Some SAS macros for BUGS data

statistics for discrete variables with PROB= or continuous variables with MEAN= and PREC= to include in the data file, by default, or the init file, if the INIT= option is specified. Here's a code snippet:

```
%_lexport(data=mta, file=mta.txt, init=mta.in, var=med beh com age,
    close=0, initclose=0, center=age, prob=med beh com);
%_sexport(data=mta, append=mta.txt, initappend=mta.in,
    var=pscale0-pscale3 tscale0-tscale3, prob=pscale0-pscale3 tscale0-tscale3);
```

After generating posterior samples with BUGS, we want to create SAS datasets from our CODA files. Here's a code snippet:

```
*select device and graphics file name;
goptions device=psl gaccess=gsasfile;
filename gsasfile 'chains.ps';

*import CODA files and generate univariate statistics;
*NOT recommended: for comparison and completeness only;

*one chain at a time: chain 1;
%coda2sas(out=post1, infile=mta.ind, chain=mta1.out, stats=1);
*one chain at a time: chain 2;
%coda2sas(out=post2, infile=mta.ind, chain=mta2.out, stats=1, gsfmode=append);

data post;
    set post1 post2;
run;
```

The CODA2SAS macro was originally written with BUGS in mind. It can process CODA files, but doesn't handle multiple chains automatically. You have to import each of the chains manually. Set INFILE= to the name of your index file and CHAIN= to the name of your chain file. So, I imported the first chain into the SAS dataset post1. This SAS dataset contains three variables c_1-c_3 which correspond to the monitored array c[] where c[1] was translated as c_1, etc. But first, I set my graphics output file with a FILENAME GSASFILE statement and my graphics device with a GOPTIONS DEVICE= statement. This is necessary with a summary request (STATS=1); statistics and kernel density plots with histograms are generated. Without FILENAME/GOPTIONS statements, the plots will be displayed on your graphics display device. However, if you choose a device that is a graphics file type supporting multiple images (like PostScript or PDF), then a graphics file is generated. If you want more graphics output appended to the same graphics file in a subsequent request, specify GSFMODE=APPEND. If you choose a device that is a graphics file type that supports only single images (like Encapsulated PS or JPEG), then you need to specify TYPE=. For example, if you selected the JPEG device type and specified TYPE=jpg, then the following graphics files are generated: c_1.jpg, c_2.jpg and c_3.jpg.

```
*select device and graphics file name;
```

```
goptions device=psl gaccess=gsasfile;
filename gsasfile 'chains.ps';

*import CODA files and generate univariate statistics;
*this is recommended the way;

*all chains at once;
%_decoda(out=post3, infile=mta.ind, chains=2, var=c, mu0=1);
```

Next, I read in both chains with the _DECODA macro. Name your index file either NAME.ind, NAMEIndex.txt or NAME.ind.txt and each of your chains NAME#.out, NAME#.txt or NAME#.out.txt respectively, where # is the number of your chains (1-2 in this example). If you follow this naming convention, then an unlimited number of chains are supported by specifying only INFILE= for the index file and CHAINS= for the number of chains. If the chain files follow some other naming convention, then you can specify each of them manually as CHAIN1= up to CHAIN10=. The INFILE= and OUT= parameters are required as well as either CHAINS= or CHAIN1=, etc. The posterior samples from both chains are contained in the SAS dataset post3.

The same CODA2SAS comments apply to _DECODA with respect to GFSMODE= and TYPE= (although the syntax is the same for TYPE=, if the variable of interest is a monitored array, stick with CODA2SAS). Although STATS= is still accepted, VAR= is the new recommended name of the option which is more SAS-ish. If one or more SAS variable names contained in the SAS dataset created is/are provided as arguments via VAR=, then only those SAS variables are summarized. If you want all SAS variables summarized, then you specify that the SAS way as VAR=_all_ instead of STATS=1. And, note that the syntax has changed. Instead of VAR=c_1-c_3, you specify VAR=c. This is more intuitive since that was the name of the monitored array, c[]. But, this requires the introduction of the SAS variable OBS into the OUT= SAS dataset that represents the element of the array, i.e. 1-3. If the monitored variable is not an array, then OBS=0. In addition, more summaries are available. Tests and tables for location are performed and the default location of 0 can be changed with MU0=. If you specify AUTOCORR=1 or set NLAG= to something other than 25, auto-correlations are generated

```
*transform data so that each record contains all 3 variables;
proc transpose data=post3 out=post prefix=c;
    where 1<=obs<=3;
    by chain iter;
    id obs;
    var c;
run;

*produce simultaneous confidence intervals;
%bayesintervals(data=post, vars=c1-c3, tail=U);
```

Lastly, I used the SAS macro BAYESINTERVALS by RD Wolfinger which is available at http://ftp.sas.com/samples/A56648. It constructs simultaneous intervals of the posterior for c1-c3.

_CEXPORT

Also available at the same URL is BAYESTESTS by PH Westfall which is for Bayesian multiple

_LSXPORT]TJ/F20 11.9552 Tf 6.54490 Td [(_LEMPORT)]TJ/F17 11.9552 Tf 5.45478 3

'

hypothesis testing (BAYESTESTS requires an ESTIMATE macro which you can construct by hand or which can be created by MAKEGLMSTATS by RD Tobias which is also provided).

There are four other SAS macros that you might find useful: _DEBUGS, SAS2CODA, _LIMPORT and _CEXPORT. _DEBUGS is similar to _DECODA except that it only provides the summaries, it does not read CODA files so you have to specify DATA=. Also, _DEBUGS allows you to create a subset of the data with OUT= and THIN= and/or WHERE=. SAS2CODA reads a SAS dataset and creates CODA files. _LIMPORT creates a SAS dataset from a \list" file with two required options, OUT= for the new SAS dataset and INFILE= for the \list" file. However, importing can be tricky so a temporary SAS/IML program (which you can name with FILE=) is created and run automatically. If the importing fails, then you should be able to make corrections to the program and run it manually. _CEXPORT (with similar syntax to _LEXPORT) reads a SAS dataset and creates a Comma Separated Values (CSV)

_LEXPORT

'

_DEBUGS

'

only tt a

cbtpy0nidedm(prog