Introduction to Anglican

Jan-Willem van de Meent
Anatomy of an Anglican Program

```clojure
(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
    [anglican emit runtime stat])
  (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let[samples (doquery :rmh one-flip [true]]
    (prn (frequencies
      (map :result
        (take 1000 samples)))))
```

Anatomy of an Anglican Program

```
(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
  [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let[samples (doquery :rmh one-flip [true]])
    (prn (frequencies
      (map :result
        (take 1000 samples)))))
```
Anatomy of an Anglican Program

```
(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
  [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let[samples (doquery :rmh one-flip [true]])
    (prn (frequencies
      (map :result
        (take 1000 samples))))))
```
Anatomy of an Anglican Program

```
(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
   [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
 [outcome]
 (let [theta (sample (beta 5 3))]
   (observe (flip theta) outcome)
   (> theta 0.7)))

(defn -main
 [& args]
 (let [samples (doquery :rmh one-flip [true])]
   (prn (frequencies
         (map :result
             (take 1000 samples)))))
```
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
  [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
 [outcome]
 (let [theta (sample (beta 5 3))]
  (observe (flip theta) outcome)
  (> theta 0.7)))

(defn -main
 [& args]
 (let [samples (doquery :rmh one-flip [true]])
  (prn (frequencies
       (map :result
            (take 1000 samples)))))

Anglican program
(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
    [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true])]
    (prn (frequencies
      (map :result
        (take 1000 samples)))))
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
  [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let[samples (doquery :rmh one-flip [true]]
    (prn (frequencies
          (map :result
               (take 1000 samples))))))

Arguments
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
   [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
 [outcome]
 (let [theta (sample (beta 5 3))]
   (observe (flip theta) outcome)
   (> theta 0.7))))

(defn -main
 [& args]
 (let [samples (doquery :rmh one-flip [true]])
   (prn (frequencies
         (map :result
               (take 1000 samples))))))
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
   [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
 [outcome]
 (let [theta (sample (beta 5 3))]
   (observe (flip theta) outcome)
   (> theta 0.7)))

(defn -main
 [& args]
 (let[samples (doquery :rmh one-flip [true]])
   (prn (frequencies
       (map :result
         (take 1000 samples))))))
Anatomy of an Anglican Program

(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
           [anglican emit runtime stat])
  (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true])]
    (prn (frequencies
          (map :result
               (take 1000 samples))))))

Generate a random value
Anatomy of an Anglican Program

(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
    [anglican emit runtime stat])
  (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let[samples (doquery :rmh one-flip [true]])
    (prn (frequencies
      (map :result
        (take 1000 samples))))))
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
  [anglican emit runtime stat])
(:gen-class))

(defquery one-flip
 [outcome]
  (let [theta (sample (beta 5 3))]
   (observe (flip theta) outcome)
   (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true]])
    (prn (frequencies
           (map :result
                (take 1000 samples))))))

Lazy sequence of samples
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
  [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true])]
    (prn (frequencies
      (map :result
        (take 1000 samples))))))
Anatomy of an Anglican Program

(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
   [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true]])
    (prn (frequencies
         (map :result
          (take 1000 samples))))))
Anatomy of an Anglican Program

(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
   [anglican emit runtime stat])
  (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true])]
    (prn (frequencies
      (map :result
        (take 1000 samples))))))
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
   [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true]])
    (prn (frequencies
      (map :result
        (take 1000 samples)))))
Argument values
Anatomy of an Anglican Program

```clojure
(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
    [anglican emit runtime stat])
  (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true])]
    (prn (frequencies
           (map :result
                (take 1000 samples)))))
```

Analysis
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
   [anglican emit runtime stat])
 (:gen-class))

(defquery one-flip
 [outcome]
 (let [theta (sample (beta 5 3))]
   (observe (flip theta) outcome)
   (> theta 0.7)))

defn -main
 [& args]
 (let [samples (doquery :rmh one-flip [true])]
   (prn (frequencies
     (map :result
       (take 1000 samples)))))
Anatomy of an Anglican Program

(ns examples.one-flip
 (:use [anglican.core :exclude [-main]]
  [anglican emit runtime stat])
(:gen-class))

(defquery one-flip
 [outcome]
 (let [theta (sample (beta 5 3))]
  (observe (flip theta) outcome)
  (> theta 0.7)))

(defn -main
 [& args]
 (let [samples (doquery :rmh one-flip [true])]
  (prn (frequencies
   (map :result
    (take 1000 samples)))))
How do I run this?

(ns examples.one-flip
  (:use [anglican.core :exclude [-main]]
       [anglican emit runtime stat])
  (:gen-class))

(defquery one-flip
  [outcome]
  (let [theta (sample (beta 5 3))]
    (observe (flip theta) outcome)
    (> theta 0.7)))

(defn -main
  [& args]
  (let [samples (doquery :rmh one-flip [true])]
    (prn (frequencies
           (map :result
                 (take 1000 samples))))))