Hakaru Language: Standard Library Implementation and Language Validation Testing

Justin Staples, Mahmoud Khattab, Nevin Mahilal and Aryan Sohrabi, supervised by Dr. Christopher Anand & Dr. Jacques Carette

{staplejw, khattm, mahilank, sohraa3, anandc, carette}@mcmaster.ca

Department of Computing and Software, McMaster University

Introduction

Mathematical Background

Consider a continuous random variable, X, that is distributed according to a normal distribution. This is denoted by:

$$\boxed{X \sim \mathit{Normal}(\mu, \sigma^2)}$$

Below is a graph of the probability density function (PDF) for the standard normal distribution. It describes the probability that a sample of the random variable X will be equal to $x \in \mathbb{R}$.

The treatment of discrete random variables is much the same, except we refer to their probability mass function (PMF)

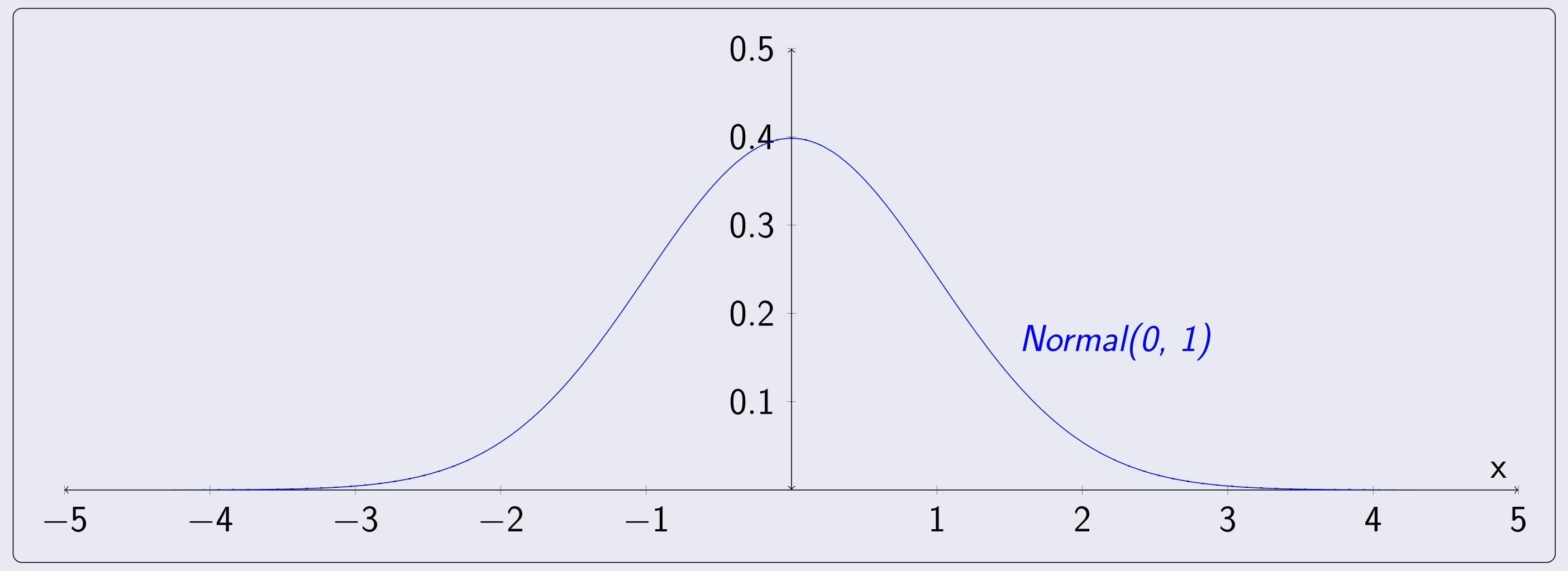


Figure 1: PDF of the normal distribution [1], with $\mu = 0$ and $\sigma = 1$

Language Guide

Hakaru is an experimental **probabilistic programming language**, which aims to simplify the implementation of statistical distributions.

- Niche application means that Hakaru is a small language with limited features.
- Running a hakaru program generates a stream of random numbers distributed according the statistical distribution modeled.
- Hakaru code can be compiled to C and Haskell so that models can be exported for use in larger applications.

`hk-maple` is a provided inference algorithm that uses Maple to perform algebraic transformations on hakaru programs. In its default mode (Simplify), it returns an equivalent hakaru program with greater sampling efficiency. In applications requiring billions of samples (e.g. machine learning) this has the potential to save significant processing time!

Key Concepts

- ⋆ *Hakaru program $\iff$ implementation of statistical distribution $\iff$ probabalistic model $\iff$ model.*
- ⋆ *Values pulled from a distribution are called measures. Measures $\iff$ samples.*
- ⋆ '$\sim$' vs '$<\sim$':
  - $X \sim \mathit{Normal}(0, 1)$ means that the random variable, X, is distributed according to the standard normal distribution (statistics literature).
  - `x <~ normal(0, 1)` means to pull a random variable from the standard normal distribution and bind it to the variable, `x` (Hakaru code).
- ⋆ *PDF: Probability Density Function (continuous random variables).*
- ⋆ *PMF: Probability Mass Function (discrete random variables).*
- ⋆ *UDR chart: Univariate Distribution Relationship chart (describes relationships and transformations between different distributions).*

Motivation

Objectives

- Increase language accessibility: add new language features such as primivite mathematical functions (e.g. log, choose), new distributions, error handling, etc.
- Test language validity: use relationships between distributions to test the validity of program transformations like `hk-maple`.

Endeavours

- Standard library development: the UDR is used to branch out and implement new distributions.
- Syntax-highlighting-for-hakaru package for Sublime Text: this package makes developing Hakaru code much more convenient and streamlined (Mahmoud).
- Test case writing: we focus mainly on comprehensively testing only a select few distributions (e.g. chi-square).

Standard Library Development

Accessibility means the standard library should implement commonly used probabilistic models. We have implemented the distributions described in the UDR chart, seen in part below. The following principles have guided the development of the standard library:

- Whenever possible, implement distributions as transformations on pre-existing models.
- Reference the UDR chart for possible ways to implement each distribution.
- In the case of multiple possible implementations, take the shortest possible path from a primitive distribution on the UDR.

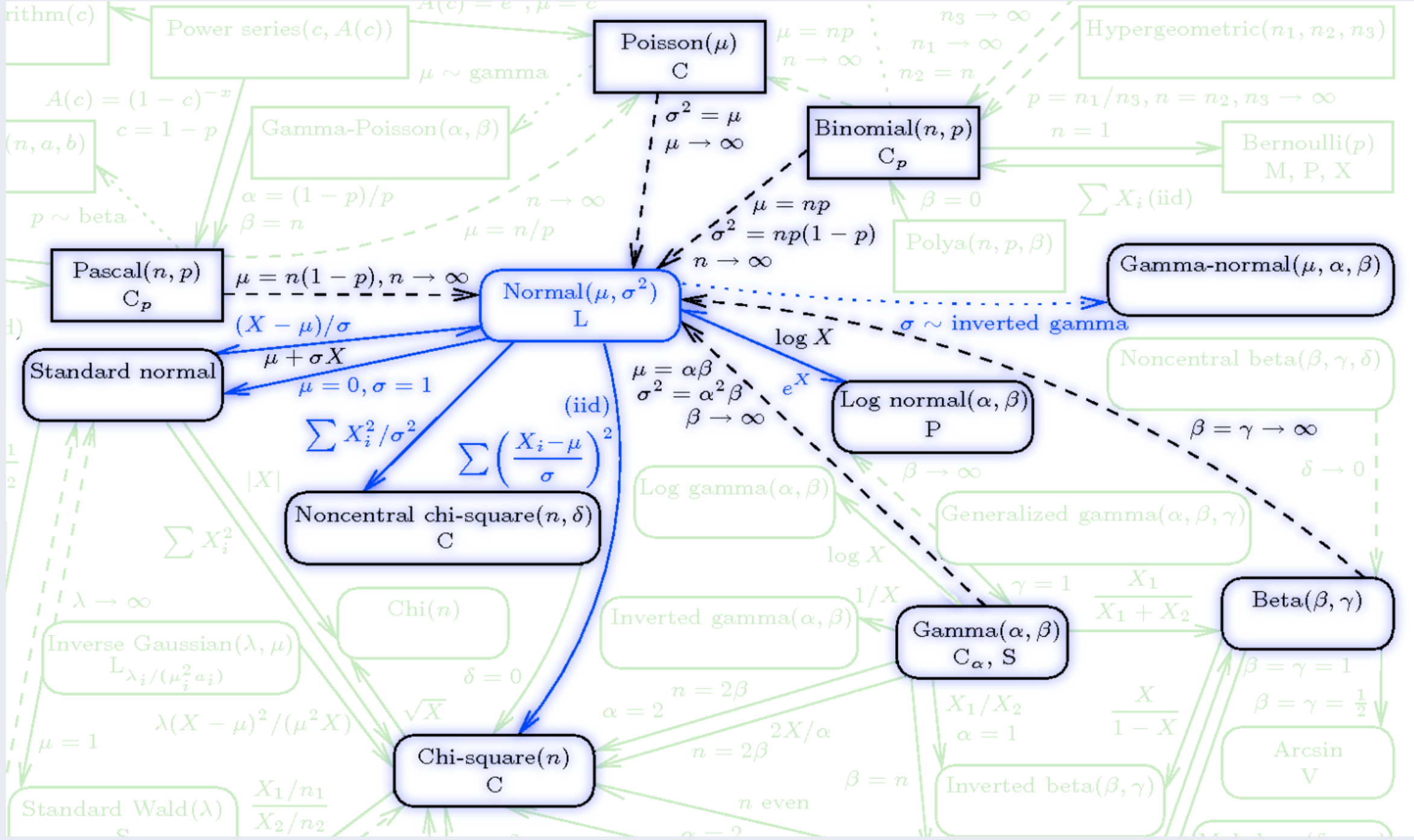


Figure 2: A snapshot of the UDR [1] shows how the normal distribution can be transformed into a multitude of other distributions.

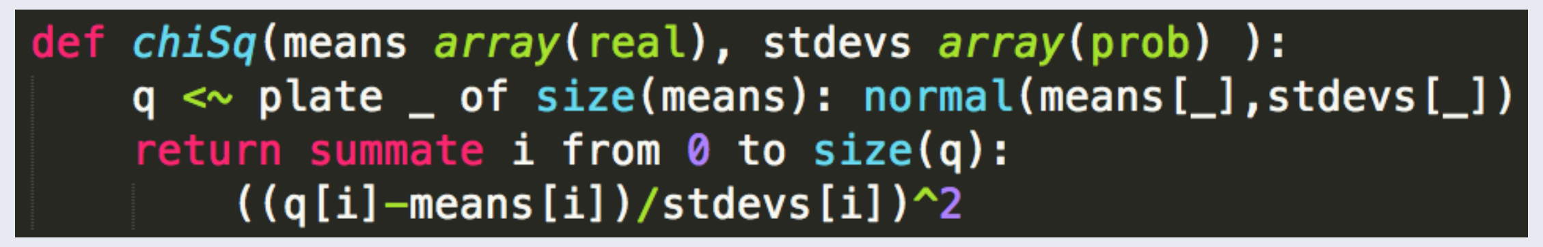
Hakaru does not allow us to do algebraic operations on the PDF of a given distribution. The only operator we can use on a model is bind ($<\sim$), to pull a sample and bind it to a variable (AKA a random variable). We are able to transform samples however we like. Therefore, we are interested in implementing transformations of the form:

$$\boxed{R(p, q) \Rightarrow X \sim A(p) \Rightarrow f(X) \sim B(q)}$$

p and q are sets of parameters on the distributions A and B. $R(p, q)$ is a set of relationships between p and q that must be satisfied. f is a transformation on the random variable X. We can extend this definition to include transformations defined in terms of an aggregation of multiple independent samples. For example, the standard chi-square distribution is defined as the sum of the squares of n normal random variables (see Figure 3).

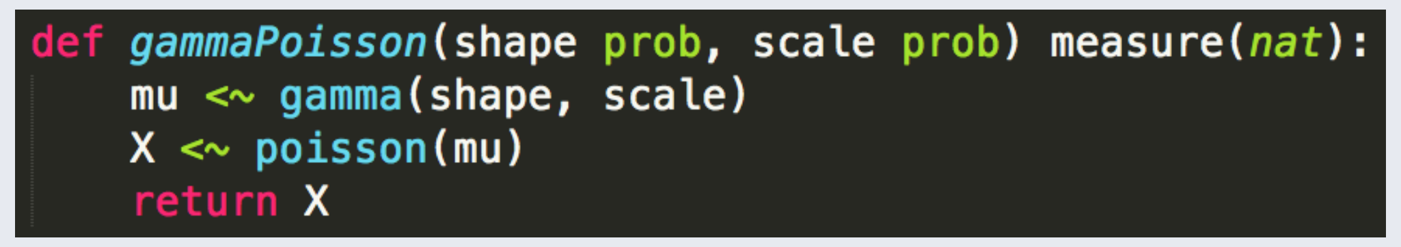
Hakaru also lends itself well to Bayesian transformations where we want to pull a sample from a distribution which is parameterized with a value sampled from another distribution. These transformations take the following form. Here, we are saying that if X is parameterized by p, then the distribution C, which is parameterized by p and q, is equal to another distribution, B, which is parameterized by q and X. The gamma-poisson distribution can be described by such a transformation (see Figure 4).

$$\boxed{X \sim A(p) \Rightarrow Y \sim B(q, X) = C(p, q)}$$



```
def chiSq(means array(real), stdevs array(prob) ):
    q <~ plate _ of size(means): normal(means[_],stdevs[_])
    return summate i from 0 to size(q):
        ((q[i]-means[i])/stdevs[i])^2
```

Figure 3: Our implementation of the chi-square distribution.

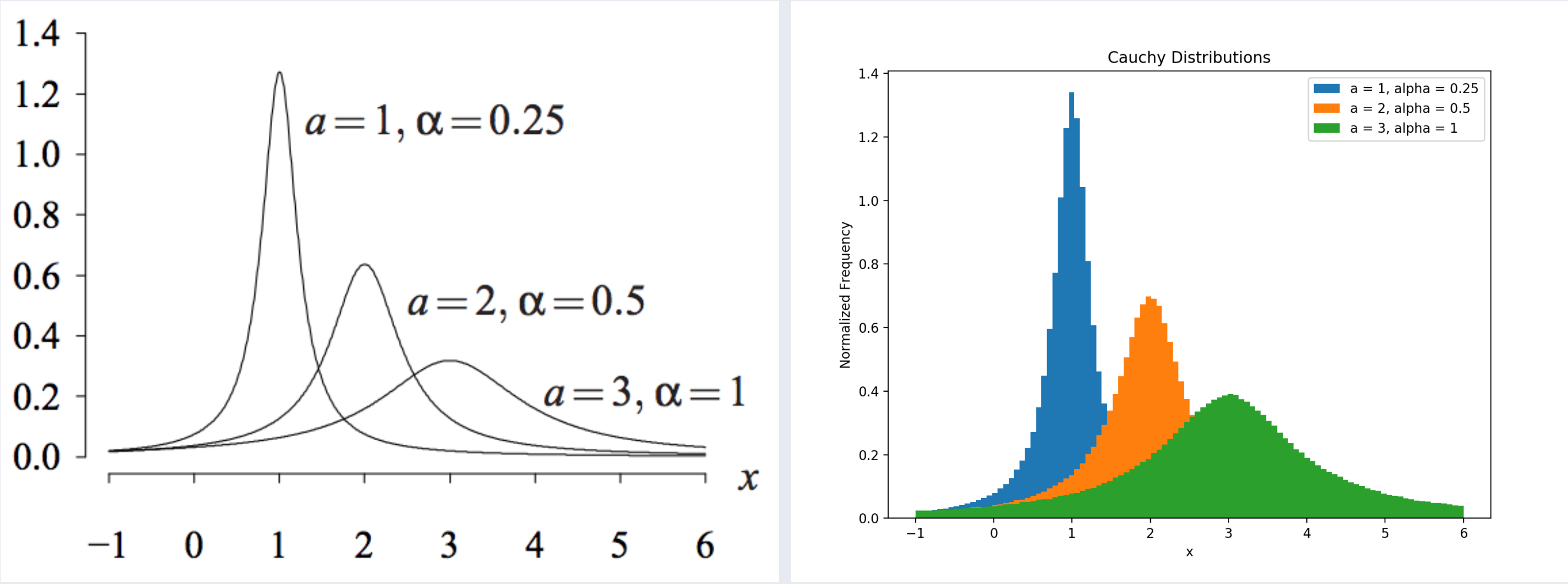


```
def gammaPoisson(shape prob, scale prob) measure(nat):
    mu <~ gamma(shape, scale)
    X <~ poisson(mu)
    return X
```

Figure 4: Our implementation of the gamma-poisson distribution.

Some distributions on the UDR are unreachable from Hakaru's primitive distributions using transformations like the 2 described above. In these cases, we have to implement the model in terms of the PMF for a discrete distribution or in terms of the PDF for a continuous distribution.

A Python script has been developed that samples a large amount of data and plots the results in a histogram. This is a useful tool for gaining confidence in the correctness of our implementations. The plots shown below were created by sampling a Hakaru program 1,000,000 times.



(a) (b)

Figure 5: A few plots of the Cauchy distribution are shown [1], along with a few histograms of data that have been sampled from a Hakaru program.

Testing Relationships Between Distributions

When developing the standard library, the UDR chart often implied multiple possible implementations for a given distribution. We expect alternative implementations for the same distribution to result in equivalent probabilistic models. This line of thought is the basis for the kinds of test cases implemented. More generally, we have the following hypothesis.

Assume we know a proven relationship between 2 statistical distributions, A and B, which allows us to transform A and B into distributions that are equivalent to each other. Further assume this transformation takes on one of the forms discussed in the Standard Library Development section. We hypothesize that by applying the appropriate transformations to implementations of A and B, we can create two Hakaru programs whose `hk-maple` outputs will be equivalent to each other. Test cases that prove our hypothesis true indicate the validity of the Hakaru language implementation. Test cases that prove our hypothesis false indicate an underlying bug in the language definition which is to be passed back to the language developers.

Conclusions & Future Work

This project was aimed at increasing language accessibility and testing the validity of Hakaru. A lot of ground was covered in both of these aspects in this project, but compared to other popular languages, Hakaru is still under development and has a lot of room for improvement. Here are just a few examples of where major development can still be done.

- Languages features: Hakaru can be expanded further with new features like import statements and error/exception handling.
- Standard library development: there are still many types of distributions left to be implemented, such as multivariate distributions.
- Testing: There are a plethora of relationships between distributions that have yet to be investigated.

References

[1] L. Leemis, "Univariate Distribution Relationship Chart", Math.wm.edu, 2018. [Online]. Available: http://www.math.wm.edu/ leemis/chart/UDR/UDR.html.

[2] P. Narayanan, J. Carette, W. Romano, C. Shan and R. Zinkov, Probabilistic Inference by Program Transformation in Hakaru (System Description), Functional and Logic Programming, pp. 62-79, 2016.