Identification of parameters

Regression and Numerical integration

Julien Arino
University of Manitoba
julien.arino@umanitoba.ca

Fall 2024

The University of Manitoba campuses are located on original lands of Anishinaabeg, Ininew, Anisininew, Dakota and Dene peoples, and on the National Homeland of the Red River Métis.
We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of Reconciliation and collaboration.

Outline

Statement of the problem

Case of the logistic equation

Using nonlinear regression

Using simulations

Statement of the problem

Case of the logistic equation

Using nonlinear regression

Using simulations

Objective

We are given a table with the population census at different time intervals between a date a and a date b, and we have a model to describe the evolution of this population

We want to **find parameters** of the model so that solutions of the model **fit** the data as well as possible

Sources of uncertainty

- Some parameters are known with reasonable accuracy. Others are known within a range of possible values
- Data is obtained through measurement, and this measurement is not necessarily very precise
- Data is usually intrinsically "noisy"
- The model you have is usually wrong (**all** models are wrong, the problem is to find one that is not too wrong, i.e., capable of answering your question)

Be aware of these limitations

The US population from 1790 to 2000 (revised numbers)

Year	Population (millions)	Year	Population (millions)
1790	3.929	1900	76.212
1800	5.308	1910	92.228
1810	7.240	1920	106.021
1820	9.638	1930	123.202
1830	12.866	1940	132.164
1840	17.069	1950	151.325
1850	23.192	1960	179.323
1860	31.443	1970	203.302
1870	38.558	1980	226.542
1880	50.156	1990	248.709
1890	62.948	2000	281.421

The logistic equation

r the intrinsic growth rate of the population, K the carrying capacity,

$$N' = rN\left(1 - \frac{N}{K}\right) \tag{Logistic}$$

Parameter identification

To identify parameters, we can use **nonlinear regression**. With the logistic equation, there are two methods:

1. Since the solution $N(t)$ to (Logistic) is known, we can use nonlinear regression directly on $N(t)$
2. We use nonlinear regression on the constructed (simulated) solution to (Logistic)

Finding the solution of (Logistic) using maple

```
eq := diff(N(t),t) = r*N(t)*(1-N(t)/K)
```

$$\frac{d}{dt}N(t) = rN(t)\left(1 - \frac{N(t)}{K}\right)$$

Solve without specifying an initial condition:

```
dsolve(eq, N(t))
```

$$N(t) = \frac{K}{1 + e^{-rt}\_C1K}$$

Solve with initial condition $N(0) = C$:

```
dsolve({eq, N(t0) = C}, N(t))
```

$$N(t) = \frac{CKe^{-rt_0}}{Ce^{-rt_0} + e^{-rt}K - e^{-rt}C}$$

We use the solution

$$N(t) = \frac{N_0 K e^{-rt_0}}{N_0 e^{-rt_0} + e^{-rt}(K - N_0)}$$

(we have replaced C by N_0)

To reduce the number of parameters to find, we assume that the initial point is $(t_0, N_0) = (1790, 3.929)$, the first data point

Note that we are working in millions (this is important later)

Write the points as (t_k, N_k), $k = 2, \ldots, 22$ –there are 22 data points, but we use the first as (t_0, N_0) or, to make things more convenient to write, (t_1, N_1). We want to minimize

$$S = \sum_{k=2}^{22} \left(N(t_k) - N_k\right)^2 ,$$

where t_k are the known dates, N_k are the known populations, and

$$N(t_k) = \frac{N_0 K e^{-rt_0}}{N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)}$$

Emphasize dependence on r, K:

$$S(r, K) = \sum_{k=2}^{22} \left(\frac{N_0 K e^{-rt_0}}{N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)} - N_k \right)^2$$

This is maximal if (necessary condition) $\partial S / \partial r = \partial S / \partial K = 0$.

We have, for a given $k = 2, \ldots, 22$,

$$\frac{1}{2}\,\frac{\partial}{\partial r}\left(\frac{N_0 K e^{-rt_0}}{N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)} - N_k\right)^2 =$$

$$\frac{K\left(N_k(N_0 - K)e^{-rt_k} + N_0 e^{-rt_0}(K - N_k)\right) N_0 e^{-r(t_0+t_k)}(t_0 - t_k)(N_0 - K)}{\left(N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)\right)^3}$$

and

$$\frac{1}{2}\,\frac{\partial}{\partial K}\left(\frac{N_0 K e^{-rt_0}}{N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)} - N_k\right)^2 =$$

$$\frac{\left(e^{-rt_0} - e^{-rt_k}\right)\left(N_k(N_0 - K)e^{-rt_k} + N_0 e^{-rt_0}(K - N_k)\right) {N_0}^2 e^{-rt_0}}{\left(N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)\right)^3}$$

So $\partial S/\partial r = 0 \Leftrightarrow$

$$\left(N_k(N_0 - K)e^{-rt_k} + N_0 e^{-rt_0}(K - N_k)\right) N_0 e^{-r(t_0+t_k)}(t_0 - t_k)(N_0 - K) = 0$$

(provided $\left(N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)\right)^3 \neq 0$)

That is $\partial S/\partial r = 0 \Leftrightarrow$

$$N_k(N_0 - K)e^{-rt_k} + N_0 e^{-rt_0}(K - N_k) = 0 \qquad (*)$$

or

$$t_0 - t_k = 0 \qquad \text{or} \qquad N_0 - K = 0$$

The case $t_0 = t_k$ cannot happen, since $k = 2, \ldots, 22$ (and we assume we do not have two different measurements for one time value). So we have either $K = N_0$ or (*)

Solving (*) for r, we get

$$r = -\frac{\ln\left(\frac{N_0(K-N_k)}{N_k(K-N_0)}\right)}{t_k - t_0} = \frac{\ln\left(\frac{N_k(K-N_0)}{N_0(K-N_k)}\right)}{t_k - t_0}$$

Also $\partial S/\partial K = 0 \Leftrightarrow$

$$\left(e^{-rt_0} - e^{-rt_k}\right)\left(N_k(N_0 - K)e^{-rt_k} + N_0 e^{-rt_0}(K - N_k)\right) = 0$$

(provided $\left(N_0 e^{-rt_0} + e^{-rt_k}(K - N_0)\right)^3 \neq 0$)

That is, $\partial S/\partial K = 0 \Leftrightarrow$

$$e^{-rt_0} - e^{-rt_k} = 0$$

or

$$N_k(N_0 - K)e^{-rt_k} + N_0 e^{-rt_0}(K - N_k) = 0 \qquad (**)$$

The first condition implies $t_0 = t_k$, which is impossible. So we are left with (**), which is the same equation as (*)

So this is a difficult problem.. (see the theory for nonlinear least squares if you are interested)

So we use plan B: numerics directly..

What we need to do

Let us forget that we know the explicit solution to (Logistic)

- The solution to (Logistic) can be approximated numerically
- We can construct one such numerical solution for given values of r and K
- We then can see "how far off" that solution is from our data points
- We change the parameters r and K a little, find out "how far off" the new solution is from the data points
- And repeat until we have found a solution that is better than others..

Finding the numerical solution to (Logistic)

We can use

- matlab
- octave
- scilab
- maple
- mathematica
- many others..

matlab, octave and scilab are recommended because of the “philosophy”

Using matlab

In matlab (and octave) the philosophy is very close to the "natural" way one proceeds with an ode: given the ODE

$$x' = f(t, x)$$

we must define the right hand side (RHS) function (the vector field) $f(t, x)$, and use it to compute the (numerical) solution

Reminder: Euler's method

The solution to the initial value problem

$$\begin{aligned} x' &= f(t, x) \\ x(t_0) &= x_0 \end{aligned}$$

can be approximated numerically by the following sequence:

$$\begin{aligned} t_{k+1} &= t_k + h \\ x_{k+1} &= x_k + hf(t_k, x_k) \end{aligned}$$

for a time step $h > 0$ and with first term (t_0, x_0)

Back to matlab

The techniques (a.k.a. “numerical solvers”) in matlab are much more advanced, but the idea is the same: approximate the solution to an ODE by using a numerical algorithm the uses information on the “shape” of the vector field

We need two files:

1. a RHS function defining $f(t, x)$
2. a function or command line statement that “calls” the RHS function with a numerical solver

The RHS function

For the logistic equation, we could define the following function

```
function dN=rhs_logistic(t,N,p)
% This function returns the value of dN/dt
% at the point (t,N), using parameters in the
% structure p

dN=p.r*N*(1-N/p.K);
```

which we save in a file called, say, `rhs_logistic.m`

Note that `t` is required in the function arguments even if not used in the RHS function, i.e., even if f is autonomous

Using structures

The variable `p` is defined as a *structure*. This is a very useful construct in many programming languages. Think of it as a *container*:

```
>> p.K=100;
>> p.r=2;
>> p
p =
    K: 100
    r: 2
```

Pros: `p` is passed to the function as one parameter, instead of a list of parameters.
Cons: do not forget `p.` in front of the parameter

We will see later why structures are useful

Invoking the numerical solver

The call is of the form (from the help):

```
ode23, ode45, ode113, ode15s, ode23s, ode23t, ode23tb

Solve initial value problems for ordinary differential
equations

Syntax
[T,Y] = solver(odefun,tspan,y0)
[T,Y] = solver(odefun,tspan,y0,options)
[T,Y,TE,YE,IE] = solver(odefun,tspan,y0,options)
sol = solver(odefun,[t0 tf],y0...)

where solver is one of ode45, ode23, ode113, ode15s,
ode23s, ode23t, or ode23tb
```

Typically, you can use `ode45`

Computing the numerical solution to the logistic

We call our solver as follows:

```
tspan=[1790 2000]; %The time span of the solution
IC=3.929;          %The initial condition (in 1790)
p.K=300;           %Set the parameters
p.r=0.5;
[t,N]=ode45(@rhs_logistic,tspan,IC,[],p);
```

(The one before last argument, `[]`, represents the options structure. Here we are not modifying any option, and so pass an empty vector)

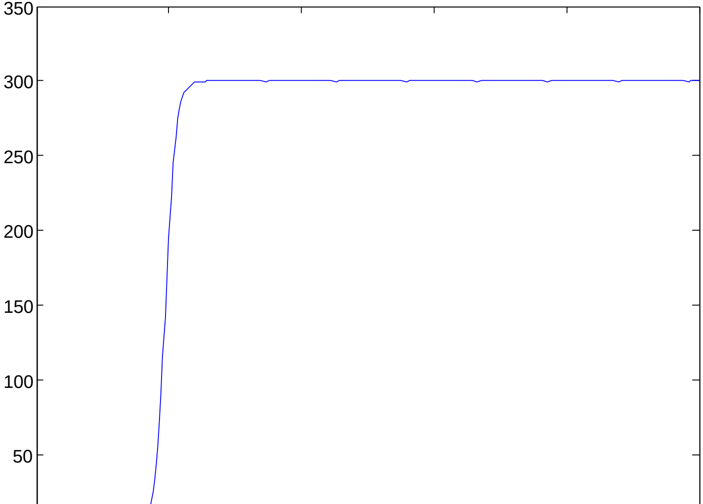
Save this file as, say, `call_solver.m`

After running it, we have a vector `t` of times (covering `tspan`) and a vector `N` of solution

Plotting the solution

```
plot(t,N)
```

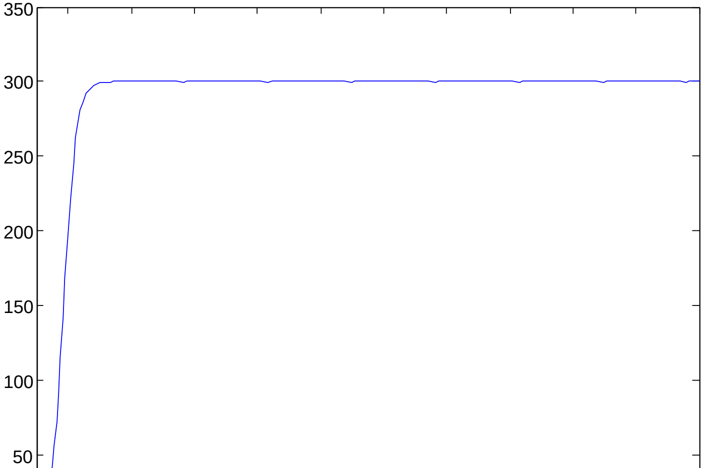
gives



Tightening the x-axis

```
plot(t,N)
xlim([t(1) t(end)])
```

gives



Using Octave

The syntax in Octave is almost identical to the matlab syntax. In fact, if you use the additional programs in the `forge` repository, a function `ode45` is defined

However, the functions (in octave) do not implement the use of a parameter by default, so a work-around must be used

Update: as of V3.0 and using ode45, parameters can be passed and the matlab code given before works, with the following little modification:

```
opt=odeset('InitialStep',0.05,'MaxStep',1);
[t,N]=ode45(@rhs_logistic,tspan,IC,opt,p);
```

which makes sure that the time step does not become too large)

Using scilab

The syntax in scilab differs a little from matlab, so beware.

```
function ydot=f(t,y);
ydot=y^2-y*sin(t)+cos(t);
endfunction
```