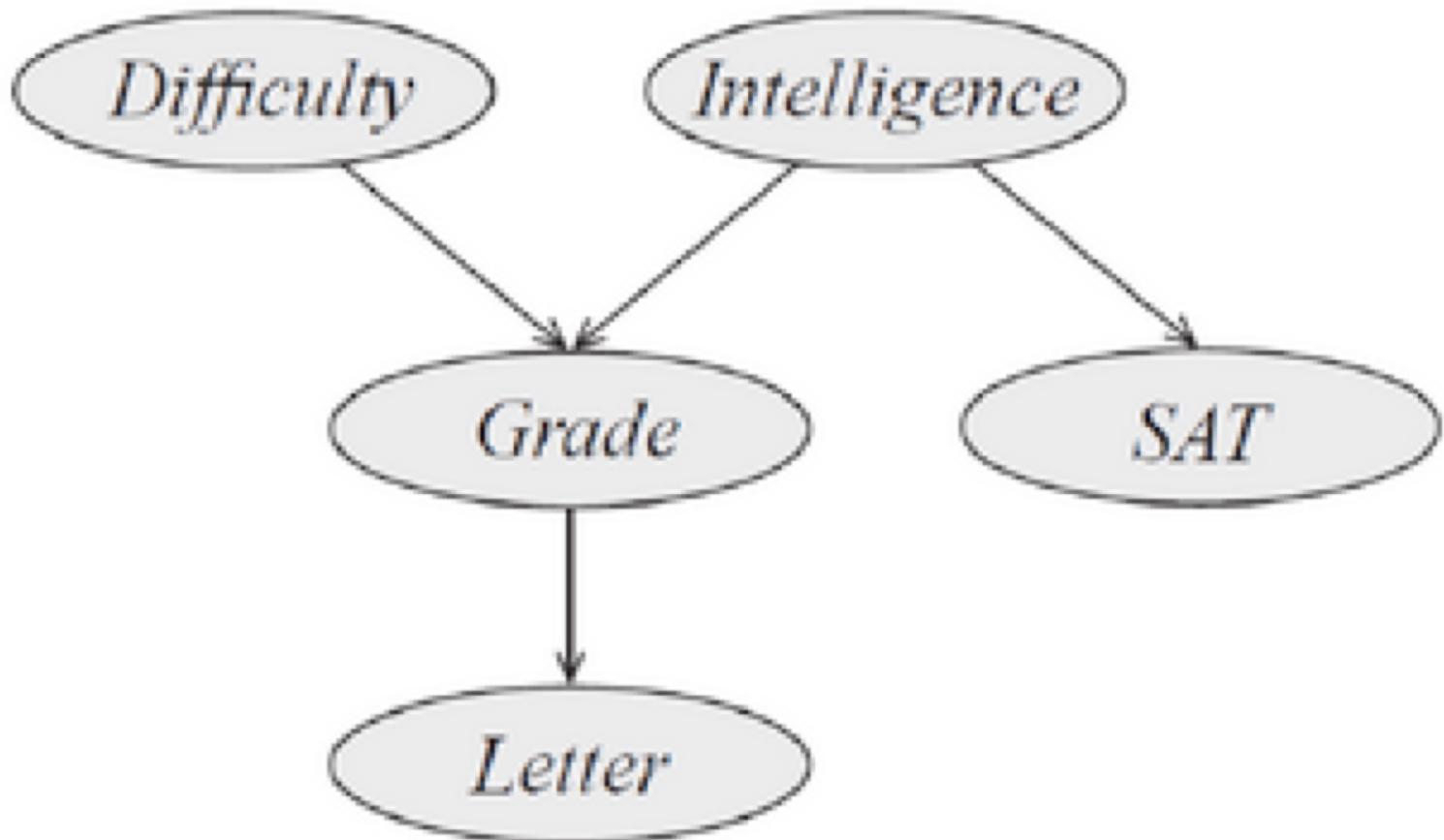
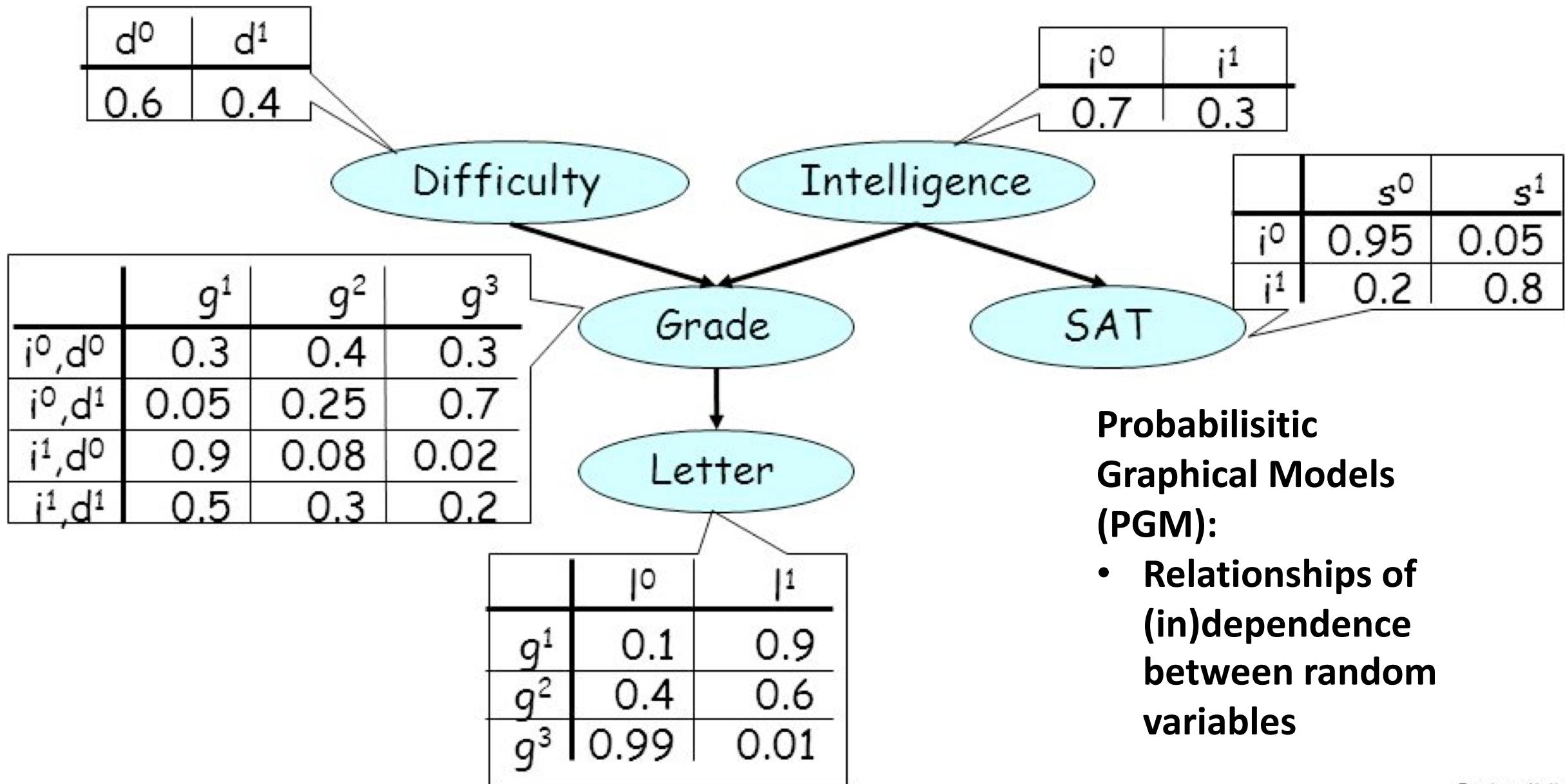


Probabilistic Models in PyMC3

June 2021



- Will the professor give the student a recommendation letter?

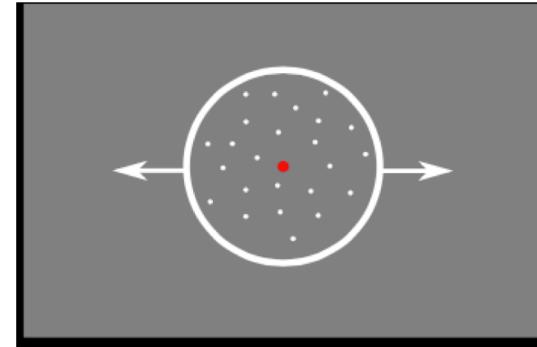
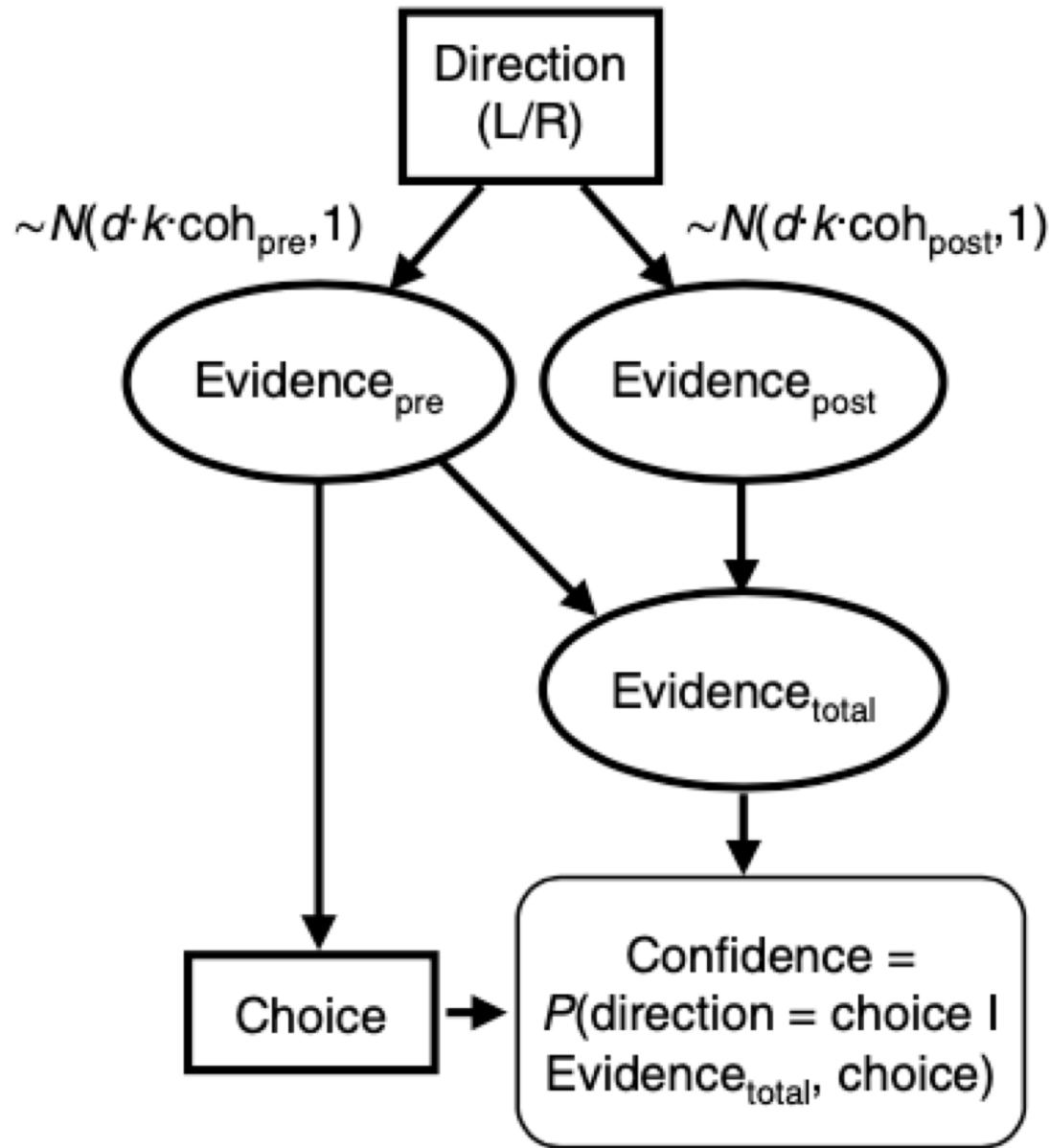


Probabilistic Graphical Models (PGM):

- Relationships of (in)dependence between random variables

b

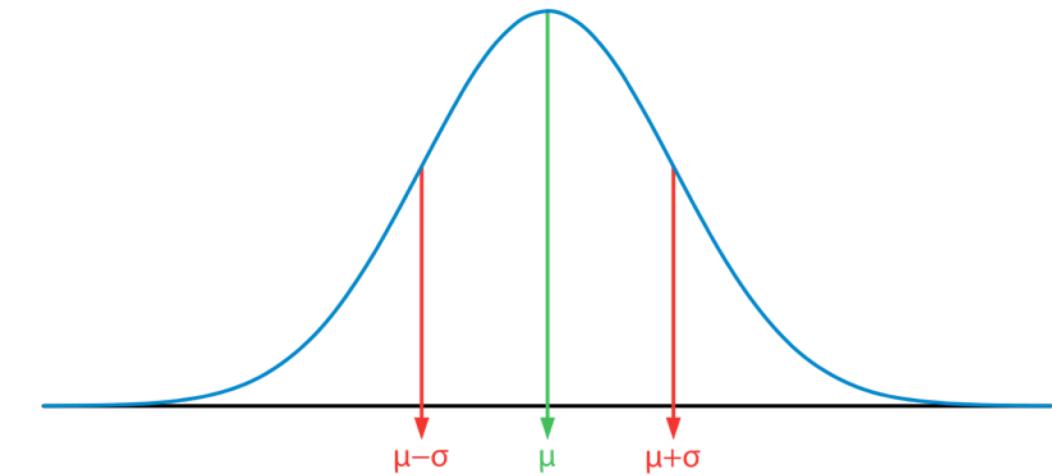
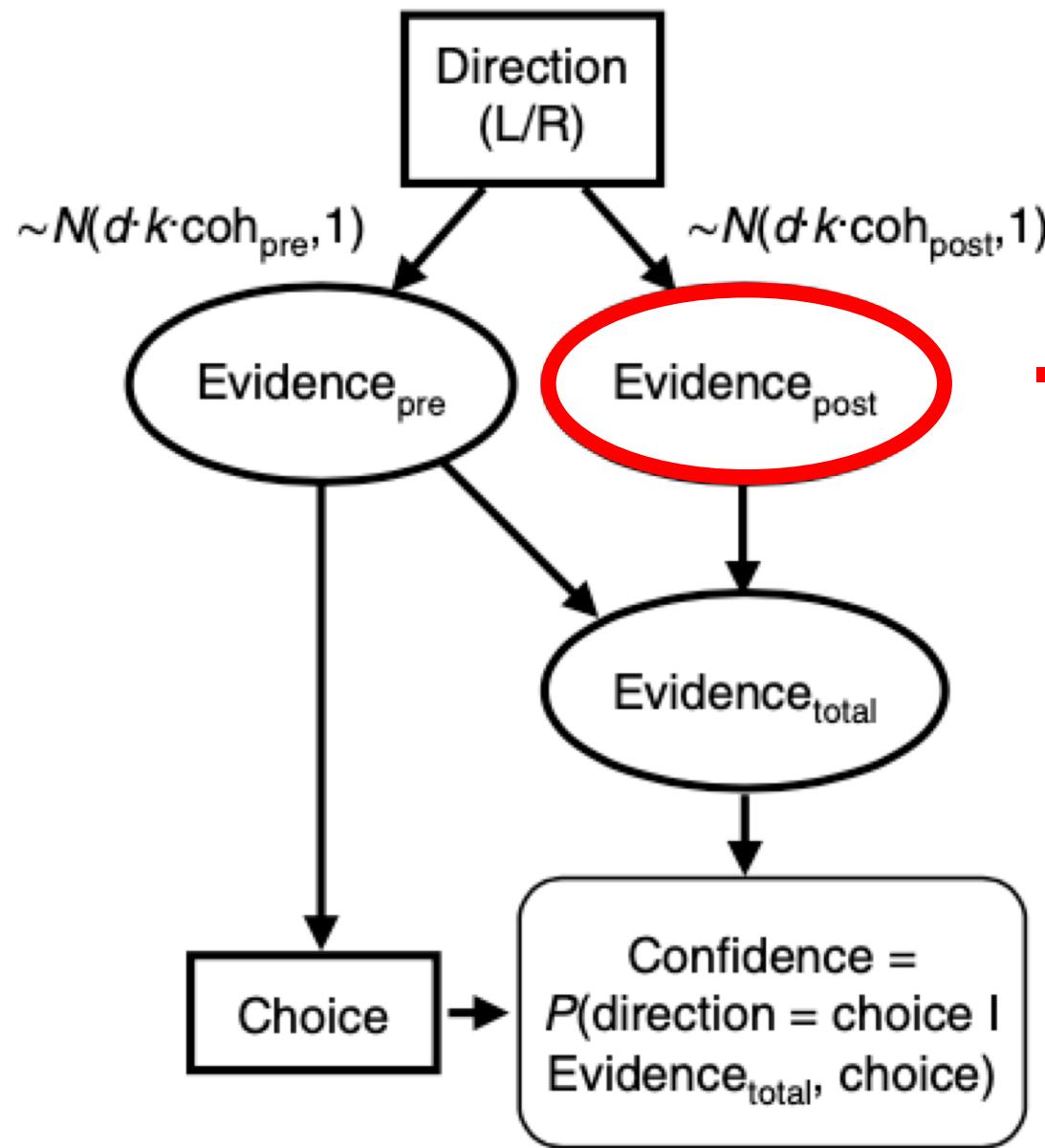
Bayesian graphical model



(Fleming et al., 2018)

b

Bayesian graphical model



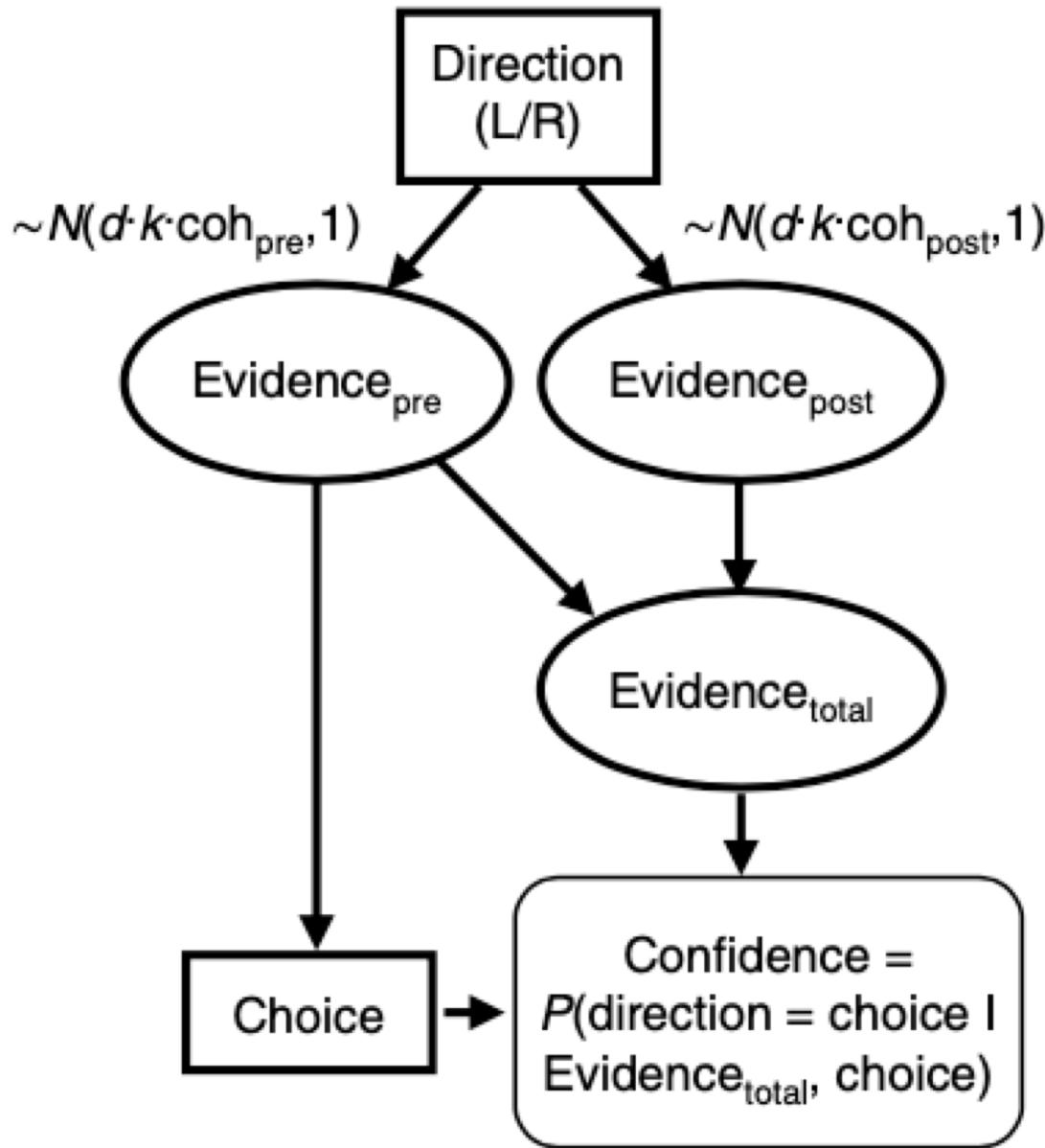
Variables
 $E \sim N(\mu = d \cdot k \cdot coh, \sigma = 1)$

Model parameters

(Fleming et al., 2018)

b

Bayesian graphical model



Latent variables

?

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

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

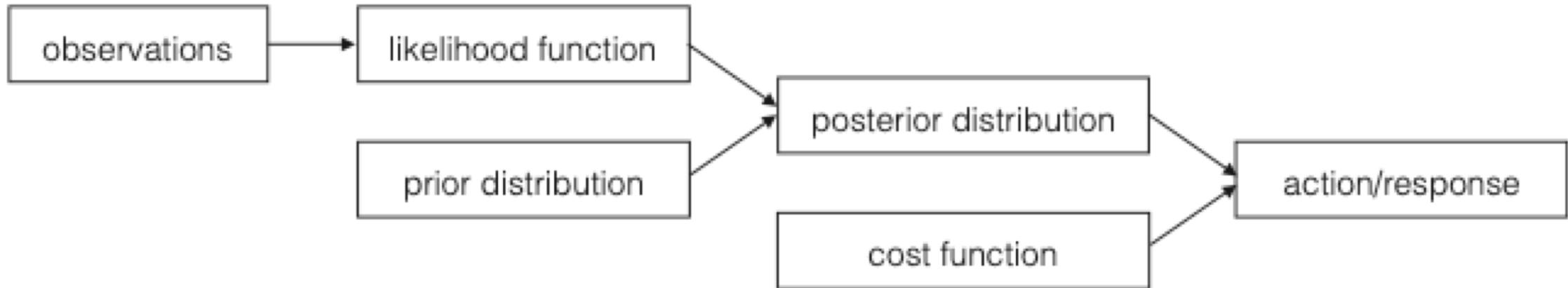
[Evidence_{pre}] $X_{\text{pre}} \sim N(dk\theta_{\text{pre}}, 1)$

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

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

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

Bayesian Decision Models



Bayesian Decision Making Schematic

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

s: state of the world

x: observation by the agent

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?)

$$p(\theta | x) = \frac{p(x | \theta) p(\theta)}{p(x)}$$

posterior

probability distribution of the parameter given the observed data

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

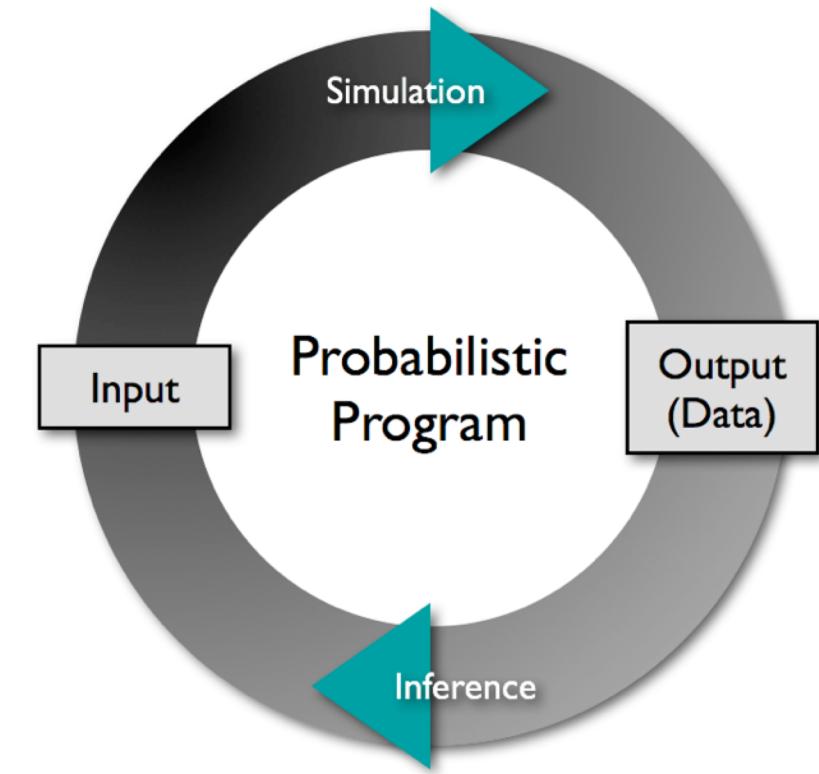
evidence

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.

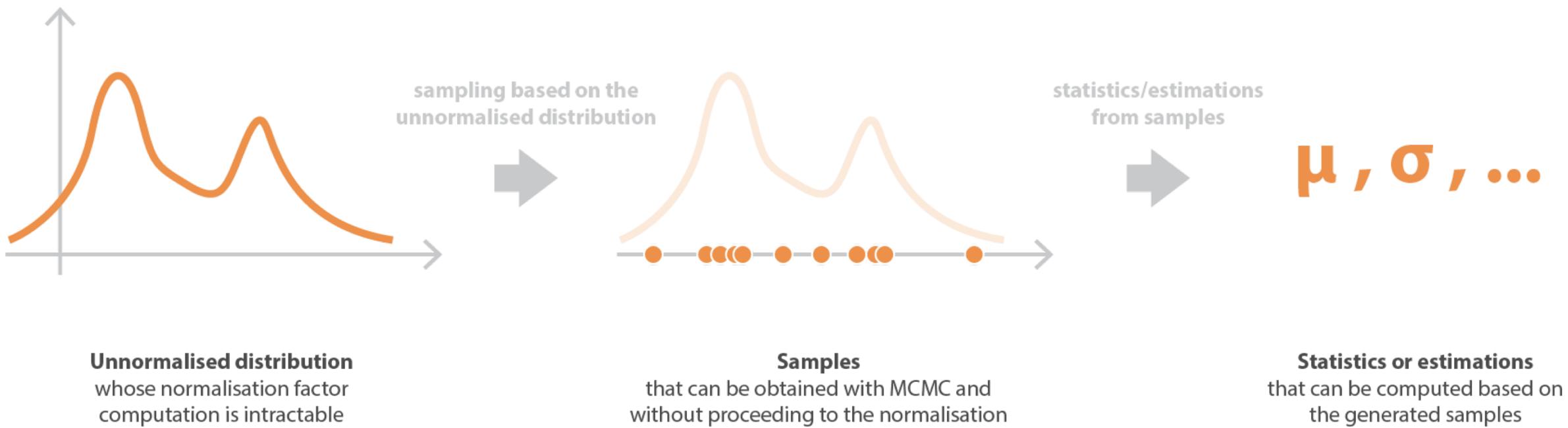


What does Probabilistic Programming involve?

- **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.

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.



Markov chain Monte Carlo

Monte Carlo: Simulations evolving randomly

Law of Large numbers (average → expected value)

the more samples
we have

Markov chain: $P(\text{future} \mid \text{present, past}) = P(\text{future} \mid \text{present, } \dots)$

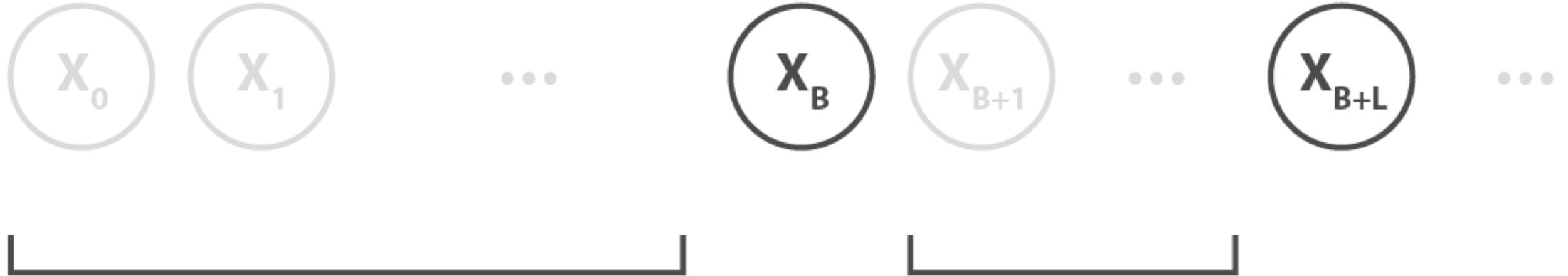
Markov property

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.



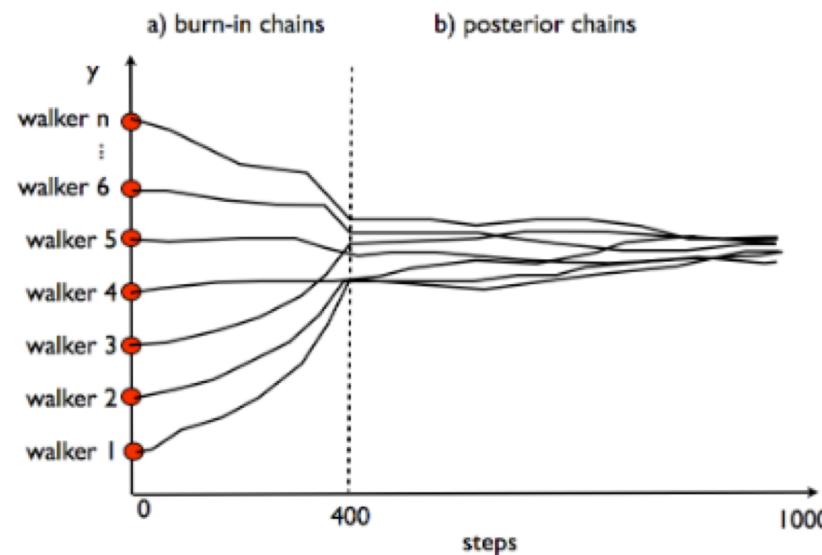
Burn-in time

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

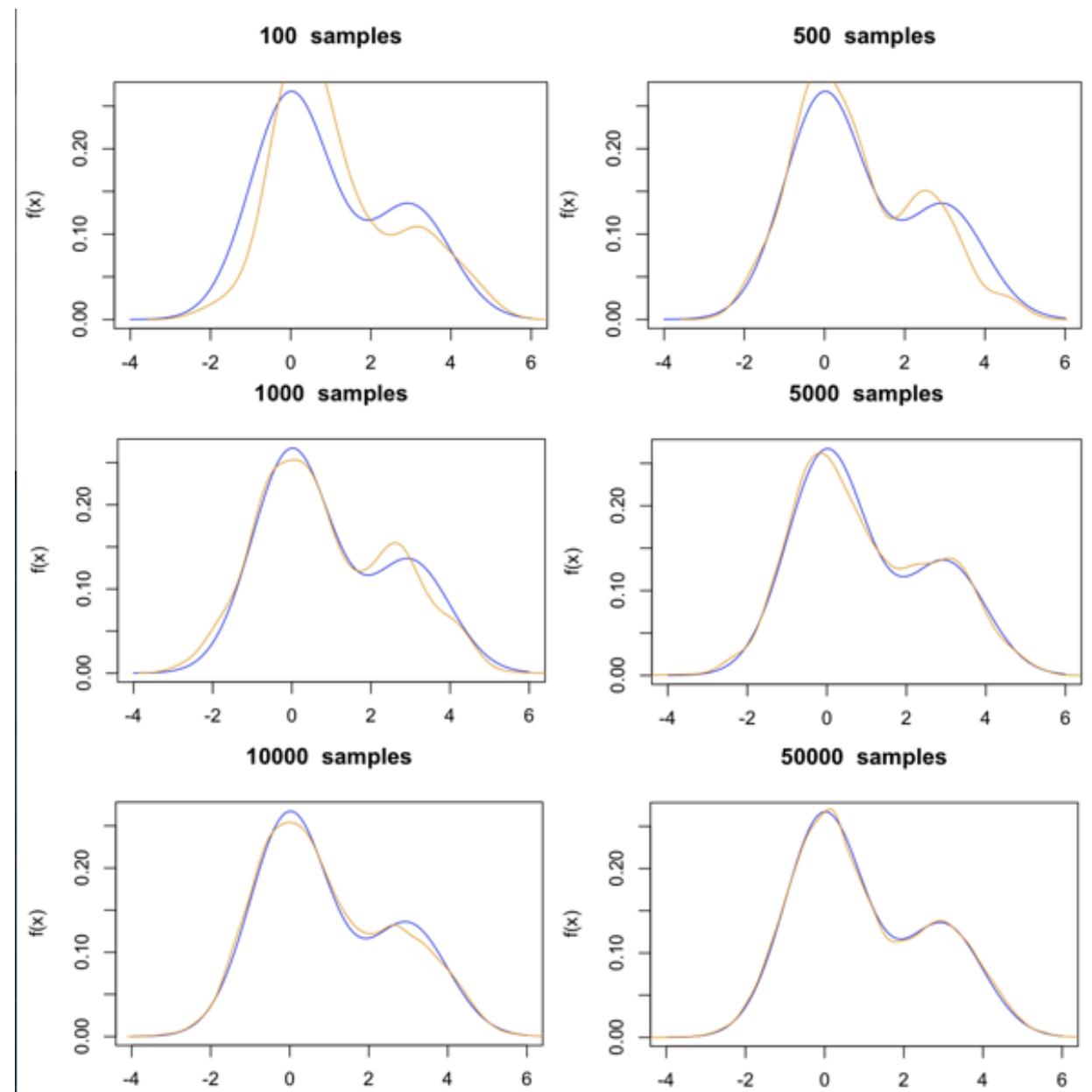
Lag

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

MCMC for one parameter



<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)

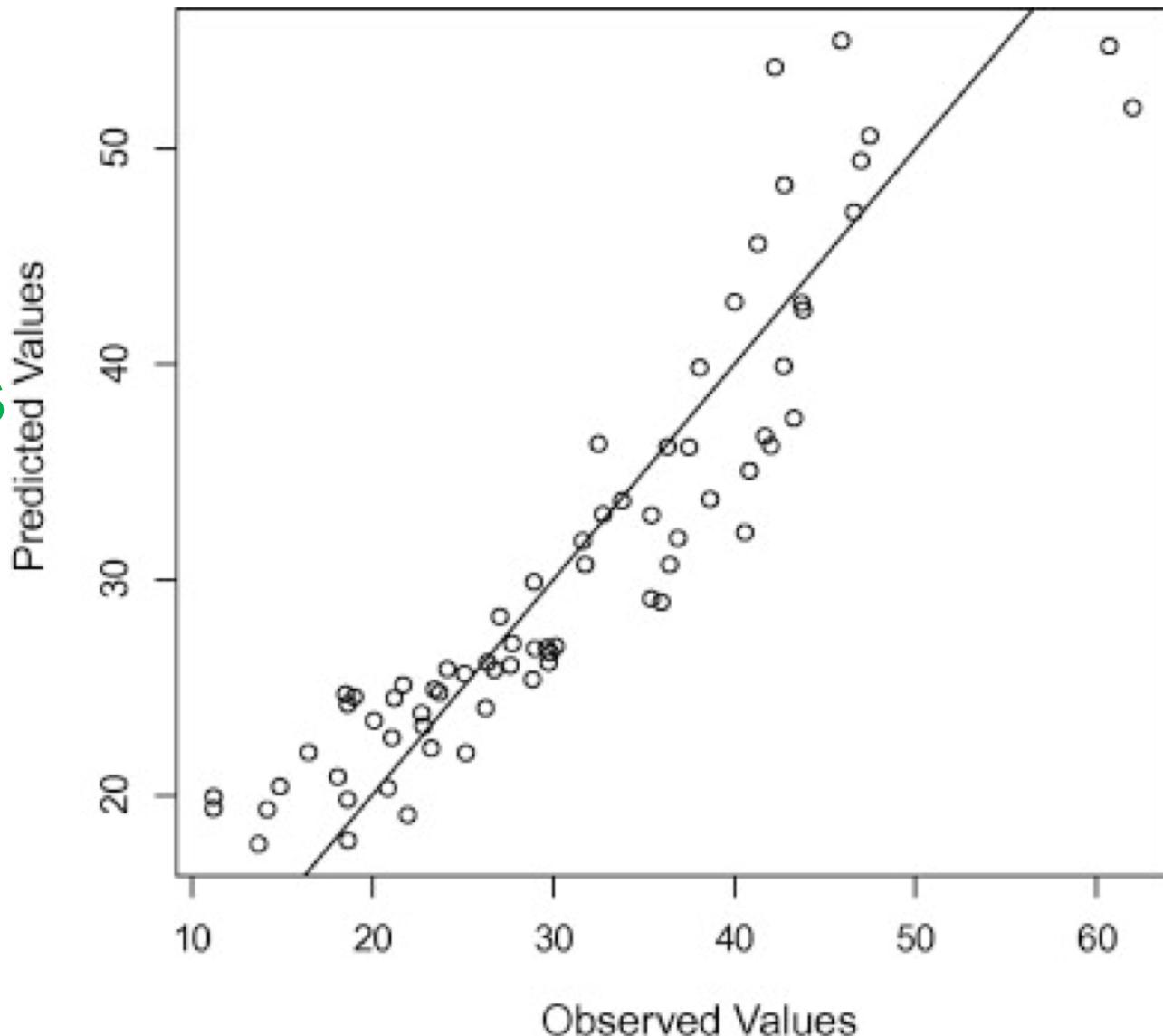
- 1. Generating Data
- 2. Model Specification
- 3. Model fitting
- 4. Posterior Analysis

1st test: Let's model a
GLM

$$Y = m^*X + \text{intercept} + s$$

m: slope for x

s: error (std dev, σ)



Tutorial use data from decision making exp

Like

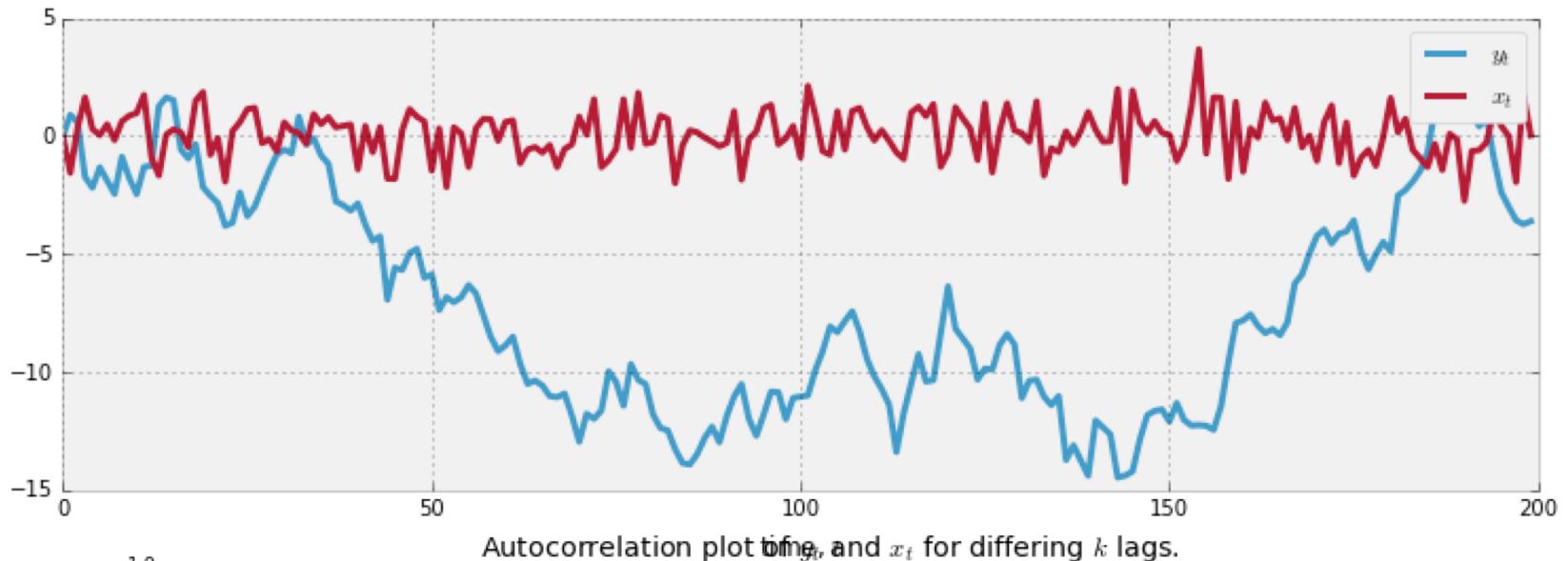


Diagnosing Convergence

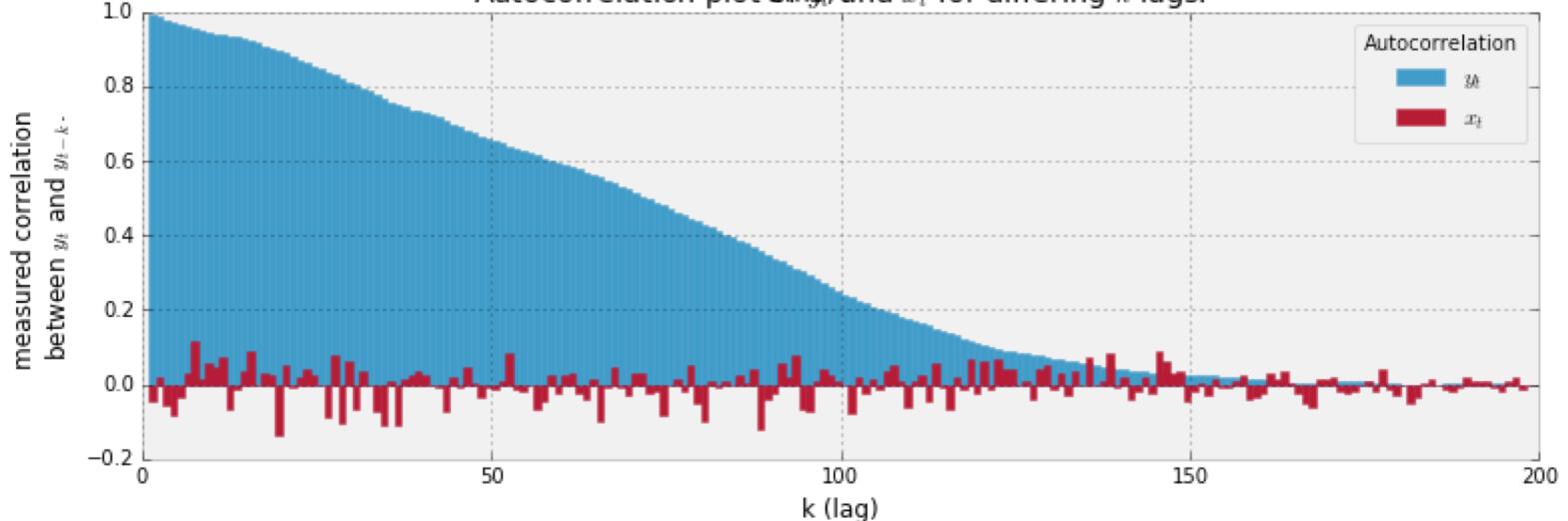
- Gelman-Rubin diagnostic. ($|R - 1| < 0.05$)
 - Effective sample size ($\text{ESS} > 100$)
 - Autocorrelation
 - Check the traces
-
- <https://pymc-devs.github.io/pymc/modelchecking.html>
 - <https://pymc3.readthedocs.io/en/latest/api/diagnostics.html>

Autocorrelation

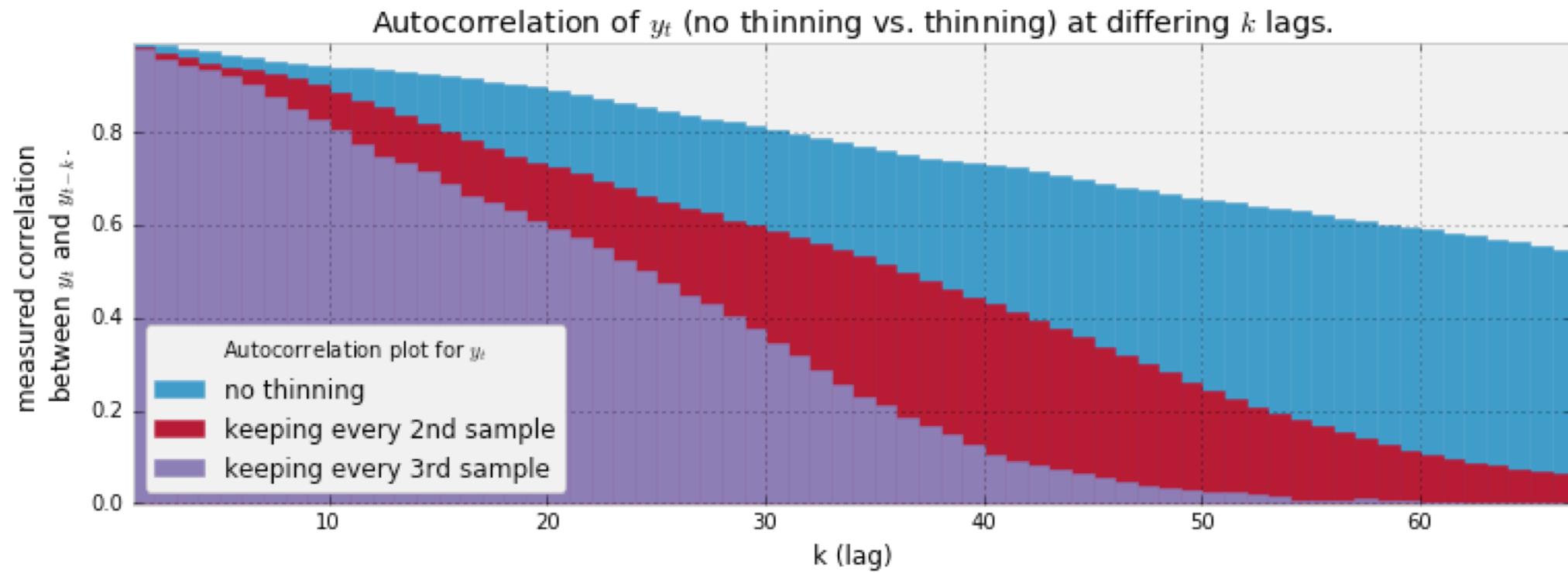
$y_t \sim \text{Normal}(y_{t-1}, 1)$, $y_0 = 0$



Autocorrelation plot of y_t and x_t for differing k lags.



Thinning to decrease autocorr



Extra References

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.

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

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

<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>