EpiMod: Lotka-Volterra

Beccuti Marco, Castagno Paolo, Pernice Simone

Contents

The Lotka–Volterra model

The Lotka–Volterra equations, also known as the predator–prey equations, are a pair of first-order nonlinear differential equations, frequently used to describe the dynamics of biological systems in which two species interact, one as a predator and the other as prey. The populations change through time according to the pair of equations:

$$\begin{aligned} \frac{dx}{dt} &= \alpha x - \beta xy, \\ \frac{dy}{dt} &= \delta xy - \gamma y, \end{aligned} \tag{1}$$

where:

1. x is the number of prey (for example, rabbits);
2. y is the number of some predator (for example, foxes);
3. $\frac{dy}{dt}$ and $\frac{dx}{dt}$ represent the instantaneous growth rates of the two populations;
4. t represents time;
5. $\alpha,\ \beta,\ \gamma,\ \theta$ are positive real parameters describing the interaction of the two species.

Starting

```
library(epimod)
```

Download all the docker images used by *epimod*:

```
downloadContainers()
```

Model generation

```
model_generation(net_fname = "./Net/Lotka-Volterra.PNPRO")
```

Sensitivity analysis

The R function *sensitivity_analysis()* implements the sensitivity analysis taking as input

1. the *.solver* file generated by the *model_generation* function, the total number of samplings to be performed,
2. the final solution time,
3. the time step defining the frequency at which explicit estimates for the system values are desired,
4. parameters_fname (*Functions_list.csv*): a textual file in which the parameters to be studied are listed associated with their range of variability. significato delle colonne, e che serve il ; come sep, spazi non nei path An example is given by the following file, where

```
#>   V1    V2              V3                        V4                      V5
#> 1  i  init  init_generation  min_init = c(0.9 , 0.8)  max_init = c(1.8 , 2)
```

5. functions_fname (*Functions.R*): ...

```
init_generation<-function(min_init , max_init)
{
    # min/max are vectors = first position interval values for the first place
    # and second position for the second place.

   i_1=runif(n=1,min=min_init[1],max=max_init[1])
   i_2=runif(n=1,min=min_init[2],max=max_init[2])

   return( c(i_1,i_2) )
}
```

6. target_value_fname (*Target.R*): ... .

Assuming the following fixed rates: $\beta = 4/3,\ \theta = 1$. Than we change the prey and predator initial conditions from 200 to 1800, and the prey and predator birth rates $\alpha,\ \gamma$ from 0.1 to 2, knowing that we are interested to the trajectory generated with the following values: $x(0) = 1000,\ y(0) = 500,\ \alpha = 2/3,\ \gamma = 1$.

PRCC and ranking This step is necessary if we are interested to calculate the PRCC to identify among the input parameters which are the sensitive ones (i.e., those that have a great effect on the model behaviour). This may simplify the calibration step reducing (1) the number of variables to be estimated and (2) the search space associated with each estimated parameter.

In order to run the simulations, the user must provide

1. the reference dataset;

2. the definition of a function to compute the distance (or error) between the models' output and the reference dataset itself.

In this case the reference dataset is the trajectory obtained from the starting point 1 predator and 1 prey. Similarly to the previouos step, we generate different initial conditions for the system but in this case.

The function defining the distance takes in inputs only the reference data and the simulation's output (i.e. a trajectory); an example it could be the following:

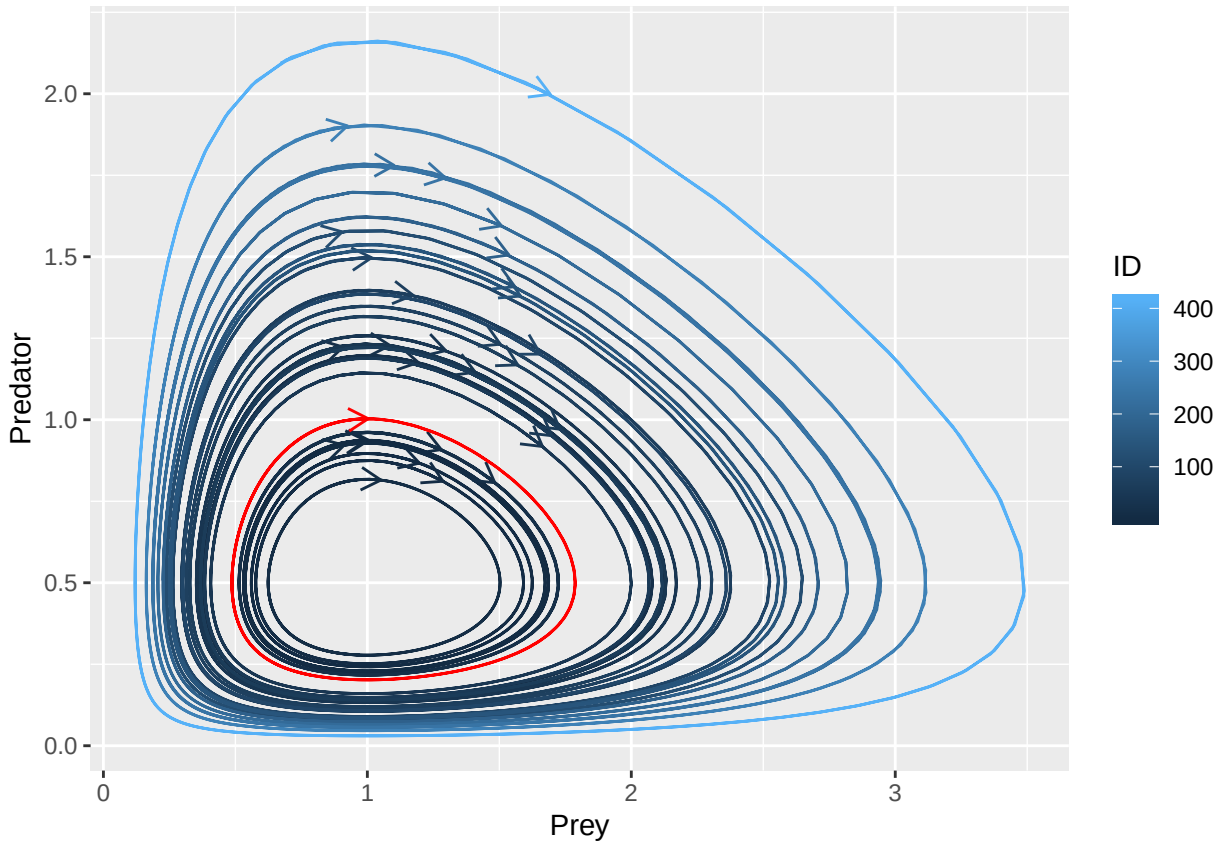
```
msqd<-function(reference, output)
{
    Predator <- output[,"Predator"]
    Prey <- output[,"Prey"]

    diff.Predator <- sum(( Predator - reference[,2] )^2 )
    diff.Prey <- sum(( Prey - reference[,1] )^2 )

    return(diff.Predator+diff.Prey)
}
```

where the squared error between the Predator and Prey trajectories obtaained from the simulation and the corresponding reference trajectories are calculated, respectively named *diff.Predator* and *diff.Prey*. Then, the sum of these errors is returned. Our purpose will be to minimize (see next section) this function in order to identify the trajectory most similar to the reference data, therefore

```
Target<-function(output)
{
    ret <- output[,"Predator"]
    return(as.data.frame(ret))
}
```

Let us note that the name of the distance and target function must have the same name of the corresponding R file.

```
sensitivity<-sensitivity_analysis(n_config = 30,
                                  parameters_fname = "Input/Functions_list.csv",
                                  functions_fname = "Rfunction/Functions.R",
                                  solver_fname = "Net/Lotka-Volterra.solver",
                                  reference_data = "Input/reference_data.csv",
                                  distance_measure_fname = "Rfunction/msqd.R" ,
                                  target_value_fname = "Rfunction/Target.R" ,
                                  parallel_processors = 2,
                                  f_time = 20,
                                  s_time = .1)
```


Predator
2.0
1.5
1.0
0.5
0.0
0
1
2
3
Prey
ID
400
300
200
100

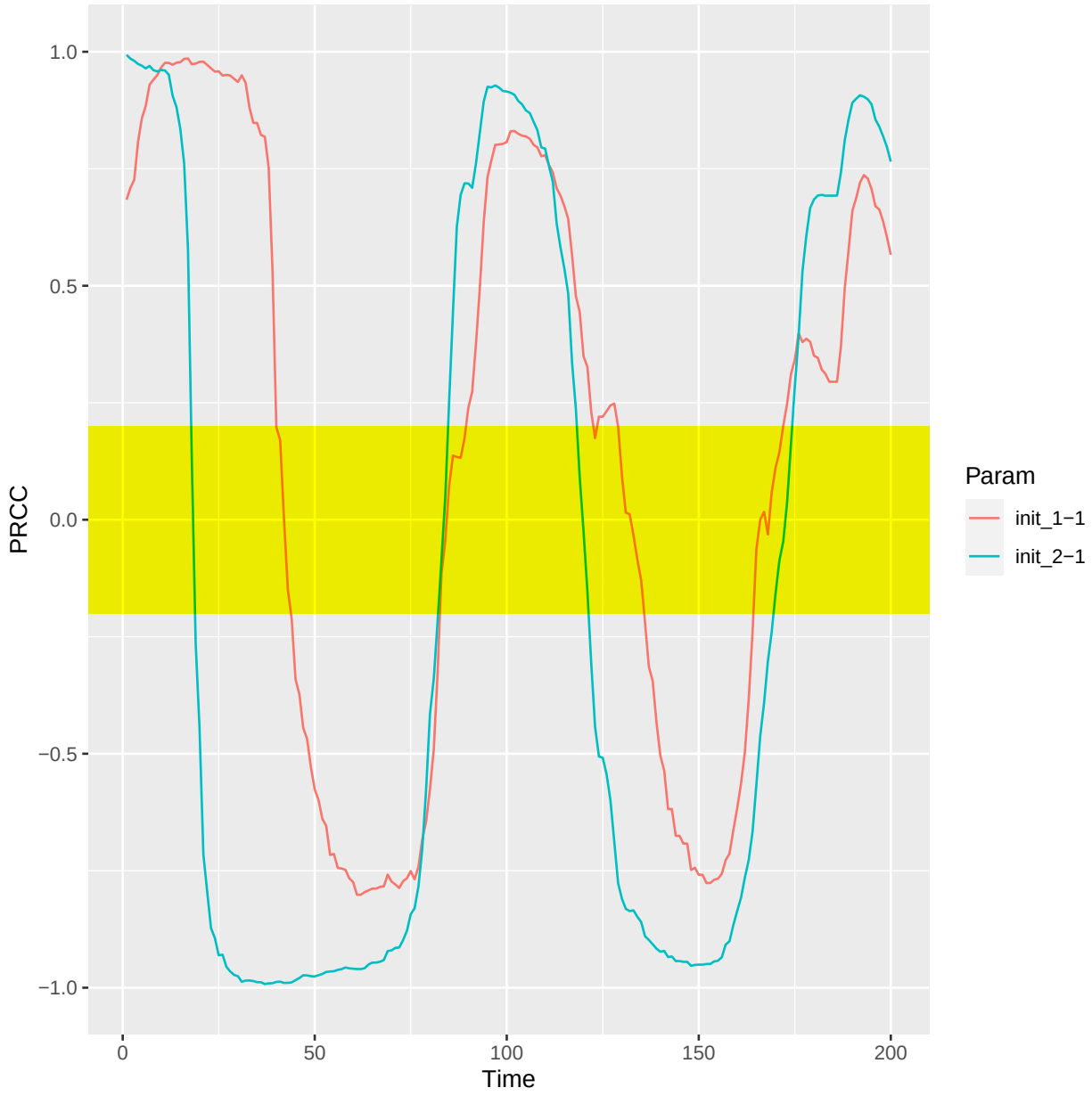


Figure 1: PRCC for the **Predator** place over time.

Running the sensisitivity analysis, we can replicate the results reported on Wikipedia, *https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations*. Indeed, textual file for each generated parameter combination (i.e. 20) containing the system solution is returned. Each file is characterized by as many rows as the solution time points (obtained dividing the final time point by the time step) and as many columns as the system components. We can now generate the phase-space plot, where it is possible to see that the predators thrive when there are plentiful prey but, ultimately, outstrip their food supply and decline. As the predator population is low, the prey population will increase again. These dynamics continue in a cycle of growth and decline.

Let us note that it is possible to run the sensitivity analysys without PRCC or ranking, in the case that we are intersted only on to have a general idea of the simulation' results.

Calibration analysis

```
model_calibration(solver_fname = "Net/Lotka-Volterra.solver",
                  reference_data = "Input/reference_data.csv",
                  distance_measure_fname = "Rfunction/msqd.R" ,
                  f_time = 20,
                  s_time = .1,
                  # Vectors to control the optimization
                  ini_v = c(5,5),
                  ub_v = c(10, 10),
                  lb_v = c(0, 0),
                  optim_vector_mod = TRUE,
                  max.time = 60 # seconds
)
```

Whatif Analysis