How Inexact Models Can Guide Decision Making in Quantitative Systems Pharmacology

Christopher Rackauckas[1]

[1]Affiliation not available

April 17, 2023

BIOLOGICAL SCIENCES

How Inexact Models Can Guide Decision Making in Quantitative Systems Pharmacology

CHRISTOPHER RACKAUCKAS

Pre-clinical Quantitiative Systems Pharmacology (QSP) is about trying to understand how a drug target effects an outcome. If I effect this part of the biological pathways, how will it induce toxicity? Will it be effective?

Recently I have been pulling in a lot of technical collegues to help with the development of next generation QSP tooling. Without a background in biological modeling, I found it difficult to explain the "how" and "why" of pharmacological modeling. Why is it differential equations, and where do these "massively expensive global optimization" runs come from? What kinds of problems can you solve with such models when you know that they are only approximate?

To solve these questions, I took a step back and tried to explain a decision making scenario with a simple model, to showcase how playing with a model can allow one to distinguish between intervention strategies and uncover a way forward. This is my attempt. Instead of talking about something small and foreign like chemical reaction concentrations, let's talk about something mathematically equivalent that's easy to visualize: ecological intervention.

BASIC MODELING AND FITTING

Let's take everyone's favorite ecology model: the Lotka-Volterra model. The model is the following:

- Left alone, the rabbit population will grow exponentially

- Rabbits are eaten wolves in proportion to the number of wolves (number of mouthes to feed), and in proportion to the number of rabbits (ease of food access: you eat more at a buffet!)

- Wolf populations grow exponentially, as long as there is a proportional amount of food around (rabbits)

- Wolves die overtime of old age, and any generation dies at a similar age (no major wolf medical discoveries)

The model is then the ODE:

```
using OrdinaryDiffEq, Plots

function f(du,u,p,t)
  du[1] = dx = p[1]*u[1] - p[2]*u[1]*u[2]
  du[2] = dy = -p[3]*u[2] + p[4]*u[1]*u[2]
end

u0 = [1.0;1.0]
tspan = (0.0,10.0)
p = [1.5,1.0,3.0,1.0]
prob = ODEProblem(f,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol,label=["Rabbits" "Wolves"])
```

Except, me showing you that picture glossed over a major detail that every piece of the model is only mechanistic, but also contains a parameter. For example, rabbits grow exponentially, but what's the growth rate? To make that plot I chose a value for that growth rate (1.5), but in reality we need to get that from data since the results can be wildly different:

```
p = [0.1,1.0,3.0,1.0]
prob = ODEProblem(f,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
```

Here the exponential growth rate of rabbits too low to sustain a wolf population, so the wolf population dies out, but then this makes the rabbits have no predators and grow exponentially, which is a common route of ecological collapse as then they will destroy the local ecosystem. More on that later.

DATA AND MODEL ISSUES

But okay, we need parameters from data, but no single data source is great. One gives us a noisy sample of the population yearly, another every month for the first two years and only on the wolves, etc.:

```
function f_true(du,u,p,t)
  du[1] = dx = p[1]*u[1] - p[2]*u[1]*u[2] - p[5]*u[1]^2
  du[2] = dy = -p[3]*u[2] + p[4]*u[1]*u[2]
end

p = [1.0,1.0,3.0,1.0,0.1]
prob = ODEProblem(f_true,u0,tspan,p)
sol = solve(prob,Tsit5())

data1 = sol(0:1:10)
data2 = sol(0:0.1:2;idxs=2)
scatter(data1)
scatter!(data2)
```

Oh, and notice that ODE is not the Lotka-Volterra model, but instead also adds a term$p[5]*u[1]^2$ for a rabbit disease which requires high rabbit density.

ASSESSING INTERVENTION

The local population is very snobby and wants the rabbit population decreased. You're trying to find out how to best intervene with the populations so that you decrease the rabbit population to always stay below 4 million rabbits, but without causing population collapse. What should you be targetting? The rabbit birth rate? The ability for predators to find rabbits?

(In systems pharmacology, this is, which reactions should I interact with in order to achieve my target goals while not introducing toxic side effects?)

In a complex system, these will all act differently, so you need to simulate what happens under uncertainty with the model. For example, if I attack birth rate too hard, we already saw that we can lead to population collapse, but is birth rate a more robust target then wolf lifespan (i.e., could I change wolf lifespan more and get the same effect, but with less chance of collapse)?

THE MODELER'S PROBLEM

But one caveat: in order to do these simulations you need to know the model and its parameters, since you want to investigate what happens when you change the model's parameters. But you just have "your best model" and "data". So you need to find out how to get "the best model you can" and the parameters of said model, since once you have that you can assess the targetting effects.

THE MODELER'S APPROACH

Let's start with the model we have:

```
function f(du,u,p,t)
  du[1] = dx = p[1]*u[1] - p[2]*u[1]*u[2]
  du[2] = dy = -p[3]*u[2] + p[4]*u[1]*u[2]
end

u0 = [1.0;1.0]
tspan = (0.0,10.0)
p = ones(4)
prob = ODEProblem(f,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
```

Here I took all of the parametrs to be1, since my only guess is their relative order of magnitude which should be about correct (maybe?).

To get some parameter values, I then need do some parameter fitting. Let's define a cost function as a difference of our result against the data sources we have:

```
using LinearAlgebra
function cost(p)
  # Needed for the optimizer to reject parameters out of the domain
  any(x->x
10.917779941345977
```

Yeah, those original parameters are pretty bad. But now let's optimize them:

```
using Optim
opt = optimize(cost,ones(4),BFGS())
p = Optim.minimizer(opt)
prob = ODEProblem(f,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
scatter!(data1)
scatter!(data2)
```

Oh wow, that fit is pretty bad! Well generally, for large models (the models are usually >200 lines long), this is pretty standard for a local opti

```
using BlackBoxOptim
bound = Tuple{Float64, Float64}[(0, 10),(0, 10),(0, 10),(0, 10)]
result = bboptimize(cost;SearchRange = bound, MaxSteps = 21e3)
p = result.archive_output.best_candidate
prob = ODEProblem(f,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
scatter!(data1)
scatter!(data2)
```

```
Starting optimization with optimizer BlackBoxOptim.DiffEvoOpt{BlackBoxOptim
.FitPopulation{Float64},BlackBoxOptim.RadiusLimitedSelector,BlackBoxOptim.A
daptiveDiffEvoRandBin{3},BlackBoxOptim.RandomBound{BlackBoxOptim.Continuous
RectSearchSpace}}
0.00 secs, 0 evals, 0 steps
0.50 secs, 521 evals, 382 steps, improv/step: 0.317 (last = 0.3168), fitnes
s=6.101131069
1.00 secs, 1419 evals, 1253 steps, improv/step: 0.241 (last = 0.2078), fitn
ess=4.441611914
1.50 secs, 2430 evals, 2264 steps, improv/step: 0.204 (last = 0.1573), fitn
ess=3.039137898
2.00 secs, 3391 evals, 3225 steps, improv/step: 0.190 (last = 0.1592), fitn
ess=2.959508579
2.50 secs, 4700 evals, 4535 steps, improv/step: 0.195 (last = 0.2069), fitn
ess=2.936981210
3.01 secs, 5985 evals, 5821 steps, improv/step: 0.192 (last = 0.1827), fitn
ess=2.936675847
3.51 secs, 7211 evals, 7048 steps, improv/step: 0.192 (last = 0.1874), fitn
ess=2.936639907
4.01 secs, 8400 evals, 8238 steps, improv/step: 0.186 (last = 0.1555), fitn
ess=2.936639382
4.51 secs, 9470 evals, 9309 steps, improv/step: 0.184 (last = 0.1662), fitn
ess=2.936639323
5.01 secs, 10567 evals, 10406 steps, improv/step: 0.184 (last = 0.1814), fi
tness=2.936639322
5.51 secs, 11665 evals, 11504 steps, improv/step: 0.185 (last = 0.1940), fi
tness=2.936639322
6.01 secs, 12659 evals, 12498 steps, improv/step: 0.187 (last = 0.2113), fi
tness=2.936639322
```

```
6.51 secs, 13866 evals, 13705 steps, improv/step: 0.185 (last = 0.1698), fi
tness=2.936639322
7.01 secs, 15059 evals, 14898 steps, improv/step: 0.179 (last = 0.1106), fi
tness=2.936639322
7.51 secs, 16209 evals, 16048 steps, improv/step: 0.170 (last = 0.0478), fi
tness=2.936639322
8.02 secs, 17394 evals, 17233 steps, improv/step: 0.160 (last = 0.0211), fi
tness=2.936639322
8.52 secs, 18612 evals, 18458 steps, improv/step: 0.151 (last = 0.0286), fi
tness=2.936639322
9.02 secs, 19813 evals, 19970 steps, improv/step: 0.142 (last = 0.0337), fi
tness=2.936639322

Optimization stopped after 21001 steps and 9.23 seconds
Termination reason: Max number of steps (21000) reached
Steps per second = 2275.05
Function evals per second = 2201.39
Improvements/step = 0.13638
Total function evaluations = 20321


Best candidate found: [0.766198, 1.31019, 2.92441, 1.12677]

Fitness: 2.936639322

p = result.archive_output.best_candidate
prob = ODEProblem(f,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
scatter!(data1)
scatter!(data2)
```

Oh dang, still no good. This means our model is misspecified. We see that we overshoot over time, so there's some kind of decay missing. T

So then we change our model. What if this disease is just old age related? Then we would have decay of rabbits unrelated to wolves, so the

```
function f2(du,u,p,t)
  du[1] = dx = p[1]*u[1] - p[2]*u[1]*u[2] - p[5]*u[1]
  du[2] = dy = -p[3]*u[2] + p[4]*u[1]*u[2]
end
```

So now let's optimize with this:

```
function cost2(p)
  # Needed for the optimizer to reject parameters out of the domain
  any(x->x
Starting optimization with optimizer BlackBoxOptim.DiffEvoOpt{BlackBoxOptim
.FitPopulation{Float64},BlackBoxOptim.RadiusLimitedSelector,BlackBoxOptim.A
daptiveDiffEvoRandBin{3},BlackBoxOptim.RandomBound{BlackBoxOptim.Continuous
RectSearchSpace}}
0.00 secs, 0 evals, 0 steps
0.50 secs, 772 evals, 672 steps, improv/step: 0.257 (last = 0.2574), fitnes
s=5.195430548
1.01 secs, 1633 evals, 1533 steps, improv/step: 0.187 (last = 0.1312), fitn
ess=4.538806718
1.51 secs, 2744 evals, 2644 steps, improv/step: 0.148 (last = 0.0954), fitn
ess=3.632220569
2.02 secs, 3772 evals, 3672 steps, improv/step: 0.136 (last = 0.1031), fitn
ess=3.120794160
2.52 secs, 4962 evals, 4862 steps, improv/step: 0.137 (last = 0.1395), fitn
ess=2.945033178
3.02 secs, 6061 evals, 5961 steps, improv/step: 0.137 (last = 0.1365), fitn
ess=2.938642172
```

```
3.52 secs, 7226 evals, 7126 steps, improv/step: 0.141 (last = 0.1648), fitn
ess=2.936764046
4.03 secs, 8494 evals, 8395 steps, improv/step: 0.141 (last = 0.1418), fitn
ess=2.936657641
4.53 secs, 9641 evals, 9542 steps, improv/step: 0.141 (last = 0.1386), fitn
ess=2.936639983
5.03 secs, 10755 evals, 10656 steps, improv/step: 0.145 (last = 0.1759), fi
tness=2.936639353
5.53 secs, 11947 evals, 11848 steps, improv/step: 0.143 (last = 0.1334), fi
tness=2.936639326
6.03 secs, 13181 evals, 13082 steps, improv/step: 0.145 (last = 0.1548), fi
tness=2.936639322
6.53 secs, 14524 evals, 14425 steps, improv/step: 0.144 (last = 0.1355), fi
tness=2.936639322
7.03 secs, 15717 evals, 15619 steps, improv/step: 0.144 (last = 0.1516), fi
tness=2.936639322
7.53 secs, 16975 evals, 16877 steps, improv/step: 0.144 (last = 0.1447), fi
tness=2.936639322
8.03 secs, 18020 evals, 17923 steps, improv/step: 0.144 (last = 0.1424), fi
tness=2.936639322
8.53 secs, 18811 evals, 18714 steps, improv/step: 0.143 (last = 0.1087), fi
tness=2.936639322
9.04 secs, 19671 evals, 19574 steps, improv/step: 0.139 (last = 0.0477), fi
tness=2.936639322
9.54 secs, 20122 evals, 20025 steps, improv/step: 0.136 (last = 0.0377), fi
tness=2.936639322
10.04 secs, 20940 evals, 20843 steps, improv/step: 0.132 (last = 0.0220), f
itness=2.936639322


Optimization stopped after 21001 steps and 10.10 seconds
Termination reason: Max number of steps (21000) reached
Steps per second = 2078.48
Function evals per second = 2088.08
Improvements/step = 0.13081
Total function evaluations = 21098


Best candidate found: [8.17578, 1.31019, 2.92441, 1.12677, 7.40958]

Fitness: 2.936639322

p = result.archive_output.best_candidate
prob = ODEProblem(f2,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
scatter!(data1)
scatter!(data2)
```

Okay, that model is not right yet, but it's better. Let's try the local one and see what we get:

```
opt = optimize(cost2,ones(5),BFGS())
p = Optim.minimizer(opt)
prob = ODEProblem(f2,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
scatter!(data1)
scatter!(data2)
```

So we go back to our colleagues who know the local ecology and tell them, that model is giving me junk. Then you learn from one of the res

```
function f3(du,u,p,t)
  du[1] = dx = p[1]*u[1] - p[2]*u[1]*u[2] - p[5]*u[1]^2
  du[2] = dy = -p[3]*u[2] + p[4]*u[1]*u[2]
end
```

```
function cost3(p)
  # Needed for the optimizer to reject parameters out of the domain
  any(x->x
```

That might just be an optimization issue? So let's run our big model with global optimization on the cluster:

```
bound = Tuple{Float64, Float64}[(0, 10),(0, 10),(0, 10),(0, 10),(0, 10)]
result = bboptimize(cost3;SearchRange = bound, MaxSteps = 21e3)
```

```
Starting optimization with optimizer BlackBoxOptim.DiffEvoOpt{BlackBoxOptim
.FitPopulation{Float64},BlackBoxOptim.RadiusLimitedSelector,BlackBoxOptim.A
daptiveDiffEvoRandBin{3},BlackBoxOptim.RandomBound{BlackBoxOptim.Continuous
RectSearchSpace}}
0.00 secs, 0 evals, 0 steps
0.50 secs, 1558 evals, 1421 steps, improv/step: 0.246 (last = 0.2456), fitn
ess=3.862852782
1.00 secs, 3159 evals, 3022 steps, improv/step: 0.195 (last = 0.1499), fitn
ess=1.992867792
1.50 secs, 4336 evals, 4199 steps, improv/step: 0.180 (last = 0.1410), fitn
ess=1.629497857
2.00 secs, 5310 evals, 5173 steps, improv/step: 0.175 (last = 0.1530), fitn
ess=0.373072771
2.50 secs, 6785 evals, 6648 steps, improv/step: 0.162 (last = 0.1186), fitn
ess=0.219904053
3.01 secs, 7974 evals, 7837 steps, improv/step: 0.162 (last = 0.1590), fitn
ess=0.083573607
3.51 secs, 9415 evals, 9278 steps, improv/step: 0.162 (last = 0.1610), fitn
ess=0.019814280
4.01 secs, 10222 evals, 10085 steps, improv/step: 0.162 (last = 0.1660), fi
tness=0.006743899
4.66 secs, 11008 evals, 10871 steps, improv/step: 0.161 (last = 0.1527), fi
tness=0.003019186
5.16 secs, 11850 evals, 11713 steps, improv/step: 0.162 (last = 0.1651), fi
tness=0.003007867
5.66 secs, 12903 evals, 12767 steps, improv/step: 0.162 (last = 0.1679), fi
tness=0.002607472
6.16 secs, 13571 evals, 13436 steps, improv/step: 0.163 (last = 0.1719), fi
tness=0.002523329
6.66 secs, 14991 evals, 14856 steps, improv/step: 0.165 (last = 0.1831), fi
tness=0.002451722
7.17 secs, 16627 evals, 16494 steps, improv/step: 0.165 (last = 0.1679), fi
tness=0.002448788
7.67 secs, 18439 evals, 18306 steps, improv/step: 0.164 (last = 0.1518), fi
tness=0.002448722
8.18 secs, 19939 evals, 19806 steps, improv/step: 0.168 (last = 0.2160), fi
tness=0.002448716
```

```
Optimization stopped after 21001 steps and 8.59 seconds
Termination reason: Max number of steps (21000) reached
Steps per second = 2445.96
Function evals per second = 2461.33
Improvements/step = 0.16895
Total function evaluations = 21133
```

Best candidate found: [0.999909, 1.00034, 3.00008, 1.00015, 0.0999369]

Fitness: 0.002448716

```
p = result.archive_output.best_candidate
```

```
prob = ODEProblem(f3,u0,tspan,p)
sol = solve(prob,Tsit5())
plot(sol)
scatter!(data1)
scatter!(data2)
```

Oh bingo! That looks good! So okay, is it safe to reduce the rabbit birth rate, and how much would we need to target it by to get it under 4?

```
test_p = p - [0.5,0,0,0,0]
prob = ODEProblem(f3,u0,tspan,test_p)
sol = solve(prob,Tsit5())
plot(sol)
```

```
test_p = p - [0.58,0,0,0,0]
prob = ODEProblem(f3,u0,tspan,test_p)
sol = solve(prob,Tsit5())
plot(sol)
```

```
test_p = p - [0.7,0,0,0,0]
prob = ODEProblem(f3,u0,tspan,test_p)
sol = solve(prob,Tsit5())
plot(sol)
```

Ouch. We see that effecting the birth rate, we would have to hit a reduction range of [0.55,0.7] to get the effect we want without introducing

```
test_p = p + [0.0,0,0.0,0.25,0]
prob = ODEProblem(f3,u0,tspan,test_p)
sol = solve(prob,Tsit5())
plot(sol)
```

```
test_p = p + [0.0,0,0.0,0.5,0]
prob = ODEProblem(f3,u0,tspan,test_p)
sol = solve(prob,Tsit5())
plot(sol)
```

```
test_p = p + [0.0,0,1.5,2.0,0]
prob = ODEProblem(f3,u0,tspan,test_p)
sol = solve(prob,Tsit5())
plot(sol)
```

Wow, notice that in 3 very different parameter scenarios ("virtual populations") that it's very easy to intervene this way and get the rabbit pop

BOARD MEETING RECOMMENDATIONS

From these results you go to your board meeting and recommend:

- Do not attempt to decrease the rabbit population by killing baby rabbits. This seems like the obvious approach, but the models show that it can easily lead to population collapse.

- However, making it easier for wolves to feast on rabbits is a robust change to the ecosystem that gives the people what they want.

- We recommend clearing a lot of bushes to give rabbits a harder chance of hiding, along with long-term investment in research for wolf binoculars.

- As we go through our results, we tell our collegues to never use a local optimizer.

- During our talk, we tell our collegues that the global optimization takes for ever, so we should probably parallelize it, or do other things, like change our models into Julia to use their faster tools.

TRANSLATING THIS EXAMPLE BACK TO SYSTEMS PHARMACOLOGY

We have a cancer pathway and our team is synthesizing molecules to target different aspects of the pathway. Our goal as the modeler is to help the guide the choice of which part of the pathway we should target. So we:

1. Talk to biologists and consult the literature to learn about the pathway

2. Build a model

3. Find some data in the literature about how the pathway should generally act

4. Try to fit the data

5. If we don't fit well, go back to (1)

6. Great, we now have a good enough model. Investigate what happens to the cancer patient if we effect X vs Y. Does X introduce toxic effects? Can we know if it only produces toxic effects in men?

7. Gather these plots to showcase that yes, the model gives a reliable fit, and the analysis shows that targeting X will cause ...

So other than the interpretation being different, it's this same workflow. It's this same mantra. Get a model that fits, and then understand the general behavior of the model to learn the best intervention strategy.

POSSIBLE IMPROVEMENTS FOR TOOLING

Given this process, the possible improvements to tooling are:

1. Solve differential equations faster.

2. Do global optimization in less steps.

3. Parallelize the global optimization.

4. Automatically find models from data for people?

However, any tooling needs to respect the interactive aspect between the modeler, the biology, the data, and the interpretation of the results.