The Journal of Open Source Software

Soss: Code Generation for Probabilistic Programming in Julia

Chad Scherrer[1] **and Taine Zhao**[2]

1 Metis **2** Department of Computer Science, University of Tsukuba

DOI: 10.21105/joss.02062

Software

- Review
- Repository
- Archive

Editor: Melissa Weber Mendonça

Reviewers:

- @kose-y
- @malmaud

Submitted: 13 December 2019
Published: 03 February 2020

Summary

Probabilistic programming is a rapidly growing field, but is still far from mainstream use, due at least in part to a common diconnect between performance and ease of use. Soss aims to achieve the best of both worlds, by offering a simple mathematical syntax and specialized code generation behind the scenes.

For example, here's a simple Gaussian model:

```
m = @model sigma,N begin
    mu ~ Cauchy(0,1)
    y ~ For(N) do j
            Normal(mu,sigma)
        end
    end
```

Given this, a user can do things like

- Specify the `sigma` and `N` arguments, and "forward sample" from the model (`rand`)
- Compute the log-density (`logpdf`)
- Call to external inference libraries that benefit from these or other inference primitives
- Transform the model to yield new models, for example using a known value for `mu` or computing the Markov blanket at a node
- Find the symbolic log-density, using John Verzani's `SymPy.jl` bindings to SymPy (Meurer et al., 2017a)
- Use the result of symbolic simplification to generated optimized code, often with significant performance benefits

At the time of this writing, Soss can connect (through the main library or optional add-ons) with Gen (Cusumano-Towner, Saad, Lew, & Mansinghka, 2019), SymPy (Meurer et al., 2017b), and MLJ (Blaom, Kiraly, Lienart, & Vollmer, 2019).

Acknowledgements

Thanks to the Julia language's (Bezanson, Edelman, Karpinski, & Shah, 2017) excellent support for modularity, libraries like Soss can be built without the need to re-implement existing capabilities. For extensive libaries that made early Soss development possible, the authors would like to acknowledge

- Tamas Papp for `DynamicHMC.jl` and associated libraries
- Ed Scheinerman for the `SimpleWorld.jl` ecosystem, which we use to track and reason about variable dependencies within a model
- Mathieu Besançon for ongoing work on `Distributions.jl` (Lin et al., 2019), and for his patient tolerance for PPL-related nitpicks of this library

We would also like to thank Seth Axen for helpful discussions and recent contributions including connection to ArviZ (Kumar, Carroll, Hartikainen, & Martin, 2019), and the Turing (Ge, Xu, & Ghahramani, 2018), Gen (Cusumano-Towner et al., 2019), and MLJ (Blaom et al., 2019) teams for helpful discussions, and for making their libraries modular and open-source.

References

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

Blaom, A., Kiraly, F., Lienart, T., & Vollmer, S. (2019). *Alan-turing-institute/mlj.jl: V0.5.3*. Zenodo. doi:10.5281/zenodo.3541506

Cusumano-Towner, M. F., Saad, F. A., Lew, A. K., & Mansinghka, V. K. (2019). Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th acm sigplan conference on programming language design and implementation*, PLDI 2019 (pp. 221–236). New York, NY, USA: ACM. doi:10.1145/3314221.3314642

Ge, H., Xu, K., & Ghahramani, Z. (2018). Turing: A language for flexible probabilistic inference. In *International conference on artificial intelligence and statistics, AISTATS 2018, 9-11 april 2018, playa blanca, lanzarote, canary islands, spain* (pp. 1682–1690). Retrieved from http://proceedings.mlr.press/v84/ge18b.html

Kumar, R., Carroll, C., Hartikainen, A., & Martin, O. A. (2019). ArviZ a unified library for exploratory analysis of Bayesian models in Python. *The Journal of Open Source Software*. doi:10.21105/joss.01143

Lin, D., White, J. M., Byrne, S., Bates, D., Noack, A., Pearson, J., Arslan, A., et al. (2019). JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions. doi:10.5281/zenodo.2647458

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., et al. (2017a). SymPy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. doi:10.7717/peerj-cs.103

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., et al. (2017b). SymPy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. doi:10.7717/peerj-cs.103