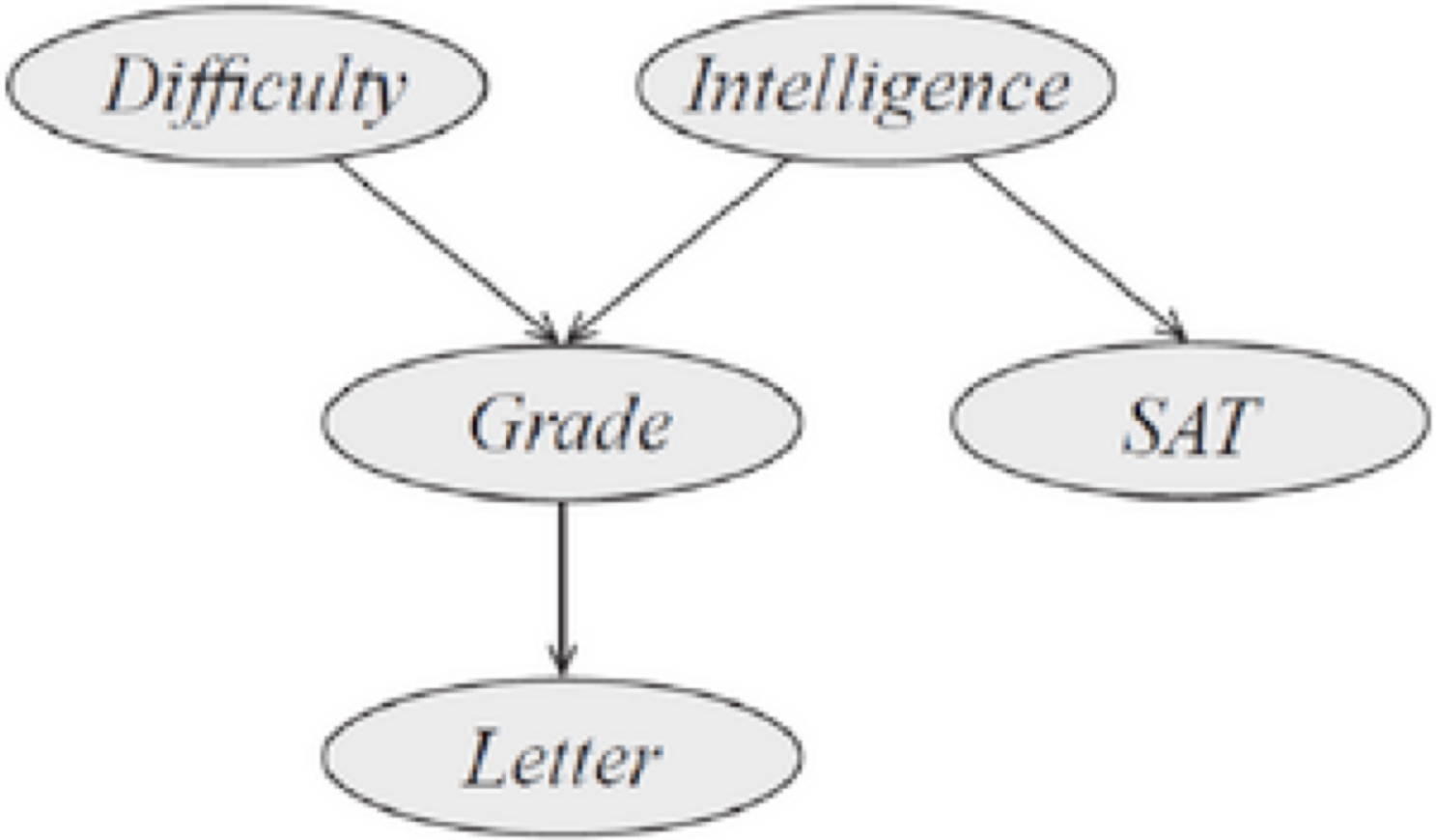
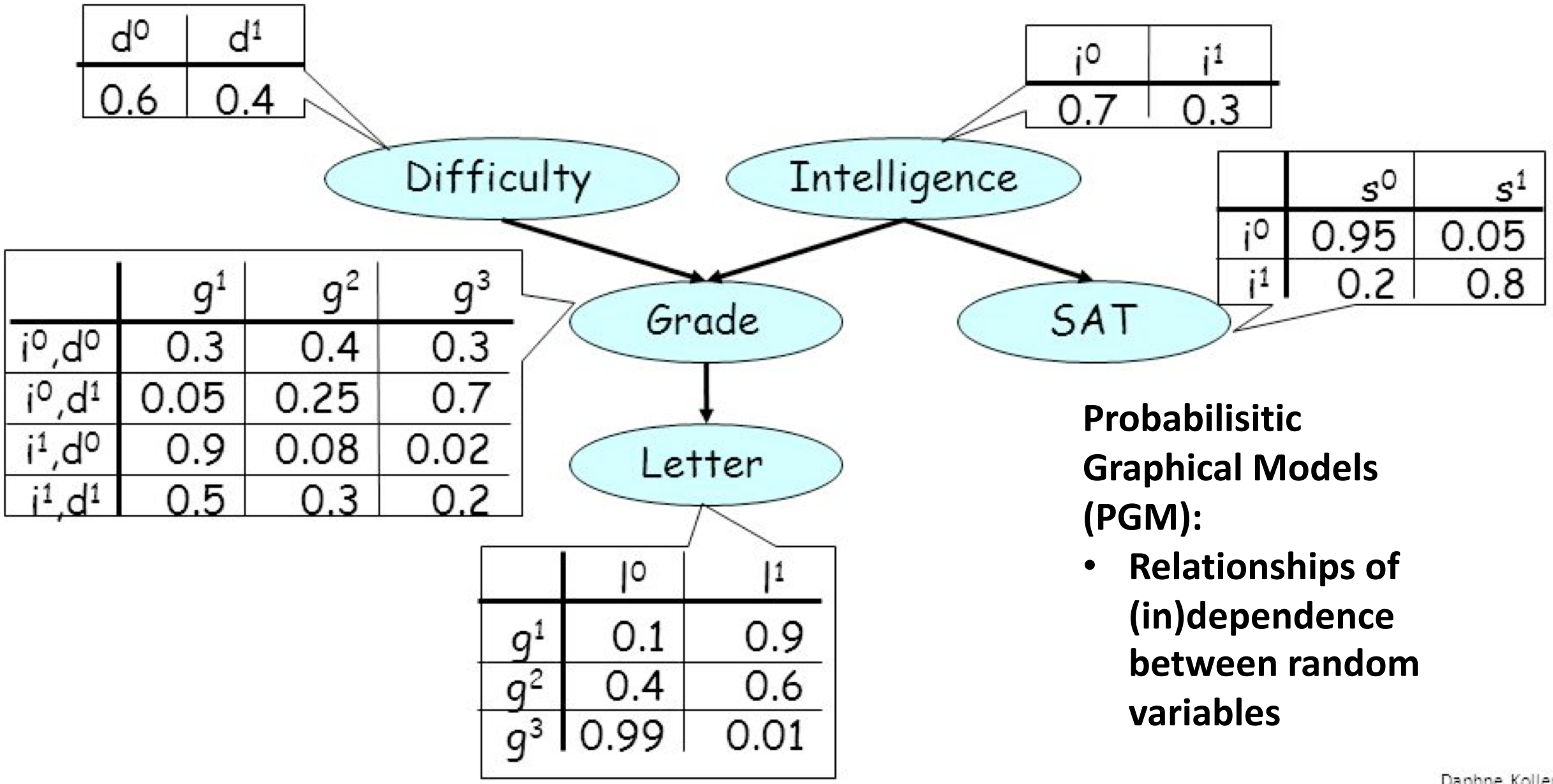
Probabilistic Models in PyMC3



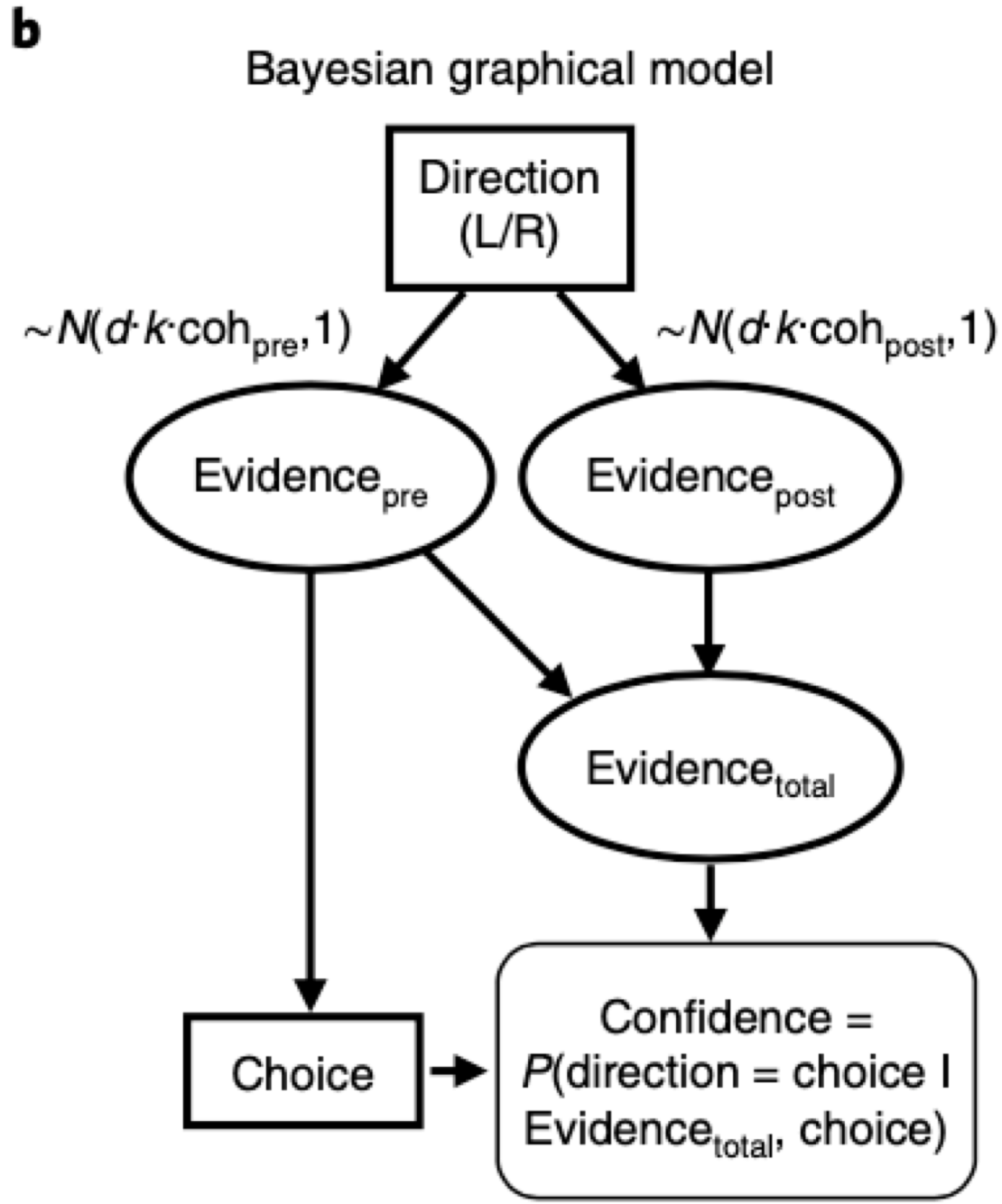
- Will the professor give the student a recommendation letter?

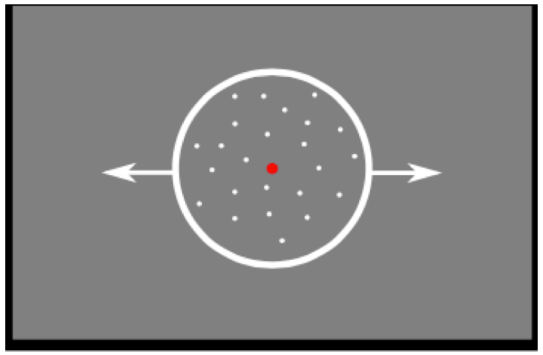


Probabilisitic Graphical Models (PGM):

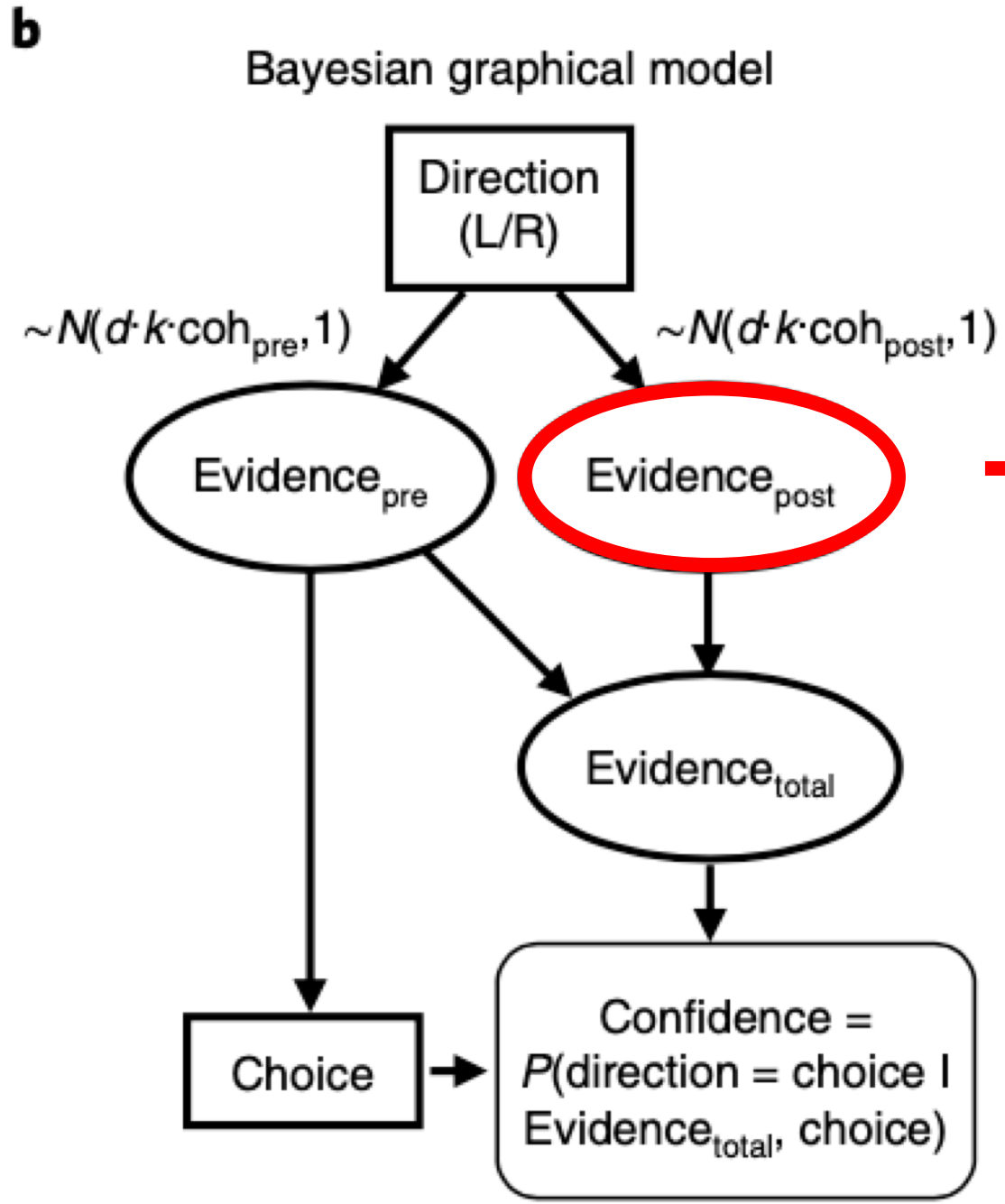
- **Relationships of (in)dependence between random variables**

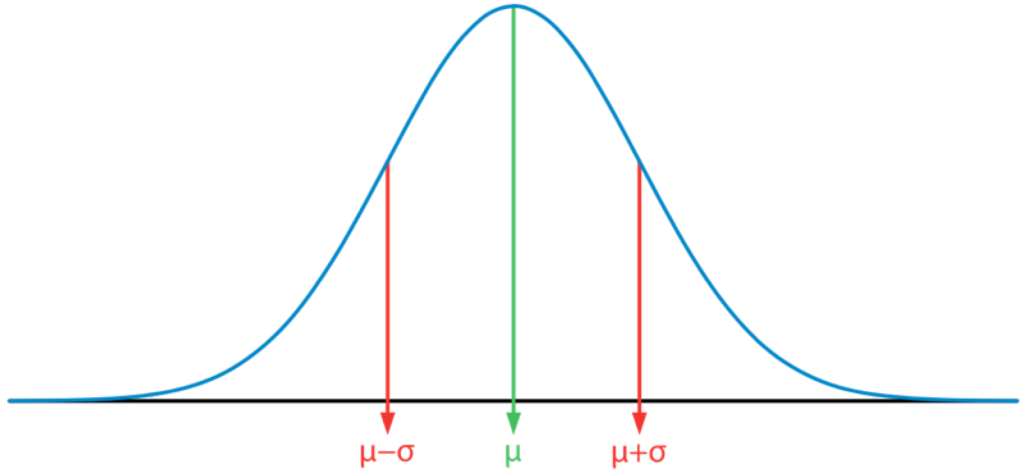
Daphne Koller





(Fleming et al., 2018)



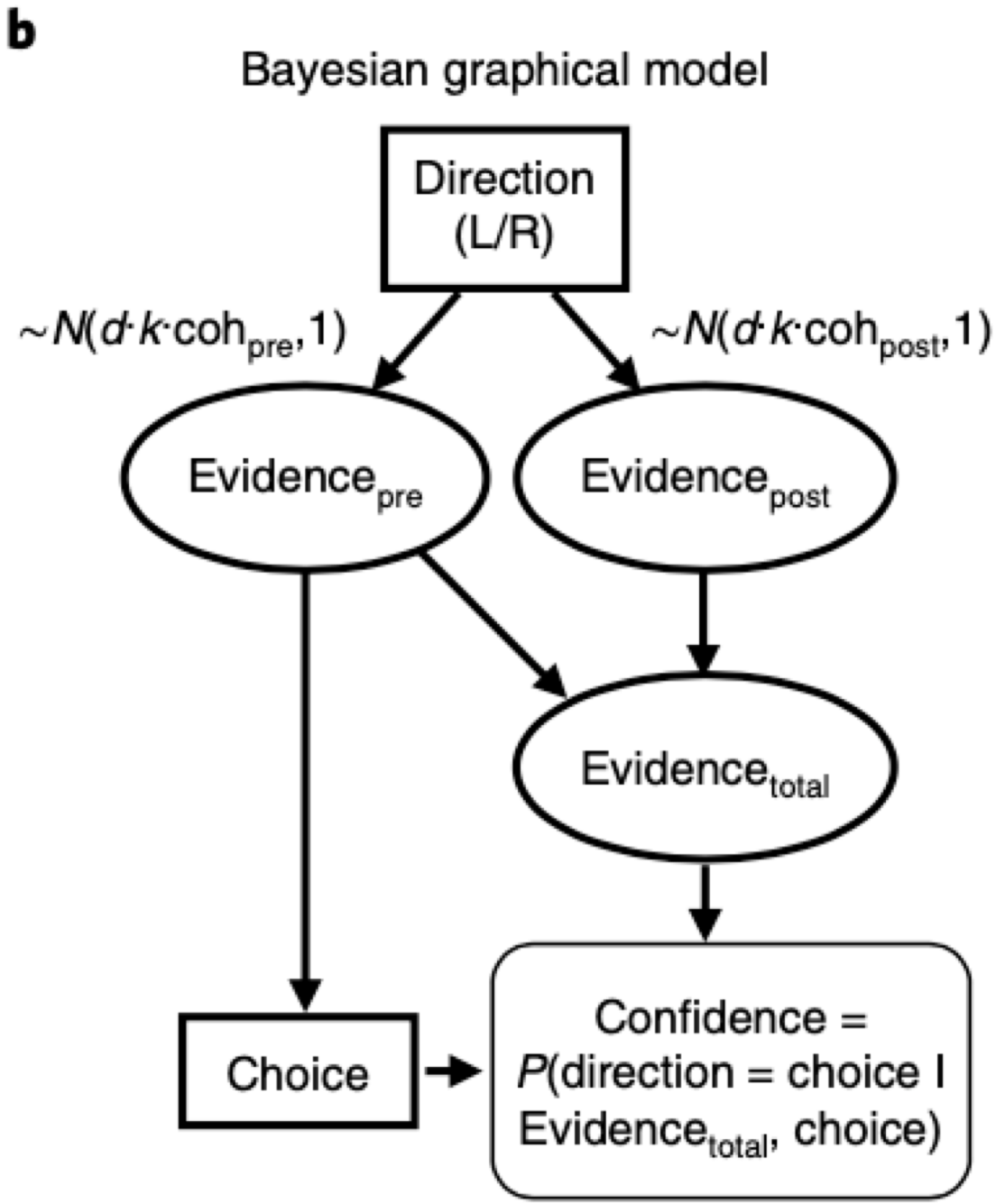


Variables

E ~N(μ = d*k*coh , σ= 1)

Model parameters

(Fleming et al., 2018)



Latent variables

?

$$m \sim N(0, 10)$$

$$k \sim N(0, 10)$$

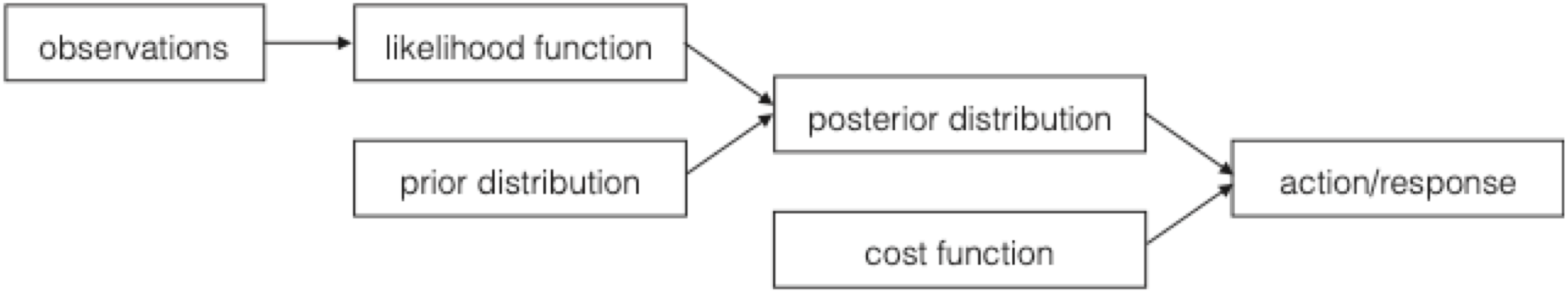
[Evidence$_{pre}$] $X_{pre} \sim N(dk\theta_{pre}, 1)$

[Evidence$_{post}$] $X_{post} \sim N(dk\theta_{post}, 1)$

[Choice] $a \sim \text{Bernoulli\_logit}(100(X_{pre} - m))$

$$r \sim N(\text{conf}, 0.025)$$

Bayesian Decision Models



Bayesian Decision Making Schematic

$$p(s \mid x_{\text{trial}}) = \frac{p(x_{\text{trial}} \mid s)p(s)}{p(x_{\text{trial}})}.$$

s: state of the world
x: observation by the agent

Ma et al., 2019

Bayesian Inference of model parameters

likelihood
probability distribution of the observed data given a parameter value

(how probable are the observed data for this parameter value?)

prior
probability distribution of the parameter independantly from any observation

(prior knowledge: how probable are each value of the parameter before any observation?)

$$p(\theta \mid x) = \frac{p(x \mid \theta)\, p(\theta)}{p(x)}$$

posterior
probability distribution of the parameter given the observed data

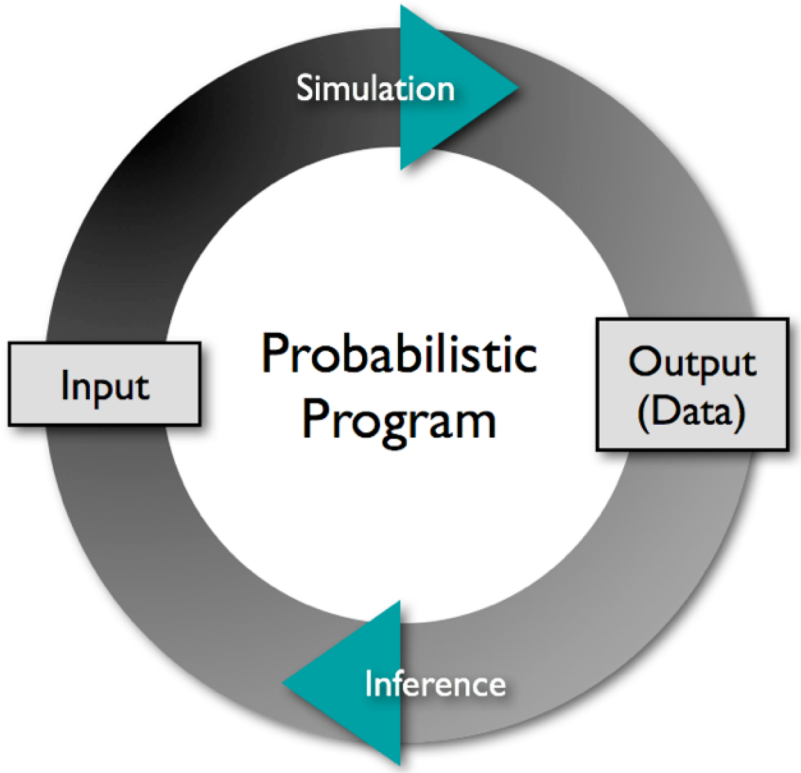
(updated knowledge: how probable are each value of the parameter given the observed data?)

evidence
probability distribution of the observed data independantly from any parameter value

(how probable is it to observe these particular data?)

Probabilistic Programming

- A probabilistic program is run in both the forward and backward direction. It runs **forward to compute the consequences of the assumptions it contains about the world** (i.e., the model space it represents), but it also **runs backward from the data to constrain the possible explanations**.
- Probabilistic programming languages (PPLs) are a tool designed specifically for doing inference.



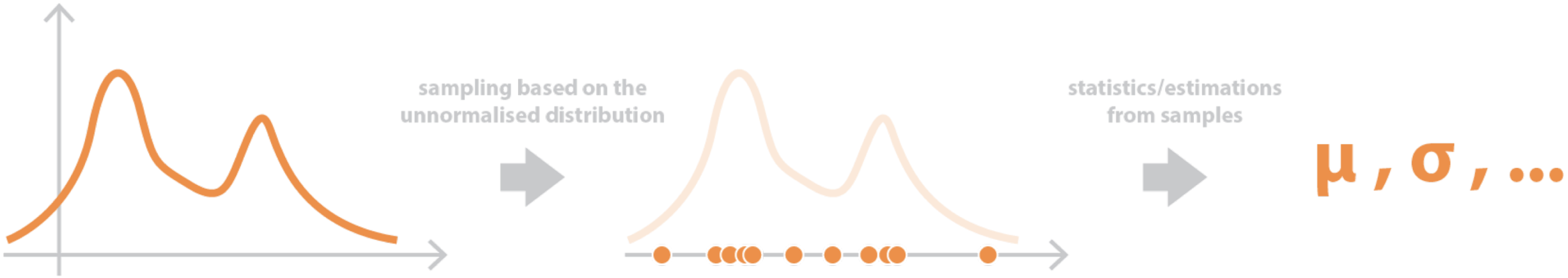
What does Probabilistic Programming involve?

- **Sampling**: Our model is probabilistic- it requires drawing values at random from probability distributions.
- **Conditioning**: We have some observed data that can be used to update probabilities
- **Inference**: We can learn something from the known data and model. This could be the parameters of the system or a “latent variable”, some underlying factor that you can’t directly observe but might influence the data. This is usually the hard part.

(Mansfield, 2019)

How can we fit the model? MCMC

- $p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \propto p(x|\theta)p(\theta)$
- P(x) difficult to obtain in more realistic models.



Unnormalised distribution
whose normalisation factor computation is intractable

Samples
that can be obtained with MCMC and without proceeding to the normalisation

Statistics or estimations
that can be computed based on the generated samples

Markov chain Monte Carlo

Monte Carlo: Simulations evolving randomly

the more samples we have

Law of Large numbers (average → expected value)

Markov chain:

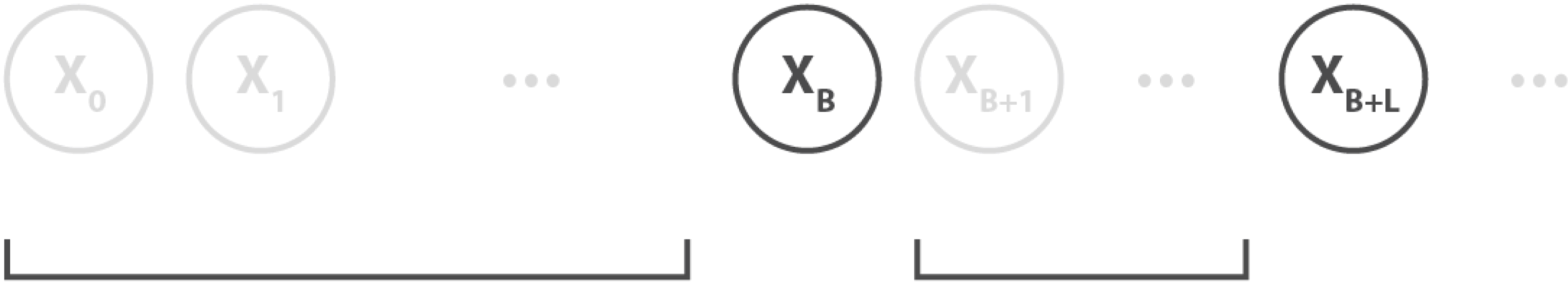
$$P(\text{future} \mid \text{present}, \text{past}) = P(\text{future} \mid \text{present}, [illegible])$$

Markov property

Next sample depends of the present sample
Not fully random sampling → allows to look for high density areas of the target distr.

Algorithms for MCMC

1. Start at current position.
2. Propose moving to a new position
3. Accept/Reject the new position based on the position's adherence to the data and prior distributions.
    1. If you accept: Move to the new position. Return to Step 1.
    2. Else: Do not move to new position. Return to Step 1.
4. After a large number of iterations, return all accepted positions.

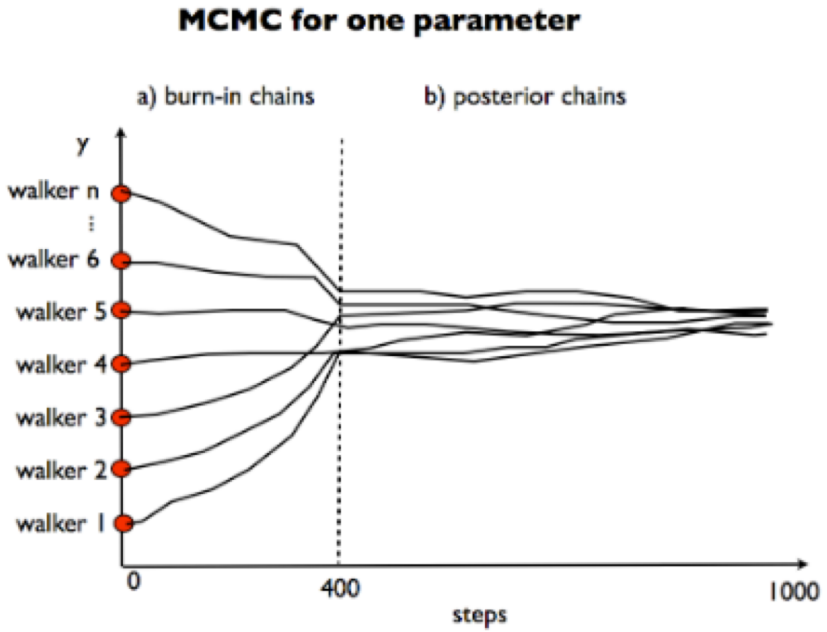


Burn-in time

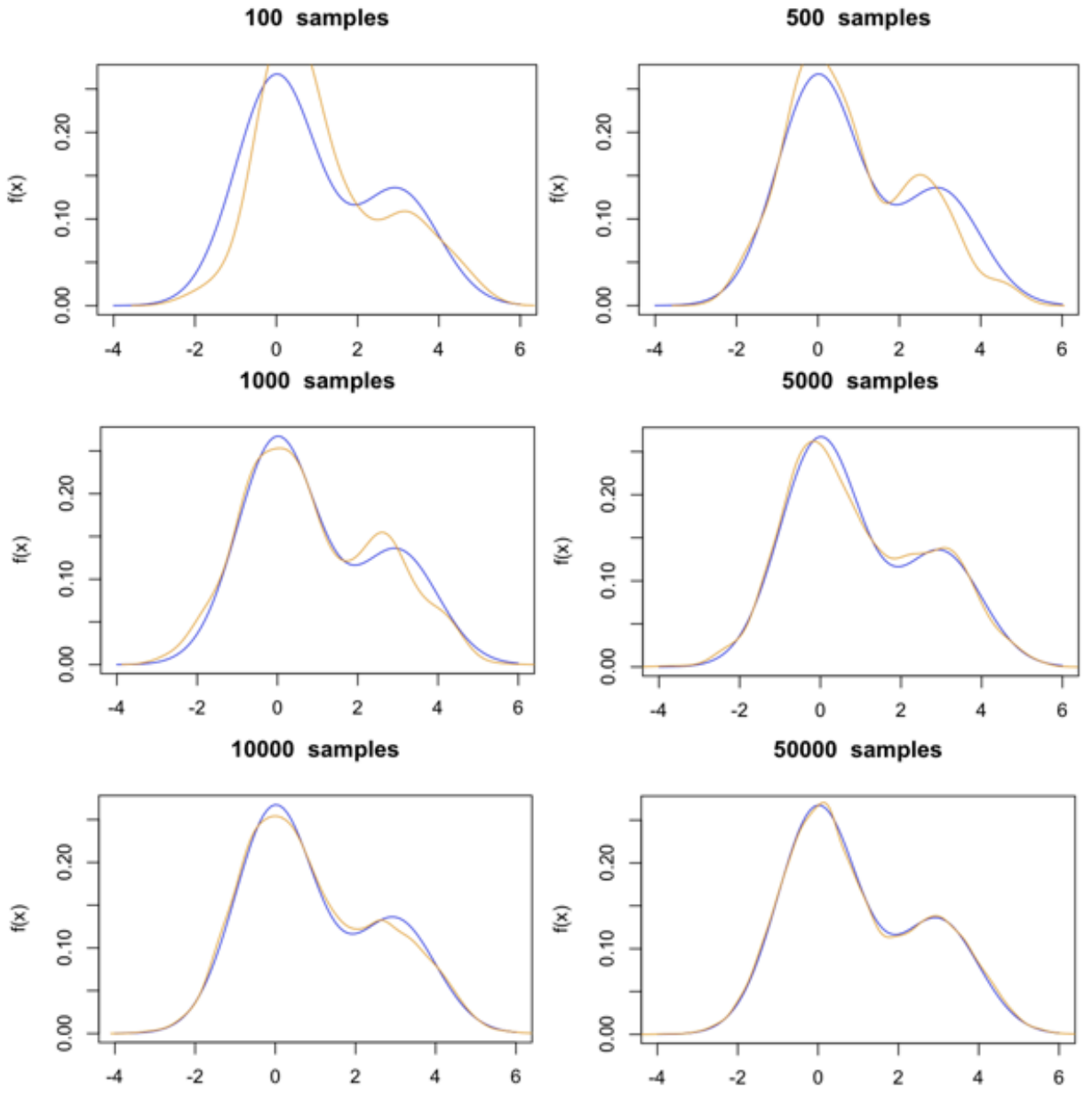
The chain is not considered to have reached the steady state yet and, so, these states do not follow the target probability distribution

Lag

These states are too correlated with X_B, so they can't be kept as we want to generate (almost) independent samples



https://events.mpifr-bonn.mpg.de/indico/event/30/material/slides/12.pdf



Convergence of the Metropolis–Hastings algorithm. Markov chain Monte Carlo attempts to approximate the blue distribution with the orange distribution.

Probabilistic Programming in Python (PyMC3)

- Probabilistic Programming allows for automatic Bayesian inference on user-defined probabilistic models.

Probabilistic Programming basic steps (PyMC3)

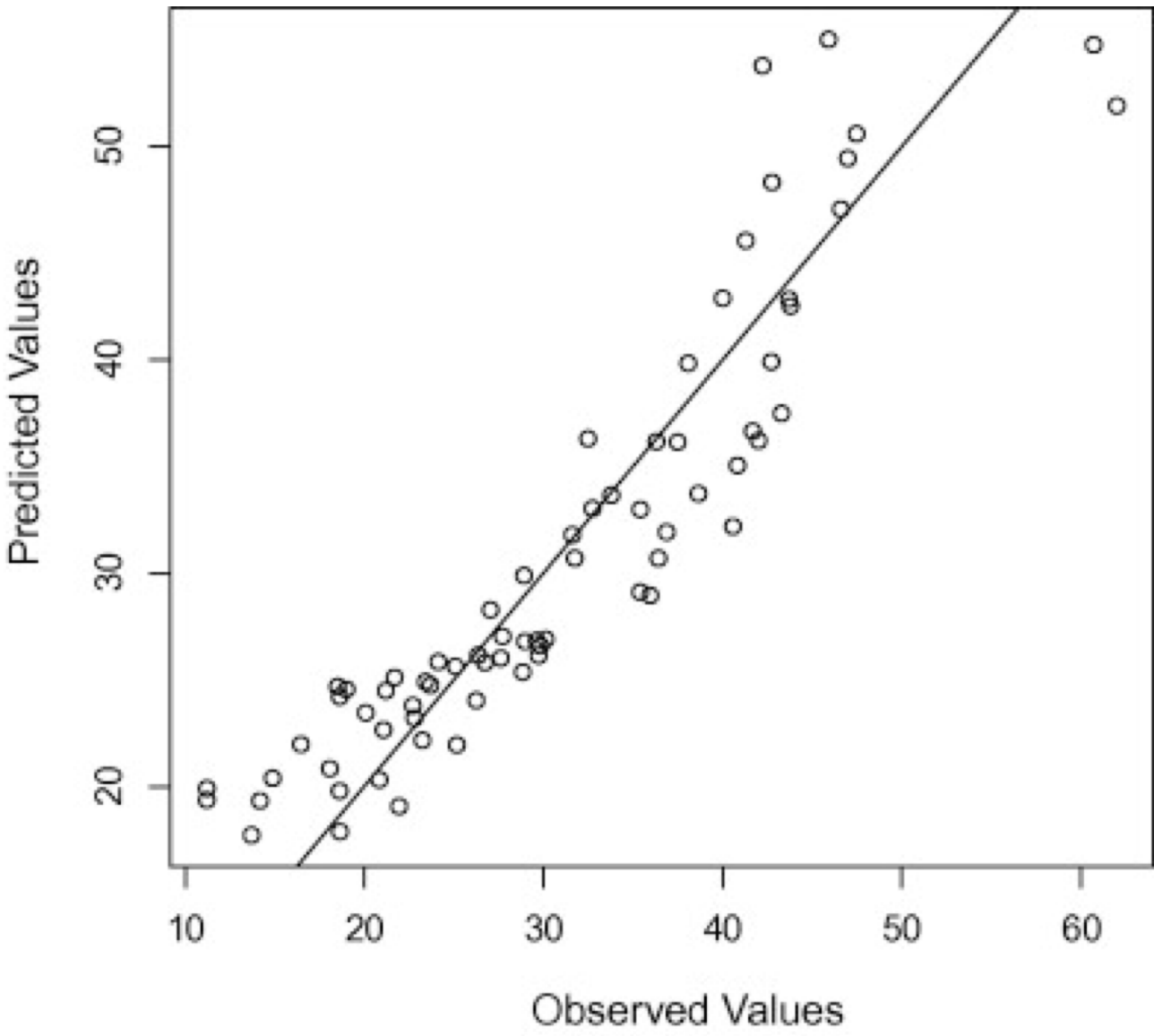
- 1. Generating Data
- 2. Model Specification
- 3. Model fitting
- 4. Posterior Analysis

1st test: Let's model a GLM

Y = m*X + intercept + s

m: slope for x

s: error (std dev, σ)



Tutorial use data from decision making exp

Like

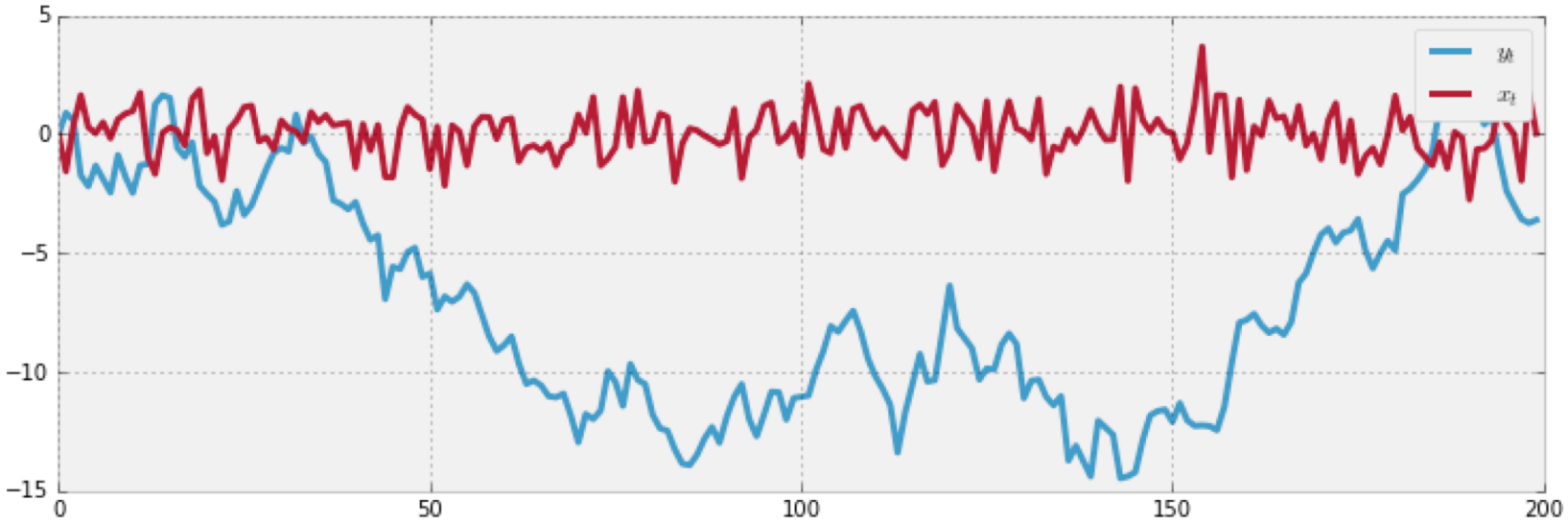
Diagnosing Convergence

- Gelman-Rubin diagnostic. ($|\hat{R} - 1| < 0.05$)
- Effective sample size (ESS>100)
- Autocorrelation
- Check the traces

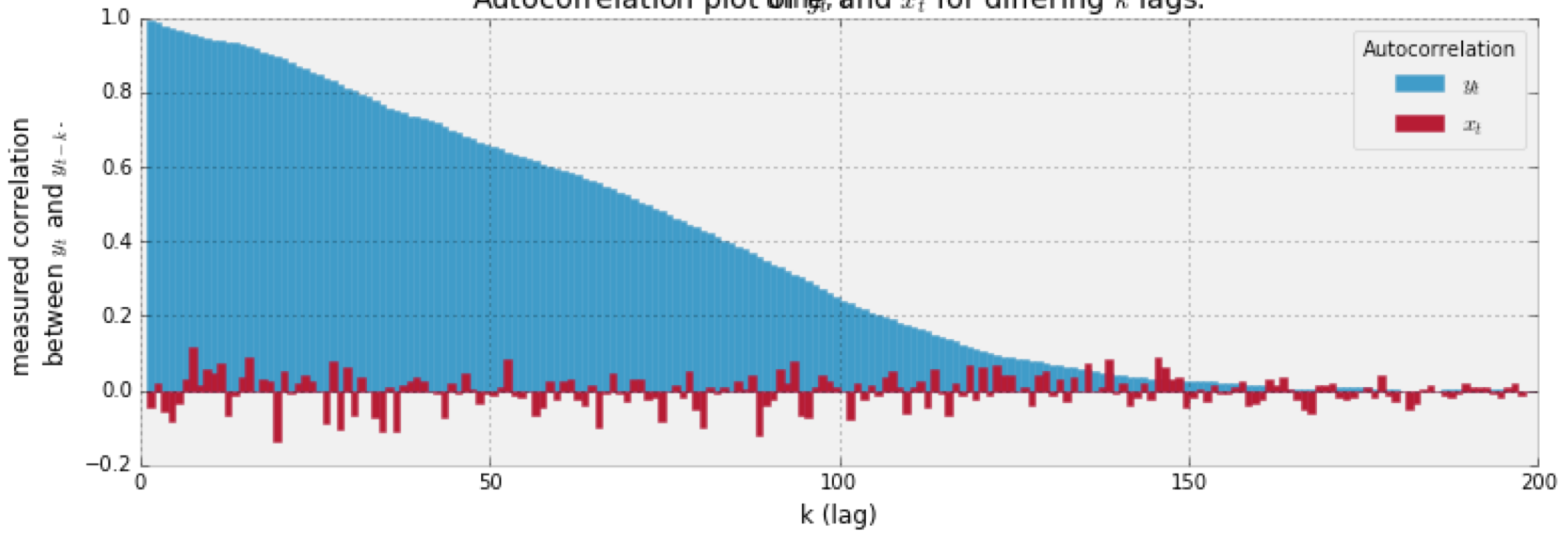
- https://pymc-devs.github.io/pymc/modelchecking.html
- https://pymc3.readthedocs.io/en/latest/api/diagnostics.html

Autocorrelation

$$y_t \sim \text{Normal}(y_{t-1}, 1), \quad y_0 = 0$$

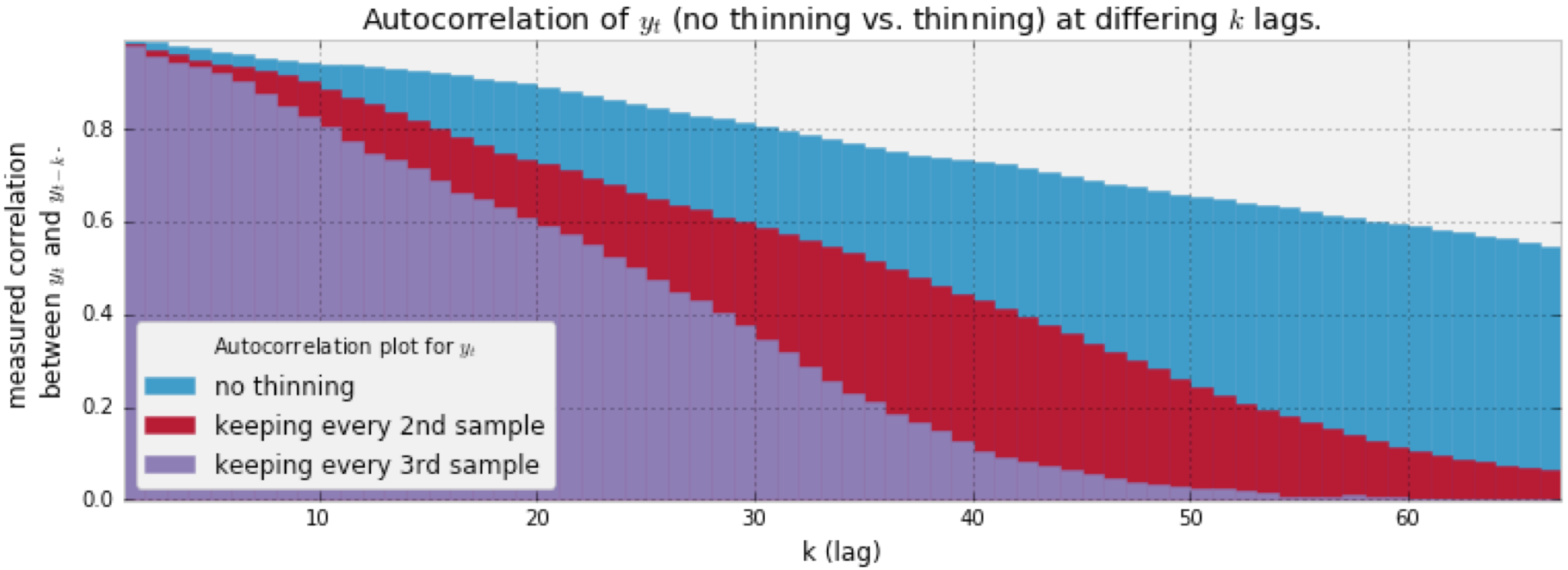


Autocorrelation plot of y_t and x_t for differing k lags.



(https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/blob/master/Chapter3_MCMC/Ch3_IntroMCMC_PyMC3.ipynb)

Thining to decrease autocorr



Refs

https://towardsdatascience.com/bayesian-inference-problem-mcmc-and-variational-inference-25a8aa9bce29

https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/blob/master/Chapter3_MCMC/Ch3_IntroMCMC_PyMC3.ipynb

https://docs.pymc.io/notebooks/getting_started.html

https://www.youtube.com/watch?v=yCv2N7wGDCw&t=10s

Fleming, S. M., Van Der Putten, E. J., & Daw, N. D. (2018). Neural mediators of changes of mind about perceptual decisions. *Nature neuroscience*, *21*(4), 617-624.

Mansfield (2019). **Probabilistic Programming.** https://medium.com/informatics-lab/probabilistic-programming-1535d7882dbe

Ma, W. J. (2019). Bayesian decision models: a primer. *Neuron*, *104*(1), 164-175.