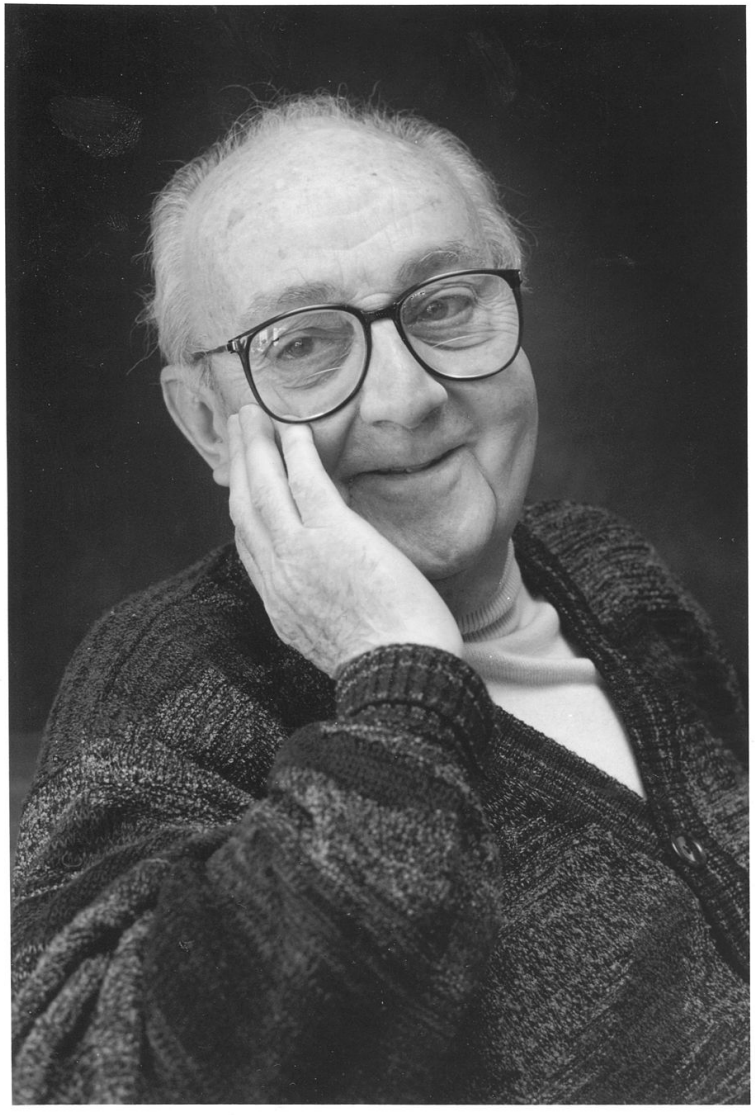
Edward: A library for probabilistic modeling, inference, and criticism

Dustin Tran, David M. Blei
Columbia University

Matt Hoffman, Rif A. Saurous, Eugene Brevdo,
Kevin Murphy
Google Brain

edwardlib.org

George E.P. Box (1919 - 2013)

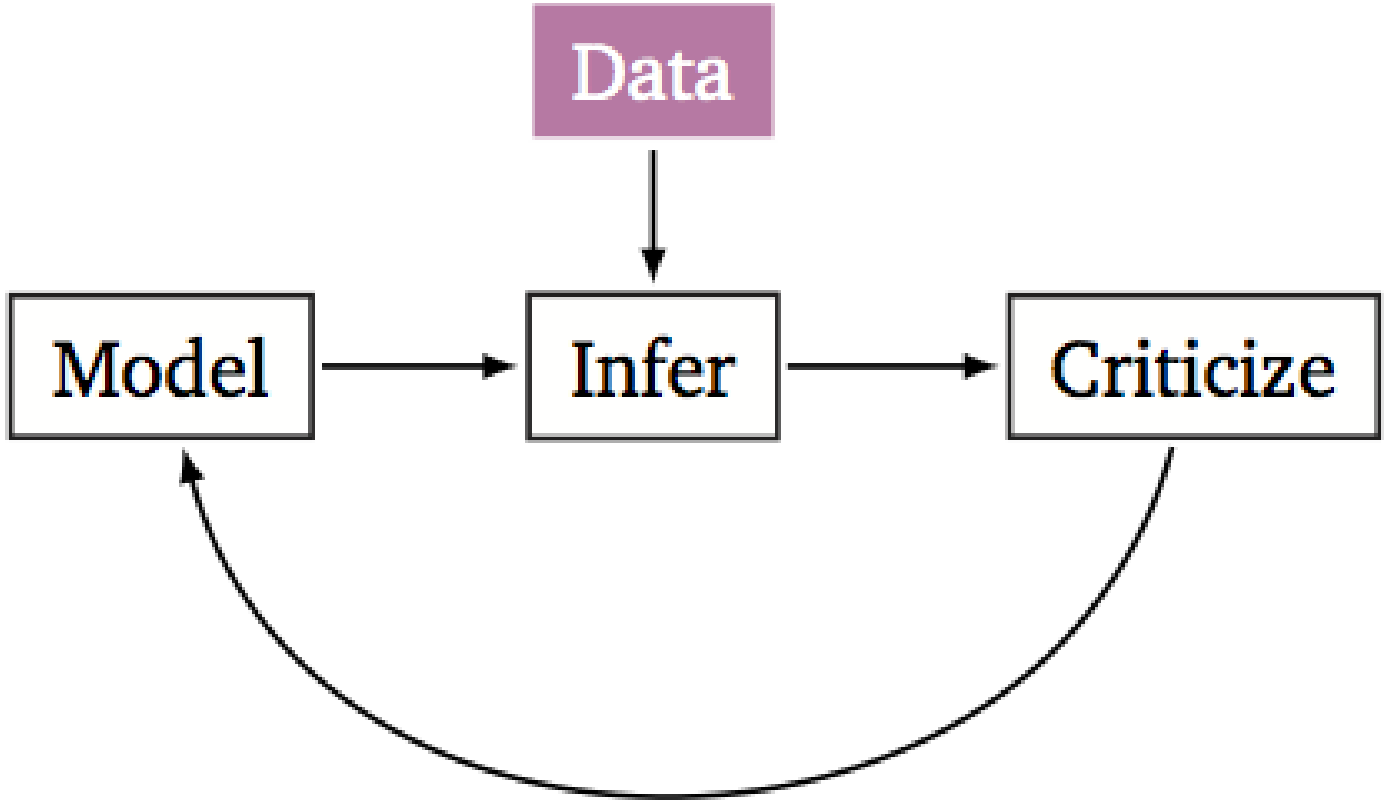


An iterative process for science:

1. Build a model of the science
2. Infer the model given data
3. Criticize the model given data

(Box & Hunter 1962, 1965; Box & Hill 1967; Box 1976, 1980)

Box's Loop



Edward is a library designed around this loop.

(Box, 1976; Box, 1980; Blei, 2014)

Edward is a probabilistic programming language, designed for fast experimentation and research.

Modeling

- Composable Turing-complete language of random variables.
- Examples: Graphical models, neural networks, probabilistic programs.
- Many data types, tensor vectorization, broadcasting, 3rd party support.

Inference

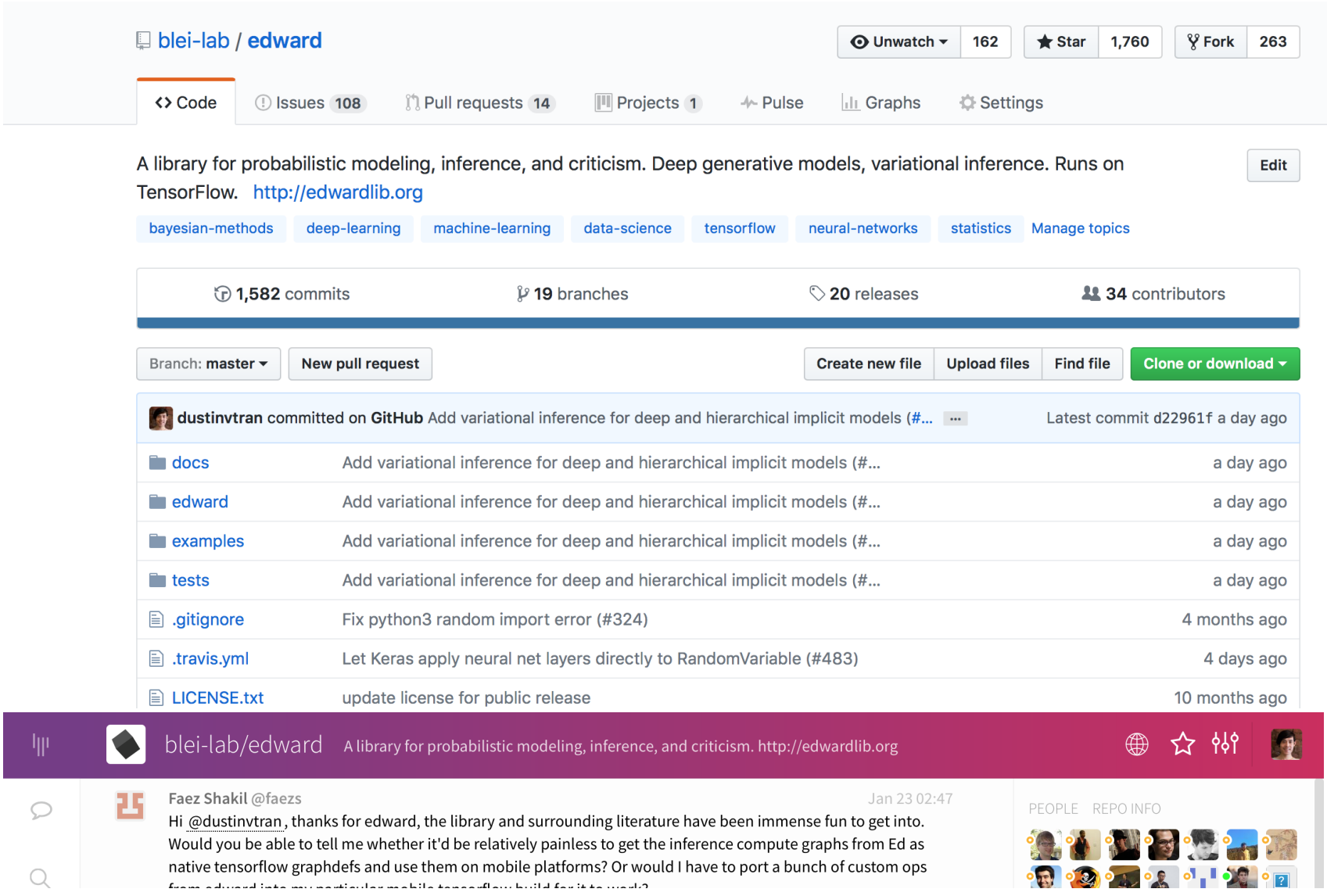
- Composable language for hybrids, message passing, data subsampling.
- Examples: Black box VI, Hamiltonian MC, stochastic gradient MCMC.
- Infrastructure to develop your own algorithms.

Criticism

- Examples: Scoring rules, hypothesis tests, predictive checks.

Built on TensorFlow (features distributed computing, GPUs, autodiff).

(Tran et al., 2016)



We have an active community of several hundred users. We have many few-commit developers.

Who is Using Edward?

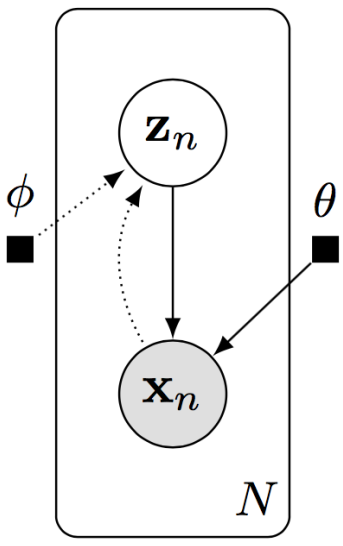
Users

1. Machine learning enthusiasts, data scientists, business analysts (*ex. hierarchical GLMs, mixture models, MAP, MCMC, ...*)
2. Probabilistic graphical modeling community (*ex. latent Dirichlet allocation, variational inference, Gibbs*)
3. Bayesian deep learning community (*ex. deep generative models, Bayesian NNs, black box inference*)

Developers

1. David Blei's group
2. Google Brain (*in conception/design*)
3. Matt Hoffman (*conjugacy*), Emily Fox's group (*time series + SGMCMC*), Justin Bayer (*stochastic RNNs*), John Pearson (*neuroscience*), a few Master's/Ph.D. students.
4. Everyone is part-time. Collaboration continues to evolve.

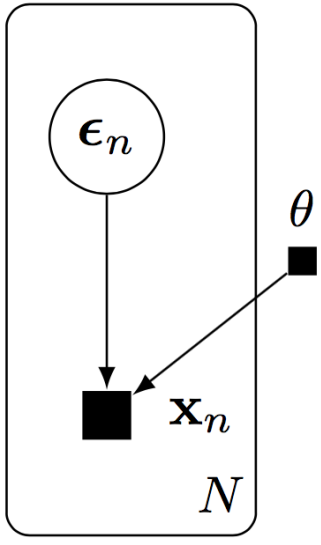
Variational Auto-Encoder




```
# Probabilistic model
z = Normal(mu=tf.zeros([N, d]), sigma=tf.ones([N, d]))
h = Dense(256, activation='relu')(z)
x = Bernoulli(logits=Dense(28 * 28, activation=None)(h))

# Variational model
qx = tf.placeholder(tf.float32, [N, 28 * 28])
qh = Dense(256, activation='relu')(qx)
qz = Normal(mu=Dense(d, activation=None)(qh),
            sigma=Dense(d, activation='softplus')(qh))
```

Generative Adversarial Network




```
def generative_network(eps):
  h = Dense(256, activation='relu')(eps)
  return Dense(28 * 28, activation=None)(h)

def discriminative_network(x):
  h = Dense(28 * 28, activation='relu')(x)
  return Dense(h, activation=None)(1)

# Probabilistic model
eps = Normal(mu=tf.zeros([M, d]), sigma=tf.ones([M, d]))
x = generative_network(eps)

inference = ed.GANInference(data={x: x_train},
    discriminator=discriminative_network)
```