The Journal of Open Source Software

GlobalSensitivity.jl: Performant and Parallel Global Sensitivity Analysis with Julia

Vaibhav Kumar Dixit[1¶] and **Christopher Rackauckas**[1,2,3]

1 Julia Computing **2** Massachusetts Institute of Technology **3** Pumas-AI ¶ Corresponding author

DOI: 10.21105/joss.04561

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @zhenwu0728
- @storyetfall

Submitted: 24 June 2022
Published: 17 August 2022

Summary

Global Sensitivity Analysis (GSA) methods are used to quantify the uncertainty in the output of a model with respect to the parameters. These methods allow practitioners to measure both parameters' individual contributions and the contribution of their interactions to the output uncertainty. GlobalSensitivity.jl is a Julia (Bezanson et al., 2017) package containing implementations of some of the most popular GSA methods. Currently it supports Delta Moment-Independent (Borgonovo, 2007; Plischke et al., 2013), DGSM (Sobol' & Kucherenko, 2009), EASI (Plischke, 2010, 2012), eFAST (A. Saltelli et al., 1999; Andrea Saltelli & Bolado, 1998), Morris (Campolongo et al., 2007; Morris, 1991), Fractional Factorial (Andrea Saltelli et al., 2008), RBD-FAST (Tarantola et al., 2006), Sobol (Andrea Saltelli, 2002; Andrea Saltelli et al., 2008; Sobol', 2001) and regression-based sensitivity (Ridolfi & Mooij, 2016) methods.

Statement of need

Global Sensitivity Analysis has become an essential part of modeling workflows for practitioners in various fields such as Quantitative Systems Pharmacology and Environmental Modeling (Jakeman et al., 2006; Andrea Saltelli et al., 2020; Sher et al., 2022; Zhang et al., 2015). It can be used primarily in two stages, either before parameter estimation to simplify the fitting problem by fixing unimportant parameters or for analysis of the input parameters' influence on the output. There are already some popular packages in R and Python, such as sensitivity and SALib (Herman & Usher, 2017) for global sensitivity analysis. GlobalSensitivity.jl provides Julia implementations of some of the popular GSA methods mentioned in the previous section. Thus it benefits from the performance advantage of Julia, provides a convenient unified API for different GSA methods by leveraging multiple dispatch, and has a parallelized implementation for some of the methods. This package allows users to conveniently perform GSA on arbitrary functions and get the sensitivity analysis results and provides out-of-the-box support for differential equations based models defined using the SciML interface (Rackauckas et al., 2020; Rackauckas & Nie, 2017).

Examples

The following tutorials in documentation 1 and 2 cover workflows of using GlobalSensitivity.jl on the Lotka-Volterra differential equation, popularly known as the predator-prey model. We present a showcase on how to use multiple GSA methods, analyze their results, and leverage Julia's parallelism capabilities to perform global sensitivity analysis at scale. The plots have been created using the Makie.jl package (Danisch & Krumbiegel, 2021), while many of the plots in the documentation use the Plots.jl package (Christ et al., 2022).

The ability to introduce parallelism with GlobalSensitivity.jl by using the batch keyword argument is shown in the below code snippet. In the batch interface, each column `p[:, i]` is a set of parameters, and we output a column for each set of parameters. Here we present the use of Ensemble Interface through EnsembleGPUArray to perform automatic multithreaded-parallelization of the ODE solves.

```
using GlobalSensitivity, QuasiMonteCarlo, OrdinaryDiffEq

function f(du, u, p, t)
  du[1] =  p[1] * u[1] - p[2] * u[1] * u[2] #prey
  du[2] = -p[3] * u[2] + p[4] * u[1] * u[2] #predator
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)
t = collect(range(0, stop = 10, length = 200))

f1 = function (p)
  prob_func(prob, i, repeat) = remake(prob; p = p[:, i])
  ensemble_prob = EnsembleProblem(prob, prob_func = prob_func)
  sol = solve(
      ensemble_prob,Tsit5(),
      EnsembleThreads();
      saveat = t,trajectories = size(p, 2))
  # Now sol[i] is the solution for the ith set of parameters
  out = zeros(2, size(p, 2))
  for i in 1:size(p, 2)
    out[1, i] = mean(sol[i][1, :])
    out[2, i] = maximum(sol[i][2, :])
  end
  out
end

samples = 10000
lb = [1.0, 1.0, 1.0, 1.0]
ub = [5.0, 5.0, 5.0, 5.0]
sampler = SobolSample()
A,B = QuasiMonteCarlo.generate_design_matrices(samples, lb, ub, sampler)

sobol_result = gsa(f1, Sobol(), A, B, batch=true)
```

Acknowledgments

This material is based upon work supported by the National Science Foundation under grant no. OAC-1835443, grant no. SII-2029670, grant no. ECCS-2029670, grant no. OAC-2103804, and grant no. PHY-2021825. We also gratefully acknowledge the U.S. Agency for International Development through Penn State for grant no. S002283-USAID. The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001211 and DE-AR0001222. We also gratefully acknowledge the U.S. Agency for International Development through Penn State for grant no. S002283-USAID. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. This material was supported by The Research Council of Norway and Equinor ASA through Research Council project "308817 - Digital wells for optimal production and drainage". Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number

FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Borgonovo, E. (2007). A new uncertainty importance measure. *Reliability Engineering and System Safety*, *92*(6), 771–784. https://doi.org/10.1016/j.ress.2006.04.015

Campolongo, F., Cariboni, J., & Saltelli, A. (2007). An effective screening design for sensitivity analysis of large models. *Environmental Modelling & Software*, *22*(10), 1509–1518. https://doi.org/10.1016/j.envsoft.2006.10.004

Christ, S., Schwabeneder, D., Rackauckas, C., Borregaard, M. K., & Breloff, T. (2022). *Plots.jl – A user extendable plotting API for the Julia programming language*. arXiv. https://doi.org/10.48550/ARXIV.2204.08775

Danisch, S., & Krumbiegel, J. (2021). Makie.jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software*, *6*(65), 3349. https://doi.org/10.21105/joss.03349

Herman, J., & Usher, W. (2017). SALib: An open-source Python library for sensitivity analysis. *Journal of Open Source Software*, *2*(9), 97. https://doi.org/10.21105/joss.00097

Jakeman, A. J., Letcher, R. A., & Norton, J. P. (2006). Ten iterative steps in development and evaluation of environmental models. *Environmental Modelling & Software*, *21*(5), 602–614. https://doi.org/10.1016/j.envsoft.2006.01.004

Morris, M. D. (1991). Factorial Sampling Plans for Preliminary Computational Experiments. *Technometrics*, *33*, 161–174. https://doi.org/10.2307/1269043

Plischke, E. (2010). An effective algorithm for computing global sensitivity indices (EASI). *Reliability Engineering & System Safety*, *95*(4), 354–360. https://doi.org/10.1016/j.ress.2009.11.005

Plischke, E. (2012). How to compute variance-based sensitivity indicators with your spreadsheet software. *Environmental Modelling & Software*, *35*, 188–191. https://doi.org/10.1016/j.envsoft.2012.03.004

Plischke, E., Borgonovo, E., & Smith, C. L. (2013). Global sensitivity measures from given data. *European Journal of Operational Research*, *226*(3), 536–550. https://doi.org/10.1016/j.ejor.2012.11.047

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., & Edelman, A. (2020). *Universal differential equations for scientific machine learning*. arXiv. https://doi.org/10.48550/ARXIV.2001.04385

Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl – A performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, *5*. https://doi.org/10.5334/jors.151

Ridolfi, G., & Mooij, E. (2016). Regression-based sensitivity analysis and robust design. In *Springer optimization and its applications* (Vol. 114, pp. 303–336). Springer. https://doi.org/10.1007/978-3-319-41508-6_12

Saltelli, Andrea. (2002). Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, *145*(2), 280–297. https://doi.org/10.1016/S0010-4655(02)00280-1

Saltelli, Andrea, Bammer, G., Bruno, I., Charters, E., Di Fiore, M., Didier, E., Nelson Espeland, W., Kay, J., Lo Piano, S., Mayo, D., & others. (2020). *Five ways to ensure that models serve society: A manifesto*. Nature Publishing Group. https://doi.org/10.1038/d41586-020-01812-9

Saltelli, Andrea, & Bolado, R. (1998). An alternative way to compute Fourier amplitude sensitivity test (FAST). *Computational Statistics & Data Analysis*, *26*(4), 445–460. https://doi.org/10.1016/S0167-9473(97)00043-1

Saltelli, Andrea, Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., & Tarantola, S. (2008). *Global Sensitivity Analysis: The Primer*. Wiley. ISBN: 9780470725177

Saltelli, A., Tarantola, S., & Chan, K. P.-S. (1999). A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output. *Technometrics*, *41*(1), 39–56. https://doi.org/10.1080/00401706.1999.10485594

Sher, A., Niederer, S. A., Mirams, G. R., Kirpichnikova, A., Allen, R., Pathmanathan, P., Gavaghan, D. J., Van Der Graaf, P. H., & Noble, D. (2022). A quantitative systems pharmacology perspective on the importance of parameter identifiability. *Bulletin of Mathematical Biology*, *84*(3), 1–15. https://doi.org/10.1007/s11538-021-00982-5

Sobol', I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, *55*(1-3), 271–280. https://doi.org/10.1016/S0378-4754(00)00270-6

Sobol', I. M., & Kucherenko, S. (2009). Derivative based global sensitivity measures and their link with global sensitivity indices. *Mathematics and Computers in Simulation*, *79*(10), 3009–3017. https://doi.org/10.1016/j.matcom.2009.01.023

Tarantola, S., Gatelli, D., & Mara, T. a. (2006). Random balance designs for the estimation of first order global sensitivity indices. *Reliability Engineering & System Safety*, *91*(6), 717–727. https://doi.org/10.1016/j.ress.2005.06.003

Zhang, X.-Y., Trame, M. N., Lesko, L. J., & Schmidt, S. (2015). Sobol sensitivity analysis: A tool to guide the development and evaluation of systems pharmacology models. *CPT: Pharmacometrics & Systems Pharmacology*, *4*(2), 69–79. https://doi.org/10.1002/psp4.6