Probabilistic Programming

For all those times you've ever wondered what would happen if programming weren't deterministic.

What is a Probabilistic Programming Language

- With the rise in popularity of AI, probabilistic Programming languages attempt to fully encapsulate probabilistic modeling
  - This essentially would replace the verbosity of writing programs meant for statistical modeling/AI
- Eliminates the need for external packages
  - Large, verbose packages like scikit-learn, scipy, numpy
- Usually operate a series of procedures on statistical distributions

Applications Include:

- Kalman filtering
- Bayesian inference in machine learning, robotics, and AI
- Natural language processing
- Criminal justice
  - Prototype FBI databases can construct 3D face models from 2D images
- Climate models

The Multi-Paradigm Paradigm

- Objected-Oriented (Figaro)

```
class Firm {
  val efficient = Flip(0.3)
  val bid = If(efficient, Uniform(0.0,10.0), Uniform(5.0,20.0))
}
val firms = Array.fromFunction(i => new Firm)(20)
def clause(firm: Firm) = (Constant(1.0), Constant(firm))
val winner = Dist(firms map clause)
val winningBid = Chain(winner, (f: Firm) => f.bid)
winningBid.constrain((bid: Double) => 20.0 - bid)
```

- Functional (Anglican, Church, Venture)

```
(defquery example
  (let [bet (sample (beta 5 3))]
    (observe (flip bet) true)
    (> bet 0.7)))
```

Anglican

It’ll make you feel like beheading your wife.

Programming Based on Distributions

- Four main functions
  - doquery, defquery
  - sample, observe
- Programs are called queries, which are created with defquery and run with doquery
- Queries act on distributions to infer results using inference methods like Hastings-Metropolis, etc.
- Types of distributions include: normal, binomial, gamma, etc...

Sample and Observe

Sample

- Used to draw random samples from a distribution
- Reference current data states

Observe

- Used to condition on data
- Draw specific values from a distribution

Example

Common Anglican example predicts whether two people will successfully meet up. They can choose between the pub and starbucks, but they can’t communicate ahead of time and both slightly prefer the pub.

```
let pub-preference = 0.6

(defquery location [pub-preference]
  (sample (flip pub-preference)) :pub :starbucks)
  (observe (flip pub-preference) (= value :pub))
```

Puritan

For when Anglican is just too Catholic and complicated.

Implementation

Builds off of the full math and logic implementation from HW6

Functions

- (**defquery** name bindings condition) - Define a Puritan program
- (**sample** distribution) - Return a random value from the distribution
- (**observe** distribution x-value) - Return the x-value of the distribution

Distributions

- (**flip** bias) - Return a random 1 or 0 with an optional bias
- (**normal** mean std) - Return a normal distribution with the given mean and standard deviation

Demonstration

Functionality

Coin flip probabilities

Bonus!

- Nondeterministic for-loops

Questions or Suggestions?

```
defquery our-answers-will-help-you-understand
[(yes -> (sample (flip .01) true)))] (> yes 1);
```