Bayesian_HW_Wk_8_Whitson

Paul Whitson

6/2/2020

1 Data

Import data and run a standard (frequentist) regression model for comparison with later models.

As shown below, all inputs are statistically significant except Examination. A Gaussian model appears to fit the data fairly well, there are no unusual outliers in the residuals, as shown in the normal probability plot.

Note that there is some fairly strong correlation among predictors, which may make feature selection more difficult.

```
library(faraway)
data('swiss')

#Note that some predictors are fairly strongly correlated with each other (r in the range of +/- 0.6)
cor(swiss)
```

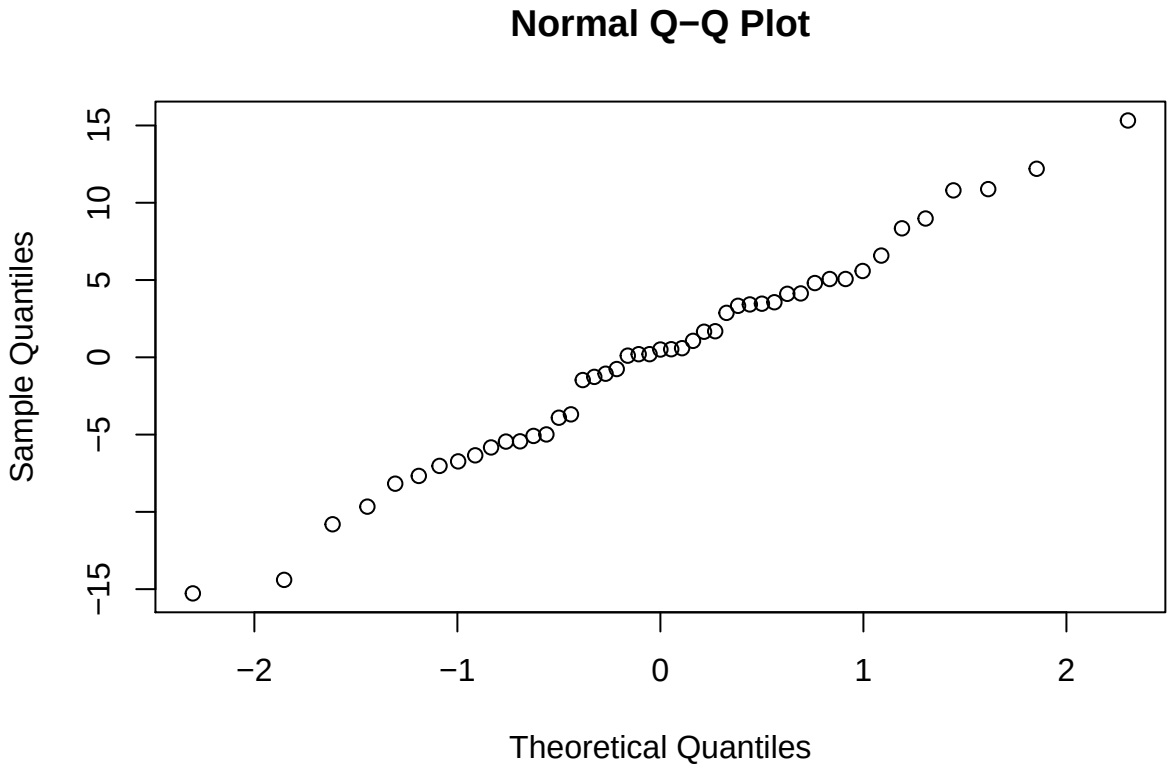
```
##                  Fertility Agriculture Examination   Education   Catholic
## Fertility         1.0000000  0.35307918  -0.6458827 -0.66378886  0.4636847
## Agriculture       0.3530792  1.00000000  -0.6865422 -0.63952252  0.4010951
## Examination      -0.6458827 -0.68654221   1.0000000  0.69841530 -0.5727418
## Education        -0.6637889 -0.63952252   0.6984153  1.00000000 -0.1538589
## Catholic          0.4636847  0.40109505  -0.5727418 -0.15385892  1.0000000
## Infant.Mortality  0.4165560 -0.06085861  -0.1140216 -0.09932185  0.1754959
##                  Infant.Mortality
## Fertility             0.41655603
## Agriculture          -0.06085861
## Examination          -0.11402160
## Education            -0.09932185
## Catholic              0.17549591
## Infant.Mortality      1.00000000
```

```
#For comparison, fit standard (non-Bayesian) linear model:
linear_model <- lm(Fertility~., data = swiss)
summary(linear_model)
```

```
##
## Call:
## lm(formula = Fertility ~ ., data = swiss)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2743  -5.2617   0.5032   4.1198  15.3213
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      66.91518   10.70604   6.250 1.91e-07 ***
## Agriculture      -0.17211    0.07030  -2.448  0.01873 *
```

```
## Examination      -0.25801    0.25388  -1.016  0.31546
## Education        -0.87094    0.18303  -4.758 2.43e-05 ***
## Catholic          0.10412    0.03526   2.953  0.00519 **
## Infant.Mortality  1.07705    0.38172   2.822  0.00734 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.165 on 41 degrees of freedom
## Multiple R-squared:  0.7067, Adjusted R-squared:  0.671
## F-statistic: 19.76 on 5 and 41 DF,  p-value: 5.594e-10
```

```
qqnorm(y=linear_model$residuals)
```



2.1: Fit robust Bayesian Regression Model Without Shrinkage

To fit a robust model, the residuals (y) will be modeled with a t-distribution, which allows for fatter tails (i.e., outliers).

```
#Prepare data
y <- as.vector(swiss$Fertility) #vector of outputs
x <- as.matrix(swiss[,2:6]) #matrix of predictors
Nx <- ncol(x) #columns in x
Ntotal <- nrow(x) #number of data points (rows)
dataList <- list(Ntotal=Ntotal, Nx=Nx, y=y, x=x)
```

```
modelString<-"
data {
    int<lower=1> Ntotal;
    int<lower=1> Nx;
    vector[Ntotal] y;
    matrix[Ntotal, Nx] x;
}
transformed data {
    real meanY;
    real sdY;
    vector[Ntotal] zy; // normalized
    vector[Nx] meanX;
    vector[Nx] sdX;
    matrix[Ntotal, Nx] zx; // normalized

    meanY = mean(y);
    sdY = sd(y);
    zy = (y - meanY) / sdY;
    for ( j in 1:Nx ) {
        meanX[j] = mean(x[,j]);
        sdX[j] = sd(x[,j]);
        for ( i in 1:Ntotal ) {
            zx[i,j] = ( x[i,j] - meanX[j] ) / sdX[j];
        }
    }
}
parameters {
    real zbeta0;
    vector[Nx] zbeta;
    real<lower=0> nu;
    real<lower=0> zsigma;
}
transformed parameters {
    vector[Ntotal] zy_hat;
    zy_hat = zbeta0 + zx * zbeta;
}
model {
    zbeta0 ~ normal(0, 2);
    zbeta  ~ normal(0, 2);
    nu ~ exponential(1/30.0);
    zsigma ~ uniform(1.0E-5 , 1.0E+1);
    zy ~ student_t(1+nu, zy_hat, zsigma);
}
generated quantities {
    real beta0;
    vector[Nx] beta;
    real sigma;

    beta0 = zbeta0*sdY  + meanY - sdY * sum( zbeta .* meanX ./ sdX );
    beta = sdY * ( zbeta ./ sdX );
    sigma = zsigma * sdY;
} "
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
#create Stan model object:
RobustRegressionNoShrinkage <- stan_model(model_code = modelString)
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Users/Paul/Library/R/3.6/
## In file included from <built-in>:1:
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:88:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknow
## namespace Eigen {
## ^
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expe
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file no
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

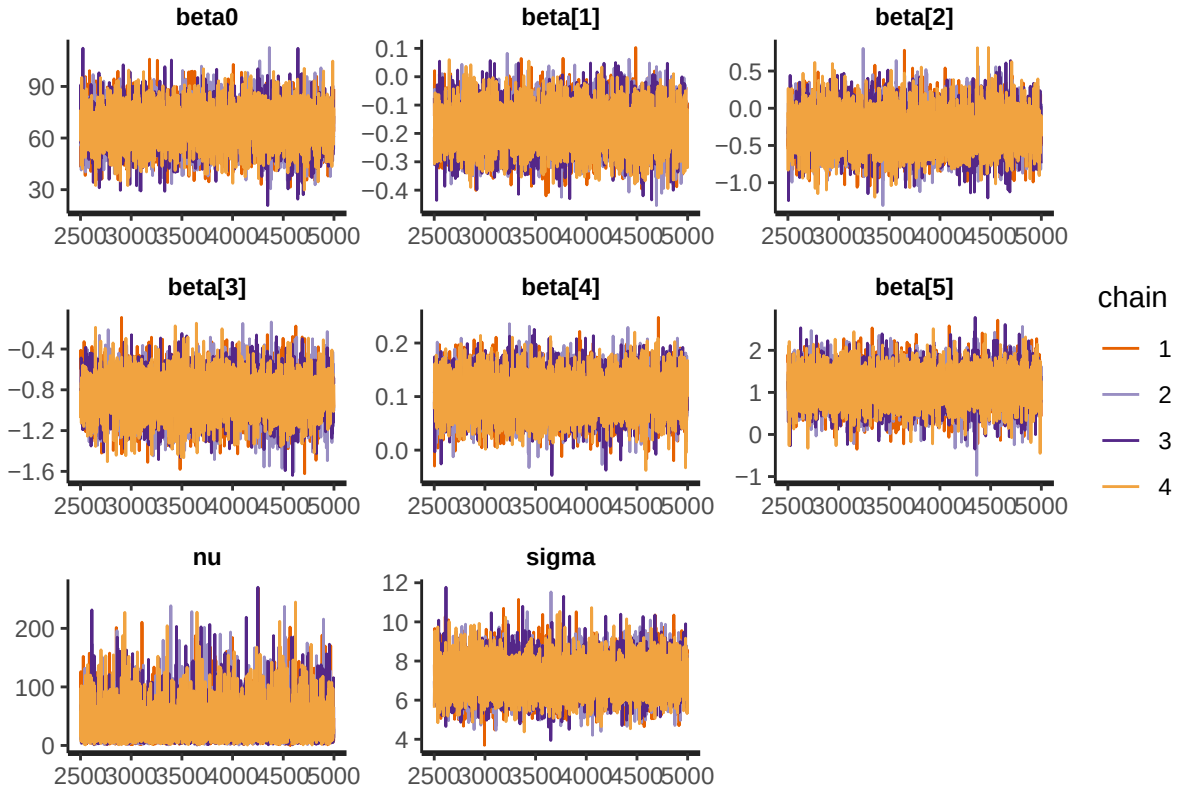
```
#Perform MCMC sampling:
FitNoShrinkage<-sampling(RobustRegressionNoShrinkage,
               data=dataList,
               pars=c('beta0', 'beta', 'nu', 'sigma'),
               iter=5000, chains = 4, cores = 4)
```

Diagnostics were perfomed on MCMC sampling. The chains appear to overlap well, and there is minimal autocorrelation for most parameters. Nu appears to have some negative autocorrelation, and a few parameters have minor autocorrelation for the first few lags.

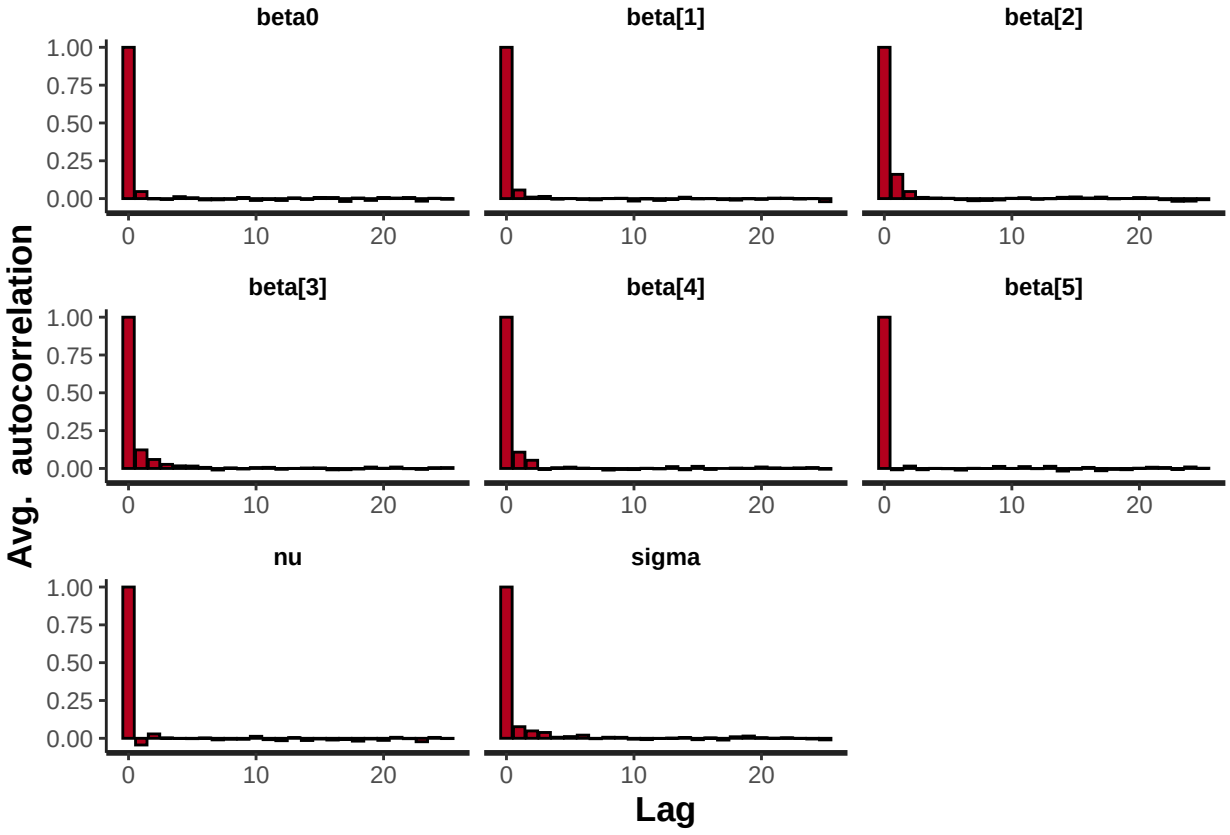
Per the histograms, betas appear normally distributed. Nu is skewed, and sigma is slightly skewed, as is expected.

The pairs plot indicates that estimates for some parameters are correlated with each other, particularly Beta0 (intercept) with beta[1] (Agriculture) and beta[5] (Infant.Mortality). Furthermore, beta[2], beta[3] and beta[4] are correlated with each other. This makes sense, as the inputs associated with these parameters had some correlation with each other.

```
stan_trace(FitNoShrinkage) #chains seem to overlap well
```

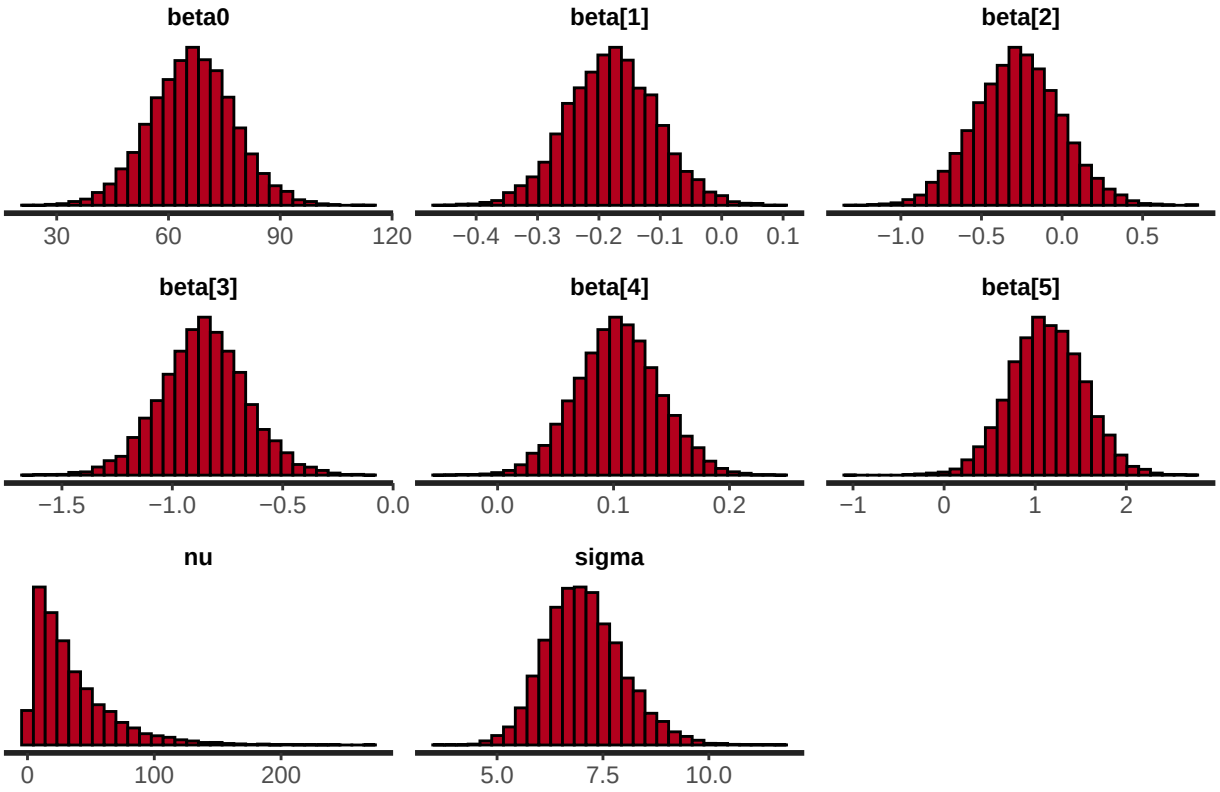



```
stan_ac(FitNoShrinkage) #minimal autocorrelation.  Nu appears to have some negative autocorrelation, an
```

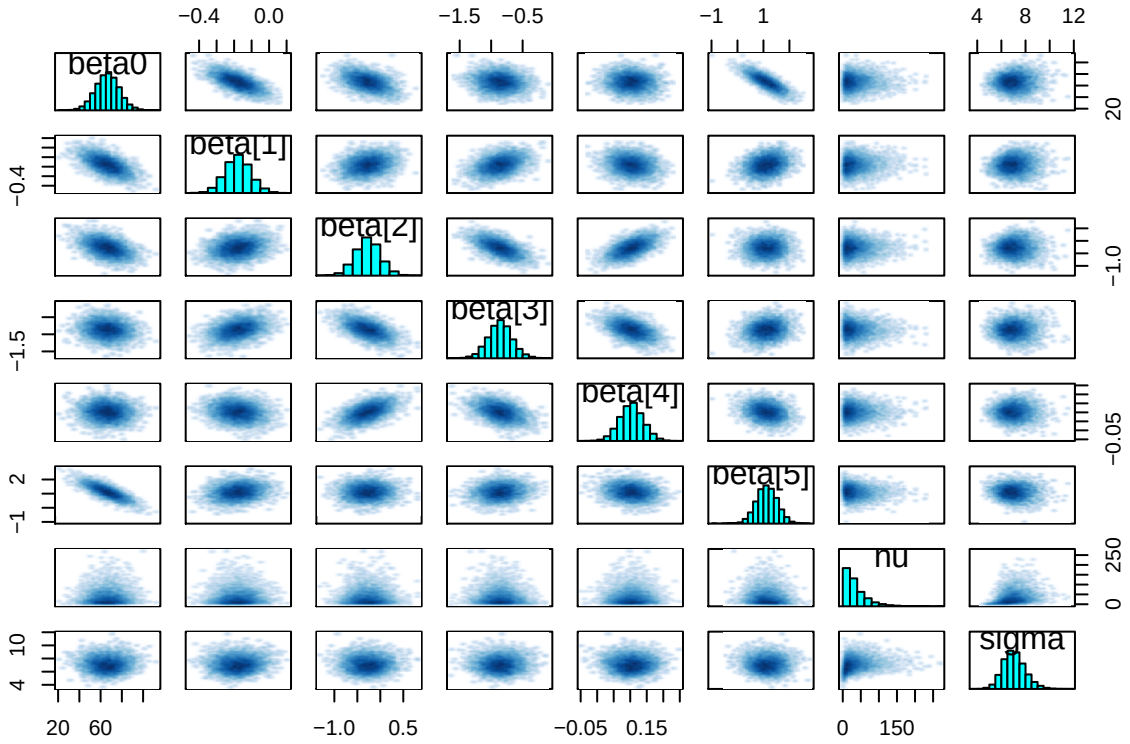



```
stan_hist(FitNoShrinkage)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
pairs(FitNoShrinkage, pars=c("beta0", "beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]", "nu", "sig
```



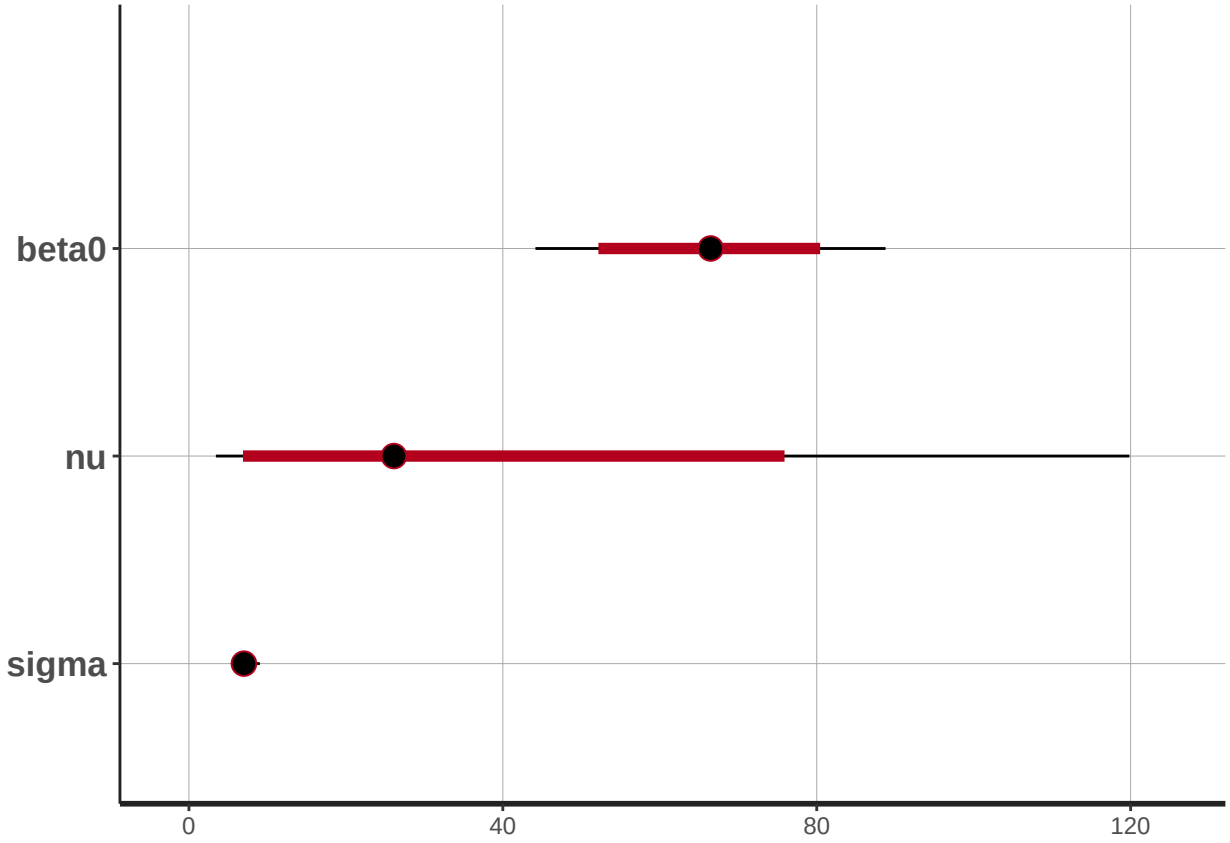
Evaluate HDIs:

Per the HDIs, beta[2] (Examination) is not significant; this is consistent with the frequentist model fitted above.

```
# Evaluate HDIs
plot(FitNoShrinkage, pars=c("beta0", "nu", "sigma")) #all are significantly different from zero
```

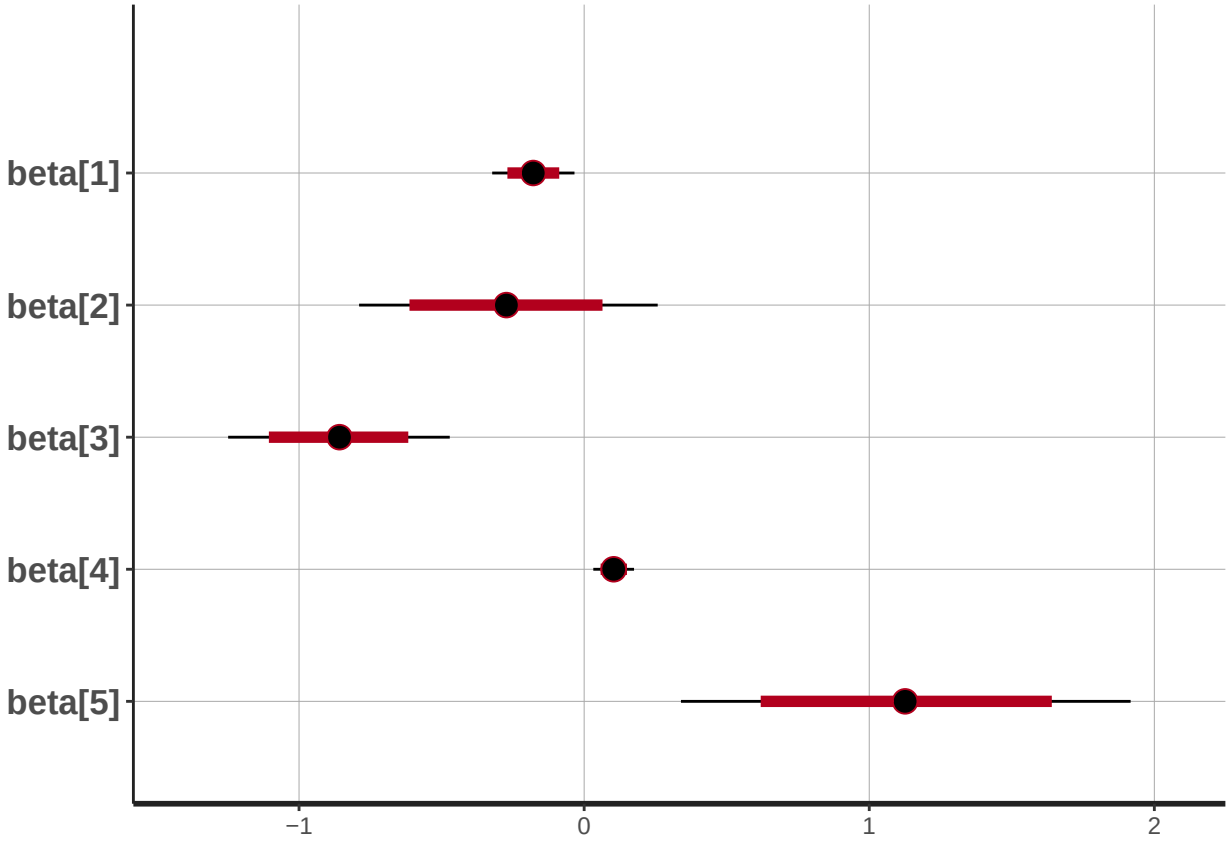
```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
plot(FitNoShrinkage, pars=c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]"))
```

```
## ci_level: 0.8 (80% intervals)
## outer_level: 0.95 (95% intervals)
```

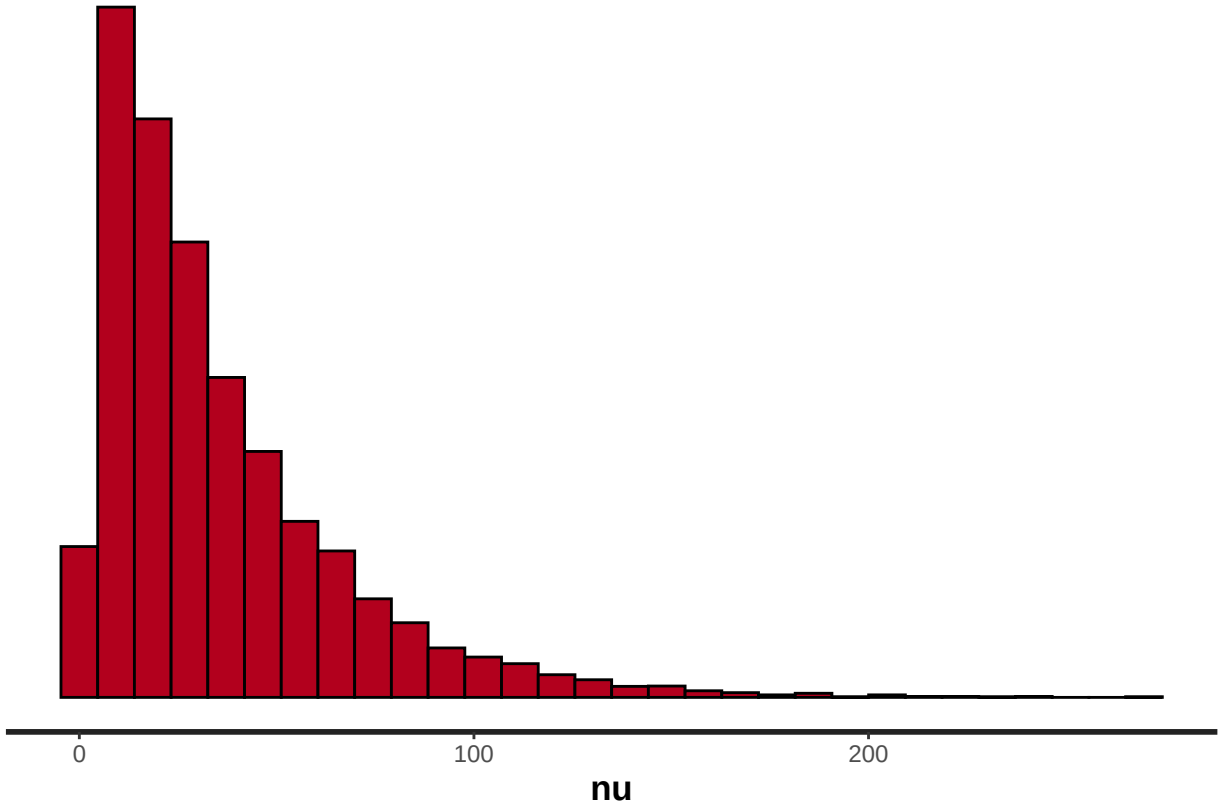


Gaussian Assumption

Per the summary and histogram below, the mean value of nu is 35.2, and the median is 26.6. While there is a fair amount of mass below 30, the mean value above 30 means that we may consider this a fairly Gaussian distribution. This is consistent with the FNP model above, in which the residuals were fit fairly well by a Gaussian distribution and there were no extreme outliers.

```
stan_hist(FitNoShrinkage, pars=c('nu'))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
summary(FitNoShrinkage)$summary[7,1:6]
```

```
##       mean    se_mean         sd       2.5%        25%        50%
## 35.4548266  0.3090785 31.3356331  3.4341777 13.1377001 26.1236585
```

Evaluating Shrinkage:

Per the table below, the parameter estimates are similar to, but not exactly the same as, the non-Bayesian model fit above. For some parameters (Agriculture, Examination, and Infant.Mortality), it appears that some shrinkage was observed. However, the values for Education and Catholic are actually larger / more extreme than in the original model. We did not design this model to promote shrinkage; therefore, it is not surprising that we did not observe much shrinkage effect.

```
remove(summaryTable)
```

```
## Warning in remove(summaryTable): object 'summaryTable' not found
```

```
summaryTable <- cbind("StandardFit" = linear_model$coefficients, "BayesNoShrink" = summary(FitNoShrinkag
summaryTable <- cbind(summaryTable, "Shrink_1" = summaryTable[,"StandardFit"]/summaryTable[,"BayesNoShri
summaryTable
```

```
##                  StandardFit BayesNoShrink  Shrink_1
```

```
## (Intercept)       66.9151817    66.3866362 1.0079616
## Agriculture       -0.1721140    -0.1785524 0.9639408
## Examination       -0.2580082    -0.2723089 0.9474836
## Education         -0.8709401    -0.8587172 1.0142338
## Catholic           0.1041153     0.1036869 1.0041317
## Infant.Mortality   1.0770481     1.1289205 0.9540513
```

2.2 Fitting Model with Shrinkage

In order to encourage shrinkage, we will fit the slope parameters with a t-distribution rather than a Gaussian distribution. By using a more fat-tailed distribution, this will allow some parameters to remain large while shrinking the rest toward zero. For the sigma parameter of this t-distribution we will use a gamma prior distribution that has a lot of weight near zero; this will encourage the sigma values to be closer to zero, which in turn will encourage at least some of the beta values to be closer to zero.

```
modelString2<-"
data {
    int<lower=1> Ntotal;
    int<lower=1> Nx;
    vector[Ntotal] y;
    matrix[Ntotal, Nx] x;
}
transformed data {
    real meanY;
    real sdY;
    vector[Ntotal] zy; // normalized
    vector[Nx] meanX;
    vector[Nx] sdX;
    matrix[Ntotal, Nx] zx; // normalized

    meanY = mean(y);
    sdY = sd(y);
    zy = (y - meanY) / sdY;
    for ( j in 1:Nx ) {
        meanX[j] = mean(x[,j]);
        sdX[j] = sd(x[,j]);
        for ( i in 1:Ntotal ) {
            zx[i,j] = ( x[i,j] - meanX[j] ) / sdX[j];
        }
    }
}
parameters {
    real zbeta0;
    real<lower=0> sigmaBeta; //add a sigma parameter for the distribution of coefficients
    vector[Nx] zbeta;
    real<lower=0> nu;
    real<lower=0> zsigma;
}
transformed parameters {
    vector[Ntotal] zy_hat;
    zy_hat = zbeta0 + zx * zbeta;
}
model {
```

```
    zbeta0 ~ normal(0, 2);
    sigmaBeta ~ gamma(2.3, 1.3); //use a gamma distribution for sigma with values concentrated near zero
    zbeta  ~ student_t(1.0/30.0, 0, sigmaBeta); //make zbeta a student t distribution with a small value
    nu ~ exponential(1/30.0);
    zsigma ~ uniform(1.0E-5 , 1.0E+1);
    zy ~ student_t(1+nu, zy_hat, zsigma);
}
generated quantities {
    real beta0;
    vector[Nx] beta;
    real sigma;

    beta0 = zbeta0*sdY  + meanY - sdY * sum( zbeta .* meanX ./ sdX );
    beta = sdY * ( zbeta ./ sdX );
    sigma = zsigma * sdY;
} "

RobustWithShrinkage <- stan_model(model_code = modelString2)
```

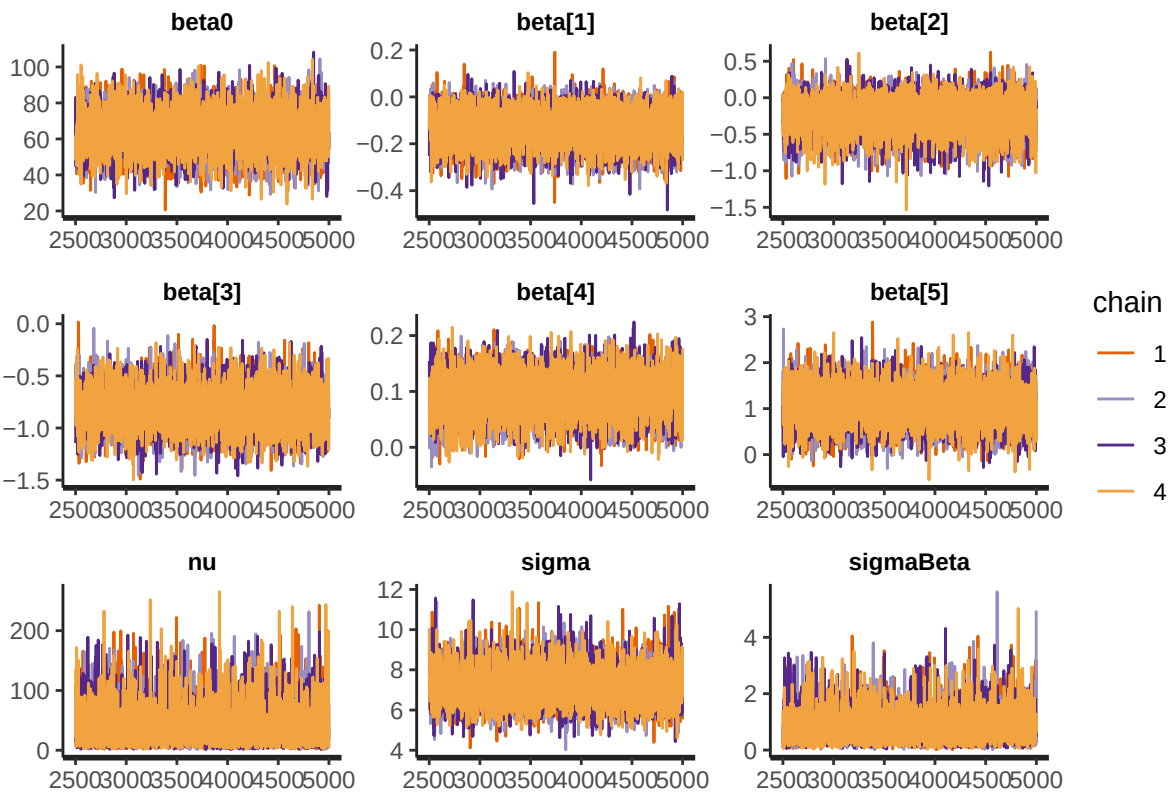
```
## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Users/Paul/Library/R/3.6/l
## In file included from <built-in>:1:
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:88:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknow
## namespace Eigen {
## ^
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expe
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file no
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

```
ShrinkageFit <- sampling(RobustWithShrinkage,
                         data=dataList,
                         pars=c('beta0', 'beta', 'nu', 'sigma', 'sigmaBeta'),
                         iter=5000, chains = 4, cores = 4)
```

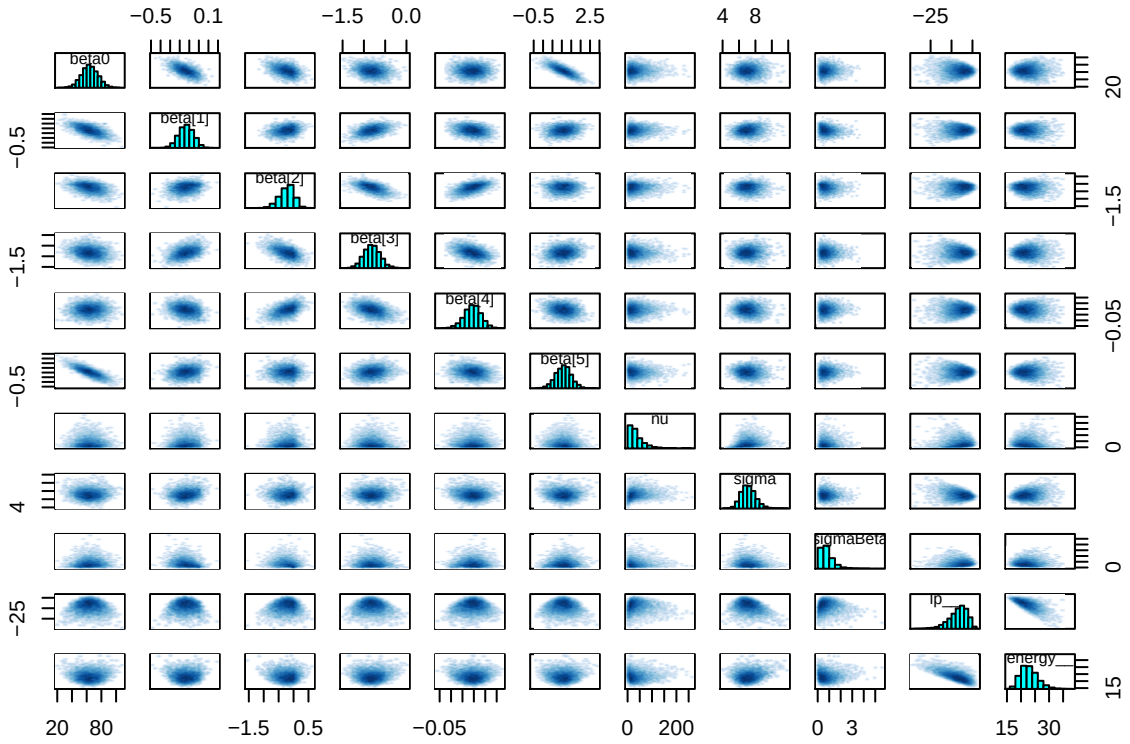
Model Diagnostics:

Note: there was a warning of one divergent transition. However, the traceplots all seem to overlap well, and the pairs plot shows only one divergent transition for all parameters.

```
traceplot(ShrinkageFit) #all seem to overlap pretty well
```




```
pairs(ShrinkageFit)
```

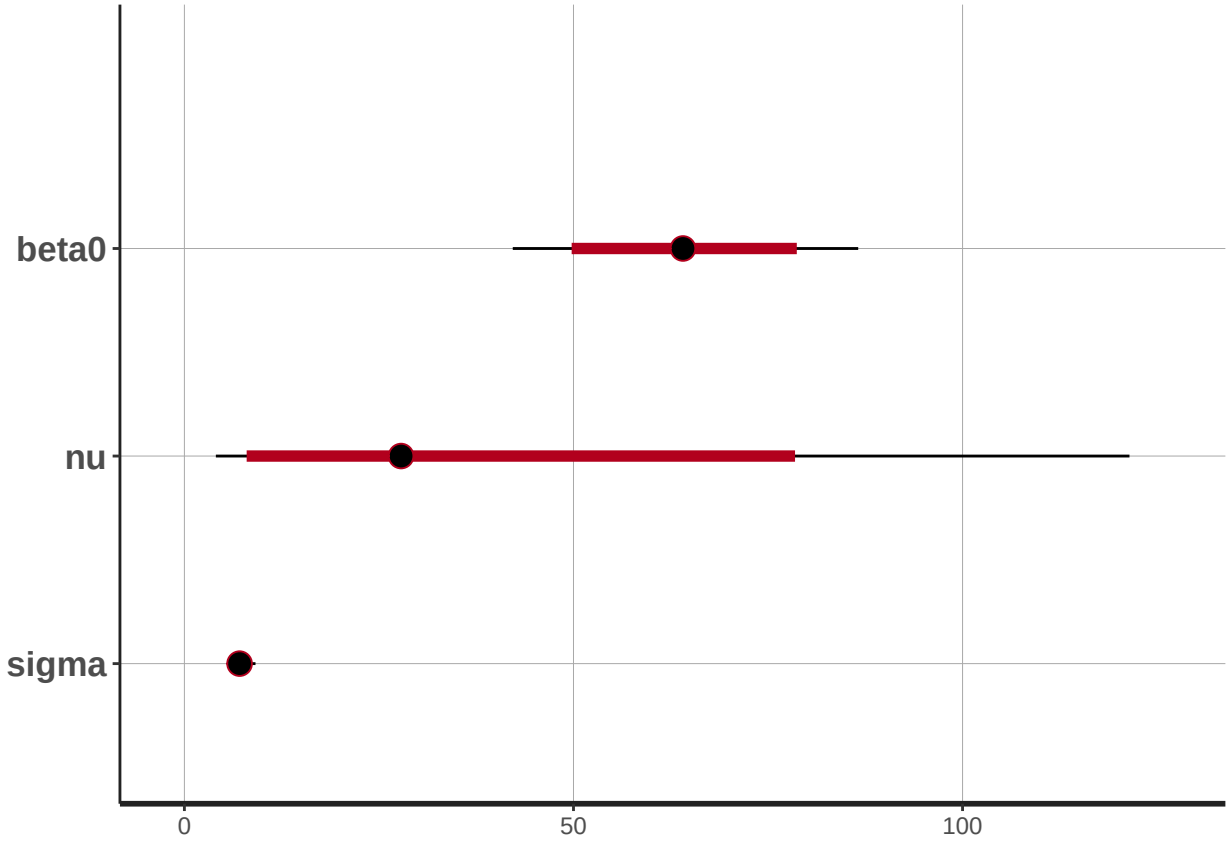


See below for distribution of posterior parameter values. Based on the HDIs in the plot, Beta[2] is again not significant. This time, beta[1] (Agriculture) is also not significant, indicating that shrinkage has occurred.

```
plot(ShrinkageFit, pars=c("beta0", "nu", "sigma")) #all are significantly different from zero
```

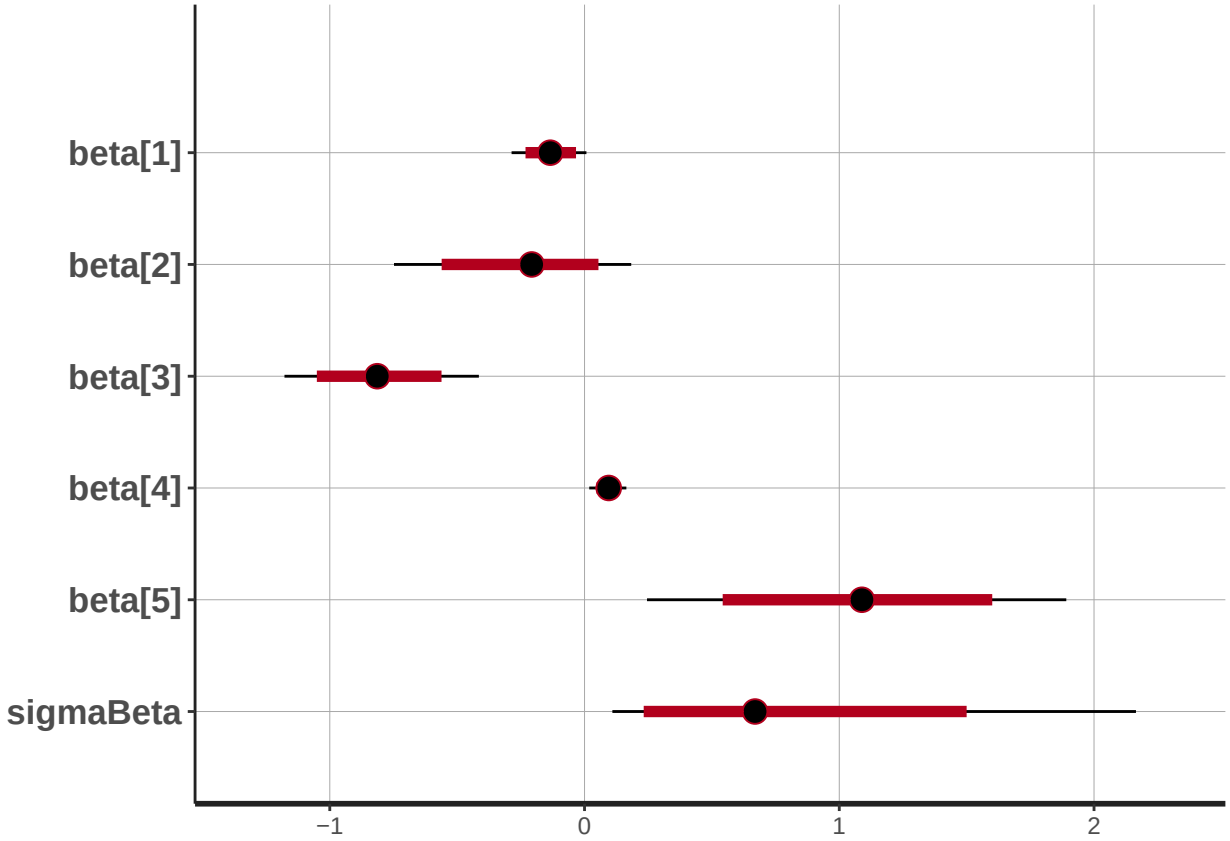
```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
plot(ShrinkageFit, pars=c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]", "sigmaBeta"))
```

```
## ci_level: 0.8 (80% intervals)
## outer_level: 0.95 (95% intervals)
```



The coefficient values of the first and second models were compared. Per the table below, most coefficients have gotten closer to zero. Infant.Mortality has not experienced shrinkage, which may indicate that it is a more significant predictor than the others. Education experienced the least amount of shrinkage among the other predictors.

```
summaryTable <- cbind(summaryTable, "BayesShrink" = summary(ShrinkageFit)$summary[1:6,1])
summaryTable <- cbind(summaryTable, "Shrink_2" = summaryTable[,4]/summaryTable[,1])
summaryTable
```

```
##                 StandardFit BayesNoShrink  Shrink_1 BayesShrink  Shrink_2
## (Intercept)       66.9151817    66.3866362 1.0079616  64.2140538 0.9596336
## Agriculture       -0.1721140    -0.1785524 0.9639408  -0.1340700 0.7789607
## Examination       -0.2580082    -0.2723089 0.9474836  -0.2295471 0.8896889
## Education         -0.8709401    -0.8587172 1.0142338  -0.8101690 0.9302236
## Catholic           0.1041153     0.1036869 1.0041317   0.0941709 0.9044864
## Infant.Mortality   1.0770481     1.1289205 0.9540513   1.0795764 1.0023474
```

Part 2.3 Model Selection:

Create model in JAGS to allow for discrete delta functions. Thinning parameter of 50 was selected, as a high degree of autocorrelation was observed in early runs of the model.

```
suppressWarnings(library(runjags))
```

```
##
```

```
## Attaching package: 'runjags'

## The following object is masked from 'package:rstan':
##
##     extract
```

```
modelString3 = "
    # Standardize the data:
    data {
        ym <- mean(y)
        ysd <- sd(y)
        for ( i in 1:Ntotal ) {
            zy[i] <- ( y[i] - ym ) / ysd
        }
        for ( j in 1:Nx ) {
            xm[j]  <- mean(x[,j])
            xsd[j] <-   sd(x[,j])
            for ( i in 1:Ntotal ) {
                zx[i,j] <- ( x[i,j] - xm[j] ) / xsd[j]
            }
        }
    }
    # Specify the model for standardized data:
    model {
        for ( i in 1:Ntotal ) {
            zy[i] ~ dt( zbeta0 + sum( delta[1:Nx] * zbeta[1:Nx] * zx[i,1:Nx] ) ,  1/zsigma^2 , nu ) #y
        }
        # Priors vague on standardized scale:
        zbeta0 ~ dnorm( 0 , 2) #same parameters as above
        for ( j in 1:Nx ) {
            zbeta[j] ~ dt( 0 , 1/sigmaBeta^2 , 1 )
            delta[j] ~ dbern( 0.5 ) #deltas are bernoulli distributed
        }
        zsigma ~ dunif( 1.0E-5 , 1.0E+1 )
        sigmaBeta ~ dgamma(2.3, 1.3) # used same gamma parameters as above models
        nu ~ dexp(1/30.0)

        # Transform to original scale:
        beta[1:Nx] <- ( delta[1:Nx] * zbeta[1:Nx] / xsd[1:Nx] )*ysd
        beta0 <- zbeta0*ysd  + ym - sum( delta[1:Nx] * zbeta[1:Nx] * xm[1:Nx] / xsd[1:Nx] )*ysd
        sigma <- zsigma*ysd
    }
"

parameters <- c("beta0",  "beta",  "sigma", "delta", "sigmaBeta", "zbeta0", "zbeta", "zsigma", "nu" )
adaptSteps <- 500
burnInSteps <- 1000
numSavedSteps <- 5000
thinSteps <- 50 #set this high because initially there was a lot of autocorrelation when using just 5
nChains <- 4
runjagsMethod <- "parallel"

# run JAGS
JagsDeltaModelOutput <- run.jags(method=runjagsMethod,
```

```
                        model=modelString3,
                        monitor=parameters,
                        data=dataList,
                        n.chains=nChains,
                        adapt=adaptSteps,
                        burnin=burnInSteps,
                        sample=ceiling(numSavedSteps/nChains),
                        thin=thinSteps,
                        summarise=FALSE,
                        plots=FALSE)
```

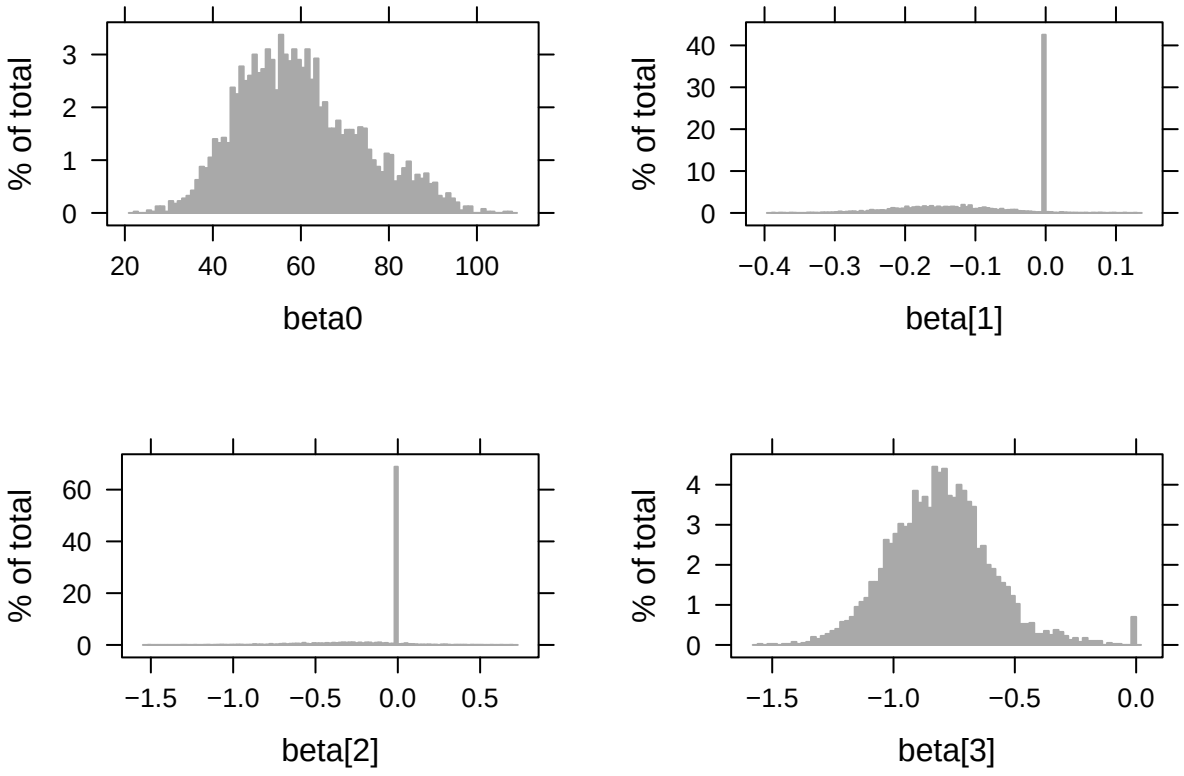
```
## Warning: No initial values were provided - JAGS will use the same initial
## values for all chains

## Warning: You attempted to start parallel chains without setting different
## PRNG for each chain, which is not recommended. Different .RNG.name values
## have been added to each set of initial values.

## Calling 4 simulations using the parallel method...
## Following the progress of chain 1 (the program will wait for all
## chains to finish before continuing):
## Welcome to JAGS 4.3.0 on Tue Jun  2 16:47:44 2020
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 14
##    Total graph size: 809
## . Reading parameter file inits1.txt
## . Initializing model
## . Adapting 500
## -------------------------------------------------| 500
## +++++++++++++++++++++++++++++++++++++++++++++++++ 100%
## Adaptation successful
## . Updating 1000
## -------------------------------------------------| 1000
## ************************************************** 100%
## . . . . . . . . . . Updating 62500
## -------------------------------------------------| 62500
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
```

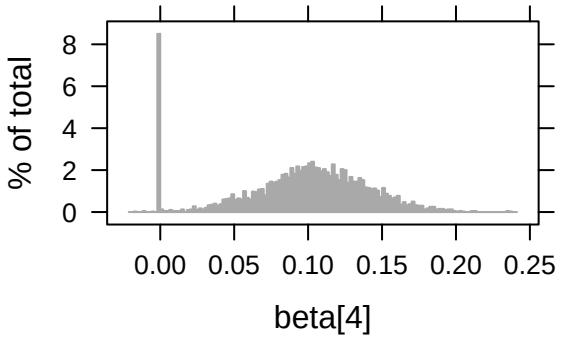
```
## All chains have finished
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Finished running the simulation
```

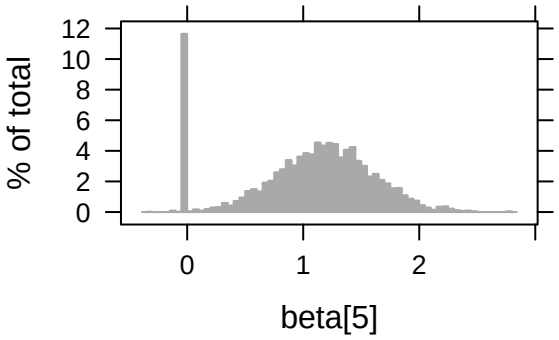
Per the diagnostic plots (not shown for the sake of brevity), the model appears to have converged well. Some of the beta parameters still have some autocorrelation, but it is not extreme. Note that the histograms for the beta parameters include spikes at zero, indicating the times when the associated delta value was zero.

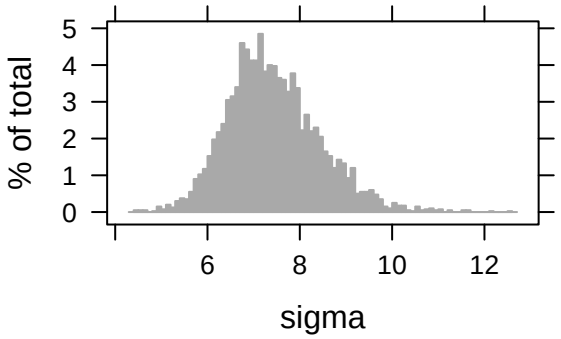
```
plot(JagsDeltaModelOutput,
     plot.type = c("histogram")
     # , vars = 'delta'
)
```

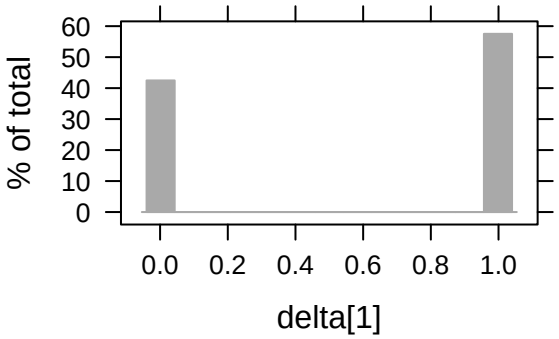
```
## Generating summary statistics and plots (these will NOT be saved
## for reuse)...
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 21 variables....
```

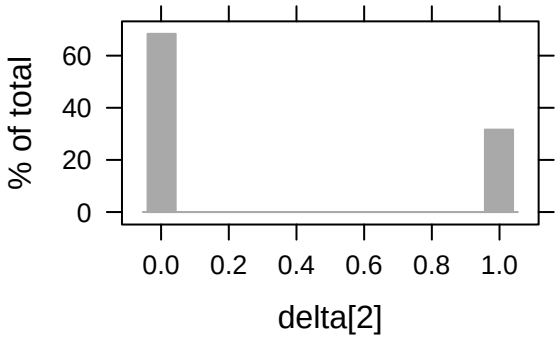


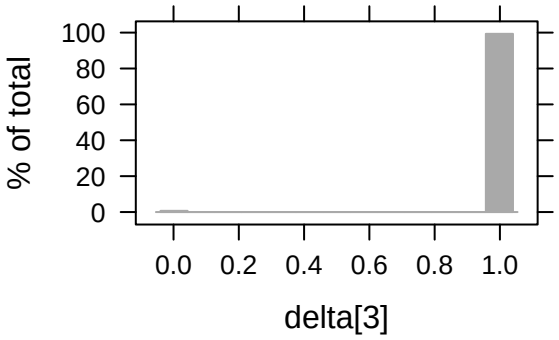

% of total
0
2
4
6
8
0.00
0.05
0.10
0.15
0.20
0.25
beta[4]

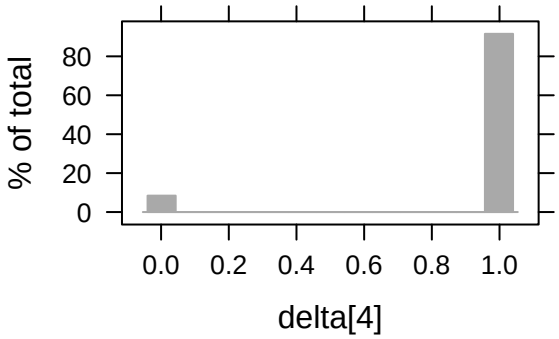

% of total
0
2
4
6
8
10
12
0
1
2
beta[5]

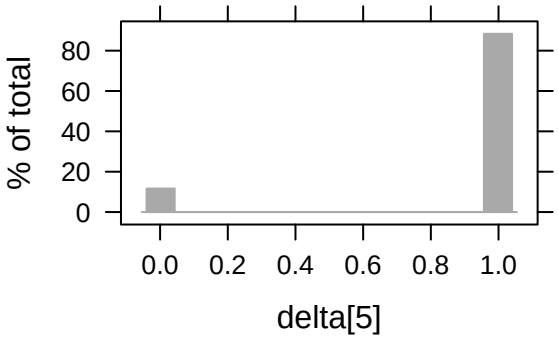

% of total
0
1
2
3
4
5
6
8
10
12
sigma

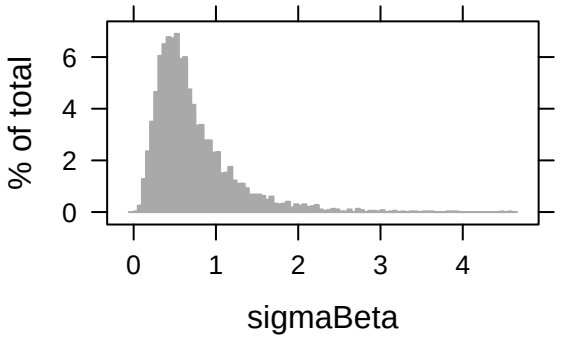

% of total
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0
delta[1]

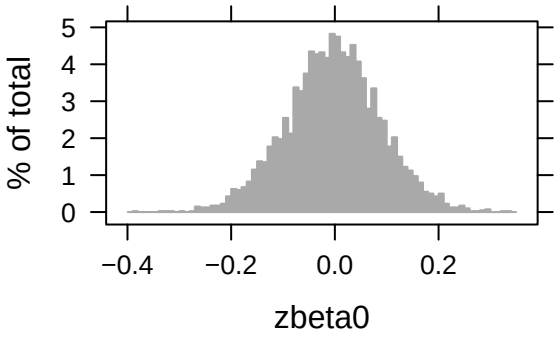

% of total
0
20
40
60
0.0
0.2
0.4
0.6
0.8
1.0
delta[2]

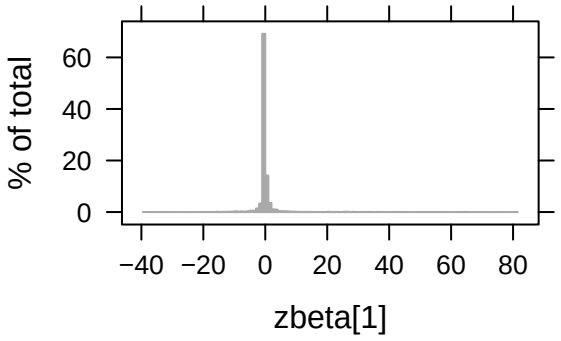

% of total
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
delta[3]

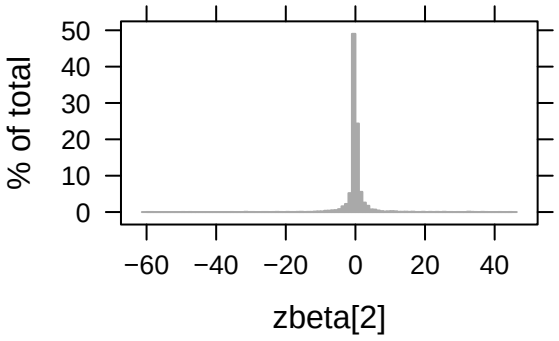

% of total
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
delta[4]

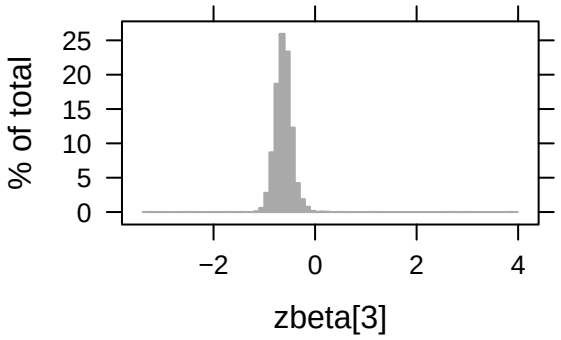

% of total
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
delta[5]

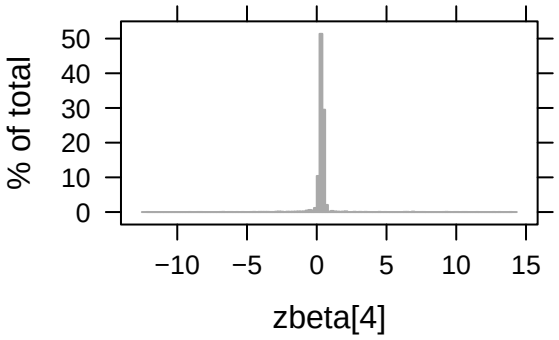

% of total
0
2
4
6
0
1
2
3
4
sigmaBeta

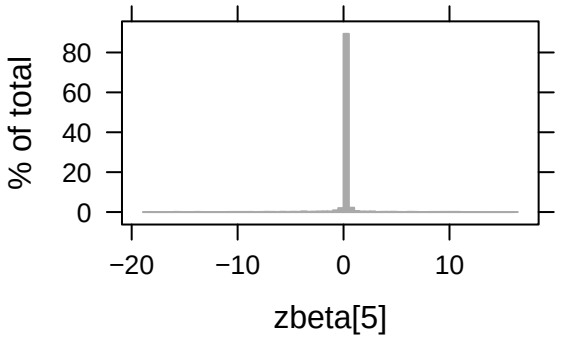

% of total
0
1
2
3
4
5
−0.4
−0.2
0.0
0.2
zbeta0

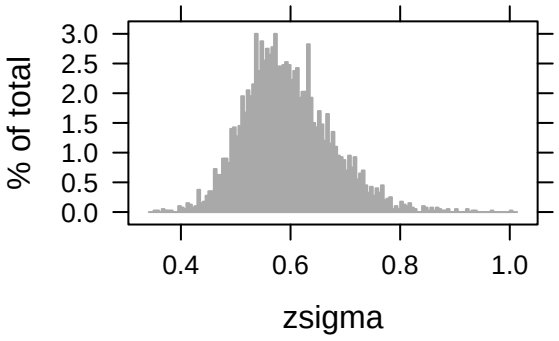

% of total
0
20
40
60
−40
−20
0
20
40
60
80
zbeta[1]

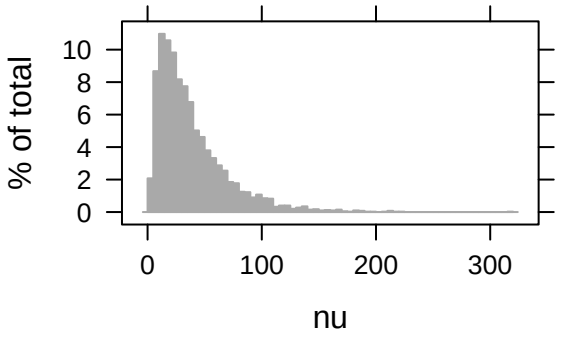

% of total
0
10
20
30
40
50
−60
−40
−20
0
20
40
zbeta[2]


% of total
25
20
15
10
5
0
−2
0
2
4
zbeta[3]


% of total
50
40
30
20
10
0
−10
−5
0
5
10
15
zbeta[4]


% of total
80
60
40
20
0
−20
−10
0
10
zbeta[5]


% of total
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.4
0.6
0.8
1.0
zsigma



Evaluating Model Importance:

The mean of each delta parameter is a measure of the proportion of times that that parameter was "switched on" in the MCMC simulation, and hence is a measure of model importance. Per the summary below, the parameters in order of decreasing importance are * 3 Education * 5 Infant.Mortality * 4 Catholic * 1 Agriculture * 2 Examination

This is consistent with our previous conclusions; Examination was least significant and was removed even without shrinkage, while Agriculture was removed after shrinkage was used.

```
summary(JagsDeltaModelOutput)
```

```
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 21 variables....

##             Lower95       Median     Upper95         Mean           SD Mode
## beta0     36.061400 58.037300000  89.3866000 59.670454240 13.87763879   NA
## beta[1]   -0.264835 -0.061086850   0.0000000 -0.083180113  0.09167354   NA
## beta[2]   -0.709963  0.000000000   0.0930853 -0.098828756  0.22121877   NA
## beta[3]   -1.247920 -0.809659000  -0.4022360 -0.805253312  0.21331383   NA
## beta[4]    0.000000  0.101328500   0.1618280  0.096337066  0.04477702   NA
## beta[5]    0.000000  1.135525000   1.8605100  1.061537478  0.54286175   NA
## sigma      5.688850  7.320400000   9.5264600  7.424462088  0.99336067   NA
## delta[1]   0.000000  1.000000000   1.0000000  0.573000000  0.49469177    1
```

```
## delta[2]   0.000000  0.000000000   1.0000000  0.315600000  0.46480087    0
## delta[3]   1.000000  1.000000000   1.0000000  0.992800000  0.08455525    1
## delta[4]   0.000000  1.000000000   1.0000000  0.916400000  0.27681468    1
## delta[5]   0.000000  1.000000000   1.0000000  0.889200000  0.31391570    1
## sigmaBeta  0.111530  0.592265500   1.6812300  0.715875447  0.47613728   NA
## zbeta0    -0.198561 -0.001749335   0.1617550 -0.001825265  0.09097647   NA
## zbeta[1]  -4.941190 -0.222839000   4.5195200  0.338346331  5.24461920   NA
## zbeta[2]  -5.770260 -0.125920000   6.7147500  0.123926512  4.68285275   NA
## zbeta[3]  -0.965481 -0.623916000  -0.3216410 -0.618715569  0.20543053   NA
## zbeta[4]  -0.302637  0.345842000   0.7362090  0.345446725  0.83856166   NA
## zbeta[5]  -0.922143  0.274343000   1.2351600  0.203797779  1.36701754   NA
## zsigma     0.455410  0.586021500   0.7626240  0.594351749  0.07952166   NA
## nu         2.796070 29.951100000 100.3030000 38.261026416 30.77492472   NA
##                 MCerr MC%ofSD SSeff       AC.500      psrf
## beta0     0.2444756127     1.8  3222  0.008628945 0.9999620
## beta[1]   0.0014208610     1.5  4163  0.018151729 1.0009980
## beta[2]   0.0037628631     1.7  3456 -0.005819549 1.0008120
## beta[3]   0.0034422464     1.6  3840  0.010614319 1.0003525
## beta[4]   0.0007675302     1.7  3403  0.018771342 1.0013364
## beta[5]   0.0098943939     1.8  3010  0.028552756 1.0000646
## sigma     0.0140482413     1.4  5000  0.014306563 1.0005656
## delta[1]  0.0081095681     1.6  3721  0.028201298 1.0003274
## delta[2]  0.0069912728     1.5  4420 -0.015054779 1.0001752
## delta[3]  0.0015799123     1.9  2864 -0.007118667 1.0288235
## delta[4]  0.0055569808     2.0  2481  0.032527105 1.0035321
## delta[5]  0.0065211443     2.1  2317  0.032256575 1.0017117
## sigmaBeta 0.0074729541     1.6  4060  0.018218606 0.9997619
## zbeta0    0.0012866015     1.4  5000  0.026965604 0.9997955
## zbeta[1]  0.1579184357     3.0  1103  0.127257341 1.2207395
## zbeta[2]  0.0950818221     2.0  2426  0.001859473 1.0644052
## zbeta[3]  0.0030840590     1.5  4437  0.004557019 1.0145488
## zbeta[4]  0.0236508656     2.8  1257  0.028640168 1.0357414
## zbeta[5]  0.0463161905     3.4   871  0.054560175 1.0732252
## zsigma    0.0011246061     1.4  5000  0.014306591 1.0005656
## nu        0.4352231593     1.4  5000 -0.027140153 1.0003407
```

The following models were evaluated for their importance, based on the frequencies that they occurred in the sample.

- The model that occurred most often is Agriculture, Education, Catholic, Infant.Mortality.
- The second-most common model is that containing Education, Catholic & Infant.Mortality.

In addition, a tabulation of ALL permutations of the model was performed. Beyond the two models already mentioned, the next most-common model was that with all predictors.

Conclusions

From the above models, we may conclude that Examination was the least-significant factor, with Agriculture the second least-significant.

Education, Infant.Mortality and Catholic were consistently significant predictors.

```
trajectoriesDelta <- as.matrix(JagsDeltaModelOutput$mcmc[,8:12])
colnames(trajectoriesDelta) <- colnames(swiss)[2:6]

#Calculate proprotion of observed values
Agr_Exam <- sum(apply(trajectoriesDelta,1,function(z) prod(z==c(1,1,0,0,0))))/nrow(trajectoriesDelta)
Edu_Cath_InfMort <- sum(apply(trajectoriesDelta,1,function(z) prod(z==c(0,0,1,1,1))))/nrow(trajectoriesD
Agr_Edu_Cath_InfMort <- sum(apply(trajectoriesDelta,1,function(z) prod(z==c(1,0,1,1,1))))/nrow(trajecto
NullModel <- sum(apply(trajectoriesDelta,1,function(z) prod(z==c(0,0,0,0,0))))/nrow(trajectoriesDelta)

paste("Agriculture plus Exam: ", Agr_Exam)
```

```
## [1] "Agriculture plus Exam:  0"
```

```
paste("Education, Catholic & Infant.Mortality: ", Edu_Cath_InfMort)
```

```
## [1] "Education, Catholic & Infant.Mortality:  0.2664"
```

```
paste("Agriculture, Education, Catholic, Infant.Mortality: ", Agr_Edu_Cath_InfMort)
```

```
## [1] "Agriculture, Education, Catholic, Infant.Mortality:  0.3194"
```

```
paste("Model with No Predictors: ", NullModel)
```

```
## [1] "Model with No Predictors:  0"
```

```
DeltaModels <- vector(mode = "character", length = nrow(trajectoriesDelta))
nRows <- nrow(trajectoriesDelta)
for (i in 1:nRows) {
    DeltaModels[i] = paste(as.numeric(trajectoriesDelta[i,1]),
                           as.numeric(trajectoriesDelta[i,2]),
                           as.numeric(trajectoriesDelta[i,3]),
                           as.numeric(trajectoriesDelta[i,4]),
                           as.numeric(trajectoriesDelta[i,5]),
                           sep = "_")
    }

modelSummary <- table(DeltaModels)
modelSummary <- modelSummary/5000
modelSummary[order(modelSummary, decreasing = TRUE)]
```

```
## DeltaModels
## 1_0_1_1_1 0_0_1_1_1 1_1_1_1_1 0_1_1_1_1 1_0_1_1_0 0_1_1_0_1 0_0_1_1_0
##    0.3194    0.2664    0.1484    0.0740    0.0586    0.0356    0.0222
## 1_1_1_0_1 1_1_1_1_0 0_0_1_0_1 0_1_1_1_0 0_1_0_0_1 0_1_1_0_0 1_0_1_0_1
##    0.0212    0.0202    0.0152    0.0056    0.0042    0.0020    0.0020
## 1_1_1_0_0 1_1_0_0_1 0_1_0_1_1 0_0_1_0_0 1_1_0_1_1 0_1_0_0_0
##    0.0014    0.0012    0.0010    0.0006    0.0006    0.0002
```