

Breakout Reporter

EcoSwarm software developed by:
Steve Railsback, Lang Railsback & Assoc. (LRA@Northcoast.com)
Steve Jackson, Jackson Scientific Computing (jackson3@humboldt1.com)

Overview and Conventions

BreakoutReporter is an Objective C class that reports statistics on a list of objects in a Swarm model, with the output broken out by up to five variables and written to a file. (The model objects are referred to here as *agents*, even though they can be any objects meeting the conventions defined below.) For example, an animal model could use the BreakoutReporter to periodically write to an output file the number of animals and their mean weight, broken out by species, sex, and age. Additional output columns can be added to report model variables such as the simulation date.

BreakoutReporter uses a “breakout map”—a structure of five nested Swarm maps to sort the objects by the breakout variables. The breakout variables are defined during the create phase. One copy of the breakout map contains Swarm lists, onto which are sorted the agents when the breakoutReporter is updated. Other copies of the breakout map contain Swarm averagers, attached to the lists of agents, that calculate the output statistics.

A breakoutReporter can be updated during a simulation by either (1) adding more agents to the lists for which statistics are reported, or (2) replacing all the agents on these lists. For example, the number of live animals in a model can be reported using a breakoutReporter updated by replacing all the live animal objects (because some that were previously alive are now dead); but the cumulative number of dead animals can be reported using a breakoutReporter updated by adding the newly deceased animal objects (because dead animals stay dead).

The output file is in column format so it can easily be imported into a spreadsheet or data base. The columns are automatically labeled, and a system time stamp is automatically written at the top of the output file.

The following conventions are used.

- Between zero and five variables can be used to break out the statistics. If zero breakout variables are defined, statistics are reported on the whole list of objects.
- The agent variables used to break out statistics must contain an object, which is used as a key on the breakout maps. This “breakout key” is normally a Swarm symbol, but it can be any object that has a getName method like that of a symbol. The name returned by this getName method is used to label columns in the output file.
- All statistical output is written in floating point format. (In constrast, output generated via the `addDataColumnWithLabel` method has a format appropriate for the data type.)

Files

The code archive `BreakoutReporter.tgz` contains Objective C code for four classes, each of which must be included in a model's makefile. These are:

- `BreakoutReporter`;
- `BreakoutAverager` (a subclass of Swarm's `Averager`);
- `BreakoutMessageProbe` (a subclass of Swarm's `MessageProbe`); and
- `BreakoutVarProbe` (a subclass of Swarm's `VarProbe`);

Creating

Creating a `breakoutReporter` requires (1) a `createBegin` message, (2) between zero and five `breakOutUsingSelector` messages, (3) zero or more `createOutputWithLabel` messages, (4) zero or more `addColumnWithLabel` messages, and finally (5) a `createEnd` message. The `createBegin` and `createEnd` messages must start and end the create process, but the other messages can be called in any sequence.

```
+ createBegin: (id <Zone>) aZone
    forList: (id <List>) aList
    withOutputFilename: (char *) aFileName
    withFileOverwrite: (bool) aBool
    withColumnWidth: (int) aColumnWidth
```

Creates and returns a `breakoutReporter` object. Sets the list (`aList`) of agents for which statistics will be reported. Provides the name of the output file and determines whether the output file will be overwritten (`aBool = YES`) or appended (`aBool = NO`). The parameter `aColumnWidth` sets the number of characters in each column of output; this parameter should be set so that it is longer than the breakout key objects's names, which are used as column labels. The file name `aFileName` is limited to 50 characters.

```
- breakOutUsingSelector: (SEL) aBreakoutVariableSelector
    withListOfKeys: (id <List>) aListOfKeys
```

Defines one variable by which statistics are broken out. Each agent must have a method `aBreakoutVariableSelector` that returns an object that is used as a key to sort the agent onto the breakout map. Normally this breakout variable object is a Swarm symbol, but any object having a `getName` method like that of Swarm's `Symbol` class (returning a constant `char`) will work. The breakout map is defined for this variable by `aListOfKeys`; each object on `aListOfKeys` is used as a category to break out the agents.

The output file's column labels are generated automatically by executing the `getName` method of each object on `aListOfKeys`. Therefore, any object used as a key must have its name set for column labels to be generated correctly.

For example, each agent may have a method `getSpeciesSymbol` that returns a Swarm symbol representing the agent's species. Symbols for all species are on the model's `listOfSpeciesSymbols`. Then, `[breakoutReporter breakOutUsingSelector:`

@selector(getSpeciesSymbol) withListOfKeys: listOfSpeciesSymbols] will cause output to be broken out by species.

If aBreakoutVariableSelector for an agent returns an object that is not on aListOfKeys, then the agent is left off the breakout map and is not included in the reported output.

This method can be used between zero and five times. (Attempting to add more than five breakout levels raises an error message.) The breakout hierarchy follows the order in which new breakout variables are added by executing this method.

```
- createOutputWithLabel: (char *) anOutputLabel
  withSelector: (SEL) anOutputSelector
  withAveragerType: (char *) anAveragerType
```

Defines one agent variable for which statistics are reported, by defining a Swarm averager. (See the Swarm documentation for Averager, in the Analysis library.) The columns containing this output are labeled using anOutputLabel, which should have fewer characters than the output file column width (if not, an error message to that effect is issued). The agents must have a method anOutputSelector that returns the value of the variable being reported (if not, an error message is issued). The type of statistic to be reported is defined by anAveragerType; valid values are:

- “Count” (the number of agents),
- “Average” (the mean among agents of values returned by anOutputSelector),
- “Total” (the total of values returned by anOutputSelector),
- “Min” (the minimum), and
- “Max” (the maximum).

If anAveragerType is set to “Count”, then the value returned by anOutputSelector does not affect results (the averager simply calculates the number of agents). However, the Swarm Averager still requires anOutputSelector to be a valid selector that returns a numerical value (not an object).

This method may be used any number of times, including zero. Each use creates a new set of output columns, broken out according to the breakOutUsingSelector messages. If this method is not used, only output generated by the **addDataColumnWithLabel** method will be reported.

As an example, if the number of agents in each breakout category is to be reported, then this message would be used:

```
[breakoutReporter createOutputWithLabel: "NumberOfAgents"
  withSelector: @selector(getSpecies)
  withAveragerType: "Count"];
```

If the mean weight of agents is to be reported, then this message would be used:

```
[breakoutReporter createOutputWithLabel: "MeanWeight"
```

```
withSelector: @selector(getWeight)
withAveragerType: "Average"];
```

Finally, in this example output is obtained for a boolean variable: the value returned by selector `ateLastStep` is YES (1) if the agent succeeded in feeding and otherwise NO (0). The mean of this variable over all agents is equivalent to the fraction of agents that succeeded in feeding:

```
[breakoutReporter createOutputWithLabel: "FractionThatAte"
 withSelector: @selector(ateLastStep)
 withAveragerType: "Average"];
```

```
- addColumnWithValueOfVariable: (const char *) aVariable
   fromObject: (id) aDataObject
   withType: (char *) aDataType
   withLabel: (char *) aDataLabel
```

This method adds a single column to the output file, containing data unrelated to the breakout analysis. Its purpose is add useful, single variables to the output file; examples include the simulation date or time, and habitat variables in ecological models. The data column is labeled with the value of `aDataLabel`, which should have fewer characters than the breakoutReporter's column width. The value reported in the column is obtained from `aDataObject` via a `VarProbe` that returns the current value of the variable `aVariable`. The value of `aDataType` is used by the `VarProbe` and to format the column. Valid values are "id" (for object pointers); "double" (for single or double-precision floating point variables), "int" (for integers or long integers), and "string" (for character strings). If the type of `aVariable` does not match the value of `aDataType`, an error condition will result.

An example is adding output for the model date, in character string format. The model date is obtained from an object called the `timeManager`.

```
[breakoutReporter addColumnWithValueOfVariable: "modelDate"
 fromObject: timeManager
 withType: "string"
 withLabel: "Date"];
```

```
- addColumnWithValueFromSelector: (SEL) aDataSelector
   fromObject: (id) aDataObject
   withType: (char *) aDataType
   withLabel: (char *) aDataLabel
```

This method is similar to **addColumnWithValueOfVariable** except that the value reported is obtained via a message probe to `aDataSelector`. This allows output to be obtained by executing a method. The valid values of `aDataType` are the same as for **addColumnWithValueOfVariable** *except* that "long" must be used instead of "int".

```
- suppressColumnLabels: (bool) aBool
```

This optional method keeps the column labels from being written if the value of `aBool` is NO. By default, column labels are written when **createEnd** is executed. However, suppressing

column labels is useful when `withFileOverwrite` is set to NO in `createBegin` (so output is appended to an existing file). These options allow multiple model runs to create a single output file that does not have column labels repeated in it.

- `createEnd`

The `createEnd` message follows all `breakOutUsingSelector`, `createOutputWithLabel`, and `addDataColumnWithLabel` messages. This method completes creation of the `breakoutReporter` by building the breakout maps, and writes the output file's header lines. However, it does not populate the breakout map; an `update` method must be executed before output can be generated.

Using

Writing new lines to the `breakoutReporter`'s output file requires (1) using one of two update methods, then (2) using the `output` method. The update and output methods are separate so they can be used, if desired, with different frequencies. For example, the `updateByAccumulation` method can be used to generate cumulative statistics on agents that are no longer in the model (e.g., dead animals). The `updateByAccumulation` method can be called every model step to accumulate the agents removed from the simulation on that step, while the `output` method is used, for example, only every 10th step to print the total number of accumulated agents.

The update methods check the *first* agent added to the breakout map to make sure it responds to the selector specified in the `createOutputWithLabel` method; if not, an error condition is raised. To reduce execution time, subsequent agents are not checked; if any subsequent agent does not respond to the selector the program will crash without an error message.

- `updateByReplacement`

This method flushes and re-builds the breakout map of agents from which statistics are generated. All agents are removed from the map, and then all the agents currently on `aList` (as specified in the `createBegin` message) are then sorted onto the map.

- `updateByAccumulation`

This method adds all the agents currently on `aList` (as specified in the `createBegin` message) to the breakout map of agents from which statistics are generated. Agents are *not* removed from the map by this method.

- `output`

The `output` method calculates and writes a new line of output to the file. New values are obtained for the data variables specified by `addDataColumnWithLabel` messages. The averages that calculate output statistics are updated, then results written to the file.

- `drop`

The `drop` method closes the output file and drops the `breakoutReporter`. (If `drop` is not called, the output file will be closed when the program exits.)

Example

The following code is from a trout model that uses BreakoutReporter to output trout population statistics.

The trout model swarm maintains a list of all the live fish (`liveFish`). Each fish has Swarm symbol variables representing its species and its age. The list of valid age symbols is `ageSymbolList` and the list of valid species symbols is `speciesSymbolList`. The following model swarm method builds a breakoutReporter to output fish abundance and mean length, broken out by species and age. Data columns are added to report the simulated date and river flow.

```
- buildBreakoutReporters
{
    fileOverWrite = TRUE;

    liveFishReporter = [BreakoutReporter   createBegin: modelZone
                                                forList: liveFish
                                                withOutputFilename: "LiveFishBreakout.Rpt"
                                                withFileOverwrite: fileOverWrite
                                                withColumnWidth: 25];

    [liveFishReporter breakOutUsingSelector: @selector(getSpecies)
                        withListOfKeys: speciesSymbolList];

    [liveFishReporter breakOutUsingSelector: @selector(getAgeSymbol)
                        withListOfKeys: ageSymbolList];

    [liveFishReporter createOutputWithLabel: "Count"
                        withSelector: @selector(getFishLength)
                        withAveragerType: "Count"];

    [liveFishReporter createOutputWithLabel: "AverageFishLength"
                        withSelector: @selector(getFishLength)
                        withAveragerType: "Average"];

    [liveFishReporter addColumnWithValueOfVariable: "modelDate"
                        fromObject: self
                        withType: "string"
                        withLabel: "ModelDate"];

    [liveFishReporter addColumnWithValueOfVariable: "riverFlow"
                        fromObject: (id) habitatSpace
                        withType: "double"
                        withLabel: "RiverFlow"];

    [liveFishReporter createEnd];

    return self;
}
```

Elsewhere in the model swarm, the following two statements are included in actions scheduled at the end of each time step:

```
[liveFishReporter updateByReplacement];  
[liveFishReporter output];
```

Document History

First posted version: SFRailsback, 2/21/2002.

Correction in valid types for addColumn methods: SFRailsback, 3/5/2002.

Check for invalid selector in update methods: SFRailsback, 3/6/2002.

SuppressColumnLabels added: SFRailsback, 6/28/2004.