## The Rhapsody package: core classes for system dynamics modelling

The rhapsody (**R a**pplied **s**ystem **dy**namics) package contains a set of classes or containers for building system dynamics modelling in R. It provides tools to build system dynamics models using input templates, as well as capacity for batch-running, scenario building and parameter selection with approximate Bayesian computing. This document gives an overview of model classes, how to use model templates, and special conventions for returning flow outputs to a model object. **NB: the note on returning flow outputs to the model object (section 3) involves changes to how function return arguments typically work.** Vignettes and help are available within the package (accessed by ?function\_name)).

The classes contained in the package are:

* Stock

A core model class used to hold parameters and values for model stocks. Stocks can hold a combination of spatial and non-spatial inputs and parameters.

* Flow

A flow holds a combination of input data and parameters, a flow function and outputs. It contains an integrated taxonomy for easy passing of information between stocks in a model.

* Input

Information passed to the model at each time step; can be spatial or non-spatial

* Output

Draws information from model at each timestep: holds results

* Model

The model object groups together stocks, flows, inputs, outputs as well as a list of global parameters. The model scheduler determines the order of execution at each timestep. The model can be for a single timestep or for a given number of timesteps specified in the run parameters.

* Experiment

The experiment class is used for batch-running of the model with differing parameter or data inputs. It has associated methods for building scenarios and for parameter selection based on history-matching with observations.

Rhapsody is available to install from the DEFRA Github.

## Building the model structure with the rhapsody package

Defining the structure of the model system can be done either within R or using template input files. Here, we cover how to use the template inputs, though the process and formatting requirements are analogous for direct R implementation.

### Defining model stocks

The stock key has six required columns, as well as any number of user defined columns.

Stocks have two kinds of parameters: parameters, which are defined in relationship with other model components (inputs, flows, global parameters) and internal parameters, which are specific to a given stock, and update from a csv file at each timestep. More information on options for parameterising model stocks is given in section 3: parameterising the model.

The required columns in the stock template, their purpose and the required format for data entry are set out below (Table A1.1), followed by a template for a stock data file (Table A1.2), which is used dynamically by the model to update the stock’s internal parameters.

Table A1.1. Overview of template for defining model stocks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Purpose** | **Required data format** | **Example** |
| s\_type | Defines whether a stock is spatial or non-spatial | One of: “stock” or “spatial\_stock” | spatial\_stock |
| s\_name | Defines stock name | A character string beginning with s\_ and ending with any relevant name | s\_cereal |
| s\_family | Defines the family a stock belongs to: e.g. its land cover class. This can be used by family flows to run a similar flow process between multiple stocks simultaneously | A character string beginning with s\_ and ending with a | s\_cropland |
| s\_id | Unique stock identifier | s followed by a number | s1 |
| s\_parameters | Initialises stock parameter fields that are influenced by other model objects | Relevant parameter names separated by comma space “, ” | Area, price\_per\_Tonne |
| s\_internal\_pars | Initialises stock parameters that are internal to a stock; these are updated at each timestep | Relevant parameter names separated by comma space “, “ | Yield |
| … | Any number of additional columns that specify stock attributes, e.g., commodity produced, or UK region | User defined; one value per column | User defined; one value per column |

Table A1.2. Example model stock data file; each stock data file must be named after the model stock.

|  |  |  |
| --- | --- | --- |
| **Column1** | **Column 2…** | **Column n….** |
| The first column MUST define which model timestep the rows refer to: e.g. 1990 | The remaining columns should be named after a declared stock internal parameter in the stock key | Column name |
| 1991 | Each row of the stock data file should contain a single piece of data referring to a given model timestep | number or character string |
| 1992 (etc) | number or character string | number or character string |

### Defining model flows

Model flows are defined using a structure similar to that for stocks (Table A1.3). They are then combined with an R function that defines the relationship between the relevant model stocks. The order in which functions are declared in the key determines the order in which they are executed by the model. The R functions have a special taxonomy that is used by the model object to identify relevant model components as inputs, and to push back relevant outputs to the model (Table A1.4). The model contains two types of flows – ordinary flows (or just “flows”) and family flows. The main difference is that flows take individual stocks as arguments, whilst family flows take lists of stocks as arguments. In the case of a family flow, stocks are grouped by their respective family (s\_family above – Table A1.1).

Table A1.3. Overview of template for defining model flows.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Purpose** | **Required data format** | **Example** |
| f\_type | Defines whether a flow is a single or family flow | One of: “flow” or “family\_flow” | flow or family\_flow |
| f\_name | Defines flow name | A character string beginning with f\_ or ff\_ and ending with any relevant name | ff\_land\_allocation |
| f\_id | Unique flow identifier | f or ff followed by a number | f1 |
| f\_function | Identifies the name of the function (in the R global environment) that runs the flow | Character string | land\_allocation |

Table A1.4. Model taxonomy for function **inputs**; the ff\_ object is only relevant to a family flow and where used must be the first argument passed to the flow function

|  |  |  |
| --- | --- | --- |
| Model object | Function taxonomy | Example |
| Stock | s\_ | s\_cereals |
| Input | i\_ | i\_demand\_cereals |
| parameter | p\_ | p\_land\_transition |
| family of stocks | ff\_ | ff\_ (must be the 1st function argument) |

### 2.3 Defining model inputs and outputs

Similarly to model stocks and flows, inputs and outputs can be defined using template input keys. These contain many of the same features as stocks and flows (Table A1.5 and A1.6). In addition, inputs should have an associated input data file, similar to a stock data file. This should be a two column matrix, the first of which contains the model timestep, the second of which should contain the relevant data for that timestep.

Table A1.5. Overview of template for defining model inputs.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Purpose** | **Required data format** | **Example** |
| i\_type | Defines the class | Currently only “input” | input |
| i\_name | Defines input name | A character string beginning with i\_ and ending with any relevant name | I\_farmer\_age |
| i\_id | Unique input | i followed by an integer | i12 |
| i\_to | Identifies the name of the stocks the input is passed to; only for record – not used in model simulations | Character string | s\_veg |
| i\_dat\_type | Is the data spatial or non-spatial | “spatial” or “non\_spat” | “non\_spat” |

### 2.4 Model parameters

Model parameters are defined as a named list within R.

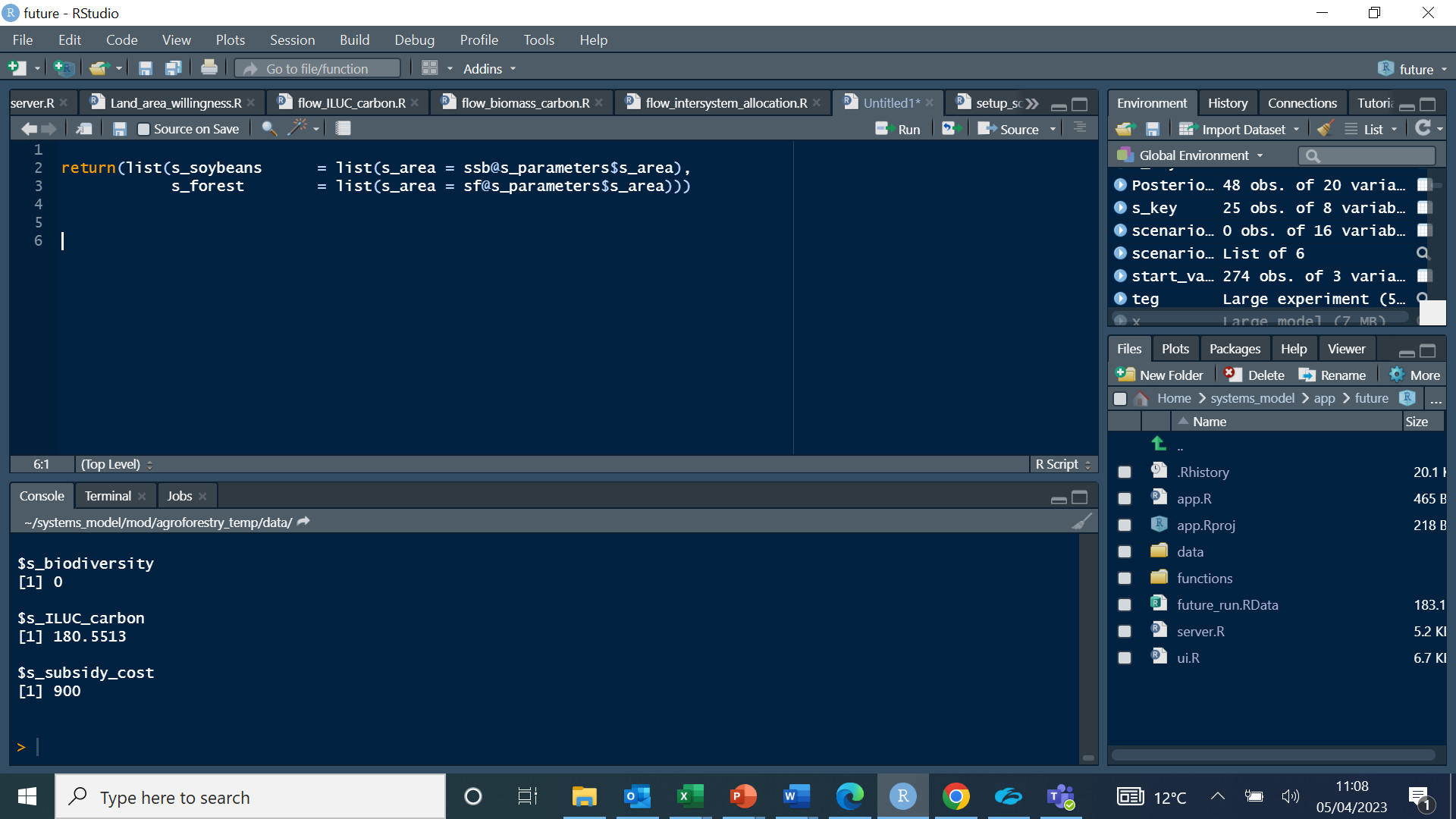
Table A1.6. Overview of template for defining model outputs

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Purpose** | **Required data format** | **Example** |
| o\_type | Defines the class | Currently only “output” | output |
| o\_name | Defines output name | A character string beginning with o\_ and ending with any relevant name | o\_cereal\_area |
| o\_id | Unique output id | o followed by an integer | o12 |
| o\_from\_stocks | Identifies the name of the stocks the output is drawn from; where more than one is used, these are separated by “, “ | Character string, separated by comma space | s\_biomass, s\_lfa\_biomass |
| o\_from\_var | Which stock parameter is the output drawn from? | Character string | s\_area |

## How to run flows: passing flow arguments back to the model

### 3.1 For a regular flow

For a regular (i.e. not family) flow, the return argument should comprise a named list. Each element name should refer to a model stock. Each element of the list should be a named list, where the name corresponds to a stock parameter and the associated value the function output to be passed back.

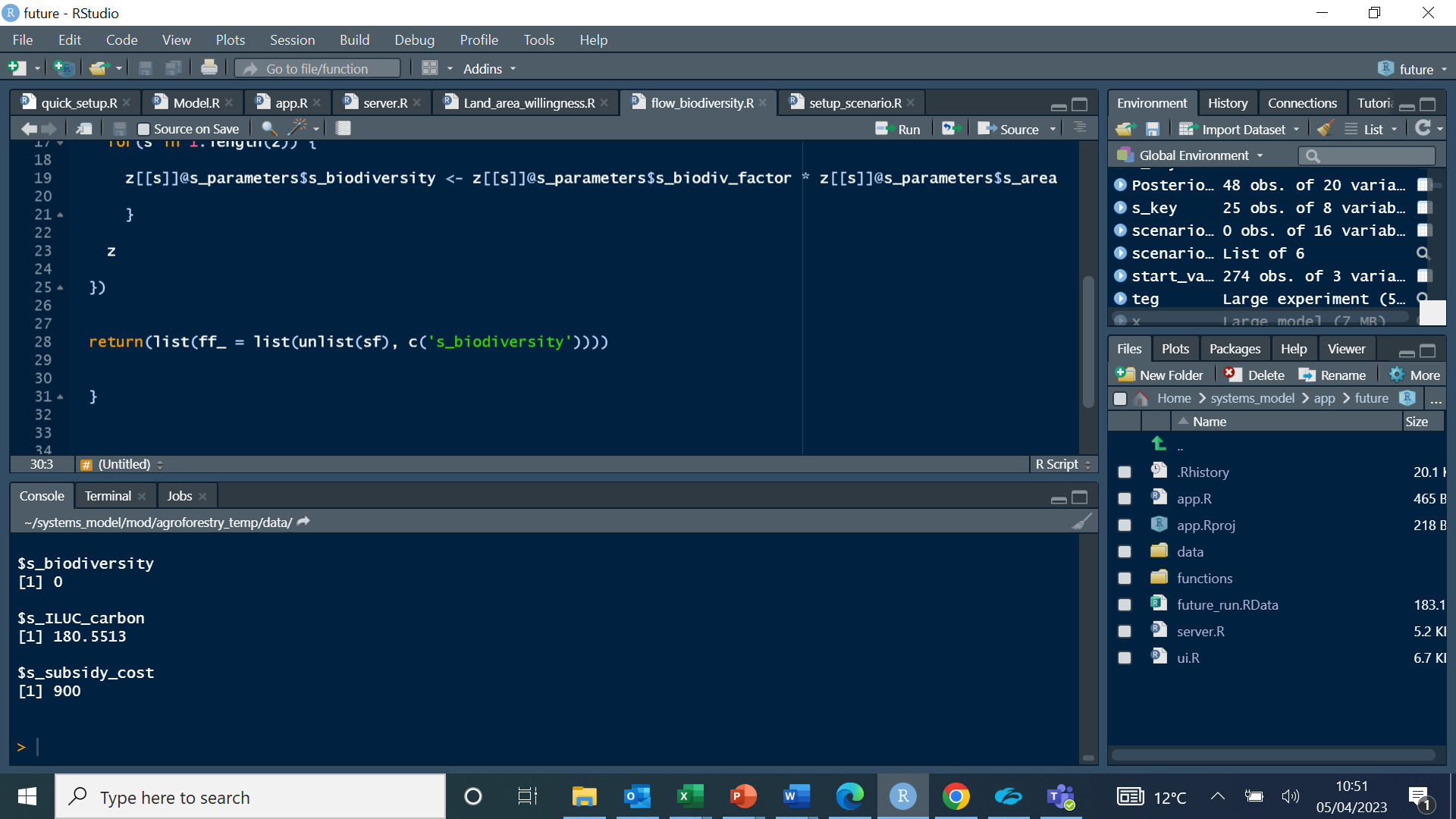


In this example, the flow will push outputs to two stocks (s\_soybeans and s\_forest). In each case, it will update their respective area parameter.

### 3.2 For a family flow

A family flow return argument can adopt the same taxonomy as a regular flow – i.e., it can pass back specific outputs to specific named flows. However, it will frequently be easier and tidier to pass back an output from all the stocks passed to a family flow. In this case, the following taxonomy is used.

ff\_ = list(stock\_list, character\_vector)

where stock\_list is a flat list of stocks (i.e. not containing nested list structures), and character\_vector is a vector of the parameter values that should passed back to the model for each of the stocks in stock list. Each element of the character vector should correspond to a named item of each stock parameter list. E.g.

In this example, sf is a flattened list of stocks 1:n, for which the biodiversity calculation in the flow will be passed back to the model.