Phase plane analysis and parameter estimation in R

Rob J de Boer, Theoretical Biology, Utrecht University

Basics of phase plane analysis with Grind

This tutorial describes an R-script, `grind.R`, that allows students and investigators whom are not very familiar with the R language to perform phase plane analysis and parameter fitting. Phase plane analysis is powerful graphical method to analyze the steady states of low-dimensional mathematical models formulated as ordinary differential eqautions (ODEs). Fitting ODE models to data has become a very common practice in biology.

Grind is a "wrapper" around the commonly-used R-packages `deSolve`, `FME` and `rootSolve` developed by Karline Soetaert and colleagues [Soetaert *et al*, 2009a, 2009b, 2010a, 2010b]. These packages have to be installed beforehand (see the section on **Installation** below). Our aim with developing `grind.R` was to define five easy-to-use functions:

- `run()` integrates a model numerically and provides a time plot or a trajectory in the phase plane,
- `plane()` draws nullclines and can provide a vector field or a phase portrait,
- `newton()` finds steady states and can provide the Jacobian with its eigenvalues and eigenvectors.
- `continue()` performs parameter continuation of a steady state, providing a bifurcation diagram,
- `fit()` fits a model to data by estimating its free parameters, and depicts the result in a timeplot.

Here is a link to Grind's homepage.

The Lotka Volterra model in Grind

Consider the Lotka Volterra model $R' = rR(1 - R/K) - aRN$ and $N' = caRN - dN$, with parameter values, $r = K = a = c = 1$, and $d = 0.5$, and the initial condition $R = 1$ and $N = 0.01$. Using Grind's default names for the model, parameters, and state, i.e., `model()`, `p`, and `s`, this should be written as

```
model <- function(t, state, parms) {
  with(as.list(c(state,parms)), {
    dR <- r*R*(1 - R/K) - a*R*N
    dN <- c*a*R*N - d*N
    return(list(c(dR, dN)))
  })
}
p <- c(r=1,K=1,a=1,c=1,d=0.5)
s <- c(R=1,N=0.01)
```

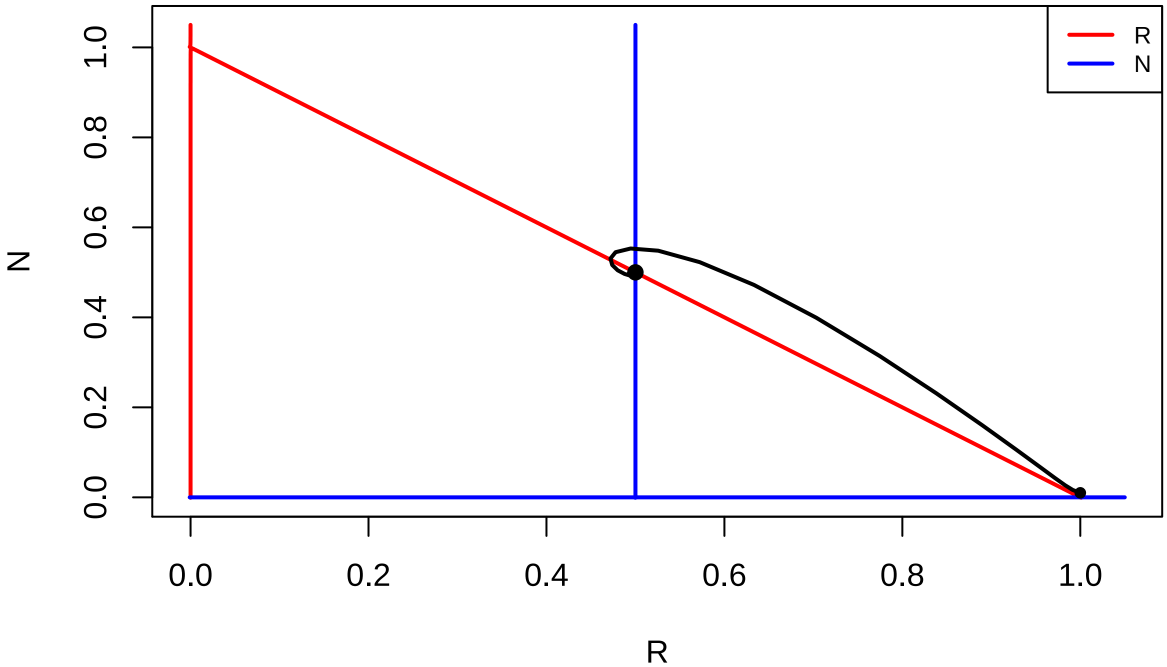
where the arguments `(t, state, parms)` provide the time, the state (R, N), and the parameters (as defined by the vector `p`) to the equations. The `with(as.list(c(state,parms)),..)` function allows one to identify parameters and variables by their name (instead of having to write `parms["K"]` and `state["R"]`, and so on; see the webpage on with if you want to know more about `with()`). Note that the derivatives are returned as a list, and that **the order of the variables in this list has to be the same as their order in the state**!

Having defined a model, its initial state, and its parameters one can just call `run()` to start a numerical integration, `plane()` to plot a phase plane, or `newton()` to find a steady state close to the state `s`. Here is an example (starting with sourcing the `grind.R` script):

```
source("grind.R")
```

```
## grind.R (03-05-2024) was sourced
```

```
plane()
s <- run(traject=TRUE)
s <- newton(plot=TRUE)
```




```
##   R   N
## 0.5 0.5
## Stable spiral point, eigenvalues:
## [1] -0.25+0.4330127i -0.25-0.4330127i
```

where the option `traject=TRUE` tells `run()` to sketch a trajectory (instead of making a timeplot), and the option `plot=TRUE` adds a bullet to the phase where the steady state was found. The call to `newton()` prints the eigenvalues to the screen (and returns the state that was found). If this fails, please check that the R-packages were properly installed.

Examples

The best way to learn Grind is to try a few of the following examples:

- phaseplane Illustrates the usage of `run()`, `plane()`, `newton()`, and `continue()`. The latter allows for **simple bifurcation analysis** by parameter continuation.
- fitting Illustrates the usage of `fit()` to **estimate parameters** by fitting a model to one or more data sets.
- running Illustrates how one can add **noise and events** while running a model, add columns to the output of `run()`, explains how to solve **delay differential equations** (DDEs), and illustrates the definition of **maps** by analyzing the logistic map.
- saddle Digs a little deeper into phase plane analysis by depicting **saddle-node bifurcations**, **eigenvectors** and plotting a **separatrix**.

- hysteresis Depicts **saddle-node bifurcations** in a model for a vegetation in arid areas, and shows how increasing the grazing rate can lead to a catastrophic collaps that is difficult to repair due to **hysteresis**.
- chaos3d Illustrates the usage of the `cube()` extension of Grind by plotting **3-dimensional nullclines**, and plotting a 3-D trajectory approaching a chaotic attractor.
- shiny Illustrates how one can make a **Shiny app** with sliders to set the parameters of a model.

Finally, the manual explains **all options** of Grind's functions in alphabetical order.

Installation

Since Grind is a wrapper around the `deSolve`, `FME` and `rootSolve` R-packages, and because `FME` requires the `coda` package, these four packages need to be installed the first time you use Grind. This can be done by **Install Packages** in the **Tools** menu of RStudio.

Download the the R-codes grind.R and lotka.R, save them in a local directory, and open them in RStudio. It is probably convenient to set the working directory to the folder the files were stored (`Set working directory` in the `Session` menu of RStudio). To get started type `source("grind.R")`, or click the sourcebutton in RStudio, and begin executing the `lotka.R` script.

References

- Soetaert, K., 2009a. rootSolve: Nonlinear root finding, equilibrium and steady-state analysis of ordinary differential equations. R package 1.6.
- Soetaert, K. and Herman, P. M., 2009b. A Practical Guide to Ecological Modelling. Using R as a Simulation Platform. Springer. ISBN 978-1-4020-8623-6.
- Soetaert, K. and Petzoldt, T., 2010a. Inverse modelling, sensitivity and Monte Carlo analysis in R using package FME. Journal of Statistical Software 33:1–28.
- Soetaert, K., Petzoldt, T., and Setzer, R. W., 2010b. Solving differential equations in R: Package deSolve. Journal of Statistical Software 33:1–25.

Websites

Check Grind's homepage for links and more info. All files can also be downloaded from the Grind's directory.