Stochastic Search Variable Selection with PROC MCMC

Overview

Suppose you want to model the relationship between a response variable and a set of potential explanatory variables, but you have reason to believe that some of the potential explanatory variables are redundant or irrelevant. With 2^p possible models from which to choose, how do you find the best subset of explanatory variables to include in your model? To solve this problem, you must choose a criterion by which the candidate models can be ranked, and you must have a computationally feasible strategy for searching through the model space to find the candidate models that exhibit optimal values of the chosen criterion. Stochastic search variable selection (SSVS) is a Bayesian modeling method that enables you to select promising subsets of the potential explanatory variables for further consideration.

For SSVS, you express the relationship between the response variable and the candidate predictors in the framework of a hierarchical normal mixture model, where latent variables are used to identify subset choices. In this framework, the promising subsets of predictors are identified as those that have a higher posterior probability. SSVS uses Markov chain Monte Carlo (MCMC) sampling to indirectly sample from this posterior distribution on the set of possible subset choices. Subsets that have a higher posterior probability are identified by their more frequent appearance in the MCMC sample. In this way, SSVS avoids the problem of computing the posterior probabilities of all 2^p subsets (George and McCulloch 1993).

The following analysis presents SSVS in the context of linear regression by using the MCMC procedure in SAS/STAT software. However, SSVS can be readily extended to accommodate generalized linear models.

The SAS source code for this example is available as an attachment in a text file. In Adobe Acrobat, right-click the icon and select **Save Embedded File to Disk**. You can also double-click the icon to open the file immediately.

source code

Analysis

Consider a linear regression model,

$$\mathbf{Y}|\boldsymbol{\beta}, \sigma^2 \sim \mathrm{N}_n(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}) \tag{1}$$

where $\boldsymbol{\beta}$ and σ^2 are the model's unknown parameters. SSVS extracts information relevant to variable selection by embedding this regression model in a larger hierarchical model. The key feature of this hierarchical model is that each component of $\boldsymbol{\beta}$ is modeled as having come from a mixture of two normal distributions that have different variances (George and McCulloch 1993).

To extract the relevant information, you first index the 2^p possible subset choices by a vector of binary variables,

$$\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)' \tag{2}$$

where $\gamma_i = 0$ if β_i is "small" (meaning that $\mathbf{X}_i$ is to be excluded from the model), $\gamma_i = 1$ if β_i is "large" (meaning that $\mathbf{X}_i$ is to be included in the model), and

$$\Pr(\gamma_i = 1) = 1 - \Pr(\gamma_i = 0) = (\omega_i) \tag{3}$$

You then specify a multivariate normal prior distribution for $\boldsymbol{\beta}|\boldsymbol{\gamma}$ of the form

$$\boldsymbol{\beta}|\boldsymbol{\gamma} \sim \mathbf{N}_p(0, \mathbf{D}_\gamma \mathbf{R} \mathbf{D}_\gamma) \tag{4}$$

where $\mathbf{R}$ is the prior correlation matrix of $\boldsymbol{\beta}$ conditional on $\boldsymbol{\gamma}$ and $\mathbf{D}_\gamma$ is a diagonal matrix whose diagonal elements are

$$\mathbf{D}_{\boldsymbol{\gamma}\, ii} = \begin{cases} \sqrt{v_{0i}} & \text{when } \gamma_i = 0 \\ \sqrt{v_{1i}} & \text{when } \gamma_i = 1 \end{cases}$$

Now each $\beta_i|\gamma_i$ has a prior of the form

$$\beta_i|\gamma_i \sim (1 - \gamma_i) N(0, v_{0i}) + \gamma_i N(0, v_{1i}) \tag{5}$$

The hyperparameters v_{0i} and v_{1i} can be thought of as tuning constants by which you calibrate SSVS. The idea is to set v_{0i} small, so that when $\gamma_i = 0$ and $\beta_i \sim N(0, v_{0i})$, the prior distribution for $\beta_i|\gamma_i$ is concentrated around 0, accentuating prior influence and driving the estimate of β_i to 0. In contrast, you set v_{1i} large, so that when $\gamma_i = 1$ and $\beta_i \sim N(0, v_{1i})$, the prior distribution for β_i is diffuse, reducing prior influence and providing support for values of β_i that are substantively different from 0 (Chipman et al. 2001).

Like v_{0i} and v_{1i}, the prior correlation matrix $\mathbf{R}$ can be regarded as a tuning constant. The most common choices are $\mathbf{R} = \mathbf{I}$ and $\mathbf{R} \propto (\mathbf{X}'\mathbf{X})^{-1}$. When $\mathbf{R} = \mathbf{I}$, the components of $\boldsymbol{\beta}$ are independent under $f(\boldsymbol{\beta}|\boldsymbol{\gamma})$. When $\mathbf{R} \propto (\mathbf{X}'\mathbf{X})^{-1}$, the prior correlation is identical to the design correlation. Another alternative is to specify a prior for $\mathbf{R}$, but doing so substantially increases the computational burden.

The next step is to specify a prior for $\boldsymbol{\gamma}$ that satisfies equation (3). Many Bayesian variable selection implementations use independence priors of the form

$$f(\boldsymbol{\gamma}) = \prod \omega_i^{\gamma_i} (1 = \omega_i)^{1-\gamma_i} \tag{6}$$

Under this prior, each X_i enters the model independent of the other regressors. However, you can use any discrete distribution that has support on the 2^p possible values of $\boldsymbol{\gamma}$.

To complete the model, you specify a prior for the residual variance σ^2. The most common choice is an inverse gamma prior for σ^2 of the form

$$\sigma^2|\boldsymbol{\gamma} \sim \text{IG}(\nu_\gamma/2, \nu_\gamma\lambda_\gamma/2) \tag{7}$$

Usually, ν_γ and λ_γ are treated as constants; that is, $\nu_\gamma \equiv \nu$ and $\lambda_\gamma \equiv \lambda$. However, the more general representation of equation (7) enables you to model σ^2 as a function of the number of predictors that are included in the model.

Embedding the normal linear model in the hierarchical mixture model enables you to obtain the marginal posterior distribution $f(\boldsymbol{\gamma}|\mathbf{Y}) \propto f(\mathbf{Y}|\boldsymbol{\gamma})f(\boldsymbol{\gamma})$, which contains the information relevant to variable selection. Given the data $\mathbf{Y}$, the posterior $f(\boldsymbol{\gamma}|\mathbf{Y})$ updates the prior probabilities of each of the 2^p possible values of $\boldsymbol{\gamma}$. Those $\boldsymbol{\gamma}$ that have the higher posterior probability $f(\boldsymbol{\gamma}|\mathbf{Y})$ identify the submodels that are supported most by the data and by your prior information. Thus, SSVS satisfies the first stated goal of variable selection: $f(\boldsymbol{\gamma}|\mathbf{Y})$ provides a criterion by which the candidate models can be ranked.

The second stated goal of variable selection is to have a computationally feasible strategy for searching through the model space to find candidate models that exhibit optimal values of the chosen criterion. Rather than calculate all 2^p posterior probabilities in $f(\boldsymbol{\gamma}|\mathbf{Y})$, SSVS satisfies the second goal by using an MCMC sampler to generate the sequence

$$\boldsymbol{\gamma}^1, \ldots, \boldsymbol{\gamma}^m \tag{8}$$

In many cases, this sequence converges rapidly in distribution to $\boldsymbol{\gamma} \sim f(\boldsymbol{\gamma}|\mathbf{Y})$. Because those $\boldsymbol{\gamma}$ that have the highest posterior probability appear most frequently in the sample, the sequence contains exactly the information that is relevant to variable selection. Those $\boldsymbol{\gamma}$ that appear infrequently or not at all are simply not of interest and can be disregarded.

NOTE: It is assumed that $\mathbf{X}_1, \ldots, \mathbf{X}_p$ contains no variable that would be included in every possible model. If there is a subset $\mathbf{X}_1^*, \ldots, \mathbf{X}_r^*$ that is to be included in every possible model, then $\mathbf{X}_1^*, \ldots, \mathbf{X}_r^*$ should be removed from $\mathbf{X}_1, \ldots, \mathbf{X}_p$, and $\mathbf{Y}$ and the remaining $\mathbf{X}_i$ should be replaced by the residual vectors $(\mathbf{I} - \mathbf{X}^*(\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}\mathbf{X}^{*\prime})\mathbf{Y}$ and $(\mathbf{I} - \mathbf{X}^*(\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}\mathbf{X}^{*\prime})\mathbf{X}_i$. For example, if an intercept is to be included in every model, then you should exclude $\mathbf{1}_p \equiv (1, \ldots, 1)$ from the set of potential predictors and replace $\mathbf{Y}$ and $\mathbf{X}$ with their centered counterparts (George and McCulloch 1993).

Example: Modeling Baseball Players' Salaries

The following data set contains salary and performance information about Major League Baseball players (excluding pitchers) who played at least one game in both the 1986 and 1987 seasons. The salaries are from the 1987 season (Time Inc. 1987), and the performance measures are from the 1986 season (Collier Books 1987). Suppose you want to investigate whether you can model the players' salaries for the 1987 season

by using performance measures from the 1986 season. This example shows how you can use SSVS as a starting point for such an analysis. Because the variation in salaries is much greater for the higher salaries, this example applies a log transformation to the salaries before performing the model selection.

```
data baseball;
   length name $ 18;
   length team $ 12;
   input name $ 1-18 nAtBat nHits nHome nRuns nRBI nBB
         yrMajor crAtBat crHits crHome crRuns crRbi crBB
         league $ division $ team $ position $ nOuts nAssts
         nError salary;
   logSalary = log(Salary);
   label name="Player's Name"
         nAtBat="Times at Bat in 1986"
         nHits="Hits in 1986"
         nHome="Home Runs in 1986"
         nRuns="Runs in 1986"
         nRBI="RBIs in 1986"
         nBB="Walks in 1986"
         yrMajor="Years in the Major Leagues"
         crAtBat="Career times at bat"
         crHits="Career Hits"
         crHome="Career Home Runs"
         crRuns="Career Runs"
         crRbi="Career RBIs"
         crBB="Career Walks"
         league="League at the end of 1986"
         division="Division at the end of 1986"
         team="Team at the end of 1986"
         position="Position(s) in 1986"
         nOuts="Put Outs in 1986"
         nAssts="Assists in 1986"
         nError="Errors in 1986"
         salary="1987 Salary in $ Thousands";
   datalines;
Allanson, Andy       293    66     1    30    29    14
                       1   293    66     1    30    29    14
                    American East Cleveland C 446 33 20 .
Ashby, Alan          315    81     7    24    38    39
                      14  3449   835    69   321   414   375
                    National West Houston C 632 43 10 475
Davis, Alan          479   130    18    66    72    76
                       3  1624   457    63   224   266   263
                    American West Seattle 1B 880 82 14 480
Dawson, Andre        496   141    20    65    78    37
                      11  5628  1575   225   828   838   354

   ... more lines ...

Upshaw, Willie       573   144     9    85    60    78
                       8  3198   857    97   470   420   332
                    American East Toronto 1B 1314 131 12 960
Wilson, Willie       631   170     9    77    44    31
```

```
                    11  4908  1457    30   775   357   249
                American West KansasCity CF 408 4 3 1000
;
```

PROC MCMC does not support a CLASS statement, so if your data contain categorical variables, you need to construct the correct design matrix before you call PROC MCMC. A convenient tool to use is the TRANSREG procedure, which offers both indicator and effects coding methods. You can specify any categorical variables in the CLASS statement and use the ZERO= option to select a reference category. The transformation options in the MODEL statement also provide a convenient means of centering the data. Another benefit of using the TRANSREG procedure to construct the design matrix is that it automatically generates a global macro variable, &_TrgInd, that contains a list of the independent variables that are created. This macro variable is useful later when you specify the regression model in PROC MCMC.

The following statements use the TRANSREG procedure to prepare the input data set Baseball for use with the MCMC procedure. The DESIGN option in the PROC TRANSREG statement specifies that your primary goal is design matrix coding, not analysis. The IDENTITY and TSTANDARD=CENTER options in the MODEL statement center the specified variables. The CLASS option expands the specified variables to a set of indicator variables, and the ZERO=LAST option sets to missing the coded variable for the last of the sorted categories. The OUTPUT statement saves the the transformed variables in the SAS data set Design. The DROP= data set option drops the intercept from the design matrix. The DREPLACE and IREPLACE options replace the original dependent and independent variables, respectively, with the transformed (centered) variables in the OUT= data set.

```
proc transreg data=baseball(where=(~missing(logsalary))) design;
   model identity(logsalary / tstandard=center) =  class(division league/ zero=last)
         identity(nAtBat nHits nHome nRuns nRBI nBB yrMajor crAtBat
         crHits crHome crRuns crRbi crBB nOuts nAssts nError / tstandard=center);
   output out=design(drop=_: Int:) dreplace ireplace;
run;
```

The following SAS statements count the number of "words"—each word is the name of an independent variable—in the macro variable &_TrgInd and store the value in the global macro variable &p. The macro variable &p is used later, when you use the MCMC procedure to implement SSVS.

```
%global p;
%let p=%eval(%sysfunc(countw(&_trgind)));
```

The following SAS statements use PROC MCMC to implement SSVS. This example uses the simplest specifications. That is, the prior correlation matrix $\mathbf{R}$ is assumed to be an identity matrix, and therefore $\mathbf{DRD}$ is a simple diagonal matrix. The tuning parameters, v_{0i}, v_{1i}, ω_i, ν_γ, and λ_γ, are all specified as constants. Specifically, $v_0 = 1e-8$, $v_1 = 1$, $\omega = 0.5$, $\nu = 0.2$, and $\lambda = 100$.

```
ods graphics on;
proc mcmc data=design nmc=2000 seed=194735 propcov=quanew ntu=1000
          outpost=outpost monitor=(i1-i&p) diag=none plots(unpack)=trace;

  array DRD[&p,&p];
  array V[2];
  array mu0[&p];
  array X[&p] &_trgind;
```

```
  array beta[&p] beta1-beta&p;

  begincnst;
     call identity(DRD);
     call zeromatrix(mu0);
     V[1]= 1e-8;
     V[2]= 1;
  endcnst;

  %macro loop;
     %do k = 1 %to %eval(&p);
        DRD[&k,&k]=V[i&k+1];
        parms i&k;
        prior i&k~binary(.5);
  %end;
  %mend loop;
  %loop;

  parms beta 0;
  prior beta ~ mvn(mu0, DRD);
  parms sigma2 1;
  prior sigma2 ~ igamma(shape = .1, iscale = .1);

  call mult(beta, X, mu);

  model logSalary ~ n(mu, var = sigma2);
run;
```

The PROC MCMC statement invokes the procedure and provides options that enable you to control the simulation. The NMC= option requests that 2,000 MCMC samples be generated. The SEED= option sets the seed for the pseudorandom number generator and ensures reproducibility. The PROPCOV= option specifies that a quasi-Newton optimization method be used in constructing the initial covariance matrix for the Metropolis-Hastings algorithm, and the NTU= option requests 1,000 tuning iterations. The OUTPOST= option requests that the posterior samples of parameters be saved in the data set Outpost. The MONITOR= option requests that posterior analyses (such as plotting, diagnostics, and summaries) be performed on the symbols i1 through i&p; these are the indicator variables in $\boldsymbol{\gamma}$. There is no need to perform posterior analyses on the other model parameters, because the Markov chains are not expected to converge and no statistical inference is to be made by using the parameter estimates. The DIAG=NONE option suppresses all diagnostic tests and statistics. Because you are not fitting a single model, these statistics are not interpretable and in some cases cannot even be computed.

The first block of statements consists of five ARRAY statements that specify the dimensions of the matrices to be used in SSVS. The array **DRD** represents the covariance matrix $\mathbf{D}_\gamma\mathbf{R}\mathbf{D}_\gamma$ of equation (4). The array **V** holds the values for v_0 and v_1 from equation (5). The array **Mu0** represents the mean vector for the prior distribution of $\boldsymbol{\beta}|\boldsymbol{\gamma}$ of equation (4). The arrays **X** and **Beta** represent the design matrix and the parameter vector $\boldsymbol{\beta}$, respectively, and are used to compute the mean of the univariate normal distribution of equation (1).

The next block of statements, which begins with the BEGINCNST statement and ends with the ENDCNST statement, sets the initial values for the arrays **DRD**, **V**, and **Mu0**. The array **DRD** is initialized as an identity matrix. In later statements, the diagonal of **DRD** is altered. All elements of the array **Mu0** are initialized to 0. The first element of the array **V** contains v0 and is set to 1e-8; the second element of **V** contains v1 and is set to 1. SSVS is particularly sensitive to the values that are chosen for v_0 and v_1; finding acceptable values

often requires experimentation.

The macro %LOOP defines the diagonal elements of the array **DRD** and specifies the PARMS and PRIOR statements for the parameters i1–i&p (which represent the parameters γ_i in equation (2). The parameters i1–i&p have binary prior distributions and are placed in their own parameter block. This means that those parameters do not require any tuning; the inverse-CDF method applies. When you place these parameters in their own blocks, the computational costs increase linearly with the number of parameters, because each block of Metropolis parameters requires one additional pass through the data set; but the benefits are faster convergence and better mixing. This macro must be placed outside the BEGINCNST and ENDCNST statement block, because the assignment of the diagonal elements of **DRD** depends on the unknown parameters i1–i&p.

The next block of statements specifies the PARMS and PRIOR statements for the parameters Beta1–Beta&p (which represent $\boldsymbol{\beta}$) and the parameter Sigma2 (which represents the residual variance σ^2 in equations (1) and (7)). The first PARMS statement requests that the parameters Beta1–Beta&p be evaluated in a single parameter block and that a starting value of 0 be used for each parameter. Because these parameters are to have a multivariate prior distribution, you specify them as an array in the PARMS statement rather than specifying the name of each individual parameter. The subsequent PRIOR statement specifies that the parameters in the array **Beta** have a multivariate normal prior distribution whose mean is represented by the array **Mu0** and whose covariance matrix is represented by the array **DRD**. The second PARMS statement requests that the parameter Sigma2 be evaluated in its own parameter block and specifies a starting value of 1. The subsequent PRIOR statement specifies that Sigma2 have an inverse gamma prior distribution whose shape and iscale parameters both equal 0.1. The values of the shape and scale parameters imply that ν and λ in equation (7) equal 0.2 and 100, respectively.

Next, the CALL statement computes $\mu = \mathbf{X}\boldsymbol{\beta}$, the mean of the univariate normal distribution of equation (1).

Finally, the MODEL statement specifies that LogSalary be normally distributed, with mean equal to Mu and variance equal to Sigma2.

The following macro %TABLES uses the FREQ procedure to tabulate the frequencies of the models that the Markov chain visited. The tabulated results are then sorted by frequency, and the models that have the highest frequencies are printed. The macro is written specifically for this example, but you could easily generalize it. Specifically, the macro assumes the existence of the global macro variables &_TrgInd and &p, and it assumes that the posterior samples that the MCMC procedure produces are saved in a data set named Outpost.

```
%macro tables;
  %let tables=i1;
  %do k = 2 %to &p;
    %let tables = &tables*i&k;
  %end;


  ods select none;
  proc freq data=outpost;
     format i: 1.;
     tables &tables /
            list nocum;
     ods output List=Models(drop=table);
  run;

  proc sort data=Models out=Models;
    by descending frequency;
```

```
 run;

data models;
   set models;
   %do j=1 %to &p;
     %let n = %scan(&_trgind,&j);
     %let i=i&j;
     label i&j = "&n";
   %end;
 run;

 ods select all;
 proc print data=Models(where=(percent>5)) noobs label heading=v;
   format i: 1.;
   var i1-i&p frequency percent;
 run;

%mend tables;


%tables
```

Output 1 shows that the model that has the highest frequency was visited 99.75% of the time and contains the variables DivisionEast, LeagueAmerican, nHits, nHome, nBB, YrMajor, and nError.

Output 1 Most Frequently Visited Models

	d i v i s i o n E a s t	l e a g u e A m e r i c B a t t e r	n A t H i t s	n	n	n	R	n	y r M a j o r	c r A t B a t B	c r H i t s	c r H i m e s	c r R u n s	c r b i	n c O u t s	n A s s	n E r o r	F r e q u e n c y	P e r c e n t
1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	1995	99.75

An alternative to the previous model is to treat v_0 and v_1 as hyperparameters, assign prior distributions, and estimate their values. You do this by adding two PARMS statements and two PRIOR statements. In the following statements, v0 is assigned a uniform(0,1) prior, and v1 is assigned a uniform(0,25) prior. As a result of this change, the array **V** can no longer be processed inside the BEGINCNST/ENDCNST statement block. Everything else matches in the previous example.

```
proc mcmc data=design nmc=2000 seed=194735 propcov=quanew ntu=1000
          outpost=outpost monitor=(i1-i&p) diag=none plots(unpack)=trace;

  array DRD[&p,&p];
  array V[2];
  array mu0[&p];
  array X[&p] &_trgind;
  array beta[&p] beta1-beta&p;

  begincnst;
     call identity(DRD);
     call zeromatrix(mu0);
  endcnst;

  parms s0;
  parms s1;
  prior s0~uniform(0,1);
  prior s1~uniform(0,25);
  V[1]=s0;
  V[2]=s1;

  %macro loop;
     %do k = 1 %to %eval(&p);
        DRD[&k,&k]=V[i&k+1];
        parms i&k;
        prior i&k~binary(.5);
  %end;
  %mend loop;
  %loop;

  parms beta 0;
  prior beta ~ mvn(mu0, DRD);
  parms sigma2 1;
  prior sigma2 ~ igamma(shape = .1, iscale = .1);

  call mult(beta, X, mu);

  model logSalary ~ n(mu, var = sigma2);
run;


%tables
```

Output 2 shows that the model that has the highest frequency was visited 67.45% of the time and contains the variables DivisionEast, LeagueAmerican, nHits, nHome, nBB, YrMajor, and nError. The model that has the second highest frequency was visited 6.05% of the time and adds the variable nRBI to the highest-frequency model.

Output 2 Most Frequently Visited Models

divisionEast	leagueAmerican	nAtBat	nHits	nHmRun	nRuns	nRBI	nBB	yrMajor	crAtBat	crHits	crHmRun	crRuns	crRBI	crBB	nOuts	nAssts	nErrors	Freque ncy	Percent
1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	1349	67.45
1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	1	121	6.05

Another alternative is to set the prior correlation matrix **R** equal to $(\mathbf{X}'\mathbf{X})^{-1}$. This requires a little more programming than the previous two examples. First you need to decide how to compute $(\mathbf{X}'\mathbf{X})^{-1}$. SAS offers a number of options. You can compute this matrix directly in PROC MCMC, you can compute it by using another SAS/STAT procedure, or you can use SAS/IML. The following example shows how you can use the GLM procedure with the INVERSE option in the MODEL statement to compute and save the bordered $(\mathbf{X}'\mathbf{X})^{-1}$ matrix. The border is then eliminated in a subsequent DATA step. After you save $(\mathbf{X}'\mathbf{X})^{-1}$ in a data set, PROC MCMC enables you to import it to an array.

In the following statements, the GLM procedure saves the bordered $(\mathbf{X}'\mathbf{X})^{-1}$ matrix in the data set InvXPX:

```
proc glm data=design;
   model logsalary = &_trgind / noint INVERSE;
   ods output InvXPX=InvXPX;
run;


data InvXPX(drop=Parameter logSalary);
  set InvXPX(where=(Parameter ne 'logSalary') );
run;
```

In order to use $(\mathbf{X}'\mathbf{X})^{-1}$ as the prior covariance matrix in PROC MCMC procedure, in addition to the arrays **DRD**, **V**, **Mu0**, **X**, and **Beta**, which were used in the previous two examples, you must create three additional arrays: **D**, **R**, and **DR**.

You declare the array **R** to be a dynamic array by specifying a dimension of 1 and also specifying the NOSYMBOLS option. You declare both **D** and **DR** to be $p \times p$ dimensional arrays. Then you use the READ_ARRAY function (within the BEGINCNST/ENDCNCST statement block) to read the data in InvXPX into the array **R**, and you initialize **D** as an identity matrix. You then replace the diagonal elements of **D** inside the macro %LOOP.

The next block of statements completes the creation of the covariance matrix **DRD** by performing two matrix multiplication operations. These operations are placed inside a BEGINNODATA/ENDNODATA statement

block so that PROC MCMC executes the operations only twice: at the first and last observations of the data set.

```
proc mcmc data=design nmc=2000 seed=194735
          propcov=quanew ntu=1000 maxtune=100
          outpost=outpost monitor=(i1-i&p) diag=none plots(unpack)=trace;

   array R[1] / nosymbols;
   array D[&p,&p];
   array DR[&p,&p];
   array DRD[&p,&p];
   array V[2];
   array mu0[&p];
   array X[&p] &_trgind;
   array beta[&p] beta1-beta&p;

   begincnst;
     rc = read_array("InvXPX", R);
     call identity(D);
     call zeromatrix(mu0);
     V[1]= .01;
     V[2]= 1;
   endcnst;

   %macro loop;
      %do k = 1 %to %eval(&p);
        D[&k,&k]=V[i&k+1];
        parms i&k;
        prior i&k~binary(.5);
      %end;
   %mend loop;
   %loop;

   beginnodata;
     call mult(D,R,DR);
     call mult(DR,D,DRD);
   endnodata;

   parms beta 0;
   prior beta ~ mvn(mu0, DRD);
   parms sigma2 1;
   prior sigma2 ~ igamma(shape = .1, iscale = .1);

   call mult(beta, X, mu);

   model logSalary ~ n(mu, var = sigma2);
run;


%tables
```

Output 3 shows that the model that has the highest frequency was visited 63.90% of the time and contains the variables DivisionEast, LeagueAmerican, nAtBats, nHits, nRuns, nBB, YrMajor, crAtBats, crHome, crBB,

nAssts, and nError. The model that has the second-highest frequency was visited 24.40% of the time and adds the variable crHits to the highest-frequency model.

Output 3 Most Frequently Visited Models

division	league	nAtBats	nHits	nHmRuns	nRuns	nRBI	nBB	yrMajor	crAtBat	crHits	crHmRun	crRuns	crRBI	crBB	nOuts	nAssts	nError	Freq	Percent
1	1	1	1	0	1	0	1	1	1	0	1	0	0	1	0	1	1	1278	63.90
1	1	1	1	0	1	0	1	1	1	1	1	0	0	1	0	1	1	488	24.40
1	1	1	1	0	1	0	1	1	1	0	1	0	1	1	0	1	1	126	6.30

References

Chipman, H., George, E. I., McCulloch, R. E., Clyde, M., Foster, D. P., and Stine, R. A. (2001), "The Practical Implementation of Bayesian Model Selection," *Institute of Mathematical Statistics Lecture Notes—Monograph Series*, 38, 65–134.

Collier Books (1987), *The 1987 Baseball Encyclopedia Update*, New York: Macmillan.

George, E. I. and McCulloch, R. E. (1993), "Variable Selection via Gibbs Sampling," *Journal of the American Statistical Association*, 88, 881–889.

Time Inc. (1987), "What They Make," *Sports Illustrated*, April, 54–81.