Composable, Modular Probabilistic Models

Minh Nguyen   min.nguyen@bristol.ac.uk   Supervised by: Dr Meng Wang, Dr Roly Perera   (ACM: 0619104. Graduate)

Concept

This presents a shallow-embedded DSL in Haskell for syntactically constructing probabilistic models as first-class citizens; these can be interpreted for both simulation and inference, and can be composed and combined with higher-order functions. The implementation mainly relies on algebraic effects, extensible data, a distribution effect Dist, and a reader effect for affine environments AffReader.

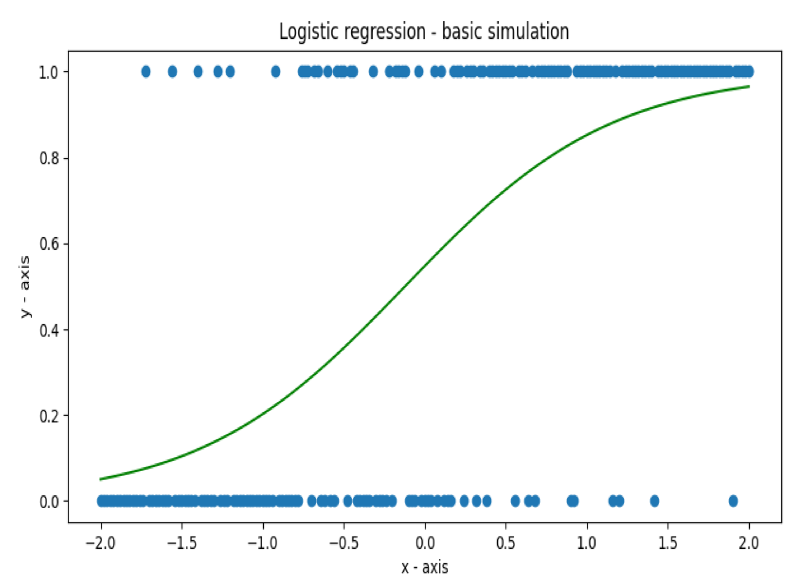
```
newtype Model env ts a = Model { runModel :: (Member Dist ts, Member (AffReader env) ts) ⇒ Freer ts a }
```

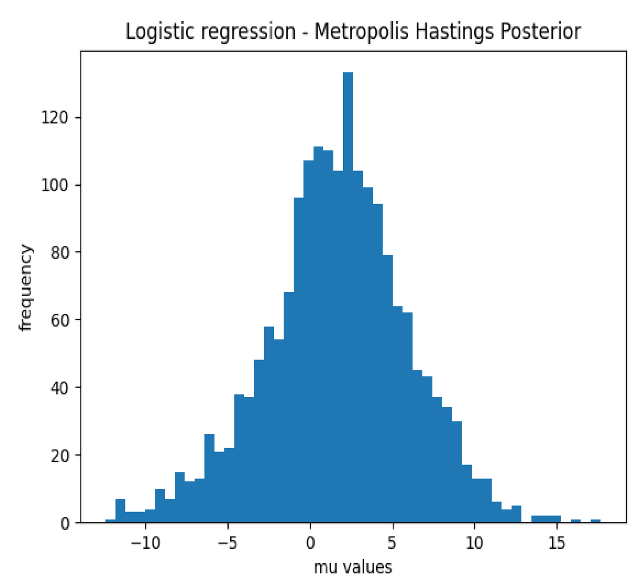
Simulation and inference is implemented using effect handlers to perform composable program transformations; this embeds a model into the context of a specific algorithm.

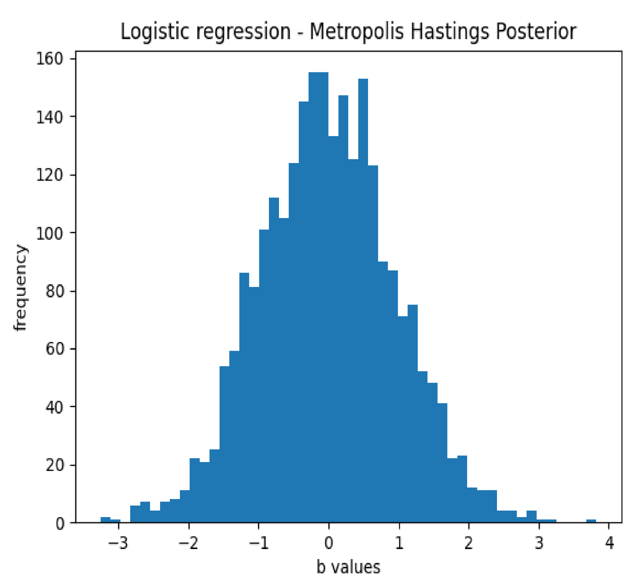
Logistic Regression

A logistic regression model can be expressed in our language as given below. The LHS figure demonstrates simulating from this model, and the two RHS figures approximate the posterior distributions of the model's parameters as a result of Metropolis-Hastings inference.

```
logisticRegression :: (Observable env "label" Bool, Observables env ["m", "b"]) ⇒
                      Double → Model env ts (Double, Bool)
logisticRegression x = do
  m     ← normal' 0 5 #m
  b     ← normal' 0 1 #b
  sigma ← gamma 1 1
  y     ← normal (m ∗ x + b) sigma
  l     ← bernoulli ' (sigmoid y) #label
  return (x, l)
```



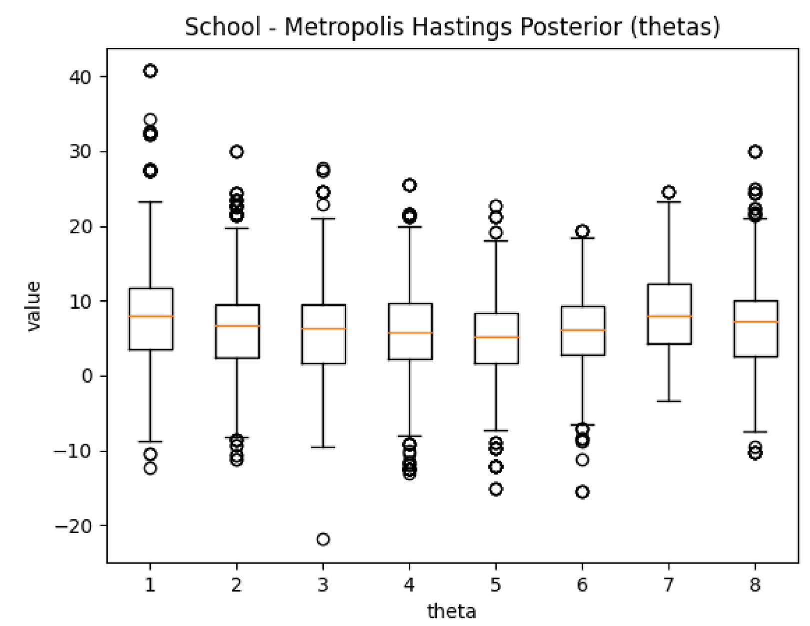




Hierchical Random Effects (School) Model

This model considers the effectiveness of SAT coaching programs conducted in parallel at eight schools, using Metropolis-Hastings to yield the posterior distribution seen below.

```
schoolModel :: (Observables env ["mu", "y"] Double
               , Observable env "θs" [Double])
            ⇒ Int → [Double] → Model s es [Double]
schoolModel n_schools σs = do
  μ   ← normal' 0 10 #mu
  τ   ← halfNormal 10
  ηs  ← replicateM n_schools (normal 0 1)
  θs  ← deterministic ' (map ((μ +) . (τ ∗)) ηs) #θs
  ys  ← mapM (\(θ, σ) −> normal' θ σ #y) (zip θs σs)
  return θs
```



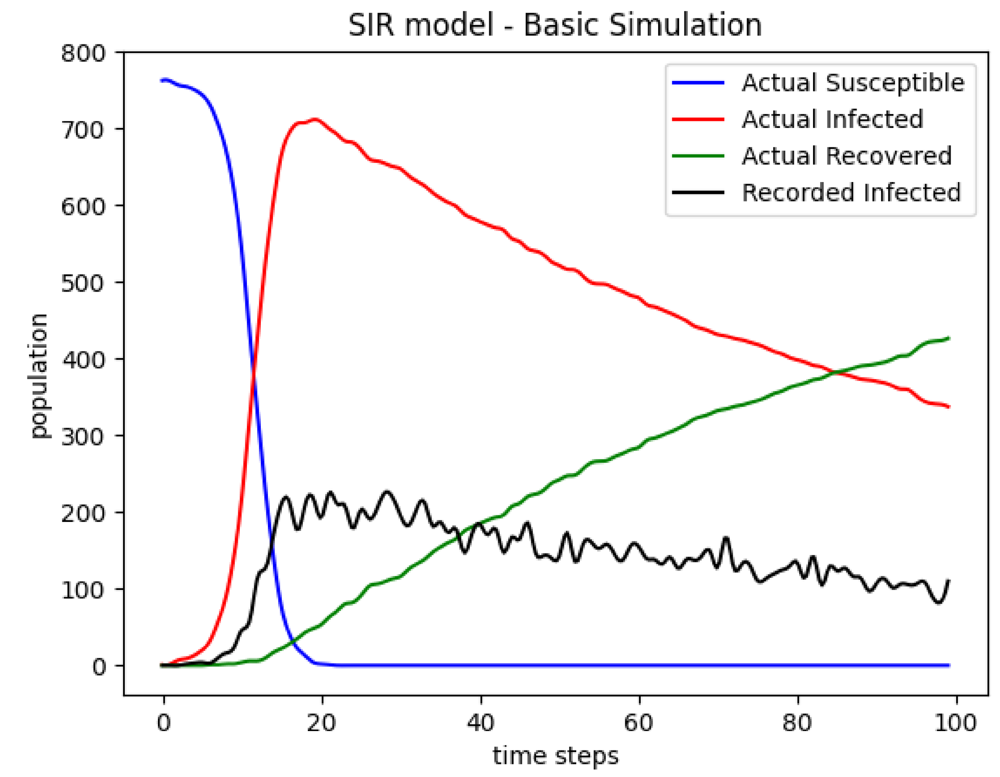
Hidden Markov (Pandemic) Model

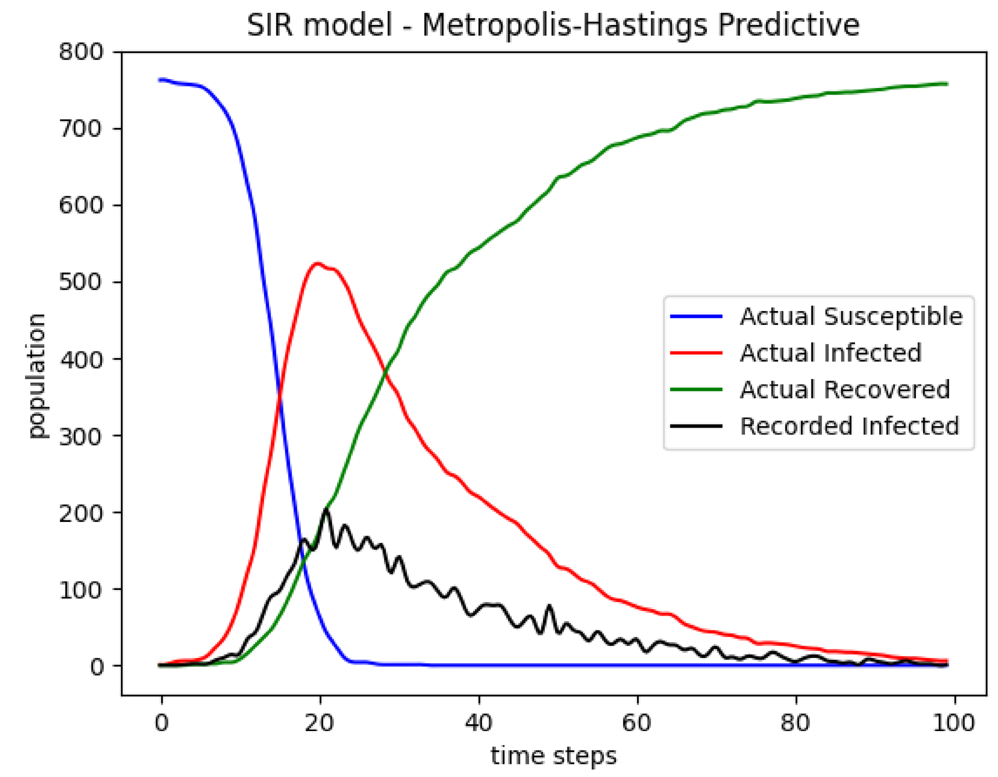
This uses a hidden markov model to characterize the change in the number of susceptible, infected, and recovered people, during a pandemic. The LHS figure simply simulates from the model, whereas the RHS figure shows the predictive distribution from trying to learn the model's parameters.

```
hmmSIR :: Observable env "inf" Int ⇒ FixedParams → Params → LatentState → Model env ts LatentState
hmmSIR fixedParams params latentState = do
  latentState '  ← transitionSIR fixedParams params latentState
  infectionCount ← observeSIR params latentState
  return (latentState ', infectionCount)

paramsPrior :: (Observables env ["ρ", "β", "γ"] Double) ⇒ Model env ts Params
paramsPrior = do
  pRho   ← beta' 2 7 #ρ
  pBeta  ← gamma' 2 1 #β
  pGamma ← gamma' 1 (1/8) #γ
  return (Params pRho pBeta pGamma)

hmmSIRNsteps :: (Observable env "inf" Int, Observables env ["ρ", "β", "γ"] Double)
             ⇒ FixedParams → Int → LatentState → Model env ts LatentState
hmmSIRNsteps fixedParams n latentState = do
  params ← paramsPrior
  foldl (>=>) return (replicate n (hmmSIR fixedParams params)) latentState
```





Bayesian Neural Network

We can also model neural networks in a Bayesian context. The following program illustrates two-dimensional logistic regression, which we can learn and infer a predictive distribution from.

```
nnLogModel :: (Observables env ["wA", "wB", "wC"] Double
              , Observable env "label" Bool)
           ⇒ Int → [Double] → Model env es Bool
nnLogModel n_nodes xs = do
  wA ← replicateM2 (length xs) n_nodes (normal' 0 1 #wA)
  let outputA = map2 tanh ([xs] • wA)
  wB ← replicateM2 n_nodes n_nodes (normal' 0 1 #wB)
  let outputB = map2 tanh (outputA • wB)
  wC ← replicateM2 n_nodes 1 (normal' 0 1 #wC)
  let outputC = sigmoid (outputB • wC)
  bernoulli ' outputC #label
```