Comparison of Bayesian Functionality between Pumas and NONMEM

Vaibhav Dixit[1], Andreas Noack[2], Simon Byrne[2], Joakim Nyberg[3], Christopher Rackuackas[4,5], and Vijay Ivaturi[5]

[1]Indian Institute of Technology (B.H.U.), Varanasi, India, [2] Julia Computing; [3] Pharmetheus, Uppsala, Sweden; [4] Massachusetts Institute of Technology, Cambridge MA;
[5] Center for Translational Medicine, University of Maryland School of Pharmacy, Baltimore, MD, USA

Introduction to Probabilistic Programming in Pumas

Pumas can do Bayesian inference by leveraging the MCMC backend of Turing.jl[1], one of the many Probabilistic Programming Languages (PPLs) available in the Julia Language. Pumas uses No-U-Turn Sampler (NUTS) implemented in the AdvancedHMC.jl[2] package which is the MCMC backend for Turing.jl. The NUTS is a gradient based MCMC sampling method and Pumas computes gradients of the objective function in full precision with forward mode automatic differentiation provided by the ForwardDiff.jl package. Important features of Bayesian inference in Pumas.

- Model specification is identical to maximum likelihood methods
- *Future release will use the core Julia features for chain level parallelism and GPU acceleration*

Here we showcase the comparison of Bayes in Pumas and NONMEM using the commonly used Theophylline dataset. We also showcase the readily available parameter summarization and plotting of posteriors that are available in Pumas.

Model specification in Pumas

```
using Pumas, CSV, Random, StatsPlots

theopp = read_pumas(example_data("event_data/THEOPP"),cvs = [:WT,:SEX])

theopmodel_bayes = @model begin
  @param begin
    θ ~ Constrained(
      MvNormal(
        [1.9, 0.0781, 0.0463, 1.5, 0.4],
        Diagonal([9025, 15.25, 5.36, 5625, 400]),
      ),
      lower = [0.1, 0.008, 0.0004, 0.1, 0.0001],
      upper = [5, 0.5, 0.09, 5, 1.5],
      init = [1.9, 0.0781, 0.0463, 1.5, 0.4],
    )
    Ω ~ InverseWishart(2, fill(0.9, 1, 1) .* (2 + 1 + 1))
    σ ∈ RealDomain(lower = 0.0, init = 0.388)
  end

  @random begin
    η ~ MvNormal(Ω)
  end

  @pre begin
    Ka = (SEX == 1 ? θ[1] : θ[4]) + η[1]
    K = θ[2]
    CL = θ[3] * (WT / 70)^θ[5]
    V = CL / K
    SC = V / (WT / 70)
  end

  @covariates SEX WT

  @vars begin
    conc = Central / SC
  end

  @dynamics OneCompartmentModel

  @derived begin
    dv ~ @. Normal(conc, sqrt(σ) + eps())
  end
end

param = Pumas.init_param(theopmodel_bayes)

Random.seed!(1)
# fit using Bayes
b = Pumas.fit(
  theopmodel_bayes,
  theopp,
  param,
  Pumas.BayesMCMC(),
  nsamples = 2000,
)
# extract the chains
c = Pumas.Chains(b)

#plot the outcomes
plot(
  c,
  xtickfont = font(17),
  ytickfont = font(17),
  guidefontsize = 17,
  linecolor = :red,
  dpi = 600,
  size = (1400, 1900),
  left_margin = 50 * Plots.px,
  titlefontsize = 20,
)
```

Real time sampler metrics

```
Sampling100%|████████████████████████████████| Time: 0:00:28
  iterations:           2000
  step_size:            0.11801961508140225
  n_steps:              63
  is_accept:            true
  acceptance_rate:      0.9611716682409724
  log_density:          -230.8354522691456
  hamiltonian_energy:   241.11340798560775
  tree_depth:           6
  numerical_error:      false
  mass_matrix:          DiagEuclideanMetric([0.32628353156892637, 0.010 ...])
┌ Info: Finished 2000 sampling steps in 28.788896374 (s)
│   h = Hamiltonian(metric=DiagEuclideanMetric([0.32628353156892637, 0.010 ...]))
│   τ = NUTS{Multinomial,Generalised}(integrator=Leapfrog(ϵ=0.105), max_depth=10), Δ_max=1000.0)
│   EBFMI_est = 807.9282162569328
└   average_acceptance_rate = 0.7942364046336777
```

Results

Parameters match between Pumas and NONMEM

```
julia> b
Object of type Chains, with data of type 2000×7×1 Array{Float64,3}

Iterations        = 1:2000
Thinning interval = 1
Chains            = 1
Samples per chain = 2000
parameters        = Ω₁,₁, θ₁, θ₂, θ₃, θ₄, θ₅, σ

2-element Array{MCMCChains.ChainDataFrame,1}

Summary Statistics

│ Row │ parameters │ mean      │ std        │ naive_se    │ mcse        │ ess     │ r_hat    │
│     │ Symbol     │ Float64   │ Float64    │ Float64     │ Float64     │ Any     │ Any      │
├─────┼────────────┼───────────┼────────────┼─────────────┼─────────────┼─────────┼──────────┤
│ 1   │ Ω₁,₁       │ 1.52537   │ 0.930696   │ 0.020811    │ 0.0341814   │ 684.572 │ 1.00211  │
│ 2   │ θ₁         │ 1.80226   │ 0.553956   │ 0.0123868   │ 0.0203273   │ 605.616 │ 1.00023  │
│ 3   │ θ₂         │ 0.08481   │ 0.00668063 │ 0.000149383 │ 0.000182465 │ 1142.72 │ 1.00087  │
│ 4   │ θ₃         │ 0.0388126 │ 0.00207063 │ 4.63008e-5  │ 5.4018e-5   │ 1142.76 │ 1.00071  │
│ 5   │ θ₄         │ 1.77289   │ 0.489914   │ 0.0109548   │ 0.0238898   │ 439.793 │ 0.999592 │
│ 6   │ θ₅         │ 0.439502  │ 0.148625   │ 0.00332335  │ 0.00479611  │ 1347.37 │ 0.999654 │
│ 7   │ σ          │ 1.2664    │ 0.188114   │ 0.00420637  │ 0.00415513  │ 105.608 │ 1.00064  │

Quantiles

│ Row │ parameters │ 2.5%      │ 25.0%     │ 50.0%     │ 75.0%     │ 97.5%     │
│     │ Symbol     │ Float64   │ Float64   │ Float64   │ Float64   │ Float64   │
├─────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┤
│ 1   │ Ω₁,₁       │ 0.505834  │ 0.904754  │ 1.27075   │ 1.8841    │ 3.96763   │
│ 2   │ θ₁         │ 0.731061  │ 1.46514   │ 1.77688   │ 2.15609   │ 2.94756   │
│ 3   │ θ₂         │ 0.0721375 │ 0.0802596 │ 0.0847193 │ 0.089011  │ 0.0981951 │
│ 4   │ θ₃         │ 0.0347906 │ 0.0374436 │ 0.0388461 │ 0.0401246 │ 0.0429441 │
│ 5   │ θ₄         │ 0.901555  │ 1.43513   │ 1.75501   │ 2.08853   │ 2.83131   │
│ 6   │ θ₅         │ 0.135626  │ 0.33932   │ 0.442959  │ 0.537151  │ 0.721356  │
│ 7   │ σ          │ 0.969358  │ 1.14315   │ 1.24917   │ 1.36625   │ 1.64281   │
```

```
-----------------------------------------------------------------------

run5_reparam3.lst

Successful minimization                                         [    OK   ]
No rounding errors                                              [    OK   ]
Zero gradient information not read                              [    -    ]
Final zero gradient information not read                        [    -    ]
Hessian information not read                                    [    -    ]
No parameter near boundary                                      [    OK   ]
Covariance step                                                 [    OK   ]

Large correlations between parameter estimates found            [ WARNING ]
    CL - K                    0.932542

Total run time for model (hours:min:sec):                 0:00:37
Estimation time for subproblem, sum over $EST (seconds):   35.2

Objective function value: 175.0825

Number of observation records: 132
Number of individuals: 12

                       THETA                  OMEGA                     SIGMA
                 KA    1.723   (0.2679)  KA__  1.043  (0.3218)  Additive  1.112  (0.06811)
                  K  0.08566  (0.08079)
                 CL  0.03904  (0.05523)
                KA_    1.734   (0.2735)
Scaling CL with WT   0.4323   (0.3558)

$EST METHOD=NUTS FILE=run5_reparam3.txt NITER=2000
PRINT=100
```

A Note on using Priors in Pumas

In the `@param` block within the model specification, it is possible to assign distributions to each of the parameters instead of just specifying domains for the parameters. When estimating the model with MCMC, the distributions will serve as prior distributions for the parameters. E.g. a variance parameter could be assigned an Inverse Gamma Prior distribution with shape and scale parameters 0.1 by setting

$$\sigma^2 \sim \mathrm{InverseGamma}(0.6, 0.2)$$

Any distribution definition from the Distributions.jl [3] package or custom distributions that follow the same interface can be used as prior distributions. This allows for flexibility in experimenting with alternative prior distributions. If a parameter is not assigned a distribution but just a domain, the parameter will implicitly be assigned a flat (improper) prior.

It is assumed that all parameters are independent meaning that the definition

$$\sigma^2\_prop \sim \mathrm{InverseGamma}(0.6, 0.2)$$

$$\sigma^2\_add \sim \mathrm{InverseGamma}(0.6, 0.2)$$

will result in independent priors for σ^2_prop and σ^2_add. A joint distribution for the variance parameters can be specified by using a single matrix valued parameter for the variances

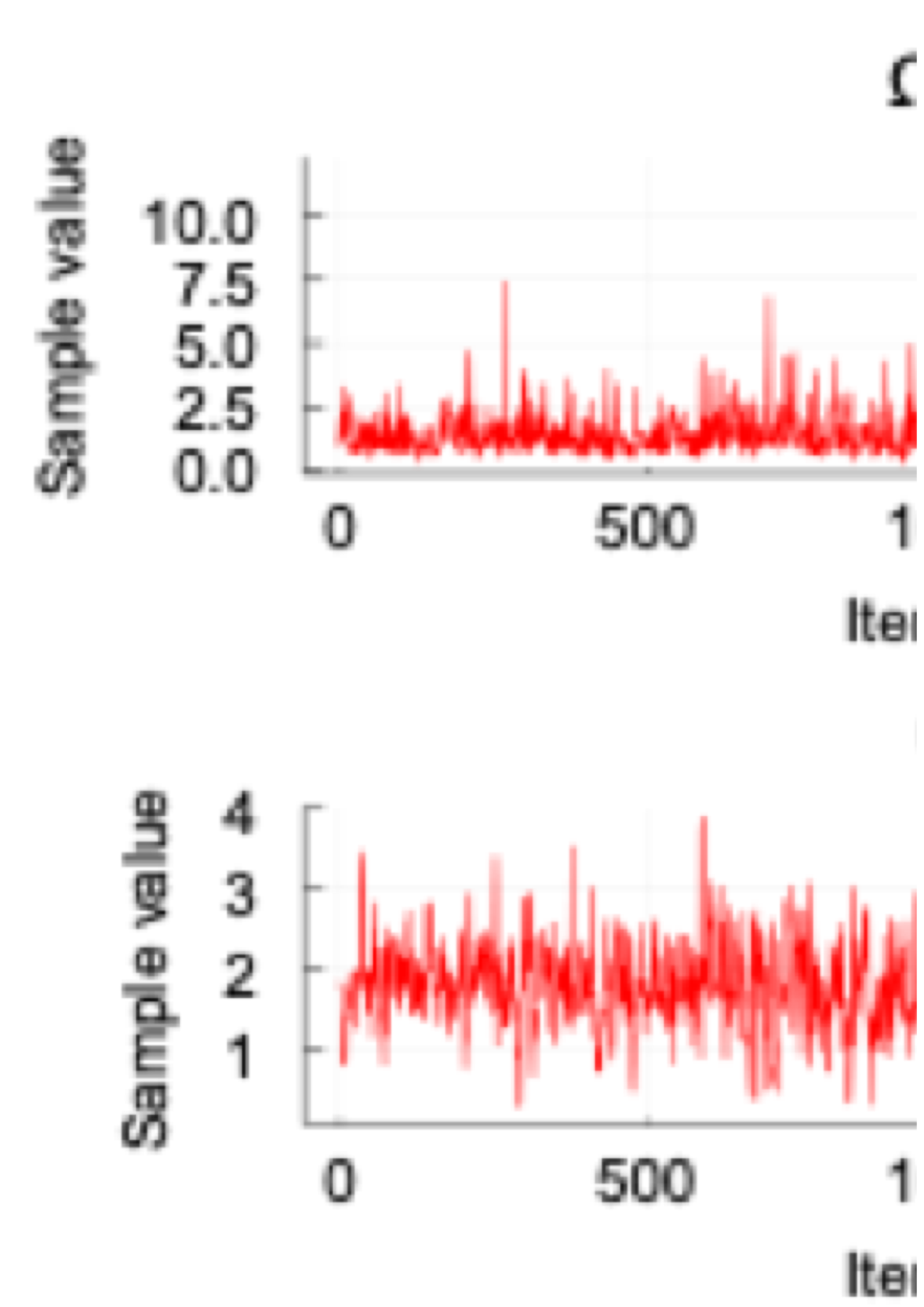
$$\Sigma \sim \mathrm{InverseWishart}(\nu, [0.125\ 0.1; 0.1\ 0.125]*(\nu + 3))$$

Typically, the fixed effects parameters are assigned a constrained multivariate normal distribution to allow for correlation between the parameters and avoid extreme parameter value. An example could look like

θ ~ Constrained(MvNormal([1.9, 0.0781,0.0463,1.5,0.4],
Diagonal([9025, 15.25, 5.36, 5625, 400])),
lower=[0.1,0.008,0.0004,0.1,0.0001],
upper=[5,0.5,0.09,5,1.5],
init=[1.9,0.0781,0.0463,1.5,0.4])

It is possible to use the maximum likelihood estimation methods FO, FOCE(I), and Laplace(I) for models with prior distributions. In that case, the prior distribution is only used for specifying the domain.

Posterior plots of parameter from a single chain



References

1. https://github.com/TuringLang/AdvancedHMC.jl
2. https://turing.ml/dev/
3. https://github.com/JuliaStats/Distributions.jl