Marine Biology, Oceanography, Computer Science and Computational Science working together.

Claudio Iturra, 2024.

T1:

Here we simulate phytoplankton cells as individuals in a well mixed reactor, like a lab experiment.

1. Import packages

```julia
begin
    using PlanktonIndividuals, Plots, JLD2
    using Plots.PlotMeasures
    using PlutoUI
end
```

2. Grid Setup

First we generate grid information (one grid box, 256m thick, and 128x128 in width) and the computational architecture (CPU).

```
arch =  CPU()
```

```julia
arch=CPU()
```

```
grid =
RegularRectilinearGrid{Float32, PlanktonIndividuals.Grids.Periodic, PlanktonIndividuals.(
domain: x ∈ [0.0, 128.0], y ∈ [0.0, 128.0], z ∈ [0.0, -128.0]
topology (Tx, Ty, Tz):     (Periodic, Periodic, Bounded)
resolution (Nx, Ny, Nz):   (1, 1, 1)
halo size (Hx, Hy, Hz):    (2, 2, 2)
grid spacing (Δx, Δy, Δz): 128.0, 128.0, [min=128.0, max=128.0])
```

```julia
grid = RectilinearGrid(size=(1,1,1), x = (0, 128meters), y = (0,128meters), z =
(0,-128meters))
```

3. Model Setup

Next we setup the individual-based model by specifying the computational architecture, grid, and plankton community.

```
model =
PlanktonModel:
├── floating point data type: Float32
├── grid: RegularRectilinearGrid{Float32, PlanktonIndividuals.Grids.Periodic, PlanktonInd
├── PlanktonIndividuals.QuotaMode() is selected for phytoplankton physiology
├── individuals: 1 phytoplankton species with 1024 individuals for each species
└── maximum number of individuals: 16384 per species
```

```julia
model = PlanktonModel(arch, grid; N_species = 1,
                                  N_individual = [2^10],
                                  max_individuals = 2^10*16)
```

And we setup diagnostics.

```
diags = PlanktonDiagnostics:
        ├── diagnostics of tracers: (:PAR, :NH4, :NO3, :DOC, :T)
        ├── diagnostics of individuals: (:num, :graz, :mort, :dvid, :PS, :BS, :Chl)
        └── save averaged diagnostics every 1 timesteps
```

```julia
diags = PlanktonDiagnostics(model; tracer=(:PAR, :NH4, :NO3, :DOC),
                                   plankton = (:num, :graz, :mort, :dvid, :PS, :BS, :Chl),
                                   iteration_interval = 1)
```

Then we setup the duration of the model simulation, a run directory location, and the kind of output we want.

```
sim = PlanktonSimulation:
      ├── ΔT: 300.0s
      ├── model time: 0.0s
      ├── simulation stops after: 288 iterations
      ├── no velocity provided
      ├── diagnostics:
      │   ├── diagnostics of tracers: (:PAR, :NH4, :NO3, :DOC, :T)
      │   ├── diagnostics of individuals: (:num, :graz, :mort, :dvid, :PS, :BS, :Chl)
      │   └── save averaged diagnostics every 1 timesteps
      └── output writer: nothing
```

```julia
sim = PlanktonSimulation(model, ΔT = 300.0seconds,
                                iterations = 288,
                                diags = diags)
```

Finally we setup the output writer.

```
PlanktonOutputWriter:
├── files are saved at ./results
├── diagnostics are saved as ./results/diags.jld2
├── individuals are not saved
├── write log: false
└── Maximum file size: Inf YiB
```

```julia
sim.output_writer = PlanktonOutputWriter(save_diags = true)
```

4. Model Run

```
update!(sim)
```

5. Access Results

Results have been stored in a `jld2` file. Let's open the file, look inside, and retrieve results.

```
file = JLDFile /home/cl/Downloads/results/diags.jld2 (read-only)
        └─📂 timeseries
           ├─📂 t
           │  ├─🔢 1
           │  ├─🔢 2
           │  ├─🔢 3
           │  ├─🔢 4
           │  ├─🔢 5
           │  ├─🔢 6
           │  └─ ⋯ (282 more entries)
           └─ ⋯ (6 more entries)
```

```
file = jldopen(sim.output_writer.diags_file, "r")
```

```
["t", "PAR", "NH4", "NO3", "DOC", "T", "sp1"]
```

```
keys(file["timeseries"])
```

Extract a vector of iterations

```
iterations =
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,   more ,279, 280, 281
```

```
iterations = parse.(Int, keys(file["timeseries/t"]))
```

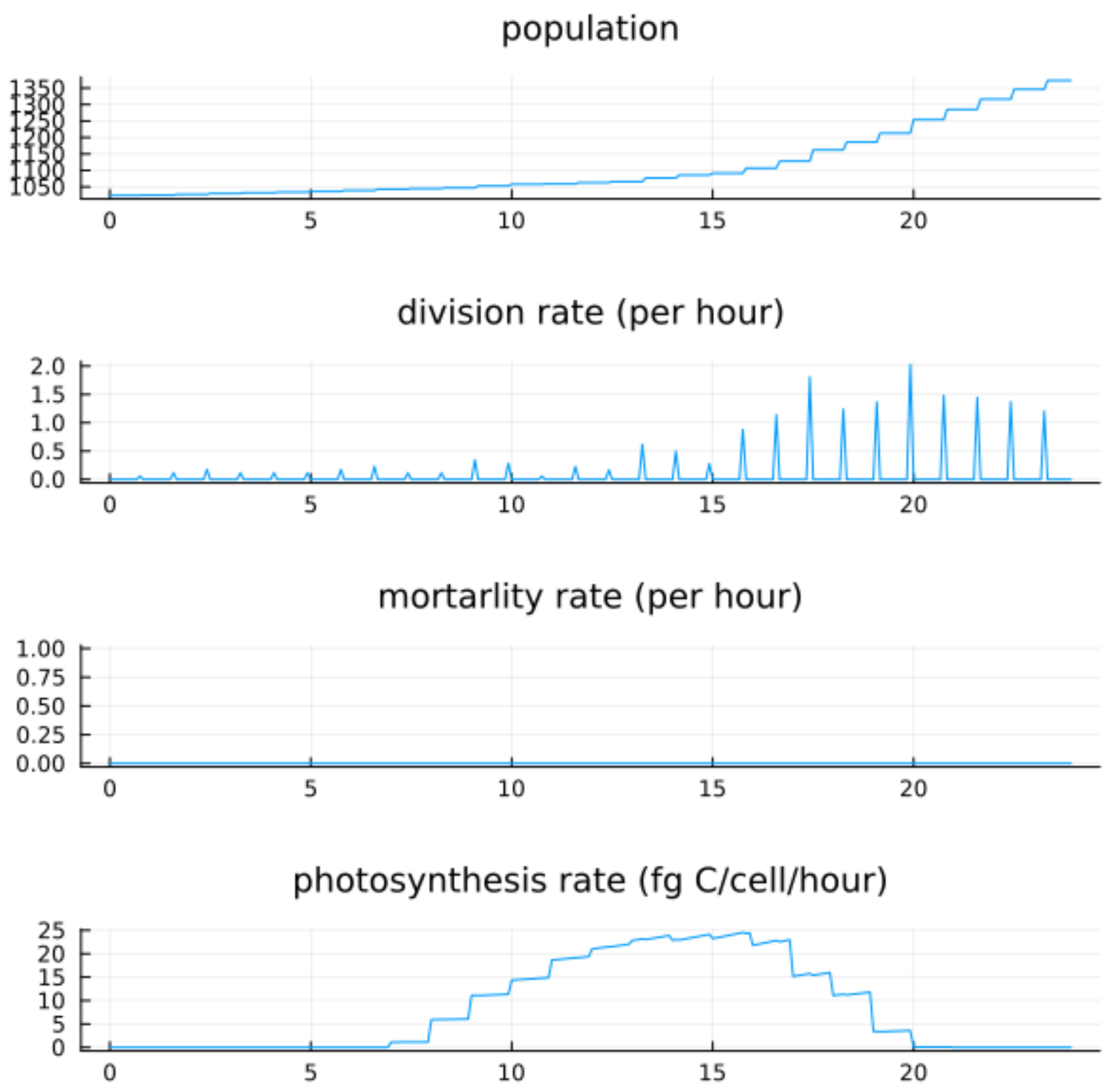
Read results into arrays and close file.

```
begin
    (PAR, NH4, NO3, DOC) = (zeros(288),zeros(288),zeros(288),zeros(288))
    fil2 = sim.output_writer.diags_file
    get_time_series!(fil2,"PAR",PAR)
    get_time_series!(fil2,"NH4",NH4)
    get_time_series!(fil2,"NO3",NO3)
    get_time_series!(fil2,"DOC",DOC)
end
```

```
begin
    (num,dvid,mort,PS,Chl) =
    (zeros(288),zeros(288),zeros(288),zeros(288),zeros(288))
    fil = sim.output_writer.diags_file
    get_time_series!(fil,"sp1/num",num)
    get_time_series!(fil,"sp1/dvid",dvid)
    get_time_series!(fil,"sp1/mort",mort)
    get_time_series!(fil,"sp1/PS",PS)
    get_time_series!(fil,"sp1/Chl",Chl)
end
```

6. Vizualize Results

Now we plot the plankton population as function of time.



And then the environmental variables.

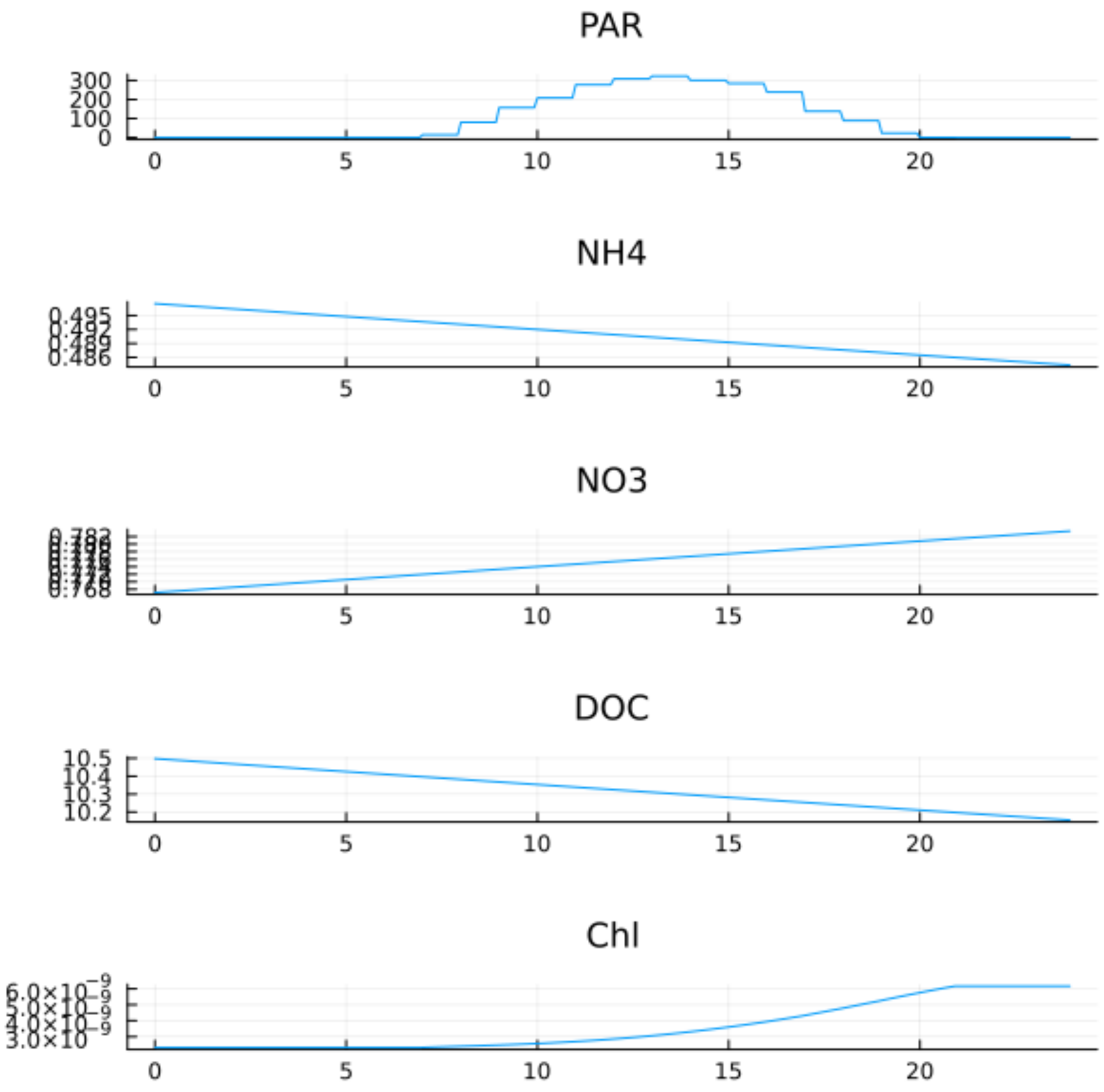


Table of Contents

Marine Biology, Oceanography, Computer Science and Computational Science working together.

Appendix: Helper Functions

```
get_time_series! (generic function with 1 method)
```