Lotka-Volterra Bayesian Parameter Estimation Benchmarks

Vaibhav Dixit, Chris Rackauckas

July 5, 2020

0.1 Parameter Estimation of Lotka-Volterra Equation using DiffEqBayes.jl

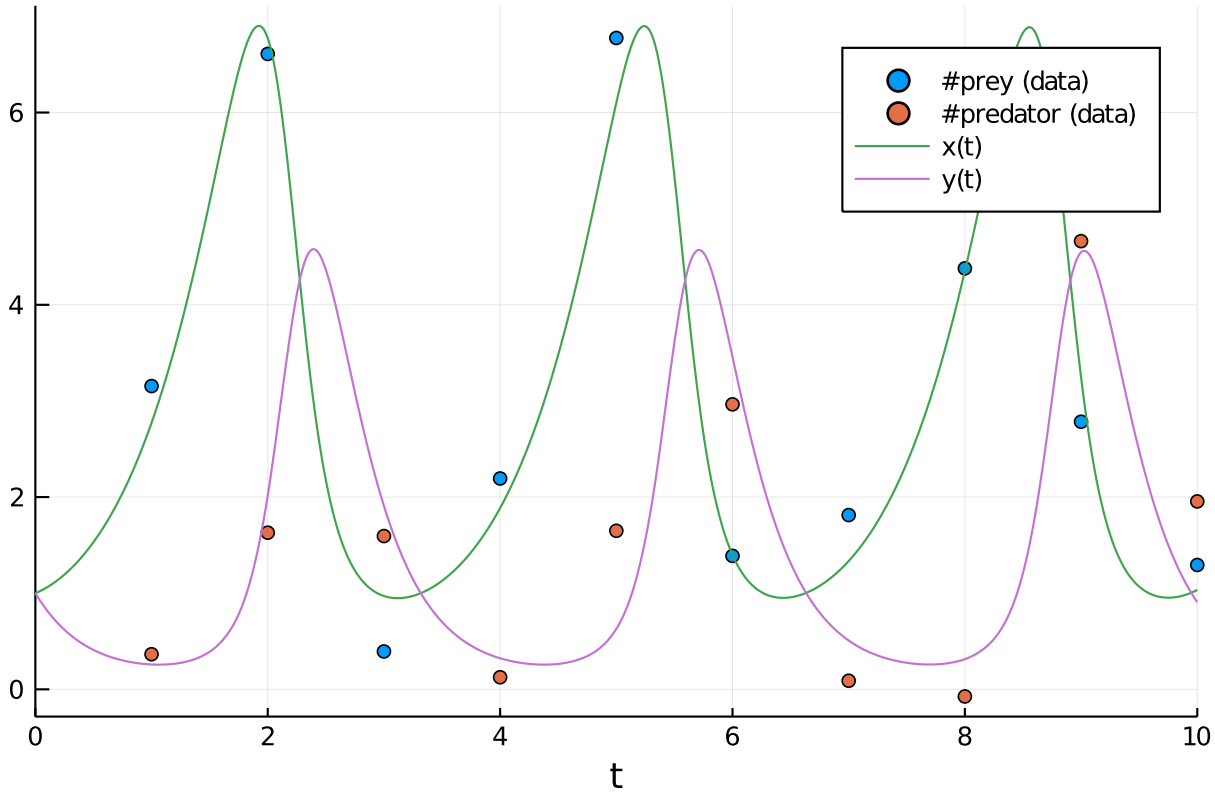
```
using DiffEqBayes, CmdStan, DynamicHMC

using Distributions, BenchmarkTools
using OrdinaryDiffEq, RecursiveArrayTools, ParameterizedFunctions
using Plots

gr(fmt=:png)

Plots.GRBackend()

f = @ode_def LotkaVolterraTest begin
    dx = a*x - b*x*y
    dy = -c*y + d*x*y
end a b c d

(::Main.##WeaveSandBox#320.LotkaVolterraTest{Main.##WeaveSandBox#320.var"##
#ParameterizedDiffEqFunction#340",Main.##WeaveSandBox#320.var"###Parameteri
zedTGradFunction#341",Main.##WeaveSandBox#320.var"###ParameterizedJacobianF
unction#342",Nothing,Nothing,ModelingToolkit.ODESystem}) (generic function
with 1 method)

u0 = [1.0,1.0]
tspan = (0.0,10.0)
p = [1.5,1.0,3.0,1,0]

5-element Array{Float64,1}:
 1.5
 1.0
 3.0
 1.0
 0.0

prob = ODEProblem(f,u0,tspan,p)
sol = solve(prob,Tsit5())

retcode: Success
Interpolation: specialized 4th order "free" interpolation
t: 34-element Array{Float64,1}:
  0.0
```

```
  0.0776084743154256
  0.23264513699277584
  0.4291185174543143
  0.6790821776882875
  0.9444045910389707
  1.2674601253261835
  1.6192913723304114
  1.9869755337814992
  2.264090367186479
  ⋮
  7.584862904164952
  7.978068388305894
  8.483164907244102
  8.719247868929038
  8.949206527971544
  9.200184813643565
  9.438028630962807
  9.711807852444823
 10.0
u: 34-element Array{Array{Float64,1},1}:
 [1.0, 1.0]
 [1.0454942346944578, 0.8576684823217127]
 [1.1758715885138267, 0.639459570317544]
 [1.4196809607170826, 0.4569962601282084]
 [1.8767193485546056, 0.32473343696185236]
 [2.5882499852859384, 0.26336255804531]
 [3.860708771268753, 0.2794458027885767]
 [5.750812903389158, 0.5220073140479389]
 [6.814978737433837, 1.917783300239219]
 [4.3929977807914105, 4.194671536988031]
 ⋮
 [2.614252575185928, 0.26416950055716665]
 [4.2410731685818694, 0.30512345857554246]
 [6.791122470590543, 1.1345265418479897]
 [6.26537352594436, 2.741690196017545]
 [3.78076791065078, 4.431164786168439]
 [1.8164212283793362, 4.0640577258289365]
 [1.1465027088171469, 2.791172606389902]
 [0.9557986534742364, 1.6235632025270912]
 [1.03375813933372, 0.9063703701433561]
```

```
t = collect(range(1,stop=10,length=10))
sig = 0.49
data = convert(Array, VectorOfArray([(sol(t[i]) + sig*randn(2)) for i in 1:length(t)]))
```

```
2×10 Array{Float64,2}:
 3.15441   6.60949  0.394083  2.19311   ...   4.37846    2.7835   1.29358
 0.365175  1.62944  1.59447   0.125652      -0.0741418  4.66213  1.95463
```

```
scatter(t, data[1,:], lab="#prey (data)")
scatter!(t, data[2,:], lab="#predator (data)")
plot!(sol)
```




```
priors =
[truncated(Normal(1.5,0.5),0.5,2.5),truncated(Normal(1.2,0.5),0,2),truncated(Normal(3.0,0.5),1,4),trun
```

```
4-element Array{Distributions.Truncated{Distributions.Normal{Float64},Distr
ibutions.Continuous,Float64},1}:
 Truncated(Distributions.Normal{Float64}(μ=1.5, σ=0.5), range=(0.5, 2.5))
 Truncated(Distributions.Normal{Float64}(μ=1.2, σ=0.5), range=(0.0, 2.0))
 Truncated(Distributions.Normal{Float64}(μ=3.0, σ=0.5), range=(1.0, 4.0))
 Truncated(Distributions.Normal{Float64}(μ=1.0, σ=0.5), range=(0.0, 2.0))
```

0.1.1 Stan.jl backend

The solution converges for tolerance values lower than 1e-3, lower tolerance leads to better accuracy in result but is accompanied by longer warmup and sampling time, truncated normal priors are used for preventing Stan from stepping into negative values.

```
@btime bayesian_result_stan =
stan_inference(prob,t,data,priors,num_samples=10_000,printsummary=false)
```

```
File /builds/JuliaGPU/DiffEqBenchmarks.jl/tmp/parameter_estimation_model.st
an will be updated.
```

```
Error: IOError: chdir : no such file or directory (ENOENT)
```

0.1.2 Turing.jl backend

```
@btime bayesian_result_turing =
turing_inference(prob,Tsit5(),t,data,priors,num_samples=10_000)

15.316 s (119308818 allocations: 8.65 GiB)
Object of type Chains, with data of type 9000×17×1 Array{Float64,3}

Iterations        = 1:9000
Thinning interval = 1
Chains            = 1
Samples per chain = 9000
internals         = acceptance_rate, hamiltonian_energy, hamiltonian_energy
_error, is_accept, log_density, lp, max_hamiltonian_energy_error, n_steps,
nom_step_size, numerical_error, step_size, tree_depth
parameters        = theta[1], theta[2], theta[3], theta[4], σ[1]

2-element Array{MCMCChains.ChainDataFrame,1}

Summary Statistics
  parameters    mean     std  naive_se    mcse        ess   r_hat

    theta[1]  1.5674  0.1100    0.0012  0.0026  1948.7940  0.9999
    theta[2]  1.0715  0.1133    0.0012  0.0022  2265.8077  0.9999
    theta[3]  2.9095  0.2905    0.0031  0.0068  1936.6315  1.0000
    theta[4]  0.9621  0.1069    0.0011  0.0024  1938.1548  0.9999
        σ[1]  0.5778  0.1105    0.0012  0.0023  2712.4786  1.0000

Quantiles
  parameters    2.5%   25.0%   50.0%   75.0%   97.5%

    theta[1]  1.3701  1.4917  1.5591  1.6366  1.8059
    theta[2]  0.8778  0.9945  1.0612  1.1353  1.3212
    theta[3]  2.3752  2.7105  2.8979  3.0940  3.5190
    theta[4]  0.7666  0.8888  0.9573  1.0283  1.1924
        σ[1]  0.4072  0.4988  0.5625  0.6396  0.8388
```

0.1.3 DynamicHMC.jl backend

```
@btime bayesian_result_dynamichmc =
dynamichmc_inference(prob,Tsit5(),t,data,priors,num_samples=10_000)

21.852 s (121053559 allocations: 12.29 GiB)
(posterior = NamedTuple{(:parameters, :σ),Tuple{Array{Float64,1},Array{Floa
t64,1}}}[(parameters = [1.5903636397954646, 1.1174893330917344, 2.939436733
7072733, 0.8895121159162759], σ = [0.8563262042516746, 0.7223587333816006])
, (parameters = [1.5647461044544877, 1.0585809637279242, 2.980632929143942,
 0.8815693075561578], σ = [0.8394707374189243, 0.6661062761933709]), (param
eters = [1.4953547679677701, 1.0799031996362258, 3.015082033693628, 1.11002
90473205114], σ = [0.640044729751732, 0.48505693482567686]), (parameters =
[1.7211124322337368, 1.0846889948799903, 2.534823715468887, 0.8490767372858
916], σ = [0.7124116647654715, 0.6468457904789071]), (parameters = [1.65215
18111649174, 1.1825202384530262, 2.7206710649924704, 0.8589156881795513], σ
 = [0.6861717279573327, 0.5168621215728801]), (parameters = [1.768959738086
4681, 1.24325866328329, 2.4573108361686025, 0.7898118432397422], σ = [0.229
97523847635876, 0.785709614188915]), (parameters = [1.6175202193488105, 1.1
376769511535698, 2.755144901456317, 0.9191577895248334], σ = [0.43114209215
978533, 1.0252069697120612]), (parameters = [1.606557950107861, 1.295734880
4233322, 2.83625967341435, 0.9114204107947661], σ = [0.47146301125294837, 0
.6921998771520491]), (parameters = [1.6294194578830234, 1.2758813774299835,
```

2.701851824861157, 0.9227152844509272], σ = [0.5054680517332507, 0.7505370
72789889]), (parameters = [1.586966884800482, 0.9319699313466139, 2.7836845
652251556, 0.9563555477391544], σ = [0.3600051017562757, 0.5777477112912783
]) ... (parameters = [1.4172616981833102, 0.8377954130816514, 3.37698251980
226, 1.1098492477935298], σ = [0.321250741469595, 0.8359223084280589]), (pa
rameters = [1.826857134986644, 1.2975346230630964, 2.3491079103175, 0.77645
73189699056], σ = [0.33402874371906344, 0.5199203476315297]), (parameters =
[1.7860208765257513, 1.2277936890155117, 2.4218046069858534, 0.77445823363
8488], σ = [0.33468613457933183, 0.5398054206651813]), (parameters = [1.579
8539750058258, 1.0216531951952432, 2.849301126372521, 0.930345928273044], σ
= [0.305775421253026, 1.09107571919259]), (parameters = [1.659121069017907
2, 1.041073740801286, 2.6783366158786475, 0.8699620432430349], σ = [0.30342
37376545852, 0.9043366426060427]), (parameters = [1.6949686370223622, 1.263
056930892571, 2.5847859043689816, 0.8406682475792041], σ = [0.3010455225486
128, 1.2199880559447287]), (parameters = [1.7251135832872955, 1.13878184623
45873, 2.5001757618398215, 0.8222594914907355], σ = [0.35405698012455594, 1
.123656837216656]), (parameters = [1.7435739814580828, 1.1654957031734927,
2.487313782875732, 0.8153155559906624], σ = [0.43221680362426673, 0.9444924
853048218]), (parameters = [1.6316734216935866, 1.1401865528008455, 2.76693
6265498163, 0.8881138942210195], σ = [0.4013501206602525, 0.466292828536084
2]), (parameters = [1.6617689297278067, 1.169789782855946, 2.66989249704652
48, 0.8551664688551557], σ = [0.34300284740612386, 0.4673968143324931])], c
hain = [[0.4639626943571816, 0.11108450212824857, 1.0782179758210262, -0.11
708215092766835, -0.15510389569957006, -0.3252334028443507], [0.44772357725
54193, 0.0569292977823531, 1.0921356702954372, -0.12605165566756737, -0.174
98366022641498, -0.40630604730838243], [0.40236348167922426, 0.076871407152
66523, 1.1036270383309366, 0.10438618373906437, -0.44621721483354715, -0.72
3489003541336], [0.5429708446909483, 0.0812933052395324, 0.9301240943761516
, -0.16360571125956277, -0.33909935380192857, -0.4356473583631287], [0.5020
78566254944, 0.1676479562083383, 1.0008785649495089, -0.15208451294476133,
-0.37662735028483485, -0.6599791294558119], [0.57039165521625, 0.2177358868
436428, 0.899067595984402, -0.23596053500304084, -1.4697836346530062, -0.24
116800241230765], [0.48089424768499706, 0.12898842119822795, 1.013470037031
6823, -0.08429747438445732, -0.8413175630258435, 0.024894513877238483], [0.
47409397119698815, 0.25907800942869325, 1.0424861677413912, -0.092751005412
15323, -0.7519146290782841, -0.36788052525555626], [0.4882237905703855, 0.2
4363721620457615, 0.9939373989747635, -0.08043455961564389, -0.682270443837
8882, -0.28696623167375246], [0.4618245747862238, -0.07045472732062474, 1.0
237754333320859, -0.044625523211769144, -1.0216370760871865, -0.54861799125
25797] ... [0.34872662833060636, -0.17698134544044292, 1.2169825651395925,
0.10422419330222762, -1.1355333346988108, -0.17921960271824988], [0.6025970
778340637, 0.2604660201435761, 0.8540356434541018, -0.2530136038153377, -1.
0965282306691913, -0.6540796567712291], [0.5799901713701748, 0.205218809922
72015, 0.8845129676345495, -0.2555915474502196, -1.094562098036662, -0.6165
46536460544], [0.4573324218854138, 0.021422094853598306, 1.0470737453928498
, -0.0721987961120016, -1.1849043639528005, 0.08716410791887495], [0.506287
9856510373, 0.040252623628642535, 0.9851959361131406, -0.13930569674153814,
-1.1926249763194168, -0.10055359564742866], [0.5276642374295819, 0.2335349
182769077, 0.949642682236039, -0.17355817058091208, -1.2004937879654973, 0.
19884106848800193], [0.5452928936995504, 0.1299591351589161, 0.916361034138
8215, -0.19569925066462251, -1.0382974179973326, 0.11658839988906168], [0.5
559370189594993, 0.15314649280604903, 0.9112033260928901, -0.20418005541353
007, -0.838827956386402, -0.05710754834027944], [0.48960612726100583, 0.131
1918918261396, 1.0177406667477336, -0.11865528495657753, -0.912921113768611
6, -0.7629414547927024], [0.5078826553161084, 0.1568240598938827, 0.9820382
083231154, -0.15645912858165612, -1.0700165303750744, -0.7605766726205615]]
, tree_statistics = DynamicHMC.TreeStatisticsNUTS[DynamicHMC.TreeStatistics
NUTS(-24.121508664000462, 5, turning at positions -22:9, 0.9999029470643556
, 31, DynamicHMC.Directions(0x40cb6729)), DynamicHMC.TreeStatisticsNUTS(-28

```
.461665490450525, 4, turning at positions -9:6, 0.907609175279148, 15, Dyna
micHMC.Directions(0xa2db27d6)), DynamicHMC.TreeStatisticsNUTS(-27.569848963
604855, 6, turning at positions -40:-47, 0.9729130667812537, 79, DynamicHMC
.Directions(0x11cccea0)), DynamicHMC.TreeStatisticsNUTS(-29.395424047627063
, 6, turning at positions -45:-76, 0.9896772567097918, 95, DynamicHMC.Direc
tions(0xd00ba293)), DynamicHMC.TreeStatisticsNUTS(-23.326862427051626, 6, t
urning at positions -38:25, 0.9917860853835383, 63, DynamicHMC.Directions(0
xe57fa459)), DynamicHMC.TreeStatisticsNUTS(-23.98680964031905, 6, turning a
t positions -6:57, 0.9899603963474748, 63, DynamicHMC.Directions(0x34870379
)), DynamicHMC.TreeStatisticsNUTS(-24.22382480782541, 6, turning at positio
ns -21:42, 0.7942022617703451, 63, DynamicHMC.Directions(0x4cc3832a)), Dyna
micHMC.TreeStatisticsNUTS(-23.239311887214992, 5, turning at positions -4:2
7, 0.9990083742535106, 31, DynamicHMC.Directions(0x1e8fc47b)), DynamicHMC.T
reeStatisticsNUTS(-21.97344237539661, 5, turning at positions -22:9, 0.9912
521877122292, 31, DynamicHMC.Directions(0xd66fdea9)), DynamicHMC.TreeStatis
ticsNUTS(-22.875632150934134, 6, turning at positions -8:55, 0.970278464059
7402, 63, DynamicHMC.Directions(0x4f0a1277))  ...  DynamicHMC.TreeStatisticsN
UTS(-24.785243530392123, 6, turning at positions -22:41, 0.8893545551601426
, 63, DynamicHMC.Directions(0xe37a0da9)), DynamicHMC.TreeStatisticsNUTS(-22
.56727593779247, 6, turning at positions -52:-83, 0.9923099213907066, 95, D
ynamicHMC.Directions(0xeb0d9d0c)), DynamicHMC.TreeStatisticsNUTS(-20.696267
37474725, 5, turning at positions -21:10, 0.9921711238143355, 31, DynamicHM
C.Directions(0xec15964a)), DynamicHMC.TreeStatisticsNUTS(-21.38382591092808
, 6, turning at positions -5:-36, 0.9987908822412389, 95, DynamicHMC.Direct
ions(0xb82b6ebb)), DynamicHMC.TreeStatisticsNUTS(-21.308346679818012, 6, tu
rning at positions -37:26, 0.811341191049669, 63, DynamicHMC.Directions(0x3
146abda)), DynamicHMC.TreeStatisticsNUTS(-24.503054649332682, 5, turning at
 positions -4:27, 0.9202434131119908, 31, DynamicHMC.Directions(0xca44995b)
), DynamicHMC.TreeStatisticsNUTS(-22.507992479834492, 5, turning at positio
ns -6:25, 1.0, 31, DynamicHMC.Directions(0x86566f19)), DynamicHMC.TreeStati
sticsNUTS(-21.970937443086406, 5, turning at positions -5:26, 0.98664197529
56275, 31, DynamicHMC.Directions(0x3b64d99a)), DynamicHMC.TreeStatisticsNUT
S(-21.5411666207429, 6, turning at positions -46:17, 0.952876539717085, 63,
 DynamicHMC.Directions(0x9a19e1d1)), DynamicHMC.TreeStatisticsNUTS(-20.3351
96039359214, 5, turning at positions -12:19, 0.9821929480815776, 31, Dynami
cHMC.Directions(0x0d25fd13))], κ = Gaussian kinetic energy (LinearAlgebra.D
iagonal), √diag(M⁻¹): [0.06867218240450856, 0.11059533727122431, 0.09727589
118935197, 0.1048795124249329, 0.29699144752505235, 0.26318012869503915], ϵ
 = 0.0650958110745061)
```

0.2 Conclusion

Lotka-Volterra Equation is a "predator-prey" model, it models population of two species in which one is the predator (wolf) and the other is the prey (rabbit). It depicts a cyclic behaviour, which is also seen in its Uncertainity Quantification Plots. This behaviour makes it easy to estimate even at very high tolerance values (1e-3).

```
using DiffEqBenchmarks
DiffEqBenchmarks.bench_footer(WEAVE_ARGS[:folder],WEAVE_ARGS[:file])
```

0.3 Appendix

These benchmarks are a part of the DiffEqBenchmarks.jl repository, found at: https://github.com/JuliaDi

To locally run this tutorial, do the following commands:

```
using DiffEqBenchmarks
DiffEqBenchmarks.weave_file("ParameterEstimation","DiffEqBayesLotkaVolterra.jmd")
```

Computer Information:

```
Julia Version 1.4.2
Commit 44fa15b150* (2020-05-23 18:35 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-8.0.1 (ORCJIT, skylake)
Environment:
  JULIA_DEPOT_PATH = /builds/JuliaGPU/DiffEqBenchmarks.jl/.julia
  JULIA_CUDA_MEMORY_LIMIT = 2147483648
  JULIA_PROJECT = @.
  JULIA_NUM_THREADS = 8
```

Package Information:

```
Status: `/builds/JuliaGPU/DiffEqBenchmarks.jl/benchmarks/ParameterEstimation/Project.to
[6e4b80f9-dd63-53aa-95a3-0cdb28fa8baf] BenchmarkTools 0.5.0
[a134a8b2-14d6-55f6-9291-3336d3ab0209] BlackBoxOptim 0.5.0
[593b3428-ca2f-500c-ae53-031589ec8ddd] CmdStan 6.0.6
[ebbdde9d-f333-5424-9be2-dbf1e9acfb5e] DiffEqBayes 2.16.0
[1130ab10-4a5a-5621-a13d-e4788d82bd4c] DiffEqParamEstim 1.15.0
[ef61062a-5684-51dc-bb67-a0fcdec5c97d] DiffEqUncertainty 1.4.1
[31c24e10-a181-5473-b8eb-7969acd0382f] Distributions 0.23.4
[bbc10e6e-7c05-544b-b16e-64fede858acb] DynamicHMC 2.1.5
[76087f3c-5699-56af-9a33-bf431cd00edd] NLopt 0.6.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.41.0
[65888b18-ceab-5e60-b2b9-181511a3b968] ParameterizedFunctions 5.3.0
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 1.5.3
[731186ca-8d62-57ce-b412-fbd966d074cd] RecursiveArrayTools 2.5.0
```