Tutorial for 'fitting' experimental data.

Grad School course (SysBioWorkshop) - 5th May 2017
philippe.robert@ens-lyon.org

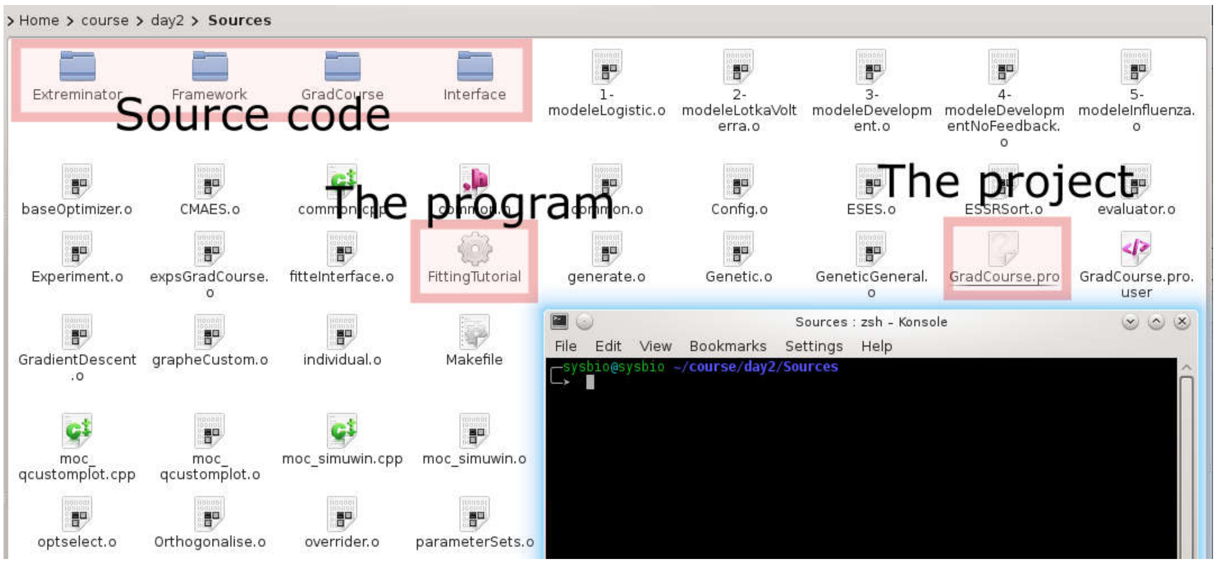
Getting started:

Open folder Course/Day2/Sources/

At any time, to open command line,
Inside a folder, right click, Actions, Open a terminal here.

To configure the keyboard in another language,
Open a command line (wherever)
setxkbmap de
other options: us

Open folder Course/Day2/Sources/
and open a terminal in this folder.



```
┌sysbio@sysbio ~/course/day2/Sources
└╼ █
```

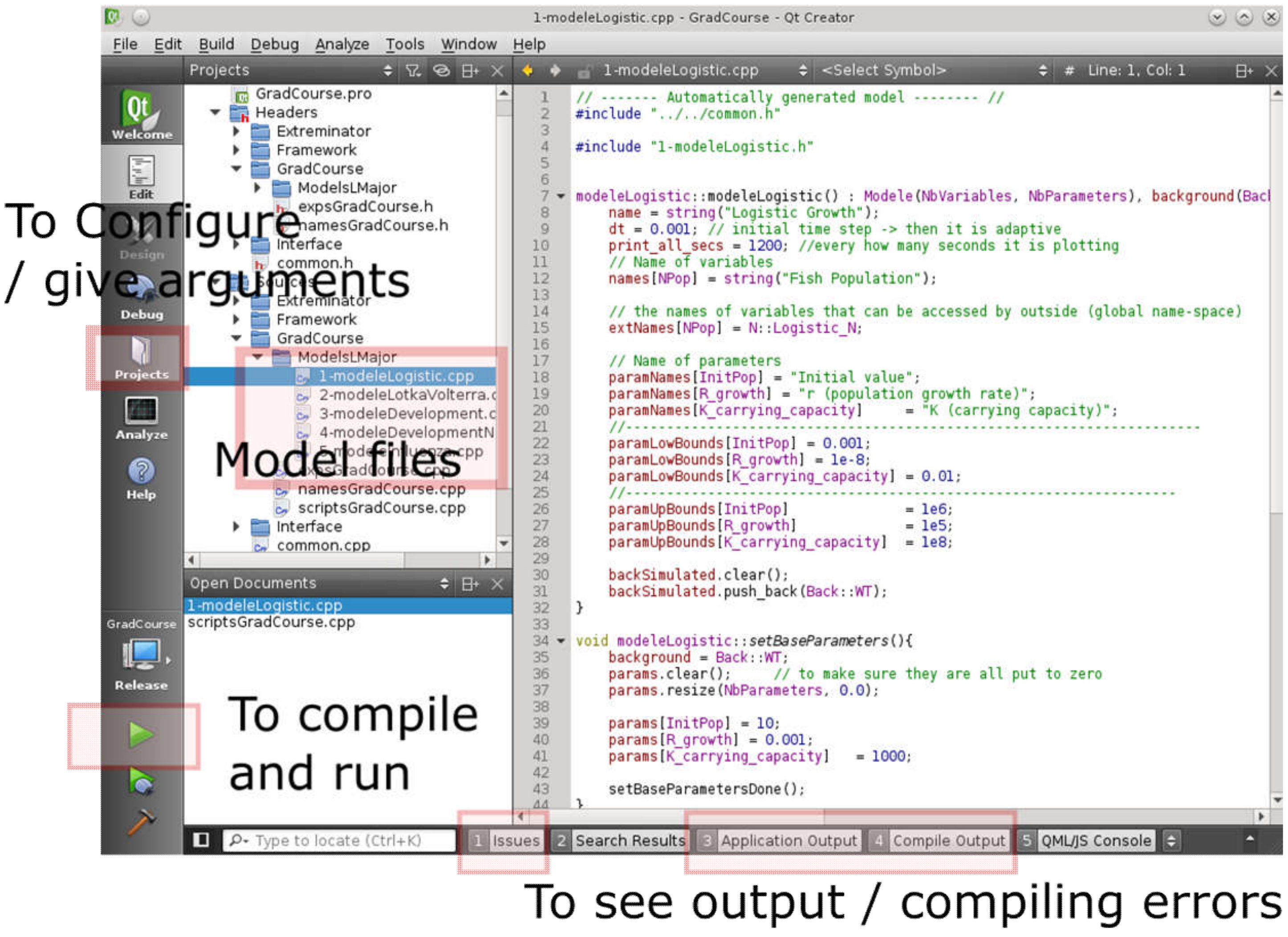
Simplest way to run the code:
in the terminal, for each part, you will need to call the program, and to give the number of the part to run. : type: (at any time, type TAB to get help finding files)

ex: For running part 1:

```
./FittingTutorial 1
```

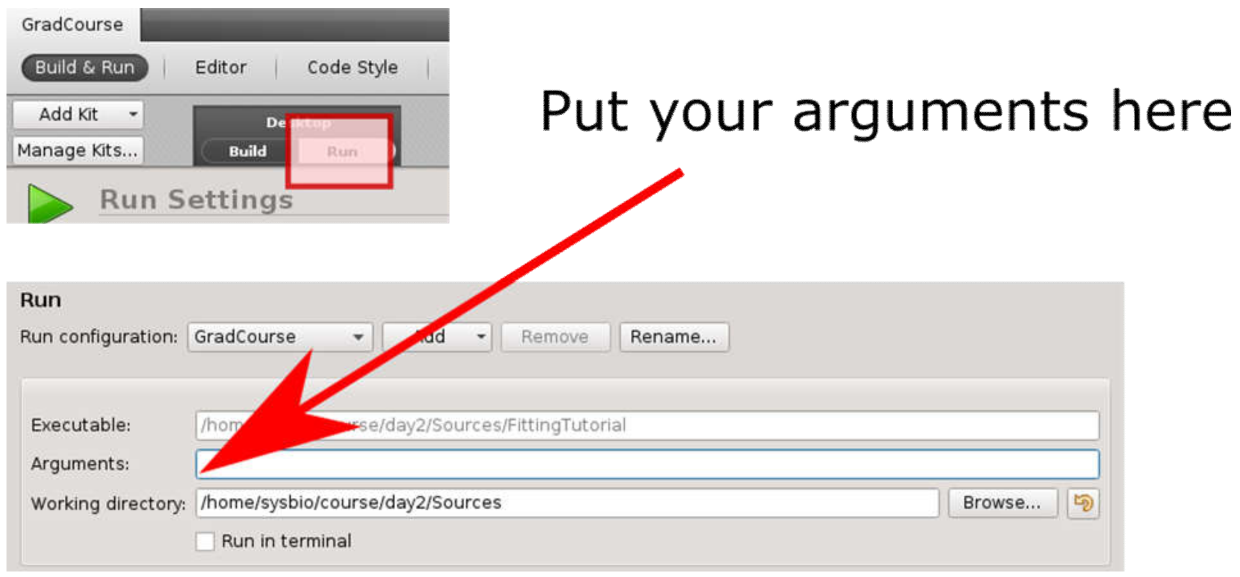
Programmer way to use the code:

open the file **"GradCourse.pro"** (by double clicking). It will open in QtCreator.
click on 'Configure Project'. (it happens at every change of computer).



```cpp
// ------- Automatically generated model -------- //
#include "../../common.h"

#include "1-modeleLogistic.h"


modeleLogistic::modeleLogistic() : Modele(NbVariables, NbParameters), background(Bac
    name = string("Logistic Growth");
    dt = 0.001; // initial time step -> then it is adaptive
    print_all_secs = 1200; //every how many seconds it is plotting
    // Name of variables
    names[NPop] = string("Fish Population");

    // the names of variables that can be accessed by outside (global name-space)
    extNames[NPop] = N::Logistic_N;

    // Name of parameters
    paramNames[InitPop] = "Initial value";
    paramNames[R_growth] = "r (population growth rate)";
    paramNames[K_carrying_capacity]     = "K (carrying capacity)";
    //----------------------------------------------------------------------
    paramLowBounds[InitPop] = 0.001;
    paramLowBounds[R_growth] = 1e-8;
    paramLowBounds[K_carrying_capacity] = 0.01;
    //-------------------------------------------------------------------
    paramUpBounds[InitPop]              = 1e6;
    paramUpBounds[R_growth]             = 1e5;
    paramUpBounds[K_carrying_capacity]  = 1e8;

    backSimulated.clear();
    backSimulated.push_back(Back::WT);
}

void modeleLogistic::setBaseParameters(){
    background = Back::WT;
    params.clear();     // to make sure they are all put to zero
    params.resize(NbParameters, 0.0);

    params[InitPop] = 10;
    params[R_growth] = 0.001;
    params[K_carrying_capacity]   = 1000;

    setBaseParametersDone();
}
```

To configure arguments, click on 'Projects', then on 'Run'



Run configuration: GradCourse

Executable: /hom…se/day2/Sources/FittingTutorial

Arguments:

Working directory: /home/sysbio/course/day2/Sources

☐ Run in terminal

=> Put as argument **1** or **2** or ... for each part of the tutorial.

Part 0: First fitting, logistic growth.

Biological problem : You measure the population of fishes in a lake over time.

Collected data:

months	Fish Number
0	20
3	40
6	100
12	375
15	530
18	690
24	790
30	840
40	850

Model: Let suppose an exponential growth with maximum carrying capacity (Logistic growth).

$$\frac{dP}{dt} = rP \cdot \left(1 - \frac{P}{K}\right)$$

K = carrying capacity of the lake,
r = proliferation growth

Questions: what is

1/ the carrying capacity of the lake
2/ the growth rate of the fish
3/ Let say we put 50 fishes in the lake, what will be the time of maximal growth ?
(I.e. the best time to harvest fish !)

=> Open a terminal in the course/day2/Sources/ folder, and call
./FittingTutorial 0
(or with QtCreator, put 0 as argument and run with the green arrow).
See next pages for interface explanation.

=> Your fits

	Best Fitting
r=	
K=	
cost=	
best harvesting at t=	

Questions: Can you trust the parameters you found ? **(Identifiability)**
Optimize again, but with a fixed r value, around the best r value you found:

	50% of r	90% of r	95% of r	Best r	110% of r	150% of r	200% of r
Fixed r value:							
Cost=							
K=							

Questions: Can you trust your prediction **(Sensitivity). T**est if your prediction for optimal picking of fishes is dependent on each parameter. From your best parameter set, modify the parameters one by one around their 'best' value and look at the effect on your prediction.

	50% of r	90% of r	95% of r	Best r	110% of r	150% of r	200% of r
Cost (without fitting)							
Your prediction (best time)							
	50% of K	90% of K	95% of K	Best K	110% of K	150% of K	200% of K
Cost (without fitting)							
Your prediction (best time)							

Anatomy of a fitting.

What you need to decide / know before a fitting :

0/ A model, namely:
- A list of variables (and their equations)
- A list of parameters

1/ For a simulation, you need initial values and parameter values.
- What are the **initial values** (are they known ? => do they need to be estimated from data as well ?)
- Which parameter values are already known (from literature or other experiments).

2/ How are the equations solved ?
- Euler method ? Runge kutta ? Adaptive methods ?
- Here it is an **adaptive Runge Kutta 4** (you can find details in Framework/Modele.cpp)
- What is a good dt for simulations ?

3/ Which cost function do you give to compare a simulation to data ?
- Here, it is the sum of |log(data[x,t]) - log(simulation[x,t])| for each time point t of each variable x.
- It can also be the sum of |data[x,t]^2 - simulation[x,t]|, (called RSS), and can also be normalized by standard deviation for each datapoint.

4/ For finding parameters ("Parameter Estimation"), you need

4a/ What are the boundaries of possible parameter values for the parameters to find.

4b/ Which optimizer to use.

[Optimizer: It is a program that takes your problem as a black box:
It knows a list of parameters to find and their boundaries.
It calls your simulations for millions of parameter sets, and asks for your "cost" (compared to experiment).
Output : it returns the best parameter set]

Here, the optimization is an **"Evolutionary Strategy"** optimizer. (details in "GeneticGeneral.cpp")
- How many iterations will be needed to find a good parameter ?
- Usually, minimum 100 x number of parameters.
- Depending on the optimizer, it might have its own parameters : population size, etc...

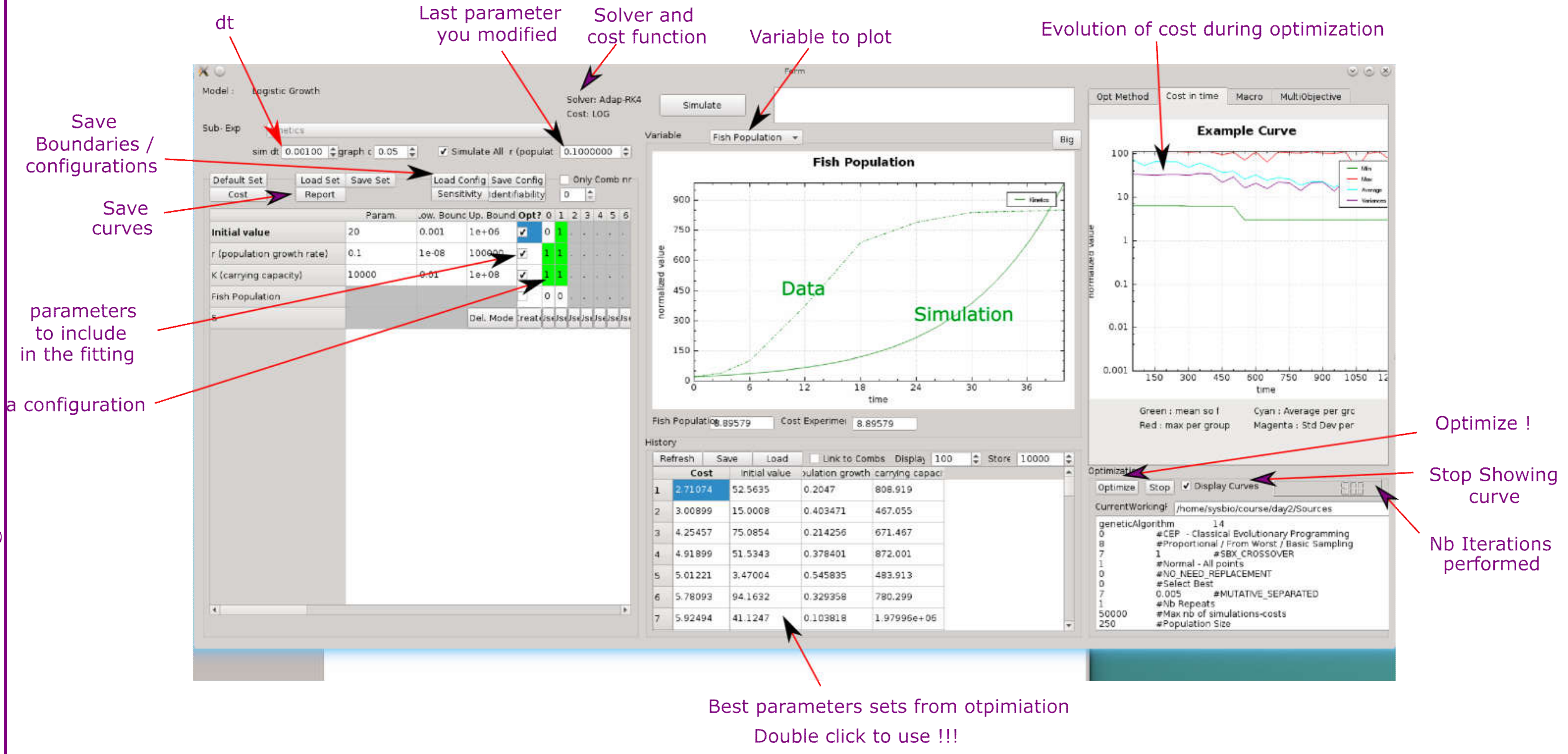
4c/ How are the parameters scaled for optimization:
- Here, the parameters are fitted in a "Logarithmic" scale.
- It allows all orders of magnitude to have the same weight.
  - **ex:** if parameter K can be between 10e-5 and 10e3. Usually, optimizers start from a random, uniform guess of parameters. Without scaling, there is much more probabilities that the optimizer will try parameters between 10e2 and 10e3 (compared to between 10e-5 and 1 for instance). Thanks to logarithmic scaling, the optimizer has equal chances to look in the intervals [10e-5 - 10e-4], [10e-4 - 10e-3], [10e-3 - 10e-2] ...

In a paper using fitting, ALL THESE INFORMATIONS have to be communicated.

First example:

Variables:	N
Parameters:	2 : r and k
Initial values:	Known : N(0) = 20
Parameters already known:	None
Solver:	Adaptive RK4
dt and unit	0.05 months (you can play with it)
Boundaries for r	? => take large : 1e-8 - 1e6
Boundaries for K	? => take large : 0.01 - 1e8
Cost Function:	Log: Sum(log(simu) - log(data))
Optimizer:	Evolutionary Strategies
Number of Iterations	take large : 50000.
Parameter Scaling:	Logarithmic scale

How to make a fitting within the tutorial:



Model : Logistic Growth

Solver: Adap-RK4
Cost: LOG

sim dt 0.00100 graph c 0.05 ✓ Simulate All r (populat 0.1000000

	Param.	Low. Bound	Up. Bound	Opt?	0	1
Initial value	20	0.001	1e+06	✓	0	1
r (population growth rate)	0.1	1e-08	100000	✓	1	1
K (carrying capacity)	10000	0.01	1e+08	✓	1	1
Fish Population					0	0

Fish Population 8.89579 Cost Experiment 8.89579

History

	Cost	Initial value	population growth	carrying capacity
1	2.71074	52.5635	0.2047	808.919
2	3.00899	15.0008	0.403471	467.055
3	4.25457	75.0854	0.214256	671.467
4	4.91899	51.5343	0.378401	872.001
5	5.01221	3.47004	0.545835	483.913
6	5.78093	94.1632	0.329358	780.299
7	5.92494	41.1247	0.103818	1.97996e+06

Green : mean so f Cyan : Average per grc
Red : max per group Magenta : Std Dev per

CurrentWorking: /home/sysbio/course/day2/Sources

```
geneticAlgorithm         14
0          #CEP  - Classical Evolutionary Programming
8          #Proportional / From Worst / Basic Sampling
7          1          #SBX_CROSSOVER
1          #Normal - All points
0          #NO_NEED_REPLACEMENT
0          #Select Best
7          0.005          #MUTATIVE_SEPARATED
1          #Nb Repeats
50000      #Max nb of simulations-costs
250        #Population Size
```

Part 1: Same fitting, logistic growth, but different dataset.

New lake, new fishes, new dataset !

Months	Fish numbers
0	20
3	22,5
6	25
12	31
15	34
18	38
24	44
30	51
40	65

=> Do ./FittingTutorial 1 (or with QtCreator, put 1 as argument and run).

Questions: what is

1/ the carrying capacity of the lake
2/ the growth rate of the fish
3/ Which parameters can you trust from the data ?

Variables:	
Parameters:	
Initial values:	
Parameters already known:	
Solver:	Adaptive RK4
dt and unit	
Boundaries for r	
Boundaries for K	
Cost Function:	Log: Sum(log(simu) - log(data))
Optimizer:	Evolutionary Strategies
Number of Iterations	
Parameter Scaling:	Logarithmic scale

interface explanation.

=> Your fits

	Best Fitting
r=	
K=	
cost=	
best harvesting at t=	

Questions: Can you trust the parameters you found ? **(Identifiability)**
Optimize again, but with a fixed r value, around the best r value you found:

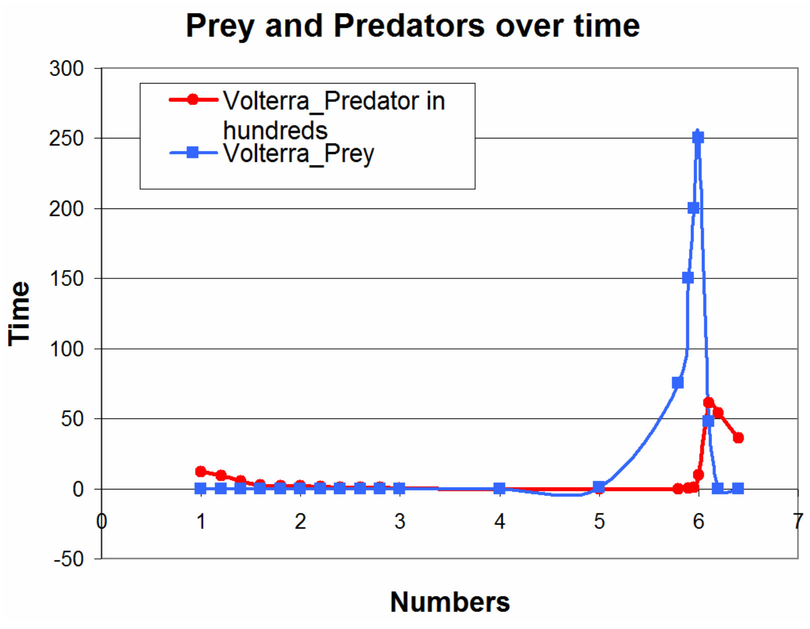
	50% of r	90% of r	95% of r	Best r	110% of r	150% of r	200% of r
Fixed r value:							
Cost=							
K=							

Part 2: Fittings two datasets : preys and predators.

Let say we are following the number of two species of animals : preys and predators
The prey (x), proliferates with rate α, but is eaten by the predator with rate β
The predator (y), proliferates with a rate (δ.x(t)), meaning that they proliferate more if they eat more.
their death rate γ is supposed constant.
These equations are called Lotka-Volterra equations:

$$\begin{cases} \dfrac{\mathrm{d}x(t)}{\mathrm{d}t} = x(t)\left(\alpha - \beta y(t)\right) \\ \dfrac{\mathrm{d}y(t)}{\mathrm{d}t} = -y(t)\left(\gamma - \delta x(t)\right) \end{cases}$$

Dataset :



Years	Volterra_Predator	Volterra_Prey
1	1200	0
1,2	950	0
1,4	550	0
1,6	280	0
1,8	200	0
2	180	0
2,2	130	0
2,4	100	0
2,6	80	0
2,8	65	0
3	40	0
4	2	0
5	1	1
5,8	1	75
5,9	40	150
5,95	100	200
6	1000	250
6,1	6100	48
6,2	5400	0
6,4	3600	0

Questions :

1/ Can you find a good set of parameters to explain the data with a usual fitting ?
Suggestion: try to guess good boundaries from parameter sets.

2/ Perform **iterative fitting**, (meaning that you replace one curve by data and fit the other one).
(from the interface, you do it by clicking the box 'opt' for the variables to be overriden by data).

3/ (Global fitting around the iterative fittings)
Then, use the best parameters found from both fittings and fitte again with close boundaries around the best parameters. Does it improve the cost of fitting ?

=> Do ./FittingTutorial 2 (or with QtCreator, put 2 as argument and run).

Your results:

	Global fitting (1)	Iterative fitting for Prey (2)	Iterative fitting for Predator (2)	Global fitting around iterative fittings (3)
Alpha (Prey repro)				
Beta (Capture Prey)				
Gamma (Death Predator)				
Delta (Food dep prolif predator)				
cost=				
		(partial cost)	(partial cost)	

Part 3: Differentiation of cell populations, without feedback.

You measure the amount of 3 populations of cells in an organ, namely, two steps of progenitors, and a mature population.

P1 :	Progenitor 1
P2 :	Progenitor 2
P3 :	Mature cells

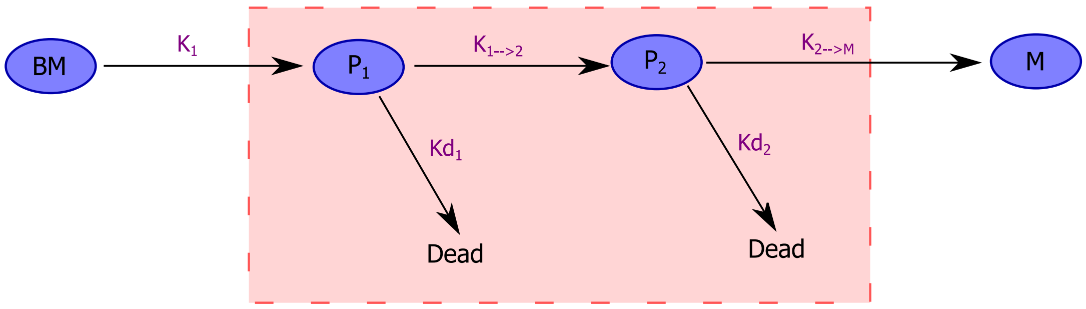
You have also a **deficient mice** for your gene, that show an interesting dynamics of these progenitors, but you don't know where the gene has an effect. You are interested by two biological questions :

1/ Where is the gene likely to have an effect, and on which population ? Proliferation ? Death ?

2/ Do the mature cells exert a negative feedback on the development of the progenitors ?

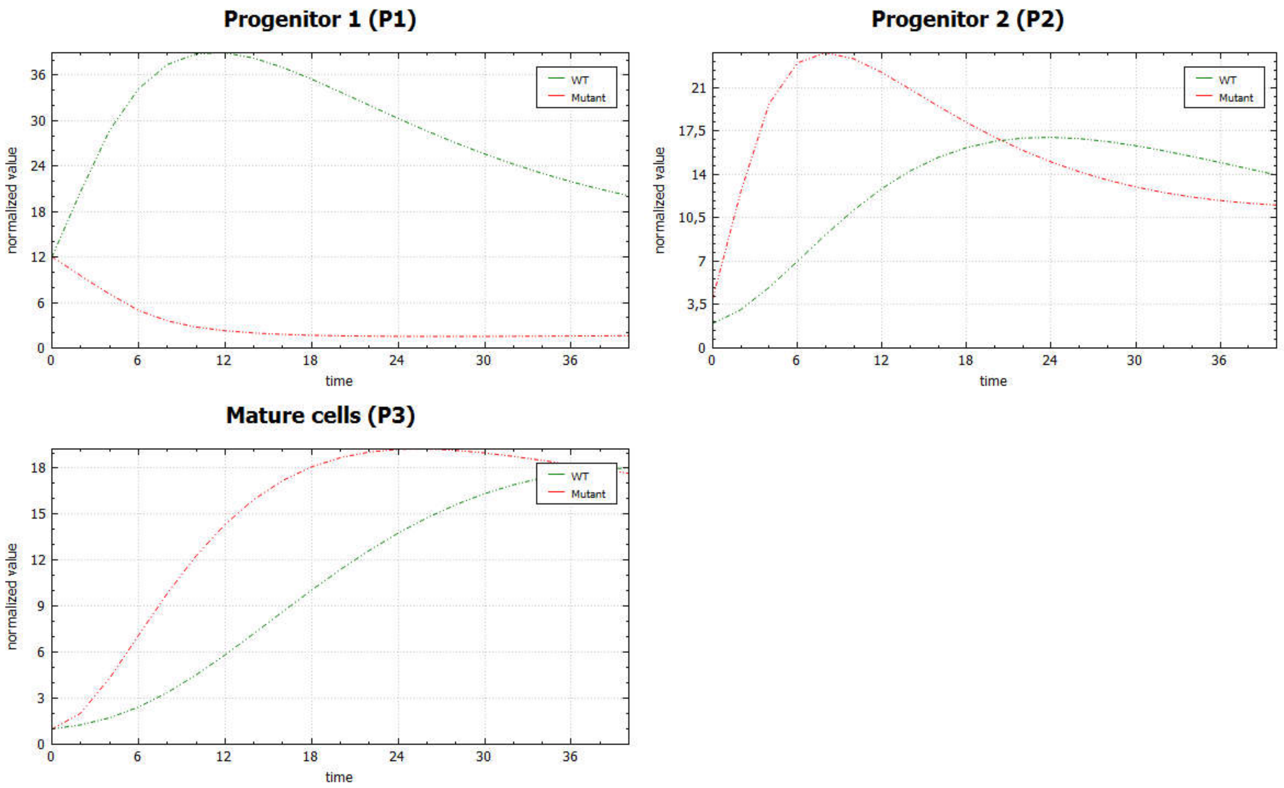
(literature is conflicting on this subject, you would like to know if your data can answer it).

So we start from a simple model, (without feedback yet):



$$\frac{dT_1}{dt} = K_1 - Kd_1.T_1 - K_{1\to 2}.T_1$$
$$\frac{dT_2}{dt} = K_{1\to 2}.T_1 - K_{2\to M}.T_2 - Kd_2.T_2$$

Dataset :



=> Do ./FittingTutorial 3 (or with QtCreator, put 3 as argument and run).

Questions :

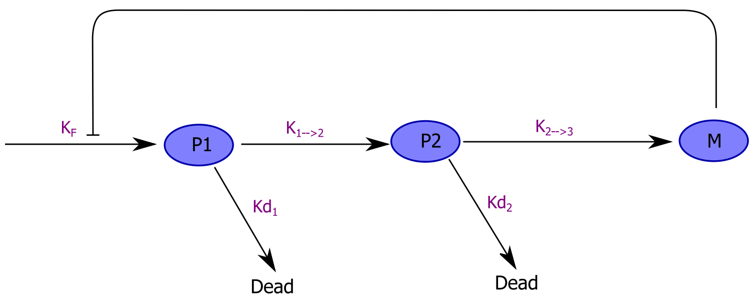
1/ Can you find a good parameter set with this model for the WT curve ?

Note: Put the mutant parameters as '1' to have no effect,
and override the Mut data to have no effect on the cost

2/ Can you find a good parameter set where only one mutant parameter is not '1'

Part 4: Differentiation of cell populations, with feedback.

Now, the model is including a feedback:



Equations :

```
dxdt[P1] = params[kinput] * (params[kinhib3_1] / (params[maxinhib3_1] * (x[P3] +
      params[kinhib3_1]))) + x[P1] * (- params[usedkd1] - params[usedk1_2]);
dxdt[P2] = x[P1] * params[usedk1_2] +  x[P2] * (- params[kd2] - params[k2_3]);
dxdt[P3] = x[P2] * params[k2_3] - x[P3] * params[k3_out] - params[kd3] * x[P3];
```

=> Do ./FittingTutorial 4 (or with QtCreator, put 4 as argument and run).

Questions : with a feedback, can you explain better the data / the mutant phenotype ?

Part 5: Influenza dynamics.

Here is a model for :

U: uninfected cells,
I : Infected cells,
V : virus load,
T : CD8+ T cells

$$\frac{dU}{dt} = -\beta UV$$
$$\frac{dI}{dt} = \beta UV - \delta IT$$
$$\frac{dV}{dt} = pI - cV$$
$$\frac{dT}{dt} = S_T + \rho TV - \delta_T T$$

A dataset for virus load over time is available in the program, and you can try to see which parameters you can find for it :

=> Do ./FittingTutorial 5 (or with QtCreator, put 5 as argument and run).