# Computer programs for simulation-based estimation of peer effects

Brian Krauth
Simon Fraser University

Version 2.0 - (RELEASE DATE???)

This file is the documentation for a set of computer programs I have written to implement the structural estimation method described in my paper "Simulation-based estimation of peer effects" [1].

## 1 Installation and use

### 1.1 Installation

To install the programs, simply unzip the file `smle.zip` into a directory of your choice. This will create 4 subdirectories:

- `windows-binaries`: contains Windows executable files. There are two estimation programs:
  - `smle.exe`: Estimates structural model from an individual-based sample.
  - `s2.exe`: Estimates structural model from a group-based sample.

  and two support programs:
  - `probit.exe`: Estimates a standard (naive) probit model from an individual-based or group-based sample.
  - `psim.exe`: Generates simulated data from the model for use in Monte Carlo experiments.

  All four of these programs are self-contained, and can be copied to and run from any directory.
- `doc`: contains documentation, including this file.
- `examples`: contains examples.
- `code`: contains the source code.

### 1.2 Basic use

To use any of the programs, you first need to create two input files:

- A *data* file, i.e., a text file containing the data set from which you are estimating the model.
- A *parameter* file, i.e., a text file describing your preferred settings for the various user options.

The format of the data and parameter files varies across the 4 programs. Details on how to construct these files can be found in Sections 2-5.

Once you have created the data and parameter files, place the program you wish to use in the same directory, open a Windows command prompt in that directory and execute the command

```
program-name parameter-file-name
```

where `program-name` is the name of the program (either `smle`, `s2`, `probit`, or `psim`) and `parameter-file-name` is the name[1] of the parameter file you have created.

The program will run for anywhere from a few minutes to a few weeks, depending on the size of your dataset and the options selected. While running it will create several output files, including a log file and a file of results.

**Example:** Suppose you have installed the program into the directory `c:/my-directory/`. Open the Windows command prompt and execute the following sequence of commands:

```
cd \my-directory\examples\smle
copy \my-directory\windows-binaries\smle.exe .
smle example1.txt
```

This will run the `smle` program with the example parameter file `example1.txt`. It will run for a few minutes and create output files `ex1_log.txt` and `ex1_results.txt`. The contents of both the parameter file and the output files will be discussed in Section 2.

### 1.3 Advanced use

If you have a modern Fortran compiler (i.e., one that can compile Fortran 90/95), 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.

## 2 Using the `smle` program

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

### 2.1 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 Own Choice ($y_{gi}$) | Average Peer Choice ($\bar{y}_{gi}$) | Number of Peers ($n_g - 1$) | Aggregate Explanatory Variables ($\mathbf{z}_g$) | Individual Explanatory Variables ($\mathbf{x}_{gi}$) |
| **Range** | $\{0, 1\}$ | $[0.0, 1.0]$ | $\{1, 2, 3, \ldots\}$ | $(-\infty, \infty)$ | $(-\infty, \infty)$ |

These variables are as described in the paper.

**Example**: The first three rows in the example data set `examples/ex1_data.txt` look like this:

```
1 0.25 4 0.28822476
0 1.00 4 0.084842203
0 0.75 4 0.16752506
```

This data set features a single explanatory variable. For the first observation, we have $y_{gi} = 1$, $\bar{y}_{gi} = 0.25$, $(n_g - 1) = 4$ (so that $n_g = 5$), and $x_{gi} = 0.28822476$.

[1]Note that in previous versions, the programs did not take any arguments, and the parameter file had to be named `parm.dat`.

## 2.2 Parameter file and user options

The parameter file for the `smle` program is just a text file specifying your preferred settings for various user options.

The program is very primitive in how it reads 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 33rd line in the file. The first four lines and any even numbered lines are ignored, so they can be used for any comments one might want. When creating your own parameter file, it is best to start with an example parameter file from the `examples` directory, and then edit the file as needed for your application. The contents of a properly-constructed parameter file are listed below:

**Line 5** (`NVAR`): Number of exogenous explanatory variables (both group-level and individual-level) in data set (integer).

**Line 7** (`NOBS`): Number of observations in data set (integer).

**Line 9** (`NUMAGG`): The first `NUMAGG` explanatory variables in the data set will be treated as aggregate (i.e., group-level) variables (integer). Group-level explanatory variables are treated differently in the model from individual-level explanatory variables. See Section 4.4 in the paper for details. Note that the estimation method requires at least one individual-level variable, so `NUMAGG` must be strictly less than `NVAR`.

**Line 11** (`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. One way to see if your value of `NSIM` is big enough is to estimate the model several times and see if the parameter estimates change substantially. If they do, then `NSIM` is not big enough.

**Line 13** (`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 of DFP. Simulated annealing is usually much slower than DFP but is more robust in solving optimization problems with lots of local optima.

**Line 15** (`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`.

The GHK simulator and DFP search algorithm is the recommended combination, when available.

**Line 17** (`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 (upper and lower) on $\gamma$ using the likelihood bounds method, as described in Section 4.2 of the paper.
- `P(lot)`: Calculate selection-rule-free bounds on the likelihood function for plotting (as in Figure 2 of the paper).

- `M(inumum)`: Find selection-rule-free bounds (lower bound only) on $\gamma$ using the likelihood bounds method.

**Line 19** (`UNDERREPORTING_CORRECTION`): Indicates whether or not to correct for underreporting (logical). See Section 4.3 in the paper for details.

**Line 21** (`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.`).

**Line 23** (`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).

**Line 25** (`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 restriction in the paper.
- `F(ixed)`: Fix $\rho_\epsilon$ at the value of `FIXED_RHO` specified in line 27.
- `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.

**Line 27** (`FIXED_RHO`): Value at which to fix $\rho_\epsilon$. Ignored if `RHO_TYPE` $\neq$ `FIXED`.

**Line 29** (`FIX_GAMMA`): Indicates whether the value of $\gamma$ should be fixed rather than estimated (logical). See section 2.4 in the paper for details.

**Line 31** (`FIXED_GAMMA`): Value at which to fix $\gamma$ (real). Ignored if `FIX_GAMMA = .false.`.

**Line 33** (`DATAFILE`): Name of file from which data will be read.

**Line 35** (`LOGFILE`): Name of file to which log data will be written. File will be overwritten.

**Line 37** (`RESULTFILE`): Name of file to which estimation results will be written. Results will be appended to file.

**Line 39** (`UFILE`): Name of file in which random numbers are stored. 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.

**Line 41** (`BOOTFILE`): Same as `UFILE`, except the `BOOTFILE` is where information on the bootstrap sample is stored/loaded.

**Line 43** (`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.

**Example**: Take a look at the example parameter file `examples/smle/example1.txt` provided in the distribution:

```
EXAMPLE1.TXT: SMLE example parameter file #1
This file shows a simple example, with the standard options
Note: data is in fixed format; do not delete lines
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)
.true.
RHO_TYPE (X, Fixed, Interval, or Estimate)
X
FIXED_RHO (Used if RHO_TYPE=FIXED, real)
0.0
FIX_GAMMA (logical)
.false.
FIXED_GAMMA (Used if FIX_GAMMA=.true., real)
0.0
DATAFILE (Name of data file)
ex1_data.txt
LOGFILE (Name of file to write log info)
ex1_log.txt
RESULTFILE (Name of file to write results to)
ex1_result.txt
UFILE (Name of file to write random numbers to, or read them from)
ex1_u.txt
BOOTFILE (Name of file to write bootstrap sample to, or read them from)
ex1_boot.dat
FIXEDEFFECTS (Number of aggregate variables to treat as fixed effects)
0
```

This example shows a typical[2] implementation with basic options. The data are in a file named `ex1_data.txt`, and consist of 1,000 observations (`NOBS`= 1000) with a single explanatory variable (`NVAR`= 1) which is individual-level rather than group-level (`NUMAGG`= 0). The likelihood function is to be estimated using the GHK simulator (`SIMULATOR_TYPE`=`GHK`) based on 100 simulations (`NSIM`= 100), and is to be maximized using the DFP method with 3 restarts (`RESTARTS`= 3). The model is to be estimated from the original data and not a bootstrap resample of the data (`BOOTSTRAP`=`.FALSE.`) under the assumptions that individuals accurately report the behavior of their peers (`UNDERREPORTING_CORRECTION`=`.FALSE.`), that groups always play the "low-activity" equilibrium (`EQUILIBRIUM_TYPE`=`Low`), and that the within-group correlation in unobservables is equal to the within-group correlation in observables (`RHO_TYPE`=`X`). Intermediate results are to be output to the log file `ex1_log.txt`, and final results are to be output to the results file `ex1_result.txt`.

## 2.3 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.

### 2.3.1 Log file

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

[2] One way that this implementation is not typical is that it sets `LOAD_U`=`.TRUE.`, i.e. the program is asked to construct the simulations based on the random numbers provided in the file `ex1_u.txt`. The usual setting would be `LOAD_U`=`.FALSE.`, i.e., the program should generate a new set of random numbers each time it runs.

### 2.3.2 Results file (ordinary estimation)

Estimation results are written[3] to the file specified as `RESULTFILE` in the parameter file. The first number written to the file is the (maximized) log-likelihood. After that come the estimates $(\hat{\rho}_x, \hat{\rho}_\epsilon, \hat{\gamma}, \hat{\beta})$.

**Example:** If you execute the command `smle example1.txt` from the `examples/smle/` directory, the results file might[4] look like this:

```
-2269.71293038489 0.269466878031412 0.269466878031412 -6.618180087163724E-003 -2.755343936372363E-002 0.377144904928951
```

The interpretation of these results is simple. The maximized log-likelihood is approximately $-2269.71$. The estimated correlation in observables is $\hat{\rho}_x \approx 0.269$. The estimated correlation in unobservables is also $\hat{\rho}_\epsilon \approx 0.269$ (note that $\hat{\rho}_x = \hat{\rho}_\epsilon$, as specified in the parameter file). The estimated peer effect is $\hat{\gamma} \approx -6.618 \times 10^{-3}$ (essentially zero). The estimated intercept is $\hat{\beta}_0 \approx -2.755 \times 10^{-2}$ and the estimated coefficient on the single explanatory variable is $\hat{\beta}_1 \approx 0.377$.

### 2.3.3 Results file (bootstrap estimation)

If the user option `BOOTSTRAP` (line 21 of the parameter file) is set to `.TRUE.`, the program will estimate the model on a (naive) bootstrap sample of the original data, and then write the results to `RESULTFILE`. The program will repeat the process 100 times[5], drawing a new bootstrap sample each time. This may take a very long time - if it takes 10 minutes to estimate the model once, it will take 17 hours to estimate it 100 times.

**Example:** If you execute the command `smle example6.txt` the program will create a results file called `ex6_results.txt` that looks something like this:

```
-2.2697129518736533E+03   0.2689591776554643  -1.0613797086889261E-02  -2.7751454571931952E-02   0.3768691221571507
-2.2617211025870947E+03   0.2707122386550440  -9.0122868167062196E-03  -8.1050563747404186E-03   0.3787628374547526
etc.
```

Note that the program does not calculate or report bootstrap standard errors. Instead, you will need to read these bootstrap results into some standard statistical analysis package and calculate the covariance matrix from them.

### 2.3.4 Results file (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:

**Line 1** Standard assumption, $\rho_\epsilon = \rho_x$, $\gamma$ to be estimated.

**Line 2** $\gamma = 0$, $\rho_\epsilon$ to be estimated.

**Line 3** $\rho_\epsilon = 0.0$, $\gamma$ to be estimated.

**Line 4** $\rho_\epsilon = 0.1$, $\gamma$ to be estimated.

$\vdots$

**Line 12** $\rho_\epsilon = 0.9$, $\gamma$ to be estimated.

[3] If the `RESULTFILE` already exists, the results are appended to whatever is already in the file.

[4] On many systems the program will break long lines, so be sure to look directly at the text file before importing it into some other program.

[5] If you wish to generate a larger bootstrap sample, just run the program again a few times. Because the program appends the results to `RESULTFILE` your new bootstrap results will be appended to your old results.

#### 2.3.5 Results file (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.

#### 2.3.6 Checkpoint files

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.

# 3 Using the s2 program

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

## 3.1 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
```

## 3.2 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.

### 3.3 Output files

The output, logging, and checkpoint files for the `s2` program take almost the same form as described in Section 2.3 for the `smle` program. The only difference is that if `COVMAT=OPG` or `COVMAT=HESSIAN`, the estimated covariance matrix is reported as well.

## 4 Using the `probit` program

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

### 4.1 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.

### 4.2 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 2.
   - G(roup): Group-based sample, as described in Section 3. 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.

### 4.3 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.

## 5 Using the psim program

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

### 5.1 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`".

## 5.2 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.

# B Version history

- Version 1.1 - March 1, 2005. First publicly released version.
- Version 2.0 - ???.
  - Windows binaries are now compiled on Intel Fortran compiler, leading to a 43% reduction in program time for SMLE.EXE and a 16% reduction for S2.EXE.
  - Program can now be compiled on IBM XL Fortran compiler. This involved 3 changes to code:
    * Workaround inserted to deal with compiler bug related to use of MATMUL command with calculated indices.
    * Code that reads in data from text files now strips CR and LF characters from end of each line.
    * Stack overflow error repaired.
  - Program (windows binary version) can now take user-supplied name of parameter file. Previous version only allowed the parameter file to be named "parm.dat".

# References

[1] Krauth, Brian V., 2006. "Simulation-based estimation of peer effects," *Journal of Econometrics* 133(1): 243-271.