SUPPLEMENTARY MATERIALS FOR:

'Jointness in Bayesian Variable Selection with Applications to Growth Regression'

Eduardo Ley
The World Bank, Washington DC, U.S.A.

Mark F.J. Steel
Department of Statistics, University of Warwick, U.K.

Version: February 22, 2008

Abstract. This document contains some supplementary graphs, and computational notes to accompany our paper E. Ley and M.F.J. Steel (2007), "Jointness in Bayesian Variable Selection with Applications to Growth Regression," *Journal of Macroeconomics*, 29(3): 476–493.

Address. Eduardo Ley, The World Bank, Washington DC 20433. Email: eduley@gmail.com. Mark F.J. Steel, Department of Statistics, University of Warwick, Coventry CV4 7AL, U.K. Vox: +44(0)24 7652 3369. Email: M.F.Steel@stats.warwick.ac.uk

1. Overview

This document contains some supplementary materials for the paper E. Ley and M.F.J. Steel (2007), "Jointness in Bayesian Variable Selection with Applications to Growth Regression," *Journal of Macroeconomics* (LS henceforth).

Files in `*.zip` archive:

File	**Directory**	**Contents**	**Format**	**See**
`0readme.pdf`	`ls5/`	This file	PDF	-
`ls5bma.f`	`ls5/code/`	f77 source file	Text	Section on Fortran code
`ls5bma.par`	`ls5/code/`	Parameter file	Text	Section on Parameter file
`grk41t72.dat`	`ls5/code/`	FLS data file	Text	Section on Data
`grk67t88.dat`	`ls5/code/`	SDM data file	Text	Section on Data
`k41i9_Std_.out`	`ls5/out/k41/std/`	FLS standardized out file	Text	Section on Output Files
`k41i9_Std_bj.dat`	`ls5/out/k41/std/`	Bi-variate Jointness Measures	Text	Section on Output Files
`k41i9_Std_tj.dat`	`ls5/out/k41/std/`	Tri-variate Jointness Measures	Text	Section on Output Files
`k41i9_NStd.out`	`ls5/out/k41/nstd/`	FLS non-standardized out file	Text	Section on Output Files
`k67i9_Std_.out`	`ls5/out/k67/std/`	SDM standardized out file	Text	Section on Output Files
`k67i9_Std_bj.dat`	`ls5/out/k67/std/`	Bi-variate Jointness Measures	Text	Section on Output Files
`k67i9_Std_tj.dat`	`ls5/out/k67/std/`	Tri-variate Jointness Measures	Text	Section on Output Files
`k67i9_NStd.out`	`ls5/out/k67/nstd/`	SDM non-standardized out file	Text	Section on Output Files

2. Supplementary Figures and Tables for LS

Fig. 1 below displays the behavior of the measures of bivariate jointness for the SDM data, Fig. 2 from LS is also reproduced here for ease of comparison.

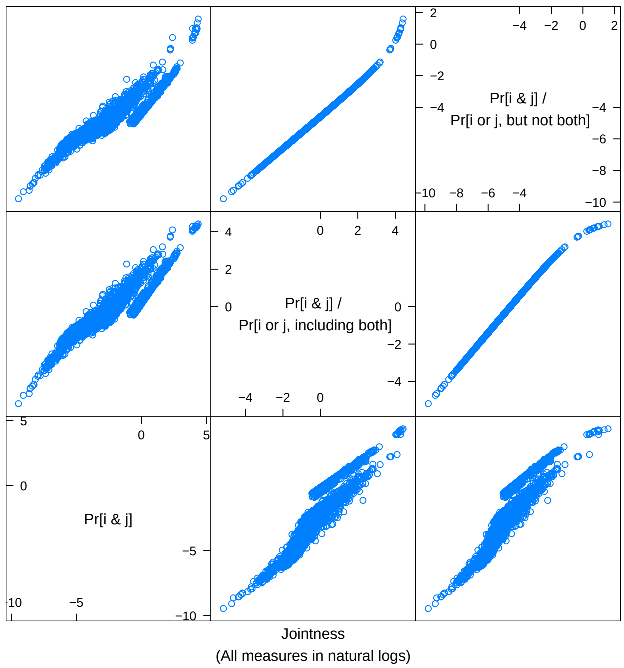


Fig 1. SDM data: Joint inclusion probabilities, bivariate jointness $\mathcal{J}^{\star}_{ij}$ and $\mathcal{J}_{ij}$.

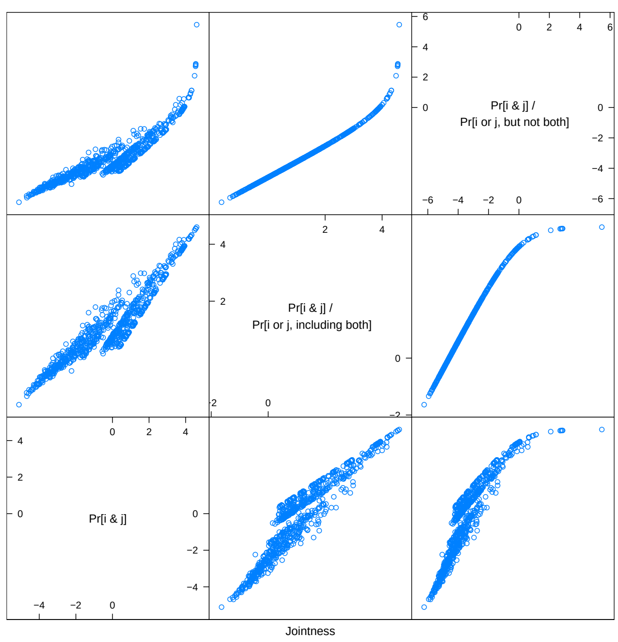


Fig 2. FLS data: Joint inclusion probabilities, bivariate jointness $\mathcal{J}^{\star}_{ij}$ and $\mathcal{J}_{ij}$.

The enclosed `k67i9_Std_.out` output file (described below) includes the complete Table 6 in LS for all 67 regressors. The `*NStd.out` files also include the results for non-standardized regressors for both the SDM and FLS datasets.

3. Code and Data Files

Overview: In order to reproduce the results in the paper, you'll need to compile the f77 file `ls5bma.f` and generate an executable, say, `bma.exe`. You'll need the data (`*.dat`) file and the parameter (`ls5bma.par`) file in the same directory—this file controls some options explained below.

The successful execution will always produce an output file `*.out`, and, if the jointness option (`dojoint`) is set to true, also two additional files containing the jointness measures for all pairs and triplets (see below).

3.1. Fortran code

The file `ls5bma.f` contains the **f77 source code** to produce the results in the paper. It is standard f77 code except for the time-date functions. (If you cannot link to the standard Unix (`-unixlib`) or VAX/VMS (`-vmslib`) libraries, either remove the calls to `wr_date()` and `wr_time()`, or empty the body of these subroutines so that they do nothing when they are called.)

There are several **parameters** that you may need to modify before compiling the code if you are going to make different runs than the ones included here:

`maxm` Specifies the maximum number of models expected to be visited, currently set to 300,000. If you run longer chains, or use different datasets, and find the chain visiting more than `maxm` models, you'll have to set this parameter to a larger value. The number of 'visits out' in the `*.out` file will be an upper bound on the increment needed to `maxm`, since some of these may be to repeated models.

`maxk` Maximum number of regressors in your dataset.

`maxn` Maximum number of observations in your dataset.

`maxnf` Maximum number of observations to be used for out-of-sample prediction, if `lpsloop` set to true in `ls5bma.par` file.

The program has been tested on different machines and with different compilers:

- **Absoft's Pro Fortran for MacOS X v9.0** on a G4 iMac. It is important to use static storage (`-s`). Other suggested options include: optimization (`-O3 -cpu:host`), fold to upper case (`-N109`), maximum internal handle (`-T 100000`), temporary string size (`-t102400`).
- **Intel Fortran Compiler 9.1 for MacOS** on an Intel Core 2 duo iMac, under MacOS X. You simply need to disable the compiler's inline expansion (`-Ob0`):

```
> ifort ls5bma.f -Ob0 -o bma.exe
```

- **Intel Fortran Compiler 10 for MacOS** on an Intel Core 2 duo iMac, under MacOS X; no directives required:

```
> ifort ls5bma.f -o bma.exe
```

  A typical run on a 2.16GHz Intel Core 2 duo iMac takes less than 30 minutes.

- **Compaq Visual Fortran 6.6** on a dual Xeon workstation running Windows XP. Full optimization settings can be used.

3.2. Parameter File

The `ls5bma.par` file controls some of the execution time parameters.

The first line sets the name of the different output files (the bivariate and trivariate `*.dat` files will have '`bj`' and '`tj`' appended to this name.

The second line must contain the *exact* name of the data file.

The rest of the parameters are fairly self-explanatory. (Also, check the routine `setup` in `ls5bma.f` to see what the parameters do.) For a description of prediction issues and the G&M convergence estimate refer to FLS, and random θ priors are discussed in LSb.

An example of a `ls5bma.par` file follows:

```
1  useanyname         OUT namefile, char10
2  grk41t72.dat       DAT namefile
3  -81665432          negative int to init random number generator
4  9                  int 1-9, choice for g, see computefj and [FLS]
5  100000             warmup draws
6  500000             chain draws
7  T                  standard --- standardise Xs?
8  F                  LPSloop --- do prediction?
9  10                 integer specifying nf
10 0.85d0             real taking care of sample split
11 T                  wrpost --- write posterior results
12 T                  dogm --- do G&M 2nd chain to assess convergence?
13 T                  dojoint --- produce jointness results?
14 T                  fixed theta --- F means random theta, see [LS]
15 20.5               prior expected model size
```

3.3. Data File

We enclose the two data files used in LS; FLS has $k = 41, t = 72$, and SDM has $k = 67, t = 88$. As noted, the exact name of the data file to use at execution must be specified in the second line in `ls5bma.par`.

The data file must have the following structure.

Line 1:	The first line must have the number of observations (`ntot`).
Line 2:	The second line the number of regressors (`kreg`).
Lines 3 to (`kreg` + 2):	Then there must follow `kreg` lines with the corresponding variable names.
Lines (`kreg` + 3) to (2 × `ntot` + 2):	Next, another `ntot` lines, each with the values of the dependent variable, y, and the `kreg` regressors, X, in free format—*i.e.*, each with a total of `kreg` + 1 columns.

We show the listing of the first 48 lines of `grk41t72.dat` below. (Note that, to save space, lines from 44 onwards, each containing a ($kreg + 1$) array of data for each observation, are shown here truncated, without wrapup.)

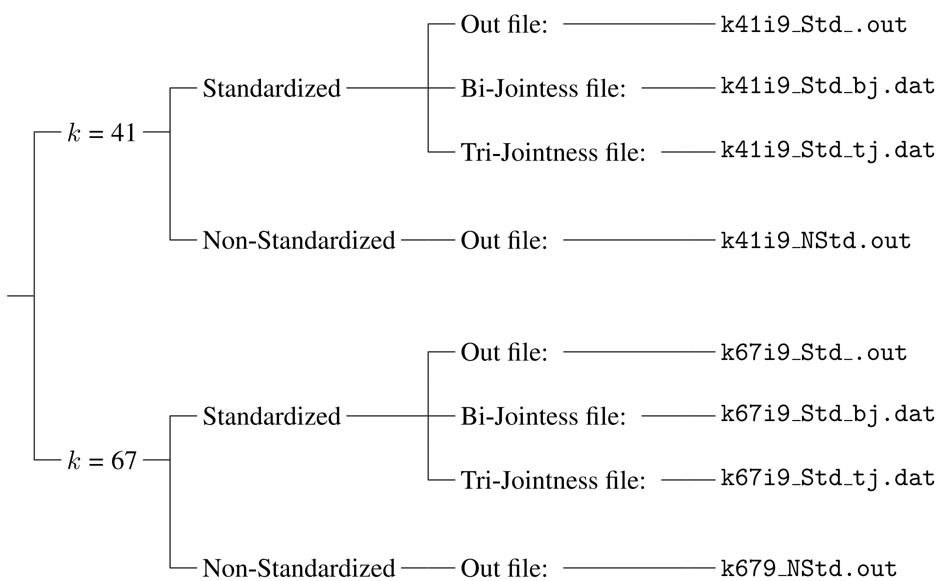
```
1  72
2  41
3  GDPsh560
4  Confuncious
```

```
Life Exp
Equip Inv
SubSahara
Muslim
Rule of Law
Yrs Open
Eco Org
Protestants
NEquip Inv
Mining
LatAmerica
PrSc Enroll
Buddha
Bl Mkt Pm
Catholic
Civl Lib
Hindu
Pr Exports
Pol Rights
R FEX Dist
Age
War Dummy
English %
Foreign %
Lab Force
Spanish Col
EthnoL Frac
std(BMP)
French Col
Abs Lat
Work/Pop
High Enroll
Pop g
Brit Col
Outwar Or
Jewish
Rev & Coup
%Publ Edu
Area
1.36900 7.438972 0.000 47.30 0.05070 0.0 0.990 0.3333 0.000 0.0 0.005 0.19070 0.196 0.0 0.46 0.00 0.131 0.000 5.88
5.61950 6.284134 0.000 45.70 0.13090 1.0 0.000 0.8333 0.356 5.0 0.250 0.15300 0.533 0.0 0.42 0.00 0.072 0.250 3.00
1.22390 6.546785 0.000 43.40 0.00400 1.0 0.160 0.5000 0.156 5.0 0.170 0.12500 0.088 0.0 0.65 0.00 0.050 0.160 5.55
2.23220 6.968851 0.000 47.30 0.04980 1.0 0.020 0.3333 0.000 1.0 0.250 0.23840 0.168 0.0 0.78 0.00 0.050 0.250 6.16
-0.05530 5.517453 0.000 42.20 0.00600 1.0 0.420 0.5000 0.000 0.0 0.000 0.04880 0.001 0.0 0.07 0.00 0.156 0.000 6.6
```

```
49 ...
```

3.4. Output Files

For each run, we produce three output files. The `*.out` file contains the output of the chain, while the two `*.dat` files contain the different bi- or tri-variate jointess measures. (These two files are only generated when `dojoint` in the `ls5bma.par` file is set to true (`T`).)



4. Updating of the code to handle $k > 52$

A model is represented by an array of 0s and 1s: $m = (m_1, m_2, \ldots, m_k)$ with $m_i \in \{0, 1\}$. An index $idx \in \{1, 2, \ldots, 2^k\}$ tracks the 2^k possible models.

When idx is a `real*8` variable, it can count only to 2^{52}. This means that the loop below will print 53 — same result will be obtained in GAUSS and Matlab.

`initiliaze` $(d = 1, i = 1)$

`while` $(d > 0)$ `do`

  $i \leftarrow (i + 1)$

  $x_1 \leftarrow 2^i$

  $x_2 \leftarrow (1 + 2^i)$

  $d \leftarrow (x_2 - x_1)$

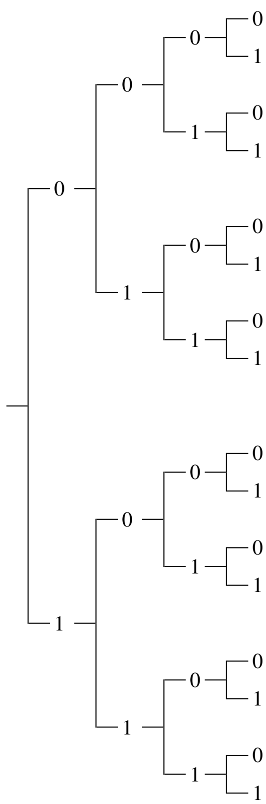
`enddo`

`print` i

Thus, we cannot distinguish between 2^{52} and $1 + 2^{52}$

To get around this problem, we can use 2 indices to keep track of models — idx_1 keeps track of the first 52 vars and idx_2 keeps track of the remaining $k - 52$ vars. Now w can handle up to 104 vars, or 2^{104} models.

Example: 4 variables; 2^4 = 16 models. If we could only count to 4, we could just use 2 indexes, idx_1 and idx_2, each in $\{1, 2, 3, 4\}$ to account for the 16 models indexed by idx.



[0000] $(idx_1, idx_2) = (1, 1) \to idx = 1$
[0001] $(idx_1, idx_2) = (1, 2) \to idx = 2$
[0010] $(idx_1, idx_2) = (1, 3) \to idx = 3$
[0011] $(idx_1, idx_2) = (1, 4) \to idx = 4$
[0100] $(idx_1, idx_2) = (2, 1) \to idx = 5$
[0101] $(idx_1, idx_2) = (2, 2) \to idx = 6$
[0110] $(idx_1, idx_2) = (2, 3) \to idx = 7$
[0111] $(idx_1, idx_2) = (2, 4) \to idx = 8$
[1000] $(idx_1, idx_2) = (3, 1) \to idx = 9$
[1001] $(idx_1, idx_2) = (3, 2) \to idx = 10$
[1010] $(idx_1, idx_2) = (3, 3) \to idx = 11$
[1011] $(idx_1, idx_2) = (3, 4) \to idx = 12$
[1100] $(idx_1, idx_2) = (4, 1) \to idx = 13$
[1101] $(idx_1, idx_2) = (4, 2) \to idx = 14$
[1110] $(idx_1, idx_2) = (4, 3) \to idx = 15$
[1111] $(idx_1, idx_2) = (4, 4) \to idx = 16$

Counting—Notice that the first $2^{k-1} = 2^3 = 8$ integers are associated with vectors that have with $m_1 = 0$. Start out with $i = 1$, then a 1 in m_1 adds $2^{k-1} = 8$ to idx, a 1 in m_2 adds $2^{k-2} = 4$ to idx, ..., a 1 in m_k simply adds $2^0 = 1$ to idx.

The function `getmodidx(k,m)` starts with $idx = 1$ and makes $idx = idx + zm_i$ for $z = 2^{k-1}$ down to 1. Now it will have to be replaced by a subroutine with 2 calls to the old FLS function, each call taking care of a portion of the binary array m.

```
c
c**********************************************
      real*8 function getmodidx(k,m)
c**********************************************
c
c inputs:
```

```
c     k.le.52.......is the number of all possible regressors
c     m(k)....contains 0s or 1s in each coordinate
c output:
c     getmodidx..is the real*8 associated with the state vector
c
      real*8 idx,z
      integer  k,i,m(k)

        z = 2.0d0**(k-1)
        idx = 1
        do 10 i=1,k
         idx = idx + z*m(i)
         z = z/2.0d0
 10     continue
        getmodidx = idx
        return

      end
c
c
c**********************************************
      subroutine get2modidx(k,m,idx1,idx2)
c**********************************************
c
c inputs:
c     k>52.......is the number of all possible regressors
c     m(1:k)....contains 0s or 1s in each coordinate
c output:
c     idx1..is the real*8 associated with the 1st 52-var state vector
c     idx2..is the real*8 associated with the 2nd 52-var state vector
c
      integer  i,k, m(k), m1(52), m2(52)
      real*8 idx1,idx2,getmodidx

      k1=min(52,k)
      do 10 i=1,k1
         m1(i)=m(i)
 10   continue
      idx1 = getmodidx(k1,m1)

      k2 = k-52
      if (k2.ge.1) then
         do 15 i=1,k2
            m2(i)=m(i+52)
```

```
 15        continue
           idx2 = getmodidx(k2,m2)
        else
           idx2=0.0d0
        endif

        return
        end
c
c***********************************************
```

The subroutine gmodel(idx,k,m) does the reverse—starting with idx constructs the binary array m representing the model. (If $idx > 2^{k-j}$ then m_j = 1, otherwise m_j = 0, for $j = 1, k-1$.) Now it will be replaced by another routine specifying 2 indexes: idx_1 and idx_2.

```
c***********************************************
      subroutine gmodel(idx,k,m)
c***********************************************
c
c  given the model index idx and the number of all possible regressors
c  k<53
c  it returns a binary array, m, of dimension k.  if the ith coordinate is 1
c  the ith regressor is included in the model.  otherwise it is excluded.
c
c inputs:
c     idx.....is the model index (real*8)
c     k.......is the number of all possible regressors k=1..52
c
c output:
c     m(k)...contains 0s or 1s in each coordinate
c
      integer i, k, m(k)
      real*8 idx,x,z

      x = 2.0d0**(k-1)
      z = idx

      do 20 i=1,k
         if (z.gt.x) then
            m(i) = 1
            z = z - x
           else
            m(i) = 0
         endif
         x = x / 2.0d0
```

```
 20   continue
      return
      end
c
c**********************************************
      subroutine g2model(idx1,idx2,k,m)
c**********************************************
c
c  given the model index idx and the number of all possible regressors k
c  it returns a binary array, m, of dimension k.  if the ith coordinate is 1
c  the ith regressor is included in the model.  otherwise it is excluded.
c
c inputs:
c     idx.....is the model index (real*8)
c     k>52....is the number of all possible regressors
c
c output:
c     m(k)...contains 0s or 1s in each coordinate
c
      integer  i,k,m1(52),m2(52),m(k)
      real*8 idx1,idx2

      k1 = min(k,52)
      call gmodel(idx1,k1,m1)
      do 10 i=1,k1
         m(i)=m1(i)
 10   continue

      k2 = k-52
      if (k2.ge.1) then
         call gmodel(idx2,k2,m2)
         do 15 i=1,k2
            m(52+i)=m2(i)
 15      continue
      endif

      return
      end
c
c**********************************************
```

5. References

[FLS] Fernández, Carmen, Eduardo Ley and Mark F.J. Steel (2001) "Model Uncertainty in Cross-Country Growth Regressions," *Journal of Applied Econometrics*, 16: 563–76.

[LS5] Ley, Eduardo and Mark F.J. Steel (2007) "Jointness in Bayesian Variable Selection with Applications to Growth Regression," *Journal of Macroeconomics* 29(3): 476–493.

[LS6] Ley, Eduardo and Mark F.J. Steel (2008) "On the Effect of Prior Assumptions in Bayesian Model Averaging with Applications to Growth Regression," *Journal of Applied Econometrics*, forthcoming.

[SDM] Sala-i-Martin, Xavier X., Gernot Doppelhofer and Ronald I. Miller (2004) "Determinants of Long-Term Growth: A Bayesian averaging of classical estimates (BACE) approach." *American Economic Review*, 94: 813–835.