Extending Distributions with Expectations.jl

Arnav Sood [1]

[1]Vancouver School of Economics, University of British Columbia

Advanced Use

Package Composition

Because of Julia's multiple dispatch, we can build distribution and expectation objects using any numeric types that implement basic arithmetic functions:

```
julia> mu, sigma = 0.5 ± 0.01, 1.0 ± 0.01
(0.5 ± 0.01, 1.0 ± 0.01)

julia> d = Normal(mu, sigma);

julia> E = expectation(d);

julia> E(x -> (x - mu)^2)
1.0 ± 0.02

julia> sigma^2
1.0 ± 0.02
```

Autodifferentiation

Our expectation operators are compatible with the `ForwardDiff.jl` autodifferentiation library. This means that they can be embedded in ML training loops.

```
julia> f = x -> E(y -> y^2 * x)
#59 (generic function with 1 method)

julia> ForwardDiff.derivative(f, 2)
0.9999999999999984
```

Algorithms

If a distribution is named in the following table, there is an optimized method for it available in the package.

Distribution	Quadrature Algorithm	Parameters and Defaults
Normal	Gauss-Hermite	$n = 30$
LogNormal	Gauss-Hermite	$n = 30$
Beta	Gauss-Jacobi	$n = 32$
ChiSq	Gauss-Laguerre	$n = 32$
Uniform	Gauss-Legendre	$n = 30$
Exponential	Gauss-Laguere	$n = 32$
Gamma	Gauss-Laguerre	$n = 32$
Continuous Univariate (compact)	Gauss-Legendre	$n = 500$
Continuous Univariate	QuantileRange	$q = \{0.001, 0.002, ..., 0.999\}$
Univariate	Trapezoidal	N/A

Table 1. Specialized methods and generic fallbacks available in `Expectations.jl`. The n parameter is the number of Gaussian quadrature nodes. Defaults were chosen based on either existing software or direct experimentation.

Abstract

Many statistical problems require taking an expectation of some function $f(x)$, where x is drawn from some known distribution. For example, a well-known economic model of job search [2] involves calculating an expected value $\mathbb{E}[f(w)]$, where w is a random wage offer, and $f(\cdot)$ is the lifetime value of that offer.

Julia's "Distributions.jl" [1] package provides many random variable objects, but taking expectations is still a laborious process. Traditional approaches include Monte Carlo simulation (slow, and potentially inaccurate), or custom numerical integration (inaccessible for non-statisticians.) And both of these approaches fail to capitalize on one of Julia's key features: the similarity between math and Julia code.

The "Expectations.jl" package addresses these weaknesses.

1. We use **tailored Gaussian quadrature** instead of naive Monte Carlo, which makes our expectation operators smaller (storing two vectors of size $n \approx 32$ instead of massive samples), faster, and deterministic. **Accuracy is not compromised**; in testing, two pairs of 32-node vectors are sufficient to compute expectations to machine precision.
2. Expectation operators are **reusable objects**, which makes expectation-taking a one-time cost.
3. Our operators are **notationally faithful**, which lets us **explot Julia's syntax clarity** and **provide faithful mathematical behavior**. Expectations.jl operators can be used as valid linear operators (acting on vectors, supporting scalar multiplication, implementing arithmetic mixture, etc.)

Performance and Accuracy Benchmarks

Distribution	Expectations.jl	1000-Sample Monte Carlo	10^6-Sample Monte Carlo
Normal	(-5, -18)	(-6, -2)	(-3, -5)
LogNormal	(-5, -15)	(-5, -2)	(-2, -3)
Beta	(-5, -∞)	(-4, -2)	(-1, -4)
Gamma	(-5, -15)	(-6, -2)	(-3, -4)

Table 2. Timed with BenchmarkTools.jl. All Expectation operators are constructed with default arguments. Distributions use standard parameters. Operator construction time is included in benchmark. Results are in (time exponent, error exponent)

Related Software, Dependencies, and Acknowledgements

There are analogous packages in other programming languages. For example, in Python the `sympy.stats` module provides for both symbolic and numerical representations of random variable expectations. And `scipy.stats` also has support for numerical integration with respect to pdfs.

Within Julia, we benefit from two major packages:

- **FastGuassQuadrature.jl** [3] is what we call to compute Gaussian nodes and weights. The package provides weights to 16 digit accuracy in $O(n)$ time, using a variety of methods. Examples include:
  - An analytic expresson (e.g., for Gauss-Legendre with $n \leq 5$),
  - Using Newton's method to solve an indexed polynomial $P_n(x) = 0$.
  - Using asymptotic expansions (e.g., Gauss-Legendre with $n \geq 60$).
- **DistQuads.jl**, which was created by **Patrick K. Mogensen** of **Julia Computing**, and which was the forerunner to this package.

We also benefit from the generosity of the **QuantEcon** and its supporters (notably NumFOCUS and the Alfred P. Sloan Foundation).

Numerical Integration for Expectations

For a (continuous) univariate random variable X, following a cumulative distribution function $G(\cdot)$, the expectation is defined as:

$$\mathbb{E}[f(X)] = \int_S f(x) dG(x) \tag{1}$$

The integral is what makes this quantity challenging to compute. The defining feature of an expectation estimator is the flavor of numerical integration it employs.

Monte Carlo

A Monte Carlo method might approximate it by drawing a large sample of points, and then simply taking the average:

$$\mathbb{E}_{MC}[f(X)] \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{2}$$

Gaussian Quadrature

We compute the integral via so-called *Gaussian quadrature*, which attempts to intelligently select a relatively small set of points x_i and weights w_i to approximate the integral as:

$$\mathbb{E}_{GC}[f(X)] \approx f(\mathbf{x}) \cdot \mathbf{w} \tag{3}$$

There are numerous variations, tailored to specific domains (compact, infinite, semi-infinite, etc.)

For example, the **Gauss-Chebyshev Quadrature** rule (for $[-1, 1]$) is often written:

$$x_i = cos\left(\frac{2i-1}{2n}\pi\right), \qquad w_i = \frac{\pi}{i}$$

References

[1] Dahua Lin, John Myles White, Simon Byrne, Douglas Bates, Andreas Noack, John Pearson, Alex Arslan, Kevin Squire, David Anthoff, Theodore Papamarkou, Mathieu Besançon, Jan Drugowitsch, Moritz Schauer, and other contributors. JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions, 2019.

[2] J. J. McCall. Economics of information and job search. *The Quarterly Journal of Economics*, 84(1):113–126, 1970.

[3] Alex Townsend, Thomas Trogdon, and Sheehan Olver. Fast computation of gauss quadrature nodes and weights on the whole real line, 2014.