Summary

The responses of organisms and populations to environmental stressors are the outcome of complex interactions within organisms, between organisms and with the environment, which can be unravelled and possibly predicted with dynamic models. To this end Dynamic Energy Budget (DEB) theory provides a generic theory to model the life history of organisms and their responses to the environment.

Statement of need

`DEBBase.jl` is a Julia package to simulate individuals and populations exposed to chemcial stressors. It defines a system of ordinary differential equations (ODE) which can be simulated directly using efficient solvers provided by `OrdinaryDiffEq.jl`, or be integrated into an agent-based model (ABM). `DEBBase.jl` provides a common API to simulate the ODE and ABM and data types to organize and configure the large amounts of parameters which can be necessary in an DEB-ABM.

With currently available software, performing extrapolations from the individual level to the population level typically requires to switch between modelling platforms, or to perform individual-level simulations in highly specialized platforms which are not designed for the efficient solving of ODEs.

`DEBBase.jl` aims to fill this gap, by providing a more efficient workflow to simulate the same DEB model in different contexts.

In addition, DEB models often require modifications, for example to account for the idiosyncracies in the physiology of particular organisms. This can often lead to scattered codebases, where it can take substantial effort to identify which parts of a base model have been changed and how. In the design of `DEBBase.jl`, we aimed to allow for model extensions so that the source code only includes newly defined functions, and simultaneously exposes all components of the model at a glance.

Extensions can either be defined in new modules or *ad hoc* within an interactive session or script.
The latter is useful to document the behaviour of alternative model versions during implementation, if the model structure is justified based on basic aspects of model behvaiour rathern than a data-driven model selection.

Methods

Model description

A detailed description of the model is given in `SI_modeldescription`.
In short `DEBBase.jl` provides a base model which is based on the DEBkiss model (**citation needed**). The base model includes a toxicokinetic-toxicodynamic (TKTD) component which allows to simulate an arbitrary number of chemical stressors in mixture, assuming combined effects to occur via independent action (IA), i.e. multiplication of the relative responses.
The model can be simulated as system of differential equations or agent-based model.

The agent-based model makes some simplistic assumptions about aging, reproduction and starvation mortality (see model description and source code for deatils). These assumptions are sufficient to simulate plausible population dynamics, but need to be revisited for specific use cases.

Parameter structures and defaults

The parameter sets for DEB-TKTD models can grow large, especially when dealing with mixtures and AMBs.

Parameters are therefore organized in a designated datatype (`AbstractParamCollection`). The default parameter collection `Params` contains the entries `glb`, `spc` and `agn`, all of which are `AbstractParams`.
`glb` stands for *global* and contains the global parameters such as simulated timespan, resource input rate and temperature.

`spc` stands for *species* and contains species-specific parameters. These are essentially the DEB and TKTD parameters.

Upon initialization of a parameter structure, `glb` and `spc` contain default values, unless modified values were given as keyword arguments.

`agn` stands for *agent* and contains agent-specific parameters. These are used to induce individual variability, either in repeated simulations of the ODE or during simulation of population dynamics with the ABM.
Upon initialization of a parameter structure, `agn` has the value `nothing`. Users typically do not need to interact with `agn` directly, as this field is set internally upon initializaton of an agent.

Running the base model

To run the base model, one can start by initializing the default parameters.

```
using DEBBase.DEBODE
p = Params()
sim = DEBODE.simulator(p)
```

This returns a data frame with all state variables over time.
To simulate different environmental scenarios, the `p.glb` is modified. The code snippet below simulates five nutrient input rates (`Xdot_in`) and stores all results in a single data frame.

```
using DEBBase.DEBODE, DataFrames
p = Params() #
sim = DataFrame()
let Tvec = [15., 20., 25.] # simulate three ambient temperatures
    for T in Tvec
        p.glb.T = T # succesively lower the food input rate
        sim_i = DEBODE.simulator(p) # generate the prededction
        append!(sim, sim_i) #
    end
end
```

To take this to the population-level, we only need to change the simulator. Instead of `DEBODE.simulator`, we call `AgentBased.simulator`.
It is also a good idea to include individual variability, which is done through the zoom factor `Z` (see model

description for details). Since the ABM is always stochastic, it is also a good idea to run repeated simulations. This can be done by manually writing a for loop, or with the `@replicates` macro.

```
using DEBBase.AgentBased, DataFrames, Distributions

p = Params() # initialize the default parameters
p.glb.t_max = 56. # extend the simulated timespan
p.spc.Z = Truncated(Normal(1, 0.1), 0, Inf) # induce individual variability
sim = DataFrame()
let Tvec = [15., 20., 25.] # simulate three ambient temperatures
    for T in Tvec
        p.glb.T = T # succesively lower the food input rate
        sim_i = @replicates AgentBased.simulator(p) 10 # generate the predidction
        append!(sim, sim_i) #
    end
end
```

Experimental futures and outlook