Introduction to graphical models using RevBayes

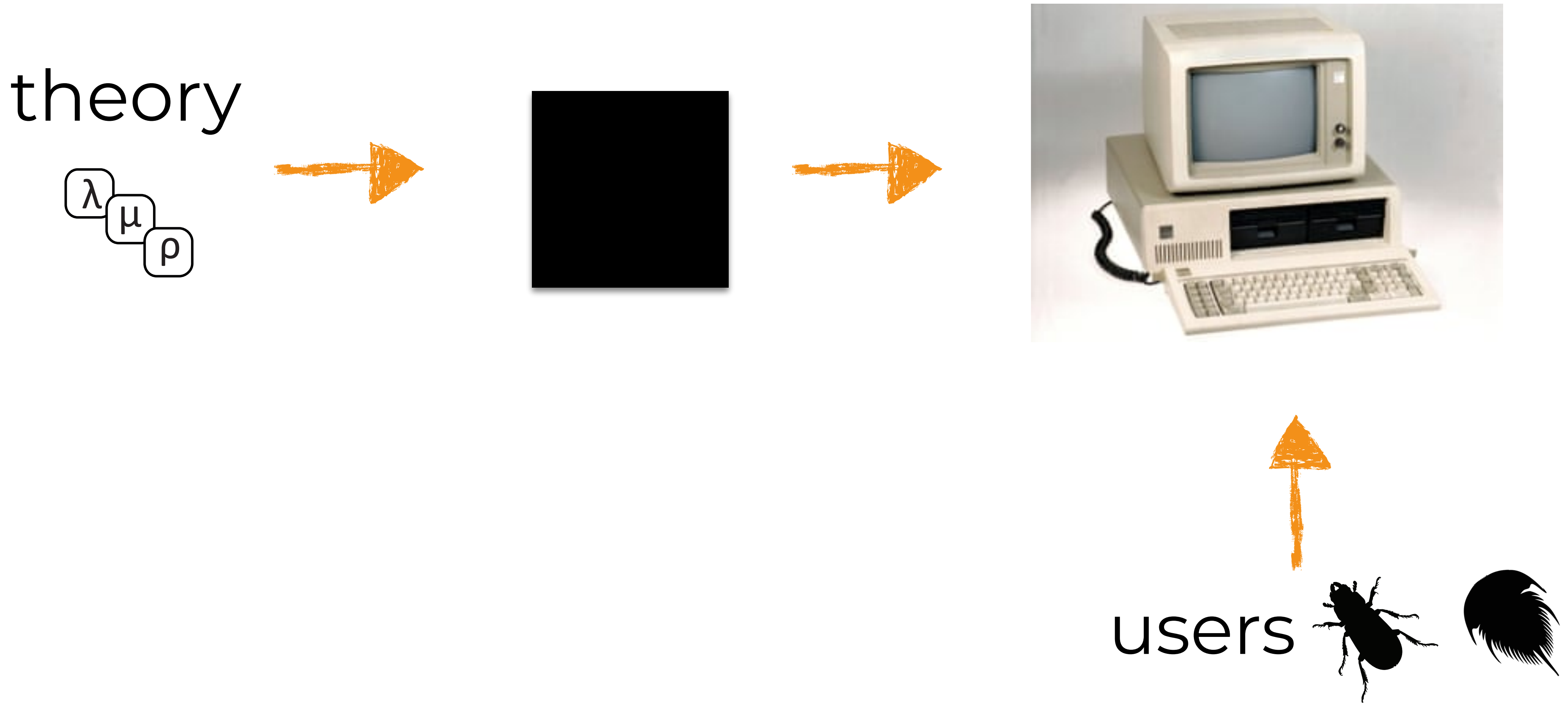
Rachel Warnock, Laura Mulvey

FAU

05.09. 2022

Friedrich-Alexander-Universität Erlangen-Nürnberg

Phylogenetic inference — the old way



Phylogenetic inference — a better way?

theory

λ μ ρ

users

Aims for RevBayes

Flexible model specification

- Availability of (common) models
- Extendability

Computational efficiency

- Fast likelihood calculators
- Efficient (MCMC) algorithms

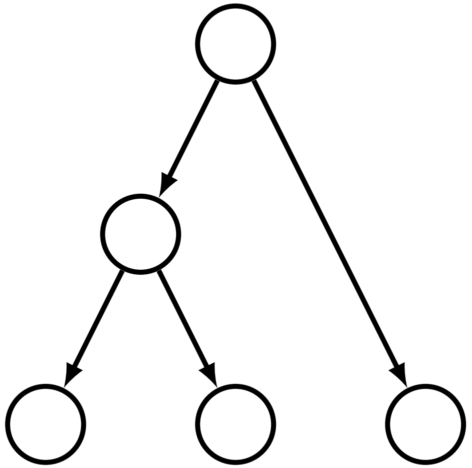
Easy to learn

- Well structured model specification
- Explicit models
- Documentation, examples and tutorials

RevBayes uses a graphical model framework

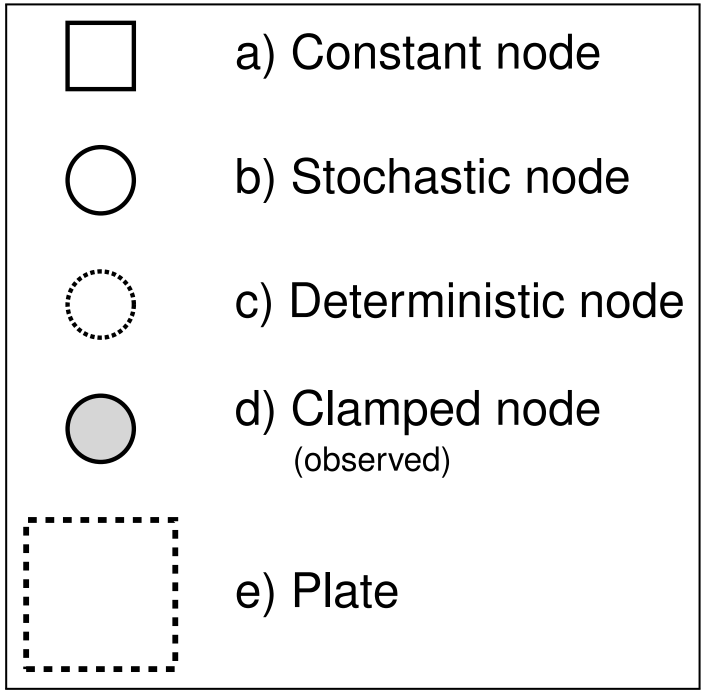
Graphical models provide tools for visually and computationally representing complex, parameter-rich probabilistic models.

We can depict the conditional dependence structure of various parameters and other random variables.



Höhna et al. 2014. Sys Bio

Graphical models — types of variables (nodes)



a) fixed-value variables

b) random variables that depend on other variables

c) variables determined by a specific function applied to another variable (transformations)

d) observed stochastic variables (data)

e) replication over a set of variables

Specifying graphical models using the Rev syntax

Table 1: `Rev` assignment operators, clamp function, and plate/loop syntax.

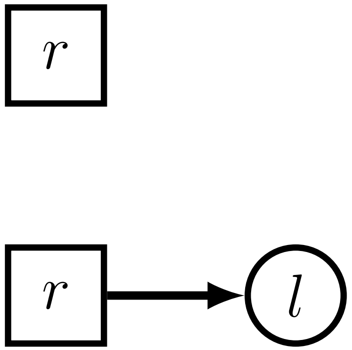
Operator	Variable
`<-`	constant variable
`~`	stochastic variable
`:=`	deterministic variable
`node.clamp(data)`	clamped variable
`=`	inference (*i.e.*, non-model) variable
`for(i in 1:N){...}`	plate

a) $\boxed{r}$

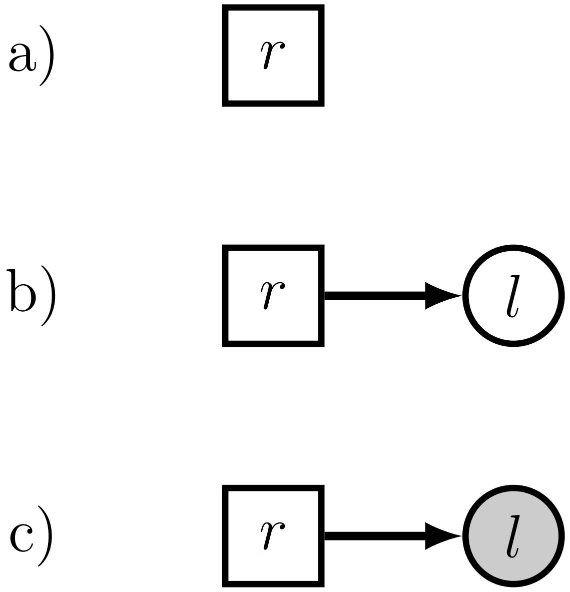
```
# constant node
r <- 10
```

a)

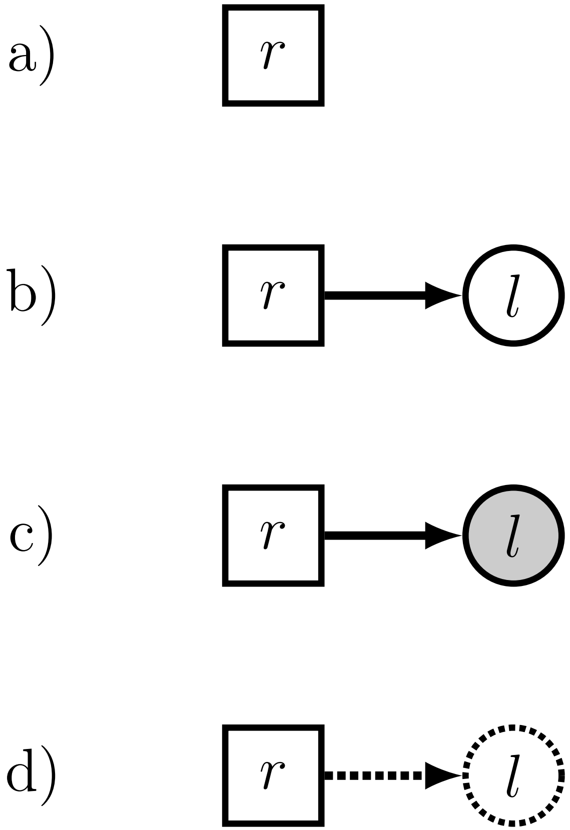
b)



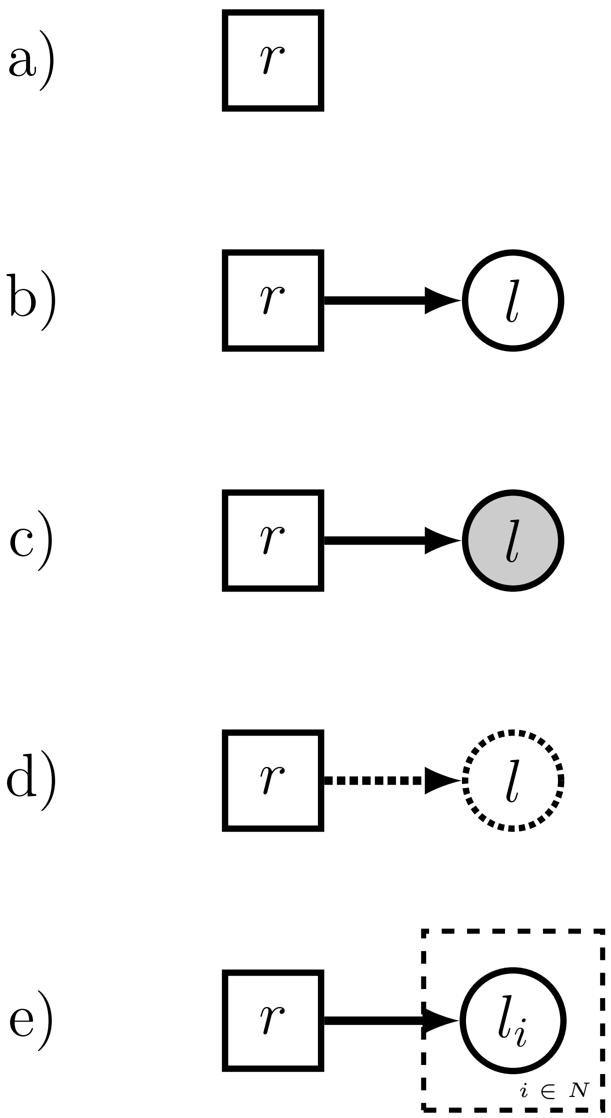

```
# constant node
r <- 10


# stochastic node
l ~ dnExp(r)
```



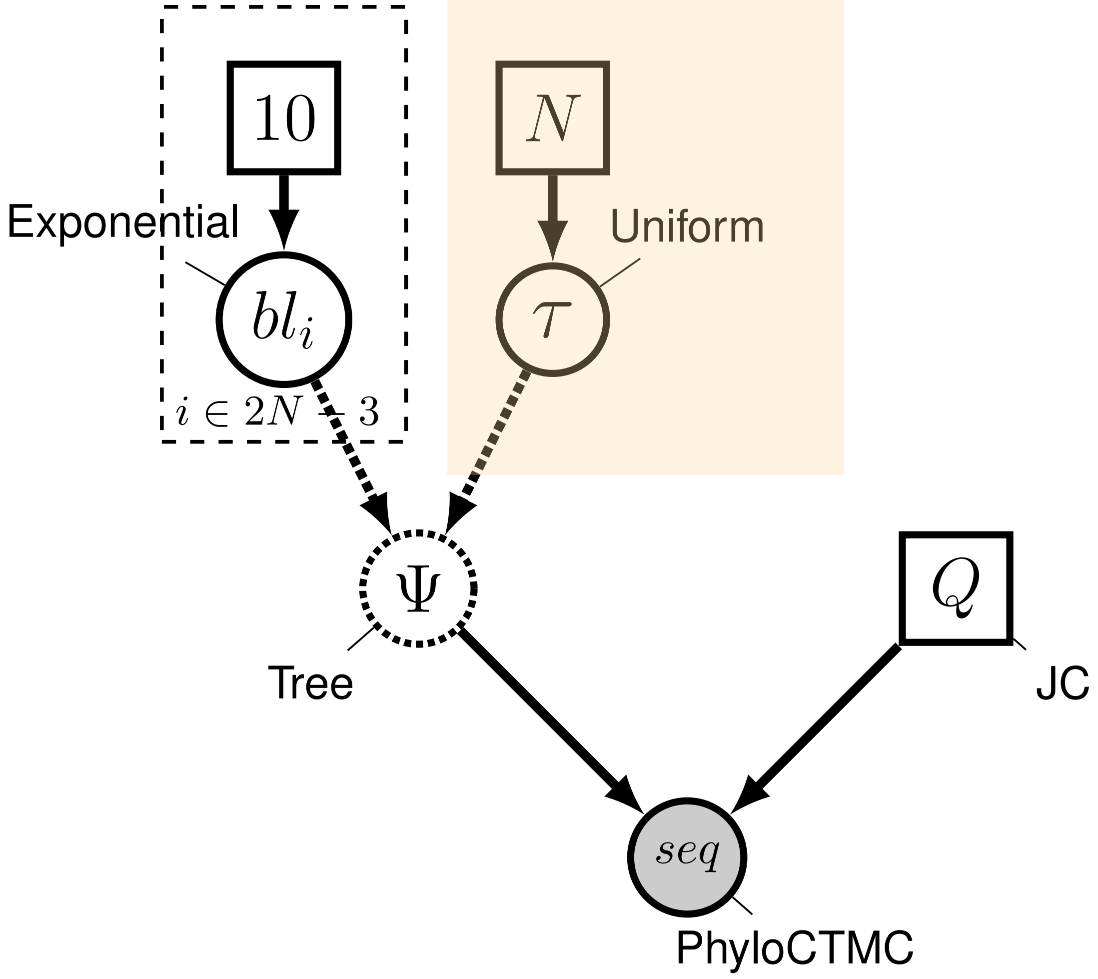

```
# constant node
r <- 10

# stochastic node
l ~ dnExp(r)

# stochastic node (observed)
l.clamp(0.1)
```



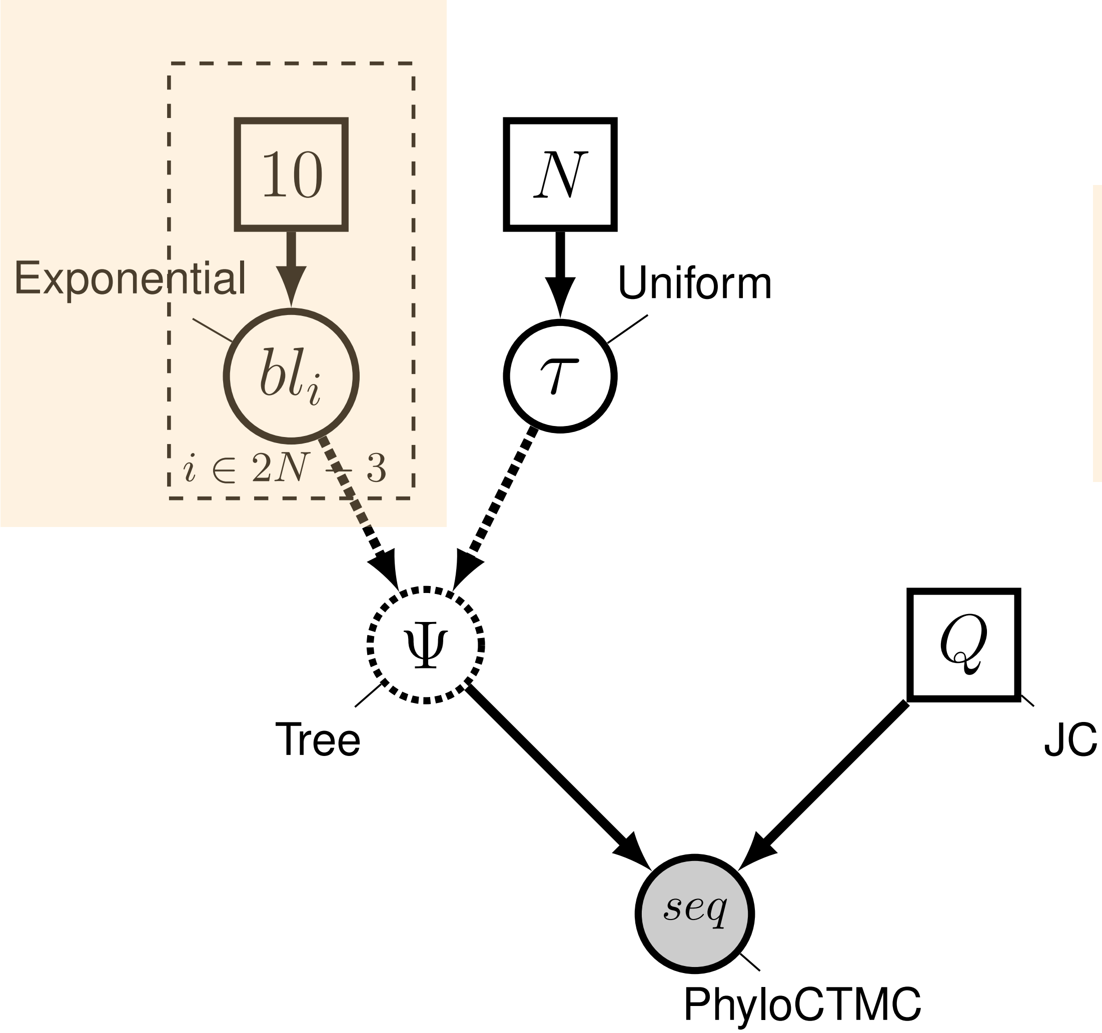

```
# constant node
r <- 10

# stochastic node
l ~ dnExp(r)

# stochastic node (observed)
l.clamp(0.1)

# deterministic node
l := exp(r)
```



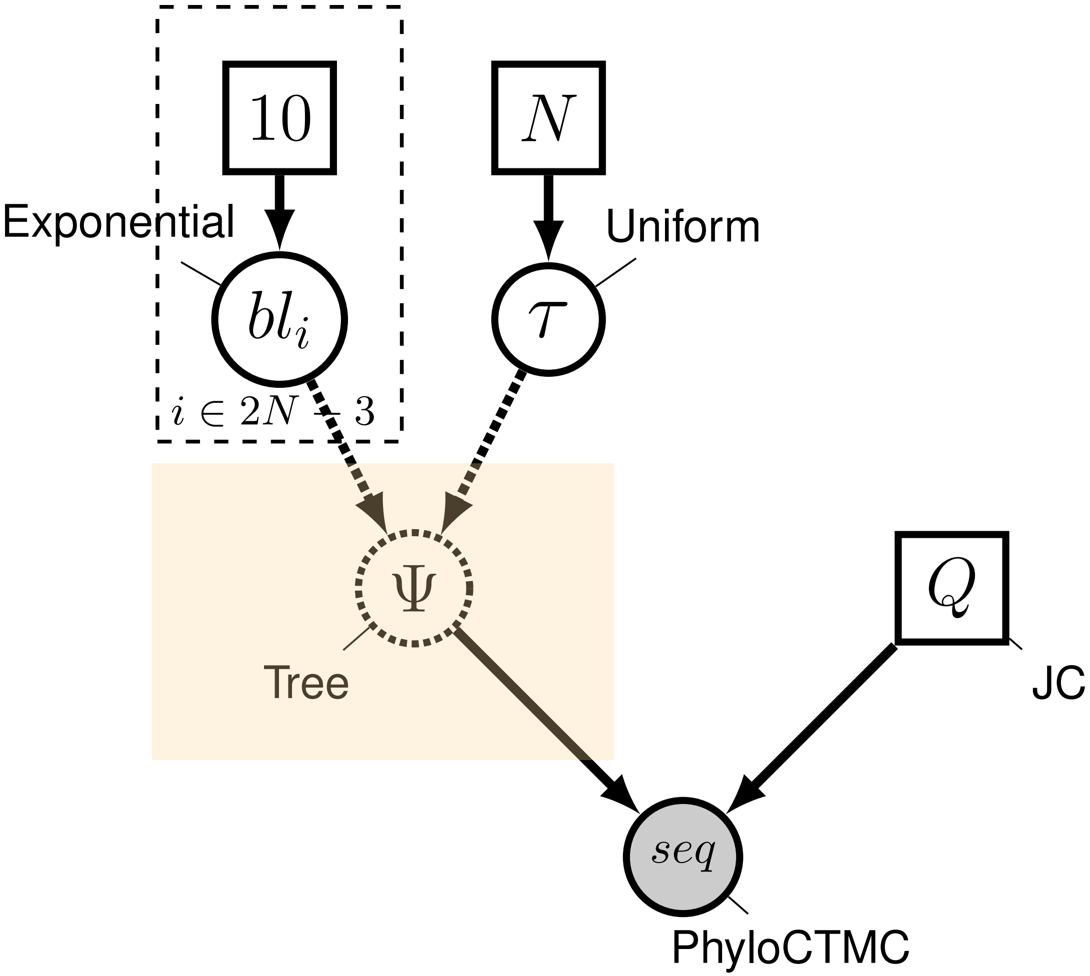

```
# constant node
r <- 10

# stochastic node
l ~ dnExp(r)

# stochastic node (observed)
l.clamp(0.1)

# deterministic node
l := exp(r)

# stochastic nodes (iid)
for (i in 1:N) {
    l[i] ~ dnExp(r)
}
```



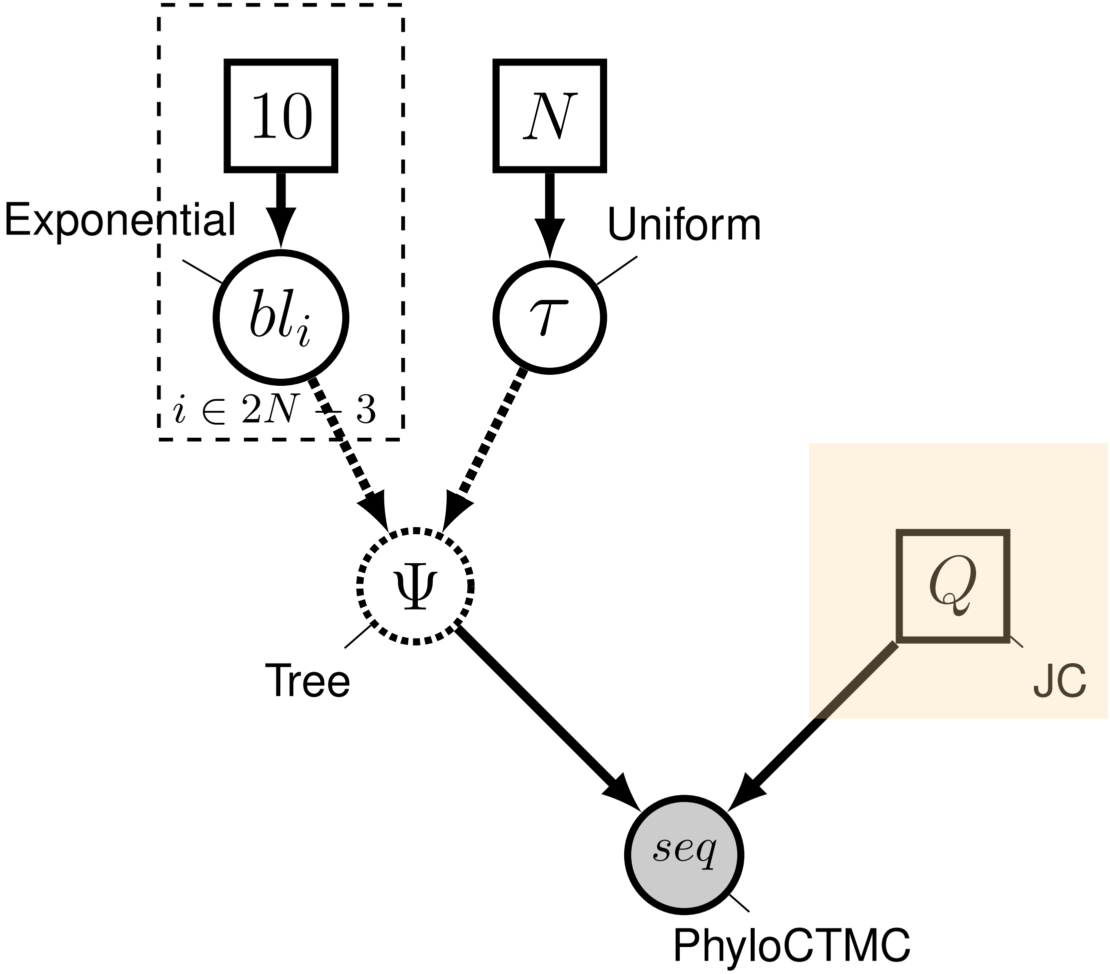

```
# prior on the tree topology
topology ~ dnUniformTopology(taxa)

# prior on the branch lengths
for (i in 1:num_branches) {
   br_lens[i] ~ dnExponential(10)
   moves.append( mvScale(br_lens[i]) )
}

tree := treeAssembly(topology, br_lens)

TL := sum(br_lens)

# 4 state rate maxtrix (JC model)
Q <- fnJC(4)

# attach the model to your sequence data
seq ~ dnPhyloCTMC(tree = tree, Q = Q, type = "DNA")
seq.clamp(data)
```



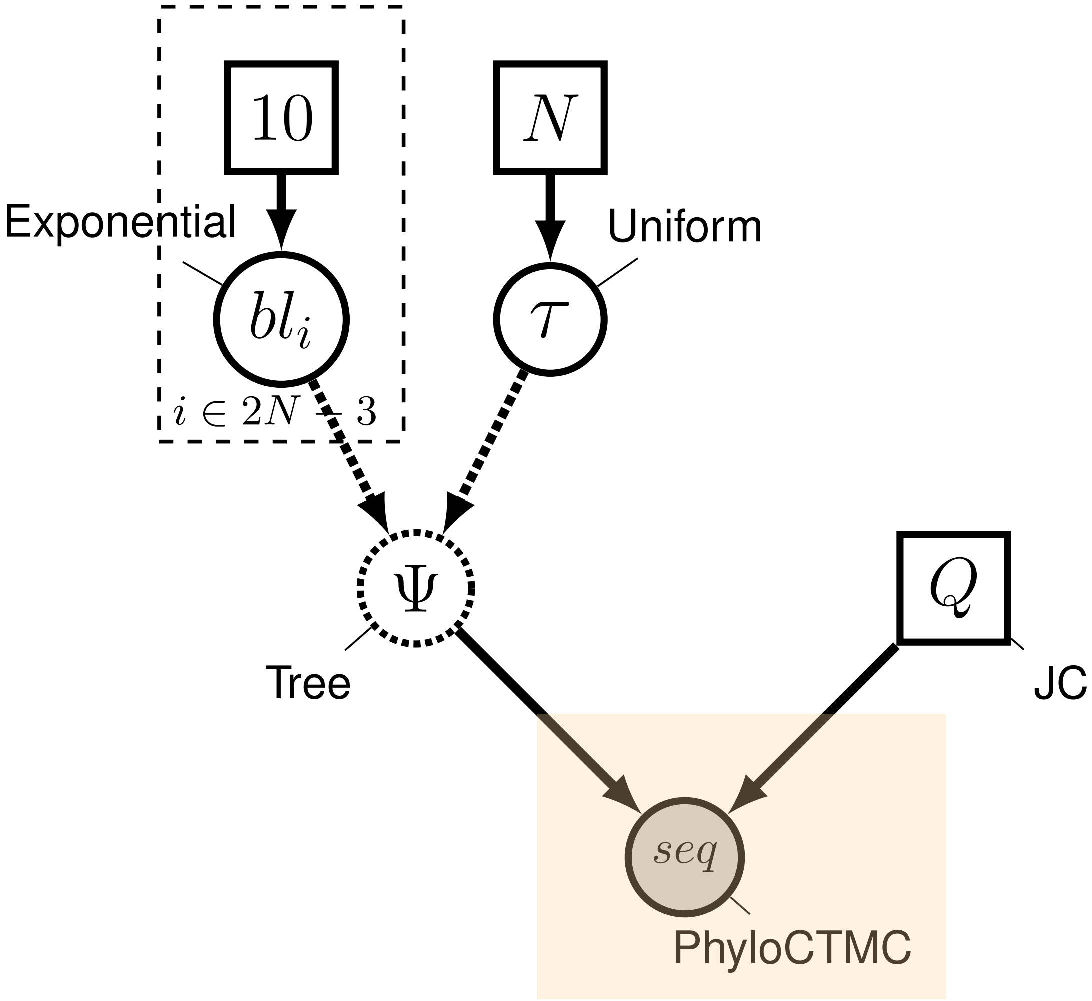

```
# prior on the tree topology
topology ~ dnUniformTopology(taxa)

# prior on the branch lengths
for (i in 1:num_branches) {
   br_lens[i] ~ dnExponential(10)
   moves.append( mvScale(br_lens[i]) )
}

tree := treeAssembly(topology, br_lens)

TL := sum(br_lens)

# 4 state rate maxtrix (JC model)
Q <- fnJC(4)

# attach the model to your sequence data
seq ~ dnPhyloCTMC(tree = tree, Q = Q, type = "DNA")
seq.clamp(data)
```




```
# prior on the tree topology
topology ~ dnUniformTopology(taxa)

# prior on the branch lengths
for (i in 1:num_branches) {
   br_lens[i] ~ dnExponential(10)
   moves.append( mvScale(br_lens[i]) )
}

tree := treeAssembly(topology, br_lens)

TL := sum(br_lens)

# 4 state rate maxtrix (JC model)
Q <- fnJC(4)

# attach the model to your sequence data
seq ~ dnPhyloCTMC(tree = tree, Q = Q, type = "DNA")
seq.clamp(data)
```




```
# prior on the tree topology
topology ~ dnUniformTopology(taxa)

# prior on the branch lengths
for (i in 1:num_branches) {
   br_lens[i] ~ dnExponential(10)
   moves.append( mvScale(br_lens[i]) )
}

tree := treeAssembly(topology, br_lens)

TL := sum(br_lens)

# 4 state rate maxtrix (JC model)
Q <- fnJC(4)

# attach the model to your sequence data
seq ~ dnPhyloCTMC(tree = tree, Q = Q, type = "DNA")
seq.clamp(data)
```




```
# prior on the tree topology
topology ~ dnUniformTopology(taxa)

# prior on the branch lengths
for (i in 1:num_branches) {
   br_lens[i] ~ dnExponential(10)
   moves.append( mvScale(br_lens[i]) )
}

tree := treeAssembly(topology, br_lens)

TL := sum(br_lens)

# 4 state rate maxtrix (JC model)
Q <- fnJC(4)

# attach the model to your sequence data
seq ~ dnPhyloCTMC(tree = tree, Q = Q, type = "DNA")
seq.clamp(data)
```

The Weeping Woman by Picasso

Bayesian inference is like a cubist painting, the more you stare at it the more it begins to make sense.

—Something Kat spotted in a textbook