Probabilistic Programming with Lea

by Pierre Denis
pie.denis@skynet.be

PROB PROG 2020

Lea in a nutshell…

Lea 3

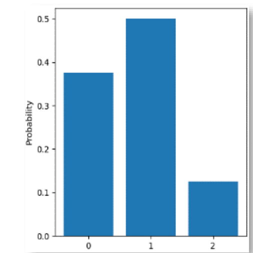
- **PP library for Python** 2.6+ and 3.x
- **Discrete probabilities only** – support: numbers, strings, times, …
- **Comprehensive toolkit** – CPT, BN, JPD, Markov chains, probabilistic arithmetic, standard indicators, information theory, random sampling, plotting, etc.
- **Open probability representation** – float, fraction, decimal
- **Symbolic computation** – enablable by SymPy library
- **Exact algorithm**, by default – new "Statues" algorithm
- **Approximate algorithms**, if needed – rejection sampling, likelihood weighting
- **Machine learning** – maximum likelihood, EM algorithms
- **Easy!** – comprehensive Wiki tutorials & examples / Jupyter notebooks
- **Open-source** – LGPL license – Find Lea on Git repo & PyPI

Check out online
Lea playground

PP made easy!

```
>>> flipA = bernoulli(0.50)
>>> flipB = bernoulli(0.25)
>>> flips = flipA + flipB
>>> P(flipB == 0)
0.75
>>> flips
0 : 0.375
1 : 0.5
2 : 0.125
>>> flips.plot()
>>> P(flips <= 1)
0.875
>>> P((flips==0) | (flips==1))
0.875
>>> P((flipA==1) & (flips<=1))
0.375
>>> P(flipA==1) * P(flips<=1)
0.4375
>>> P((flipA==1).given(flips<=1))
0.42857142857142855
```

All operators are overloaded:
`+ - * / < == & | ~ …`



0.375 ≠ 0.4375
→ `flipA` and `flips` are interdependent!

conditional probability = $\frac{0.375}{0.875}$

Open probability representation

from fractions…

```
>>> flipA = bernoulli('1/2')
>>> flipB = bernoulli('1/4')
>>> flips = flipA + flipB
>>> flips
0 : 3/8
1 : 4/8
2 : 1/8
>>> flips.mean
3/4
>>> P((flipA==1).given(flips<=1))
3/7
```

… to symbolic computation

```
>>> flipA = bernoulli('a')
>>> flipB = bernoulli('b')
>>> flips = flipA + flipB
>>> flips
0 : (a - 1)*(b - 1)
1 : -2*a*b + a + b
2 : a*b
>>> flips.mean
a + b
>>> P((flipA==1).given(flips<=1))
a*(b - 1)/(a*b - 1)
```

powered by **SymPy**

Jupyter Notebook session
with automatic **LaTex** rendering

```
In [2]:  x = binom(2,'p')
         y = binom(4,'q')
         z = x + y
         P((y==2).given(z<=3,x>=1))
```

Out[2]:

$$\frac{12q^2\,(p-1)}{9pq^2 + 2pq + p - 6q^2 - 4q - 2}$$

A murder party (BN example)

```
>>> killer = pmf({ "Colonel Mustard": 0.60,
                   "Mrs. White"     : 0.30,
                   "Mrs. Peacock"   : 0.10 })
>>> mrs_white_is_absent = \
    if_( killer == "Mrs. White", event(0.90),
                                 event(0.20))
>>> mrs_peacock_knows_who_is_killer = \
    if_( killer != "Mrs. Peacock", event(0.75),
                                   True       )
>>> mrs_peacock_is_absent = \
    if_( (killer == "Mrs. Peacock")
        | mrs_peacock_knows_who_is_killer, event(0.99),
                                           event(0.10))
>>> evidences = [ mrs_peacock_is_absent,
                 ~mrs_white_is_absent  ,
                  killer[:4] == "Mrs." ]
>>> killer.given(*evidences)
Mrs. Peacock : 0.7747615553925166
Mrs. White   : 0.22523844460748343
```

prior probabilitiies

if Mrs. White is the killer,
then she's absent with prob. 90%
else she's absent with prob. 20%

if Mrs. Peacock is innocent,
then she knows who's the killer
with prob. 75%

if Mrs. Peacock is the killer
or she knows who's the killer
then she's absent with prob. 99%
else she's absent with prob. 10%

Evidences:
1. Mrs. Peacock is absent.
2. Mrs White is present.
3. The killer is a woman.