef[44,1] coef[45,1] coef[46,1] coef[47,1] coef[48,1] coef[49,1]
##  11571.729   9941.109  12272.533  11715.428  12001.694  12379.715  11029.574
## coef[50,1] coef[51,1] coef[52,1] coef[53,1] coef[54,1] coef[55,1] coef[56,1]
##  11852.042  10661.925  10662.616  12676.791  11838.567  11426.455  10649.712
## coef[57,1] coef[58,1] coef[59,1] coef[60,1] coef[61,1] coef[62,1] coef[63,1]
##  11102.397  11409.684  10573.223  12080.644  10766.316   9894.893   8756.412
## coef[64,1] coef[65,1] coef[66,1] coef[67,1] coef[68,1] coef[69,1] coef[70,1]
##  15040.630  12662.640  10685.000  10552.983   8812.989   9228.840   9507.295
## coef[71,1] coef[72,1] coef[73,1] coef[74,1] coef[75,1] coef[76,1] coef[77,1]
##   9684.398  11177.266   8683.186  15003.549  10375.963  13790.459  10419.096
## coef[78,1] coef[79,1] coef[80,1] coef[81,1]        int      sigma
##   9920.131   9055.389  11683.443  12033.366  13558.250  11826.365
```

```
mcmc_intervals(draws)
```


coef[1,1]
coef[2,1]
coef[3,1]
coef[4,1]
coef[5,1]
coef[6,1]
coef[7,1]
coef[8,1]
coef[9,1]
coef[10,1]
coef[11,1]
coef[12,1]
coef[13,1]
coef[14,1]
coef[15,1]
coef[16,1]
coef[17,1]
coef[18,1]
coef[19,1]
coef[20,1]
coef[21,1]
coef[22,1]
coef[23,1]
coef[24,1]
coef[25,1]
coef[26,1]
coef[27,1]
coef[28,1]
coef[29,1]
coef[30,1]
coef[31,1]
coef[32,1]
coef[33,1]
coef[34,1]
coef[35,1]
coef[36,1]
coef[37,1]
coef[38,1]
coef[39,1]
coef[40,1]
coef[41,1]
coef[42,1]
coef[43,1]
coef[44,1]
coef[45,1]
coef[46,1]
coef[47,1]
coef[48,1]
coef[49,1]
coef[50,1]
coef[51,1]
coef[52,1]
coef[53,1]
coef[54,1]
coef[55,1]
coef[56,1]
coef[57,1]
coef[58,1]
coef[59,1]
coef[60,1]
coef[61,1]
coef[62,1]
coef[63,1]
coef[64,1]
coef[65,1]
coef[66,1]
coef[67,1]
coef[68,1]
coef[69,1]
coef[70,1]
coef[71,1]
coef[72,1]
coef[73,1]
coef[74,1]
coef[75,1]
coef[76,1]
coef[77,1]
coef[78,1]
coef[79,1]
coef[80,1]
coef[81,1]
int
sigma
0.00
0.25
0.50
0.75
1.00

```
mcmc_trace(draws[[1]][,1:30], facet_args = list(ncol=3))
```


coef[1,1]
coef[2,1]
coef[3,1]
coef[4,1]
coef[5,1]
coef[6,1]
coef[7,1]
coef[8,1]
coef[9,1]
coef[10,1]
coef[11,1]
coef[12,1]
coef[13,1]
coef[14,1]
coef[15,1]
coef[16,1]
coef[17,1]
coef[18,1]
coef[19,1]
coef[20,1]
coef[21,1]
coef[22,1]
coef[23,1]
coef[24,1]
coef[25,1]
coef[26,1]
coef[27,1]
coef[28,1]
coef[29,1]
coef[30,1]

```
results = summary(draws)
```

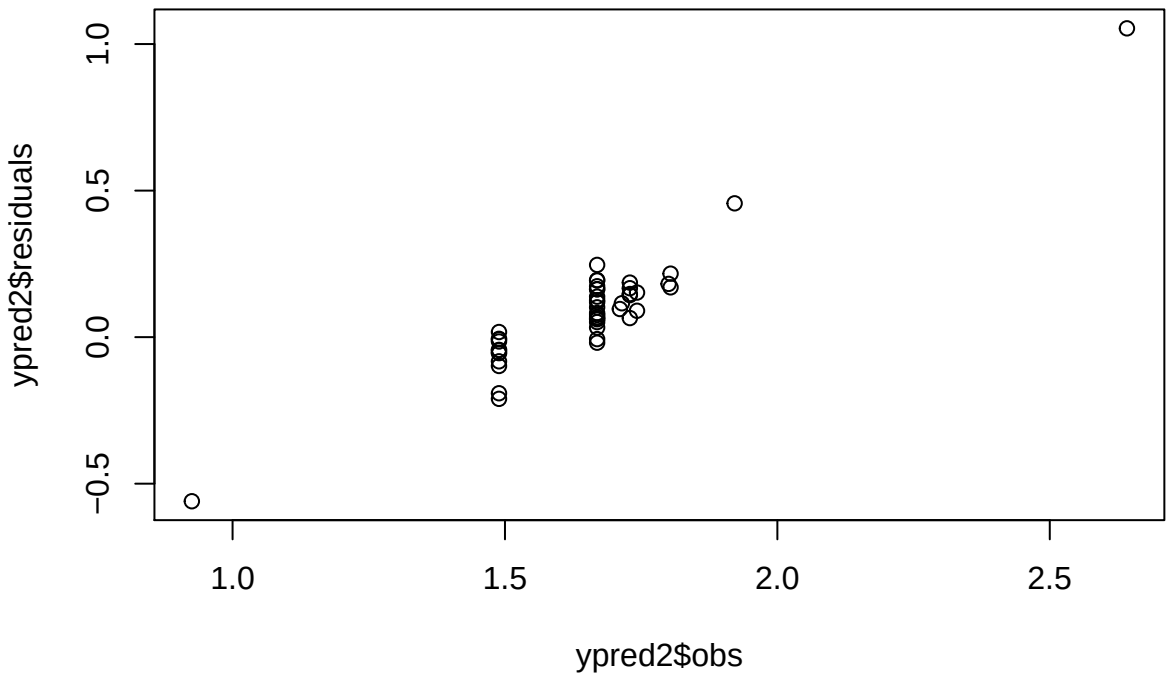
```
# get table of just quanitles for coefficients
coefs = results$quantiles[1:(nrow(results$quantiles)-2), ]
coefs = data.table(coefs, keep.rownames = TRUE)
coefs[,significant:=ifelse((0 < `97.5%`)&(0>`2.5%`), 0, 1)]
vars_keep = (coefs[significant==1,rn])
vars_keep_bool = coefs$significant==1
vars_names = colnames(First_group$x)[vars_keep_bool]
vars_names
```

```
## [1] "signYear"
```

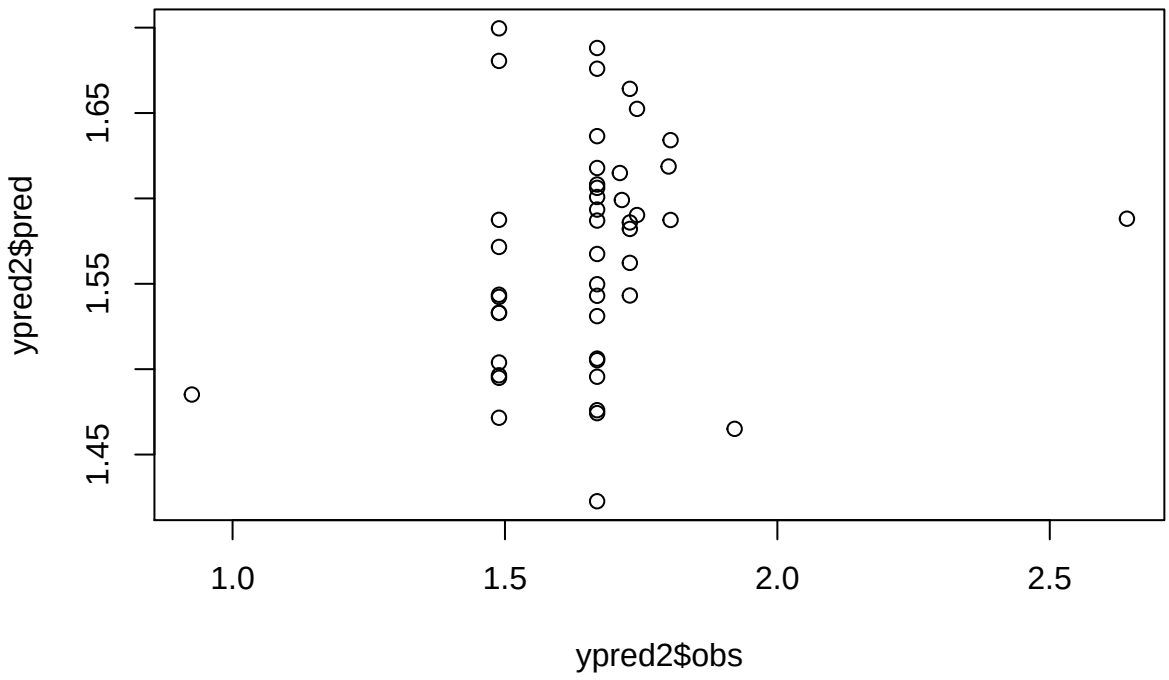
```
# get predictionss
ypred2 = calculate(mup, values=draws, trace_batch_size = 250, nsim=100)
ypred2 = colMeans(ypred2$mup)
ypred2 = predict(y_norm, ypred2, inverse=TRUE)
ypred2 = data.frame(pred=ypred2, obs=Second_group$yTest)
caret::postResample(pred=ypred2, obs=Second_group$yTest)
```

```
##       RMSE   Rsquared        MAE
##         NA 0.03540267         NA
```

```
# make residuals
ypred2$residuals = (ypred2$obs-ypred2$pred)
#ypred2 = ypred[ypred$obs<10,]
plot(ypred2$obs, ypred2$residuals)
```




```
plot(ypred2$obs, ypred2$pred)
```




```
#caret::postResample(pred=ypred2$pred, ypred2$obs)
```

dwight powell sign year 2016 is the extreme outlier here

```
test3 = test[groups==3, ]
train3 = train[groups==3,]
third_group = make_train_test_data(data=train3,
                                   test_data=test3, target='aav',
                                   remove_cols=c('totalValue', 'aav',
                                                 'fullName','termSecondYear',
                                                 'termFirstYear', 'groups'),
                                   )
y_norm =bestNormalize(third_group$y)
y_normalized = y_norm$x.t
# set priors for the coefficients, intercept, the multiplicative constants
# in the beta
sigma = inverse_gamma(.01, .01)
int = normal(0, 10)
coef = laplace(0, 0.01, dim=nvar)
# define the expected value
greta_x = as_data(third_group$x)
greta_y = as_data(y_normalized)
mu = int + greta_x %*% coef
# indicated likelihood distribution
max_trunc = predict(y_norm, newdata=.5*112)
```

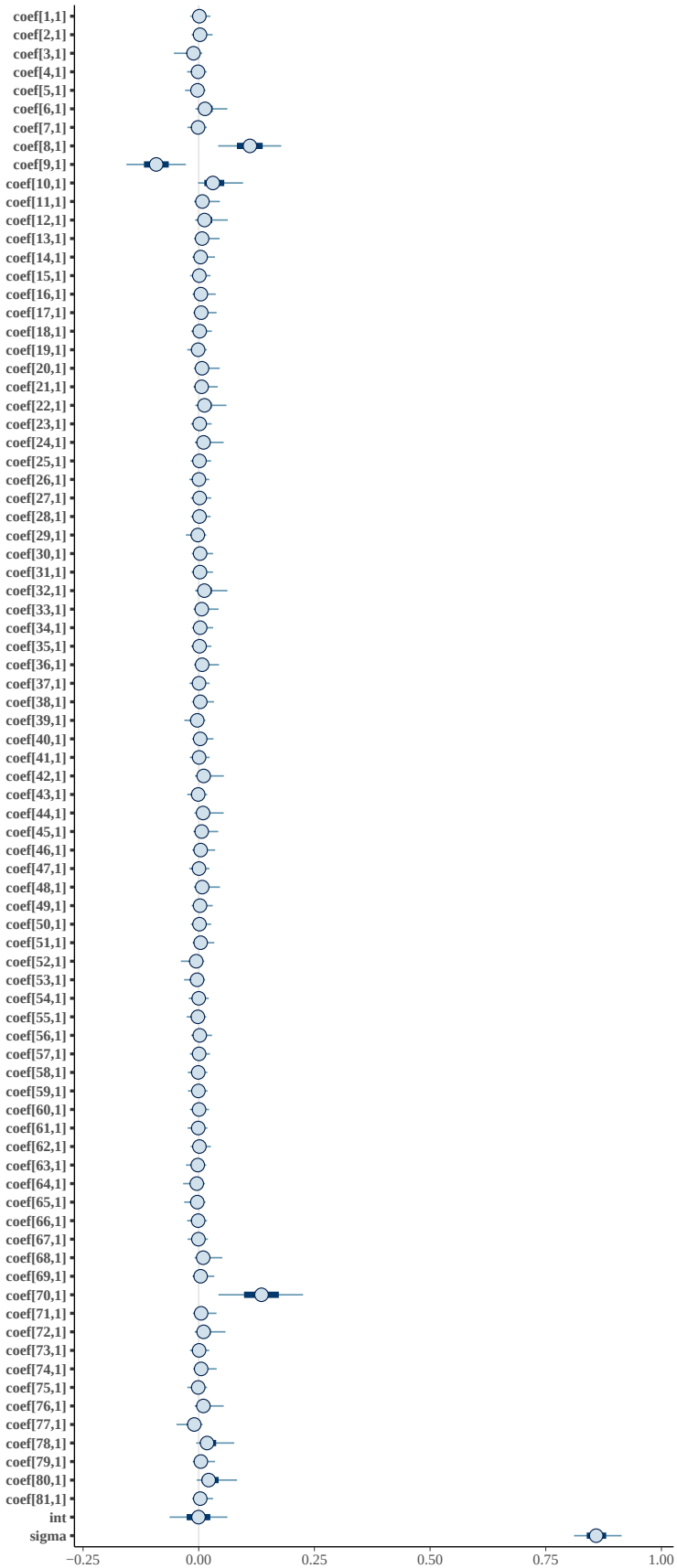
```
min_trunc = predict(y_norm, newdata=.125)
distribution(greta_y) = normal(mu, sigma, truncation=c(min_trunc,
                                                       max_trunc))
yTest_normalized = predict(y_norm, newdata=third_group$yTest)
greta_y_test = as_data(yTest_normalized)
greta_x_test = as_data(third_group$xTest)
mup = int + greta_x_test %*% coef

# make model
m = model( coef, int, sigma)
```

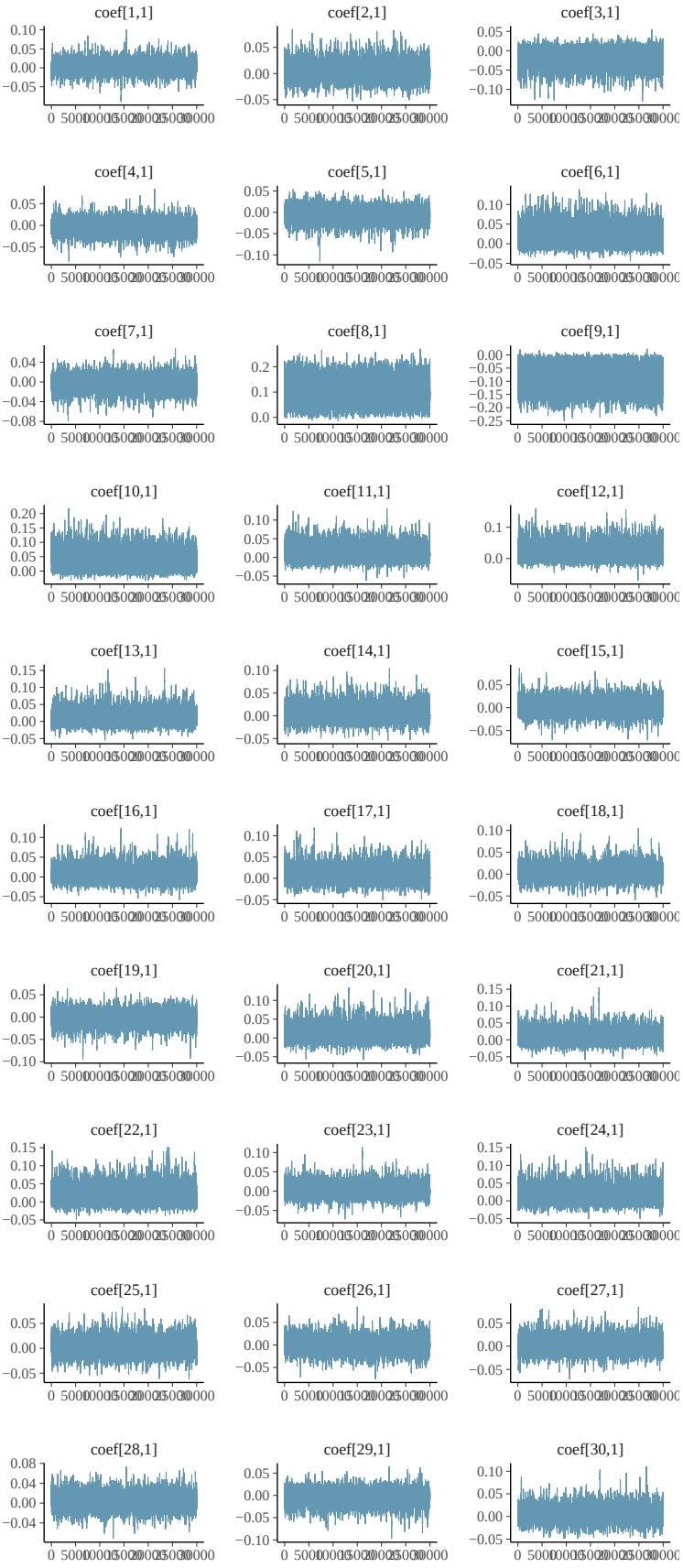
```
draws = greta::mcmc(m, n_samples = 30000, n_cores=4, chains=2,
                    warmup=10000, verbose=FALSE)
effectiveSize(draws)
```

```
##  coef[1,1]  coef[2,1]  coef[3,1]  coef[4,1]  coef[5,1]  coef[6,1]  coef[7,1]
##   8952.963   7324.968   6417.785   8784.787   7919.861   6992.204   8387.209
##  coef[8,1]  coef[9,1] coef[10,1] coef[11,1] coef[12,1] coef[13,1] coef[14,1]
##   8905.229   9774.183   6299.624   7830.469   5878.875   6674.245   7257.035
## coef[15,1] coef[16,1] coef[17,1] coef[18,1] coef[19,1] coef[20,1] coef[21,1]
##   7485.595   6777.946   6776.212   7242.547   9382.759   5369.259   7502.126
## coef[22,1] coef[23,1] coef[24,1] coef[25,1] coef[26,1] coef[27,1] coef[28,1]
##   6159.586   9900.739   5892.790   7946.994   9703.073   7636.197   7891.219
## coef[29,1] coef[30,1] coef[31,1] coef[32,1] coef[33,1] coef[34,1] coef[35,1]
##   7192.500   6603.710   8173.673   4869.492   6363.984   8574.169   8430.443
## coef[36,1] coef[37,1] coef[38,1] coef[39,1] coef[40,1] coef[41,1] coef[42,1]
##   6659.731   7857.593   6802.680   9453.811   8726.440   8232.750   6512.301
## coef[43,1] coef[44,1] coef[45,1] coef[46,1] coef[47,1] coef[48,1] coef[49,1]
##   7407.876   5831.474   6834.454   6793.335   7834.204   5776.100   8388.885
## coef[50,1] coef[51,1] coef[52,1] coef[53,1] coef[54,1] coef[55,1] coef[56,1]
##   8699.862   7149.970   7328.171   9600.774   9526.767   9624.224   7341.968
## coef[57,1] coef[58,1] coef[59,1] coef[60,1] coef[61,1] coef[62,1] coef[63,1]
##   7156.389   8451.866   9812.307   8388.929   8670.769   8874.338   7439.947
## coef[64,1] coef[65,1] coef[66,1] coef[67,1] coef[68,1] coef[69,1] coef[70,1]
##   7584.661   7735.781   8440.495   7996.648   7847.899   7708.798   6022.073
## coef[71,1] coef[72,1] coef[73,1] coef[74,1] coef[75,1] coef[76,1] coef[77,1]
##   6644.141   5081.786   7782.153   5555.581   8840.495   6264.708   6532.353
## coef[78,1] coef[79,1] coef[80,1] coef[81,1]        int      sigma
##   5424.302   8032.831   5927.051   8274.478  10101.715   9299.313
```

```
mcmc_intervals(draws)
```


coef[1,1]
coef[2,1]
coef[3,1]
coef[4,1]
coef[5,1]
coef[6,1]
coef[7,1]
coef[8,1]
coef[9,1]
coef[10,1]
coef[11,1]
coef[12,1]
coef[13,1]
coef[14,1]
coef[15,1]
coef[16,1]
coef[17,1]
coef[18,1]
coef[19,1]
coef[20,1]
coef[21,1]
coef[22,1]
coef[23,1]
coef[24,1]
coef[25,1]
coef[26,1]
coef[27,1]
coef[28,1]
coef[29,1]
coef[30,1]
coef[31,1]
coef[32,1]
coef[33,1]
coef[34,1]
coef[35,1]
coef[36,1]
coef[37,1]
coef[38,1]
coef[39,1]
coef[40,1]
coef[41,1]
coef[42,1]
coef[43,1]
coef[44,1]
coef[45,1]
coef[46,1]
coef[47,1]
coef[48,1]
coef[49,1]
coef[50,1]
coef[51,1]
coef[52,1]
coef[53,1]
coef[54,1]
coef[55,1]
coef[56,1]
coef[57,1]
coef[58,1]
coef[59,1]
coef[60,1]
coef[61,1]
coef[62,1]
coef[63,1]
coef[64,1]
coef[65,1]
coef[66,1]
coef[67,1]
coef[68,1]
coef[69,1]
coef[70,1]
coef[71,1]
coef[72,1]
coef[73,1]
coef[74,1]
coef[75,1]
coef[76,1]
coef[77,1]
coef[78,1]
coef[79,1]
coef[80,1]
coef[81,1]
int
sigma
−0.25
0.00
0.25
0.50
0.75
1.00

```
mcmc_trace(draws[[1]][,1:30], facet_args = list(ncol=3))
```


coef[1,1]
coef[2,1]
coef[3,1]
coef[4,1]
coef[5,1]
coef[6,1]
coef[7,1]
coef[8,1]
coef[9,1]
coef[10,1]
coef[11,1]
coef[12,1]
coef[13,1]
coef[14,1]
coef[15,1]
coef[16,1]
coef[17,1]
coef[18,1]
coef[19,1]
coef[20,1]
coef[21,1]
coef[22,1]
coef[23,1]
coef[24,1]
coef[25,1]
coef[26,1]
coef[27,1]
coef[28,1]
coef[29,1]
coef[30,1]

```
results = summary(draws)
```

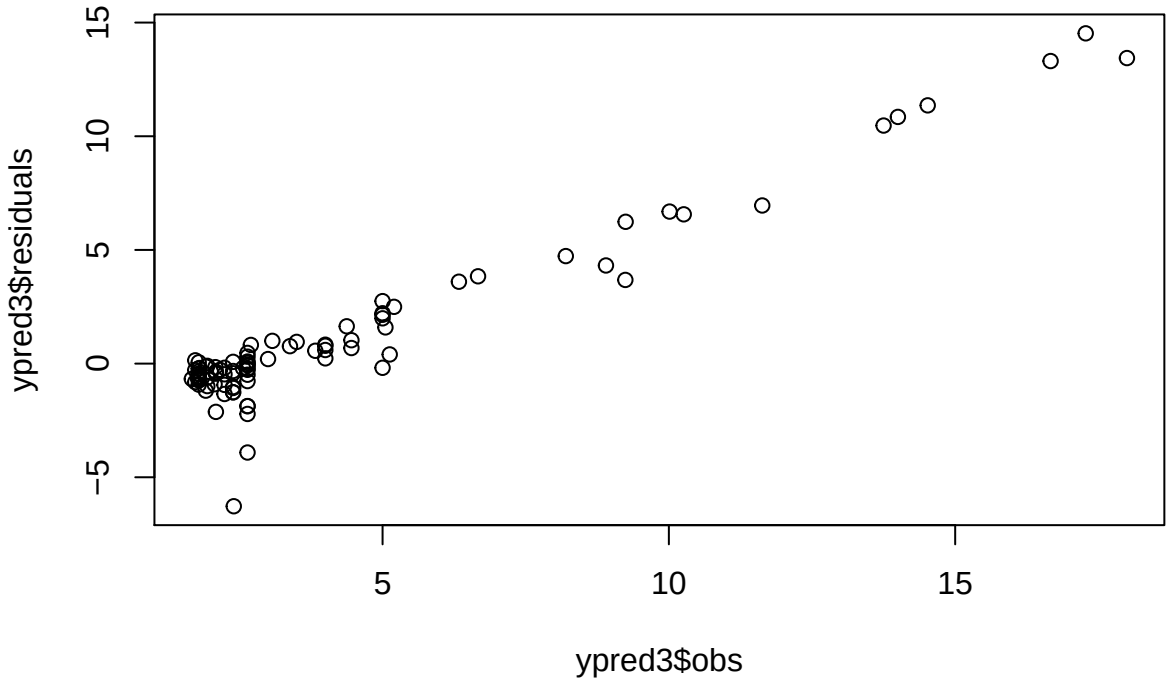
```
# get table of just quanitles for coefficients
coefs = results$quantiles[1:(nrow(results$quantiles)-2), ]
coefs = data.table(coefs, keep.rownames = TRUE)
coefs[,significant:=ifelse((0 < `97.5%`)&(0>`2.5%`), 0, 1)]
vars_keep = (coefs[significant==1,rn])
vars_keep_bool = coefs$significant==1
vars_names = colnames(First_group$x)[vars_keep_bool]
vars_names
```

```
## [1] "signYear"       "latestContract" "WADWS"
```

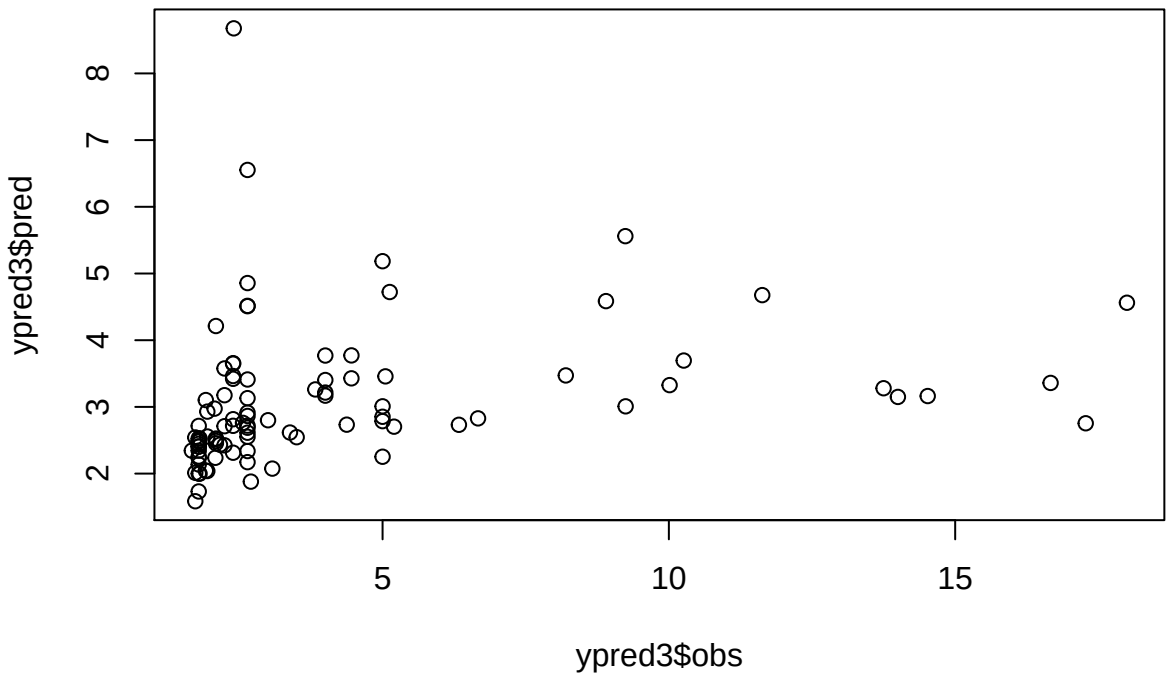
```
# get predictionss
ypred3 = calculate(mup, values=draws, trace_batch_size = 250, nsim=100)
ypred3 = colMeans(ypred3$mup)
ypred3 = predict(y_norm, ypred3, inverse=TRUE)
ypred3 = data.frame(pred=ypred3, obs=third_group$yTest)
caret::postResample(pred=ypred3, obs=third_group$yTest)
```

```
##       RMSE   Rsquared        MAE
##         NA 0.07167324         NA
```

```
# make residuals
ypred3$residuals = (ypred3$obs-ypred3$pred)
#ypred2 = ypred[ypred$obs<10,]
plot(ypred3$obs, ypred3$residuals)
```




```
plot(ypred3$obs, ypred3$pred)
```

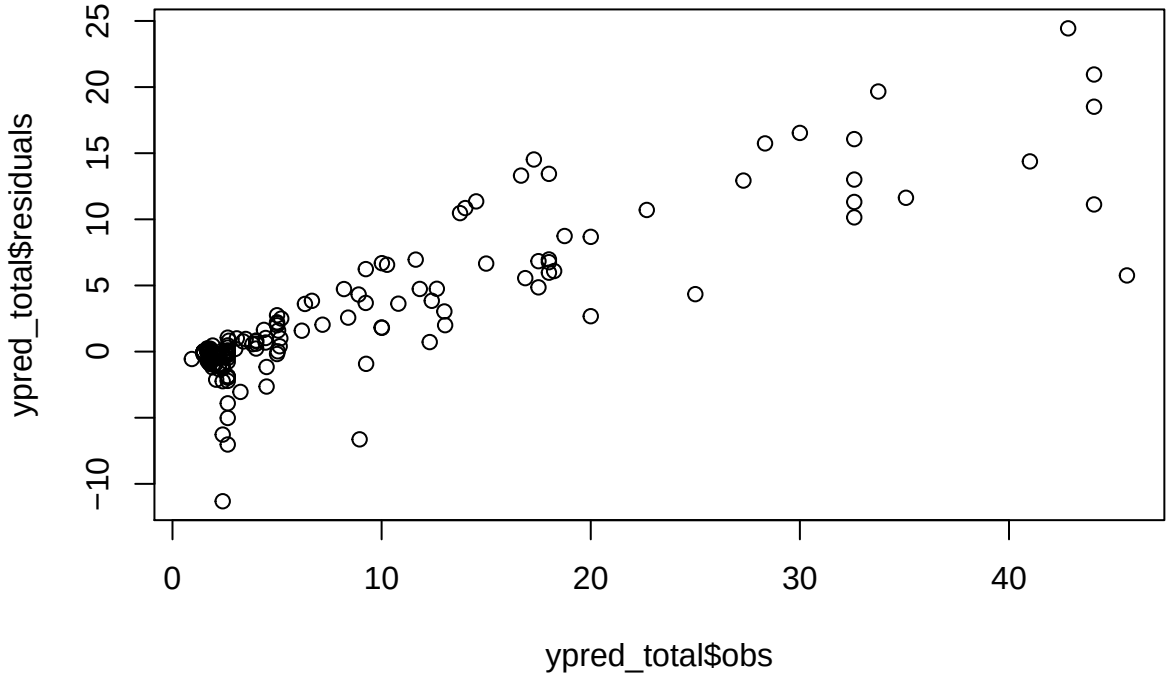



```
#caret::postResample(pred=ypred2$pred, ypred2$obs)
```

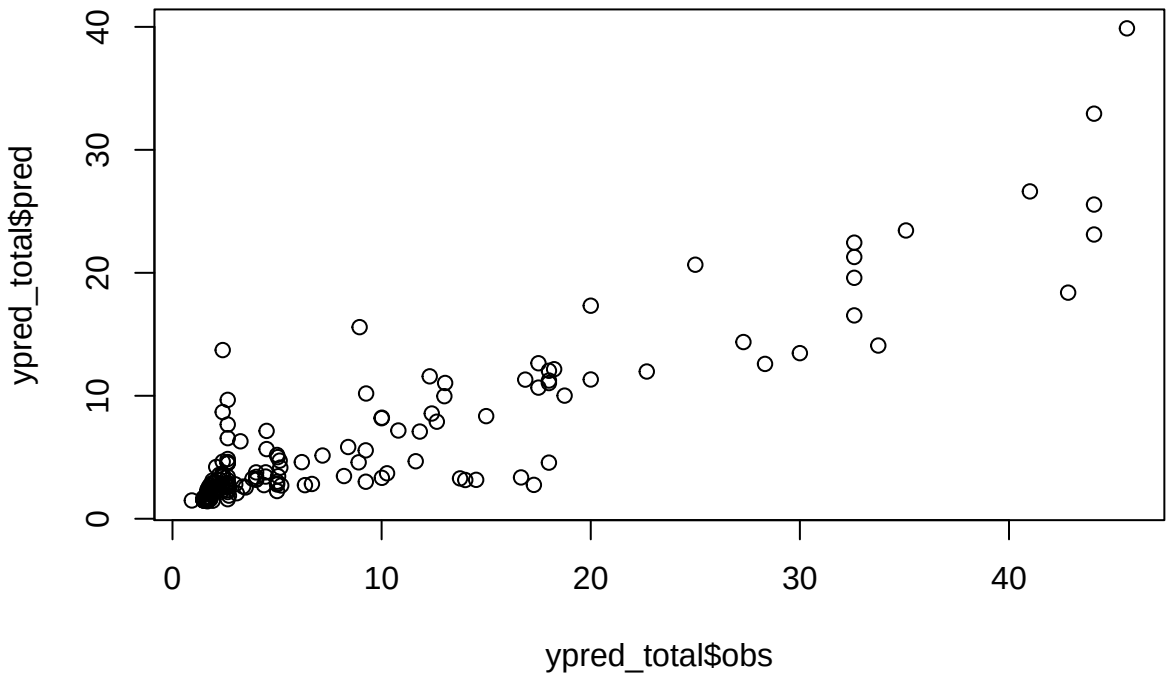
```
ypred_total = rbind(ypred, ypred2, ypred3)
ypred_total = unique(ypred_total)
caret::postResample(ypred_total$pred, ypred_total$obs)
```

```
##      RMSE  Rsquared       MAE
## 5.5213912 0.8247908 2.9908878
```

```
plot(ypred_total$obs, ypred_total$residuals )
```




```
plot(ypred_total$obs, ypred_total$pred)
```


ypred_total$pred
0
10
20
30
40
ypred_total$obs
0
10
20
30
40