Default GMSE data structures

GMSE: an R package for generalised management strategy evaluation (Supporting Information 7)

A. Bradley Duthie[13], Jeremy J. Cusack[1], Isabel L. Jones[1], Jeroen Minderman[1], Erlend B. Nilsen[2], Rocío A. Pozo[1], O. Sarobidy Rakotonarivo[1], Bram Van Moorter[2], and Nils Bunnefeld[1]

[1] Biological and Environmental Sciences, University of Stirling, Stirling, UK [2] Norwegian Institute for Nature Research, Trondheim, Norway [3] alexander.duthie@stir.ac.uk

The most important (default) GMSE data structures

The default submodels of GMSE (`resource`, `observation`, `manager`, and `user`) use a small number of default data structures to hold the information needed in simulations. While these default submodels do not necessarily need to be used in every run in GMSE (see use of use of `gmse_apply`), they will be used in any run of `gmse`, and in any call of `gmse_apply` that does not run with entirely custom submodels. Simulation and model inference does not require an understanding of the default data structures, but such an understanding can be especially useful when running `gmse_apply` if there is a need to extract uncommonly used information, change key simulated values (e.g., landscape properties, agent budgets, or resource movement rules as in Supporting Information 4), or build custom individual-based submodels. Here we provide a brief explanation of the following key data structures (each name below is listed as named in the output `gmse_apply` when `get_res = "Full"`).

1. `AGENTS`
2. `resource_array` (or `RESOURCES`)
3. `observation_array` (or `OBSERVATION`)
4. `manager_array` (or `COST`)
5. `user_array` (or `ACTION`)
6. `LAND`

Note that these are not the only data structures used in GMSE, but they are the only ones that can potentially be usefully modified in GMSE v0.4.0.3 (see, e.g., Supporting Infromation 4), so they are the ones that we focus on here. Additionally, any custom subfunction that returns an array rather than a single value should adhere to the same structure as the these defaults if any default GMSE functions are to be used in `gmse_apply`. We can investigate each data structure by running a single simulation of `gmse_apply`.

```
sim <- gmse_apply(get_res = "Full");
```

The full list output of `sim` holds each structure by name (in the case where two names are used, e.g., `resource_array` and `RESOURCES`, both are identical, but the lower case `resohttps://twitter.com/existentialcoms/status` takes precedence in case of a change). Each data structure can be examined, changed, and incorporated into a new simulation (e.g., `new_sim <- gmse_apply(old_list = sim)`).

1. AGENTS

The `AGENTS` data structure is a two dimensional array with a fixed number of 17 columns and a number of rows that is always equal to the total number of manager and users (each row is an individual agent).

```
print(sim$AGENTS);
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    1    0    0    0   23   46   50    0   10     0    54     0     0
## [2,]    2    1    0    0    6   30   50    0   10     0     0     0     0
## [3,]    3    1    0    0    4   48   50    0   10     0     0     0     0
## [4,]    4    1    0    0   82   91   50    0   10     0     0     0     0
## [5,]    5    1    0    0   36   43   50    0   10     0     0     0     0
##      [,14] [,15]    [,16] [,17]
## [1,]     0     0 9535.938  1000
## [2,]     0     0    0.000  1000
## [3,]     0     0    0.000  1000
## [4,]     0     0    0.000  1000
## [5,]     0     0    0.000  1000
```

In the default case above, there are five agents (one manager and four user), each represented by a unique row. Columns in the array represent the agent traits listed below.

1. ID (each agent gets a unique number)
2. Type 1 (0 indicates the manager; 1 indicates users)
3. Type 2 (currently unused)
4. Type 3 (currently unused)
5. x-location on the landscape (typically ignored)
6. y-location on the landscape (typically ignored)
7. Movement distance (typically ignored)
8. Time parameter (typically ignored)
9. Distance of vision (currently used only for managers)
10. Error parameter (currently unused)
11. Resource marking parameter (currently used only for managers)
12. Resource tally parameter (currently used only for managers)
13. Unused column 1
14. Unused column 2
15. Unused column 3
16. Yield from owned land (zero for users when default `land_ownership = FALSE`)
17. Budget

It is obvious from the above list that most columns represent traits that are either typically ignored or currently not in use. This is intended to allow for easier future development of default model options and potential customisation of submodels in `gmse_apply`. We anticipate that future versions of GMSE will contain multiple user types with unique traits and among-user interactions.

2. resource_array

The `resource_array` (also accessible as `RESOURCES`) is a two dimensional array with a fixed number of 20 columns and a number of rows that is always equal to the total number of resources (each row is an individual resource). In the above simulation, `sim$resource_array` includes 1080 rows, so we only print out the first eight for illustration.

```
print(sim$resource_array[1:8,]);
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    1    1    0    0   71   73   20    1    0   0.3     0     3     0
## [2,]    2    1    0    0   55    3   20    1    0   0.3     0     4     0
## [3,]    3    1    0    0   12   44   20    1    0   0.3     0     5     0
## [4,]    4    1    0    0   83   87   20    1    0   0.3     0     4     0
## [5,]    6    1    0    0   31   74   20    1    0   0.3     0     5     0
## [6,]    7    1    0    0    9   36   20    1    0   0.3     0     2     0
```

```
## [7,]   10    1    0    0   27   53   20    1    0   0.3     0     5     0
## [8,]   11    1    0    0   43   74   20    1    0   0.3     0     2     0
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20]
## [1,]     0   0.5     0     0     0     0     0
## [2,]     0   0.5     0     0     0     0     0
## [3,]     0   0.5     0     0     0     0     0
## [4,]     0   0.5     0     0     0     0     0
## [5,]     0   0.5     0     0     0     0     0
## [6,]     0   0.5     0     0     0     0     0
## [7,]     0   0.5     0     0     0     0     0
## [8,]     0   0.5     0     0     0     0     0
```

Columns in the resource array represent the individual resource traits listed below.

1. ID (each resource gets a unique number)
2. Type 1 (currently all resources are of type 1)
3. Type 2 (currently unused)
4. Type 3 (currently unused)
5. x-location on the landscape
6. y-location on the landscape
7. Movement distance
8. Time parameter (typically ignored)
9. Removal (i.e., death) probability
10. Growth (i.e., birth) probability
11. Offspring produced
12. Age (initial resources are given a random age between 1 and the maximum age sampled from a uniform distribution; offspring always start at age zero in their time step of birth)
13. Marking indicator (used in the observation function)
14. Tallying indicator (used in the observation function)
15. Proportion of a landscape cell the resource consumes in a time step
16. Has the resource been scared by an agent?
17. Has the resource been culled by an agent?
18. Has the resource been castrated by an agent?
19. Has the resource's growth rath been increased by an agent?
20. Has the resource's offspring production been increased by an agent?

In the case of columns 16-20, the value is either zero (if no action has occurred), or some positive integer that matches the ID of the agent that has performed the act (e.g., if column 17 equals 3, then that means that the agent with ID 3 culled the resource in the corresponding row; where more than one agent's action is possible per time step – as in scaring – the integer reflects the most recently acting agent). We anticipate that future versions of gmse will contain multiple resource types, and might add rows to include additional resource traits.

3. observation_array

The `observation_array` (also accessible as `OBSERVATION`) is a two dimensional array, the number of rows and columns of which depend on the type of observation being made (i.e., `observe_type`, which can take integer values from 0-3; see the GMSE reference manual for more information about built-in observation types that are available in GMSE). The first 20 columns of `observation_array` contain the same individual resource traits as in `resource_array`, while any additional columns provide information about how and when a resource was observed. The number of rows in `observation_array` is always equal to or less than that of `resource_array`; each resource that is observed at least once is placed into one unique row, while unobserved resources are not included as rows in the `observation_array`. In `sim`, there are 54 rows, meaning that 1026 were not observed at all in this time step. Below we print out the first eight rows of the observation array.

```
print(sim$observation_array[1:8,]);
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]   10    1    0    0   27   53   20    1    0   0.3     0     5     1
## [2,]   16    1    0    0   30   45   20    1    0   0.3     1     4     1
## [3,]   59    1    0    0   15   50   20    1    0   0.3     0     3     1
## [4,]  118    1    0    0   16   37   20    1    0   0.3     0     2     1
## [5,]  133    1    0    0   32   52   20    1    0   0.3     0     3     1
## [6,]  150    1    0    0   20   37   20    1    0   0.3     0     2     1
## [7,]  172    1    0    0   19   36   20    1    0   0.3     2     2     1
## [8,]  180    1    0    0   22   55   20    1    0   0.3     0     5     1
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
## [1,]     0   0.5     0     0     0     0     0     0     1
## [2,]     0   0.5     0     0     0     0     0     0     1
## [3,]     0   0.5     0     0     0     0     0     0     1
## [4,]     0   0.5     0     0     0     0     0     0     1
## [5,]     0   0.5     0     0     0     0     0     0     1
## [6,]     0   0.5     0     0     0     0     0     0     1
## [7,]     0   0.5     0     0     0     0     0     0     1
## [8,]     0   0.5     0     0     0     0     0     0     1
```

In the case of the default parameters, the observation array has only two additional columns; the first added column 21 is currently unused, and all values in this column are zero. The second added column 22 contains a value of 1 confirming that the resource was observed. Additional options will add different numbers of columns with different values. For example, when `observe_type = 0` (managers observe all resources on a random subset of the landscape, the size of which is determined by their distance of vision) but `times_observe > 1`, managers sample more than one random subset of the landscape. A new column is added for each sampled subset, and a 1 is placed in the relevant column if the resource is observed (these collected data are then used to estimate population size). An example where `times_observe = 4` is shown below.

```
sim_t0_4 <- gmse_apply(get_res = "Full", times_observe = 4);
print(sim_t0_4$observation_array[1:8,]);
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    6    1    0    0   69   76   20    1    0   0.3     0     3     1
## [2,]   12    1    0    0   55    3   20    1    0   0.3     1     4     1
## [3,]   14    1    0    0   37   33   20    1    0   0.3     0     4     1
## [4,]   15    1    0    0   40   33   20    1    0   0.3     0     4     1
## [5,]   18    1    0    0   50   27   20    1    0   0.3     0     4     1
## [6,]   26    1    0    0   60    2   20    1    0   0.3     1     5     1
## [7,]   35    1    0    0   60   19   20    1    0   0.3     0     5     1
## [8,]   38    1    0    0   64   19   20    1    0   0.3     0     2     1
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]     0   0.5     0     0     0     0     0     0     1     0     0
## [2,]     0   0.5     0     0     0     0     0     0     0     0     0
## [3,]     0   0.5     0     0     0     0     0     0     0     1     0
## [4,]     0   0.5     0     0     0     0     0     0     0     1     0
## [5,]     0   0.5     0     0     0     0     0     0     0     1     0
## [6,]     0   0.5     0     0     0     0     0     0     0     0     0
## [7,]     0   0.5     0     0     0     0     0     0     0     0     0
## [8,]     0   0.5     0     0     0     0     0     0     0     0     0
##      [,25]
## [1,]     0
## [2,]     1
## [3,]     0
```

```
## [4,]     0
## [5,]     0
## [6,]     1
## [7,]     1
## [8,]     1
```

This process simulates the data collection of resources (and potentially resource trait measurements) as might be performed by observers within the system. It therefore takes a virtual ecologist approach; this enables the integration of theory and empirical work and can improve the mechanistic understanding of social-ecological systems (Zurell et al., 2010).

4. manager_array

For context, it might be easier to understand `manager_array` after reading about `user_array` below. The `manager_array` (also accessible as `COST`) is a three dimensional array, each layer of which corresponds to a unique agent (rows of agents correpond to layers of `manager_array`). Hence, in the simulation output `sim$manager_array`, there are 5 layers. Each layer in `manager_array` has 13 columns, and a number of rows that varies depending on the number of agents and resource types. As of GMSE v0.4.0.3, only the first three rows are used. Two layers of `sim$manager_array` are shown below, the first being that of the manager and the second being that of the first user.

```
print(sim$manager_array[,,1:2]);
```

```
## , , 1
##
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
## [1,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [2,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [3,] 100001 100001 100001 100001 100001 100001 100001 100001     10 100001
## [4,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [5,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [6,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [7,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
##       [,11]  [,12]  [,13]
## [1,] 100001 100001     10
## [2,] 100001 100001     10
## [3,] 100001 100001     10
## [4,] 100001 100001 100001
## [5,] 100001 100001 100001
## [6,] 100001 100001 100001
## [7,] 100001 100001 100001
##
## , , 2
##
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
## [1,] 100001 100001 100001 100001 100001 100001 100001 100001     54 100001
## [2,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [3,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [4,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [5,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [6,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
## [7,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
##       [,11]  [,12]  [,13]
## [1,] 100001 100001     10
```

```
## [2,] 100001 100001     10
## [3,] 100001 100001 100001
## [4,] 100001 100001 100001
## [5,] 100001 100001 100001
## [6,] 100001 100001 100001
## [7,] 100001 100001 100001
```

Each element in the array indicates the cost of performing a particular action. In the code, this is the cost of changing an element in `user_array` (which has the same dimensions as `manager_array`). The minimum value in `sim$manager_array` is therefore 10, reflecting the default `minimum_cost` value of 10. The maximum value is 100001, which is one higher than the maximum possible manager or user budget. Where cost is 100001, actions can therefore never be performed. An explanation of the rows and columns of `manager_array` is provided below in the description of `user_array`.

5. user_array

When considering the three dimensional `user_array` (also accessible as `ACTION`), it is helpful to keep in mind that each layer corresponds to the actions of a particular agent, that each column corresponds to a particular type of action, and that each row corresponds to a particular resource, agent, or group that the action will affect. The cost of performing any action in this array is held in `manager_array`, wherein an action's cost in `manager_array` is held in the same array element as the action itself in `user_array`. Recall from the manager array that the first layer of `user_array` corresponds to the manager actions, and that remaining layers correspond to user actions; there are therefore as many layers in `user_array` as there are agents in the model, and each row of `AGENTS` corresponds to equivalent layer of `user_array` (e.g., the manager agent, ID = 1, is in the first row of `AGENTS` and the first layer of `user_array`). The first two layers of `user_array` are shown below.

```
print(sim$user_array[,,1:2]);
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]      [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,]   -2    1    0    0 1000.0000    0    0    0    0     0     0     0
## [2,]   -1    1    0    0    0.0000    0    0    0    0     0     0     0
## [3,]    1    1    0    0 -224.4898    0    0   10   54    10    10    10
## [4,]    2    1    0    0    0.0000    0    0    0    0     0     0     0
## [5,]    3    1    0    0    0.0000    0    0    0    0     0     0     0
## [6,]    4    1    0    0    0.0000    0    0    0    0     0     0     0
## [7,]    5    1    0    0    0.0000    0    0    0    0     0     0     0
##      [,13]
## [1,]     0
## [2,]     0
## [3,]    66
## [4,]     0
## [5,]     0
## [6,]     0
## [7,]     0
##
## , , 2
##
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]   -2    1    0    0   -1    0    0    0   18     0     0     0     0
## [2,]   -1    1    0    0    0    0    0    0    0     0     0     0     0
## [3,]    1    1    0    0    0    0    0    0    0     0     0     0     0
```

```
## [4,]    2    1    0    0    0    0    0    0    0    0    0    0    0
## [5,]    3    1    0    0    0    0    0    0    0    0    0    0    0
## [6,]    4    1    0    0    0    0    0    0    0    0    0    0    0
## [7,]    5    1    0    0    0    0    0    0    0    0    0    0    0
```

Note that there are more columns in this array than there are possible actions in GMSE. This is because there are several actions that are not actions per se, but properties of agents. As of GMSE v0.4.0.3, these properties cannot be changed by other agents. Column of `user_array` are as follows.

1. The type of agent or resource being affected by an action. A value of -2 indicates that actions have a direct effect on a resource (e.g., scaring, culling, etc.). A value of -1 indicates that actions have a direct effect on a landscape layer. Positive integer values indicate actions that affect other agents, where each integer corresponds to the agents' IDs. Where the integer value is identical with the agent's own ID (e.g., row 3 in layer 1 where the element `sim$user_array[3, 1, 1] = 1`), actions affect all other agents in the model. As of GMSE v0.4.0.3, all rows except 1-3 are unused because agents do not affect one anothers actions individually; they either affect all other agents' actions indiscriminately (in the case of the manager setting policy) or do not affect other agents' actions at all (in the case of users). This data structure, however, is designed so that future versions of GMSE will allow users to affect one another directly (representing, e.g., different groups of agents lobbying for different interests, among-user conflict, etc.).
2. Type 1 of the agent or resource of interest (in practice, this is currently unused).
3. Type 2 of the agent or resource of interest (currently unused).
4. Type 3 of the agent or resource of interest (currently unused).
5. Utility associated with the recipient of the action. For example, in the case of the resource (row 1), positive values indicate that the agent wants more of these resources, while negative values indicate that the agent wants fewer. In the case of the manager (layer 1), the value in the first row is `manage_target`, while the value in the third row is the change in resource number needed to achieve target value (i.e., `manage_target =` 1000, and the manager's estimate is `sim$observation_vector =` 1224.4897959. The former minus the latter is -224.4898).
6. Whether or not the utility associated with the recipient of the action is dependent upon that recipient being on land owned by the actor (e.g., if users only care about resources on landscape cells that they own, then this value is 1 instead of 0).
7. Whether or not actions on the recipient are possible if the recipient is not on land owned by the actor (e.g., if users cannot cull resources that are not on their own land, then this value is 1 instead of 0).

References

Zurell, D., Berger, U., Cabral, J. S., Jeltsch, F., Meynard, C. N., Münkemüller, T., Nehrbass, N., Pagel, J., Reineking, B., Schröder, B., and Grimm, V. (2010). The virtual ecologist approach: Simulating data and observers. *Oikos*, 119(4):622–635.