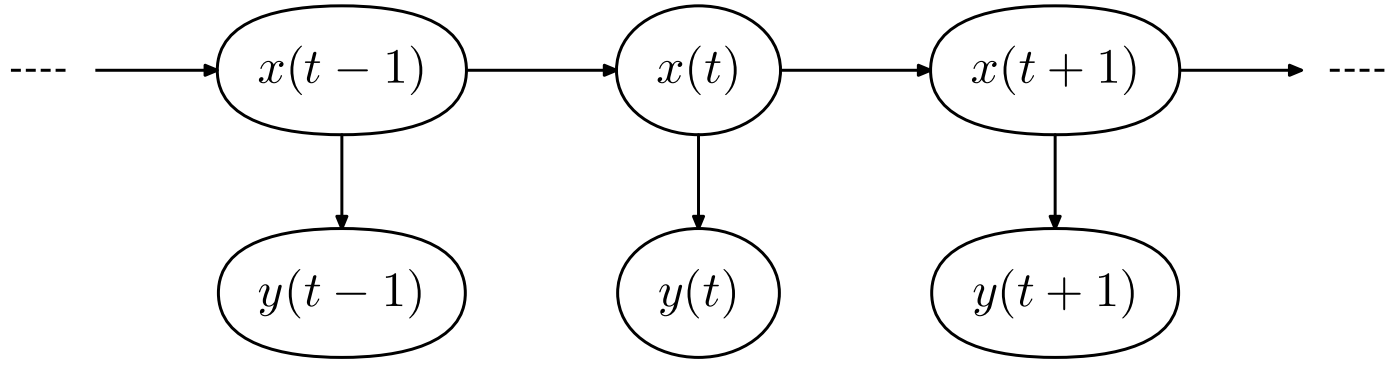


Specifying Graphical Models in `RevBayes`

Will Freyman

Department of Ecology, Evolution & Behavior
University of Minnesota

willfreyman@gmail.com
http://willfreyman.org

UC Berkeley Workshop, February 26-27 2018

Workshop goals:

- not simply demonstrate "standard" phylogenetic analyses in `RevBayes`
- instead we'll explore the flexibility of a graphical modeling framework
- use graphical models to see how the models underlying tree inference and downstream tree use (comparative methods) are linked
- enable participants to specify custom and unique phylogenetic analyses in `RevBayes`

What are we doing in phylogenetics?

- "inference"? (i.e. statistical inference?)
- "learning"? (i.e. machine learning?)
- "prediction"?

In a probabilistic framework (maximum likelihood or Bayesian) inference and learning are the same

When we "train" a machine learning algorithm we are doing parameter estimation

The field of machine learning includes camps that use principled probabilistic approaches and camps that use heuristic ad hoc methods (like in phylogenetics!)

In phylogenetics we should be doing more prediction!
i.e. model adequacy/posterior predictive tests

What is a model?

- in statistics?
- in machine learning?
- in biology?

maybe:

- a way to relate data to hypotheses?
  - what about heuristic or ad hoc approaches?
  - is parsimony in phylogenetics a model, an algorithm, or a philosophy?
- a set of assumptions about the data-generating process?
- "a formal representation of a theory"?
- a set of mathematical equations that relate one or more random variables?

The distinction between models and algorithms:

model	algorithm
phylogenetic models	pruning algorithm
Bayesian graphical model	belief propagation
neural networks	backpropagation
Hidden Markov model	forward-backward
k-means clustering	Lloyd's algorithm
linear regression	least-squares

Learning algorithms typically either optimize $\hat{\theta}$ or integrate to infer $p(\theta|D)$

They are often *very similar* and can be used with other models

Any well defined model can be treated in a *probabilistic* framework and then we can use Bayesian or maximum likelihood approaches

Probabilistic models:

Instead of a hodgepodge of different heuristic methods these models use the principles of probability theory

Why use them?

- Quantify uncertainty: they know when they don't know
  - what is the best prediction/decision/inference given data?
  - what is the best model/hypothesis given the data?
  - do I need more/different data?
- natural complexity control
  - preventing overfitting / regularization
- modularity
  - models as "lego kits"
  - different inferential algorithms can use the same model
  - different models can use the same inferential algorithm

Discriminative vs Generative Models

Discriminative (or conditional) models:

1. models a response variable conditioned on a predictor variable
2. models the conditional distribution $p(y|x)$
3. makes fewer assumptions about the data: $p(x)$ not necessary

Phylogenetic examples:

- estimating divergence times over a fixed topology
- estimating ancestral states on a fixed tree
- estimating shifts in diversification rates over a fixed tree

Discriminative vs Generative Models

Generative models:

1. models the entire process used to generate the data
2. models the joint distribution $p(x, y)$
3. makes more assumptions about the data: need to define $p(x)$
4. richer representation of the relations between variables
5. more powerful: allows us to compute $p(y|x)$ or $p(x|y)$
6. more powerful: can simulate both x and y

Phylogenetic examples:

- jointly estimating divergence times and the tree topology
- jointly estimating ancestral states and the tree
- jointly estimating shifting diversification rates and the tree

What is a graphical model?

Also called:

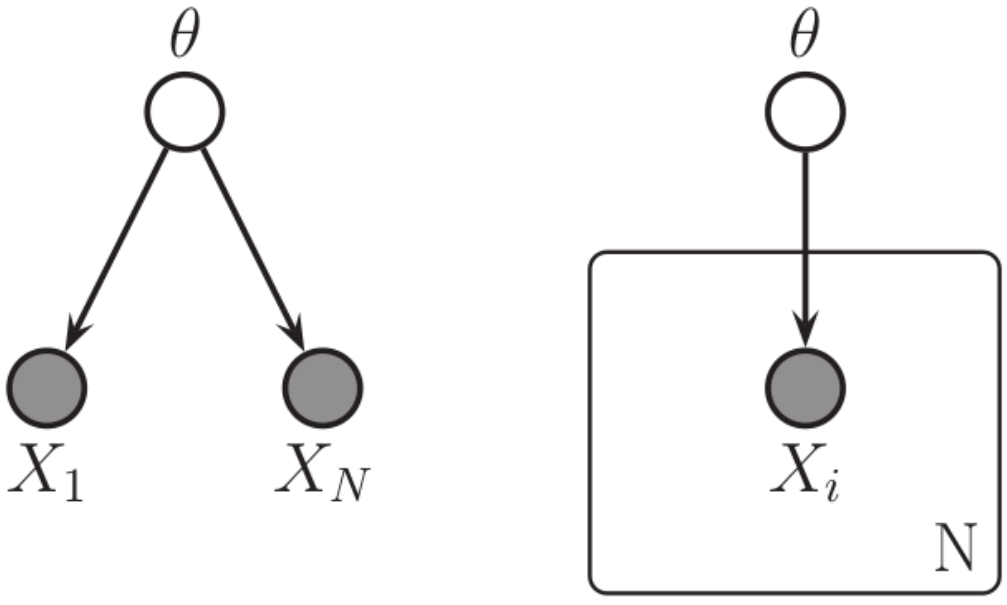
1. Bayesian networks
2. belief networks
3. causal networks

A useful way to represent a probabilistic model: a joint distribution of random variables.

We can specify both generative and discriminative models as graphical models.

What is a graphical model?

Nodes represent variables and edges represent conditional dependencies:



$$p(\theta, \mathcal{D}) = p(\theta)\Big[\prod_{i=1}^{N} p(x_i|\theta)\Big]$$

Image from Murphy (2012)

What is a graphical model?

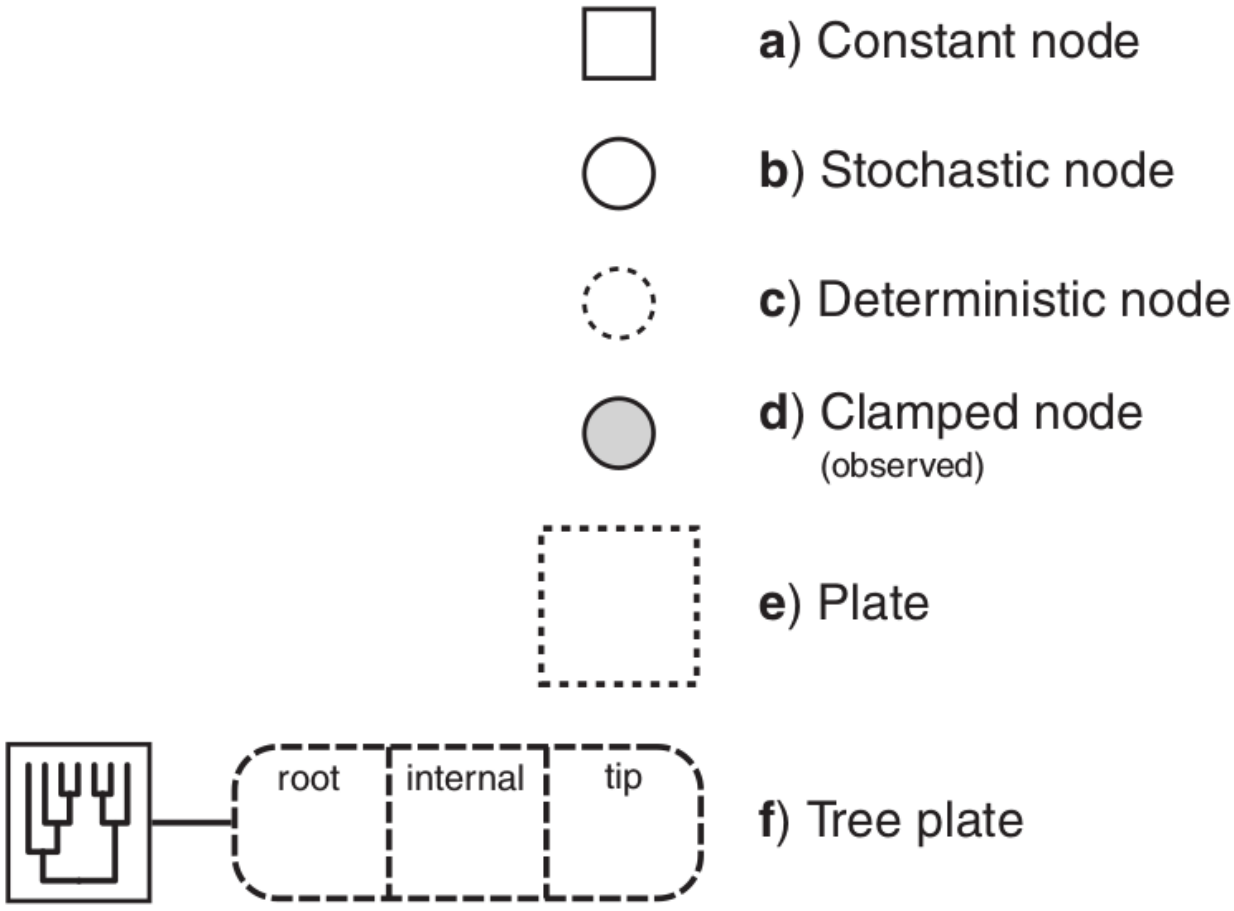


Image from Höehna et al. (2014)

phylogenetic graphical model

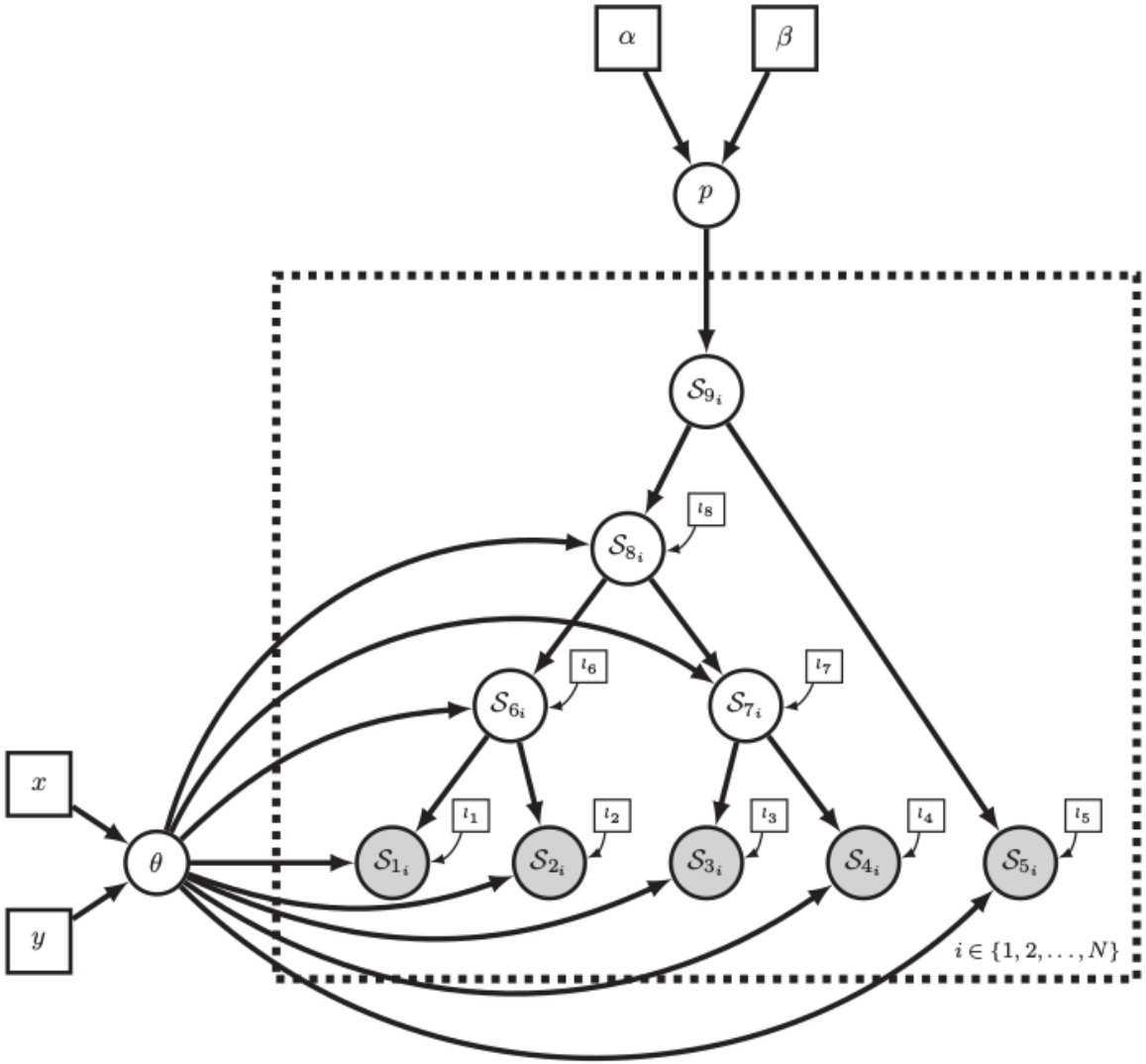


FIGURE 4. A phylogenetic graphical model of N independently evolving binary characters. When sampling N different binary characters for each extant species, we assume that these characters are independent and identically distributed. Thus the model for each character is the same as in Figure 3b. Yet, the state for each character $1,\ldots,N$ can be different. We use the *plate* notation to represent repetition over a vector of elements. In this figure, the dashed box and the iterator i indicate the replicated variables. Thus, the plate represents separate variables of binary character evolution for i in characters $1,2,3,\ldots,N$.

Image from Höehna et al. (2014)

phylogenetic graphical models as modules

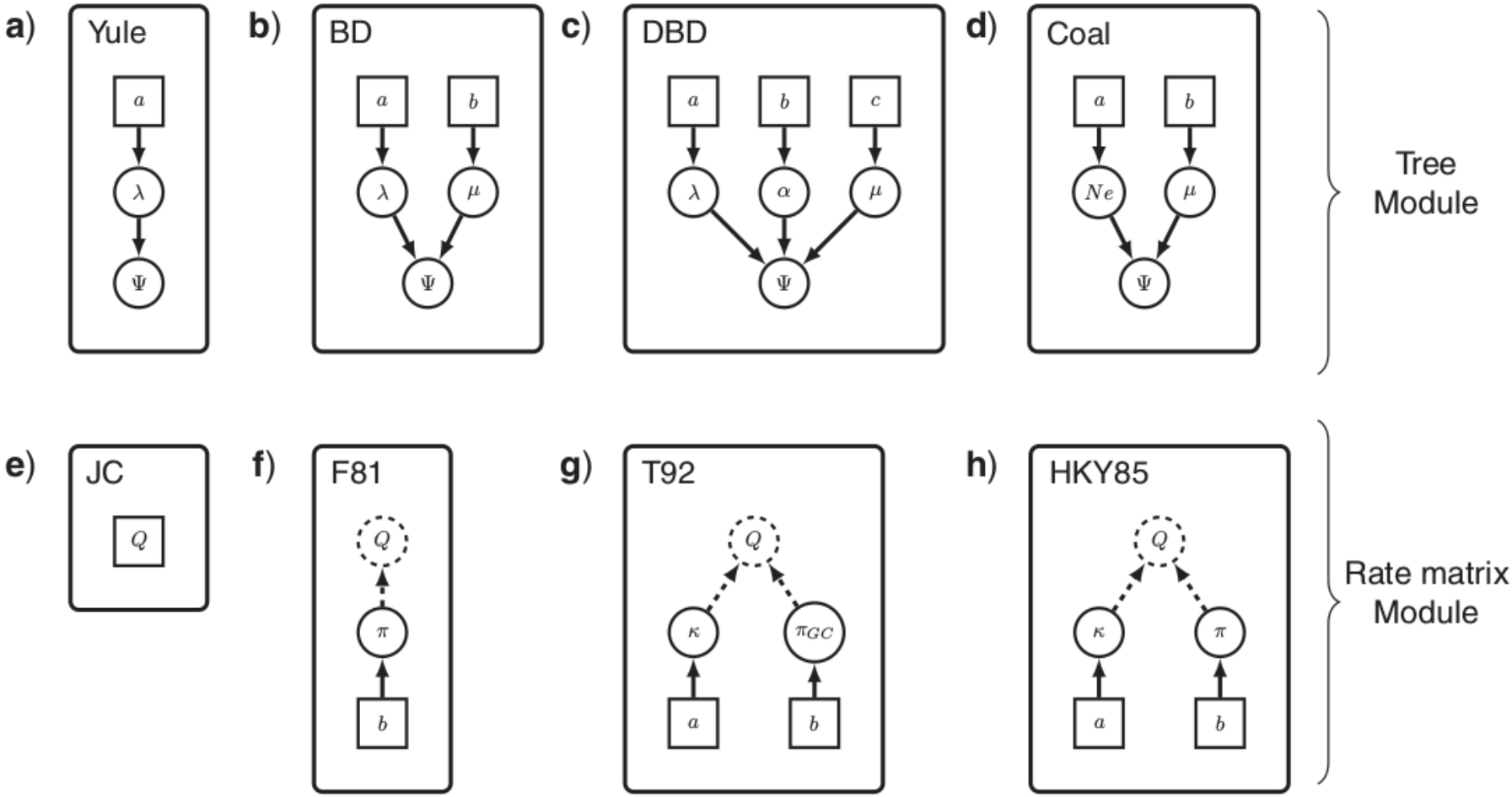


Image from Höehna et al. (2014)

assembling phylogenetic models like lego kits

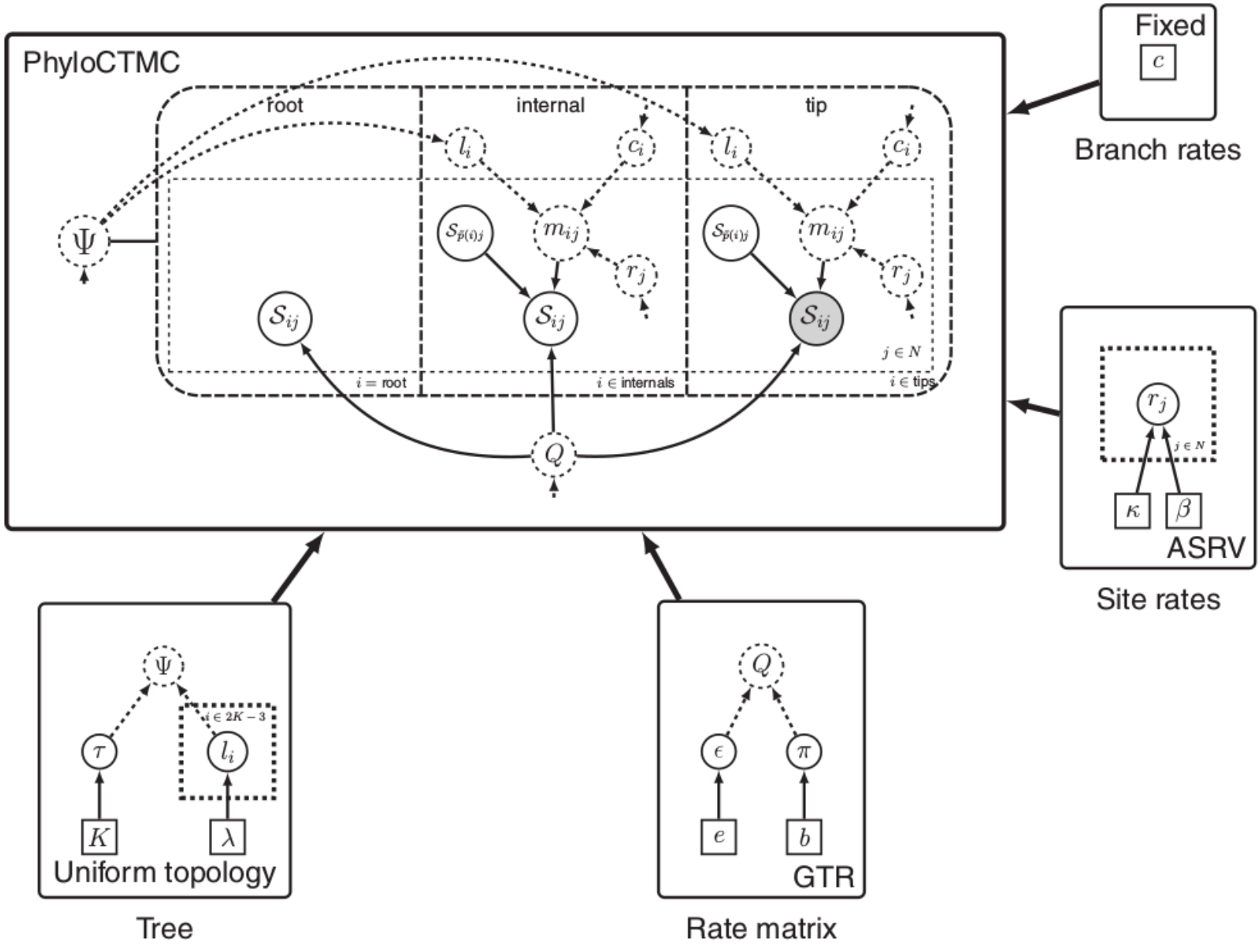


Image from Höehna et al. (2014)

Is the graphical model paradigm really helpful?

Disadvantages:

1. steep learning curve...
   - constant, stochastic, deterministic nodes
   - clamping
   - MCMC proposals

Advantages:

1. transparency: all modeling assumptions are specified
2. power and flexibility: build custom models that test your specific hypotheses
3. efficiency: customize inference algorithm to efficiently perform inference
4. applicability: the same concepts are widely used in many probabilistic programming languages like Stan, BUGS, Edward, PyMC3... and `Rev`!

The Rev probabilistic programming language:

Most Rev scripts have two important aspects woven together:

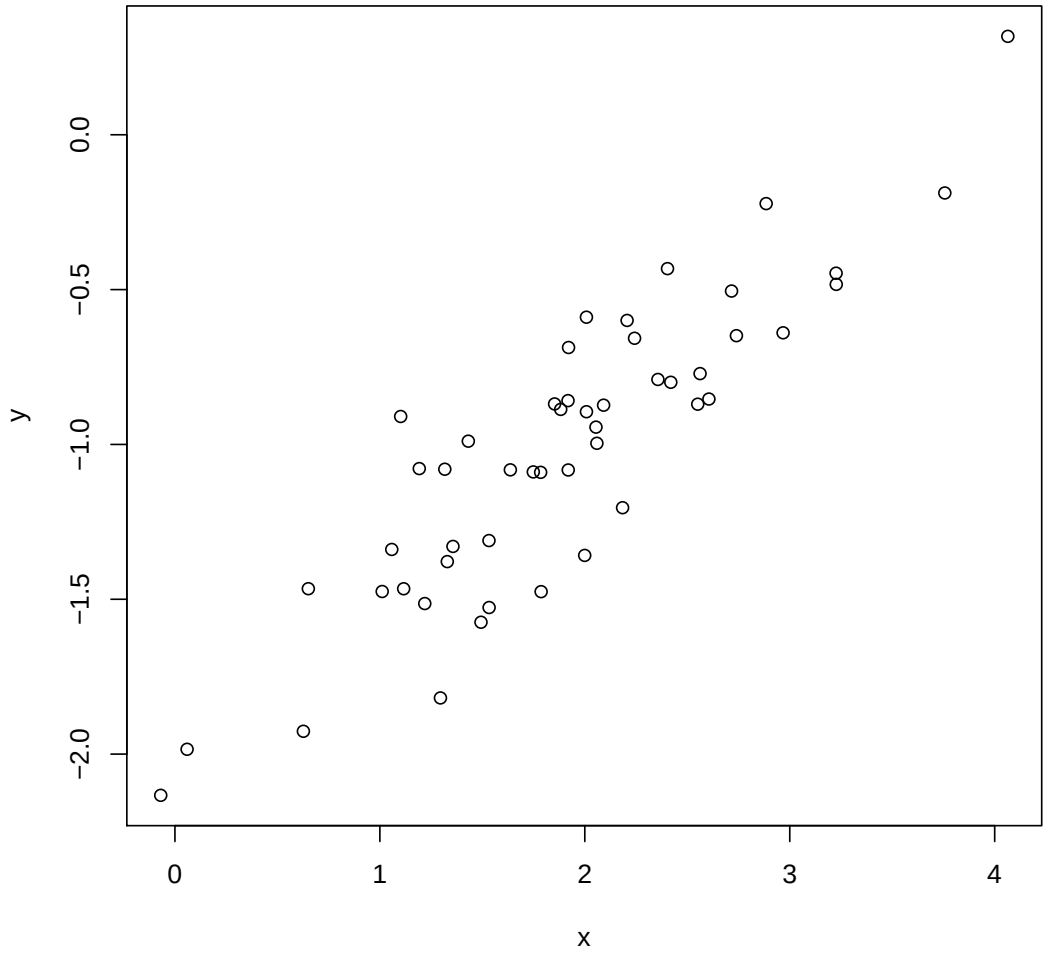
1. the graphical model specification
2. the inference algorithm specification (MCMC moves, etc.)

Since our goal is think abstractly in terms of graphical models we're going to learn these two aspects separately.

In Rev we can specify any type of probabilistic model, not just phylogenetic models:

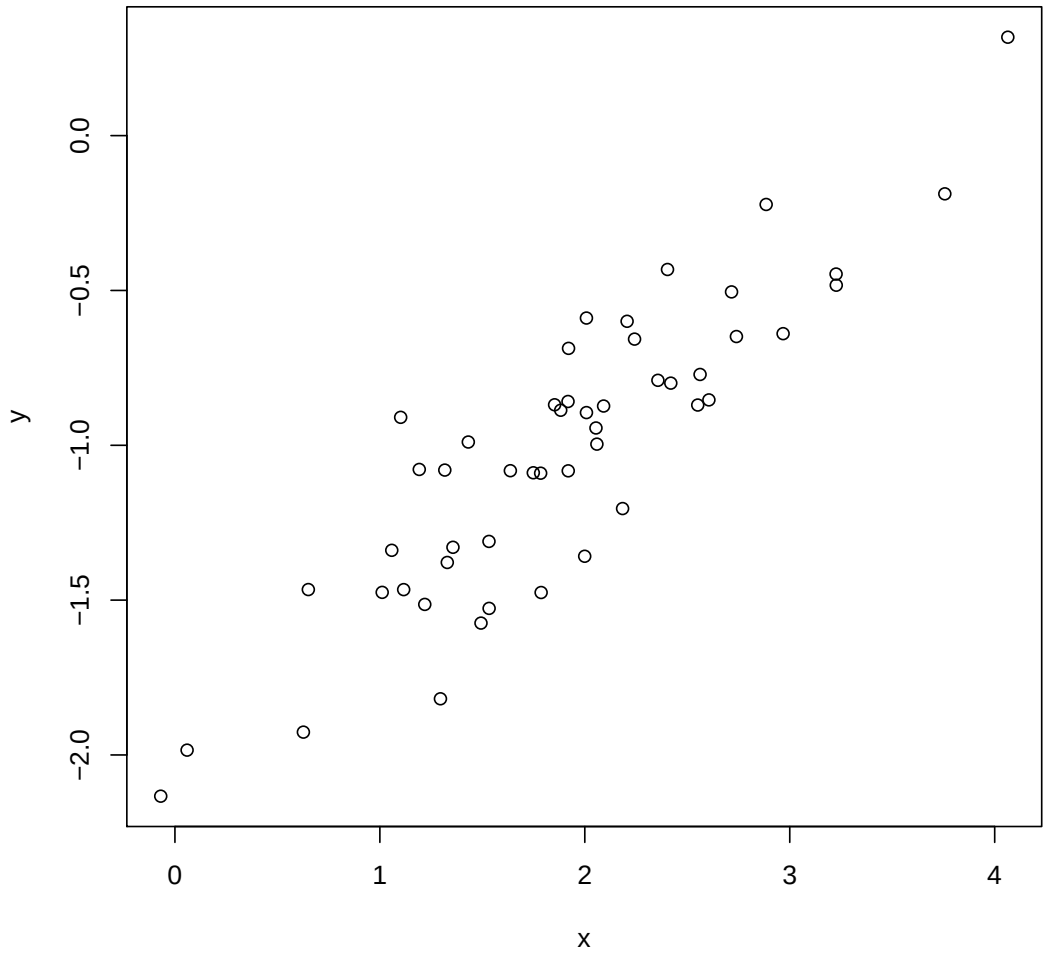
- classification models
- time series models
- neural networks
- etc...

linear regression as a graphical model



$$y = \beta x + \alpha + \epsilon$$

linear regression as a graphical model



$$\mu_y = \beta x + \alpha$$
$$y \sim \text{Normal}(\mu_y, \sigma_\epsilon)$$

Bayesian linear regression

$$\mu_y = \beta x + \alpha$$
$$y \sim \text{Normal}(\mu_y, \sigma_\epsilon)$$

We need some priors!

$$\beta \sim \text{Normal}(\mu = 0, \sigma^2 = 1)$$
$$\alpha \sim \text{Normal}(\mu = 0, \sigma^2 = 1)$$
$$\sigma_\epsilon \sim \text{Exponential}(\lambda = 1)$$

Now let's program the model in `Rev`...

First clone the workshop's repo and start `rb`:

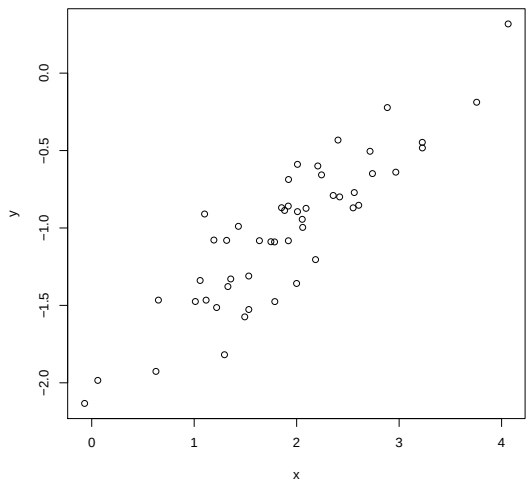
```
git clone https://github.com/wf8/
    RevBayes_UC_Berkeley_2018_Workshop.git
cd RevBayes_UC_Berkeley_2018_Workshop
rb
```

The full example scripts are in the `src/` directory, but instead of using `source()` let's walk through the script interactively...

Our observed data are constant nodes:

```
x_obs <- readDataDelimitedFile(file="data/x.csv", header=FALSE
    , delimiter=",")[1]
y_obs <- readDataDelimitedFile(file="data/y.csv", header=FALSE
    , delimiter=",")[1]
```

Take a look at `x_obs` and `y_obs`: they are simply vectors containing these points:



We'll specify relatively uninformative priors for our parameters. These are stochastic nodes:

```
beta ~ dnNormal(0, 1)
alpha ~ dnNormal(0, 1)
sigma ~ dnExponential(1)
```

Deterministic nodes for μ_y and stochastic nodes for y:

```
for (i in 1:x_obs.size()) {
    mu_y[i] := (beta * x_obs[i]) + alpha
    y[i] ~ dnNormal(mu_y[i], sigma)
}
```

We have now simulated values of y conditioned on x. So this is a discriminative model.

What happens if we set the value of β or α to something else? Try the `setValue()` and `redraw()` member methods.

To estimate parameter values, we need to clamp the observed values of y:

```
for (i in 1:x_obs.size()) {
    mu_y[i] := (beta * x_obs[i]) + alpha
    y[i] ~ dnNormal(mu_y[i], sigma)
    y[i].clamp(y_obs[i])
}

mymodel = model(beta)
```

So now we have specified the full linear regression model in a discriminative form: y conditioned on x.

Let's set up the MCMC...

We need at least one MCMC move for each parameter we want to estimate:

```
moves[1] = mvSlide(beta)
moves[2] = mvSlide(alpha)
moves[3] = mvSlide(sigma)
```

We also need monitors for the MCMC:

```
monitors[1] = mnScreen()
monitors[2] = mnModel("output/linear_regression.log")
```

Now run the MCMC:

```
mymcmc = mcmc(mymodel, moves, monitors)
mymcmc.run(10000)
```

Look at the results in Tracer. What estimates do you get for α, β, and σ?

The true values were $\alpha = -2$, $\beta = 0.5$, and $\sigma = 0.2$.

Exercise: modify this to be a fully generative model.

Exercise: experiment with different MCMC moves to make the analysis converge more efficiently.

Finally phylogenetics!

Let's use the distinction between a discriminative (or conditional) model and a generative model to explore the link between tree inference models and comparative methods...

One of the most basic comparative methods is ancestral state estimation.

Ancestral state estimation conditioned over a tree

Here we will estimate ancestral states for a binary character conditioned over a fixed tree.

This is not a generative model because it does not model the process that generated all our data (which is both the tip data and the tree).

Read in the tree and tip data:

```
tree_obs <- readTrees("data/phylo.tree")[1]
morph_data <- readCharacterData("data/phylo_morph.nex")
```

Ancestral state estimation conditioned over a tree

character state transition rate matrix

```
q_01 ~ dnExponential(10)
q_10 ~ dnExponential(10)
Q_morph := fnFreeK([q_01, q_10], rescaled=FALSE)
```

Ancestral state estimation conditioned over a tree

root frequencies:

```
pi_morph ~ dnDirichlet([1,1])
```

the phylogenetic continuous-time Markov process:

```
ctmc_morph ~ dnPhyloCTMC(tree_obs, Q=Q_morph, rootFrequencies=
    pi_morph, nSites=1, type="Standard")
```

Before clamping the observed data, look at the simulated values for `ctmc_morph`.

```
ctmc_morph.clamp(morph_data)

mymodel = model(ctmc_morph)
```

Ancestral state estimation conditioned over a tree

Now that we have specified our phylogenetic model, just like in the linear regression example we need moves and monitors to run the MCMC.

```
moves[1] = mvScale(q_01)
moves[2] = mvScale(q_10)
moves[3] = mvSimplexElementScale(pi_morph)

monitors[1] = mnScreen(printgen=10)
monitors[2] = mnModel(printgen=10, filename="output/
    conditional.log")
monitors[3] = mnJointConditionalAncestralState(printgen=10,
                                               filename="output/
                                                   conditional_anc
                                                   .log",
                                               tree=tree_obs,
                                               ctmc=ctmc_morph,
                                               type="Standard")
```

Ancestral state estimation conditioned over a tree

Run the MCMC:

```
mymcmc = mcmc(mymodel, moves, monitors)
mymcmc.run(10000)
```

and summarize the ancestral states:

```
anc_trace = readAncestralStateTrace("output/conditional_anc.
    log")
ancestralStateTree(tree_obs,
                   ancestral_state_trace_vector=anc_trace,
                   file="output/map_anc_conditional.tree")
```

Ancestral state estimation conditioned over a tree

Take a look at the results in FigTree and note the high posterior probabilities.

This reconstruction does not take phylogenetic uncertainty into account, so it may be an overly confident estimate.

A better approach would be to jointly model the ancestral states and tree inference.

How would we modify this model so that it is a fully generative model?

Ancestral states estimated jointly with the tree!

We need to add a model of how trees are generated – the diversification process – perhaps a birth-death process or a coalescent tree prior.

The binary morphological character probably won't help inform the topology and divergence times, so let's use molecular data too. A molecular data alignment is provided `data/phylo_mol.nex`.

We will need a joint model of:

1. the diversification process
2. morphological character evolution
3. molecular character evolution

Ancestral states estimated jointly with the tree!

Exercise: code up an analysis that jointly estimates ancestral states, the tree topology, and divergence times.

Suggestions:

1. the diversification process: use `dnBirthDeath()`
2. morphological character evolution: use the model already shown
3. molecular character evolution: use `fnGTR()`

One solution is in `src/ancestral_states_joint.Rev`, work through this script if you get stuck.

Ask questions!

Check out the tutorials on `http://revbayes.com`