# Software Libraries for PGMs

Kevin Rothi

# Very popular tools for ML/NNs/Deep Learning...

- SciKit Learn
- Tensorflow
- Keras
- Torch
- CUDA
- Theano
- Caffe

# No shortage of small libraries for graphical models…

http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html

(Last updated 16 June 2014)

69 Libraries

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| UC Irvine | Dechter (UCI) | Y | D | N | N | N | N | 0 | UD | AND/OR Search | Bucket Elimination, AND/OR search for P(evidence), MPE in Bay |

| Name | Authors | Src | Cts | GUI | Params | Struct | Utility | Free | Undir | Inference | Comments |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AgenaRisk | Agena | N | Cx | Y | Y | N | N | $ | D | JTree | Simulation by Dynamic discretisation |
| Analytica | Lumina | N | G | Y | N | N | Y | $ | D | sampling | spread sheet compatible |
| B-course | U. Helsinki | N | Cd | Y | Y | Y | N | 0 | D | ? | Runs on their server: view results using a web browser. |
| Banjo | Hartemink | Java | Cd | N | N | Y | N | 0 | D | none | structure learning of static or dynamic networks of discrete variables |
| Bassist | U. Helsinki | C++ | G | N | Y | N | N | 0 | D | MH | Generates C++ for MCMC. (No longer maintained) |
| BayesBuilder | Nijman (U. Nijme | N | D | Y | N | N | N | 0 | D | ? | - |
| BayesiaLab | Bayesia Ltd | N | Cd | Y | Y | Y | N | $ | CG | jtree,G | Structural learning, adaptive questionnaires, dynamic models |
| Bayes-Scala | Daniel Korzekwa | Scala | D | N | Y | N | N | Y | UD | Loopy BP | Loopy BP in Cluster Graph, EM learning (BN, unrolled DBN, incomplete data) |
| Bayes Server | Bayes Server | N | G | Y | Y | N | N | $ | D | RelevanceTree,V | Supports inference and learning with Dynamic Bayesian networks and continuous variables |
| Bayesware Disco | Bayesware | N | Cd | Y | Y | Y | N | $ | D | ? | Uses bound and collapse for learning with missing data. |
| BayesBlocks | Helsinki | Python/C++ | Y | N | Y | N | N | 0 | Dir | Variational | Non-Gaussian Latent variable models |
| Bayes net learne | Moore, Wong (C | N | D | N | Y | Y | N | 0 | D | SL | optimal reinsertion algorithm |
| Belief net power | Cheng (U.Alberta | N | D | Y | Y | CI | N | 0 | D | ? | - |
| Blaise | Bonnowitz and M | Java | Y | N | Y | N | N | 0 | Fgraph | MCMC, SMC | General MC toolkit, also handles non-parametric Bayesian models |
| BNT | Murphy (U.C.Ber | Matlab/C | G | N | Y | Y | Y | 0 | D,U | Many | Also handles dynamic models, like HMMs and Kalman filters. |
| BNJ | Hsu (Kansas) | Java | D | Y | N | Y | N | 0 | D | jtree, IS | - |
| BNL | frank rijmen | Matlab | D | N | N | N | N | 0 | D | jtree | Supports (ordinal) logistic regression CPDs and EM learning |
| BUGS | MRC/Imperial Co | N | Cs | W | Y | N | N | 0 | D | Gibbs | - |
| Causal discovere | Vanderbilt | N | - | - | N | Y | N | 0 | D | - | structure learning only |
| CoCo+Xlisp | Badsberg (U. Aa | C/lisp | D | Y | Y | CI | N | 0 | U | Jtree | Designed for contingency tables. |
| CIspace | Poole et al. (UBC | Java | D | Y | N | N | N | 0 | D | Varelim | - |
| CRFtoolbox | Schmidt and Mur | Matlab/C | N | N | Y | N | N | 0 | U | Loopy BP | Conditional random fields, arbitrary structure |
| Darwin | Stephen Gould | C++ | N | N | N | N | N | Y | U | Various | Approx. inference, focus on CRFs for vision applications |
| DBNbox | Roberts et al | Matlab | Y | N | Y | N | N | Y | D | Various | DBNs |
| Deal | Bottcher et al | R | G | Y | Y | Y | N | 0 | D | None | Structure learning. |
| DeriveIt | DeriveIt LLC | N | ? | ? | Y | Y | ? | $ | D | Jtree, Gibbs | Exploits local structure in CPDs. |
| Dimple | Shawn Hershey ( | MATLAB/Java | G, Cs, Cd | N | Y | N | N | 0 | UD, CG | BP,G,MH | Parameter learning is mostly limited to EM for now |
| Elvira | Elvira consortium | Java | Cd,Cx | Y | Y | Y | Y | 0 | D | JTree,varelim,IS | Also includes classification, abductive inference and model fusion |
| Ergo | Noetic systems | N | D | Y | N | N | N | $ | D | jtree | - |
| Factorie | Andrew McCallu | Scala | N | N | Y | N | N | 0 | U | Various | Also supports relational factor graphs |
| FastInf | Jaimovich et al (I | C++ | D | N | Y | N | N | 0 | U | JTree,G,VMP | Also supports GBP,TRBP |
| Figaro | Avi Pfeffer/ Charl | Scala | - | - | Y | Y | Y | $ | U | S | Also supports probabilistic relational models |
| GDAGsim | Wilkinson (U. Ne | C | G | N | N | N | N | 0 | D | Exact | Bayesian analysis of large linear Gaussian directed models. |
| GeNIe and SMIL | Decision System | SMILE wrappers | Cs,equations | W,U,M | Y | Y | Y | free | D | JTree,sampling | DBNs, support for diagnostic applications |
| GGM | West et al (Duke | C++ | G | N | Y | Y | N | 0 | U | SL | MCMC and stochastic search for structure learning of GGMs |
| GMRFsim | Rue (U. Trondhe | C | G | N | N | N | N | 0 | U | MCMC | Bayesian analysis of large linear Gaussian undirected models. |
| GMTk | Bilmes (UW), Zw | N | D | N | Y | Y | N | 0 | D | Jtree | Designed for speech recognition. |
| GOBNILP | U. of York | Y | D | N | N | Y | N | 0 | D | none | exact structure learning from data or local scores, k-best learning possible |
| gR | Lauritzen et al. | R | - | - | - | - | - | 0 | - | - | Various packages |
| Grappa | Green (Bristol) | R | D | N | N) | N | N | 0 | D | Jtree | - |
| HdBCS | Dobra et al (Was | C++ | G | N | Y | Y | N | 0 | U | SL | stochastic search for structure learning of GGMs |
| Hugin Expert | Hugin | N | G | W | Y | CI | Y | $ | CG | Jtree | - |
| Hydra | Warnes | Java | Cs | Y | Y | N | N | 0 | U,D | MCMC | - |
| IBAyes | Artificial Intellige | N | D | Y | N | N | N | 0 | D | Junction Tree, S | - |
| Infer.NET | John Winn, Tom | C# | Y | N | Y | N | N | 0 | Y | VMP, EP, Gibbs | Bayesian parameter estimation as well |
| JAGS | Martyn Plummer | Java | Y | N | Y | N | N | 0 | Y | Gibbs | Similar to BUGS |
| Java Bayes | Cozman (CMU) | Java | D | Y | N | N | Y | 0 | D | Varelim, jtree | - |
| LADR | Structured Data | N | Cd | N | N | Y | N | $ | D | none | Structure learning of massive static or dynamic networks |
| LibB | Friedman (Hebre | N | D | N | Y | Y | N | 0 | D | SL | Structure learning |
| libDAI | Mooij et al. | C++ | D | N | Y | N | N | 0 | Fgraph | JTree,VarElim,G, | also supports GBP,HAK,LCBP,TreeEP,TRWBP |
| MIM | HyperGraph Soft | N | G | Y | Y | Y | N | $ | CG | Jtree | Up to 52 variables. |
| Mocapy++ | U. Copenhagen | C++ | G | N | Y | N | N | 0 | D | Gibbs sampling | Support for directional statistics |
| MSBNx | Microsoft | N | D | W | N | N | Y | 0 | D | Jtree | - |
| Netica | Norsys | N | G | W | Y | N | Y | $ | D | jtree | - |
| OpenGM2 | B. Andres, T. Bei | C++, Matlab, Pyt | D | Y | N | N | N | 0 | Fgraph | Many | LP, ILP, Multicut, LBP, TRBP, QPBO, AStar, many graphcut-based methods, methods based on dual decomposition, and many more. Includes wrappers for several other related projects. |
| OpenMarkov | CISIAD(UNED, S | Y | Cs,Cd | Y | Y | Y | Y | Y | UD | jtree,varelim,sam | Java, open source, extensible; dynamic models, object oriented networks, interactive learning, ProbModelXML format |
| PMT | Pavlovic (BU) | Matlab/C | D | N | Y | N | N | 0 | D | special purpose | - |
| PNL | Eruhimov (Intel) | C++ | D | N | Y | Y | N | 0 | U,D | Jtree | A C++ version of BNT; will be released 12/03. |
| Pulcinella | IRIDIA | Lisp | D | Y | N | N | N | 0 | D | ? | Uses valuation systems for non-probabilistic calculi. |
| RISO | Dodier (U.Colora | Java | G | Y | N | N | N | 0 | D | Polytree | Distributed implementation. |
| Sam Iam | Darwiche (UCLA | N | G ? | Y | Y | N ? | Y | 0 | D | Recursive condit | Also does sensitivity Analysis |
| Stan | Gelman et al (Co | C++ | Y | N | Y | N | N | 0 | D | Hybrid Monte Ca | Generates efficient MCMC code for BUGS-like models |
| Tetrad | CMU | N | G | N | Y | CI | N | 0 | U,D | SL | - |
| UC Irvine | Dechter (UCI) | Y | D | N | N | N | N | 0 | UD | AND/OR Search | Bucket Elimination, AND/OR search for P(evidence), MPE in Bayesian networks |
| UnBBayes | Prof. Marcelo La | Java | G, Cs, Cd | Y | Y | Y | Y | 0 | D | jtree, G, samplin | Supports other probabilistic graphical models: MSBN, OOBN, PRM, and MEBN. |
| Uninet | Technical Univer | N | G, Cs, Cx | Y | Y | Y | N | 0 | UD | sampling | All probabilsitic nodes (discrete, Gaussian, non-Gaussian) are supported analytically through the Vine-Copula method with Gaussian copula. Functional nodes are supported by sampling. |
| Vibes | Winn (MSR Cam | Java | Cx | Y | Y | N | N | 0 | D | VMP | |
| WinMine | Microsoft | N | Cx | Y | Y | Y | N | 0 | U,D | SL | Learns BN or dependency net structure. |
| XBAIES 2.0 | Cowell (City U.) | N | G | Y | Y | N | Y | 0 | CG | Jtree | - |

# Of these...

23 use junction trees for inference (some use Jtrees in addition to other algos)

5 use gibbs sampling

Many seem to be defunct, unsupported, or abandoned…

Why are there so many of these?

“It’s hard to strike a balance between generality and usability.” -Prof. Ihler

# Positive qualities of software libraries… (CISQ)

Reliable

Efficient

Secure

Maintainable

Appropriately Scoped (Size)

“CISQ has defined five major desirable characteristics of a piece of software needed to provide business value…” (https://en.wikipedia.org/wiki/Software_quality)

The rest of this talk will focus on the libraries that can begin to convincingly claim to fulfill these qualities (in my opinion) graphical models library

All Videos News Shopping Images More Settings Tools

About 27,500,000 results (0.38 seconds)

**Scholarly articles for graphical models library**

... for discrete approximate inference in **graphical models** - Mooij - Cited by 256

... : A program for analysis of Bayesian **graphical models** ... - Hornik - Cited by 2332

Learning in **graphical models** - Jordan - Cited by 1767

**GitHub - pgmpy/pgmpy: Python Library for Probabilistic Graphical ...**

https://github.com/pgmpy/pgmpy ▾

pgmpy is a python **library** for working with Probabilistic **Graphical Models**. Documentation and list of algorithms supported is at our official site http://pgmpy.org/

Pgmpy · Pgmpy/pgmpy_notebook · Issues 155

**GitHub - opengm/opengm: A C++ Library for Discrete Graphical Models**

https://github.com/opengm/opengm ▾

OpenGM is a C++ template **library** for discrete factor graph **models** and distributive operations on these **models**. It includes state-of-the-art optimization and inference algorithms beyond message passing. ... The **graphical model** data structure, inference algorithms and different encodings ...

**machine learning - Good libraries for working with probabilistic ...**

https://stats.stackexchange.com/.../good-libraries-for-working-with-probabilistic-graph... ▾

Nov 7, 2013 - A relatively new (and rapidly evolving) python **library** is pgmpy (Github Link).

...

**Edward – Home**

edwardlib.org/ ▾

Edward is a Python **library** for probabilistic modeling, inference, and criticism. ... Directed **graphical models**; Neural networks (via **libraries** such as tf.layers and ...

pgmpy

Generality

Usability

# “Python library for Probabilistic Graphical Models”

- Details are sparse, but it seems that this library has its origins as a Google Summer of Code project. There appear to be 4 major contributors: Ankur Ankan from Radboud University, Yashu Seth, Abinash Panda, Utkarsh Khalibartan, and an unnamed GitHub user contributing under the handle “vivek425ster”.
- Open source
- Version 0.1.2
- Still under development (last commit on April 11)
- MIT License
- 48 contributors

# Models

Bayesian Model
Markov Model
Factor Graph
Cluster Graph
Junction Tree
Markov Chain
NoisyOr Model
Naive Bayes
DynamicBayesianNetwork

# Sampling Methods

Gibbs Sampler

Bayesian Model Samplers

Hamiltonian Monte Carlo

No U-Turn Sampler

# Algorithms

Variable Elimination
Belief Propagation
MPLP
Dynamic Bayesian Network Inference

# Positives

Very approachable (well documented)

Actively supported (bug fixes, features added)

Python

# Negatives

Not backed by Big 4 company

Development seems to be slowing down (fewer commits over time)

2nd half of talk will focus on examples of what you can do with pgmpy...

OpenGM2


Generality
Usability
OpenGM2

# “A C++ Library for Discrete Graphical Models”

- Developed at The Heidelberg Collaboratory for Image Processing at the University of Heidelberg. There are 3 main developers: Bjoern Andres, Thorsten Beier, and Joerg H. Kappes.
- Open source
- Version 2.0.2
- Still under development (last commit on April 5)
- MIT License
- 38 contributors
- Wrappers for Python and Matlab

# Models

Graphs of any order and structure, from second order grid graphs to irregular higher-order models

# Algorithms

- Combinatorial/Global Optimal Methods
- Linear Programming Relaxations
- Message Passing Methods
- Move Making Methods
- Sampling
- Wrapped External Code for Discrete Graphical Models

(41 total by my count)

# Positives

Highly general

C++

Extensive Documentation

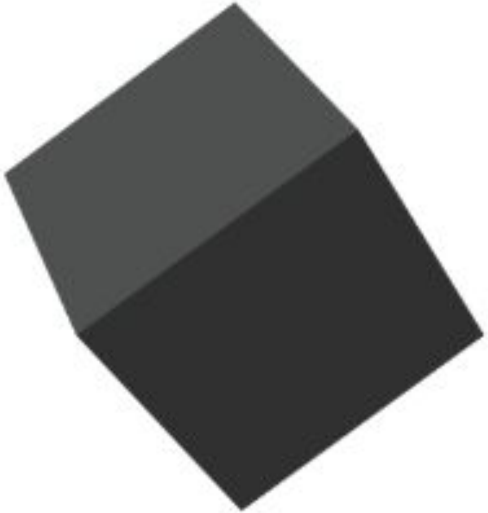
# Negatives

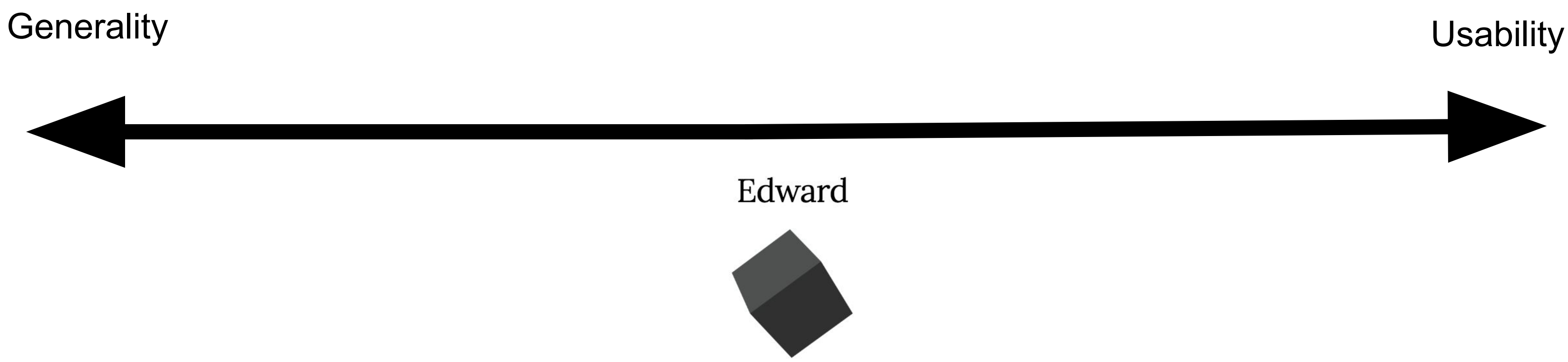
Not backed by a Big 4 company

Highly general

C++

# Edward




Generality
Usability
Edward

“Edward is a Python library for probabilistic modeling, inference, and criticism. It is a testbed for fast experimentation and research with probabilistic models, ranging from classical hierarchical models on small data sets to complex deep probabilistic models on large data sets. Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.”

“Formally, Edward is a Turing-complete probabilistic programming language.”

- Developed at Columbia University. Primary Developer: Dustin Tran
- Open source
- Version 1.3.5
- Still under development (last commit on June 1)
- MIT License
- 77 contributors

# An abstraction over tensorflow

Directed graphical models
Neural networks (via libraries such as tf.layers and Keras)
Implicit generative models
Bayesian nonparametrics and probabilistic programs

# Inference with...

- Variational inference
  - Black box variational inference
  - Stochastic variational inference
  - Generative adversarial networks
  - Maximum a posteriori estimation
- Monte Carlo
  - Gibbs sampling
  - Hamiltonian Monte Carlo
  - Stochastic gradient Langevin dynamics
- Compositions of inference
  - Expectation-Maximization
  - Pseudo-marginal and ABC methods
  - Message passing algorithms

Samlam

Generality

Usability

Samlam

# “Samlam is a comprehensive tool for modeling and reasoning with Bayesian networks”

- Developed at University of California, Los Angeles by the Automated Reasoning Group of Professor Adnan Darwiche.
- Closed source

# Kevin’s notes on SamIam

I took a look at this tool. It’s impressive in the sense that the UI is very well designed and the fact that it’s a Java program means that it can run on any machine with a Java virtual machine implementation, but the project is not open source. I can call into the code, but I can neither see nor edit the code. In my opinion, this is a serious issue. Why *not* host the code on Github? Also, it’s not clear what the licensing is for this software. Can I use it in an industrial/commercial application? All of these factors limit SamIam’s utility, unfortunately.

pgmpy

# Installation...

pip install if you’re on linux

Easy, fast, basically error-proof

# (As an aside…)

There's an R package called bnlearn (http://www.bnlearn.com/)

If you go to http://www.bnlearn.com/bnrepository/ there are Bayesian networks (large and small) to test with!

**Small Networks (<20 nodes)**

| Name | Nodes | Arcs | Parameters |
|---|---|---|---|
| ASIA | 8 | 8 | 18 |
| CANCER | 5 | 4 | 10 |
| EARTHQUAKE | 5 | 4 | 10 |
| SACHS | 11 | 17 | 178 |
| SURVEY | 6 | 6 | 21 |

**Medium Networks (20–50 nodes)**

| Name | Nodes | Arcs | Parameters |
|---|---|---|---|
| ALARM | 37 | 46 | 509 |
| BARLEY | 48 | 84 | 114005 |
| CHILD | 20 | 25 | 230 |
| INSURANCE | 27 | 52 | 984 |
| MILDEW | 35 | 46 | 540150 |
| WATER | 32 | 66 | 10083 |

**Large Networks (50–100 nodes)**

| Name | Nodes | Arcs | Parameters |
|---|---|---|---|
| HAILFINDER | 56 | 66 | 2656 |
| HEPAR II | 70 | 123 | 1453 |
| WIN95PTS | 76 | 112 | 574 |

**Very Large Networks (100–1000 nodes)**

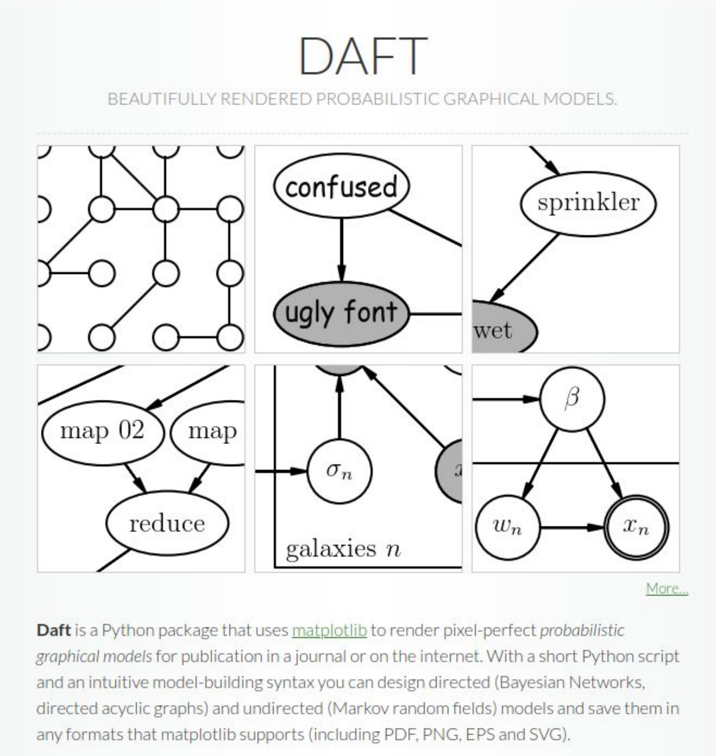
| Name | Nodes | Arcs | Parameters |
|---|---|---|---|
| ANDES | 223 | 338 | 1157 |
| DIABETES | 413 | 602 | 429409 |
| LINK | 724 | 1125 | 14211 |
| MUNIN (4 subnetworks) | 186–1041 | 273–1388 | 15622–80352 |
| PATHFINDER | 135 | 200 | 77155 |
| PIGS | 441 | 592 | 5618 |

**Massive Networks (>1000 nodes)**

| Name | Nodes | Arcs | Parameters |
|---|---|---|---|
| MUNIN (full network) | 1041 | 1397 | 80592 |
| MUNIN (4 subnetworks) | 186–1041 | 273–1388 | 15622–80352 |

# (As *another* aside…)

daft-pgm.org

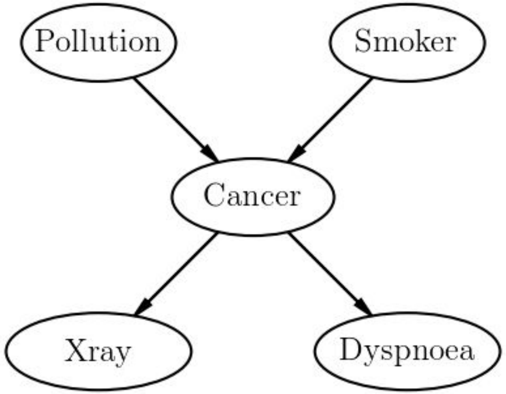


```python
from matplotlib import rc
rc("font", family="serif", size=12)
rc("text", usetex=True)

import daft

pgm = daft. PGM([3.6, 2.7], origin=[1.15, 0.65])
pgm.add_node(daft.Node("Pollution", r"Pollution", 2, 3, aspect=2))
pgm.add_node(daft.Node("Smoker", r"Smoker", 4, 3, aspect=2))
pgm.add_node(daft.Node("Cancer", r"Cancer", 3, 2, aspect=2.1))
pgm.add_node(daft.Node("Xray", r"Xray", 2, 1, aspect=2.4))
pgm.add_node(daft.Node("Dyspnoea", r"Dyspnoea", 4, 1, aspect=2.4))
pgm.add_edge("Pollution", "Cancer")
pgm.add_edge("Smoker", "Cancer")
pgm.add_edge("Cancer", "Xray")
pgm.add_edge("Cancer", "Dyspnoea")
pgm.render()
pgm.figure.savefig("wordy.pdf")
pgm.figure.savefig("wordy.png", dpi=150)

```



# Back to pgmpy...

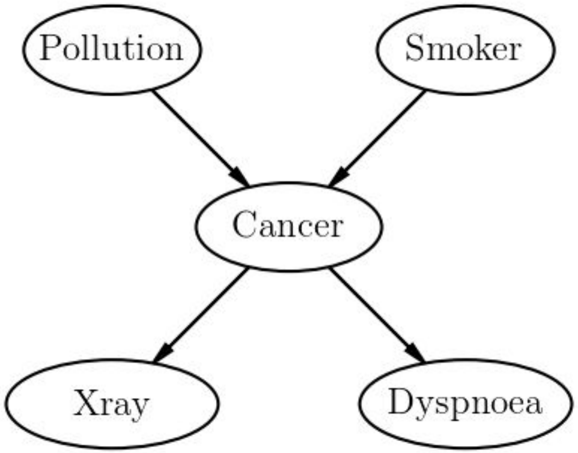
```python
# Starting with defining the network structure
from pgmpy.models import BayesianModel

cancer_model = BayesianModel([('Pollution', 'Cancer'),
                              ('Smoker', 'Cancer'),
                              ('Cancer', 'Xray'),
                              ('Cancer', 'Dyspnoea')])
```

```python
from pgmpy.factors.discrete import TabularCPD
cpd_poll = TabularCPD(variable='Pollution', variable_card=2,
                      values=[[0.9], [0.1]])
cpd_smoke = TabularCPD(variable='Smoker', variable_card=2,
                       values=[[0.3], [0.7]])
cpd_cancer = TabularCPD(variable='Cancer', variable_card=2,
                        values=[[0.03, 0.05, 0.001, 0.02],
                                [0.97, 0.95, 0.999, 0.98]],
                        evidence=['Smoker', 'Pollution'],
                        evidence_card=[2, 2])
cpd_xray = TabularCPD(variable='Xray', variable_card=2,
                      values=[[0.9, 0.2], [0.1, 0.8]],
                      evidence=['Cancer'], evidence_card=[2])
cpd_dysp = TabularCPD(variable='Dyspnoea', variable_card=2,
                      values=[[0.65, 0.3], [0.35, 0.7]],
                      evidence=['Cancer'], evidence_card=[2])
```

```
# Associating the parameters with the model structure.
cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)
# Checking if the cpds are valid for the model.
cancer_model.check_model()
assert(cancer_model.is_active_trail('Pollution', 'Smoker') == False)
assert(cancer_model.is_active_trail('Pollution', 'Smoker', observed=['Cancer']) == True)
print(cancer_model.get_independencies())
```

```python
print(cancer_model.get_independencies())
```

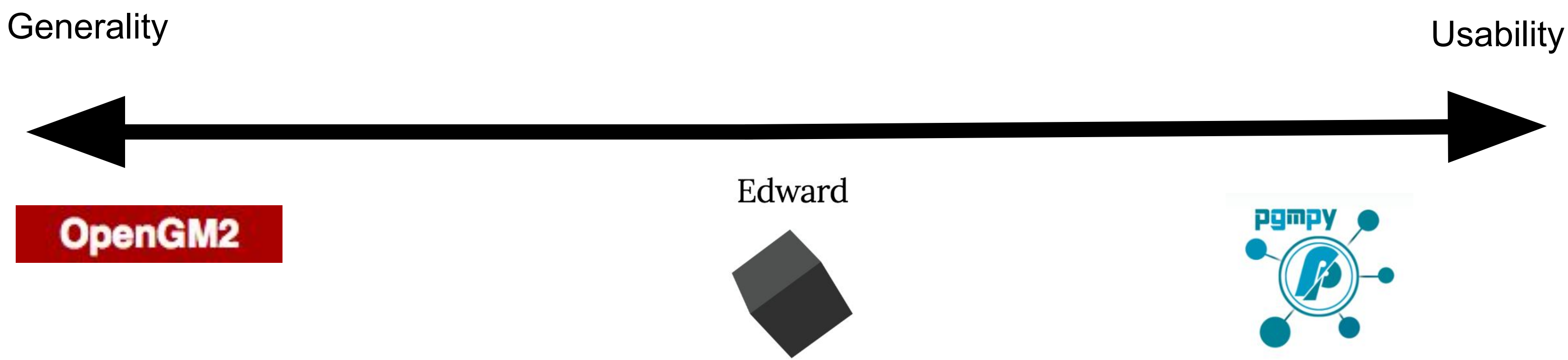



```
(Xray _|_ Dyspnoea, Smoker, Pollution | Cancer)
(Xray _|_ Smoker, Pollution | Dyspnoea, Cancer)
(Xray _|_ Dyspnoea, Pollution | Smoker, Cancer)
(Xray _|_ Dyspnoea, Smoker | Cancer, Pollution)
(Xray _|_ Pollution | Dyspnoea, Smoker, Cancer)
(Xray _|_ Smoker | Dyspnoea, Cancer, Pollution)
(Xray _|_ Dyspnoea | Smoker, Cancer, Pollution)
(Dyspnoea _|_ Xray, Smoker, Pollution | Cancer)
(Dyspnoea _|_ Smoker, Pollution | Xray, Cancer)
(Dyspnoea _|_ Xray, Pollution | Smoker, Cancer)
(Dyspnoea _|_ Xray, Smoker | Cancer, Pollution)
(Dyspnoea _|_ Pollution | Xray, Smoker, Cancer)
(Dyspnoea _|_ Smoker | Xray, Cancer, Pollution)
(Dyspnoea _|_ Xray | Smoker, Cancer, Pollution)
(Smoker _|_ Pollution)
(Smoker _|_ Xray, Dyspnoea | Cancer)
(Smoker _|_ Dyspnoea | Xray, Cancer)
(Smoker _|_ Xray | Dyspnoea, Cancer)
(Smoker _|_ Xray, Dyspnoea | Cancer, Pollution)
(Smoker _|_ Dyspnoea | Xray, Cancer, Pollution)
(Smoker _|_ Xray | Dyspnoea, Cancer, Pollution)
(Pollution _|_ Smoker)
(Pollution _|_ Xray, Dyspnoea | Cancer)
(Pollution _|_ Dyspnoea | Xray, Cancer)
(Pollution _|_ Xray | Dyspnoea, Cancer)
(Pollution _|_ Xray, Dyspnoea | Smoker, Cancer)
(Pollution _|_ Dyspnoea | Xray, Smoker, Cancer)
(Pollution _|_ Xray | Dyspnoea, Smoker, Cancer)
```

```python
# Doing exact inference using Variable Elimination
from pgmpy.inference import VariableElimination
cancer_infer = VariableElimination(cancer_model)
q = cancer_infer.query(variables=['Cancer'], evidence={'Pollution': 0, 'Smoker': 0})
print(q['Cancer'])
```

```
+----------+---------------+
| Cancer   |   phi(Cancer) |
|----------+---------------|
| Cancer_0 |        0.0300 |
| Cancer_1 |        0.9700 |
+----------+---------------+
```

```python
print(cancer_infer.induced_width(['Pollution', 'Smoker', 'Cancer', 'Xray', 'Dyspnoea']))
```


Generality
Usability
OpenGM2
Edward
pgmpy

I hope this was helpful, interesting, or provided some ideas about potential future work.

Thank you!

Questions?