Fugue: Monadic Probabilistic Programming for Rust

A probabilistic programming library for Rust built around a monad: models compose in direct style, and pluggable interpreters decide what running one means.

august 19, 2025 · Projects

---

a probabilistic programming library for rust, built around a monad.

a model in fugue is a `Model<T>` value that you compose in direct style with `bind` and `map`. what that composition _means_ is decided later, by the interpreter you hand it to: sample from the prior, replay a trace, score it, or run mcmc, hmc, smc, variational inference or abc over it. separating the model from the inference is the main idea. in most probabilistic languages the algorithm is entangled with how you declare the model, so trying a different sampler means rewriting the thing you were trying to hold constant.

the distributions are typed with their natural return types. bernoulli gives you a `bool`, poisson and binomial give you `u64`, categorical gives you a `usize`, rather than everything collapsing to `f64` and being cast back at the point of use, which is where the off-by-one bugs come from. seventeen distributions, all with validated parameters, and log-space arithmetic throughout so that multiplying a few thousand small probabilities does not quietly become zero.

there are macros for do-notation (`prob!`) and vectorization (`plate!`), because monadic code in rust is unreadable without them.

it is 0.2.x: pre-1.0, no semver guarantees yet, one primary maintainer. extensively tested, including property-based tests and statistical regressions against closed-form posteriors, but that is not the same as production-ready. pin an exact version and read the changelog before upgrading.

view on github · learn it interactively at fugue.run

https://alexnodeland.com/timeline/250819_fugue/