A Grind tutorial on estimating parameters from experimental data.

Today you will work with an R-script called `grind.R` that is a wrapper around the R-packages `deSolve`, `FME` and `rootSolve` developed by Karline Soetaert and colleagues [2–5]. These packages allow one to solve differential equations, find their steady state, and perform nonlinear parameter estimation. Today you only need three of Grind's easy-to-use functions:

- `run()` integrates a model numerically and provides a time plot or a trajectory in the phase plane,
- `fit()` fits a model to data by estimating its parameters, and depicts the result in a timeplot.
- `timePlot()` plots a data frame.

The R-script and the manual of Grind is available on the website `http://tbb.bio.uu.nl/rdb/grindR/`.

We will work in the RStudio environment. To get started perform the following:

- **If you work on your own laptop** you will need to install RStudio, R, and the Soetaert libraries. Check this tutorial on how to install RStudio and R:
  `http://tbb.bio.uu.nl/rdb/bm/Introduction_to_R.pdf`
  After that, install the three Soetaert libraries by opening RStudio, going to the Tools menu, choosing Install Packages, and then typing `deSolve`, `FME` and `rootSolve`. (Experts may prefer to type `install.packages(c("deSolve", "FME", "rootSolve")` in the R-console).
- **Otherwise**: skip the previous step and go to the University's MyWorkPlace environment.
- Make a local folder (directory) on your system where you will save the following 5 files: `grind.R`, `bact.R`, `crispr.R`, `Levin_pg13_fig2_B0.csv` and `Levin_pg13_fig2.csv`. All files can be obtained by downloading the `crispr.zip` file from Blackboard or directly from the `tbb.bio.uu.nl/rdb/practicals/crispr/` website. (In Windows use the right mouse button and use "Save target as", to store a file in a folder).
- Set the working directory of RStudio to the folder where your R-codes and data are stored (`Set working directory` in the `Session` menu of RStudio). Files will then be opened and saved in that directory.
- Use "Open file" to open `grind.R` and **source** Grind to define the functions (button on the right).
- Use "Open file" to open `bact.R` and **run** that script line by line.

Thus, after downloading the files, you first "Source" the `grind.R` file (button in right hand top corner), and then "Run" the model with its parameter and state definitions line by line ("Run" or "Control Enter"). Make sure you understand what you do and what you get.

Bacterial growth.
The `bact.R` script defines a model of two ODEs,

$$\frac{\mathrm{d}R}{\mathrm{d}t} = -e\,\frac{vR}{k+R}\,B \quad \text{and} \quad \frac{\mathrm{d}B}{\mathrm{d}t} = \frac{vR}{k+R}\,B\ .$$

The script next defines some default parameter values in the vector `p`, and defines an initial state in the vector `s` (all taken from Levin *et al.* [1]). Then it runs the model for 7 hours using the Grind function `run()`. The R-function `read.csv()` is called to read a data file, and this data is plotted. Finally, the fitting starts: first the parameter v is selected as the only "free" variable. The Grind function `fit()` is called, telling the system which data to fit, how to plot it, and use a logarithmic transformation of the data (where log1p(x) computes log(1+x)). The function reports back the summed squared residuals (SSR) and the best estimate for v. The R-function `summary()` provides additional information of the result.

Questions: Why do we transform the data? Next, try to also estimate the parameter k from the data. Why is k not identifiable from this data?

Bacterial growth with phages
The `crispr.R` script defines the full Levin et al. model [1] with phages, P, infected bacteria, M,

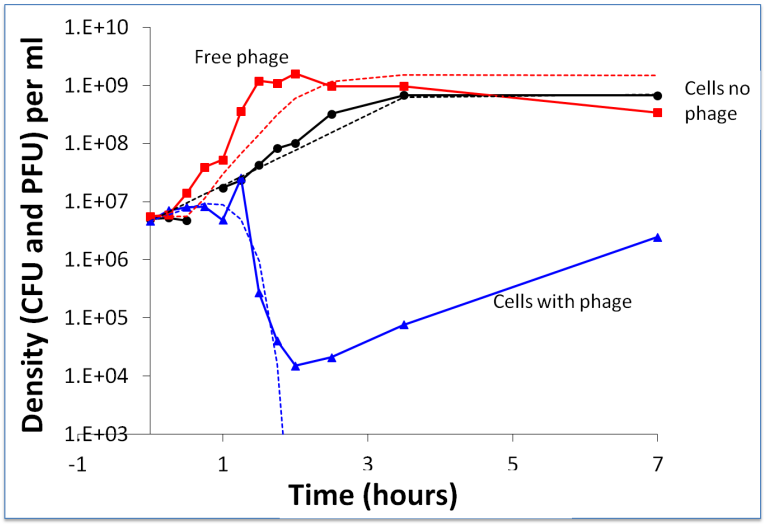

Figure 1: Figure 2 from Levin *et al.* [1]: Short-term bacterial and phage growth dynamics in the absence of immunity: change in the density of WT bacteria and WT phage. The broken lines are the densities predicted by the above model with the following parameters: $v = 1.4$ per hour, $e = 5 \times 10^{-7} \mu$g per cell, R (initial resource concentration)=350 μg/ml, $\delta = 8 \times 10^{-8}$ ml per cell per phage per hour, $b = 80$ phage particles per infected cell and, $\lambda = 0.4$ hours. At 24 hours, the estimated densities of bacteria and phage in these cultures were, respectively, 5×10^8 cells per ml for the control and 3.2×10^5 CFU and 6.2×10^7 PFU for the bacteria and phage in the mixed culture. doi:10.1371/journal.pgen.1003312.g002

resistant bacteria, B_1, and sensitive bacteria, B_0:

$$\frac{\mathrm{d}R}{\mathrm{d}t} = -e\,\frac{vR}{k+R}\left(B_0 + (1-s)B_1\right)\,, \quad \frac{\mathrm{d}B_0}{\mathrm{d}t} = \frac{vR}{k+R}\,B_0 - \delta B_0 P\,, \quad \frac{\mathrm{d}B_1}{\mathrm{d}t} = \frac{vR}{k+R}\,(1-s)B_1$$

$$\frac{\mathrm{d}M}{\mathrm{d}t} = \delta B_0 P - M/\lambda\,, \quad \text{and} \quad \frac{\mathrm{d}P}{\mathrm{d}t} = bM - \delta(B_0+B_1)P\,,$$

where δ is an infection rate, s a selection coefficient, λ the eclipse time, and b the burst size. After defining the default parameters and the initial state, the option `tweak` is defined to add a column to the data generated by the model. Here the column contains the total number of bacteria, $B = B_0 + B_1 + M$. This is required because the data only contains the number of phages, P, and the total number of bacteria, B.

Questions: Test for yourself if you can estimate the selection coefficient, s, and the initial density, B_1, of the resistant bacteria. What is your opinion on this model: does it describe the data well? Are s and B_1 both identifiable?

November 4, 2019, Rob J. de Boer, Utrecht University

References

[1] **Levin, B. R., Moineau, S., Bushman, M., and Barrangou, R.**, 2013. The population and evolutionary dynamics of phage and bacteria with CRISPR-mediated immunity. PLoS. Genet. **9**:e1003312.

[2] **Soetaert, K.**, 2009. rootSolve: Nonlinear root finding, equilibrium and steady-state analysis of ordinary differential equations. R package 1.6.

[3] **Soetaert, K. and Herman, P. M.**, 2009. A Practical Guide to Ecological Modelling. Using R as a Simulation Platform. Springer. ISBN 978-1-4020-8623-6.

[4] **Soetaert, K. and Petzoldt, T.**, 2010. Inverse modelling, sensitivity and Monte Carlo analysis in R using package FME. Journal of Statistical Software **33**:1–28.

[5] **Soetaert, K., Petzoldt, T., and Setzer, R. W.**, 2010. Solving differential equations in R: Package deSolve. Journal of Statistical Software **33**:1–25.

bact.R

```
model <- function(t, state, parms) {
  state <- ifelse(state<0,0,state)
  with(as.list(c(state,parms)), {
    f <- v*R/(k+R)
    dtR <- - f*e*B
    dtB <- f*B
    return(list(c(dtR, dtB)))
  })
}

p <- c(e=5e-7,v=1.4,k=1)
s <- c(R=350,B=5189702)

run(7,0.1,ymin=1,ymax=1e10,log="y")

data <- read.csv("Levin_pg13_fig2_B0.csv",header=TRUE)
timePlot(data,ymin=3,ymax=1e10,log="y",draw=points)

free <- c("v")
f <- fit(data,tstep=0.1,ymin=1e3,ymax=1e10,log="y",fun=log1p,free=free,legend=F)
summary(f)

free <- c("v","k")
f <- fit(data,tstep=0.1,ymin=1e3,ymax=1e10,log="y",fun=log1p,lower=0,upper=10,free=free
    ,legend=F)
summary(f)
```

crispr.R

```
model <- function(t, state, parms) {
  state <- ifelse(state<0,0,state)
  with(as.list(c(state,parms)), {
    f <- v*R/(k+R)
    dtR <- - f*e*(B0+(1-s)*B1)
    dtB0 <- f*B0 - delta*B0*P
    dtB1 <- (1-s)*f*B1
    dtM <- delta*B0*P - M/lambda
    dtP <- b*M - delta*P*(B0+B1)
    return(list(c(dtR, dtB0, dtB1, dtM, dtP)))
  })
}

p <- c(b=80,delta=5e-8,e=5e-7,lambda=0.4,v=1.5,k=1,s=0)
s <- c(R=350,B0=4382322,B1=100,M=0,P=5.58e+06)

tweak <- "nsol$B<-nsol$B0+nsol$B1+nsol$M"

run(7,0.1,ymin=1e3,ymax=1e10,log="y",tweak=tweak)

data <- read.csv("Levin_pg13_fig2.csv",header=TRUE)
timePlot(data,ymin=1e3,ymax=1e10,log="y",draw=points)

free <- c("B1","b","lambda","s")
f <- fit(data,tstep=0.1,ymin=1e3,ymax=1e10,log="y",fun=log1p,free=free,tweak=tweak)
summary(f)
```