# Computer programs for simulation-based estimation of peer effects

Brian Krauth
Simon Fraser University

Version 1.1 - March 1, 2005

## 1 Description of programs

This file is the documentation for a set of Fortran 90 computer programs I have written to implement the structural estimation method described in my paper "Simulation-based estimation of peer effects" [1]. References in this documentation refer to the July 30, 2004 draft of the paper.

The main programs in the distribution are:

- `smle:` Estimates structural model from an individual-based sample.
- `s2:` Estimates structural model from a group-based sample.
- `probit:` Estimates a standard (naive) probit model from an individual-based or group-based sample.
- `psim:` Generates simulated data from the model for use in Monte Carlo experiments.

This distribution[1] provides both Fortran 90 source code and executable programs for Windows.

## 2 Installation and basic use

To install, create a directory on your computer for the code and programs, then unzip the file `smle.zip` into that directory. Once that is done, there are two options.

1. The distribution provides (in the `windows-binaries` directory) executable files for Windows. They can be copied to any working directory and used there.
2. If you have a Fortran 90 compiler, the source code can be compiled directly. This allows for use under Linux and other operating systems, and allows for the modification of code when needed. See Appendix A for details.

To use any of the programs, just do the following:

1. Copy the executable to your work directory (i.e., the directory in which you want to put your input and output).
2. Create a parameter file in the work directory. The parameter file specifies user settings for various options.
3. Construct the data file and place in the work directory.

[1] An older version of the code is available for Gauss; however many features and options available in the Fortran version are not in the Gauss version. Email me to ask for a copy.

4. Run the program. Note that depending on the size of the data set and the user options selected, the execution time of `smle` and `s2` can vary from minutes to weeks.

5. The estimation results will be reported in a user-specified file.

The structure of the data file, parameter file, and output files varies by program. See below for details.

# 3 Using the `smle` program

The `smle` or `smle.exe` program is used to estimate the structural model described in the paper from an individual-based sample.

## 3.1 Data file

The program expects data in whitespace-delimited ASCII format, with no headers. Each row should contain a single observation, consisting of the following columns:

| **Column #** | 1 | 2 | 3 | 4+ |
|---|---|---|---|---|
| **Variable** | Respondent's Choice | Average Peer Choice | Number of Peers | Other Explanatory Variables |
| **Range** | $\{0, 1\}$ | $[0.0, 1.0]$ | $\{1, 2, 3, \ldots\}$ | $(-\infty, \infty)$ |

If you intend to treat some of the explanatory variables as aggregates, put them before the other explanatory variables.

For example, a data set with 2 explanatory variables and 3 observations might look like this:

```
1 1.000 4 -0.337  0.000
0 0.250 4 -1.734  1.000
0 0.400 5  0.532  0.000
```

## 3.2 Parameter file and user options

The parameter file for the `smle` program is named `parm.dat`. and is just a text file with a set of user options specified. It looks like this:

```
Parameter file - data is in fixed format; do not delete lines
Blank line, reserved for future use
Blank line, reserved for future use
NVAR (number of variables, positive integer)
1
NOBS (number of observations, positive integer)
1000
NUMAGG (number of variables that are aggregate, nonnegative integer)
0
NSIM (number of simulations used to calculate likelihood function)
100
RESTARTS (number of times to restart the search algorithm)
3
SIMULATOR_TYPE (GHK or HYBRID)
GHK
EQUILIBRIUM_TYPE (LOW, HIGH, RANDOM, BOUNDS, PLOT, MINIMUM BOUNDS)
LOW
UNDERREPORTING_CORRECTION (correct for underreporting, logical)
.false.
BOOTSTRAP (calculate bootstrap covariance matrix, logical)
.false.
LOAD_U (load random numbers from file rather than generate them, logical)
.false.
RHO_TYPE (X, Fixed, Interval, or Estimate)
X
FIXED_RHO (Used if RHO_TYPE=FIXED, real)
0.7
FIX_GAMMA (logical)
.false.
FIXED_GAMMA (Used if FIX_GAMMA=.true., real)
```

```
0.0
DATAFILE (Name of data file)
oneobs.txt
LOGFILE (Name of file to write log info)
lfile.log
RESULTFILE (Name of file to write results to)
smle.out
UFILE (Name of file to write random numbers to, or read them from)
testu.dat
BOOTFILE (Name of file to write bootstrap sample to, or read them from)
testboot.dat
FIXEDEFFECTS (Number of aggregate variables to treat as fixed effects)
0
```

The program is very primitive in how it reads the data from this file. In particular it looks on specific lines of the file for specific variables. For example, it will always set `DATAFILE` to whatever is in the first 12 characters of the 33rd line in the file. Even numbered lines are ignored, so they can be used for any comments one might want.

A detailed description of each line in this file is as follows:

1. `NVAR`: Number of exogenous explanatory variables in data set (integer).
2. `NOBS`: Number of observations in data set (integer).
3. `NUMAGG`: The first `NUMAGG` explanatory variables in the data set will be treated as aggregate variables (integer). Aggregate variables have the same effect on both the respondent and his or her peers. See Section 4.4 in the paper.
4. `NSIM`: Number of simulations to use in calculating the log-likelihood function (integer). The program uses randomized Halton sequences, which accurately approximate probabilities using far fewer simulations than standard random numbers. About `NSIM=100` seems to work well enough for a first pass.
5. `RESTARTS`: Number of times to run the Davidson-Fletcher-Powell (DFP) search algorithm (integer). At least `RESTARTS=3` is recommended to avoid finding a local rather than global optimum. If `RESTARTS=0`, the simulated annealing (SA) search algorithm will be used instead.
6. `SIMULATOR_TYPE`: Type of simulator to use in calculating normal rectangle probabilities. Options include (program ignores all but the first letter, not case sensitive)
   - `G(HK)`: Geweke-Hajivassiliou-Keane simulator. A slower but more accurate simulator that is currently only available in combination with `EQUILIBRIUM_TYPE=L`.
   - `H(ybrid)`: GHK-CFS hybrid simulator. A faster and more flexible simulator, but generates a discontinuous approximate likelihood function. It is recommended that the simulated annealing search algorithm be used if `SIMULATOR_TYPE = HYBRID`.
7. `EQUILIBRIUM_TYPE`: Equilibrium selection rule assumed (character). See Sections 2.3 and 4.2 in the paper for details. Options include (program ignores all but the first letter, not case sensitive):
   - `L(ow)`: Low-activity equilibrium
   - `H(igh)`: High-activity equilibrium
   - `R(andom)`: Random equilibrium
   - `B(ounds)`: Find selection-rule-free bounds on $\gamma$ using the likelihood bounds method.
   - `P(lot)`: Calculate selection-rule-free bounds on the likelihood function for plotting.
   - `M(inumum)`: Find selection-rule-free bounds (lower bound only) on $\gamma$ using the likelihood bounds method.
8. `UNDERREPORTING_CORRECTION`: Indicates whether or not to correct for underreporting (logical). See Section 4.3 in the paper for details.

9. BOOTSTRAP: Indicates whether to estimate the model once from the original sample (BOOTSTRAP = .FALSE.) or 100 times from a series of bootstrap resamples (BOOTSTRAP = .TRUE.).

10. LOAD_U: Indicates whether to use the internal random number generator to produce random numbers, or to load from the file specified as UFILE (logical).

11. RHO_TYPE: Rule for treating the within-group correlation in unobservables $\rho_\epsilon$ See Sections 2.4 and 4.1 in the paper for details. Options include:

    - X (recommended): Assume $\rho_\epsilon = \rho_x$.
    - F(ixed): Fix $\rho_\epsilon$ at the value of FIXED_RHO specified below.
    - E(stimate): Estimate $\rho_\epsilon$ directly (not recommended if FIX_GAMMA=.true.).
    - I(nterval): Estimate $\hat{\gamma}(\rho_\epsilon)$ function described in Section 4.1 of the paper.

12. FIXED_RHO: Value at which to fix $\rho_\epsilon$. Ignored if RHO_TYPE $\neq$ FIXED.

13. FIX_GAMMA: Indicates whether the value of $\gamma$ should be fixed rather than estimated (logical). See section 2.4 in the paper for details.

14. FIXED_GAMMA: Value at which to fix $\gamma$ (real). Ignored if FIX_GAMMA = .false..

15. DATAFILE: Name of file from which data will be read (up to 12 characters).

16. LOGFILE: Name of file to which log data will be written (up to 12 characters). File will be overwritten.

17. RESULTFILE: Name of file to which estimation results will be written (up to 12 characters). Results will be appended to file.

18. UFILE: Name of file in which random numbers are stored (up to 12 characters, case sensitive). If LOAD_U = .TRUE., then the random numbers will be read from this file. If LOAD_U = .FALSE., then the random numbers will be written to this file.

19. BOOTFILE: Same as UFILE, except the BOOTFILE is where information on the bootstrap sample is stored/loaded (up to 12 characters, case sensitive).

20. FIXEDEFFECTS: Usually this should be zero, or it can be left out of the file entirely. Normally, the underreporting correction assumes a constant reporting rate for all respondents. It is possible to condition the estimated reporting rate on one or more of the aggregate explanatory variables: the program will estimate the reporting rate conditional on the first FIXEDEFFECTS columns of explanatory variables in the data file.

## 3.3 Output files

The program is designed to run in the background, and does not write to standard output unless there is a problem. Several files are written out to the work directory while the program runs.

### 3.3.1 Results

Estimation results are appended to the file specified as RESULTFILE in parm.dat. The first number written to the file is the (maximized) log-likelihood. After that come the estimates $(\hat{\rho}_x, \hat{\rho}_\epsilon, \hat{\gamma}, \hat{\beta})$.

For example, with one explanatory variable, the RESULTFILE would look like this:

```
-1974.1364 0.11305 0.11305
-2.9774E-004 -1.38E-002 1.03477
```

Notice that the program does not keep the output from a single run on a single line, but rather breaks the line after about 3 numbers.

### 3.3.2 Results (interval estimation)

With the `RHO_TYPE = INTERVAL` option, the model is estimated under 12 different specifications, and the results are reported for each of these specifications, in the following order:

- Standard assumption, $\rho_\epsilon = \rho_x$, $\gamma$ to be estimated.
- $\gamma = 0$, $\rho_\epsilon$ to be estimated.
- $\rho_\epsilon = 0.0$, $\gamma$ to be estimated.
- $\rho_\epsilon = 0.1$, $\gamma$ to be estimated.
- $\vdots$
- $\rho_\epsilon = 0.9$, $\gamma$ to be estimated.

### 3.3.3 Results (likelihood bounds estimation)

Output for selection-rule-free estimation using the likelihood bounds approach is also different from the standard case.

If `EQUILIBRIUM_TYPE = BOUNDS`, the program writes out the estimated lower bound for $\gamma$, then the estimated upper bound. Bounds are not calculated or reported for the other parameters.

If `EQUILIBRIUM_TYPE = MINIMUM`, the program writes out only the estimated lower bound for $\gamma$.

If `EQUILIBRIUM_TYPE = PLOT`, the program calculates the approximate upper bound $H_g$ and lower bound $L_g$ on the log-likelihood function for $\gamma \in \{0.0, 0.1, \ldots, 4.0\}$ and writes $(\gamma, H_g, L_g)$. to the file. This can be used to construct a plot like that seen in Figure 2 of the paper.

### 3.3.4 Logging

As it runs, the program writs out detailed information on its operations to the file specified as `LOGFILE` in `parm.dat`.

### 3.3.5 Checkpointing

Because the program can potentially run for a very long time before producing the final data, "checkpointing" has been implemented.

Periodically while running the program saves a binary representation of its current state to the file `check.dat` and also a blank file called `check.lock`. These two files are both deleted on successful completion of the program. `check.lock` functions as a simple locking file - if you try to run the program in a directory that has a `check.lock` file, it will stop itself before doing much of anything. This is to avoid potential conflicts from having multiple instances of the program running in the same directory simultaneously.

If the program is interrupted, it can usually be restarted from the last checkpoint. To do this, one needs only to delete the file `check.lock`, and run the program as normally. The program will automatically search the work directory for the `check.dat` file and load it.

# 4 Using the s2 program

The `s2` program can be used to estimate the structural model from a group-based sample.

## 4.1 Data file

The program expects data in whitespace-delimited ASCII format, with no headers. Each row corresponds to an observation of an individual. The columns are as follows:

| **Column #** | **1** | 2 | 3+ |
|---|---|---|---|
| **Variable** | Group ID Number | Respondent's Choice | Explanatory Variables |
| **Range** | $\{0, 1, 2, \ldots\}$ | $\{0, 1\}$ | $(-\infty, \infty)$ |

If you intend to treat some of the explanatory variables as aggregates, put them before the other explanatory variables.

For example, with 2 explanatory variables and 5 observations the file might look like this:

```
1 1 -0.337  0.000
1 0 -1.734  1.000
1 0  0.532  0.000
2 1  0.536  1.000
2 1 -1.234  1.000
```

## 4.2 Parameter file and user options

The parameter file for the `s2` program is named `parm.dat`. and is just a text file with a set of user options specified. It looks like this:

```
Parameter file: Data is in fixed format; do not delete lines
DATAFILE: name of file where data is located
allobs.txt
RESULTFILE: name of file to which results should be appended
smle.out
LOGFILE: name of file to send logging information
lfile.log
NOBS: number of observations
1000
NVAR: number of exogenous explanatory variables
1
NUMAGG: number of exogenous explanatory variables that are aggregates
0
NSIM: number of simulations to use in calculating estimated loglikelihood
100
SEARCH_METHOD: DFP (Davidson-Fletcher-Powell) or SA (Simulated Annealing)
sa
RESTARTS: number of times to run search algorithm
2
EQUILIBRIUM_TYPE: equilibrium selection rule, either low, random, or high
Low
COVMAT_TYPE: method for calculating covariance matrix; either Hessian, OPG, or None
None
RHO_TYPE:
X
FIXED_RHO: value to fix rho_e at if FIX_RHO=.true.
0.7000
FIX_GAMMA: normally gamma is estimated, but it is fixed if this is .true.
.false.
FIXED_GAMMA: value to fix gamma at if FIX_GAMMA=.true.
0.0000
LOAD_U: .true. if you want random numbers loaded from UFILE, .false. if you want new random numbers
.false.
UFILE: name of file to which random numbers should be written (if LOAD_U=.false.) or read (if LOAD_U=.true.)
testu.dat
```

The description of each line in this file is as follows:

1. `DATAFILE`: Name of file from which data will be read (up to 12 characters, case sensitive).
2. `RESULTFILE`: Name of file to which estimation results will be written (up to 12 characters, case sensitive). Results are appended to this file.
3. `LOGFILE`: Name of file to which log data will be written (up to 12 characters, case sensitive). File will be overwritten.

4. NOBS: Number of observations in data set (integer).
5. NVAR: Number of exogenous explanatory variables in data set (integer).
6. NUMAGG: The first NUMAGG explanatory variables in the data set will be treated as aggregates (integer). See Section 4.4 in the paper for details.
7. NSIM: Number of simulations to use in calculating the log-likelihood function (integer).
8. SEARCH_METHOD: Optimization method to use. Options are:
   - S(imulated Annealing) (recommended): Use the simulated annealing search algorithm.
   - D(avidson-Fletcher-Powell): Use the DFP search algorithm

   Because the GHK-CFS hybrid simulator used in this program produces a discontinuous approximation to the log-likelihood function, the simulated annealing algorithm is recommended.
9. RESTARTS: Number of times to restart the DFP search algorithm, if applicable (integer).
10. EQUILIBRIUM_TYPE: Equilibrium selection rule assumed (character). See Sections 2.3 and 4.2 in the paper for details. Options include (program ignores all but the first letter, not case sensitive):
    - L(ow): Low-activity equilibrium.
    - H(igh): High-activity equilibrium.
    - R(andom): Randomly selected equilibrium.
    - B(ounds): Find selection-rule-free bounds on $\gamma$ using the likelihood bounds method.
    - P(lot): Calculate selection-rule-free bounds on the likelihood function for plotting.
    - M(inumum): Find selection-rule-free bounds (lower bound only) on $\gamma$ using the likelihood bounds method.
11. COVMAT_TYPE: Method for estimating covariance matrix of parameter estimates.
    - O(PG): Use outer product of gradients/BHHH method.
    - H(essian): Use inverse Hessian method. Although this method is available, it is not recommended as the small discontinuities in the approximated likelihood function lead to poor approximation of the Hessian.
    - N(one): Don't estimate covariance matrix.
12. RHO_TYPE: Rule for treating the within-group correlation in unobservables $\rho_\epsilon$ See Sections 2.4 and 4.1 in the paper for details. Options include:
    - X (recommended): Assume $\rho_\epsilon = \rho_x$. This is the baseline identifying assumption discussed in the paper.
    - F(ixed): Fix $\rho_\epsilon$ at the value of FIXED_RHO specified below.
    - E(stimate): Estimate $\rho_\epsilon$ directly. This is not recommended if FIX_GAMMA=.false., because $\rho_\epsilon$ is very weakly identified in this case.
    - I(nterval): Estimate $\hat{\gamma}(\rho_\epsilon$ function described in paper.
13. FIXED_RHO: Value at which to fix $\rho_\epsilon$ (real). Ignored if RHO_TYPE $\neq$ FIXED.
14. FIX_GAMMA: Indicates whether the value of $\gamma$ should be fixed rather than estimated (logical).
15. FIXED_GAMMA: Value at which to fix $\gamma$ (real). Ignored if FIX_GAMMA = .false..
16. LOAD_U: Indicates whether to use the internal random number generator to produce random numbers, or to load from a user-specified file (logical).
17. UFILE: Name of file in which random numbers are stored (up to 12 characters, case sensitive). If LOAD_U = .TRUE., then the random numbers will be read from this file. If LOAD_U = .FALSE., then the random numbers will be written to this file.

### 4.3 Output files

The output, logging, and checkpoint files for the `s2` program take almost the same form as described in Section 3.3 for the `smle` program. The only difference is that if `COVMAT=OPG` or `COVMAT=HESSIAN`, the estimated covariance matrix is reported as well.

## 5 Using the `probit` program

The `probit` program estimates a standard (naive) probit model, treating peer behavior as exogenous.

### 5.1 Data format

The program expects data in the same format as the `smle` program if data are from an individual-based sample, and in the same format as the `s2` program if data are from a group-based sample.

### 5.2 Parameter file and user options

The parameter file for the `probit` program is named `parm.dat`. and is just a text file with a set of user options specified. It looks like this:

```
Parameter file - data is in fixed format; do not delete lines
NVAR
1
NOBS
1000
SAMPLE_TYPE
Individual
UNDERREPORTING_CORRECTION
.false.
DATAFILE
oneobs.txt
LOGFILE
probit.log
RESULTFILE
probit.out
```

The description of each line in this file is as follows:

1. `NVAR`: Number of explanatory variables in data set (integer).
2. `NOBS`: Number of observations in data set (integer).
3. `SAMPLE_TYPE`: Format of input data. Options include:
   - `I(ndividual)`: Individual-based sample, as described in Section 3.
   - `G(roup)`: Group-based sample, as described in Section 4. Not yet implemented.
4. `UNDERREPORTING_CORRECTION`: Indicates whether or not to correct for underreporting (logical).
5. `DATAFILE`: Name of file from which data will be read (up to 12 characters, case sensitive).
6. `LOGFILE`: Name of file to which log data will be written (up to 12 characters, case sensitive).
7. `RESULTFILE`:Name of file to which estimation results will be written (up to 12 characters, case sensitive). Results are appended to this file.

### 5.3 Output files

Unlike `smle` and `s2`, the `probit` program only takes a second or two to run. As a result, there is no checkpointing.

The results of estimation are written to the file specified as `RESULTFILE`. The first number reported is the log-likelihood function, the second is the intercept, the third is the coefficient on peer behavior, and the remainder are coefficients on the other explanatory variables.

# 6 Using the psim program

The psim program generates simulated data from the model. It is useful in constructing Monte Carlo experiments.

## 6.1 Parameter file and user options

The parameter file for the psim program is named parmonte.dat. and is just a text file with a set of user options specified. It looks like this:

```
Parameter file: Data is in fixed format; do not delete lines
NGROUP: Number of groups to simulate
1000
MAXGROUPSIZE: maximum size of groups (right now all are same size)
5
NVAR: number of exogenous explanatory variables
1
NUMAGG: number of explanatory variables that are aggregate
0
EQTYPE: equilibrium type (low, high, or random)
Low
XTYPE: x type (Binary, or Normal)
N
B: coefficient vector, must be length NVAR+5
0.25 0.25 0.0 0.5 0.0 1.0
REPORTING_RATE
1.0
b2
0.0 0.00
```

The description of each line in this file is as follows:

1. NGROUP: Number of groups to simulate (integer).
2. MAXGROUPSIZE: Number of individuals per group (integer).
3. NVAR: Number of explanatory ($x$) variables (integer).
4. NUMAGG: The first NUMAGG explanatory variables will be treated as aggregates.
5. EQTYPE: Equilibrium selection rule. Options are:
   - L(ow): Low-activity equilibrium.
   - H(igh): High-activity equilibrium.
   - R(andom): Randomly selected equilibrium.
6. XTYPE: Allows for there to be non-normal explanatory variables. Options are:
   - N(ormal): $x$ is normally distributed (the usual assumption).
   - B(inary): $x$ is binary.
7. B: Coefficient vector, length NVAR+5. Order of elements is $(\rho_x, \rho_\epsilon, \gamma, \delta, \beta_0, \beta_1, \ldots, \beta_{\texttt{NVAR}})$. Note: $\delta$ is the contextual effect, if there is one.
8. REPORTING_RATE: This is a sequence of four real numbers, used to model inconsistent reporting. Let $r_{i,j}$ be the choice of person $i$ as reported by person $j$. The four numbers are, in order, $\Pr(r_{i,i} = 1|y_i = 0)$, $\Pr(r_{i,i} = 1|y_i = 1)$, $\Pr(r_{i,j} = 1|y_i = 0)$, and $\Pr(r_{i,j} = 1|y_i = 1)$). For example "0.0 1.0 0.0 1.0" will describe the base case of truthful reporting.
9. B2: This row should have two floating-point numbers. The first is the value of $corr(\beta \mathbf{x}_{gi}, \epsilon_{gi})$ and the second is the value of $corr(\beta \mathbf{x}_{gi}, \epsilon_{gj})$. Usually this will just be "0.0 0.0".

## 6.2 Output

The psim program will output two files, a data set that mimics an individual-based sample named oneobs.txt and a data set that mimics a group-based sample named allobs.txt. The files are ready to be used by smle and s2 respectively.

# A Compiling

Compiling is a matter of following these steps:

1. Procure a Fortran 90 compiler. A good reference for Fortran 90 is Metcalf and Reid's *Fortran 90/95 Explained*. The programs have been successfully compiled on Linux using the Portland Group PGF90 and PGHPF compilers, as well as the Intel IFC compiler. They have been successfully compiled on Windows using the free F compiler (The F language is a subset of Fortran) provided by The Fortran Group (`http://www.fortran.com`). The Fortran Group also provides free F compilers for Linux and other operating systems.

2. Unzip the `smle.zip` file into some appropriate directory. There will be one sub-directory for each major program, as well as library (`lib`) and documentation (`doc`) subdirectories

3. System-specific code is in a file named `lib/bklib.f90`. For example, this program includes subroutines that access the compiler's default random number generator. If you want to use a better random number generator (for example, from the NAG libaries) this file can be modified to do so.

4. To compile the `smle` program, for example, edit the batch file `compile_smle` (for Linux) or `compile_smle.bat` (for Windows):

   - Replace the call to "`f90`" or "`F`" with the name of your Fortran 90 compiler
   - If you created your own `bklib` file, replace the reference to the filename "`../lib/bklib.f90`" with the name of your version.
   - Adjust any of the compiler options as appropriate.

5. Run the batch file you just edited. If compilation is successful, there should be a new executable file called `smle` (if Linux) or `smle.exe` (if Windows).

6. Repeat for the other programs.

# References

[1] Krauth, Brian V., "Simulation-based estimation of peer effects," forthcoming, *Journal of Econometrics*. Available at `http://www.sfu.ca/~bkrauth/papers/smle.pdf`.