Probabilistic Programming

2016 Vermont Code Camp
Eric Smith
@eric_s_smith

Probabilistic Programming System

- **Probabilistic program**
  “Usual programs with two added constructs:
  (1) the ability the draw values at random from distributions
  (2) the ability to condition values of variables in a program via *observe* statements. ”

- **Probabilistic inference**
  “Computing an explicit representation of the probability distribution implicitly specified by a probabilistic program"

Probabilistic Programming (2014), Gordon et al

Possible worlds

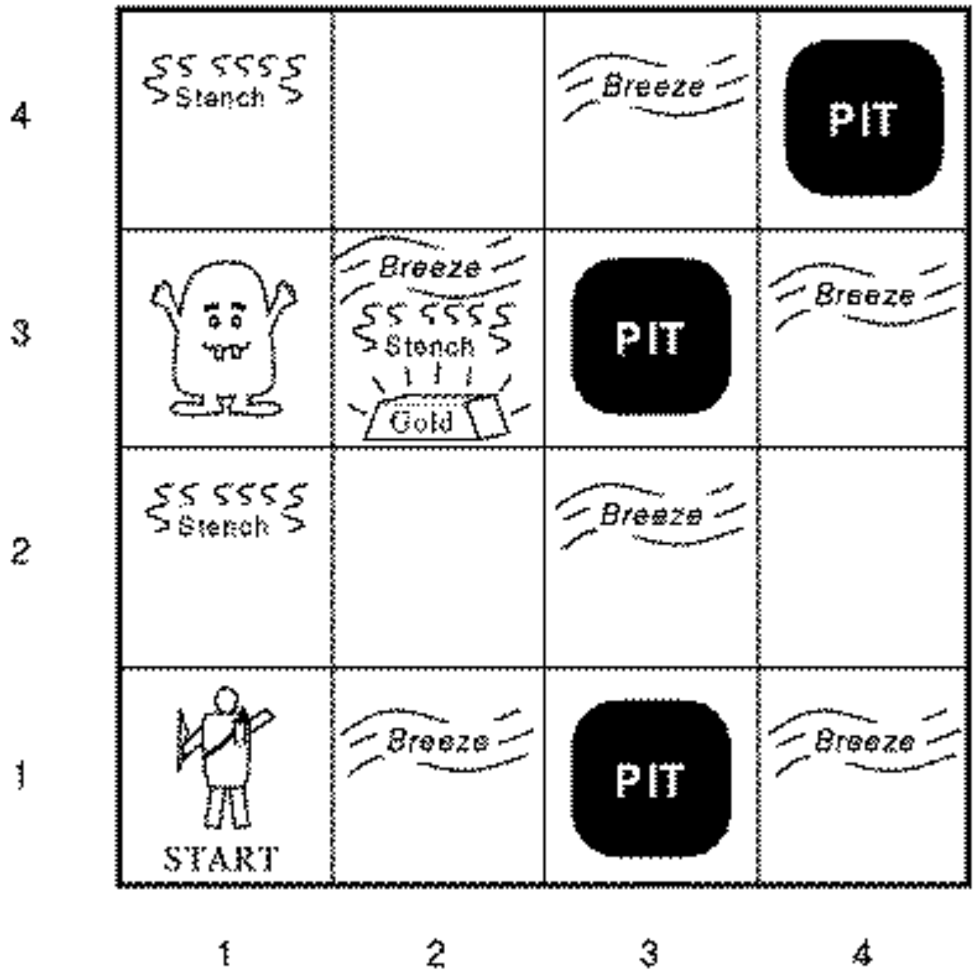
$$x + y = 4$$

When is this true?

Logical inference

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

A Wumpus World



http://www.sdsc.edu/~tbailey/teaching/cse151

Sample space

Ω = { HHH, HHT, HTH, THH, HTT, THT, TTH, TTT }

Flipping a fair coin 3 times

Event

A = { HHH, HHT, HTH, HTT }

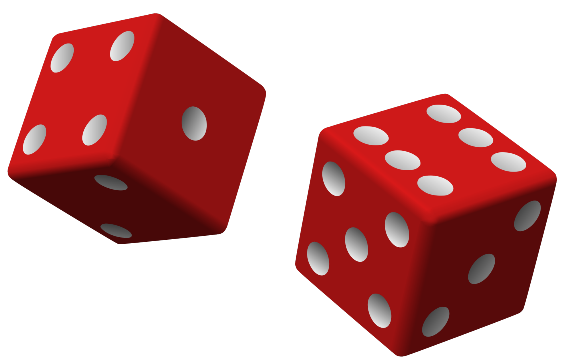
A = *Get Heads on the first flip*

B = {HHH, HHT, THH, THT }

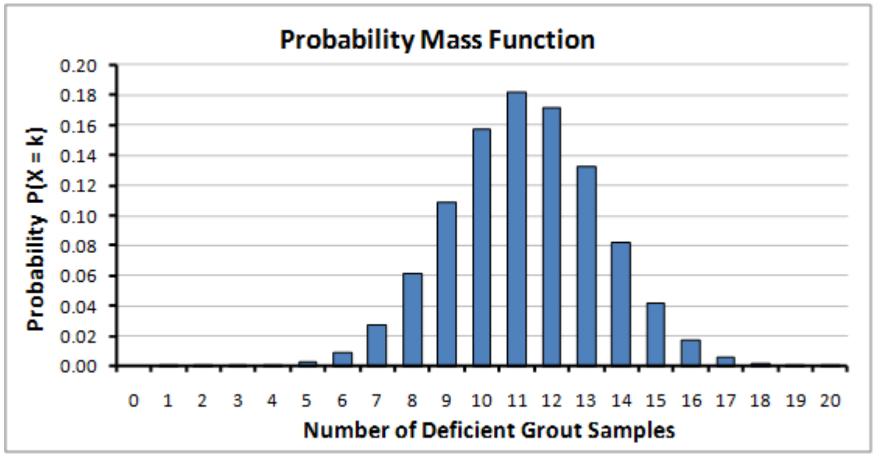
B = *Get Heads on the second flip*

Probability

Degree of belief of some event being true expressed as number from 0 to 1.



Probability Distribution



Discrete values

Continuous values

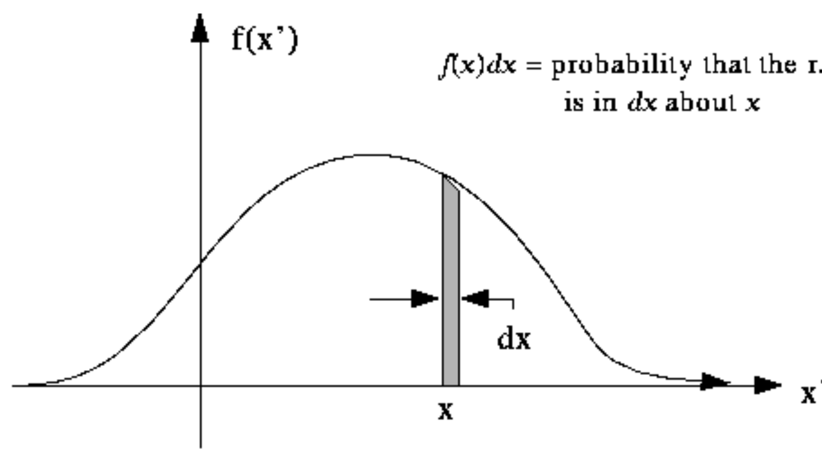
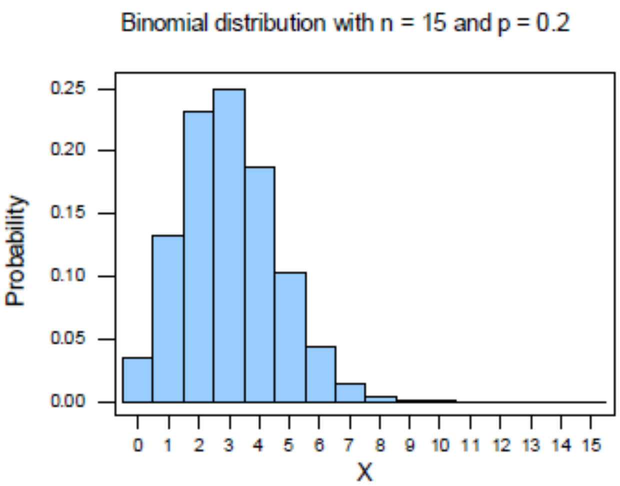
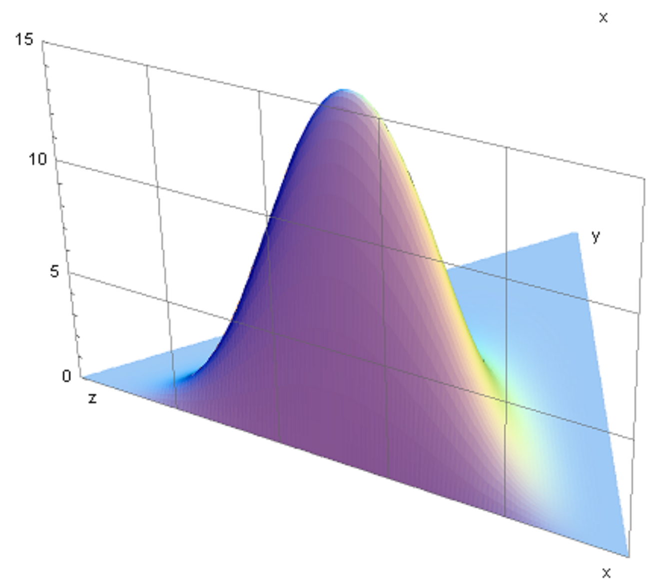
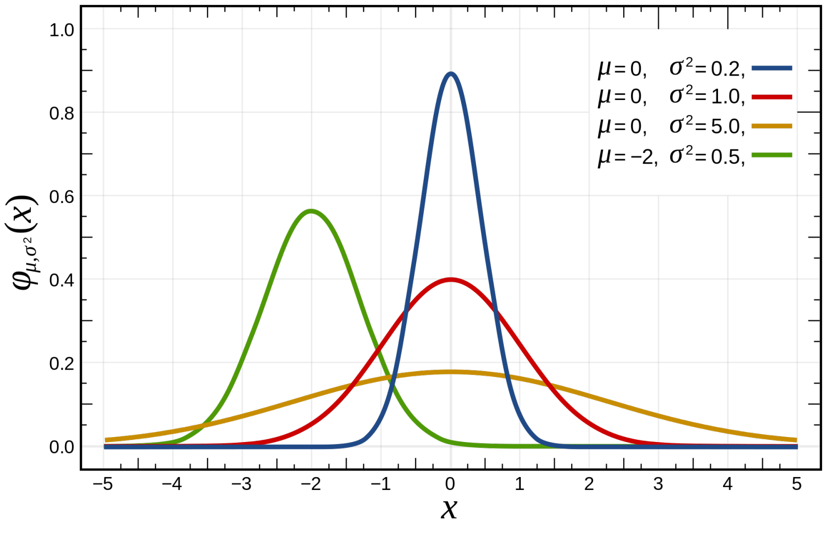


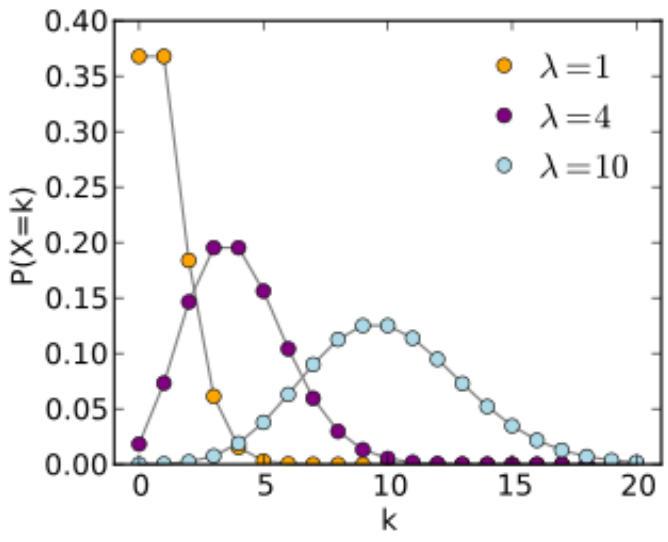
Figure 4. Typical Probability Distribution Function (*pdf*)

Distributions









Draw values at random from distributions

Anglican Demo

webppl Demo

127.0.0.1

Vermont Code Camp 8! - eric2.71828@gmail.com -... | Gorilla REPL - anglican-user | webppl - probabilistic programming for th

Gorilla REPL

Welcome to gorilla :-)

Shift + enter evaluates code. Hit ctrl+g twice in quick succession or click the menu icon (upper-right corner) for more commands ...

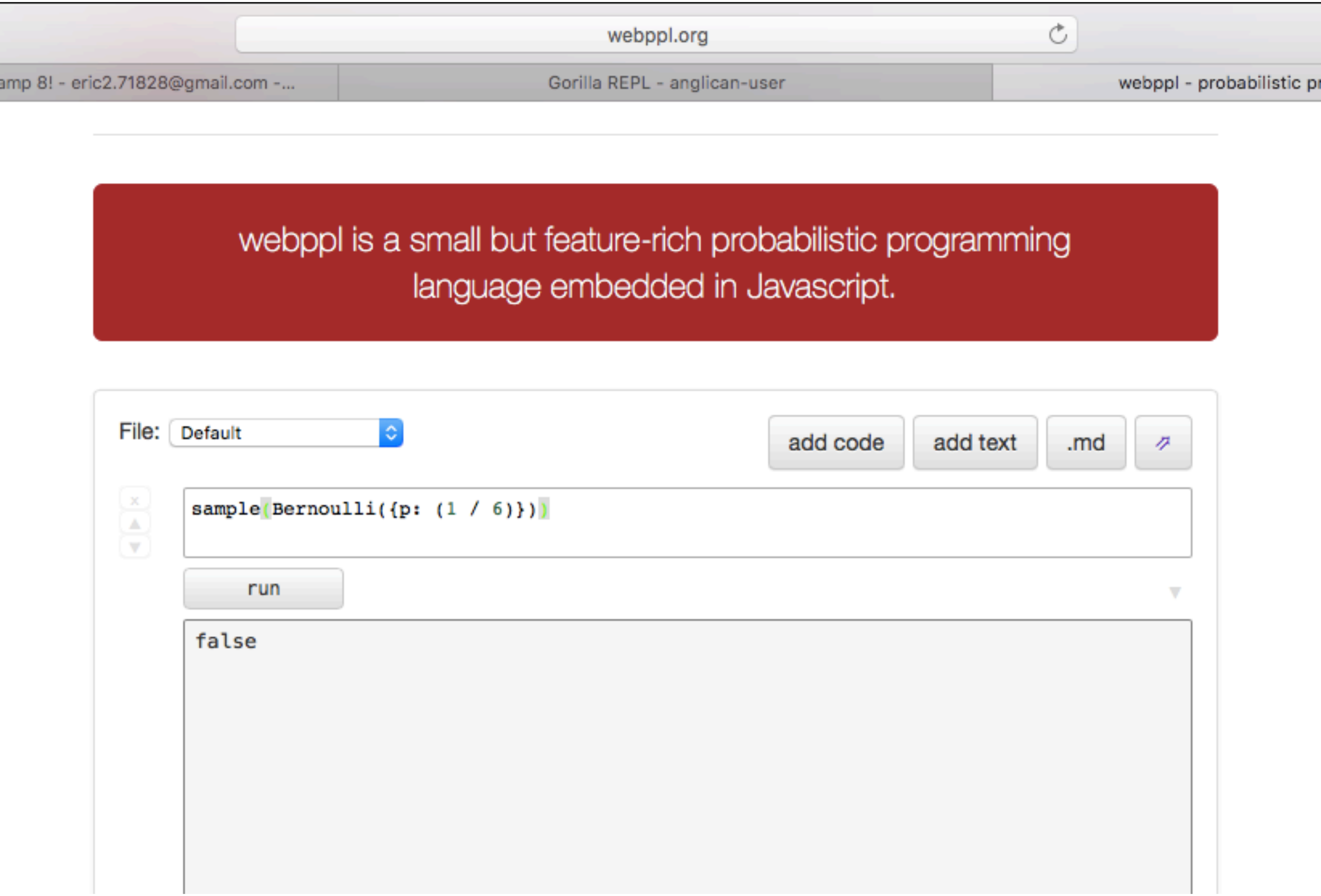
It's a good habit to run each worksheet in its own namespace: feel free to use the declaration we've provided below if you'd like.

```
(ns balmy-ocean
  (:require [gorilla-plot.core :as plot])
  (:use [anglican emit runtime]))
```

```
nil
```

```
(sample* (bernoulli 1/6))
```

```
0
```


webppl.org
amp 8! - eric2.71828@gmail.com -...
Gorilla REPL - anglican-user
webppl - probabilistic pr
webppl is a small but feature-rich probabilistic programming language embedded in Javascript.
File: Default
add code
add text
.md
sample(Bernoulli({p: (1 / 6)}))
run
false

Prior vs Posterior Probability

A *prior probability* is your understanding before making an observation.

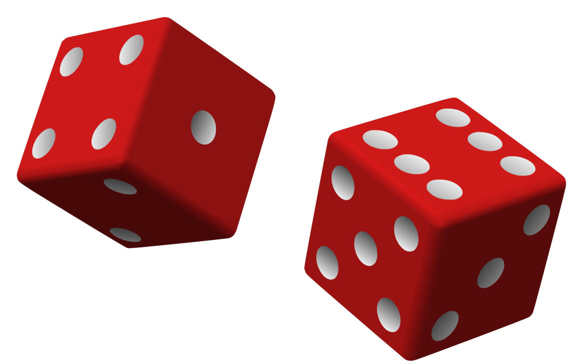
A *posterior (or conditional) probability* is using an observation to revise a prior.

Conditional Probability

A = { 3 } *3 dots*

B = { 3, 4, 5, 6 } *at least 3 dots*

P(A|B) is 1/4



Condition value of variables

Anglican Demo

127.0.0.1

Vermont Code Camp 8! - eric2.71828@gmail.com -... | Gorilla REPL - anglican-user | webppl - probabilistic programming for

Gorilla REPL

Welcome to gorilla :-)

Shift + enter evaluates code. Hit ctrl+g twice in quick succession or click the menu icon (upper-right corner) for more commands ...

It's a good habit to run each worksheet in its own namespace: feel free to use the declaration we've provided below if you'd like.

```
(ns balmy-ocean
  (:require [gorilla-plot.core :as plot])
  (:use [anglican emit runtime]))
```

```
nil
```

```
(observe* (bernoulli 1/6) 0)
```

```
-0.1823215567939547
```

Joint Distribution

	Toothache		No Toothache	
	Catch	No Catch	Catch	No Catch
Cavity	0.108	0.012	0.072	0.008
No Cavity	0.016	0.064	0.144	0.576

$$P(\text{cavity} \mid \text{toothache}) = \frac{P(\text{cavity} \land \text{toothache})}{P(\text{toothache})}$$

$$= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064}$$

$$= 0.6$$

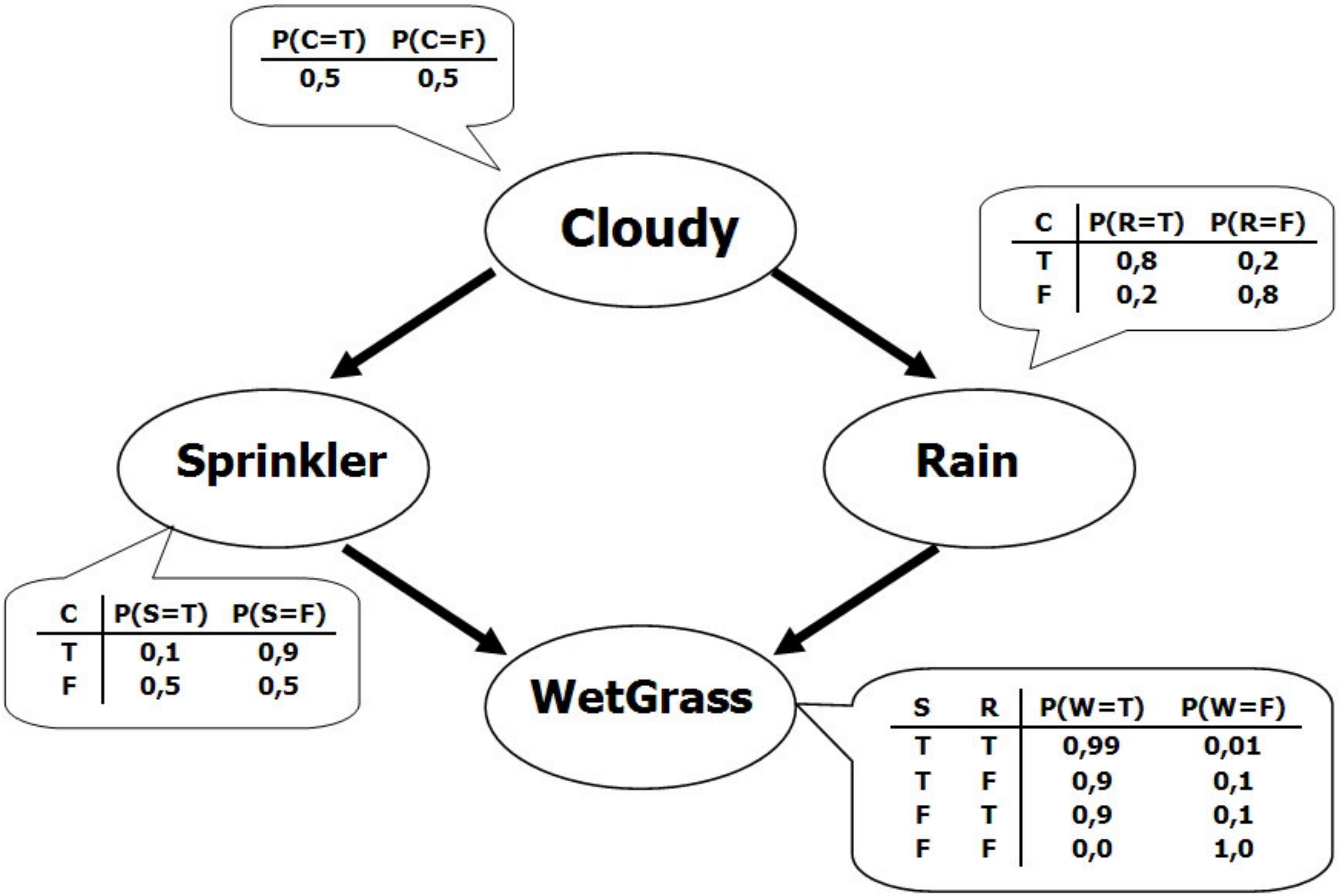
Bayes Rule

$$P(\text{cause} \mid \text{effect}) = \frac{P(\text{effect} \mid \text{cause})\, P(\text{cause})}{P(\text{effect})}$$

Meningitis can cause stiff neck 70% of the time.	$P(s \mid m) = 0.7$
Incidence of meningitis is 1/50,000.	$P(m) = 1/50{,}000$
Probability that any patient has a stiff neck is 1%.	$P(s) = 0.01$

$$P(m \mid s) = \frac{P(s \mid m)\, P(m)}{P(s)} = \frac{0.7 \times 1/50{,}000}{0.01} = 0.0014$$

Bayes Net



What is the probability that it is raining, given that the sprinkler
Is on and the grass is wet?

Compute an explicit representation of the probability distribution implicitly specified by a probabilistic program

Anglican Demo

127.0.0.1

Vermont Code Camp 8! - eric2.71828@gmail.com -... | Gorilla REPL - anglican-user | webppl - probabilistic programming for th

Gorilla REPL

Welcome to gorilla :-)

Shift + enter evaluates code. Hit ctrl+g twice in quick succession or click the menu icon (upper-right corner) for more commands ...

It's a good habit to run each worksheet in its own namespace: feel free to use the declaration we've provided below if you'd like.

```
(ns bayes-net
  (:require [gorilla-plot.core :as plot]
        [anglican.stat :as s])
  (:use clojure.repl
    [anglican core runtime emit
     [state :only [get-predicts]]
     [inference :only [collect-by]]]
    [clojure.string :only (join split blank?)]))
```

```
nil
```

```
(defquery sprinkler-bayes-net [sprinkler wet-grass]
  (let [is-cloudy (sample (flip 0.5))

    is-raining (cond (= is-cloudy true )
             (sample (flip 0.8))
             (= is-cloudy false)
             (sample (flip 0.2)))
    sprinkler-dist (cond (= is-cloudy true)
                 (flip 0.1)
                 (= is-cloudy false)
                 (flip 0.5))
    wet-grass-dist (cond
             (and (= sprinkler true)
                  (= is-raining true))
             (flip 0.99)
             (and (= sprinkler false)
                  (= is-raining false))
```

127.0.0.1

t in Vermont Code Camp 8! - eric2.71828@gmail.com -... | Gorilla REPL - anglican-user | webppl - probabilistic programming for the w

```
(defquery sprinkler-bayes-net [sprinkler wet-grass]
  (let [is-cloudy (sample (flip 0.5))

    is-raining (cond (= is-cloudy true )
                 (sample (flip 0.8))
                 (= is-cloudy false)
                 (sample (flip 0.2)))
    sprinkler-dist (cond (= is-cloudy true)
                     (flip 0.1)
                     (= is-cloudy false)
                     (flip 0.5))
    wet-grass-dist (cond
                 (and (= sprinkler true)
                      (= is-raining true))
                 (flip 0.99)
                 (and (= sprinkler false)
                      (= is-raining false))
                 (flip 0.0)
                 (or  (= sprinkler true)
                      (= is-raining true))
                 (flip 0.9))]
    (observe sprinkler-dist sprinkler)
    (observe wet-grass-dist wet-grass)

    is-raining))
```

```
#'bayes-net/sprinkler-bayes-net
```

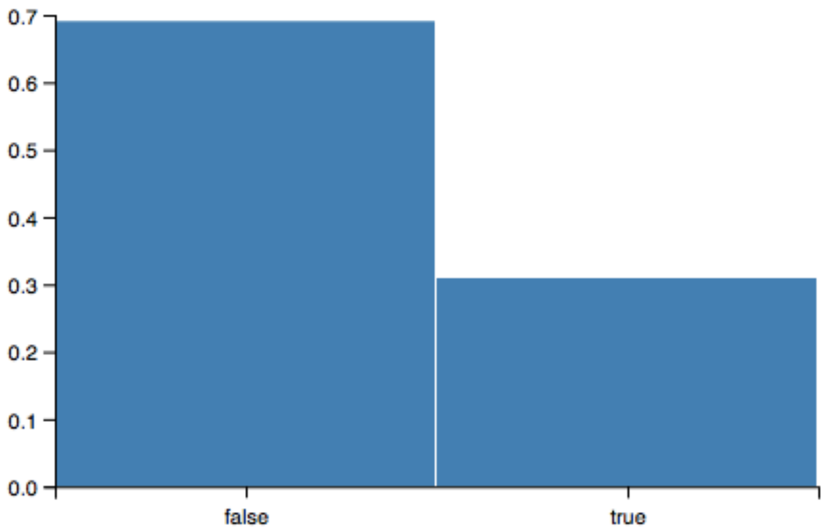
```
(->> (doquery :smc sprinkler-bayes-net [true true] :number-of-particles 100)
     (take 10000)
     (collect-by :result)
     (s/empirical-distribution)
     (#(plot/bar-chart (keys %) (vals %))))
```

127.0.0.1

Vermont Code Camp 8! - eric2.71828@gmail.com -... | Gorilla REPL - anglican-user | webppl - probabilistic programming for th

```
#'bayes-net/sprinkler-bayes-net
```

```
(->> (doquery :smc sprinkler-bayes-net [true true] :number-of-particles 100)
     (take 10000)
     (collect-by :result)
     (s/empirical-distribution)
     (#(plot/bar-chart (keys %) (vals %))))
```



Probabilistic Models of Cogniti ×

https://probmods.org/index.html

Probabilistic Models of Cognition

by Noah D. Goodman *and* Joshua B. Tenenbaum

In this book, we explore the probabilistic approach to cognitive science, which models learning and reasoning as inference in complex probabilistic models. In particular, we examine how a broad range of empirical phenomena in cognitive science (including intuitive physics, concept learning, causal reasoning, social cognition, and language understanding) can be modeled using a functional probabilistic programming language called Church.

How to use

Best viewed in Chrome/Safari on a laptop/desktop (smartphone/tablet not recommended).

This book contains exercises where you write and run Church code directly in the browser. To save your progress on these exercises, you can register an account. Registering also helps us improve the book by tracking what kinds of programs users run and what kinds of errors they encounter.

Login or register an account

How to cite

Chapters

webppl - probabilistic program ×

webppl.org

webppl *probabilistic programming for the web*

On Github

webppl is a small but feature-rich probabilistic programming language embedded in Javascript.

File: Default

add code | add text | .md

```
var geometric = function() {
  return flip(.5) ? 0 : geometric() + 1;
}

var conditionedGeometric = function() {
  var x = geometric();
  factor(x > 2 ? 0 : -Infinity);
  return x;
}

var dist = Infer(
  {method: 'enumerate', maxExecutions: 10},
  conditionedGeometric);

viz.auto(dist);
```

run

The Anglican Probabilistic Prog ×

www.robots.ox.ac.uk/~fwood/anglican/language/index.html

ANGLICAN

HOME NEWS PAPERS LANGUAGE INFERENCE USAGE CONTRIBUTE EXAMPLES

LANGUAGE SYNTAX

The programming language of Anglican is a subset of Clojure, extended with a few special forms that make it a probabilistic programming language. These forms are `sample` for drawing a samples from distributions and `observe` for conditioning on data. There are other special forms — `mem`, `store`, and `retrieve` — which make writing probabilistic programs easier.

The following documentation is quite terse because Anglican is, to a large extent, intentionally syntactially indistiguishable from Clojure.
Clojure reference materials, obtainable from the web via standard search procedures, are as essential to programming in Anglican as is this Anglican language documentation. The key to Anglican knowing Clojure *and* understanding `defquery`, the interface between Clojure and Anglican.

This interface is meant to be as transparent as possible and for as much Clojure functionality to work inside `defquery` as possible.

In general the documentation that follows indicates functionality *relative to Clojure*. For instance, the absence of an explicit statement of existence means that the Clojure language feature probably isn't supported in Anglican.

A FIRST EXAMPLE PROGRAM

Anglican programs reside in Clojure source code modules, and are delimited by `defquery` (a macro). In order to enable the Anglican language in a Clojure module, at the minimum namespaces `anglican.runtime` and `anglican.emit` must be used. A simple way to do this is to write

```
(ns example
  (:use [anglican emit runtime]))
```

in the beginning of a Clojure module, for example 'example.clj'. Clojure namespacing is notably complex; arguably the best strategy for writing a namespace that includes Anglican functionality is to copy namespace declarations from the provided examples.

Manning | Practical Probabilist ×

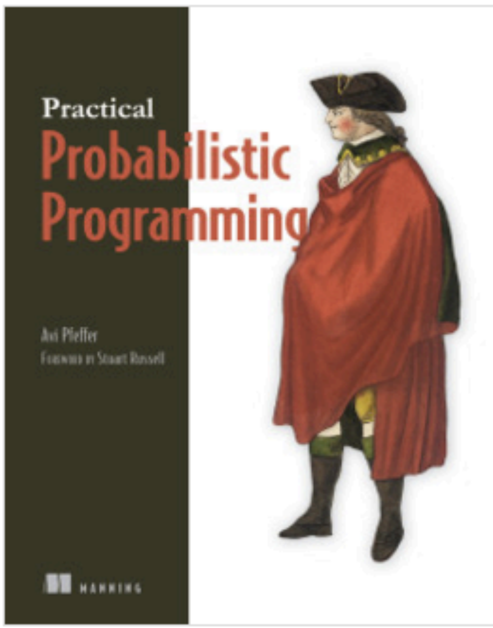
https://www.manning.com/books/practical-probabilistic-programming

home | all books | MEAP | support | contact | free content | forums

MANNING PUBLICATIONS

search manning.com

DEAL OF THE DAY: *Get half off C++ Concurrency in Action - use code dotd091816*



Practical Probabilistic Programming

Avi Pfeffer

Foreword by Stuart Russell

March 2016 · ISBN 9781617292330 · 456 pages · printed in black & white

> *An important step in moving probabilistic programming from research laboratories out into the real world.*
>
> From the Foreword by Stuart Russell, UC Berkeley

Practical Probabilistic Programming introduces the working programmer to probabilistic programming. In it, you'll learn how to use the PP paradigm to model application domains and then express those probabilistic models in code. Although PP can seem abstract, in this book you'll immediately work on practical examples, like using the Figaro language to build a spam filter and applying Bayesian and Markov networks, to diagnose computer system data problems and recover digital images.

Buy

combo
pBook +

eBook
pdf + eP

FREE DOWNLOADS

Chapter 1

Infer.NET

research.microsoft.com/en-us/um/cambridge/projects/infernet/docs/Infer.NET%20tutorials%20and%20examples.aspx

Microsoft Research

infer.net

- Home
- Download
- Job openings

Documentation

- User Guide
- Tutorials & Examples
- Learners
- API Documentation
- Resources & References
- Introduction
- Infer.NET 101

Case Studies

- Childhood Asthma
- Genetic Causes of Disease
- Papers using Infer.NET

Extensions

- KJIT

Infer.NET user guide

Tutorials & Examples

Tutorials

The following tutorials provide a step-by-step introduction to Infer.NET. Can be viewed through the **Examples Browser**.

1. **Two coins** - a first tutorial, introducing **the basics** of Infer.NET.
2. **Truncated Gaussian** - using **variables and observed values** to avoid unnecessary compilation.
3. **Learning a Gaussian** - using ranges to handle **large arrays** of data; **visualising** your model.
4. **Bayes Point Machine** - demonstrating how to **train and test** a Bayes point machine classifer.
5. **Clinical trial** - using **if blocks** for **model selection** to determine if a new medical treatment is effective.
6. **Mixture of Gaussians** - constructing a **multivariate mixture** of Gaussians.

String Tutorials

The following tutorials provide an introduction to an experimental Infer.NET feature: inference over string variables. The first two tutorials can be viewed through the **Examples Browser**, and the third one is available as a separate project.

1. **Hello, Strings!** - introduces **the basics** of performing inference over string variables in Infer.NET.
2. **StringFormat Operation** - demonstrates a powerful string operation supported in Infer.NET, **StringFormat**.
3. **Motif Finder** - defining a **complex model** combining string, arrays, integer arithmetic and control flow statements.

Short Examples

Short examples of using Infer.NET to solve a variety of different problems. Can be viewed through the

Stan

mc-stan.org

Stan

Thousands of users rely on Stan for statistical modeling, data analysis, and prediction in the social, biological, and physical sciences, engineering, and business.

Users specify log density functions in Stan's probabilistic programming language and get:

Jupyter Notebook Viewer

nbviewer.jupyter.org/github/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/blob/master/Chapter1_Intro

Looking for a printed version of Bayesian Methods for Hackers?

Bayesian Methods for Hackers is now a published book by Addison-Wesley, available on Amazon!