Wednesday 14:00-17:30

Part 7

Bayesian hierarchical modelling, simulation and MCMC

by Gero Walter

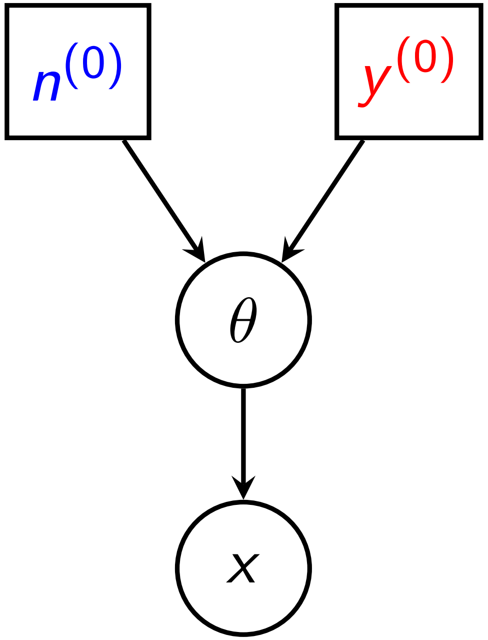
Bayesian hierarchical modelling, simulation and MCMC

Outline

Bayesian Hierarchical Modelling, a.k.a. Bayesian (Belief) Networks, a.k.a. Graphical Models

- many names for the same thing (it's a powerful tool),
  I will use the term Bayesian Networks (BNs)
- BNs as a unifying way to think about (Bayesian) statistical models
- how to build complex Bayesian models out of simple building blocks
- how to specify joint distributions (over many variables)
  via univariate distributions
  using conditional independence assumptions
- conditional independence assumptions are visualized by a graph
- the graph can establish a hierarchy between variables

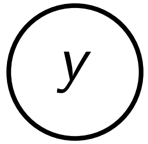
Bayesian Networks: Simple Example



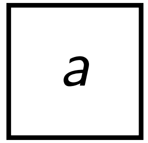
$$f(x \mid \theta) \sim \text{Binomial}(n, \theta)$$

$$f(\theta \mid n^{(0)}, y^{(0)}) \sim \text{Beta}(n^{(0)}, y^{(0)})$$

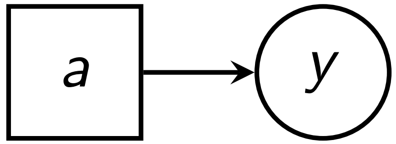
variables / parameters



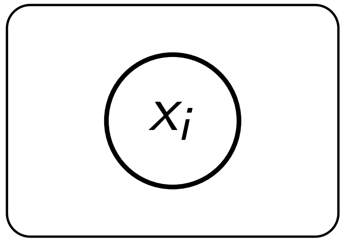
fixed values



y depends on a

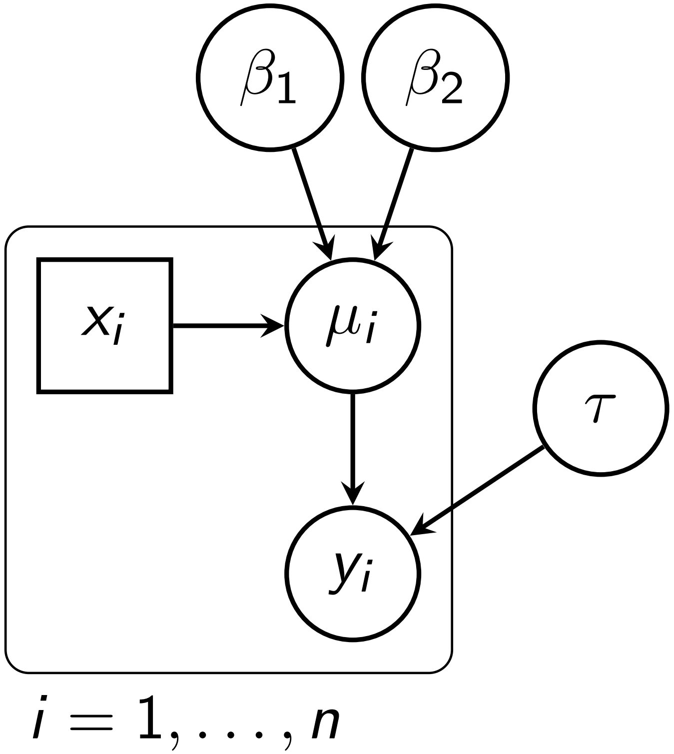


repeated nodes



$i = 1, \ldots, n$

Another Example: Linear Regression



$$y_i = \beta_1 + x_i\beta_2 + \varepsilon_i, \quad \text{where } \varepsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$$

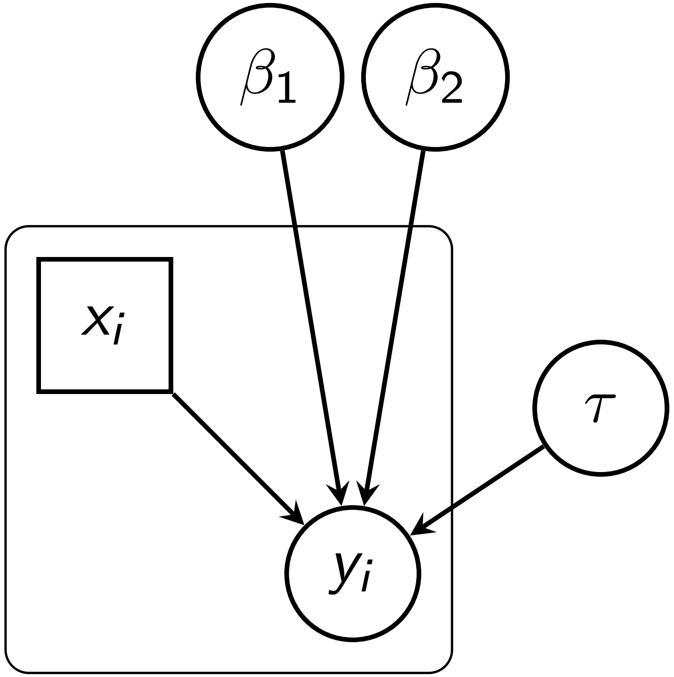
$$y_i \mid \mu_i, \tau \sim \mathrm{N}(\mu_i, 1/\tau), \quad \text{where } \mu_i = \beta_1 + x_i\beta_2$$

$$\tau \mid a, b \sim \mathrm{Gamma}(a, b)$$

$$\beta_1 \mid m_1, t_1 \sim \mathrm{N}(m_1, 1/t_1)$$

$$\beta_2 \mid m_2, t_2 \sim \mathrm{N}(m_2, 1/t_2)$$

Another Example: Linear Regression



$i = 1, \ldots, n$

$$y_i = \beta_1 + x_i\beta_2 + \varepsilon_i, \quad \text{where } \varepsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$$

$$y_i \mid \mu_i, \tau \sim \mathrm{N}(\mu_i, 1/\tau), \quad \text{where } \mu_i = \beta_1 + x_i\beta_2$$

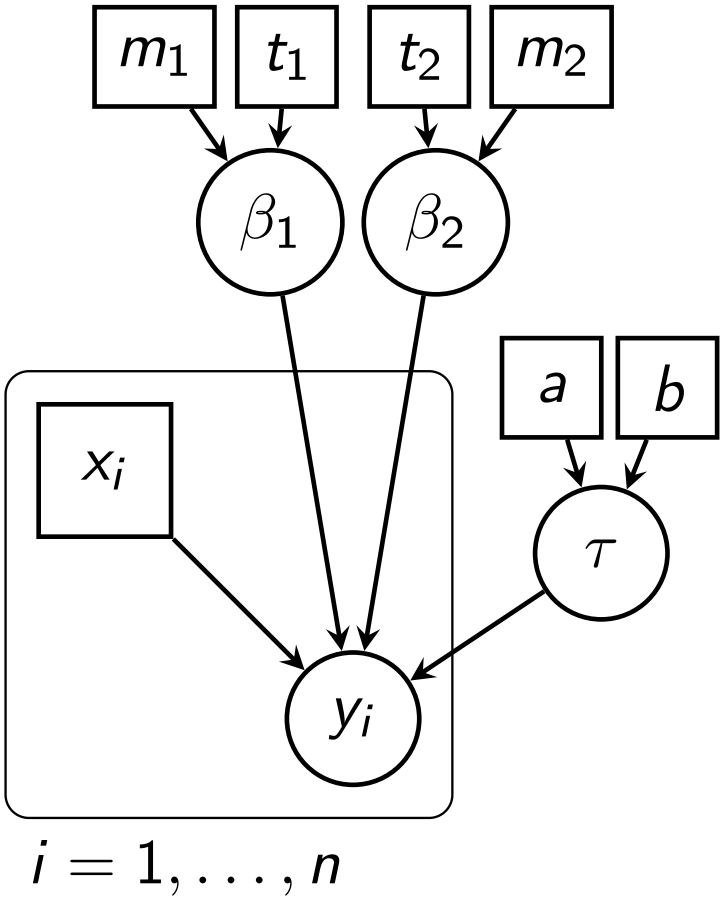
- $y_i \mid \beta_1, \beta_2, \tau \sim \mathrm{N}(\beta_1 + x_i\beta_2, 1/\tau)$

$$\tau \mid a, b \sim \mathrm{Gamma}(a, b)$$

$$\beta_1 \mid m_1, t_1 \sim \mathrm{N}(m_1, 1/t_1)$$

$$\beta_2 \mid m_2, t_2 \sim \mathrm{N}(m_2, 1/t_2)$$

Another Example: Linear Regression



$$y_i = \beta_1 + x_i\beta_2 + \varepsilon_i, \quad \text{where } \varepsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$$

$$y_i \mid \mu_i, \tau \sim \mathrm{N}(\mu_i, 1/\tau), \quad \text{where } \mu_i = \beta_1 + x_i\beta_2$$

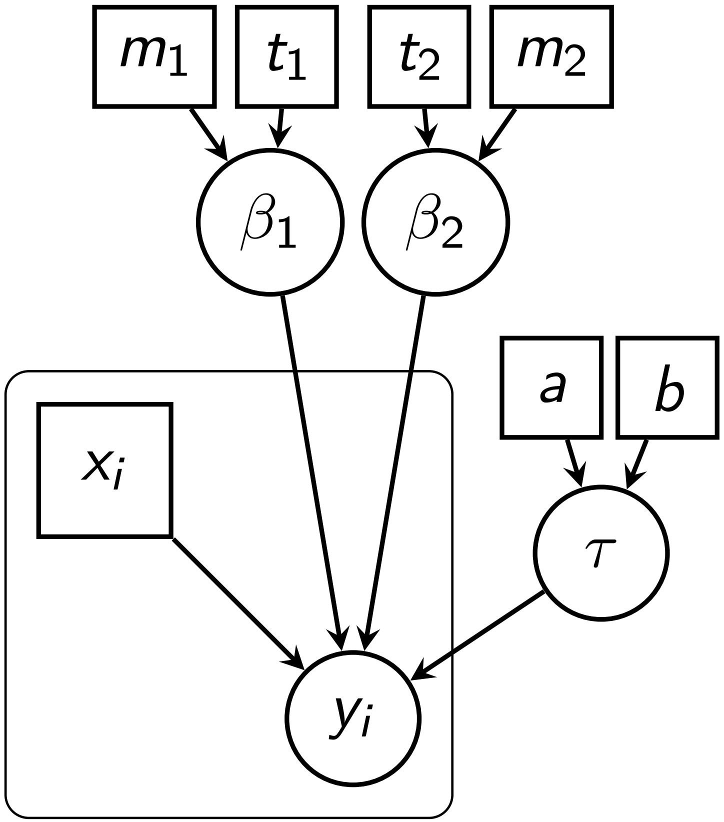
- $y_i \mid \beta_1, \beta_2, \tau \sim \mathrm{N}(\beta_1 + x_i\beta_2, 1/\tau)$

$$\tau \mid a, b \sim \mathrm{Gamma}(a, b)$$

$$\beta_1 \mid m_1, t_1 \sim \mathrm{N}(m_1, 1/t_1)$$

$$\beta_2 \mid m_2, t_2 \sim \mathrm{N}(m_2, 1/t_2)$$

Another Example: Linear Regression



$$y_i = \beta_1 + x_i\beta_2 + \varepsilon_i, \quad \text{where } \varepsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$$

$$y_i \mid \mu_i, \tau \sim \mathrm{N}(\mu_i, 1/\tau), \quad \text{where } \mu_i = \beta_1 + x_i\beta_2$$

- $y_i \mid \beta_1, \beta_2, \tau \sim \mathrm{N}(\beta_1 + x_i\beta_2, 1/\tau)$

$$\tau \mid a, b \sim \mathrm{Gamma}(a, b)$$

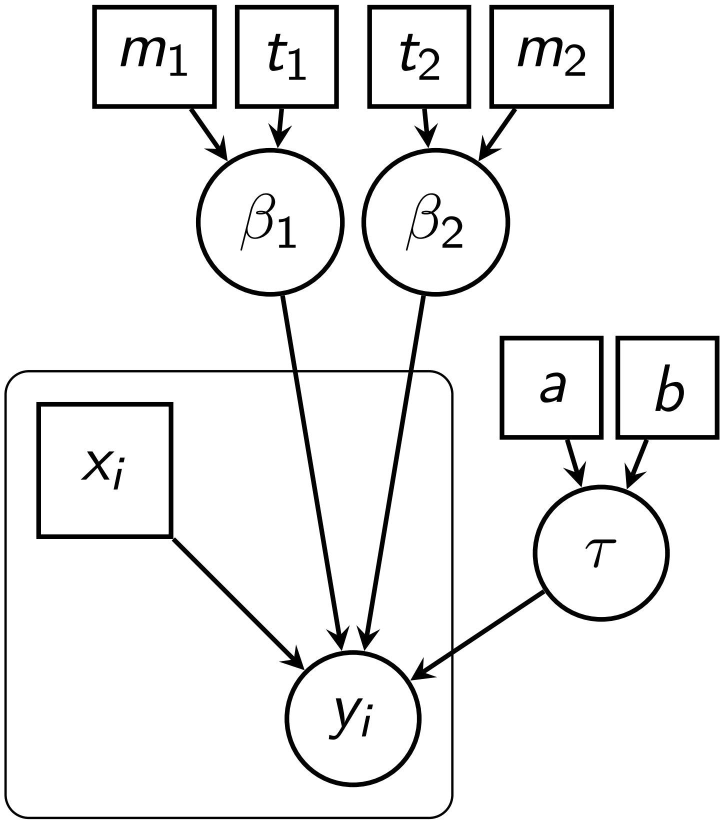
$$a = b = 10^{-3}$$

$$\beta_1 \mid m_1, t_1 \sim \mathrm{N}(m_1, 1/t_1)$$

$$\beta_2 \mid m_2, t_2 \sim \mathrm{N}(m_2, 1/t_2)$$

$$m_1 = m_2 = 0, \quad t_1 = t_2 = 10^4$$

Another Example: Linear Regression



$i = 1, \ldots, n$

$$y_i = \beta_1 + x_i\beta_2 + \varepsilon_i, \quad \text{where } \varepsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$$

$$y_i \mid \mu_i, \tau \sim \mathrm{N}(\mu_i, 1/\tau), \quad \text{where } \mu_i = \beta_1 + x_i\beta_2$$

- $y_i \mid \beta_1, \beta_2, \tau \sim \mathrm{N}(\beta_1 + x_i\beta_2, 1/\tau)$

$$\tau \mid a, b \sim \mathrm{Gamma}(a, b)$$

$$a = b = 10^{-3}$$

$$\beta_1 \mid m_1, t_1 \sim \mathrm{N}(m_1, 1/t_1)$$

$$\beta_2 \mid m_2, t_2 \sim \mathrm{N}(m_2, 1/t_2)$$

$$m_1 = m_2 = 0, \quad t_1 = t_2 = 10^4$$

graph:
cond. indep. relations

variables:
conditional distributions

Bayesian Networks: Why?

- conditional independence reduces model complexity
  (10 binary variables: joint distribution has 1023 parameters)

Bayesian Networks: Why?

- conditional independence reduces model complexity
(10 binary variables: joint distribution has 1023 parameters)
- BNs enable us to construct probability distributions that capture the important dependencies between the relevant variables in a given inference problem while keeping models (relatively) simple

Bayesian Networks: Why?

- conditional independence reduces model complexity (10 binary variables: joint distribution has 1023 parameters)
- BNs enable us to construct probability distributions that capture the important dependencies between the relevant variables in a given inference problem while keeping models (relatively) simple
- often: BN = discrete variables only, distributions defined via conditional probability tables (CPTs)

Bayesian Networks: Why?

- conditional independence reduces model complexity (10 binary variables: joint distribution has 1023 parameters)
- BNs enable us to construct probability distributions that capture the important dependencies between the relevant variables in a given inference problem while keeping models (relatively) simple
- often: BN = discrete variables only, distributions defined via conditional probability tables (CPTs)
- What kind of graphs work for expressing conditional independence relations?

Bayesian Networks: Directed Acyclic Graphs

Definition (Directed Graph)

A *directed graph* $G = (V, E)$ consists of a set of vertices V and a set of edges E, where $E \subset V \times V$. An arrow leads from $u \in V$ to $v \in V$ if and only if $(u, v) \in E$; u is the *source* and v is the *target* of edge (u, v).

Bayesian Networks: Directed Acyclic Graphs

Definition (Directed Graph)

A *directed graph* $G = (V, E)$ consists of a set of vertices V and a set of edges E, where $E \subset V \times V$. An arrow leads from $u \in V$ to $v \in V$ if and only if $(u, v) \in E$; u is the *source* and v is the *target* of edge (u, v).

Definition (Paths and Cycles)

A *path* in a graph is an ordered set of edges $\{e_i\}$ such that $t(e_i) = s(e_{i+1})$ (chain of head-to-tail arrows). A *cycle* is a path such that $t(e_N) = s(e_1)$, where N is the number of edges in the path.

Bayesian Networks: Directed Acyclic Graphs

Definition (Directed Graph)

A *directed graph* $G = (V, E)$ consists of a set of vertices V and a set of edges E, where $E \subset V \times V$. An arrow leads from $u \in V$ to $v \in V$ if and only if $(u, v) \in E$; u is the *source* and v is the *target* of edge (u, v).

Definition (Paths and Cycles)

A *path* in a graph is an ordered set of edges $\{e_i\}$ such that $t(e_i) = s(e_{i+1})$ (chain of head-to-tail arrows). A *cycle* is a path such that $t(e_N) = s(e_1)$, where N is the number of edges in the path.

Definition (Directed Acyclic Graph)

A *directed acyclic graph (DAG)* is a directed graph that does not contain a cycle, i.e. there does not exist a subset of edges that forms a cycle.

Bayesian Networks: Formal Definition

Definition (Bayesian Network)

Given a DAG $G = (V, E)$, and variables $x_V = \{x_v\}_{v \in V}$, a *Bayesian network* with respect to G and x_V is a joint probability distribution for the x_V of the form:

$$f(x_V) = \prod_{v \in V} f\left(x_v \mid x_{\mathrm{pa}(v)}\right)$$

where $\mathrm{pa}(v)$ is the set of parents of v, i.e. the set of vertices u such that (u, v) is an edge.

Bayesian Networks: Formal Definition

Definition (Bayesian Network)

Given a DAG $G = (V, E)$, and variables $x_V = \{x_v\}_{v \in V}$, a *Bayesian network* with respect to G and x_V is a joint probability distribution for the x_V of the form:

$$f(x_V) = \prod_{v \in V} f\left(x_v \mid x_{\mathrm{pa}(v)}\right)$$

where $\mathrm{pa}(v)$ is the set of parents of v, i.e. the set of vertices u such that (u, v) is an edge.

- joint distribution factorizes according to the graph!

Bayesian Networks: Factorization of the joint

One can always factorize a joint distribution by

$$f(x_1, \ldots, x_K) = f(x_K \mid x_1, \ldots, x_{K-1})\, f(x_{K-1} \mid x_1, \ldots, x_{K-2}) \cdots f(x_3 \mid x_1, x_2)\, f(x_2 \mid x_1)\, f(x_1)$$

Bayesian Networks: Factorization of the joint

One can always factorize a joint distribution by

$$f(x_1,\ldots,x_K) = f(x_K \mid x_1,\ldots,x_{K-1})\, f(x_{K-1} \mid x_1,\ldots,x_{K-2}) \\ \cdots f(x_3 \mid x_1,x_2)\, f(x_2 \mid x_1)\, f(x_1)$$

- corresponds to a *fully connected graph*: there is a link between every pair of vertices (each of the K vertices has incoming edges from all lower-numbered vertices)

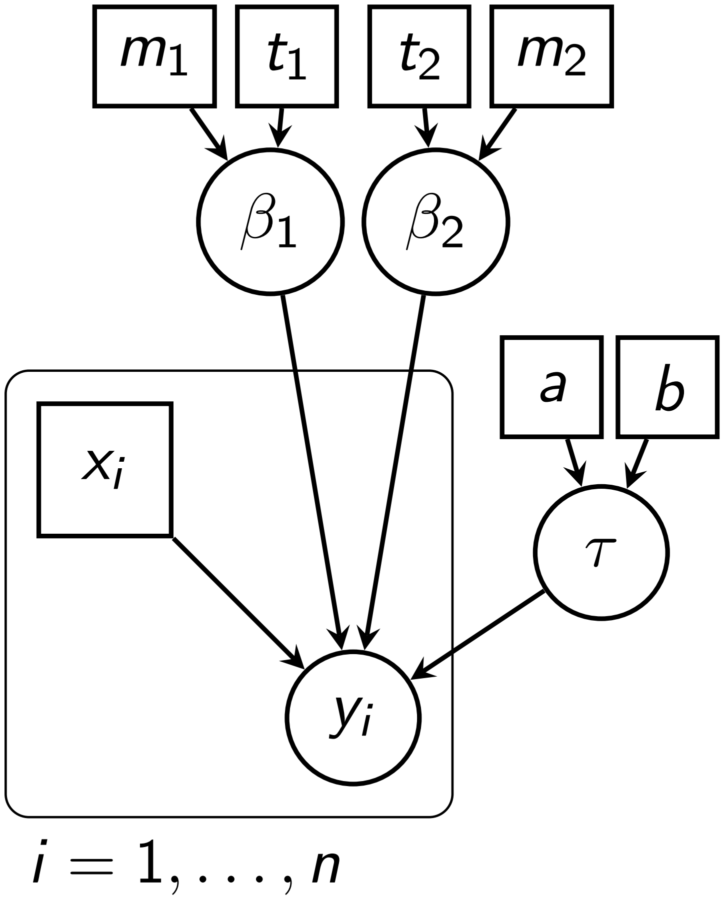
Bayesian Networks: Factorization of the joint

One can always factorize a joint distribution by

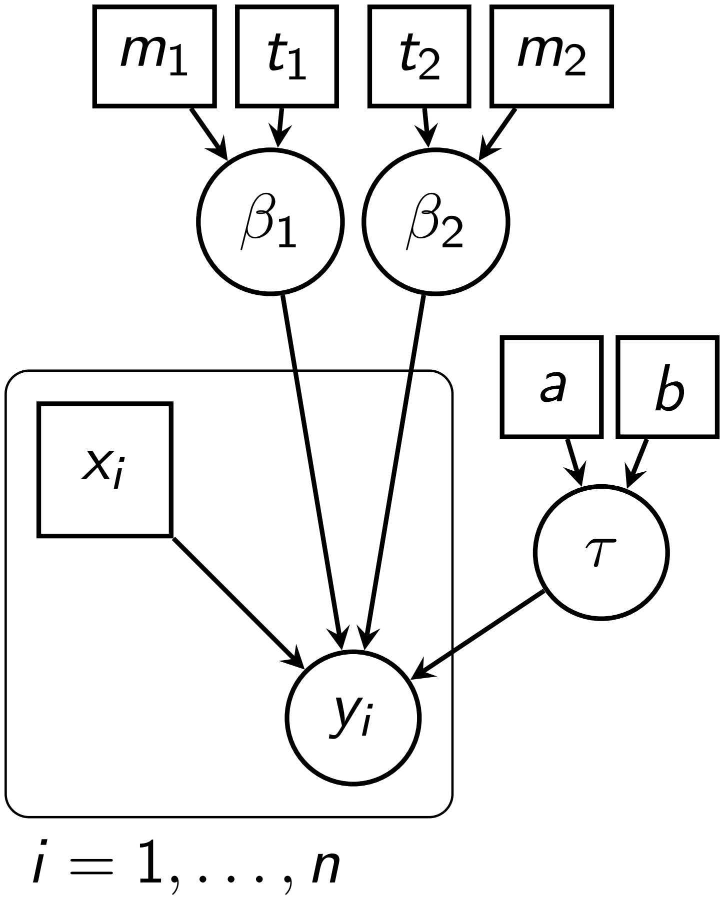
$$\begin{aligned} f(x_1, \ldots, x_K) &= f(x_K \mid x_1, \ldots, x_{K-1})\, f(x_{K-1} \mid x_1, \ldots, x_{K-2}) \\ &\qquad \cdots f(x_3 \mid x_1, x_2)\, f(x_2 \mid x_1)\, f(x_1) \end{aligned}$$

- corresponds to a *fully connected graph*: there is a link between every pair of vertices (each of the K vertices has incoming edges from all lower-numbered vertices)
- sparser graph $\implies$ nodes have fewer parents
  $\implies$ less complex joint

Factorization of the joint: Example



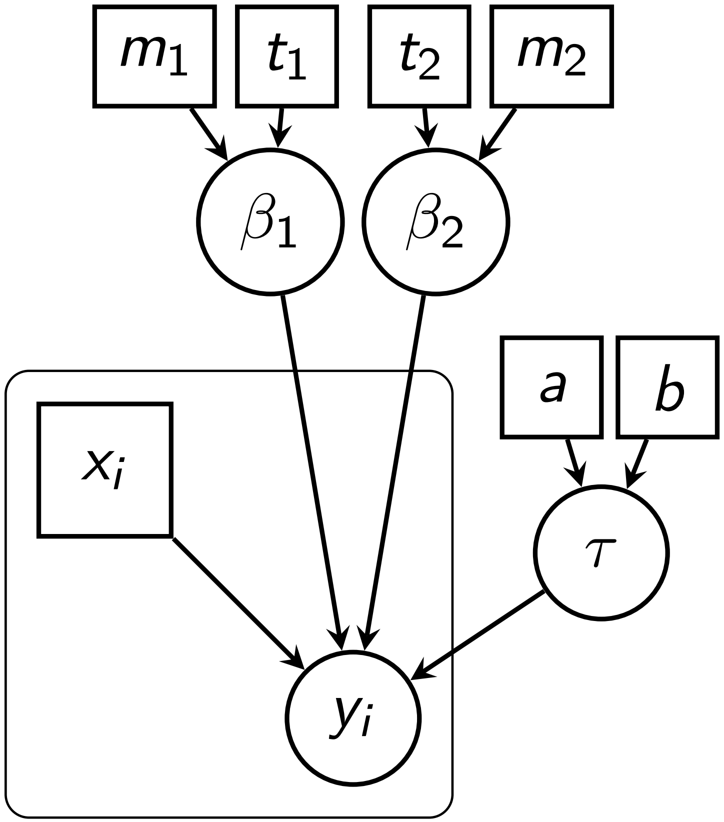
Factorization of the joint: Example



omitting the fixed values in notation: the joint distribution

$$
\begin{aligned}
&f(y_1, \ldots, y_n, \beta_1, \beta_2, \tau) \\
&= \prod_{i=1}^{n} f(y_i \mid \beta_1, \beta_2, \tau)\; f(\beta_1)\, f(\beta_2)\, f(\tau)
\end{aligned}
$$

Factorization of the joint: Example

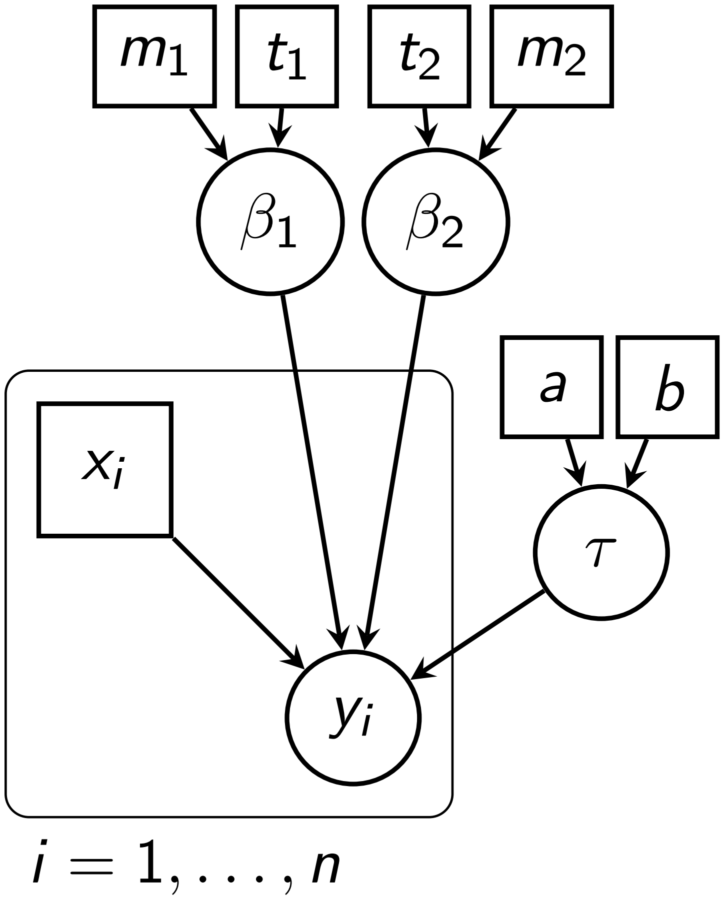


omitting the fixed values in notation: the joint distribution

$$
\begin{aligned}
&f(y_1, \ldots, y_n, \beta_1, \beta_2, \tau) \\
&= \underbrace{\prod_{i=1}^{n} f(y_i \mid \beta_1, \beta_2, \tau)}_{\text{likelihood}} \underbrace{f(\beta_1)\, f(\beta_2)\, f(\tau)}_{\textit{prior}}
\end{aligned}
$$

is $\propto$ posterior $f(\beta_1, \beta_2, \tau \mid y_1, \ldots, y_n)$

Factorization of the joint: Example



omitting the fixed values in notation: the joint distribution

$$f(y_1, \ldots, y_n, \beta_1, \beta_2, \tau)$$

$$= \underbrace{\prod_{i=1}^{n} f(y_i \mid \beta_1, \beta_2, \tau)}_{\text{likelihood}} \underbrace{f(\beta_1)\, f(\beta_2)\, f(\tau)}_{prior}$$

is $\propto$ posterior $f(\beta_1, \beta_2, \tau \mid y_1, \ldots, y_n)$

- it would be really useful to get posterior estimates based on the non-normalized density $f(y_1, \ldots, y_n, \beta_1, \beta_2, \tau)$!

Bayesian Networks: Inference

- Markov Chain Monte Carlo:
  simulate samples from the joint $\propto$ posterior (▶ next block!)

Bayesian Networks: Inference

- Markov Chain Monte Carlo:
  simulate samples from the joint $\propto$ posterior (▶ next block!)
- can get any distributions for any (set of) variables in the graph
  by conditioning and marginalizing of the joint

Bayesian Networks: Inference

- Markov Chain Monte Carlo:
  simulate samples from the joint $\propto$ posterior (▶ next block!)
- can get any distributions for any (set of) variables in the graph by conditioning and marginalizing of the joint
- for a set of M samples from the joint,
  $\{\beta_1^m, \beta_2^m, \tau^m\}, m = 1, \ldots, M,$

Bayesian Networks: Inference

- Markov Chain Monte Carlo:
  simulate samples from the joint $\propto$ posterior (▶ next block!)
- can get any distributions for any (set of) variables in the graph by conditioning and marginalizing of the joint
- for a set of M samples from the joint,
  $\{\beta_1^m, \beta_2^m, \tau^m\}, m = 1, \ldots, M$,
  - marginalizing = use only, e.g., $\{\beta_1^m\}, m = 1, \ldots, M$

Bayesian Networks: Inference

- Markov Chain Monte Carlo:
  simulate samples from the joint $\propto$ posterior (▶ next block!)
- can get any distributions for any (set of) variables in the graph by conditioning and marginalizing of the joint
- for a set of M samples from the joint,
  $\{\beta_1^m, \beta_2^m, \tau^m\}, m = 1, \ldots, M$,
  - marginalizing = use only, e.g., $\{\beta_1^m\}, m = 1, \ldots, M$
  - conditioning = use only samples m with the right value of the conditioning parameter(s)
    (or redo the sampling with fixed conditioned values)

Bayesian Networks with Imprecise Probability

- use sets of conditional distributions at nodes: credal networks (see, e.g., [2, §10], [17])

Bayesian Networks with Imprecise Probability

- use sets of conditional distributions at nodes: credal networks (see, e.g., [2, §10], [17])
- specific algorithms for discrete credal networks (see, e.g., [2, §10.5.3], or [14])

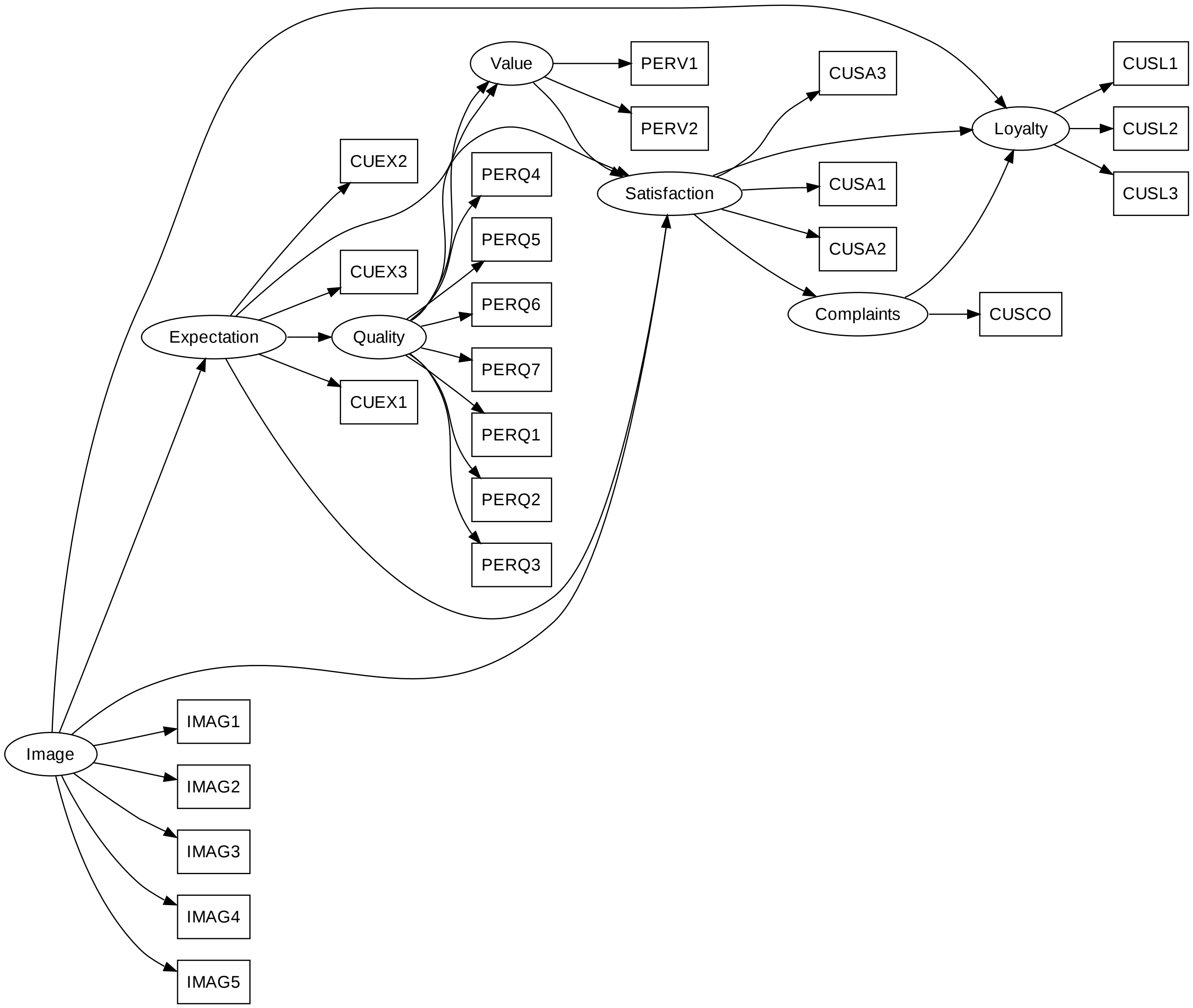
Bayesian Networks with Imprecise Probability

- use sets of conditional distributions at nodes: credal networks (see, e.g., [2, §10], [17])
- specific algorithms for discrete credal networks (see, e.g., [2, §10.5.3], or [14])
- conditional independence with IP gets very non-trivial (see, e.g., [2, §4] for the gory details)

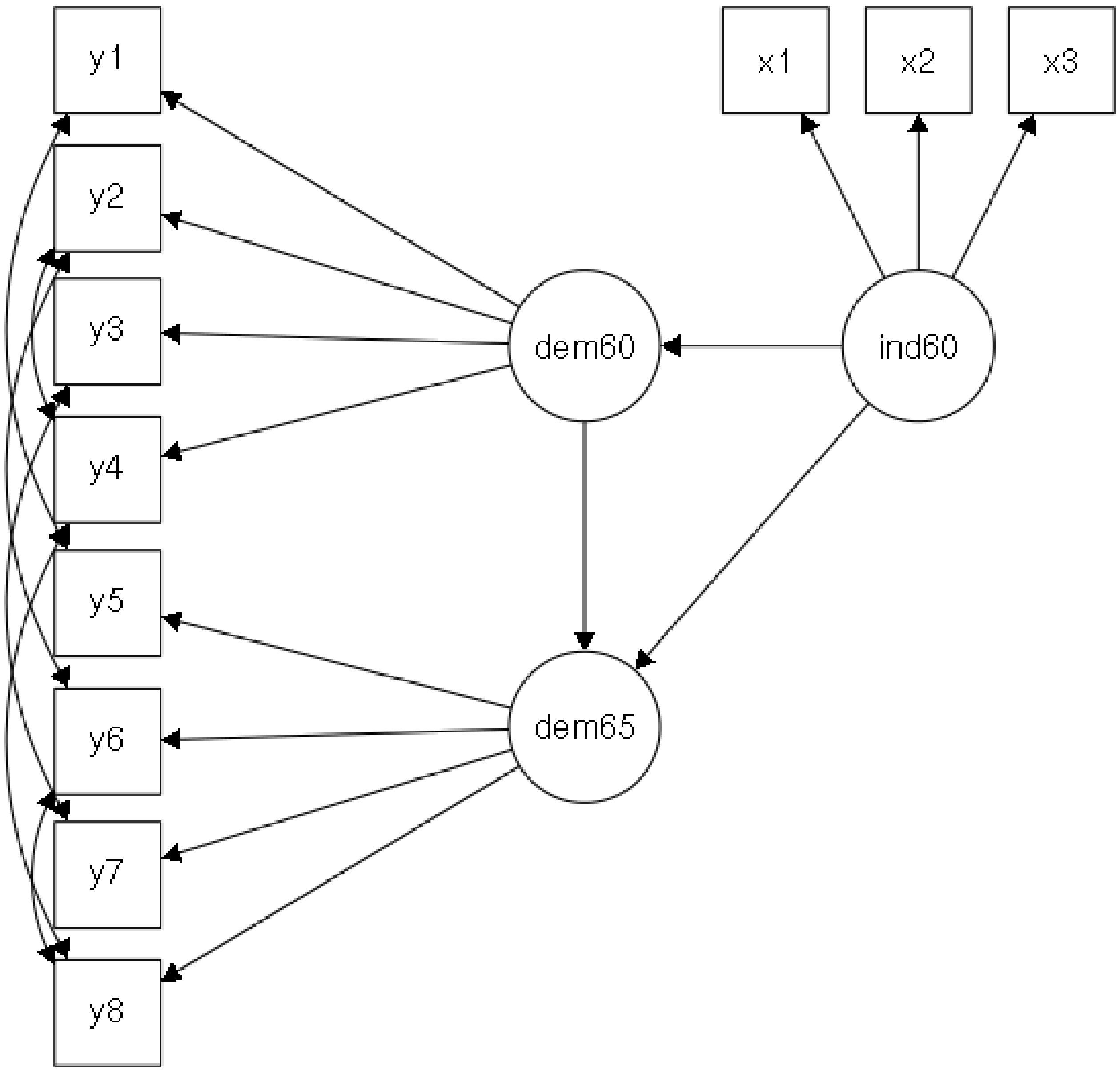
Bayesian Networks with Imprecise Probability

- use sets of conditional distributions at nodes: credal networks (see, e.g., [2, §10], [17])
- specific algorithms for discrete credal networks (see, e.g., [2, §10.5.3], or [14])
- conditional independence with IP gets very non-trivial (see, e.g., [2, §4] for the gory details)
- here: do sensitivity analysis by varying prior distributions in sets: $f(\beta_1) \in \mathcal{M}_{\beta_1}, \ldots$

Other Graph-Based Methods: SEM, Path Analysis



Other Graph-Based Methods: SEM, Path Analysis



Other Graph-Based Methods: SEM, Path Analysis

- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different

Other Graph-Based Methods: SEM, Path Analysis

- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different
- used to estimate latent constructs by assuming linear relationships with measurements (*measurement / outer model*) and relationships between latent constructs (*structural model*)

Other Graph-Based Methods: SEM, Path Analysis

- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different
- used to estimate latent constructs by assuming linear relationships with measurements (*measurement / outer model*) and relationships between latent constructs (*structural model*)
  - example: customer satisfaction is measured by survey questions 1 and 2 (measurement model); brand loyality is a function of customer satisfaction (structural model)

Other Graph-Based Methods: SEM, Path Analysis

- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different
- used to estimate latent constructs by assuming linear relationships with measurements (*measurement / outer model*) and relationships between latent constructs (*structural model*)
  - example: customer satisfaction is measured by survey questions 1 and 2 (measurement model); brand loyality is a function of customer satisfaction (structural model)
- estimation of factor loadings (= regression coefficients)

Other Graph-Based Methods: SEM, Path Analysis

- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different
- used to estimate latent constructs by assuming linear relationships with measurements (*measurement / outer model*) and relationships between latent constructs (*structural model*)
  - example: customer satisfaction is measured by survey questions 1 and 2 (measurement model); brand loyality is a function of customer satisfaction (structural model)
- estimation of factor loadings (= regression coefficients)
  - likelihood-based (**R** package `lavaan`):
    models expectations and (co)variances, not full distributions ($\rightarrow$ multivariate normal)

Other Graph-Based Methods: SEM, Path Analysis

- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different
- used to estimate latent constructs by assuming linear relationships with measurements (*measurement / outer model*) and relationships between latent constructs (*structural model*)
  - example: customer satisfaction is measured by survey questions 1 and 2 (measurement model); brand loyality is a function of customer satisfaction (structural model)
- estimation of factor loadings (= regression coefficients)
  - likelihood-based (**R** package `lavaan`): models expectations and (co)variances, not full distributions ($\rightarrow$ multivariate normal)
  - Bayesian SEM: (**R** package `blavaan`)

Other Graph-Based Methods: SEM, Path Analysis

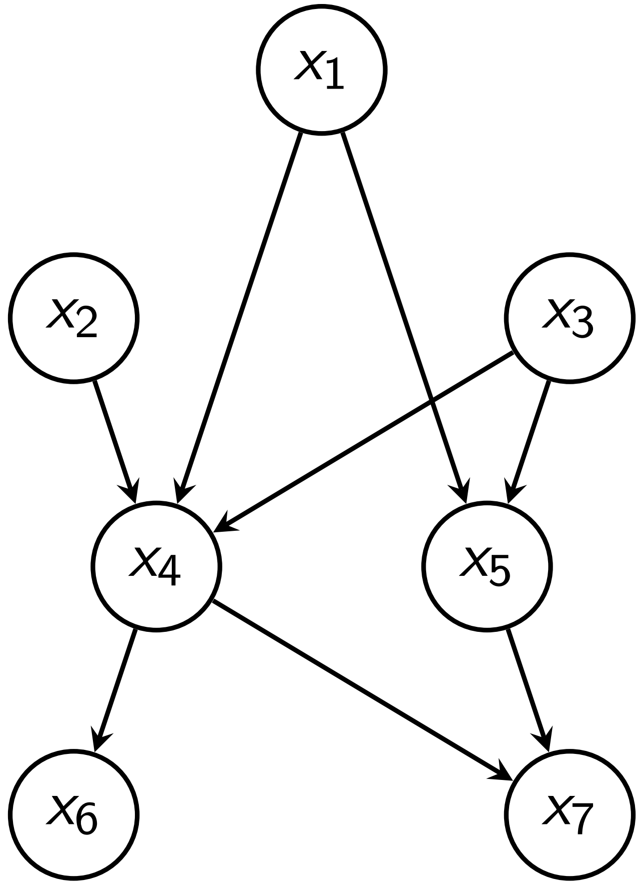
- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different
- used to estimate latent constructs by assuming linear relationships with measurements (*measurement / outer model*) and relationships between latent constructs (*structural model*)
  - example: customer satisfaction is measured by survey questions 1 and 2 (measurement model); brand loyality is a function of customer satisfaction (structural model)
- estimation of factor loadings (= regression coefficients)
  - likelihood-based (**R** package `lavaan`): models expectations and (co)variances, not full distributions ($\rightarrow$ multivariate normal)
  - Bayesian SEM: (**R** package `blavaan`)
  - partial least squares (**R** package `semPLS`): iterative fitting of latent variable values and regression coefficients via least squares

Other Graph-Based Methods: SEM, Path Analysis

- Structural Equation Modeling (SEM, a.k.a. path modeling) uses graphs like BNs, but is something different
- used to estimate latent constructs by assuming linear relationships with measurements (*measurement / outer model*) and relationships between latent constructs (*structural model*)
  - example: customer satisfaction is measured by survey questions 1 and 2 (measurement model); brand loyality is a function of customer satisfaction (structural model)
- estimation of factor loadings ($=$ regression coefficients)
  - likelihood-based (**R** package `lavaan`):
    models expectations and (co)variances, not full distributions ($\rightarrow$ multivariate normal)
  - Bayesian SEM: (**R** package `blavaan`)
  - partial least squares (**R** package `semPLS`):
    iterative fitting of latent variable values
    and regression coefficients via least squares
- path analysis: special case where a measurement can be linked to only one construct

Exercise 1: Factorization of a Joint

Which factorization of
$f\left(\{x_i\}_{i\in[1,\ldots,7]}\right)$
does this graph encode?



Exercise 2: Naive Bayes Classifier

The naive Bayes classifier from Part 6 assumes that the joint distribution of class c and attributes $a_1, \ldots, a_k$ can be factorized as

$$p(c, a) = p(c)p(a \mid c) = p(c) \prod_{i=1}^{k} p(a_i \mid c).$$

Draw the corresponding DAG!
(Hint: use either a plate or consider two attributes a_1 and a_2 only.)

Exercise 3: Naive Bayes Classifier with Dirichlet Priors

We can introduce parameters for $p(c)$ and $p(a_i \mid c)$:

$$(n(c))_{c\in\mathcal{C}} \sim \text{Multinomal}(\theta_c; c \in \mathcal{C}) \tag{36}$$

$$\forall c \in \mathcal{C}\colon (n(a_i, c))_{a_i\in\mathcal{A}_i} \mid c \sim \text{Multinomal}(\theta_{a_i|c}; a_i \in \mathcal{A}_i) \tag{37}$$

where $\mathcal{C}$ denotes the set of all possible class values, and $\mathcal{A}_i$ denotes the set of all possible values of attribute i.

The θ parameters can be estimated using a Dirichlet prior:

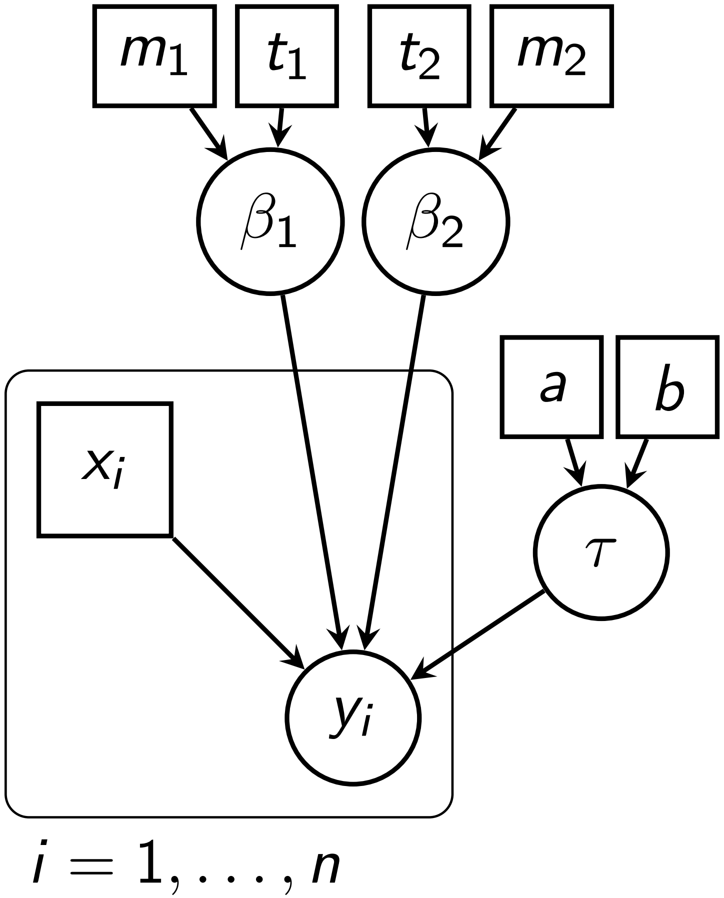
$$(\theta_c)_{c\in\mathcal{C}} \sim \text{Dir}(s, (t(c))_{c\in\mathcal{C}}) \tag{38}$$

$$\forall c \in \mathcal{C}\colon (\theta_{a_i|c})_{a_i\in\mathcal{A}_i} \mid c \sim \text{Dir}(s, (t(a_i, c))_{a_i\in\mathcal{A}_i}) \tag{39}$$

where we must have that $\sum_{a_i\in\mathcal{A}_i} t(a_i, c) = t(c)$. [Note that $t(c)$ is the prior expectation of θ_c and $t(a_i, c)/t(c)$ is the prior expectation of $\theta_{a_i|c}$.]

Draw the corresponding graph!

Exercise 4: Sensitivity Analysis



In the linear regression example there are 6 hyperparameters m_1, t_1, m_2, t_2, a, b.

How would you do sensitivity analysis over the prior in that example? What problems do you foresee?

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, . . .

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, ...
- joint $\propto$ posterior usually intractable: how to do inference?

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, ...
- joint $\propto$ posterior usually intractable: how to do inference?
- simulate samples from joint / posterior: approximate ...

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, ...
- joint $\propto$ posterior usually intractable: how to do inference?
- simulate samples from joint / posterior: approximate ...
  - ... posterior cdf by empirical cdf (density: kernel dens. est.)

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, ...
- joint $\propto$ posterior usually intractable: how to do inference?
- simulate samples from joint / posterior: approximate ...
  - ... posterior cdf by empirical cdf (density: kernel dens. est.)
  - ... posterior expectation by sample mean

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, ...
- joint $\propto$ posterior usually intractable: how to do inference?
- simulate samples from joint / posterior: approximate ...
  - ... posterior cdf by empirical cdf (density: kernel dens. est.)
  - ... posterior expectation by sample mean
  - ... any function of posterior parameters by sample equivalent

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, ...
- joint $\propto$ posterior usually intractable: how to do inference?
- simulate samples from joint / posterior: approximate ...
  - ...posterior cdf by empirical cdf (density: kernel dens. est.)
  - ...posterior expectation by sample mean
  - ...any function of posterior parameters by sample equivalent
- first: quick look at sampling from univariate distributions

Simulation & Markov Chain Monte Carlo: What? Why?

- BNs allow us to formulate complex models
  - complex variance structures, ...
- joint $\propto$ posterior usually intractable: how to do inference?
- simulate samples from joint / posterior: approximate ...
  - ... posterior cdf by empirical cdf (density: kernel dens. est.)
  - ... posterior expectation by sample mean
  - ... any function of posterior parameters by sample equivalent
- first: quick look at sampling from univariate distributions
- then: MCMC for sampling from multivariate distributions

Monte Carlo Estimation: Why does it work?

- want to estimate $\mathrm{E}\left(g(X)\right) = \int g(x)\, f(x \mid \ldots)\, dx$

Monte Carlo Estimation: Why does it work?

- want to estimate $\mathrm{E}\left(g(X)\right) = \int g(x)\, f(x \mid \ldots)\, dx$
- Monte Carlo sample $x_1, \ldots, x_M$ (M samples drawn from $f(x \mid ...)$)

Monte Carlo Estimation: Why does it work?

- want to estimate $\mathrm{E}\left(g(X)\right) = \int g(x)\, f(x \mid \ldots)\, dx$
- Monte Carlo sample $x_1, \ldots, x_M$ (M samples drawn from $f(x \mid ...)$)
- estimate $\mathrm{E}\left(g(X)\right)$ by $\mathrm{E}\widehat{\left(g(X)\right)} = \dfrac{1}{M}\sum_{i=1}^{M} g(x_i)$

Monte Carlo Estimation: Why does it work?

- want to estimate $\mathrm{E}\left(g(X)\right) = \int g(x)\, f(x \mid \ldots)\, dx$
- Monte Carlo sample $x_1, \ldots, x_M$ (M samples drawn from $f(x \mid ...)$)
- estimate $\mathrm{E}\left(g(X)\right)$ by $\widehat{\mathrm{E}\left(g(X)\right)} = \dfrac{1}{M} \sum_{i=1}^{M} g(x_i)$
- unbiased: $\mathrm{E}\left(\widehat{\mathrm{E}\left(g(X)\right)}\right) = \mathrm{E}\left(g(X)\right)$

Monte Carlo Estimation: Why does it work?

- want to estimate $\mathrm{E}\,(g(X)) = \int g(x)\, f(x \mid \ldots)\, dx$
- Monte Carlo sample $x_1, \ldots, x_M$ (M samples drawn from $f(x \mid ...)$)
- estimate $\mathrm{E}\,(g(X))$ by $\mathrm{E}\,\widehat{(g(X))} = \dfrac{1}{M} \sum_{i=1}^{M} g(x_i)$
- unbiased: $\mathrm{E}\left(\mathrm{E}\,\widehat{(g(X))}\right) = \mathrm{E}\,(g(X))$
- variance: $\mathrm{Var}\left(\mathrm{E}\,\widehat{(g(X))}\right) = \dfrac{1}{M}\mathrm{Var}\,(g(X))$
  for independent samples only!

Monte Carlo Estimation: Why does it work?

- want to estimate $\mathrm{E}\,(g(X)) = \int g(x)\, f(x \mid \ldots)\, dx$
- Monte Carlo sample $x_1, \ldots, x_M$ (M samples drawn from $f(x \mid \ldots)$)
- estimate $\mathrm{E}\,(g(X))$ by $\mathrm{E}\,\widehat{(g(X))} = \frac{1}{M} \sum_{i=1}^{M} g(x_i)$
- unbiased: $\mathrm{E}\left(\mathrm{E}\,\widehat{(g(X))}\right) = \mathrm{E}\,(g(X))$
- variance: $\mathrm{Var}\left(\mathrm{E}\,\widehat{(g(X))}\right) = \frac{1}{M} \mathrm{Var}\,(g(X))$
  for independent samples only!
- precision of MC estimate increases with M, independent of parameter dimension! (numeric integration: number of evaluation points increases exponentially with dimension)

Monte Carlo Estimation: Why does it work?

- want to estimate $\mathrm{E}\left(g(X)\right) = \int g(x)\, f(x \mid \ldots)\, dx$
- Monte Carlo sample $x_1, \ldots, x_M$ (M samples drawn from $f(x \mid ...)$)
- estimate $\mathrm{E}\left(g(X)\right)$ by $\widehat{\mathrm{E}\left(g(X)\right)} = \dfrac{1}{M}\sum_{i=1}^{M} g(x_i)$
- unbiased: $\mathrm{E}\left(\widehat{\mathrm{E}\left(g(X)\right)}\right) = \mathrm{E}\left(g(X)\right)$
- variance: $\mathrm{Var}\left(\widehat{\mathrm{E}\left(g(X)\right)}\right) = \dfrac{1}{M}\mathrm{Var}\left(g(X)\right)$
  for independent samples only!
- precision of MC estimate increases with M, independent of parameter dimension! (numeric integration: number of evaluation points increases exponentially with dimension)
- $\lim_{M \to \infty} \widehat{\mathrm{E}\left(g(X)\right)} \xrightarrow{\text{a.s.}} \mathrm{E}\left(g(X)\right)$ (strong law of large numbers)

Monte Carlo Estimation: Why does it work?

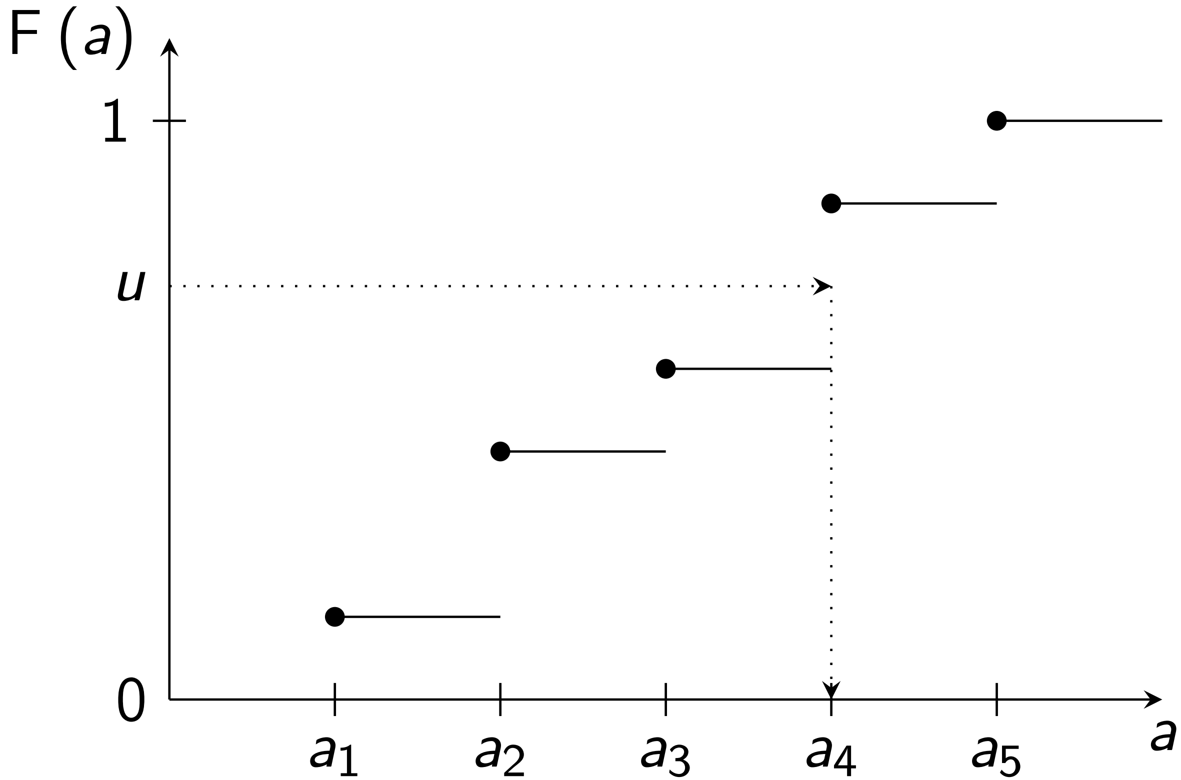
- want to estimate $\mathrm{E}\left(g(X)\right) = \int g(x)\, f(x \mid \ldots)\, dx$
- Monte Carlo sample $x_1, \ldots, x_M$ (M samples drawn from $f(x \mid \ldots)$)
- estimate $\mathrm{E}\left(g(X)\right)$ by $\widehat{\mathrm{E}\left(g(X)\right)} = \frac{1}{M} \sum_{i=1}^{M} g(x_i)$
- unbiased: $\mathrm{E}\left(\widehat{\mathrm{E}\left(g(X)\right)}\right) = \mathrm{E}\left(g(X)\right)$
- variance: $\mathrm{Var}\left(\widehat{\mathrm{E}\left(g(X)\right)}\right) = \frac{1}{M} \mathrm{Var}\left(g(X)\right)$
  for independent samples only!
- precision of MC estimate increases with M, independent of parameter dimension! (numeric integration: number of evaluation points increases exponentially with dimension)
- $\lim\limits_{M \to \infty} \widehat{\mathrm{E}\left(g(X)\right)} \xrightarrow{\text{a.s.}} \mathrm{E}\left(g(X)\right)$ (strong law of large numbers)
- $\widehat{\mathrm{E}\left(g(X)\right)} \overset{\text{a.s.}}{\sim} \mathrm{N}\left(\mathrm{E}\left(g(X)\right), \frac{1}{M}\mathrm{Var}\left(g(X)\right)\right)$ (central limit thm)

Simulation & MCMC: Univariate Sampling

- assumption for all sampling algorithms:
  we can sample from the uniform $U([0, 1])$
- done by pseudo-random number generator (PNRG), in **R**: `?RNG`

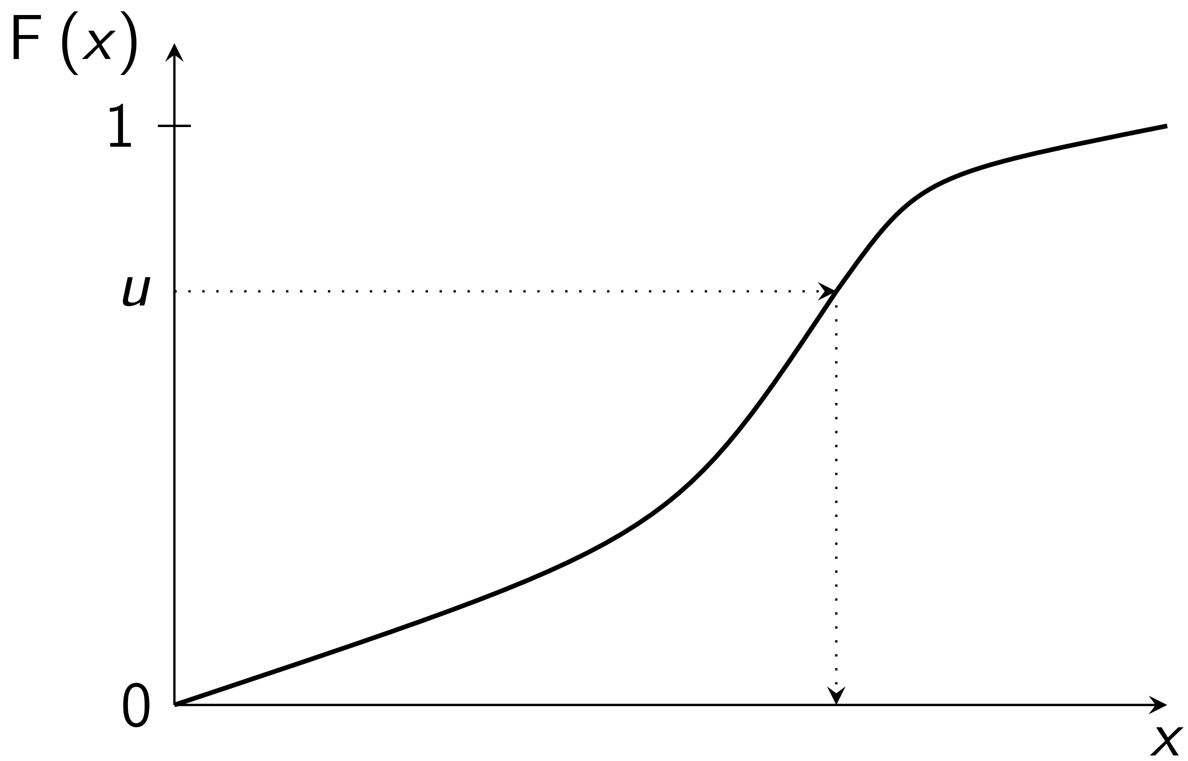
Simulation & MCMC: Univariate Sampling

- assumption for all sampling algorithms:
  we can sample from the uniform $U([0,1])$
- done by pseudo-random number generator (PNRG), in **R**: `?RNG`



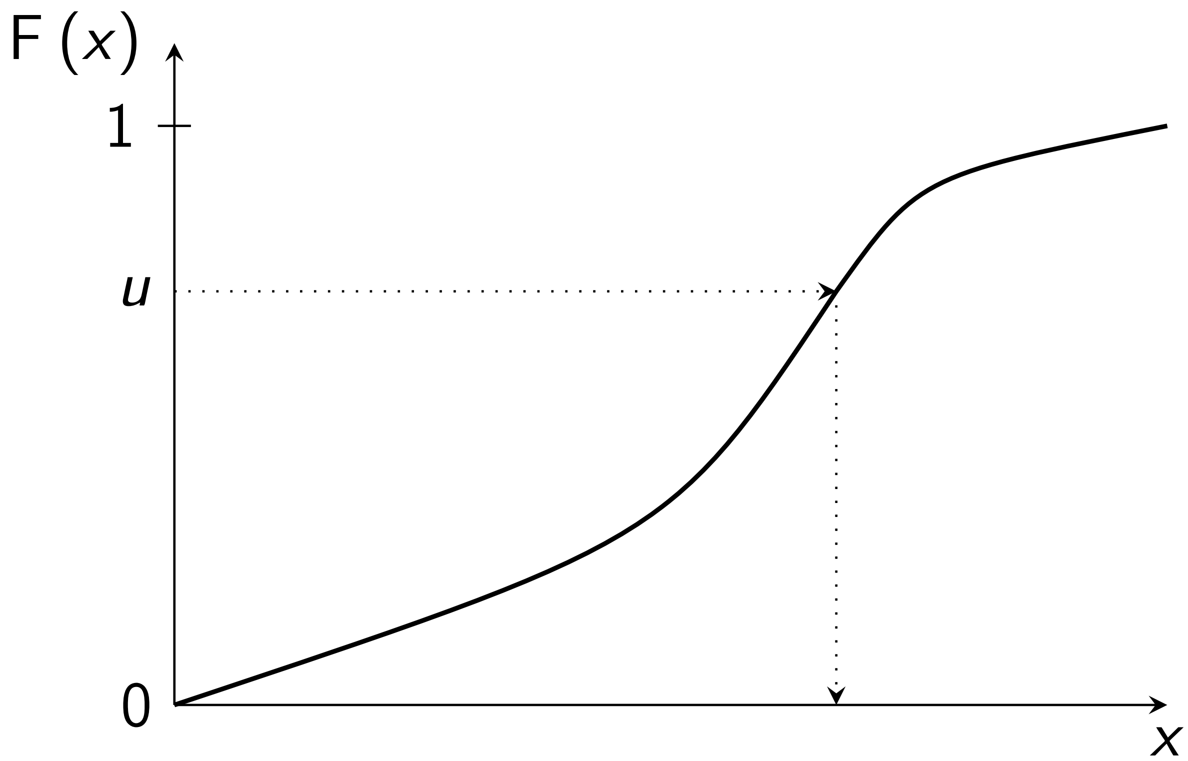
Simulation & MCMC: Univariate Sampling

- assumption for all sampling algorithms:
  we can sample from the uniform $U([0, 1])$
- done by pseudo-random number generator (PNRG), in **R**: `?RNG`



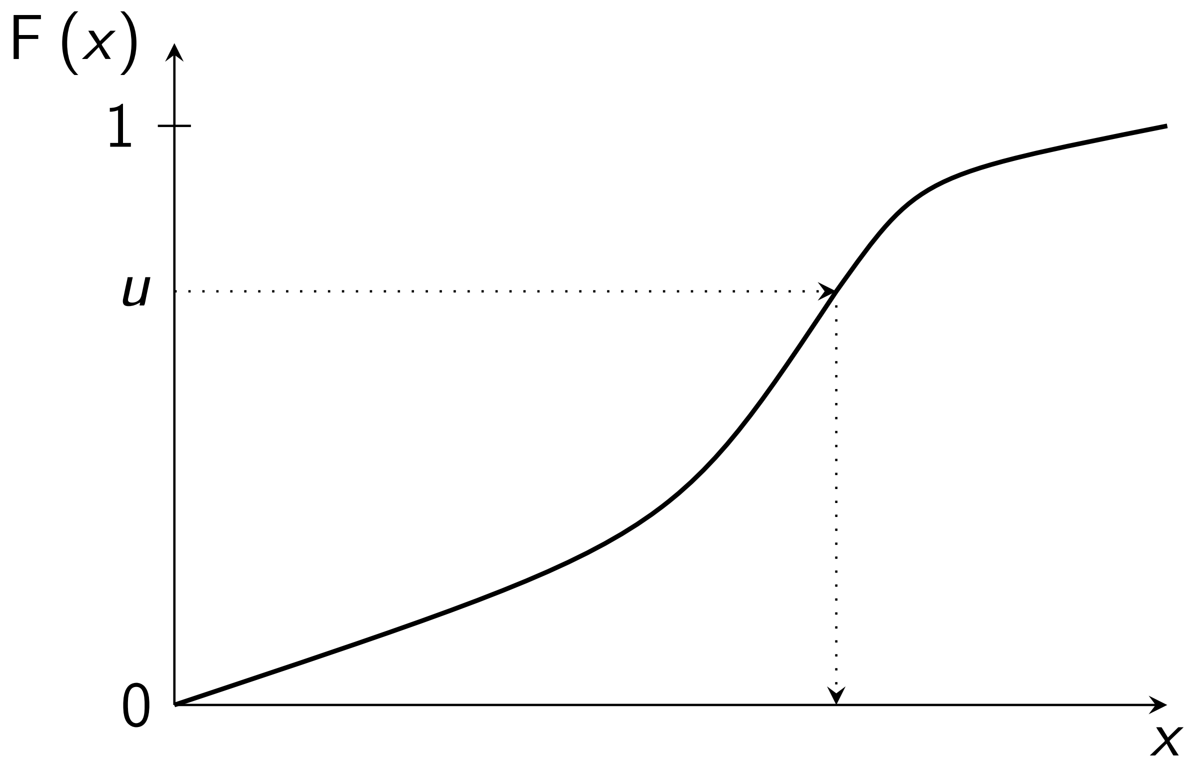
Simulation & MCMC: Univariate Sampling

- assumption for all sampling algorithms:
  we can sample from the uniform $U([0,1])$
- done by pseudo-random number generator (PNRG), in **R**: `?RNG`
- does not work well in dimensions > 1



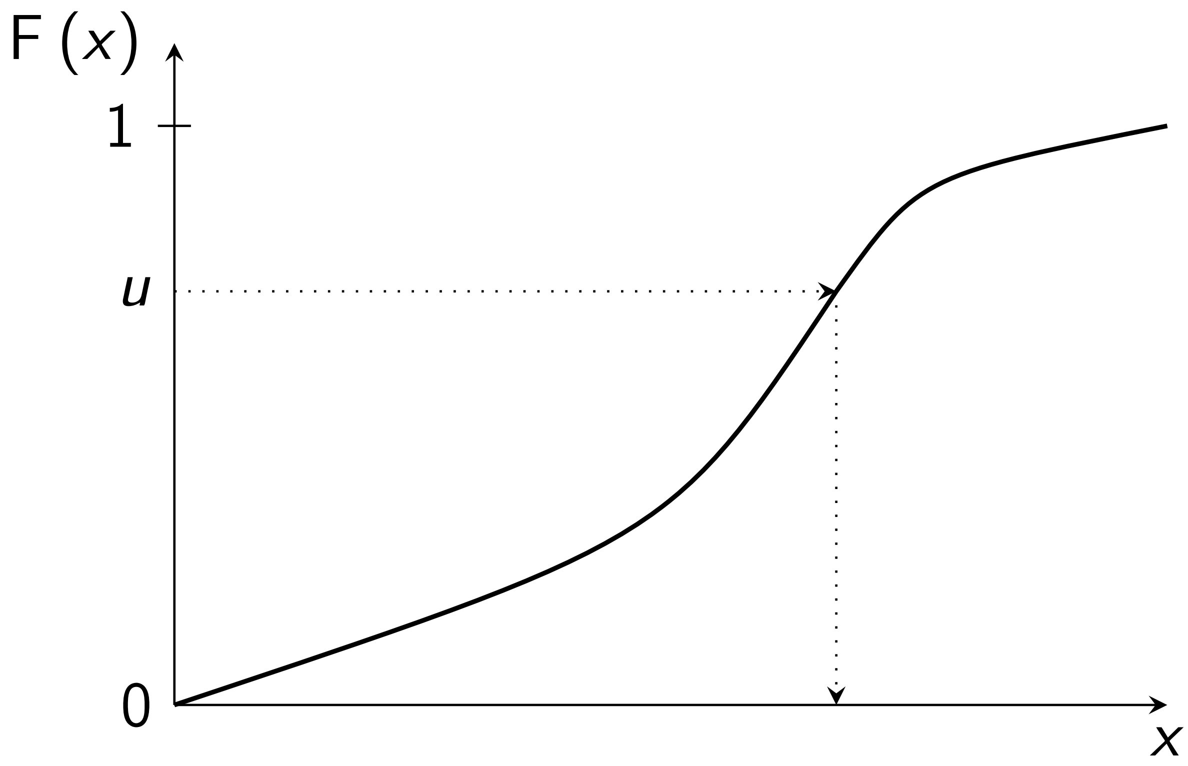
Simulation & MCMC: Univariate Sampling

- assumption for all sampling algorithms:
  we can sample from the uniform $U([0,1])$
- done by pseudo-random number generator (PNRG), in **R**: `?RNG`
- does not work well in dimensions > 1
- needs $F^{-1}(\cdot)$

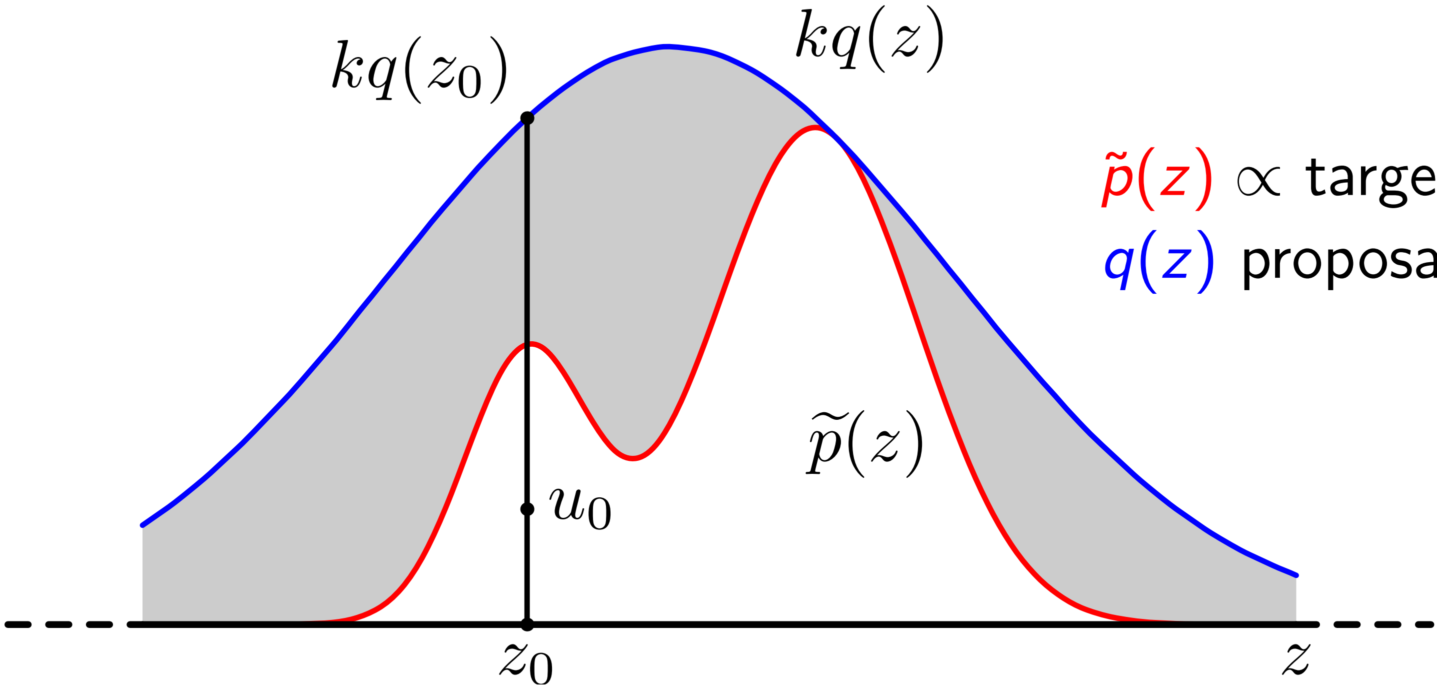


Simulation & MCMC: Univariate Sampling

- assumption for all sampling algorithms: we can sample from the uniform $U([0,1])$
- done by pseudo-random number generator (PNRG), in **R**: `?RNG`
- does not work well in dimensions > 1
- needs $F^{-1}(\cdot)$
- needs normalization factor
- rejection sampling



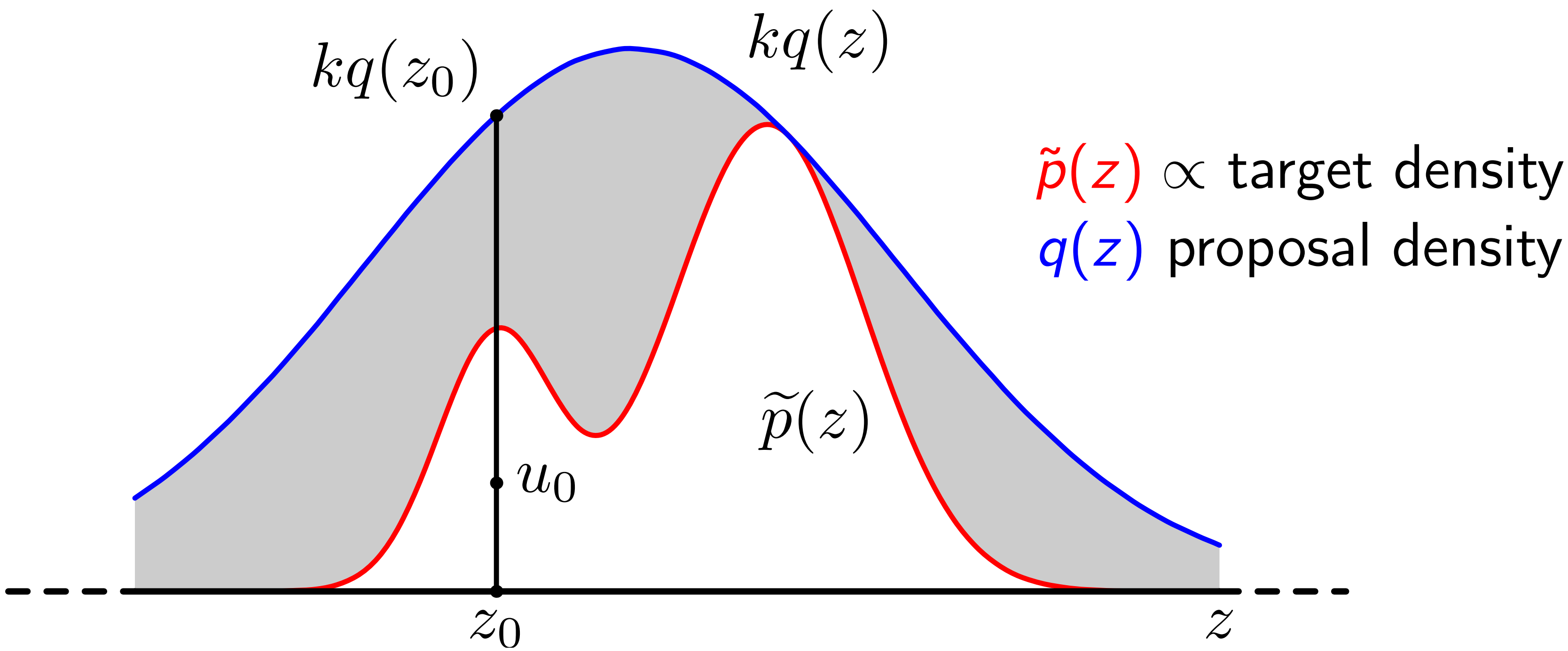
Simulation & MCMC: Rejection Sampling



$\tilde{p}(z) \propto$ target density

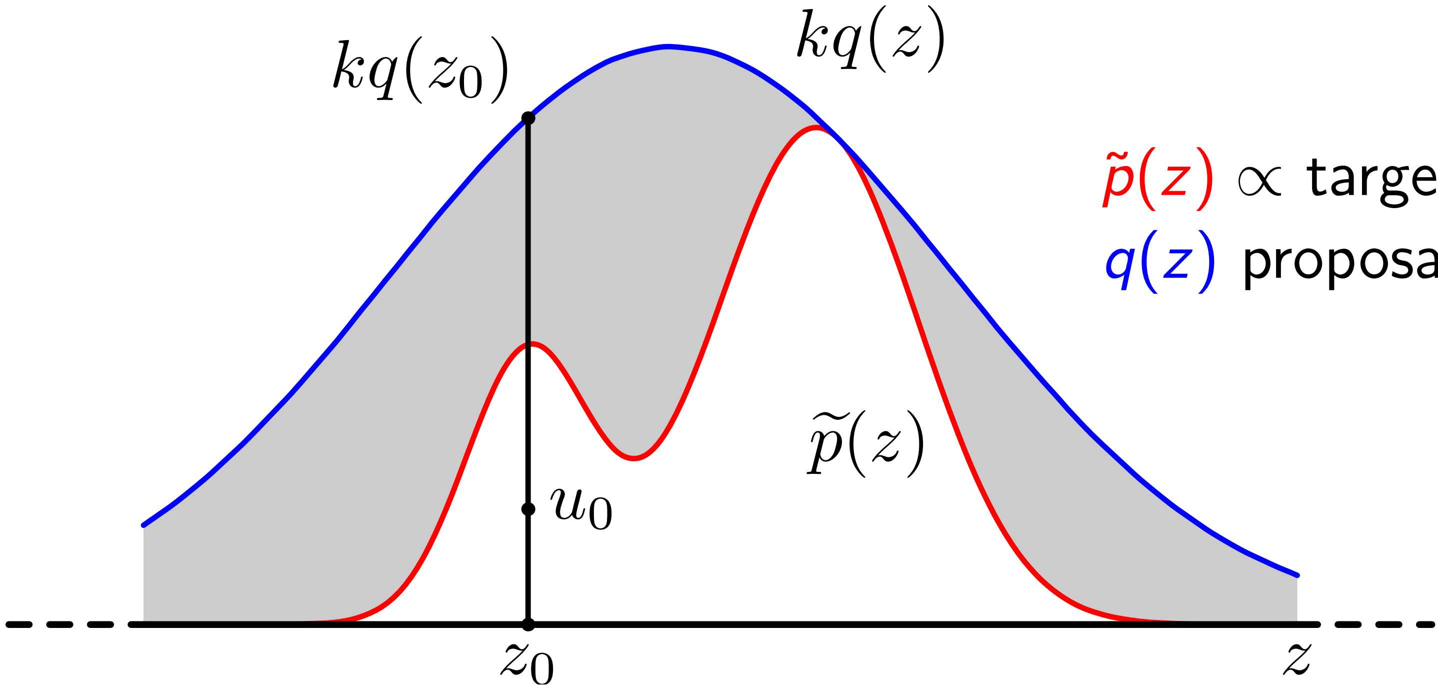
$q(z)$ proposal density

Simulation & MCMC: Rejection Sampling



1. sample z ($\leftrightarrow$) from $q(z)$

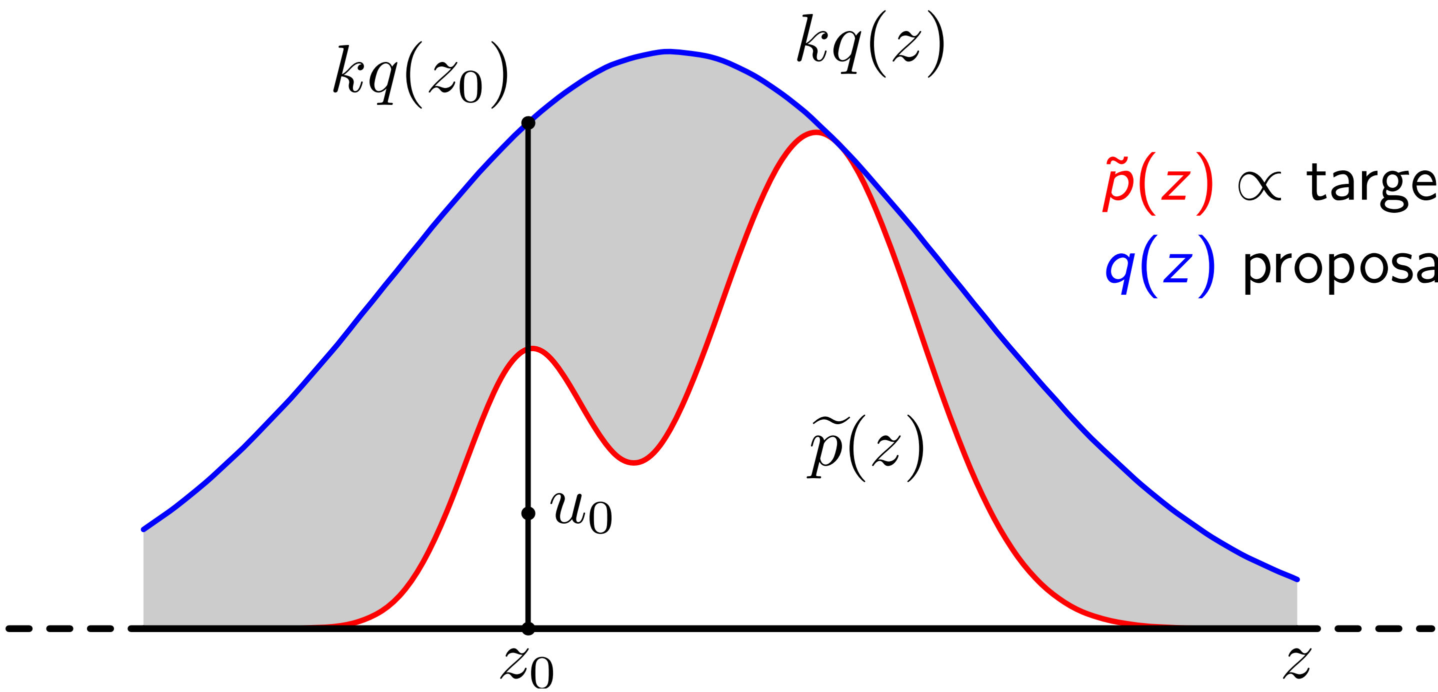
Simulation & MCMC: Rejection Sampling



$\tilde{p}(z) \propto$ target density
$q(z)$ proposal density

1. sample z ($\leftrightarrow$) from $q(z)$
2. sample u ($\updownarrow$) from $U([0, kq(z)])$

Simulation & MCMC: Rejection Sampling

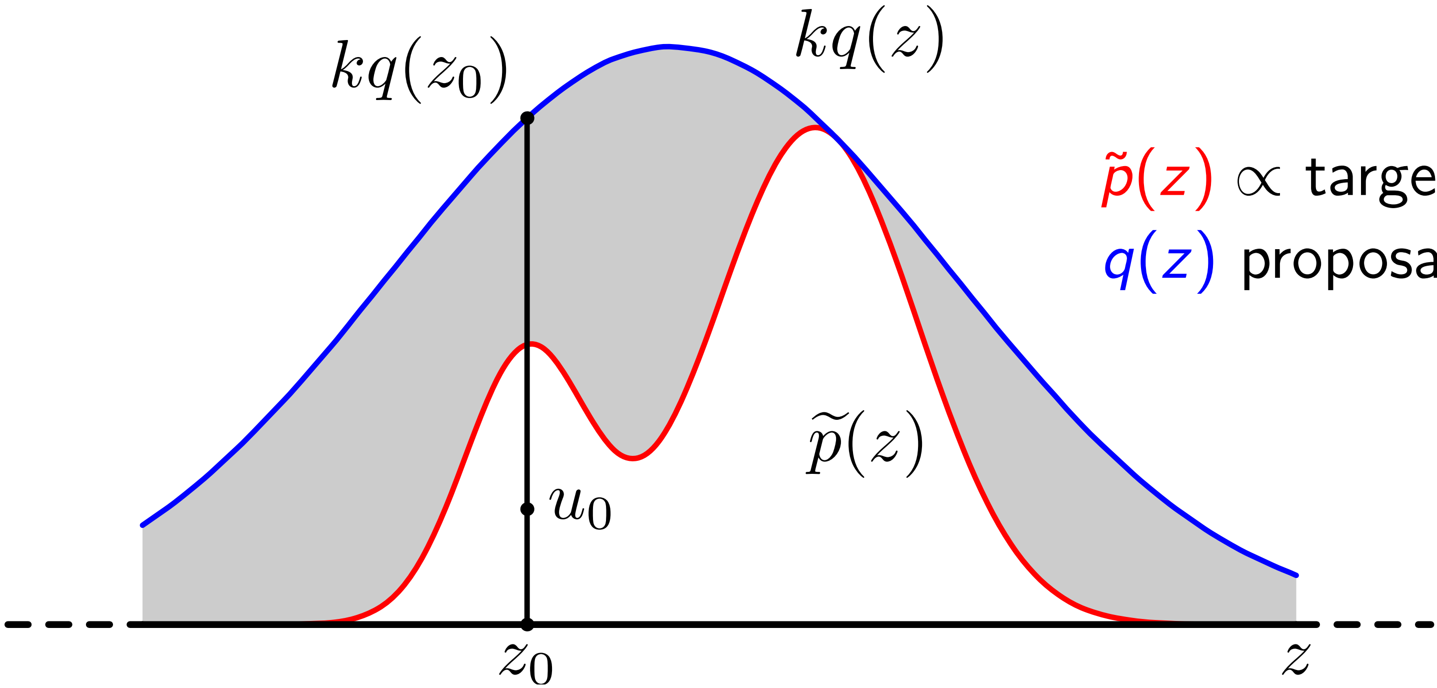


$\tilde{p}(z) \propto$ target density
$q(z)$ proposal density

1. sample z ($\leftrightarrow$) from $q(z)$
2. sample u ($\updownarrow$) from $U([0, kq(z)])$

$\Big\}$ sample points uniformly from union of white and grey areas

Simulation & MCMC: Rejection Sampling

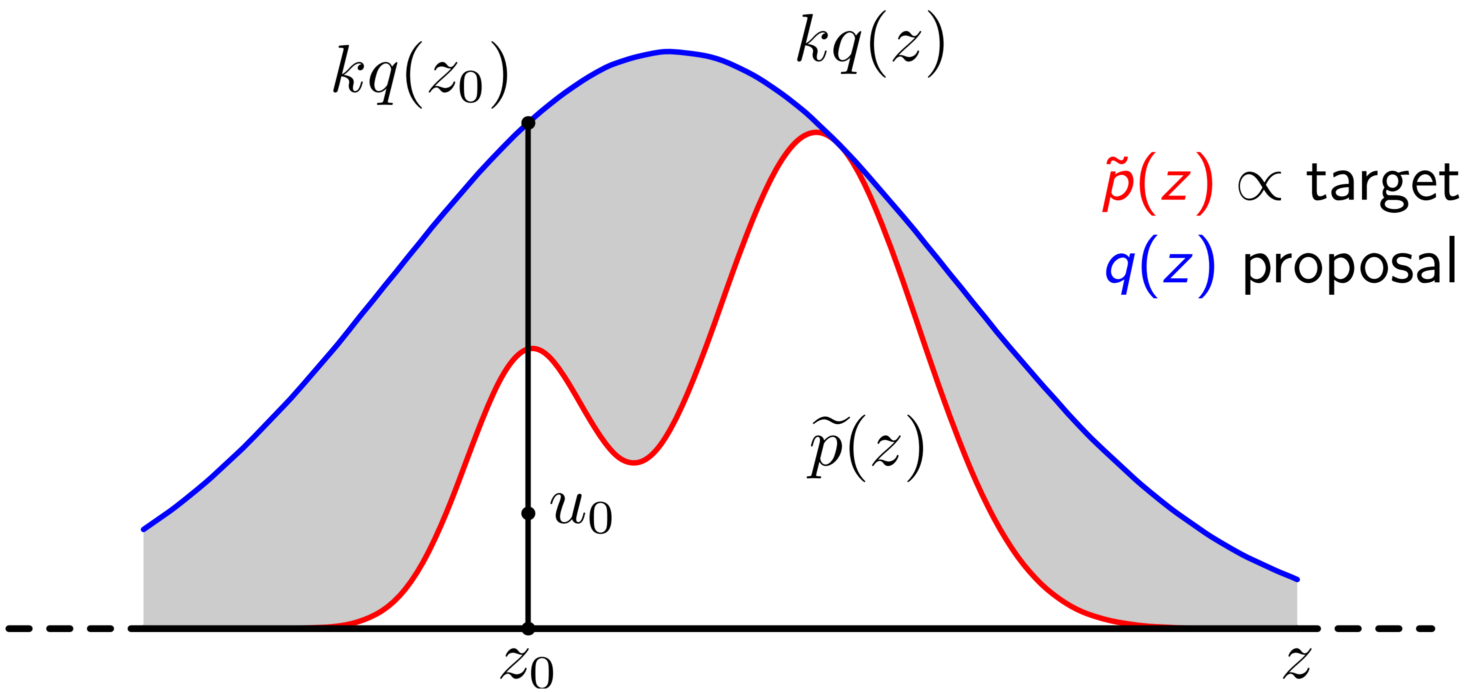


$\tilde{p}(z) \propto$ target density
$q(z)$ proposal density

1. sample z ($\leftrightarrow$) from $q(z)$
2. sample u ($\updownarrow$) from $U([0, kq(z)])$
3. reject all points in the grey area

Steps 1–2: sample points uniformly from union of white and grey areas

Simulation & MCMC: Rejection Sampling



$\tilde{p}(z) \propto$ target density
$q(z)$ proposal density

1. sample z ($\leftrightarrow$) from $q(z)$
2. sample u ($\updownarrow$) from $U([0, kq(z)])$

$\Big\}$ sample points uniformly from union of white and grey areas

3. reject all points in the grey area
4. forget about u: z distributed $\propto \tilde{p}(z)$!

Markov Chain Monte Carlo: General Idea

- need to sample from high-dimensional distributions

Markov Chain Monte Carlo: General Idea

- need to sample from high-dimensional distributions
- idea: produce samples by a Markov Chain:
  random walk over parameter space

Markov Chain Monte Carlo: General Idea

- need to sample from high-dimensional distributions
- idea: produce samples by a Markov Chain:
  random walk over parameter space
- random walk spends more time in high-probability regions

Markov Chain Monte Carlo: General Idea

- need to sample from high-dimensional distributions
- idea: produce samples by a Markov Chain:
  random walk over parameter space
- random walk spends more time in high-probability regions
- if in each step we move in one dimension only:
  need to sample from one-dimensional distribution only,
  can use previous algorithms for that!

Markov Chain Monte Carlo: General Idea

- need to sample from high-dimensional distributions
- idea: produce samples by a Markov Chain:
  random walk over parameter space
- random walk spends more time in high-probability regions
- if in each step we move in one dimension only:
  need to sample from one-dimensional distribution only,
  can use previous algorithms for that!
- but: samples are not independent!

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:
  - propose a step
    (draw from easy-to-sample-from *proposal distribution*)

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:
  - propose a step
    (draw from easy-to-sample-from *proposal distribution*)
  - accept step with certain probability
    (tailored to make chain approach the target distribution)

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:
  - propose a step
    (draw from easy-to-sample-from *proposal distribution*)
  - accept step with certain probability
    (tailored to make chain approach the target distribution)
  - `Stan` uses an improved variant called Hamiltonian MH

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:
  - propose a step
    (draw from easy-to-sample-from *proposal distribution*)
  - accept step with certain probability
    (tailored to make chain approach the target distribution)
  - `Stan` uses an improved variant called Hamiltonian MH
- Gibbs sampler:

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:
  - propose a step
    (draw from easy-to-sample-from *proposal distribution*)
  - accept step with certain probability
    (tailored to make chain approach the target distribution)
  - `Stan` uses an improved variant called Hamiltonian MH

- Gibbs sampler:
  - loop over parameter vector $(\theta_1, \theta_2, \ldots)$

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:
  - propose a step
    (draw from easy-to-sample-from *proposal distribution*)
  - accept step with certain probability
    (tailored to make chain approach the target distribution)
  - `Stan` uses an improved variant called Hamiltonian MH
- Gibbs sampler:
  - loop over parameter vector $(\theta_1, \theta_2, \ldots)$
  - draw from the *full conditionals* $f(\theta_i \mid \text{everything else}) \propto$ joint

Markov Chain Monte Carlo: Algorithms

- Metropolis-Hastings:
  - propose a step
    (draw from easy-to-sample-from *proposal distribution*)
  - accept step with certain probability
    (tailored to make chain approach the target distribution)
  - `Stan` uses an improved variant called Hamiltonian MH

- Gibbs sampler:
  - loop over parameter vector $(\theta_1, \theta_2, \ldots)$
  - draw from the *full conditionals* $f(\theta_i \mid \text{everything else}) \propto \text{joint}$
  - special case of MH where proposals are always accepted

Markov Chain Monte Carlo: Why does this work?

Algorithms create a

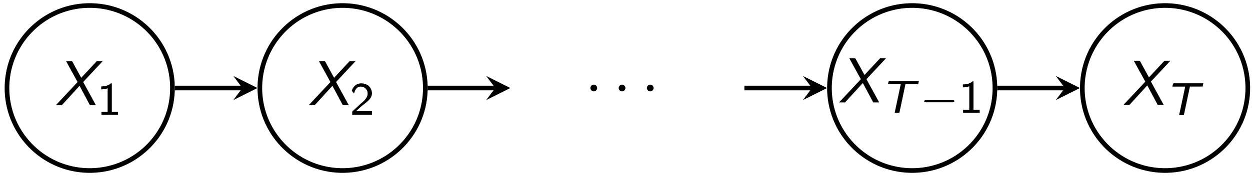
- stationary,
- irreducible and
- aperiodic

Markov chain
which has the joint as its
limiting (invariant) distribution

Markov Chain Monte Carlo: Why does this work?

Algorithms create a

- stationary,
- irreducible and
- aperiodic



Markov chain
which has the joint as its
limiting (invariant) distribution

Markov Chain Monte Carlo: Why does this work?

Algorithms create a

- **stationary**,
- irreducible and
- aperiodic

Markov chain
which has the joint as its
limiting (invariant) distribution

$$f(x_2 \mid x_1) = \cdots = f(x_n \mid x_{n-1})$$

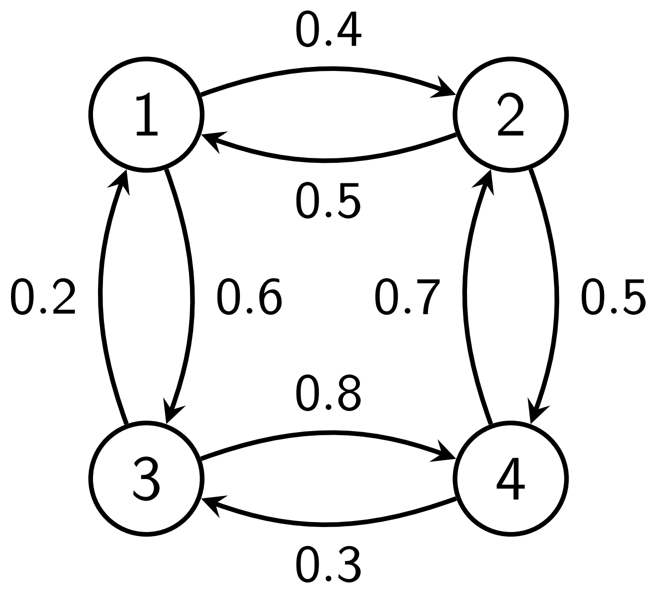
$$X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_{T-1} \rightarrow X_T$$

Markov Chain Monte Carlo: Why does this work?

Algorithms create a

- stationary,
- **irreducible** and
- aperiodic

Markov chain
which has the joint as its
limiting (invariant) distribution

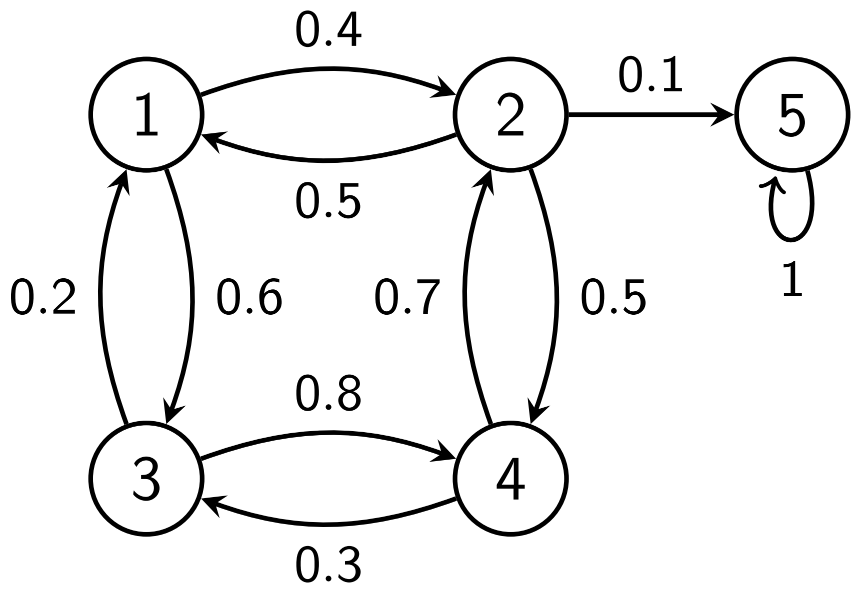


Markov Chain Monte Carlo: Why does this work?

Algorithms create a

- stationary,
- **irreducible** and
- aperiodic

Markov chain
which has the joint as its
limiting (invariant) distribution

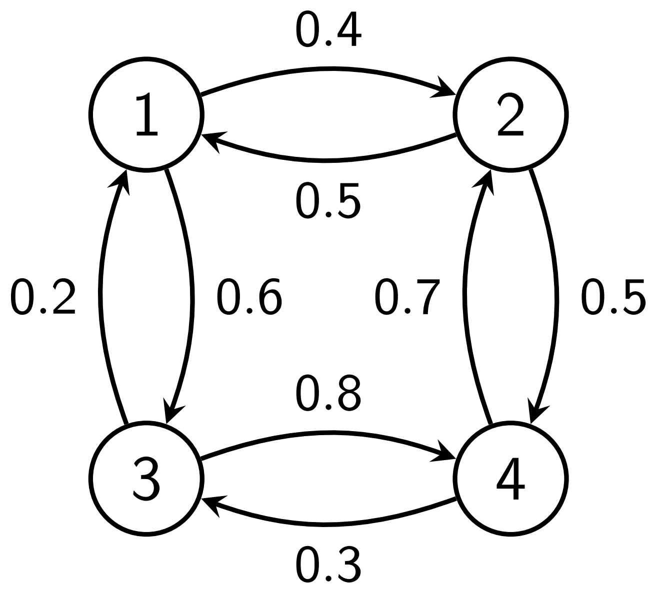


Markov Chain Monte Carlo: Why does this work?

Algorithms create a

- stationary,
- irreducible and
- **aperiodic**

Markov chain
which has the joint as its
limiting (invariant) distribution

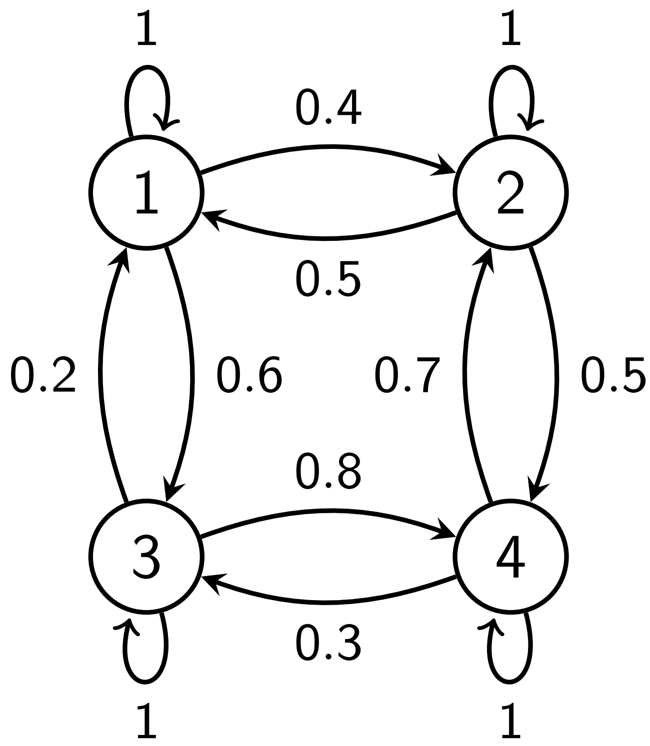


Markov Chain Monte Carlo: Why does this work?

Algorithms create a

- stationary,
- irreducible and
- aperiodic

Markov chain
which has the joint as its
limiting (invariant) distribution

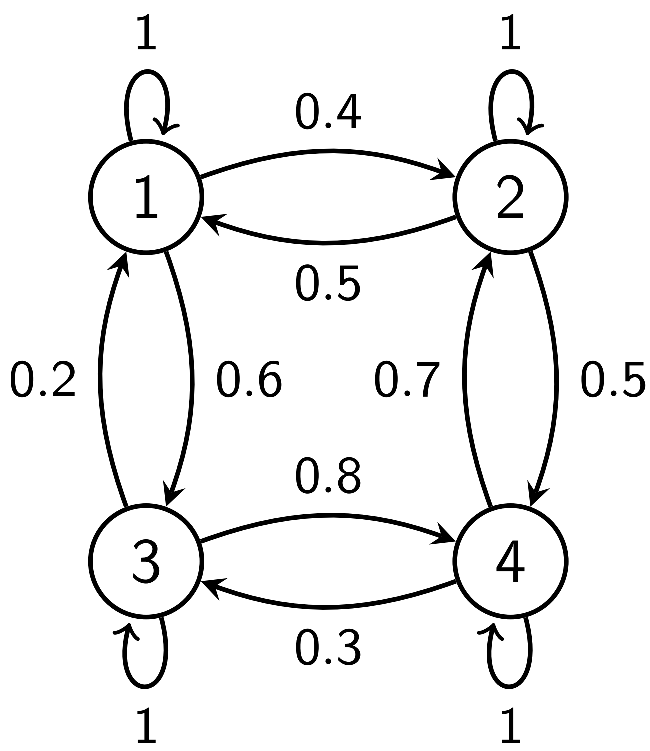


Markov Chain Monte Carlo: Why does this work?

Algorithms create a

- stationary,
- irreducible and
- aperiodic

Markov chain
which has the joint as its
limiting (invariant) distribution

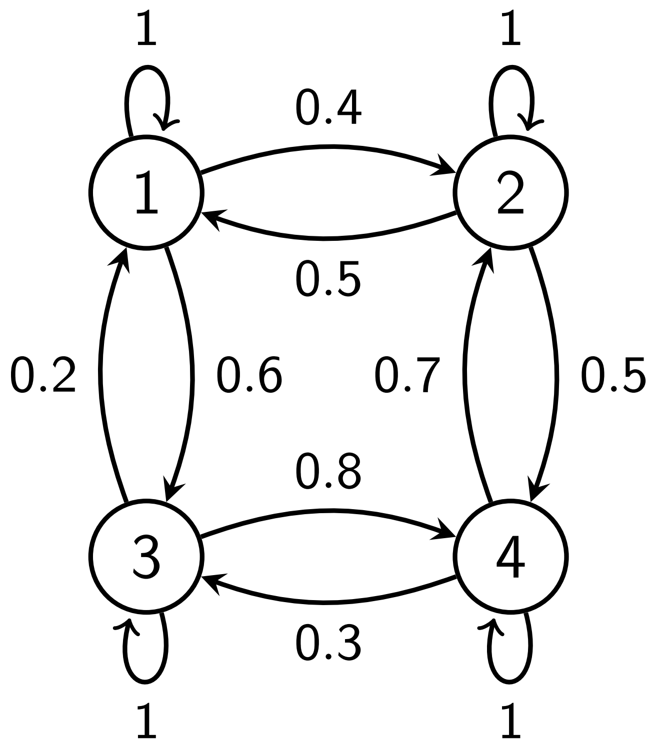


Markov Chain Monte Carlo: Why does this work?

Algorithms create a

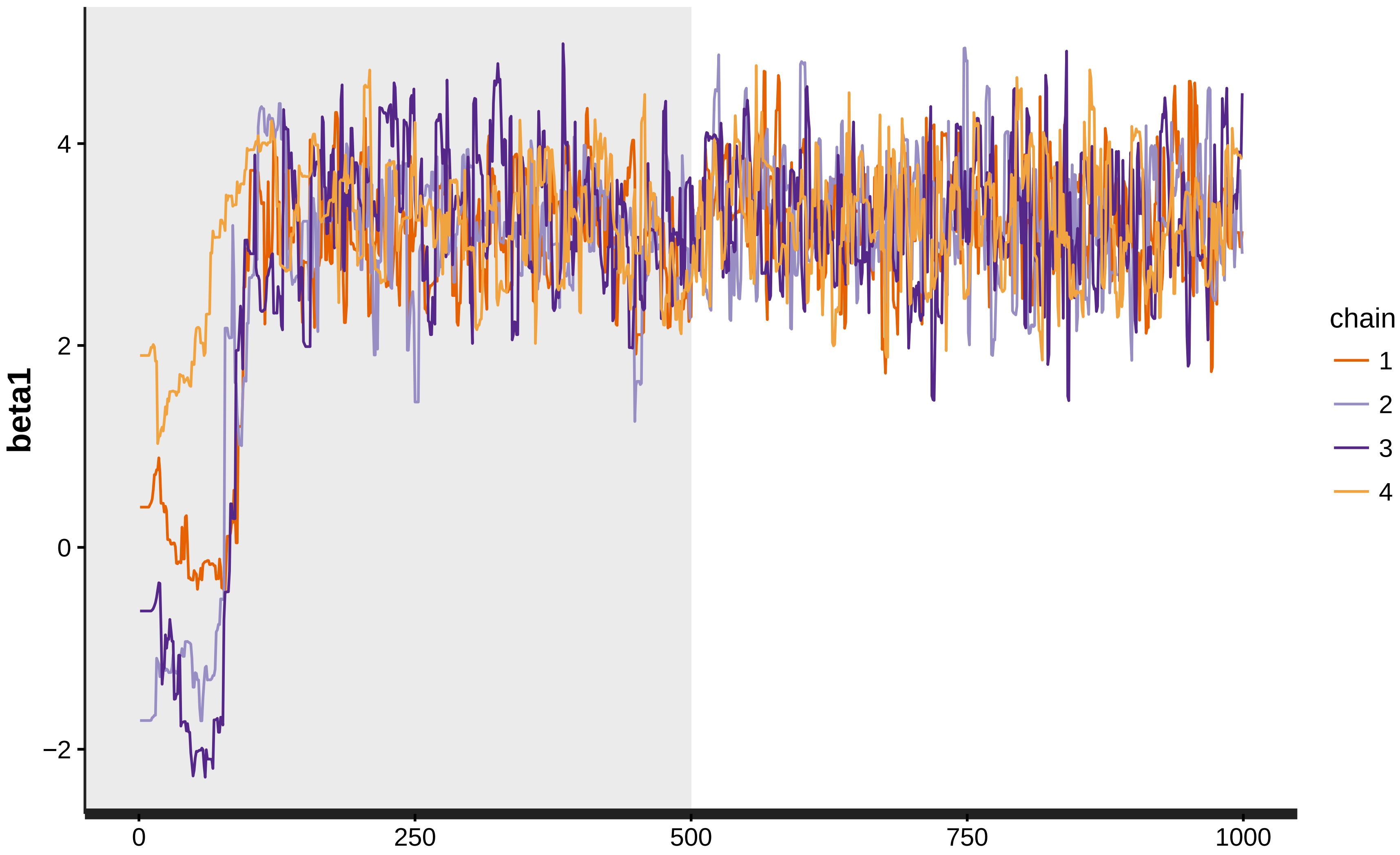
- stationary,
- irreducible and
- aperiodic

Markov chain
which has the joint as its
limiting (invariant) distribution

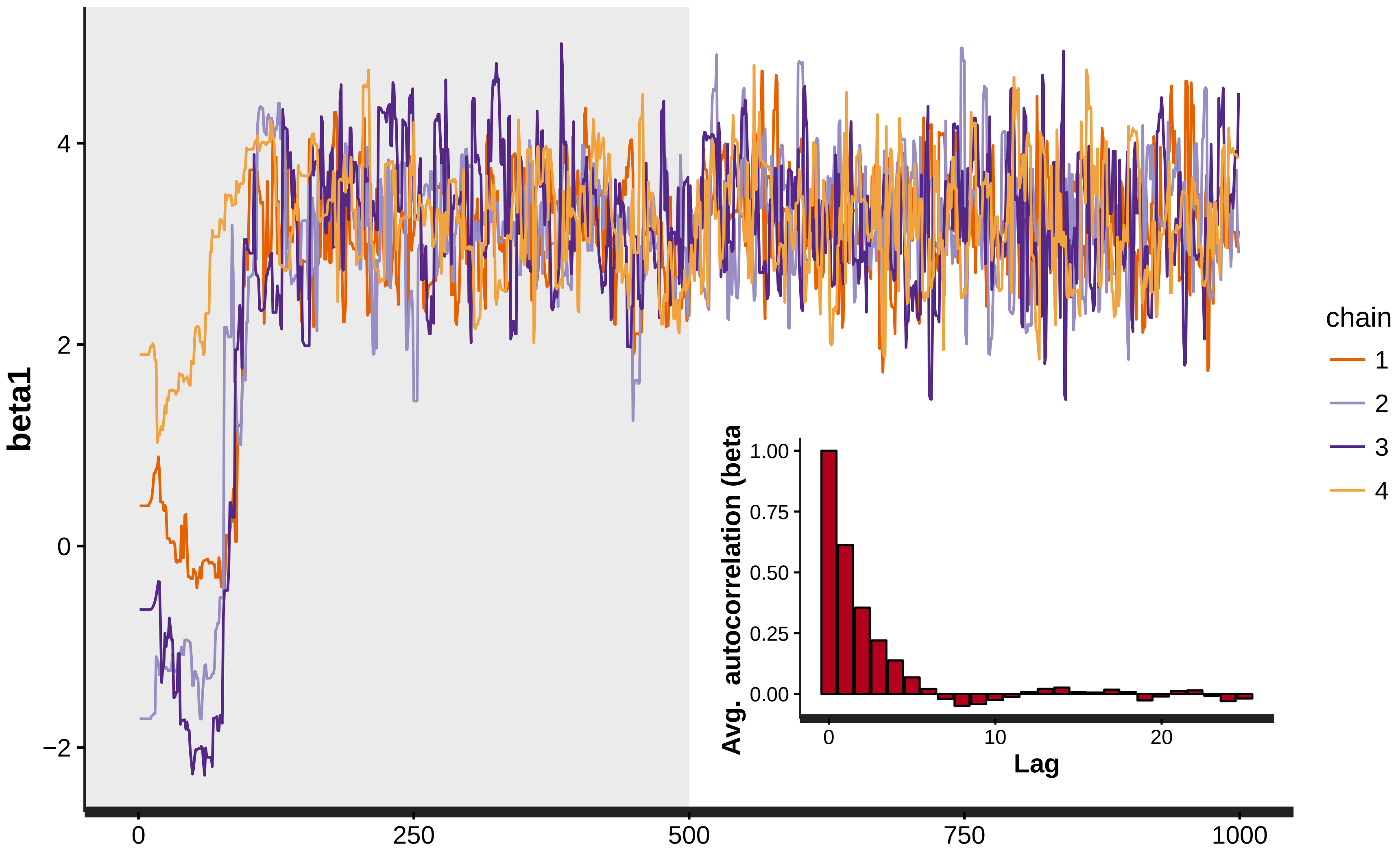


$$\big(p_1(t), p_2(t), p_3(t), p_4(t)\big) \xrightarrow{t \to \infty} \big(p_1, p_2, p_3, p_4\big)$$

MCMC: Warm-Up (= Burn-In), Mixing, Thinning



MCMC: Warm-Up (= Burn-In), Mixing, Thinning



Bayesian hierarchical modelling / Bayesian networks / graphical models

Exercises I

Simulation & MCMC

Exercises II

Exercise: Quick start RStan

A Stan model is defined by five program blocks:

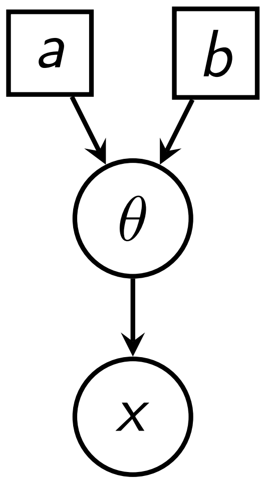
```
model1 <- "
data {
   ...
}
transformed data {
   ...
}
parameters {
   ...
}
transformed parameters {
   ...
}
model {
   ...
}
generated quantities {
   ...
}"
```

Exercise: Quick start RStan

A Stan model is defined by five program blocks:

```
model1 <- "
data {
  ...
}
transformed data {
  ...
}
parameters {
  ...              required
}
transformed parameters {
  ...
}
model {
  ...              required
}
generated quantities {
  ...
}"
```

Exercise: Quick start RStan

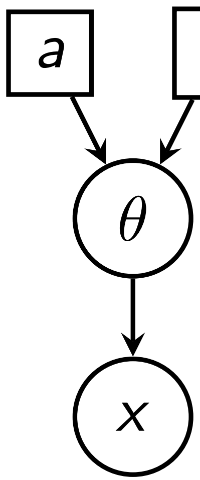


$$f(x \mid \theta) \sim \text{Binomial}(n, \theta)$$

$$f(\theta \mid a, b) \sim \text{Beta}(a, b)$$

```
library(rstan)
model0 <- "
data {
  int<lower=0> n;
  int<lower=0> x;
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  theta ~ beta(2,2);
  x ~ binomial(n, theta);
}
"
data0 <- list(n=10, x=5)
```

Exercise: Quick start RStan



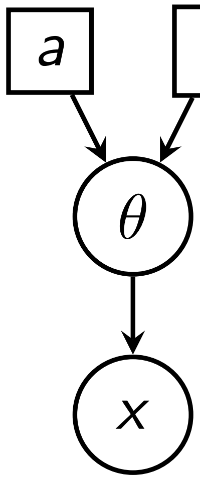
$$f(x \mid \theta) \sim \text{Binomial}(n, \theta)$$
$$f(\theta \mid a, b) \sim \text{Beta}(a, b)$$

```
library(rstan)
model0 <- "
data {
  int<lower=0> n;
  int<lower=0> x;
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  theta ~ beta(2,2);
  x ~ binomial(n, theta);
}
"
data0 <- list(n=10, x=5)
```

Running the model creates a `stanfit` object.

```
fit0 <- stan(model_code=model0, data=data0, iter=1000, chains=4)
print(fit0); plot(fit0)
```

Exercise: Quick start RStan



$$f(x \mid \theta) \sim \text{Binomial}(n, \theta)$$

$$f(\theta \mid a, b) \sim \text{Beta}(a, b)$$

```
library(rstan)
model0 <- "
data {
  int<lower=0> n;
  int<lower=0> x;
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  theta ~ beta(2,2);
  x ~ binomial(n, theta);
}
"
data0 <- list(n=10, x=5)
```

Running the model creates a `stanfit` object.

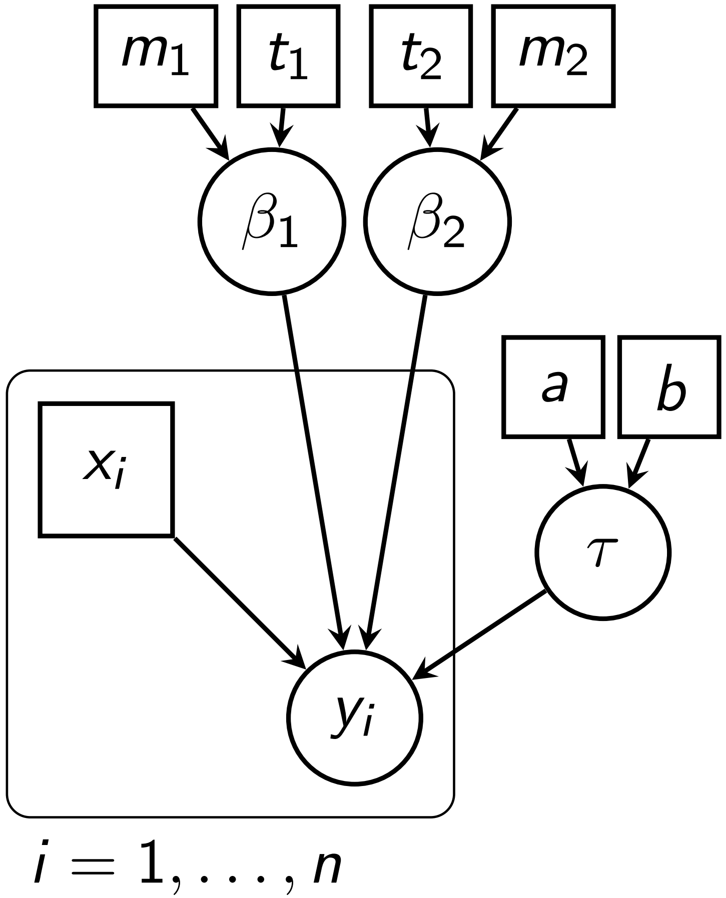
```
fit0 <- stan(model_code=model0, data=data0, iter=1000, chains=4)
print(fit0); plot(fit0)
```

The samples can be extracted by `samples0 = extract(fit0, c("theta"))`

http://mc-stan.org/documentation/

http://github.com/stan-dev/rstan/wiki/RStan-Getting-Started#how-to-use-rstan

Exercise: Linear regression in RStan



We want to estimate the parameters in the linear regression example, using RStan to sample from the posterior.

The model assumptions are:

$$
\begin{aligned}
y_i \mid \beta_1, \beta_2, \tau &\sim \mathrm{N}(\beta_1 + x_i\beta_2, 1/\tau) \\
\tau \mid a, b &\sim \mathrm{Gamma}(a, b), \quad a = b = 10^{-3} \\
\beta_1 \mid m_1, t_1 &\sim \mathrm{N}(m_1, 1/t_1), \quad m_1 = 0, t_1 = 10^4 \\
\beta_2 \mid m_2, t_2 &\sim \mathrm{N}(m_2, 1/t_2), \quad m_2 = 0, t_2 = 10^4
\end{aligned}
$$

Exercise: Linear regression in RStan

- Create an artificial data set $x_1, \ldots, x_n, y_1, \ldots, y_n$ by

  ```
  data <- list()
  data$N <- 50
  data$x <- rnorm(data$N)+30
  data$y <- 3 + 5*data$x + rnorm(data$N, sd=1/10)
  ```

  What are thus the 'true' parameter values?
- *Define the model in* `Stan`. *Include a* `transformed parameters` *block where you define $\sigma = \sqrt{1/\tau}$.* (In `Stan`, the Normal distribution is parametrized with the standard deviation σ!)
- *Simulate four chains with 1000 iterations each and use* `plot()` *and* `print()` *to get a first impression of the results. What point estimates do you get for β_1, β_2 and σ?*

Exercise: Linear regression in `RStan`

- The functions `stan_trace()`, `stan_dens()` and `stan_ac()` allow you to analyze your sample from the posterior distribution more closely. (You can include the warm-up phase in your plots by setting `inc_warmup = TRUE`.)
  How long is the warm-up phase? Do your chains mix well? Is thinning necessary?
- The function `pairs()` also works on `stanfit` objects.
  Plot pairwise scatterplots of your sample using `pairs()`.
  What do you observe about the relation between β_1 and β_2?

Exercise: Linear regression in RStan

- The high correlation between β_1 and β_2 indicates that the Markov chain cannot move around freely. You can mitigate this problem by centering the data $x_1, \ldots, x_n$. The mean for the Normal distribution of y_i is then given by $\beta_1^c + \beta_2(x_i - \bar{x})$, where $\beta_1^c = \beta_1 + \beta_2\bar{x}$.
  Add the following block to your `stan` *model definition,*

  ```
  transformed data {
    vector[N] xcentered;
    xcentered=x-mean(x);
  }
  ```

  and edit the `parameters` *and* `model` *blocks such that the model generates samples from β_1^c instead of β_1.*
- *Edit the* `transformed parameters` *block to define β_1 as $\beta_1 = \beta_1^c + \beta_2\bar{x}$.*
- *Simulate four chains with 1000 iterations each from this new model, and analyze your sample from the posterior distribution like for the first model. What has changed?*

Exercise: Linear regression in `RStan`

- *Choose an informative prior for one or both of β_1 and β_2. Try out different values for mean and standard deviation. What is the effect on the chains and the posterior densities?*