The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–27

siman: Suite for analysing simulation studies

Ella Marley-Zagar
MRC Clinical Trials Unit
University College London
London, UK
e.marley-zagar@ucl.ac.uk

Tim P. Morris
MRC Clinical Trials Unit
University College London
London, UK
tim.morris@ucl.ac.uk

Ian R. White
MRC Clinical Trials Unit
University College London
London, UK
ian.white@ucl.ac.uk

Abstract. Simulation studies are used in medical statistics to evaluate the accuracy of new and alternative statistical methods. Simulation studies involve creating data by random sampling from known probability distributions, with the aim of assessing the robustness and accuracy of new statistical techniques. We introduce the `siman` suite, a set of Stata programs that offer data manipuation and graphs to explore the results of simulation studies.

1 Introduction

The paper comprises a description of the new syntax (Section 2), illustrative examples (Section 3), details of the methods used (Section 4), a description of how the software has been tested (Section 5), and a discussion (Section 6).

2 The `siman` commands

2.1 Data formats

There are two types of data sets that the `siman` suite can be applied to:

Estimates data set

The estimates data set contains the results from analysing multiple simulated data sets, with data relating to different statistics (e.g. point estimate, p-value) for each simulated data set. The `siman` program is able to work with the results of any simulation study, by reading the user's estimates data set and formatting it into the correct set up for use by `siman`.

Performance measures data set

The performance measure data set contains the results from analysing an estimates

data set, with data relating to different performance measures (e.g. bias, coverage) summarised over estimates data sets for different data generating mechanisms. Within the `siman` suite, the performance meaures data set is created by `siman analyse` and is appended to the estimates data set unless the user chooses not to do so.

For the **input** data, there will be 4 data set *formats* permitted by the `siman` suite:

1. **Long for both targets and methods** (*longlong* format, option 1)

An example of a *longlong* data set is given below, where both the method and the target variables are listed in separate columns.

rep	dgm	target	method	true	est	se
1	1	beta	1	0	-0.14	0.07
1	1	beta	2	0	-0.23	0.11
1	1	gamma	1	0	-0.05	0.08
1	1	gamma	2	0	-0.13	0.12
1	2	beta	1	0	-0.11	0.09
1	2	beta	2	0	-0.15	0.14
1	2	gamma	1	0	-0.06	0.09
1	2	gamma	2	0	-0.16	0.13

2. **Wide for both targets and methods** (*widewide* format, option 2)

An example of a *widewide* data set is given below, where both the method and the target values from the data set above are now stubs.

rep	dgm	true	est1beta	se1beta	est2beta	se2beta	est1gamma	se1gamma	est2gamma	se2gamma
1	1	0	-0.14	0.07	-0.23	0.11	-0.05	0.08	-0.13	0.12
1	2	0	-0.11	0.09	-0.15	0.14	-0.06	0.09	-0.16	0.13

3. **Long for targets, wide for methods** (*longwide* format, option 3)

An example of a *longwide* data set is given below, where the target variable is in a column and the method values are stubs.

rep	dgm	target	true	est1	se1	est2	se2
1	1	beta	0	-0.14	0.07	-0.23	0.11
1	1	gamma	0	-0.05	0.08	-0.13	0.12
1	2	beta	0	-0.11	0.09	-0.15	0.14
1	2	gamma	0	-0.06	0.09	-0.16	0.13

4. **Wide for targets, long for methods** (*widelong* format, option 4)

An example of a *widelong* data set is given below, where the method variable is in a column and the target values are stubs.

rep	dgm	method	true	estbeta	sebeta	estgamma	segamma
1	1	1	0	-0.14	0.07	-0.05	0.08
1	1	2	0	-0.23	0.11	-0.13	0.12
1	2	1	0	-0.11	0.09	-0.06	0.09
1	2	2	0	-0.15	0.14	-0.16	0.13

2.2 The `siman` setup command

`siman setup` takes the user's raw simulation data (estimates data set) and puts it in the format required by `siman`. The user's raw data can be in either long or wide format as detailed in the Data formats section (2.1) above. The user will have an unmodified, raw simulation data set with the repetitions of a simulation experiment, `rep(`*varname*`)`, as the only compulsory variable in the syntax. Other variables of interest such as data generating mechanism (`dgm`), `target`, `method`, `estimate`, standard error (`se`) can be specified. The user will declare their variables for use in `siman setup`. `siman setup` will check the data, reformat it and attach characteristics to the data set available for use across multiple sessions. Every other `siman` command will then read the characteristics of the data created in `siman setup`. `siman setup` will allow multiple data generating mechanisms, multiple targets and multiple analysis methods. The siman estimates data set will be held in memory. There will also be a auto-summary output available for the user to confirm the data set up (using `siman describe`).

`siman setup` will automatically reshape wide-target data (i.e. wide targets, wide methods) or wide-long format data (i.e. wide targets, long methods) into long-wide format. Therefore the two output data formats of `siman setup` are *longlong* (option 1) and *longwide* (option 3).

Note that `true` must be a *variable* in the dataset for `siman trellis` (see section 2.6.2) and `siman nestloop` (see section 2.6.3) - and should be listed in both the `dgm()` and the `true()` options in `siman setup` before running these graphs.

Syntax

`siman setup` [*if*] [*in*] `, rep(`*varname*`)` [*options*]

Options for data in *longlong* input format (option 1):

`dgm(`*varlist*`) target(`*varname*`) method(`*varname*`) estimate(`*varname*`)`
`se(`*varname*`) df(`*varname*`) lci(`*varname*`) uci(`*varname*`) p(`*varname*`)`
`true(#|`*varname*`) clear`

Options for data in *widewide* input format (option 2):

dgm(*varlist*) target(*values*) method(*values*) estimate(*stub_varname*)
se(*stub_varname*) df(*stub_varname*) lci(*stub_varname*) uci(*stub_varname*)
p(*stub_varname*) true(#|*stub_varname*) order(*varname*) clear

Options for data in *longwide* input format (option 3):

dgm(*varlist*) target(*varname*) method(*values*) estimate(*stub_varname*)
se(*stub_varname*) df(*stub_varname*) lci(*stub_varname*) uci(*stub_varname*)
p(*stub_varname*) true(#|*stub_varname*) clear

Options for data in *widelong* input format (option 4):

dgm(*varlist*) target(*values*) method(*varname*) estimate(*stub_varname*)
se(*stub_varname*) df(*stub_varname*) lci(*stub_varname*) uci(*stub_varname*)
p(*stub_varname*) true(#|*stub_varname*) clear

Options

dgm(*varlist*) data generating mechanism.

target(*varname*|*values*) the target variable name or values.

method(*varname*|*values*) the method variable name or values.

estimate(*varname*|*stub_varname*) the estimate variable name or the name of it's stub if in wide format.

se(*varname*|*stub_varname*) the standard error variable name or the name of it's stub if in wide format.

df(*varname*|*stub_varname*) the degrees of freedom variable name or the name of it's stub if in wide format.

lci(*varname*|*stub_varname*) the lower confidence interval variable name or the name of it's stub if in wide format.

uci(*varname*|*stub_varname*) the upper confidence interval variable name or the name of it's stub if in wide format.

p(*varname*|*stub_varname*) the p-value variable name or the name of it's stub if in wide format.

true(#|*varname*|*stub_varname*) the true value, or variable name or the name of it's stub if in wide format.

`order(`*varname*`)` if in wide-wide format, this will be either *target* or *method*, denoting that either the target stub is first or the method stub is first in the variable names.

2.3 The `siman analyse` command

`siman analyse` takes the imported estimates data from `siman setup` and creates performance measures data via the user-written program `simsum` (White 2010).

By default `siman analyse` will append the performance measures to the estimates data set, with the perfomance measure names listed in the `rep` column. Additionally the performance measure code (as listed below) and the dataset (*estimates* or *performance*) will be listed for each dataset row.

Syntax

`siman` <u>`ana`</u>`lyse` [*if*] `,` [*performancemeasures perfonly replace*]

Options

Main:

`if` can be applied to `dgm`, `target` and `method` only.

Performance measures:

`performancemeasures` as per `simsum`. If none of the following options are specified, then all available performance measures are computed.

`bsims` reports the number of simulations with non-missing point estimates.

`sesims` reports the number of simulations with non-missing standard errors.

`bias` estimates the bias in the point estimates.

`empse` estimates the empirical standard error – the standard deviation of the point estimates.

`relprec` estimates the relative precision – the inverse squared ratio of the empirical standard error of this method to the empirical standard error of the reference method. This calculation is slow: omitting it can reduce run time by up to 90%.

`mse` estimates the mean squared error.

`modelse` estimates the model-based standard error.

`relerror` estimates the proportional error in the model-based standard error, using the empirical standard error as gold standard.

`cover` estimates the coverage of nominal confidence intervals at the specified level.

`power` estimates the power to reject the null hypothesis that the true parameter is zero, at the specified level.

`mean` the average (mean) of the point estimates.

`rmse` estimates the root mean squared error (check with Ian).

`ciwidth` estimates the width of the confidence interval at the specified level (check with Ian).

`perfonly` the program will automatically append the performance measures data to the estimates data, unless the user specifies `perfonly` for performance measures only.

`replace` if `siman analyse` has already been run and the user specifies it again then they must use the `replace` option, to replace the existing perfomance measures in the data set.

2.4 Utility commands

2.4.1 `siman reshape`

`siman reshape` converts the dataset held in memory and reshapes it in to long-long format (i.e. long targets, long methods) or long-wide format (i.e. long targets, wide methods). The reshaped data set is held in memory. There is also an auto-summary output available for the user to confirm the data set up (using `siman describe`, as described in the next section).

Syntax

`siman reshape,` [*longlong longwide*]

2.4.2 `siman describe`

This describes the data import from `siman setup`.

`siman describe` provides a summary of the data imported by `siman setup`. It will list the data format as format *(n : x y)* where *n* is the data format type (see section 2.1), *x* is the `target` format and *y* is the `method` format. `siman describe` will either list format 1 (*long-long*) or format 3 (*long-wide*) as these are the two output data format types available from `siman setup`.

For clarity the resulting `target` and `method` format will also be listed, along with the number of and values of target(s), method(s) and dgm(s).

`siman describe` will also list the `estimate`, `se`, `df`, `ci`, `p` and `true` variable names if applicable, and whether estimates are contained in the dataset.

`siman describe` will be called automatically by `siman setup`, but can also be called on it's own once the data has been imported by the `siman suite`.

Syntax

`siman describe`

`siman describe` will use the *if* and *in* conditions from `siman setup` if specified.

2.4.3 `siman table`

Describes the performance measures data created by `siman analyse`.

`siman table` uses the inbuilt Stata program `tabdisp` to provide a summary of the performance measures created by `siman analyse`. The output table lists the estimand(s) split by performance measure(s) and methods.

`siman table` is called automatically by `siman analyse`, but can also be called on its own once the performance measures data has been created by the `siman suite`.

Syntax

`siman table performancemeasures` [*if*] `,` [*column(varname)*]

Options

`if` can be applied to `dgm`, `target` and `method` only. If not specified then the if condition from `siman analyse` will be used.

`performancemeasures` as per 2.3. If none are specified, then all available performance measures are computed.

`column`(*varname*) can be used to move factors to columns.

2.5 Graphs for the Estimates data set

2.5.1 `siman scatter`

`siman scatter` draws a scatter plot of the point estimates data versus standard error data, the results of which are from analysing multiple simulated data sets with data relating to different statistics (e.g. point estimate) for each simulated data set.

`siman setup` needs to be run first before `siman scatter`.

Syntax

`siman scatter` [*if*][*in*], [*options*]

If no variables are specified, then the scatter graph will be drawn for *estimate* versus *se*. Alternatively the user can select *se* versus *estimate*.

Options

`if/in` The user can specify *if* and *in* within the `siman scatter` syntax. If they do not, but have already specified an *if/in* during `siman setup`, then the *if/in* from `siman setup` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `rep` and an error message will be issued if the user tries to do so.

`by`(*string*) specifies the nesting of the variables, with the default being `by(dgm target method)`

`bygraphoptions`(*string*) graph options for the nesting of the graphs due to the *by* option.

For the `siman scatter` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

2.5.2 `siman swarm`

`siman swarm` draws a swarm plot of the estimates data variables, the results of which are from analysing multiple simulated data sets with data relating to different statistics (e.g. point estimate, p-value) for each simulated data set.

`siman setup` needs to be run first before `siman swarm`.

If the data is not already in *longlong* format then it will be reshaped to this format to create the graphs (see `siman reshape` for further details on data format types and the `reshape` command).

`siman swarm` requires a `method` variable/values in the estimates dataset defined in the `siman setup` syntax by `method()`.

`siman swarm` requires at least 2 `methods` to compare.

The `labelsof` package (by Ben Jann) is required by `siman swarm`, which can be installed by typing `ssc install labelsof` in Stata.

Syntax

`siman swarm` [*if*][*in*], [*options*]

If no estimates data variables are specified, then the `swarm` graph will be drawn for *estimate* only.

Options

`if/in` The user can specify *if* and *in* within the `siman swarm` syntax. If they do not, but have already specified an *if/in* during `siman setup`, then the *if/in* from `siman setup` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `rep` and an error message will be issued if the user tries to do so.

`by`(*string*) specifies the nesting of the variables, with the default being `by(dgm target method)`

`meanoff` to turn off displaying the mean on the `swarm` graphs

`meangraphoptions`(*string*) graph options for the mean

`bygraphoptions`(*string*) graph options for the nesting of the graphs due to the *by* option.

`graphoptions`(*string*) graph options for the overall graphical display

For the `siman swarm` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

2.5.3 `siman blandaltman`

`siman blandaltman` draws a Bland-Altman plot comparing estimates and/or standard error data from different methods. If there are more than 2 methods in the data set, for example methods A B and C, then the first method will be taken as the reference, and the `siman blandaltman` plots will be created for method B - method A and method C - method A. Alternatively, pairs of methods can be specified for comparison using `methlist()`.

`siman setup` needs to be run first before `siman blandaltman`.

As the Bland-Altman plots are drawn per method comparison set, the `by()` variable can be used to stratify further `by(dgm)` or `by(target)`. The default is `by(dgm target)`.

The `labelsof` package (by Ben Jann) is required by `siman blandaltman`, which can be installed by typing `ssc install labelsof` in Stata.

Syntax

`siman blandaltman` [*if*][*in*], [*options*]

If no variables are specified, then the blandaltman graph will be drawn for *estimates*

only. Alternatively the user can select *se* or *estimate se*.

Options

`if/in` The user can specify *if* and *in* within the `siman blandaltman` syntax. If they do not, but have already specified an *if/in* during `siman setup`, then the *if/in* from `siman setup` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `rep` and an error message will be issued if the user tries to do so.

`by`(*string*) specifies the nesting of the variables, with the default being `by(dgm target)`.

`bygraphoptions`(*string*) graph options for the nesting of the graphs due to the *by* option.

`methlist`(*string*) if the user would like to display the graphs for a subgroup of methods, these methods can be specified in `methlist()`. For example, in a dataset with methods A, B and C, if the user would like to compare methods A and C, they would enter `methlist(A C)`.

For the `siman blandaltman` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

2.5.4 `siman comparemethodsscatter`

`siman comparemethodsscatter` draws sets of scatter plots comparing the point estimates data or standard error data for various methods, where each point represents one repetition.

`siman setup` needs to be run first before `siman comparemethodsscatter`.

For up to 3 methods (inclusive), `siman comparemethodsscatter` will plot both the estimate and the standard error. The upper triangle will display the estimate data, the lower triangle will display the standard error data. For more than 3 methods, `siman comparemethodsscatter` will plot either the estimate or the standard error depending on which the user specifies, with the default being estimate if no variables are specified. The graph for the larger number of methods is plotted using the `graph matrix` command.

If there are many methods in the data set and the user wishes to compare subsets of methods, then this can be achieved by using the `methlist()` option. However if your data has underscores, for example *wide-wide* data where the `method` and `target` are both in the variable name such as `estA_beta estA_gamma estB_beta estB_gamma estC_beta estC_gamma`, then in `siman setup`, you would have specified `method(A_ B_ C_)`. However if you would then like to graph a subset of methods A and B with `siman comparemethodsscatter` then you would enter `methlist(A B)` [not `methlist(A_ B_)`].

The graphs are split out by `dgm` (one graph per `dgm`) and they compare the methods to

each other. Therefore the only other option to split the graphs with the `by` option is `by target`, so the `by(varlist)` option will only allow `by(target)`.

The `labelsof` package (by Ben Jann) is required by `siman comparemethodsscatter`, which can be installed by typing `ssc install labelsof` in Stata.

Syntax

`siman comparemethodsscatter` *[estimate|se]* [*if*][*in*], [*options*]

The scatter graph will be drawn for `estimate` and `se` if the number of methods is <= 3. Alternatively the user can select `estimate` or `se` for more than 3 methods.

Options

`if/in` The user can specify *if* and *in* within the `siman comparemethodsscatter` syntax. If they do not, but have already specified an *if/in* during `siman setup`, then the *if/in* from `siman setup` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `rep` and an error message will be issued if the user tries to do so.

`by`(*string*) specifies the nesting of the variables. The graphs will always be by `dgm` as an unspecified default, the user can also additionally add `by(target)`.

`subgraphoptions`(*string*) changes the format of the constituent scatter graphs.

`methlist`(*string*) if the user would like to display the graphs for a subgroup of methods, these methods can be specified in `methlist()`.

For the `siman comparemethodsscatter` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

2.5.5 `siman zipplot`

`siman zipplot` draws a "zip plot" plot of the confidence interval coverage for each data-generating mechanism and analysis method in the estimates data, the results of which are from analysing multiple simulated data sets with data relating to different statistics (e.g. point estimate, p-value) for each simulated data set. Both Monte Carlo confidence intervals for percent coverage and 95% confidence intervals for individual repetitions are shown. For coverage (or type I error), true θ is used for the null value.

For each data-generating mechanism and method, the confidence intervals are fractional-centile-ranked (see Morris et al. (2019)). This ranking is used for the vertical axis and is plotted against the intervals themselves. Intervals which cover the true value are coverers (at the bottom); those which do not cover are called non-coverers (at the top). Both coverers and non-coverers are shown on the plot, along with the point

estimates.

`siman setup` needs to be run first before `siman zipplot`.

`siman zipplot` requires a true variable in the estimates dataset defined in the `siman setup` syntax by `true()`.

Syntax

`siman zipplot` [*if*][*in*], [*options*]

Options

`if/in` The user can specify *if* and *in* within the `siman zipplot` syntax. If they do not, but have already specified an *if/in* during `siman setup`, then the *if/in* from `siman setup` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `rep` and an error message will be issued if the user tries to do so.

`by`(*string*) specifies the nesting of the variables, with the default being being `by(dgm method)` if there is only one true value, and `by(dgm target method)` where there are different true values per target.

`noncoveroptions`(*string*) graph options for the non-coverers.

`coveroptions`(*string*) graph options for the coverers.

`scatteroptions`(*string*) graph options for the scatter plot of the point estimates.

`truegraphoptions`(*string*) graph options for the true value(s).

`bygraphoptions`(*string*) graph options for the nesting of the graphs due to the *by* option.

`scheme`(*string*) to change the graph scheme.

For the `siman zipplot` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

2.6 Graphs for the Performance Measures data set

2.6.1 `siman lollyplot`

`siman lollyplot` draws a lollipop plot of performance measures data. It is a graphical presentation of estimated performance where different performance measures are stacked vertically; for each performance measure, the results for each method occupy one row; results for different methods are arranged across the two columns. Monte Carlo 95% confidence intervals are represented via parentheses (a visual cue due to the usual presentation of intervals as two numbers within parentheses) (see Morris

et al. (2019)).

The graphs will be produced by `dgm`, with one point/line drawn per method.

`siman setup` and `siman analyse` need to be run first before `siman lollyplot`.

Syntax

`siman lollyplot` *[performancemeasures]* [*if*], [*options*]

If no performance measures are specified, then the lolliplot graph will be drawn for all performance measures in the data set. Alternatively the user can select a subset of performance measures to be graphed using the performance measures listed below.

Options

`if` The user can specify *if* within the `siman lollyplot` syntax. If they do not, but have already specified an if during `siman analyse`, then the *if* from `siman analyse` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `repetition` and an error message will be issued if the user tries to do so.

`graphoptions`(*string*) graph options for the constituent performance measure graphs.

For the `siman lollyplot` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

Performance measure options

See performance measures as per `siman analyse` and `simsum` in Section (2.3).

2.6.2 `siman trellis`

`siman trellis` draws a trellis plot of performance measures data. It is a graphical presentation of method performance per data generating mechanism (dgm) for each performance measure.

The graphs will be produced by `dgm`, with one line drawn per method. `siman trellis` is intended for datasets that have more than 1 true value.

`true` must be a variable in the dataset for `siman trellis`, and should be listed in both the `dgm()` and the `true()` options in `siman setup`.

`siman setup` and `siman analyse` need to be run first before `siman trellis`.

Syntax

`siman tre`llis *[performancemeasures]* [*if*], [*options*]

If no performance measures are specified, then the trellis graph will be drawn for all performance measures in the data set. Alternatively the user can select a subset of performance measures to be graphed using the performance measures listed below.

Options

`if` The user can specify *if* within the `siman trellis` syntax. If they do not, but have already specified an if during `siman analyse`, then the *if* from `siman analyse` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `repetition` and an error message will be issued if the user tries to do so.

`bygr`aphoptions(*string*) graph options for the nesting of the graphs by data generating mechanism.

For the `siman trellis` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

Performance measure options

See performance measures as per `siman analyse` and `simsum` in Section (2.3).

2.6.3 `siman nestloop`

`siman nestloop` draws a nested loop plot of performance measures data.

The nested loop plot presents all simulation results in one plot. The performance measure is split by method and is stacked according to the levels of the data generating mechanisms along the horizontal axis.

We recommend to sort the simulation dataset in such a way that the simulation parameter with the largest influence on the criterion of interest is considered first, and so forth. Further guidance can be found in Rücker and Schwarzer (2014).

`true` must be a variable in the dataset for `siman nestloop`, and should be listed in both the `dgm()` and the `true()` options in `siman setup`.

`siman setup` and `siman analyse` need to be run first before `siman nestloop`.

Syntax

`siman nes`tloop *[performancemeasures]* [*if*], [*options*]

If no performance measures are specified, then the nestloop graph will be drawn for all performance measures in the data set. Alternatively the user can select a subset of performance measures to be graphed using the performance measures listed below.

Options

`if` The user can specify *if* within the `siman nestloop` syntax. If they do not, but have already specified an if during `siman analyse`, then the *if* from `siman analyse` will be used. The *if* option will only apply to `dgm`, `target` and `method`. The *if* option is not allowed to be used on `repetition` and an error message will be issued if the user tries to do so.

`dgmorder`(*string*) order of data generating mechanisms for the nested loop plot. A negative sign infront of the variable name will display it's values on the graph in descending order.

`connect`(*string*) connecting option for main graph and descriptor graph.

`stagger`(#) stagger option for main graph. Default # is 0.

`fraclegend`(#) controls sizing for legend. Default # is 0.35.

`fracgap`(#) controls sizing for gap in legend. Default # is 0.

`legendgap`(#) controls sizing for descriptor graph. Default # is 3.

`legendcolor`(*string*) controls colours for descriptor graph.

`legendpattern`(*string*) controls pattern for descriptor graph.

`legendsize`(*string*) controls size of descriptor graph.

`legendstyle`(*string*) controls style of descriptor graph.

`legendwidth`(*string*) controls width of descriptor graph.

For the `siman nestloop` graph user-inputted options, most of the valid options for the inbuilt Stata command `scatter` are available.

Performance measure options

See performance measures as per `siman analyse` and `simsum` in Section (2.3).

3 Examples

3.1 x

```
. artbin, pr(0.1 0.05) alpha(0.05) power(0.9) wald
ART - ANALYSIS OF RESOURCES FOR TRIALS (binary version 2.0 08nov2021)
```

```
A sample size program by Abdel Babiker, Patrick Royston, Friederike Barthel,
Ella Marley-Zagar and Ian White
MRC Clinical Trials Unit at UCL, London WC1V 6LJ, UK.
```

Type of trial	superiority
Number of groups	2
Favourable/unfavourable outcome	unfavourable *Inferred by the program*
Allocation ratio	equal group sizes
Statistical test assumed	unconditional comparison of 2 binomial proportions P1 and P2 using the wald test
Local or distant	distant
Continuity correction	no
Anticipated event probabilities	0.100 , 0.050
Alpha	0.050 (two-sided) (taken as .025 one-sided)
Power (designed)	0.900
Total sample size (calculated)	1156
Sample size per group (calculated)	578 578
Expected total number of events	86.70

3.2 xx

```
. power twoproportions 0.1 0.05, alpha(0.05) power(0.9)

Performing iteration ...

Estimated sample sizes for a two-sample proportions test
Pearson´s chi-squared test
Ho: p2 = p1  versus  Ha: p2 != p1

Study parameters:

        alpha =    0.0500
        power =    0.9000
        delta =   -0.0500  (difference)
           p1 =    0.1000
           p2 =    0.0500

Estimated sample sizes:

            N =     1,164
  N per group =       582

.
. artbin, pr(0.1 0.05) alpha(0.05) power(0.9)

ART - ANALYSIS OF RESOURCES FOR TRIALS (binary version 2.0 08nov2021)
```

```
A sample size program by Abdel Babiker, Patrick Royston, Friederike Barthel,
Ella Marley-Zagar and Ian White
MRC Clinical Trials Unit at UCL, London WC1V 6LJ, UK.
```

Type of trial	superiority
Number of groups	2
Favourable/unfavourable outcome	unfavourable *Inferred by the program*
Allocation ratio	equal group sizes
Statistical test assumed	unconditional comparison of 2 binomial proportions P1 and P2

	using the score test
Local or distant	distant
Continuity correction	no
Anticipated event probabilities	0.100 , 0.050
Alpha	0.050 (two-sided) (taken as .025 one-sided)
Power (designed)	0.900
Total sample size (calculated)	1164
Sample size per group (calculated)	582 582
Expected total number of events	87.30

3.3 xxx

```
. artbin, pr(0.9 0.9) margin(-0.05) onesided
ART - ANALYSIS OF RESOURCES FOR TRIALS (binary version 2.0 08nov2021)
```

```
A sample size program by Abdel Babiker, Patrick Royston, Friederike Barthel,
Ella Marley-Zagar and Ian White
MRC Clinical Trials Unit at UCL, London WC1V 6LJ, UK.
```

Type of trial	non-inferiority
Number of groups	2
Favourable/unfavourable outcome	favourable *Inferred by the program*
Allocation ratio	equal group sizes
Statistical test assumed	unconditional comparison of 2 binomial proportions P1 and P2 using the score test
Local or distant	distant
Continuity correction	no
Null hypothesis H0:	H0: p2-p1<= -.05
Alternative hypothesis H1:	H1: p2-p1> -.05
Anticipated event probabilities	0.900 , 0.900
Alpha	0.050 (one-sided)
Power (designed)	0.800
Total sample size (calculated)	914
Sample size per group (calculated)	457 457
Expected total number of events	822.60

3.4 xxxx

```
. artbin, pr(0.1 0.2 0.3 0.4) alpha(0.1) power(0.9)
ART - ANALYSIS OF RESOURCES FOR TRIALS (binary version 2.0 08nov2021)
```

```
A sample size program by Abdel Babiker, Patrick Royston, Friederike Barthel,
Ella Marley-Zagar and Ian White
MRC Clinical Trials Unit at UCL, London WC1V 6LJ, UK.
```

Type of trial	superiority
Number of groups	4

Favourable/unfavourable outcome	not determined
Allocation ratio	equal group sizes
Statistical test assumed	unconditional comparison of 4 binomial proportions using the score test
Local or distant	distant
Continuity correction	no
Anticipated event probabilities	0.100, 0.200, 0.300, 0.400
Alpha	0.100 (two-sided)
Power (designed)	0.900
Total sample size (calculated)	176
Sample size per group (calculated)	44 44 44 44
Expected total number of events	44.00

3.5 xxxxx

```
. artbin, pr(0.7 0.75) margin(-0.1) power(0.8) ar(1 2) wald ltfu(0.2)
ART - ANALYSIS OF RESOURCES FOR TRIALS (binary version 2.0 08nov2021)
```

A sample size program by Abdel Babiker, Patrick Royston, Friederike Barthel, Ella Marley-Zagar and Ian White
MRC Clinical Trials Unit at UCL, London WC1V 6LJ, UK.

Type of trial	non-inferiority
Number of groups	2
Favourable/unfavourable outcome	favourable *Inferred by the program*
Allocation ratio	1:2
Statistical test assumed	unconditional comparison of 2 binomial proportions P1 and P2 using the wald test
Local or distant	distant
Continuity correction	no
Null hypothesis H0:	H0: p2-p1<= -.1
Alternative hypothesis H1:	H1: p2-p1> -.1
Anticipated event probabilities	0.700 , 0.750
Alpha	0.050 (two-sided) (taken as .025 one-sided)
Power (designed)	0.800
Loss to follow up assumed:	20 %
Total sample size (calculated)	399
Sample size per group (calculated)	133 266
Expected total number of events	292.60

4 Methods and formulae

5 Software Testing

6 Discussion

7 Acknowledgements

This work was supported by the Medical Research Council Unit Programme number MC_UU_00004/09. We thank .... for their very helpful comments and for testing the program.

8 References

Morris, T. P., I. R. White, and M. J. Crowther. 2019. Using simulation studies to evaluate statistical methods. *Statistics in Medicine* 38(11): 2074–2102.

Rücker, G., and G. Schwarzer. 2014. Presenting simulation results in a nested loop plot. *BMC Medical Research Methodology* 14(1): 1–8.

White, I. R. 2010. Simsum: Analyses of simulation studies including Monte Carlo error. *Stata Journal* 10(3): 369–385.

9 About the authors

Ella Marley-Zagar is a Senior Research Associate, Medical Statistician in Methodological Software at the MRC Clinical Trials Unit in London, UK. Her research interests include developing new Stata software for clinical trials and research in lower and middle income countries.

Ian White is professor of statistical methods for medicine at the MRC Clinical Trials Unit in London, UK, where he co-leads programmes of design of clinical trials, analysis of clinical trials and meta-analysis. His research interests include study design, handling missing data and noncompliance in clinical trials, statistical models for meta-analysis, and simulation studies. He is the author of other Stata software including `mvmeta`, `network` and `simsum`.

Appendix