# BCLshrink Demonstration

Md Nazir Uddin and Jeremy T Gaskins

11/17/2021

This document demonstrates the code used in the manuscript "Shared Bayesian Variable Shrinkage in Multinomial Logistic Regression" by Md Nazir Uddin and Jeremy T Gaskins.

First, we load the source file and other packages that our code is dependent upon.

```
# source('BCLshrink code source.R')
library(devtools)
source_url("https://raw.githubusercontent.com/jeremygaskins/BCLshrink/main/BCLshrink%20code%20source.R")
library(pgdraw)
library(pscl)
library(coda)
library(GIGrvg)
```

There are five functions in the BCLshrink source code file: BLCshrink.HSSS, BLCshrink.HSUS, BLCshrink.NGSS, BLCshrink.NGUS, BLCshrink.none. The first four fit the various shrinkage models proposed either using shared shrinkage (SS) or unique shrinkage (US). For each choice, the global-local prior used is either horseshoe (HS) or normal-gamma (NG). The BLCshrink.none function runs the "no shrinkage" model that simply uses a mean-zero normal prior.

## Function arguements (in all functions):

y: response vector. Should be coded as numeric with values 1,2,..,k
X: design matrix. Must be numeric. Do not include intercept.
N_it: total number of iterations used. (default: 15,000)
burn.in: number of iterations to discard as burn in. (default: 20% of N_it)
CI.level: confidence interval level. (default: 0.95)
progress: whether to print MCMC progress markers
samples: whether to return the array with all stored MCMC samples (default: false)
MCMC.seed: (default: 209578)

## Additional arguments for the NG model functions:

c0: Normal-Gamma hyperparameter (default: 0.01) c1: Normal-Gamma hyperparameter (default: 0.01) theta: Normal-Gamma tail thickness parameter (default: 0.05)

## Additional argument for the no shrinkage function:

prior.var: this model performs no shrinkage and simply has a prior covariance matrix with elements given by prior.var (default: 100)

## Loading data set

```
load("mnlsims1v1v2_df1.RData")
# File is available in the BCLshrink repository on Github
```

```
y <- mnlogit_data[, 1]
table(y)
```

```
## y
##   1   2   3
## 187 215  98
```

The function will take the largest category as the reference. For this data, that will be K=3. If a different category is desired to be the reference, the user should relabel the y vector.

```
X <- as.matrix(mnlogit_data[, -1])
X[1:10, 1:10]
```

```
##         X1           X2   X3          X4   X5          X6   X7          X8   X9
##  [1,]  0.5 -0.662774484 -0.5  1.21968543  0.5 -1.18543816 -0.5  1.08775507 -0.5
##  [2,] -0.5 -1.839536715  0.5  1.18454788  0.5 -0.76074772 -0.5  0.42548485  0.5
##  [3,] -0.5 -1.278035505  0.5 -0.07778871 -0.5  0.99019404 -0.5 -0.27880101  0.5
##  [4,] -0.5  1.087984100 -0.5 -1.52608493  0.5  0.08383140 -0.5 -0.05333455  0.5
##  [5,]  0.5 -1.666026951 -0.5  0.33724028 -0.5 -0.29848131  0.5  0.97235476  0.5
##  [6,]  0.5  0.005099735  0.5  0.10450275 -0.5  1.04169205 -0.5  0.76543428  0.5
##  [7,] -0.5  0.078523799 -0.5  0.12894318  0.5  0.03485078 -0.5 -0.26850204 -0.5
##  [8,] -0.5  0.029352442 -0.5 -1.46798957  0.5  0.37521894 -0.5 -1.09912069  0.5
##  [9,]  0.5  1.674682070  0.5 -0.41174462 -0.5 -0.24104491 -0.5 -1.92457734  0.5
## [10,] -0.5 -0.414554065 -0.5 -1.16167675  0.5 -1.19616824 -0.5  0.56820926 -0.5
##               X10
##  [1,] -0.21051462
##  [2,]  0.08601221
##  [3,] -0.32606105
##  [4,] -0.18283052
##  [5,]  2.35368508
##  [6,] -1.20552655
##  [7,]  1.61004187
##  [8,]  1.23769368
##  [9,]  0.93634956
## [10,]  1.75960106
```

Note: The intercept should not be included in the X matrix. X should be centered and scaled.

### Fitting Horseshoe shared shrinkage (HS SS) model:

```
fit.HSSS <- BLCshrink.HSSS(y = y, X = X, N_it = 15000, burn.in = 5000,
    samples = TRUE, progress = FALSE)
```

### Printing the point estimates (posterior means) for the intercept and the first 10 predictors:

```
round(fit.HSSS$kappa.hat, 3)[, 1:11]
```

```
##       int    X1     X2     X3     X4    X5     X6    X7     X8     X9    X10
## c1 -0.555 1.656  2.682  0.019 -0.001 0.013  0.022 0.004  0.007 -0.025 -0.009
## c2  0.135 0.809 -1.818 -0.001 -0.002 0.029 -0.025 0.006 -0.003  0.006 -0.004
## c3  0.000 0.000  0.000  0.000  0.000 0.000  0.000 0.000  0.000  0.000  0.000
```

Note: Each row corresponds to the particular category value with category K set to zero as default.

### Printing the credible intervals for the first 10 variables

```
round(fit.HSSS$kappa.LCI, 3)[, 1:11]
```

```
##       int    X1     X2     X3     X4     X5     X6     X7     X8     X9    X10
## c1 -1.070 0.955  2.051 -0.077 -0.091 -0.114 -0.060 -0.125 -0.060 -0.302 -0.124
## c2 -0.303 0.240 -2.321 -0.143 -0.091 -0.071 -0.223 -0.104 -0.092 -0.125 -0.100
## c3     NA    NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
```

```
round(fit.HSSS$kappa.UCI, 3)[, 1:11]
```

```
##       int    X1     X2    X3    X4    X5    X6    X7    X8    X9   X10
## c1 -0.053 2.393  3.378 0.259 0.083 0.246 0.210 0.155 0.115 0.077 0.057
## c2  0.541 1.409 -1.333 0.138 0.081 0.331 0.053 0.159 0.068 0.179 0.071
## c3     NA    NA     NA    NA    NA    NA    NA    NA    NA    NA    NA
```

Note: LCI shows the lower endpoint of the interval and UCI is the upper endpoint. The CIs are not well-defined for the K=3 reference category.

### Determining which variables are significantly associated with class membership

This is determined by whether the CI overlaps with zero and is not defined for the reference category.

```
fit.HSSS$kappa.sig[, 1:11] * 1
```

```
##    int X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
## c1   1  1  1  0  0  0  0  0  0  0   0
## c2   0  1  1  0  0  0  0  0  0  0   0
## c3  NA NA NA NA NA NA NA NA NA NA  NA
```

Note: The 1's indicate that the predictor is significantly associated. Here, we see that X1 and X2 are each associated with a difference between class 1 and 3, and between class 2 and 3. The other variables X3-X10 are not significantly associated with any significant differences.

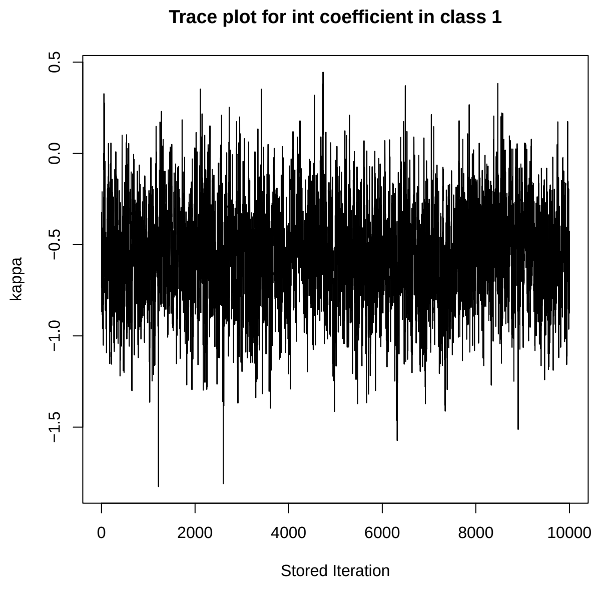
### Check convergence: compute effective sample size

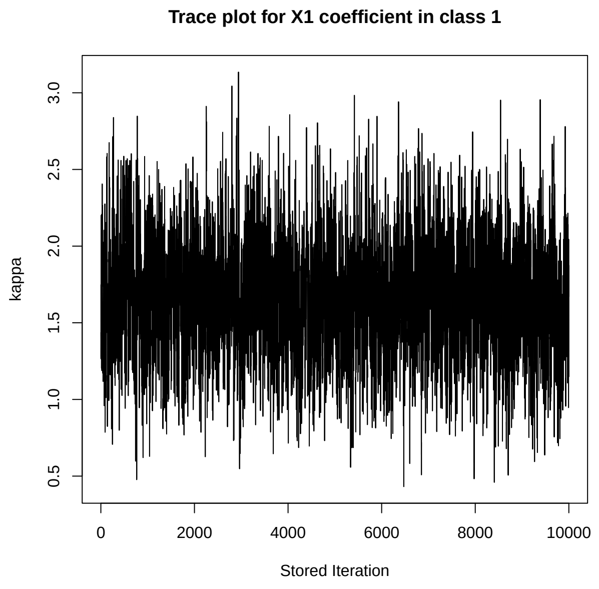
```
round(apply(fit.HSSS$samples, 2:3, coda::effectiveSize), 1)[,
    1:11]
```

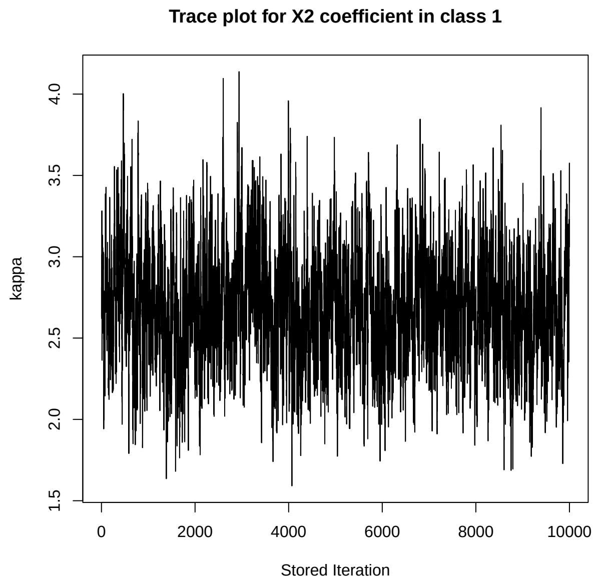
```
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
## [1,] 1187.1 1850.6  713.4 2121.5 6329.4 3195.9 1815.9 6724.2 3331.0 2086.5
## [2,] 1290.5 3149.1 1271.3 5710.9 6445.8 1210.9 1433.4 4875.4 6599.6 5542.6
## [3,]    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
##       [,11]
## [1,] 2948.8
## [2,] 5565.0
## [3,]    0.0
```

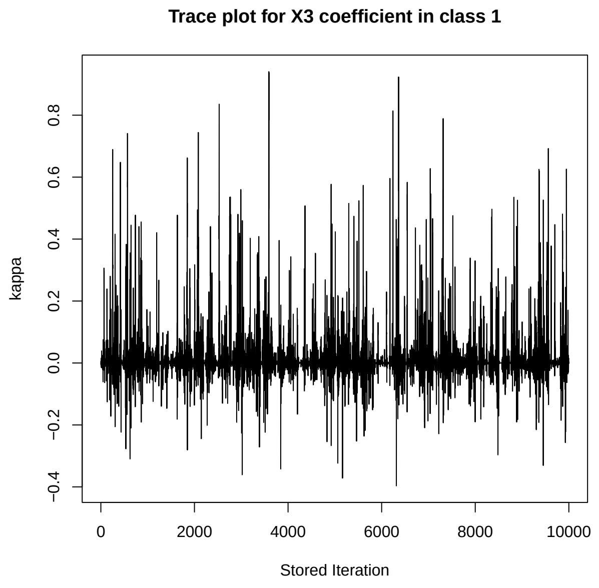
### Trace plots of the coefficients

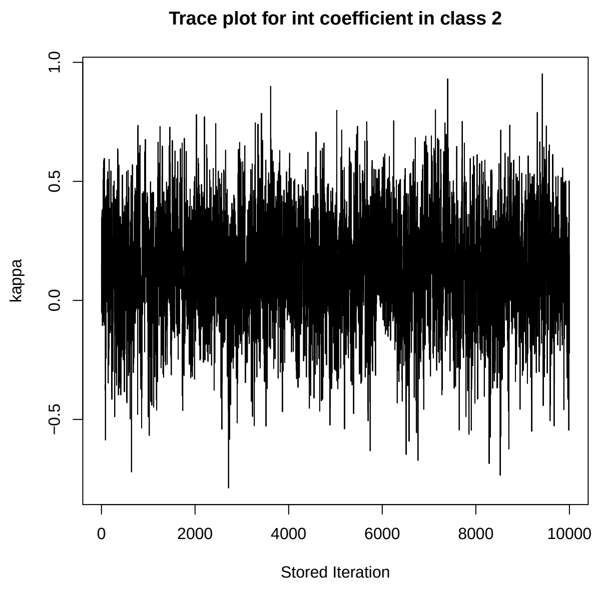
```
for (p in 1:2) {
    for (j in 1:4) {
        plot(fit.HSSS$samples[, p, j], type = "l", xlab = "Stored Iteration",
            main = paste0("Trace plot for ", colnames(fit.HSSS$kappa.hat)[j],
                " coefficient in class ", p), ylab = "kappa")
    }
}
```

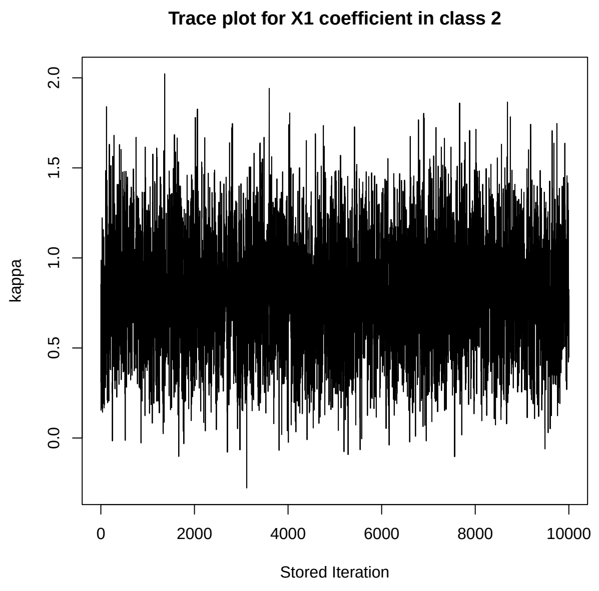

Trace plot for int coefficient in class 1
kappa
Stored Iteration
0.5
0.0
−0.5
−1.0
−1.5
0
2000
4000
6000
8000
10000

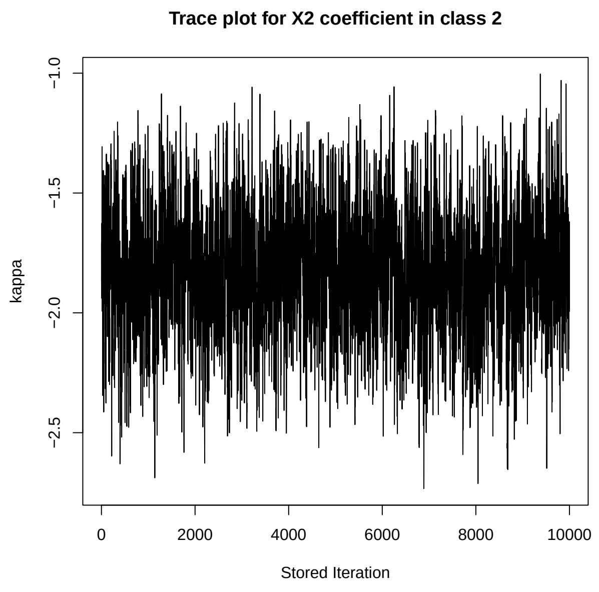

Trace plot for X1 coefficient in class 1
kappa
Stored Iteration
3.0
2.5
2.0
1.5
1.0
0.5
0
2000
4000
6000
8000
10000

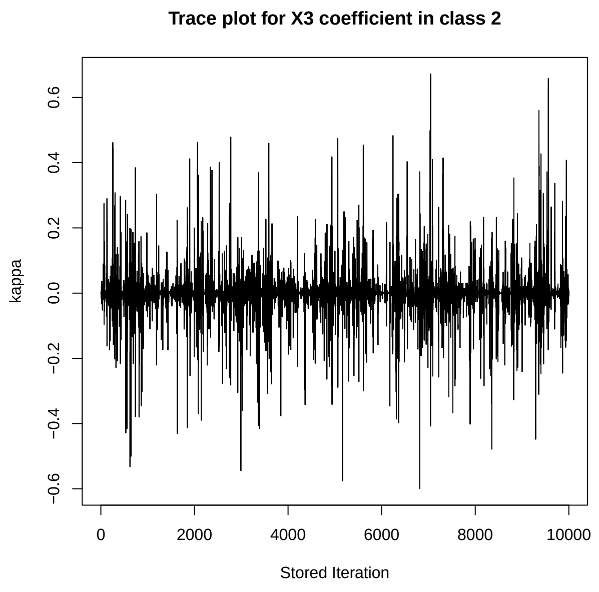

Trace plot for X2 coefficient in class 1
kappa
Stored Iteration
4.0
3.5
3.0
2.5
2.0
1.5
0
2000
4000
6000
8000
10000


Trace plot for X3 coefficient in class 1
kappa
Stored Iteration
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
2000
4000
6000
8000
10000









```
# p is for each class and j for each coefficient in the pth
# class.
```

For simplicity we are only showing the trace plots for the first 4 coefficients in each class. The user should more thoroughly investigate the mixing of all coefficients.

## Other shrinkage models

Similarly, we can run the MCMC for the other shrinkage models. Investigation of the point estimates, confidence intervals, hypothesis testing, and convergence is all done in the exact same way. For simplicity, we ony show the model fitting and the point estimates.

## Normal Gamma shared shrinkage (NG SS) model

```
fit.NGSS <- BLCshrink.NGSS(y = y, X = X, N_it = 15000, burn.in = 5000,
    samples = TRUE, progress = FALSE)
round(fit.NGSS$kappa.hat, 3)[, 1:11]
```

```
##       int    X1     X2     X3     X4    X5     X6    X7     X8     X9    X10
## c1 -0.517 1.590  2.664  0.031 -0.001 0.026  0.024 0.008  0.009 -0.040 -0.008
## c2  0.120 0.766 -1.886 -0.002 -0.001 0.049 -0.028 0.008 -0.003  0.009 -0.005
## c3  0.000 0.000  0.000  0.000  0.000 0.000  0.000 0.000  0.000  0.000  0.000
```

## Horseshoe unique shrinkage (HS US) model

```
fit.HSUS <- BLCshrink.HSUS(y = y, X = X, N_it = 15000, burn.in = 5000,
    samples = TRUE, progress = FALSE)
round(fit.HSUS$kappa.hat, 3)[, 1:11]
```

```
##       int    X1     X2     X3     X4    X5     X6    X7     X8     X9    X10
## c1 -0.608 1.477  2.758  0.029 -0.001 0.009  0.027 0.006  0.011 -0.041 -0.011
## c2  0.166 0.309 -1.811 -0.001 -0.002 0.027 -0.027 0.009 -0.005  0.011 -0.004
## c3  0.000 0.000  0.000  0.000  0.000 0.000  0.000 0.000  0.000  0.000  0.000
```

## Normal Gamma unique shrinkage (NG US) model

```
fit.NGUS <- BLCshrink.NGUS(y = y, X = X, N_it = 15000, burn.in = 5000,
    samples = TRUE, progress = FALSE)
round(fit.NGUS$kappa.hat, 3)[, 1:11]
```

```
##       int    X1     X2     X3     X4    X5     X6    X7     X8     X9    X10
## c1 -0.568 1.446  2.726  0.052 -0.001 0.020  0.028 0.014  0.013 -0.076 -0.013
## c2  0.167 0.435 -1.867 -0.001 -0.001 0.049 -0.036 0.009 -0.003  0.017 -0.007
## c3  0.000 0.000  0.000  0.000  0.000 0.000  0.000 0.000  0.000  0.000  0.000
```

## No shrinkage (NS) method

```
fit.none <- BLCshrink.none(y = y, X = X, N_it = 15000, burn.in = 5000,
    samples = TRUE, progress = FALSE)
round(fit.none$kappa.hat, 3)[, 1:11]
```

```
##       int     X1     X2    X3     X4    X5     X6    X7     X8     X9    X10
## c1 -8.305 15.359 26.914 6.360 -1.070 4.841  4.595 1.511 -0.866 -4.357  0.416
## c2 -0.289  2.958 -6.559 0.179  0.039 0.943 -1.369 0.250  0.635 -0.291 -0.345
## c3  0.000  0.000  0.000 0.000  0.000 0.000  0.000 0.000  0.000  0.000  0.000
```