

# Default GMSE data structures

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

*A. Bradley Duthie<sup>1,3</sup>, Jeremy J. Cusack<sup>1</sup>, Isabel L. Jones<sup>1</sup>, Jeroen Minderman<sup>1</sup>, Erlend B. Nilsen<sup>2</sup>, Rocío A. Pozo<sup>1</sup>, O. Sarobidy Rakotonarivo<sup>1</sup>, Bram Van Moorter<sup>2</sup>, and Nils Bunnefeld<sup>1</sup>*

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

## The most important (default) GMSE data structures

The default sub-models 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 sub-models do not necessarily need to be used in every run in GMSE (see [use of gmse\\_apply](#)), they will be used in any run of the `gmse` function, and in any call of the `gmse_apply` function that does not run with entirely custom sub-models. Simulation and model inference do 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 [SI4](#)), or build custom individual-based sub-models. Here we provide a brief explanation of the following key data structures (each name below is listed as it is 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 be easily modified in GMSE v0.4.0.3 (see, e.g., [SI4](#)), 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 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 `resource_array` 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 sum of the number of manager and users (i.e., each row is an individual agent).

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

```
39 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
40 ## [1,]    1    0    0    0    91    76    50    0    10     0    48     0     0
41 ## [2,]    2    1    0    0   100    78    50    0    10     0     0     0     0
42 ## [3,]    3    1    0    0    33    48    50    0    10     0     0     0     0
43 ## [4,]    4    1    0    0    83    47    50    0    10     0     0     0     0
44 ## [5,]    5    1    0    0    81    92    50    0    10     0     0     0     0
45 ##      [,14] [,15]      [,16] [,17]
46 ## [1,]      0      0 9520.625 1000
47 ## [2,]      0      0   0.000 1000
48 ## [3,]      0      0   0.000 1000
49 ## [4,]      0      0   0.000 1000
50 ## [5,]      0      0   0.000 1000
```

In the default case above, there are five agents (one manager and four users), 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 sub-models 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 1121 rows, so we only print out the first eight for illustration.

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

```
79 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
80 ## [1,]    1    1    0    0    65    10    20    1    0    0.3     0     3     0
81 ## [2,]    2    1    0    0    48    66    20    1    0    0.3     1     3     0
82 ## [3,]    3    1    0    0    64    56    20    1    0    0.3     1     5     0
```

```

83 ## [4,]      4      1      0      0     79      7     20      1      0     0.3      0      2      0
84 ## [5,]      5      1      0      0     42     72     20      1      0     0.3      0      2      0
85 ## [6,]      6      1      0      0     41     32     20      1      0     0.3      0      4      0
86 ## [7,]      7      1      0      0     65     84     20      1      0     0.3      0      2      0
87 ## [8,]      8      1      0      0     12     93     20      1      0     0.3      0      5      0
88 ##      [,14] [,15] [,16] [,17] [,18] [,19] [,20]
89 ## [1,]      0     0.5      0      0      0      0      0
90 ## [2,]      0     0.5      0      0      0      0      0
91 ## [3,]      0     0.5      0      0      0      0      0
92 ## [4,]      0     0.5      0      0      0      0      0
93 ## [5,]      0     0.5      0      0      0      0      0
94 ## [6,]      0     0.5      0      0      0      0      0
95 ## [7,]      0     0.5      0      0      0      0      0
96 ## [8,]      0     0.5      0      0      0      0      0

```

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

- 98 1. ID (each resource gets a unique number)
- 99 2. Type 1 (currently all resources are of type 1)
- 100 3. Type 2 (currently unused)
- 101 4. Type 3 (currently unused)
- 102 5. x-location on the landscape
- 103 6. y-location on the landscape
- 104 7. Movement distance
- 105 8. Time parameter (typically ignored)
- 106 9. Density-independent removal (i.e., death) probability
- 107 10. Growth (i.e., birth) probability
- 108 11. Offspring produced
- 109 12. Age (initial resources are given a random age between 1 and the maximum age sampled from a uniform
- 110 distribution; offspring always start at age zero in their time step of birth)
- 111 13. Marking indicator (used in the observation function)
- 112 14. Tallying indicator (used in the observation function)
- 113 15. Proportion of a landscape cell the resource consumes in a time step
- 114 16. Has the resource been scared by an agent?
- 115 17. Has the resource been culled by an agent?
- 116 18. Has the resource been castrated by an agent?
- 117 19. Has the resource's growth rate been increased by an agent?
- 118 20. Has the resource's offspring production been increased by an agent?

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

### 125 3. observation\_array

126 The **observation\_array** (also accessible as **OBSERVATION**) is a two dimensional array, the number of rows  
127 and columns of which depend on the type of observation being made (i.e., **observe\_type**, which can take  
128 integer values from 0-3; see the [GMSE reference manual](#) for more information about built-in observation  
129 types that are available in GMSE). The first 20 columns of **observation\_array** contain the same individual  
130 resource traits as in **resource\_array**, while any additional columns provide information about how and when  
131 a resource was observed. The number of rows in **observation\_array** is always equal to or less than that of

132 `resource_array`; each resource that is observed at least once is placed into one unique row, while unobserved  
133 resources are not included as rows in the `observation_array`. In `sim`, there are 48 rows, meaning that 1073  
134 resources were not observed at all in this time step. Below, we print out the first eight rows of the observation  
135 array.

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

```
136 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
137 ## [1,]   77    1    0    0   93   72   20    1    0  0.3    0    3    1
138 ## [2,]   80    1    0    0   96   73   20    1    0  0.3    1    2    1
139 ## [3,]   83    1    0    0   92   84   20    1    0  0.3    0    2    1
140 ## [4,]  111    1    0    0   86   85   20    1    0  0.3    0    2    1
141 ## [5,]  124    1    0    0   89   72   20    1    0  0.3    0    2    1
142 ## [6,]  187    1    0    0   82   82   20    1    0  0.3    0    3    1
143 ## [7,]  200    1    0    0   86   84   20    1    0  0.3    0    2    1
144 ## [8,]  225    1    0    0   84   86   20    1    0  0.3    0    4    1
145 ##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
146 ## [1,]      0  0.5    0    0    0    0    0    0    1
147 ## [2,]      0  0.5    0    0    0    0    0    0    1
148 ## [3,]      0  0.5    0    0    0    0    0    0    1
149 ## [4,]      0  0.5    0    0    0    0    0    0    1
150 ## [5,]      0  0.5    0    0    0    0    0    0    1
151 ## [6,]      0  0.5    0    0    0    0    0    0    1
152 ## [7,]      0  0.5    0    0    0    0    0    0    1
153 ## [8,]      0  0.5    0    0    0    0    0    0    1
```

154 In the case of the default parameters, the observation array has only two additional columns; the first added  
155 column 21 is currently unused, and all values in this column are zero. The second added column 22 contains a  
156 value of 1 confirming that the resource was observed. Additional options will add different numbers of columns  
157 with different values. For example, when `observe_type = 0` (managers observe all resources on a random  
158 subset of the landscape, the size of which is determined by their distance of vision) but `times_observe > 1`,  
159 managers sample more than one random subset of the landscape. A new column is added for each sampled  
160 subset, and a 1 is placed in the relevant column if the resource is observed (these collected data are then used  
161 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,]);
```

```
162 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
163 ## [1,]    8    1    0    0    7   55   20    1    0  0.3    1    4    1
164 ## [2,]   16    1    0    0   86   43   20    1    0  0.3    0    3    1
165 ## [3,]   29    1    0    0   94   42   20    1    0  0.3    0    3    1
166 ## [4,]   41    1    0    0   71   33   20    1    0  0.3    1    2    1
167 ## [5,]   53    1    0    0    7   63   20    1    0  0.3    0    4    1
168 ## [6,]   56    1    0    0   61    8   20    1    0  0.3    0    4    1
169 ## [7,]   62    1    0    0   63   42   20    1    0  0.3    0    4    1
170 ## [8,]   68    1    0    0   66   28   20    1    0  0.3    0    3    1
171 ##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
172 ## [1,]      0  0.5    0    0    0    0    0    0    1    0    0
173 ## [2,]      0  0.5    0    0    0    0    0    0    0    1    0
174 ## [3,]      0  0.5    0    0    0    0    0    0    0    1    0
175 ## [4,]      0  0.5    0    0    0    0    0    0    0    0    1
176 ## [5,]      0  0.5    0    0    0    0    0    0    1    0    0
177 ## [6,]      0  0.5    0    0    0    0    0    0    0    0    0
178 ## [7,]      0  0.5    0    0    0    0    0    0    0    0    1
179 ## [8,]      0  0.5    0    0    0    0    0    0    0    0    1
```

```

180 ##      [,25]
181 ## [1,]      0
182 ## [2,]      0
183 ## [3,]      0
184 ## [4,]      0
185 ## [5,]      0
186 ## [6,]      1
187 ## [7,]      0
188 ## [8,]      0

```

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

## 193 4. manager\_array

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

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

```

201 ## , , 1
202 ##
203 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
204 ## [1,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
205 ## [2,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
206 ## [3,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
207 ## [4,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
208 ## [5,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
209 ## [6,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
210 ## [7,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
211 ##      [,11] [,12] [,13]
212 ## [1,] 100001 100001 10
213 ## [2,] 100001 100001 10
214 ## [3,] 100001 100001 10
215 ## [4,] 100001 100001 100001
216 ## [5,] 100001 100001 100001
217 ## [6,] 100001 100001 100001
218 ## [7,] 100001 100001 100001
219 ##
220 ## , , 2
221 ##
222 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
223 ## [1,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
224 ## [2,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
225 ## [3,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
226 ## [4,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
227 ## [5,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001

```

```

228 ## [6,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
229 ## [7,] 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001
230 ##      [,11] [,12] [,13]
231 ## [1,] 100001 100001      10
232 ## [2,] 100001 100001      10
233 ## [3,] 100001 100001 100001
234 ## [4,] 100001 100001 100001
235 ## [5,] 100001 100001 100001
236 ## [6,] 100001 100001 100001
237 ## [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 allowed manager or user budget. Where a 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

The `user_array` (also accessible as `ACTION`) is a three dimensional array, each layer of which corresponds to a unique agent. When considering the three dimensional `user_array`, 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]);
```

```

256 ## , , 1
257 ##
258 ##      [,1] [,2] [,3] [,4]      [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
259 ## [1,]   -2    1    0    0 1000.00000    0    0    0    0    0    0    0
260 ## [2,]   -1    1    0    0   0.00000    0    0    0    0    0    0    0
261 ## [3,]    1    1    0    0 -88.43537    0    0   10   67   10   10   10
262 ## [4,]    2    1    0    0   0.00000    0    0    0    0    0    0    0
263 ## [5,]    3    1    0    0   0.00000    0    0    0    0    0    0    0
264 ## [6,]    4    1    0    0   0.00000    0    0    0    0    0    0    0
265 ## [7,]    5    1    0    0   0.00000    0    0    0    0    0    0    0
266 ##      [,13]
267 ## [1,]      0
268 ## [2,]      0
269 ## [3,]     53
270 ## [4,]      0
271 ## [5,]      0
272 ## [6,]      0
273 ## [7,]      0
274 ##
275 ## , , 2

```

```

276 ##
277 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
278 ## [1,]   -2    1    0    0   -1    0    0    0    0   14    0    0    0    0
279 ## [2,]   -1    1    0    0    0    0    0    0    0    0    0    0    0    2
280 ## [3,]    1    1    0    0    0    0    0    0    0    0    0    0    0    0
281 ## [4,]    2    1    0    0    0    0    0    0    0    0    0    0    0    0
282 ## [5,]    3    1    0    0    0    0    0    0    0    0    0    0    0    0
283 ## [6,]    4    1    0    0    0    0    0    0    0    0    0    0    0    0
284 ## [7,]    5    1    0    0    0    0    0    0    0    0    0    0    0    0

```

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

- 288 1. The type of agent or resource being affected by an action. A value of -2 indicates that actions have  
289 a direct effect on a resource (e.g., scaring, culling, etc.). A value of -1 indicates that actions have a  
290 direct effect on a landscape layer. Positive integer values indicate actions that affect other agents, where  
291 each integer corresponds to the agents' IDs. Where the integer value is identical with the agent's own  
292 ID (e.g., row 3 in layer 1 where the element `sim$user_array[3, 1, 1] = 1`), actions affect all other  
293 agents in the model. As of GMSE v0.4.0.3, all rows except 1-3 are unused because agents do not affect  
294 one another's actions individually; they either affect all other agents' actions indiscriminately (in the  
295 case of the manager setting policy) or do not (directly) affect other agents' actions at all (in the case of  
296 users). This data structure, however, is designed so that future versions of GMSE will allow users to  
297 affect one another directly (representing, e.g., different groups of agents lobbying for different interests,  
298 among-user conflict, etc.).
- 299 2. Type 1 of the agent or resource of interest (in practice, this is currently unused).
- 300 3. Type 2 of the agent or resource of interest (currently unused).
- 301 4. Type 3 of the agent or resource of interest (currently unused).
- 302 5. Utility associated with the recipient of the action. For example, in the case of the resource (row 1),  
303 positive values indicate that the agent wants more of these resources, while negative values indicate  
304 that the agent wants fewer. In the case of the manager (layer 1), the value in the first row equals  
305 `manage_target`, while the value in the third row is the change in resource number needed to achieve  
306 the target value (i.e., `manage_target = 1000`, and the manager's estimate is `sim$observation_vector`  
307 `= 1088.4353741`. The former minus the latter is -88.43537).
- 308 6. Whether or not the utility associated with the recipient of the action is dependent upon that recipient  
309 being on land owned by the actor (e.g., if users only care about resources on landscape cells that they  
310 own, then this value is 1 instead of 0).
- 311 7. Whether or not actions on the recipient are possible if the recipient is not on land owned by the actor  
312 (e.g., if users cannot cull resources that are not on their own land, then this value is 1 instead of 0).
- 313 8. The number of actions performed for scaring, which in row 3 of the manager layer 1 is interpreted as  
314 the scaring cost set by the manager for users.
- 315 9. The number of actions performed for culling, which in row 3 of the manager layer 1 is interpreted as  
316 the culling cost set by the manager for users.
- 317 10. The number of actions performed for castration, which in row 3 of the manager's layer 1 is interpreted  
318 as the castration cost set by the manager for users. Further, in row 2 for users (where column 1 equals  
319 -1), this value is instead the number of `tend_crop` actions (the number of cells on which crops are  
320 tended by users, which always is performed on users' own land, cannot be affected by the manager, and  
321 always equals `minimum_cost`).
- 322 11. The number of actions performed for feeding resources (increasing their growth rate, `lambda`), which in  
323 row 3 of the manager's layer 1 is interpreted as the feeding cost set by the manager for users. Further,  
324 in row 2 for users (where column 1 equals -1), this value is instead the number of `kill_crop` actions  
325 (the number of cells on which crops are destroyed by users, which always is performed on users' own  
326 land, cannot be affected by the manager, and always equals `minimum_cost`).
- 327 12. The number of actions performed for helping resource offspring (directly increasing offspring production),  
328 which in row 3 of the manager's layer 1 is interpreted as the helping offspring cost set by the manager

329       for users.

330   13. The number of actions unspent by the user or manager; any actions allocated to this row do nothing.

331       These may be used when any action would lead the agent to a less than desirable outcome, such as if

332       only culling exists as a policy option (default), but managers do not want to increase the cost of culling

333       because resource density is above `manage_target`.

334 In the [genetic algorithm](#), values in elements of a `user_array` layer are potentially modified according to each

335 agent's objective, as constrained by costs in `manager_array`.

## 336 6. LAND

337 Events in default GMSE sub-models occur on a spatially-explicit landscape `LAND`, which is stored as a three

338 dimensional array. The size of this landscape is specified with the `land_dim_1` and `land_dim_2` arguments of

339 GMSE, which determine the length, in cells, of the y and x dimensions of the landscape, respectively (e.g., if

340 `land_dim_1 = 10` and `land_dim_2 = 1000`, then the landscape will be one very long horizontal transect).

341 The total number of landscape cells on which resources and agents can interact is therefore the product of

342 `land_dim_1` and `land_dim_2`. In addition, all landscapes have three layers, which hold three separate values

343 of information for each x-y location. The first layer is unused in GMSE v0.4.0.3; the second layer holds crop

344 production on a cell, and the third layer holds the owner of the cell (corresponding to the ID of an agent,

345 where the manager's ID = 0 defines public land). An  $8 \times 8$  portion of the landscape from `sim` is shown below.

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

```
346 ## , , 1
347 ##
348 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
349 ## [1,]    1    1    1    1    1    1    1    1
350 ## [2,]    1    1    1    1    1    1    1    1
351 ## [3,]    1    1    1    1    1    1    1    1
352 ## [4,]    1    1    1    1    1    1    1    1
353 ## [5,]    1    1    1    1    1    1    1    1
354 ## [6,]    1    1    1    1    1    1    1    1
355 ## [7,]    1    1    1    1    1    1    1    1
356 ## [8,]    1    1    1    1    1    1    1    1
357 ##
358 ## , , 2
359 ##
360 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
361 ## [1,] 1.00    1 1.000    1    1 1.0 1.0 1.0
362 ## [2,] 1.00    1 1.000    1    1 1.0 1.0 1.0
363 ## [3,] 1.00    1 1.000    1    1 1.0 1.0 1.0
364 ## [4,] 1.00    1 1.000    1    1 1.0 1.0 0.5
365 ## [5,] 1.00    1 1.000    1    1 1.0 1.0 1.0
366 ## [6,] 0.50    1 1.000    1    1 0.5 1.0 0.5
367 ## [7,] 1.00    1 0.125    1    1 1.0 0.5 1.0
368 ## [8,] 0.25    1 1.000    1    1 1.0 0.5 1.0
369 ##
370 ## , , 3
371 ##
372 ##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
373 ## [1,]    1    1    1    1    1    1    1    1
374 ## [2,]    1    1    1    1    1    1    1    1
375 ## [3,]    1    1    1    1    1    1    1    1
376 ## [4,]    1    1    1    1    1    1    1    1
```



```

377 ## [5,] 1 1 1 1 1 1 1 1
378 ## [6,] 1 1 1 1 1 1 1 1
379 ## [7,] 1 1 1 1 1 1 1 1
380 ## [8,] 1 1 1 1 1 1 1 1

```

381 In the case of the above, all of the cells in this square patch of landscape are owned by agent 1 (i.e., the  
382 manager; see `sim$LAND[,3]`), and we can see that crop production on this patch of land has been decreased  
383 from 1 in several cells as a consequence of consumption by resources (see `sim$LAND[,2]`). In [SI4](#), we show  
384 how landscape cell values can be manipulated to customise the placement of land ownership.

## 385 Conclusions

386 We have focused on the data structures [AGENTS](#), [resource\\_array](#), [observation\\_array](#), [manager\\_array](#),  
387 [user\\_array](#), and [LAND](#) because these are the data structures that can be most readily manipulated to  
388 customise GMSE simulations. An example of how to do this within a loop using `gmse_apply` can be found in  
389 [SI4](#). While other data structures exist within GMSE (e.g., see the output of `gmse_apply` when `get_res =`  
390 `Full`), we do not recommend manipulating these structures for custom simulations.

391 Many data structures contain elements that are unused in GMSE v0.4.0.3, and in all cases this is designed  
392 for ease of ongoing development of new GMSE features. Requests for new features can be made on GitHub  
393 using the [GMSE Wiki](#) or the [GMSE Issues](#) page.

## 394 References

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