Notes on the Curtis-Ghosh method

September 28, 2014

Identifiability and label-switching

The model in Curtis & Ghosh is initially described as follows:

$$\begin{aligned}
Y_i|\boldsymbol{\beta}, \sigma^2 &\sim \mathrm{N}(\boldsymbol{x}_i^T\boldsymbol{\beta}, \sigma^2) \\
\beta_j &= \gamma_j\theta_j \\
\gamma_j &\sim \mathrm{Bern}(\pi) \\
\theta_j|\mathscr{P} &\sim \mathscr{P}(\cdot) \\
\mathscr{P} &\sim \mathrm{DP}\left(\alpha, \Phi_{0,\eta^2}(\cdot)\right) \\
\pi &\sim \mathrm{Unif}(0,1) \\
\sigma^2 &\sim \mathrm{InvGam}(a_\sigma, b_\sigma) \\
\eta^2 &\sim \mathrm{InvGam}(a_\eta, b_\eta) \quad .
\end{aligned}$$

The implementation they provide approximates the Dirichlet process prior with a discrete Dirichlet prior as follows:

$$\begin{aligned}
Y_i|\boldsymbol{\beta}, \sigma^2 &\sim \mathrm{N}(\boldsymbol{x}_i^T\boldsymbol{\beta}, \sigma^2) \\
\beta_j &= \gamma_j\xi_{S_j} \\
\gamma_j &\sim \mathrm{Bern}(\pi) \\
S_j &\sim \mathrm{Multinom}(p_1, \ldots, p_M) \\
\xi_j|\eta^2 &\sim \mathrm{N}(0, \eta^2) \\
(p_1, \ldots, p_M) &\sim \mathrm{Dirichlet}(\alpha/M, \ldots, \alpha/M) \\
\pi &\sim \mathrm{Unif}(0,1) \\
\sigma^2 &\sim \mathrm{InvGam}(a_\sigma, b_\sigma) \\
\eta^2 &\sim \mathrm{InvGam}(a_\eta, b_\eta) \quad .
\end{aligned}$$

Here's why I think there's a label-switching problem.

Let $\xi_j = \nu_j$, $j = 1, \ldots, M$, where ν_j are fixed constants. Now simultaneously permute the ξ_j and the p_j so that $\xi_{j'} = \nu_j$ and $p_{j'} = p_j$, $j = 1, \ldots, M$, and $j' = 1, \ldots, M$. The distribution of ξ_j is identical under the two permutations, meaning that permutations are non-identifiable although ξ_j and p_j are still identifiable. In other words, if we switch the labeling of the clusters, the posterior remains unchanged. That's the label-switching problem.

Here's some evidence that it is a problem. Matt drew the bulk of this code from the Curtis and Ghosh paper. First, the R script that sets up the data and extracts the results after `JAGS` has run.

```
## ********************************************************************* ##
## curtis_ghosh_example.R
##
## Author: Matthew Aiello-Lammens
## Date Created: 2014-07-07
##
## Purpose:
## Re-run the analyis example using Ehrlich Crime Data presented in
## Curtis and Ghosh 2011 J. Stat. Theory and Practice.
##
## ********************************************************************* ##

## Load necessary packages
library( R2jags )
library( MASS )
library( plyr )
library( dplyr )
library( car )

standardize <- function(x) {
  if (is.numeric(x)) {
    y <- (x - mean(x, na.rm=TRUE))/sd(x, na.rm=TRUE)
  } else {
    y <- x
  }
  y
}
```

```
## Load data set
crime <- UScrime

## Log-transform variables, as Curtis and Ghosh did.
## They also dropped So, presumably because it is categorical
crime <- select( crime, -So )
crime <- log( crime )

## ------------------------------------------------------------------------ ##
## Construct simple linear regression model
## ------------------------------------------------------------------------ ##

#crime_lm <- lm( formula = y ~  . , data = crime )
#crime_lm <- lm( formula = y ~ -1 + . , data = crime ) ## No intercept model
#summary( crime_lm ) ## *DOES NOT* match results presented in Curtish and Ghosh (w
#vif( crime_lm ) ## Needs model *with* interecept to make sense. Approximately mat

## ------------------------------------------------------------------------ ##
## Construct Curtis and Ghosh Baysian dimension reduction model
## ------------------------------------------------------------------------ ##

## Separate predictors into their own vectors
M <- standardize(crime$M)
Ed <- standardize(crime$Ed)
Po1 <- standardize(crime$Po1)
Po2 <- standardize(crime$Po2)
LF <- standardize(crime$LF)
M.F <- standardize(crime$M.F)
Pop <- standardize(crime$Pop)
NW <- standardize(crime$NW)
U1 <- standardize(crime$U1)
U2 <- standardize(crime$U2)
GDP <- standardize(crime$GDP)
Ineq <- standardize(crime$Ineq)
Prob <- standardize(crime$Prob)
Time <- standardize(crime$Time)
```

```
## Define the parameters in a list structure
crime.data <-
  list( "M", "Ed", "Po1", "Po2", "LF", "M.F",
        "Pop", "NW", "U1", "U2", "GDP", "Ineq",
        "Prob", "Time" )

## Alternatively extract data as a matrix
#crime.pred.mat <-
X <-
  as.matrix( select( crime, -y ) )

## Separate response variable into it's own vector
y <- crime$y

## Define model settings
n.samp <- nrow( crime )
## The next parameter, M, is somewhat arbitrarily set. The authors
## write use a "suitably large M", and define it as p + 25, where
## p is the number of coefficients in thier examples
M <- ( ncol( crime ) - 1 ) + 25
## Set number of regression coefficients
K <- ncol( crime ) - 1

jags.data <-
  c( "X", "n.samp", "M", "K", "y" )

## Set model file
if (anchored) {
  model.file <- "curtis_ghosh_anchored.jags"
} else {
  model.file <- "curtis_ghosh_example.jags"
}

## Set JAGs parameters
n.chains <- 4
n.burnin <- 5*10000
n.iter <- 5*72500
n.thin <- 5*50
```

```
jags.par <-
  c( "beta", "gamma", "S", "theta", "xi" )

gamma.matrix <- matrix(ncol=n.chains*(n.iter-n.burnin)/n.thin)
theta.matrix <- matrix(ncol=n.chains*(n.iter-n.burnin)/n.thin)
S.matrix <- matrix(ncol=n.chains*(n.iter-n.burnin)/n.thin)
xi.array <- array(dim=c(4,M,n.chains*(n.iter-n.burnin)/n.thin))
for (i in 1:4) {
  sink("temp.txt",
       append=TRUE)
  fit <-
    jags( data = jags.data,
          inits = NULL,
          parameters = jags.par,
          model.file = model.file,
          n.chains = n.chains,
          n.burnin = n.burnin,
          n.iter = n.iter,
          n.thin = n.thin,
          DIC = TRUE,
          working.directory = "." )
  sink()

  sink("curtis-ghosh-results.txt", append=TRUE)
  old.opt <- options(width=180)
  print(fit, 3)
  options(old.opt)
  cat("\n\n\n")
  sink()

  gamma <- fit$BUGSoutput$sims.list$gamma[,2]
  gamma.matrix <- rbind(gamma.matrix, gamma)

  theta <- fit$BUGSoutput$sims.list$theta[,2]
  theta.matrix <- rbind(theta.matrix, theta)

  S <- fit$BUGSoutput$sims.list$S[,2]
```

```
  S.matrix <- rbind(S.matrix, S)

  xi.array[i,,] <- fit$BUGSoutput$sims.list$xi
}

xi.matrix <- matrix(nrow=dim(xi.array)[1], ncol=dim(xi.array)[2])
xi.means <- t(aaply(xi.array, c(1,2), mean))
colnames(xi.means) <- paste("Replicate", colnames(xi.means))
```

Now the JAGS script that runs the analysis.

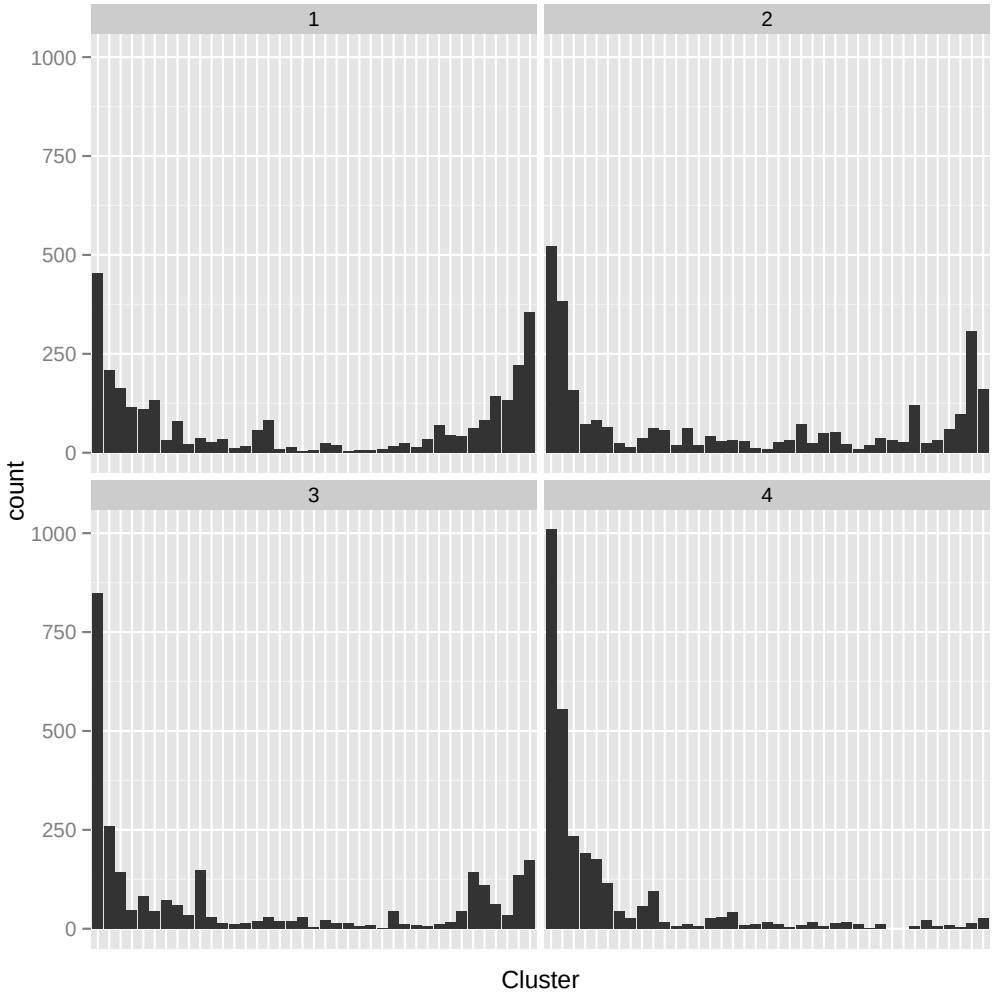
```
model{
  for( i in 1:n.samp ) {
    y[i] ~ dnorm( mu[i], isigsq )
    mu[i] <- inprod( X[i,], beta[] ) # inner product of X[i,] and beta[]
  }

  alpha <- 1

  for( j in 1:M ){  # M is adequately large
    xi[j] ~ dnorm( 0, itausq )
    a.p[j] <- alpha / M
  }

  for( j in 1:K ){  # K is the number of regression coefficients
    S[j] ~ dcat( p[1:M] )
    theta[j] <- xi[ S[j] ]
    gamma[j] ~ dbern( pi )
    beta[j] <- theta[j] * gamma[j]
  }

  p[1:M] ~ ddirch( a.p[] )
  pi ~ dunif (0 ,1)

  # Prior for sigsq
  isigsq ~ dgamma (0.1 ,0.1)
  sigsq <- 1 / isigsq
```

```
  # Prior for tausq
  itausq ~ dgamma (0.1 ,0.1)
  tausq <- 1 / itausq
}
```

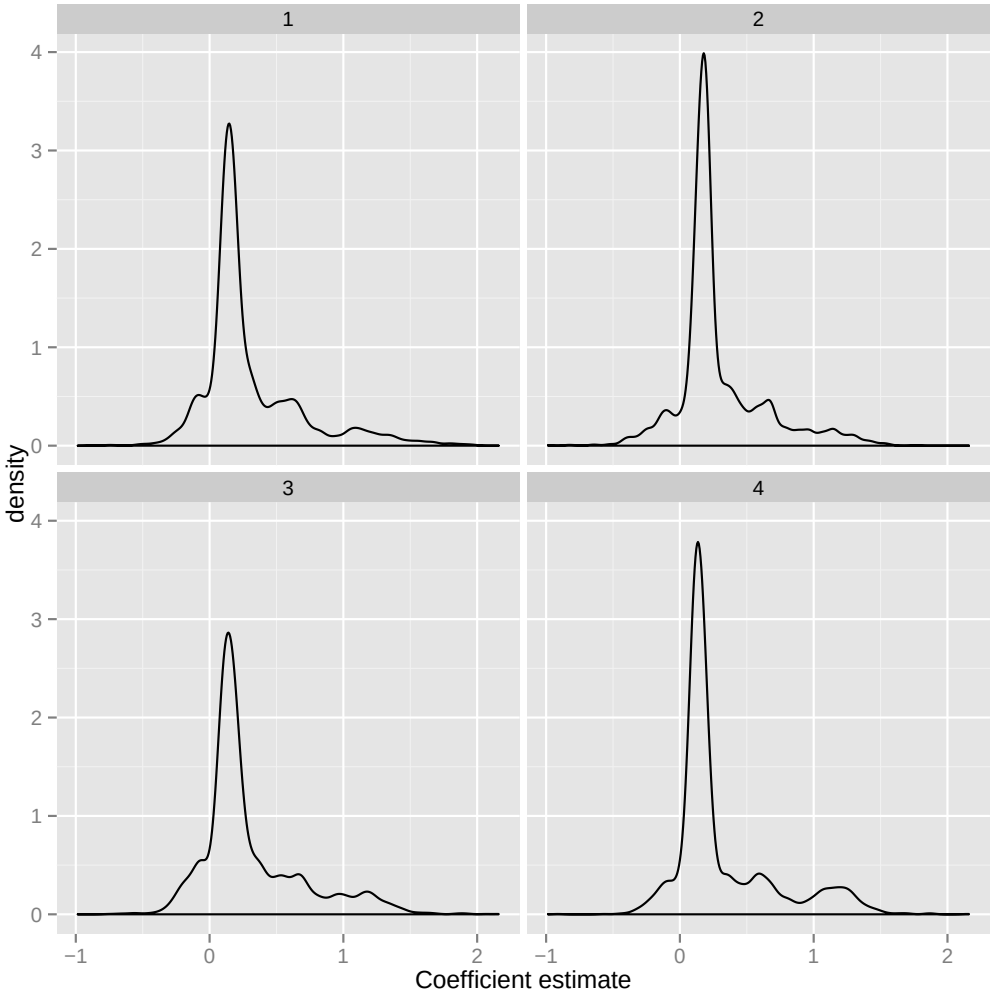
Notice that in the last lines of the R code, I'm extracting the cluster identity for the second covariate, *Ed*, and the probability that it is included in the model at each MCMC iteration. Let's look at how the cluster identity changes over iterations when *Ed* is included in the model, i.e., when $\gamma_2 > 0$.

```
require(reshape2)
require(ggplot2)

prepare <- function(S, gamma, theta) {
  ## first column will contain only NA, so remove
  ## it
  S <- t(S)[, -1]
  gamma <- t(gamma)[, -1]
  theta <- t(theta)[, -1]

  ## Bad form: hardcoded for 4 replicates
  colnames(S) <- c("1", "2", "3", "4")
  colnames(gamma) <- c("1", "2", "3", "4")
  colnames(theta) <- c("1", "2", "3", "4")

  ## convert to long form
  S.long <- melt(S)
  gamma.long <- melt(gamma)
  theta.long <- melt(theta)
  colnames(S.long) <- c("Iteration", "Replicate",
    "Cluster")
  colnames(gamma.long) <- c("Iteration", "Replicate",
    "Included")
  colnames(theta.long) <- c("Iteration", "Replicate",
    "theta")
  ## merge into one data frame
  z <- merge(S.long, gamma.long)
  z <- merge(z, theta.long)
```

```
  z
}

anchored <- FALSE

source("curtis_ghosh_example.R")

S.plot <- prepare(S.matrix, gamma.matrix, theta.matrix)

for.plot <- subset(S.plot[, c("Cluster", "Replicate")],
  S.plot$Included > 0)
p <- ggplot(for.plot, aes(x = as.factor(Cluster))) +
  geom_bar() + xlab("Cluster") + theme(axis.text.x = element_blank(),
  axis.ticks.x = element_blank()) + facet_wrap(~Replicate)
print(p)
```




```
for.plot <- subset(S.plot[, c("Replicate", "theta")],
  S.plot$Included > 0)
p <- ggplot(for.plot, aes(x = theta)) + geom_density() +
  xlab("Coefficient estimate") + facet_wrap(~Replicate)
print(p)
```



It's also worth comparing the posterior mean of β_2 in each of the 4 chains. The mean reported in Table 1 is conditional on the sample having been included in the model at that iteration, i.e., $\gamma_1 = 1$.

```
require(xtable)

theta.means <- ddply(for.plot, c("Replicate"),
  summarize, mean = mean(theta), sd = sd(theta))
theta.table <- xtable(theta.means[, -1], digits = 3,
  caption = "Unanchored", label = "table:theta")
```

```
print(theta.table, file = "theta-table.tex")
```

	mean	sd
1	0.299	0.373
2	0.289	0.329
3	0.302	0.376
4	0.313	0.379

Table 1: Unanchored

ξ_j is the mean of the jth cluster. If there is label switching within a chain, then the means of the clusters that are being switched between should be pretty close. Table 2 shows the posterior mean for each of the clusters in each replicate.

```
xi.table <- xtable(xi.means, digits = 3, caption = "Unanchored",
  label = "table:xi")
print(xi.table, file = "xi-table.tex")

fit.unanchored <- fit
```

Informal experiments, i.e., playing around, suggest that one long run with aggressive thinning might avoid the label switching problem. It appears that any one chain "locks in" to a particular identification and doesn't move (much).

Avoiding label switching

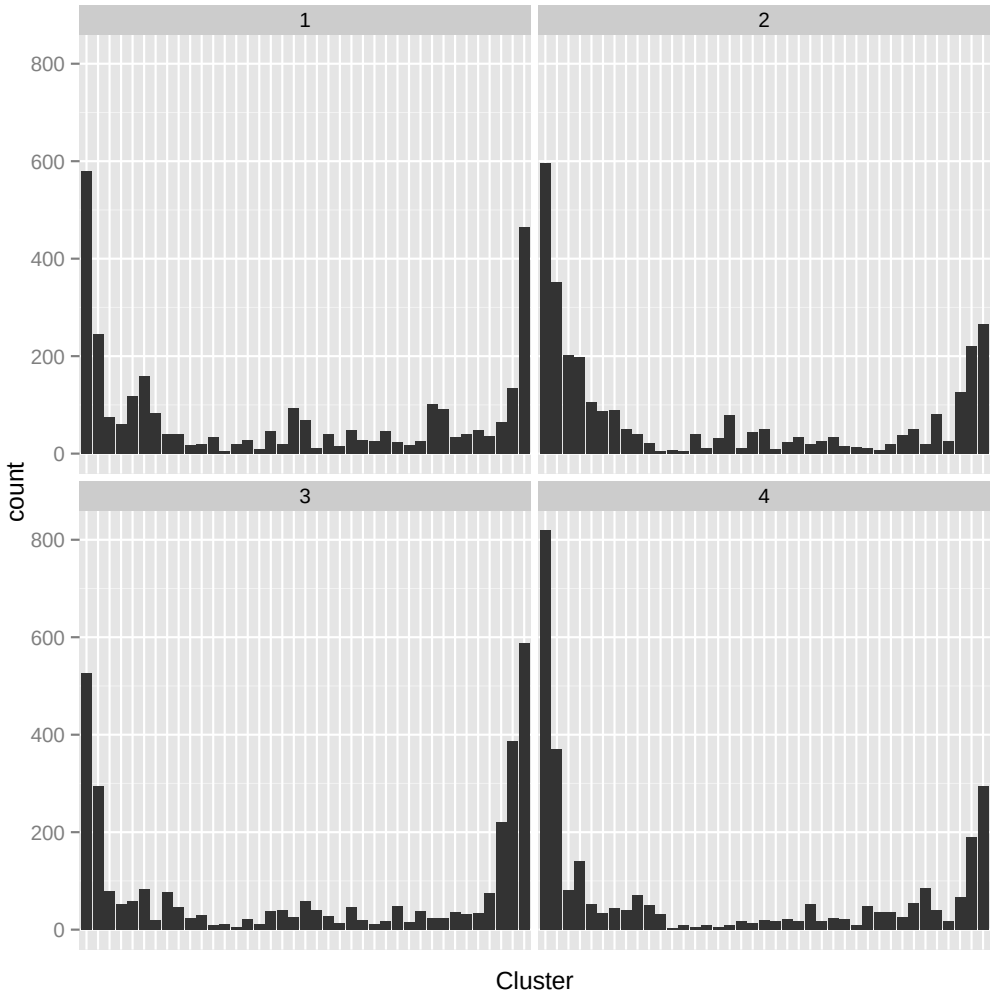
Here's an attempt to avoid (or reduce) the label switching. Rather than leaving the cluster assignment completely to `JAGS`, I partially "anchor" the assignment by hard-coding the coefficient for the first covariate as belonging to the first cluster. This is the modified `JAGS` code:

```
model{
  for( i in 1:n.samp ) {
    y[i] ~ dnorm( mu[i], isigsq )
    mu[i] <- inprod( X[i,], beta[] ) # inner product of X[i,] and beta[]
  }
```

```
  alpha <- 1

  for( j in 1:M ){  # M is adequately large
    xi[j] ~ dnorm( 0, itausq )
    a.p[j] <- alpha / M
  }

  S[1] <- 1
  for (j in 2:K) {
    S[j] ~ dcat( p[1:M] )
  }
  for( j in 1:K ){  # K is the number of regression coefficients
    theta[j] <- xi[ S[j] ]
    gamma[j] ~ dbern( pi )
    beta[j] <- theta[j] * gamma[j]
  }

  p[1:M] ~ ddirch( a.p[] )
  pi ~ dunif (0 ,1)

  # Prior for sigsq
  isigsq ~ dgamma (0.1 ,0.1)
  sigsq <- 1 / isigsq

  # Prior for tausq
  itausq ~ dgamma (0.1 ,0.1)
  tausq <- 1 / itausq
}
```

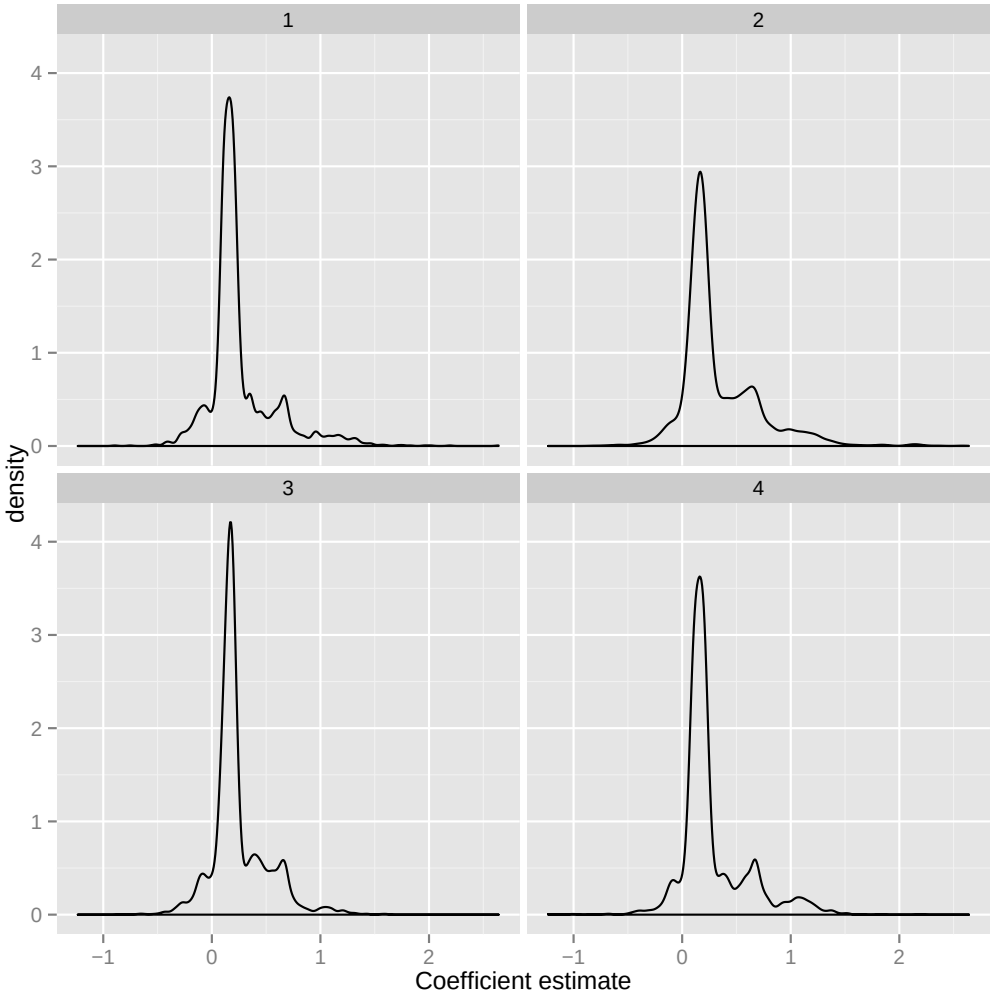
```
anchored <- TRUE

source("curtis_ghosh_example.R")
S.plot <- prepare(S.matrix, gamma.matrix, theta.matrix)

for.plot <- subset(S.plot[, c("Cluster", "Replicate")],
  S.plot$Included > 0)
```

```
p <- ggplot(for.plot, aes(x = as.factor(Cluster))) +
  geom_bar() + xlab("Cluster") + theme(axis.text.x = element_blank(),
  axis.ticks.x = element_blank()) + facet_wrap(~Replicate)
print(p)
```




```
for.plot <- subset(S.plot[, c("Replicate", "theta")],
  S.plot$Included > 0)
p <- ggplot(for.plot, aes(x = theta)) + geom_density() +
  xlab("Coefficient estimate") + facet_wrap(~Replicate)
```

```
print(p)
```



It's also worth comparing the posterior mean of β_2 in each of the 4 chains. The mean reported in Table 3 is conditional on the sample having been included in the model at that iteration, i.e., $\gamma_1 = 1$.

```
require(xtable)

theta.means <- ddply(for.plot, c("Replicate"),
  summarize, mean = mean(theta), sd = sd(theta))
```

```
theta.table <- xtable(theta.means[, -1], digits = 3,
  caption = "Anchored", label = "table:anchored-theta")
print(theta.table, file = "theta-table-anchored.tex")
```

ξ_j is the mean of the jth cluster. If there is label switching within a chain, then the means of the clusters that are being switched between should be pretty close. Table 4 shows the posterior mean for each of the clusters in each replicate.

```
xi.table <- xtable(xi.means, digits = 3, caption = "Anchored",
  label = "table:anchored-xi")
print(xi.table, file = "xi-table-anchored.tex")

fit.anchored <- fit
```

Summarizing the posterior

We are interested in identifying those covariates that have similar associationss with the response variable. One way of doing so is to calculate a dissimilarity coefficient for all pairs of covariates. It makes sense to estimate such a coefficient as the posterior mean of a dissimilarity coefficient calculated at each step of an MCMC simulation. The coefficient calculated at each step should have two properties:

1. It should be equal to 1 unless both coefficients are included in the model in that step.
2. It should be equal to (Coeff1 − Coeff2)/(Max(Coeff) − Min(Coeff)) when both coefficients are included.

```
dissim.iter <- function(x, gamma.x, y, gamma.y,
  max.coeff, min.coeff) {
  sim <- 0
  ct <- 0
  for (i in 1:length(x)) {
```

```
    if ((gamma.x[i] > 0) && (gamma.y[i] >
      0)) {
      sim <- sim + abs(x[i] - y[i])/(max.coeff -
        min.coeff)
    } else {
      sim <- sim + 1
    }
    ct <- ct + 1
  }
  sim <- sim/ct
  sim
}
```

With that code in place, it's just a matter of making all of the possible pairwise comparison after setting up the maximum difference between coefficient estimates across the entire sample. For now, I'm summarizing the result using UPGMA clustering, but we could also use the UPGMA clustering to order the variables and use `corplot()` to provide a colorful visualization of the whole distance matrix.

```
dissim.cluster <- function(fit, label) {
  max.coeff <- max(fit$BUGSoutput$sims.list$theta)
  min.coeff <- min(fit$BUGSoutput$sims.list$theta)
  dissim <- matrix(0, ncol = K, nrow = K)
  for (i in 2:K) {
    dissim[i, i] <- 0
    for (j in 1:(i - 1)) {
      dissim[i, j] <- dissim.iter(fit$BUGSoutput$sims.list$theta[,
        i], fit$BUGSoutput$sims.list$gamma[,
        i], fit$BUGSoutput$sims.list$theta[,
        j], fit$BUGSoutput$sims.list$gamma[,
        j], max.coeff, min.coeff)
      dissim[j, i] <- dissim[i, j]
    }
  }
  colnames(dissim) <- crime.data
  rownames(dissim) <- crime.data
```

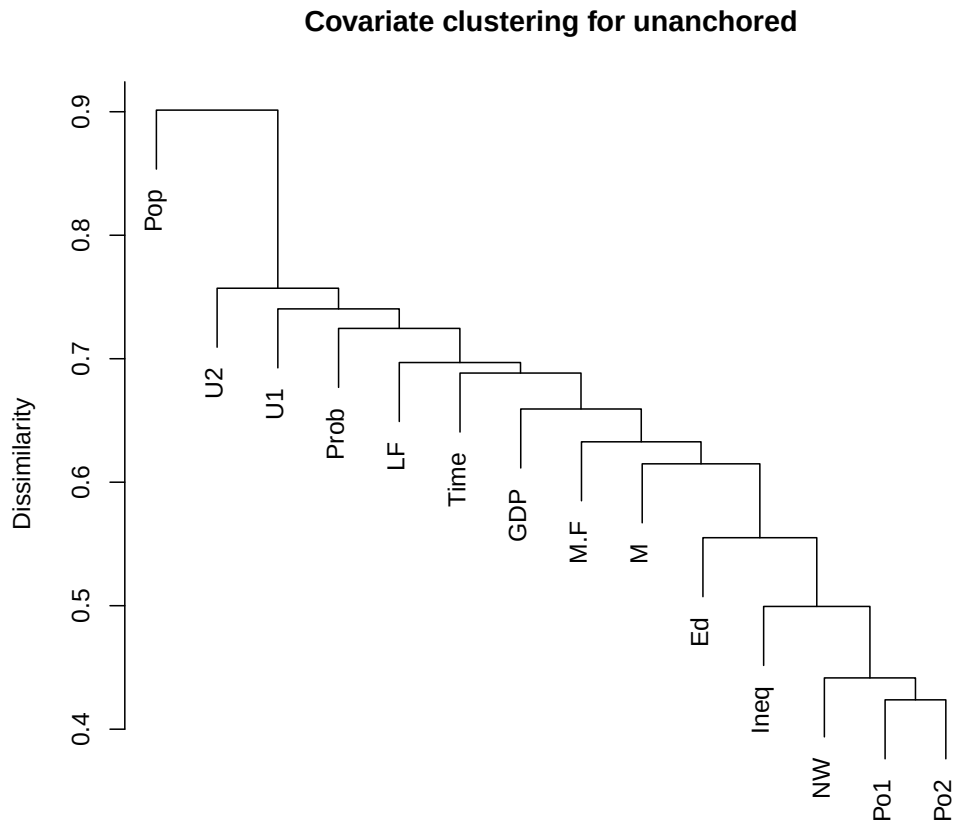
```
  dissim.hclust <- hclust(as.dist(dissim), method = "average")
  plot(dissim.hclust, main = paste("Covariate clustering for",
    label), sub = "", xlab = "", ylab = "Dissimilarity")
  ## return the dissimilarity matrix for later
  ## use
  dissim
}
```

Alternatively, we could display the results in two dimensions usins multidimensional scaling.

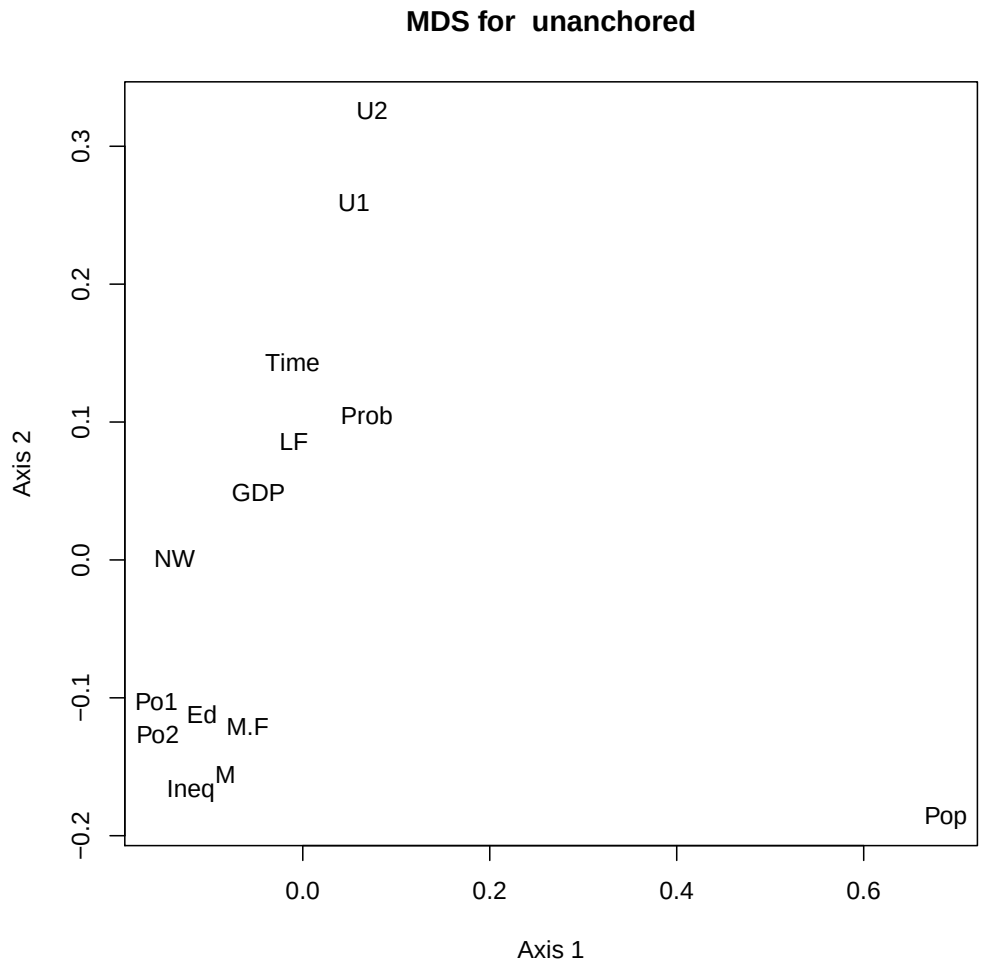
```
dissim.MDS <- function(dissim, label) {
  dissim.mds <- cmdscale(dissim)
  plot(dissim.mds, typ = "n", xlab = "Axis 1",
    ylab = "Axis 2", main = paste("MDS for ",
      label))
  text(dissim.mds, rownames(dissim))
}
```

With the functions set up, we can now compare the clustering revealed by the two different methods to see if it makes any difference on what clusters of covariates we might recognize.

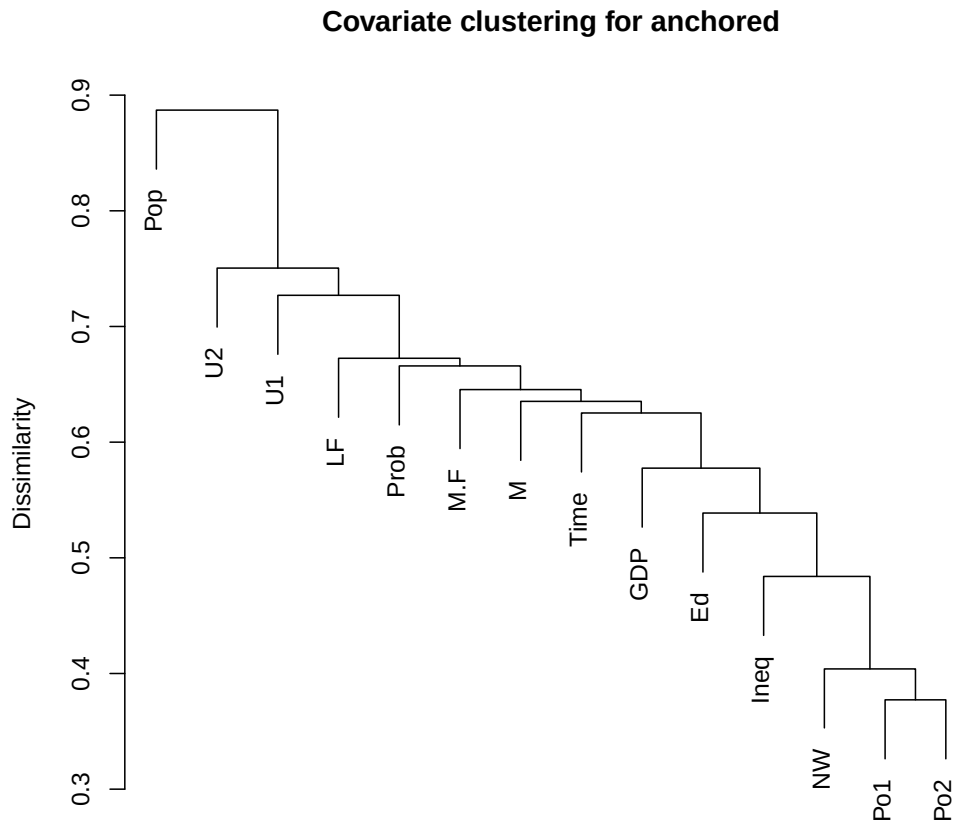
```
result <- dissim.cluster(fit.unanchored, "unanchored")
```



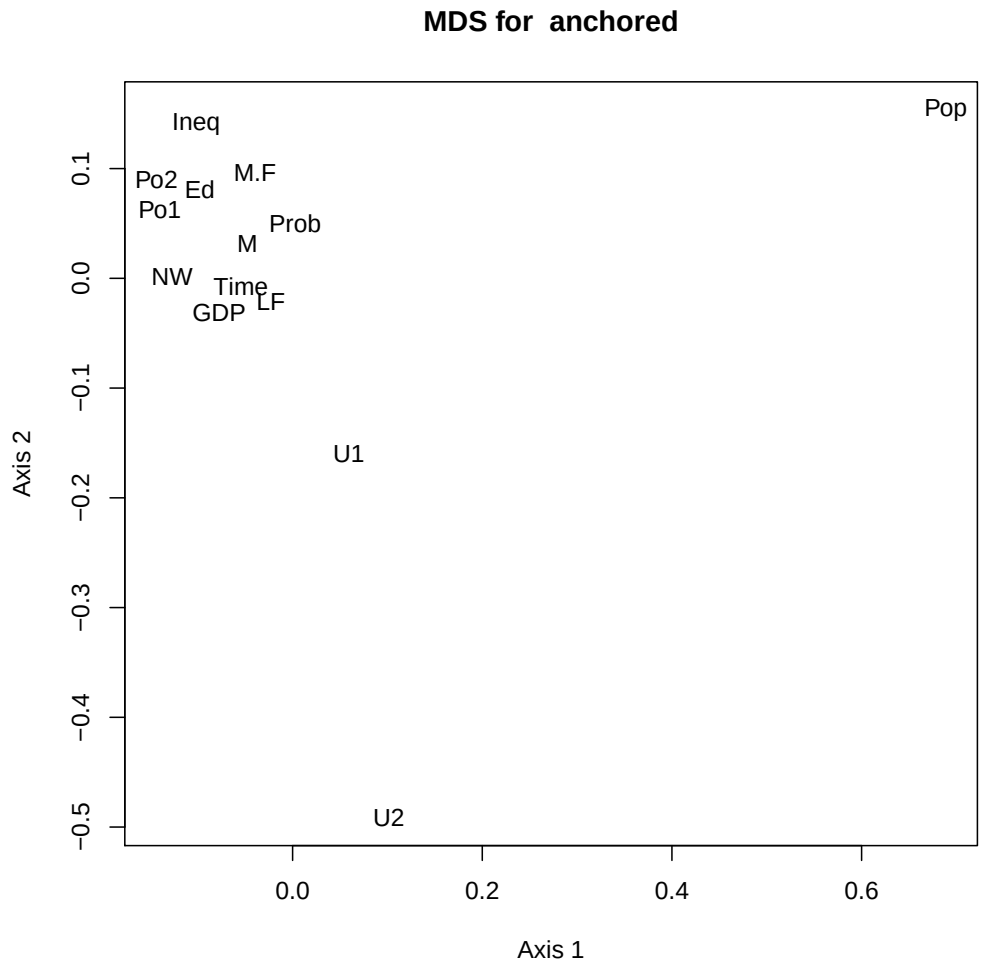

```
dissim.MDS(result, "unanchored")
```




```
result <- dissim.cluster(fit.anchored, "anchored")
```




```
dissim.MDS(result, "anchored")
```


MDS for anchored
Ineq
Pop
Po2
Ed
M.F
Po1
Prob
M
NW
Time
LF
GDP
U1
U2
Axis 2
Axis 1
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
0.0
0.2
0.4
0.6

	Replicate 1	Replicate 2	Replicate 3	Replicate 4
1	0.039	0.006	-0.428	0.047
2	-0.002	-0.007	0.270	0.081
3	-0.036	-0.013	-0.523	0.090
4	-0.017	0.019	-0.002	0.132
5	0.037	-0.026	0.203	-0.014
6	0.032	-0.010	-0.193	-0.067
7	-0.035	0.028	-0.851	0.024
8	0.039	-0.009	0.021	0.028
9	-0.014	-0.018	0.502	-0.004
10	0.090	-0.006	0.129	0.114
11	0.007	0.020	-0.505	-0.007
12	0.051	0.010	0.254	-0.003
13	0.057	0.073	-0.595	-0.066
14	-0.020	-0.085	-0.566	-0.021
15	0.013	0.030	-0.233	-0.021
16	0.011	0.049	0.146	0.002
17	0.038	0.015	-0.151	-0.033
18	-0.026	0.003	-0.243	0.027
19	0.029	0.051	-0.077	-0.033
20	0.010	0.068	-0.317	-0.006
21	0.050	0.020	0.560	-0.087
22	-0.034	0.048	-0.174	-0.051
23	0.024	0.028	-0.092	-0.056
24	0.013	-0.077	-0.353	-0.019
25	0.022	-0.039	-0.234	-0.011
26	-0.012	-0.015	0.451	-0.164
27	0.024	-0.054	0.775	0.120
28	0.056	-0.071	0.394	-0.001
29	-0.008	-0.045	0.386	0.053
30	-0.018	0.000	0.050	0.008
31	0.018	-0.053	0.081	0.020
32	0.039	0.066	-0.025	0.016
33	-0.018	-0.038	0.163	0.012
34	0.024	0.016	-0.594	0.028
35	-0.007	0.039	0.154	0.139
36	-0.021	-0.021	0.065	0.112
37	0.016	-0.005	0.024	-0.051
38	-0.007	-0.009	-0.438	-0.013
39	0.047	0.041	-0.202	0.069

Table 2: Unanchored

	mean	sd
1	0.261	0.314
2	0.330	0.352
3	0.243	0.256
4	0.275	0.313

Table 3: Anchored

	Replicate 1	Replicate 2	Replicate 3	Replicate 4
1	0.085	-0.034	0.033	0.001
2	-0.066	-0.002	0.044	0.026
3	-0.002	0.002	0.023	0.034
4	0.005	0.036	-0.013	0.052
5	0.054	0.007	0.026	-0.020
6	-0.060	0.011	-0.010	-0.001
7	0.017	0.062	0.003	0.076
8	0.057	0.004	0.023	-0.022
9	0.144	-0.022	0.004	0.024
10	0.037	0.016	0.026	-0.019
11	0.017	0.012	0.031	0.061
12	-0.007	0.011	0.007	0.082
13	-0.092	0.019	0.020	0.029
14	-0.006	-0.005	-0.006	0.012
15	0.049	-0.011	0.022	0.008
16	-0.051	0.012	0.003	0.055
17	0.050	0.014	0.004	0.018
18	0.047	0.008	0.014	0.005
19	0.048	0.008	0.020	0.082
20	-0.102	-0.006	0.023	-0.022
21	0.054	0.047	0.023	0.043
22	0.046	0.022	-0.003	0.028
23	-0.007	-0.036	0.024	-0.011
24	0.007	0.018	-0.026	-0.032
25	0.003	0.061	0.013	-0.012
26	-0.029	0.020	0.016	-0.035
27	-0.031	0.035	0.036	0.067
28	-0.014	0.030	0.002	0.045
29	0.003	0.027	0.014	-0.006
30	-0.017	0.003	-0.004	0.032
31	-0.032	0.006	0.023	-0.020
32	-0.048	0.014	-0.022	-0.028
33	-0.032	0.015	0.008	-0.002
34	0.052	0.026	0.004	-0.051
35	0.028	0.019	0.011	-0.015
36	-0.058	-0.014	-0.023	-0.005
37	-0.041	0.023	0.022	0.045
38	0.027	0.008	0.026	0.007
39	0.079	-0.009	0.009	0.013

Table 4: Anchored