

# WinBUGS/OpenBUGS

- A tool that will do the MCMC sampling for you.
- **What you need to do?**
  - **Write *logical definition*** of your model:
    - Prior and likelihood.
    - Model can be hierarchical with several layers.
  - Define what your data are. (fixed values).
  - The model should constitute a proper posterior distribution. (Or prior if no data).
  - Compile and run, monitor results, check convergence, analyze results.

# WinBUGS/OpenBUGS

- First example: Binomial data
  - Recall the conjugate solution.
  - $p(\theta | X) = p(X | N, \theta) p(\theta) / c$
  - To compute the posterior, we define  $p(X | N, \theta)$  and  $p(\theta)$ . And we set a value for  $X$ .
  - We do not need to define or solve  $c$ !

# WinBUGS/OpenBUGS

- **BUGS –language:**

```
model{
```

```
X ~ dbin(theta,N)
```

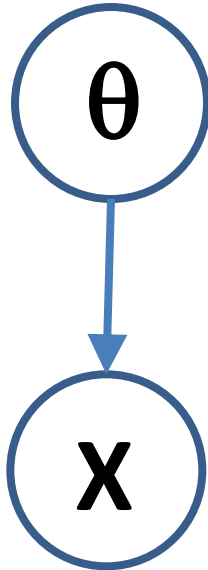
```
theta ~ dunif(0,1) # or dbeta(a,b)
```

```
}
```

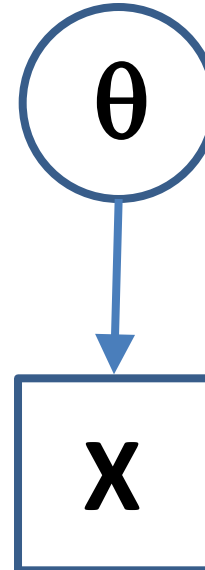
```
list(X=3,N=20)
```

# Directed Acyclic Graphs: DAG

- **Graphical representation: DAG**
  - Describes conditional distributions



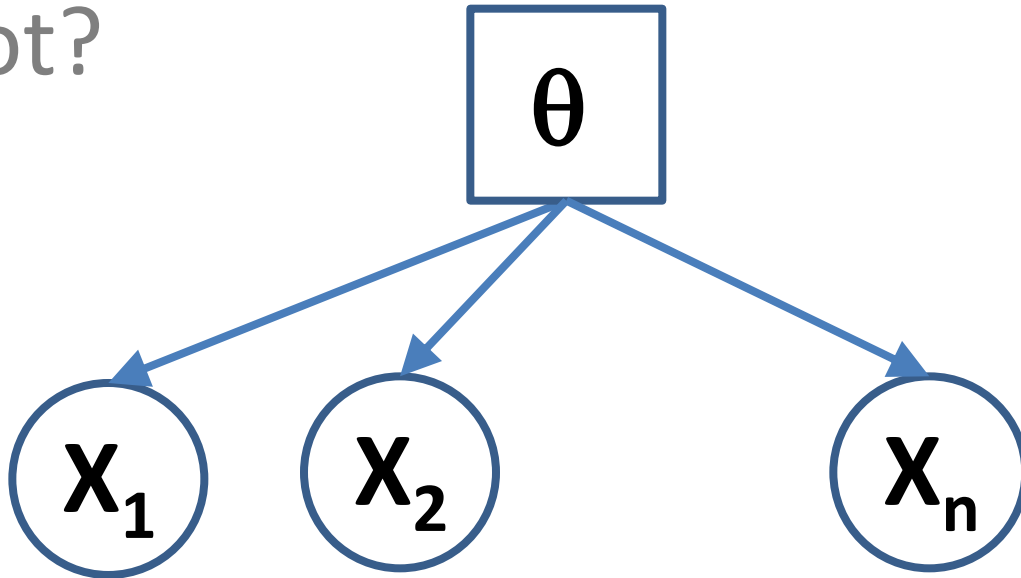
When  $X$  is unknown



When  $X$  is observed (fixed) as data

# DAG

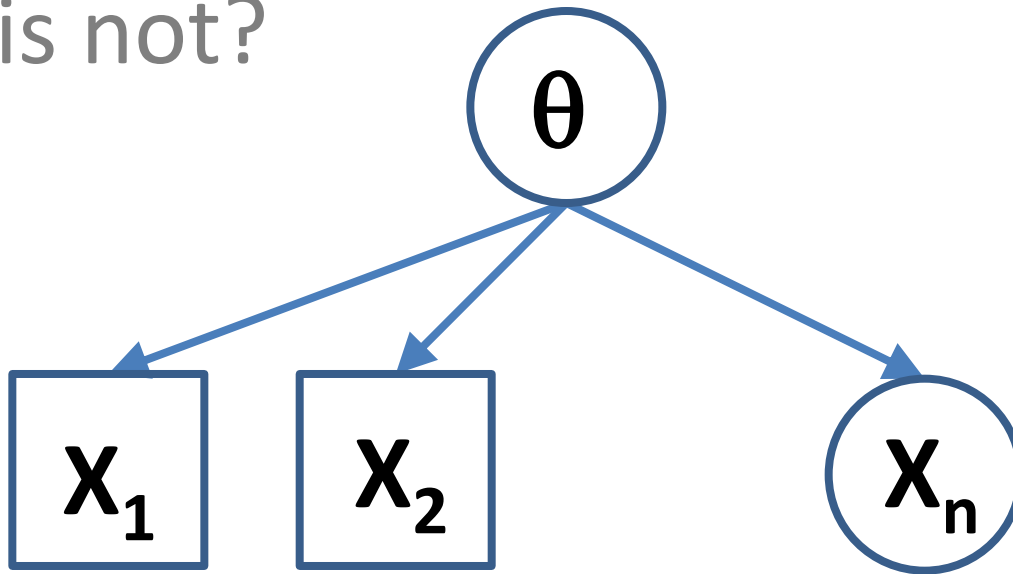
- What happens if  $\theta$  is fixed,  $X$  are not?



- $X$  will be independent of each other, given  $\theta$ . E.g. simulate  $X$  with given parameters.

# DAG

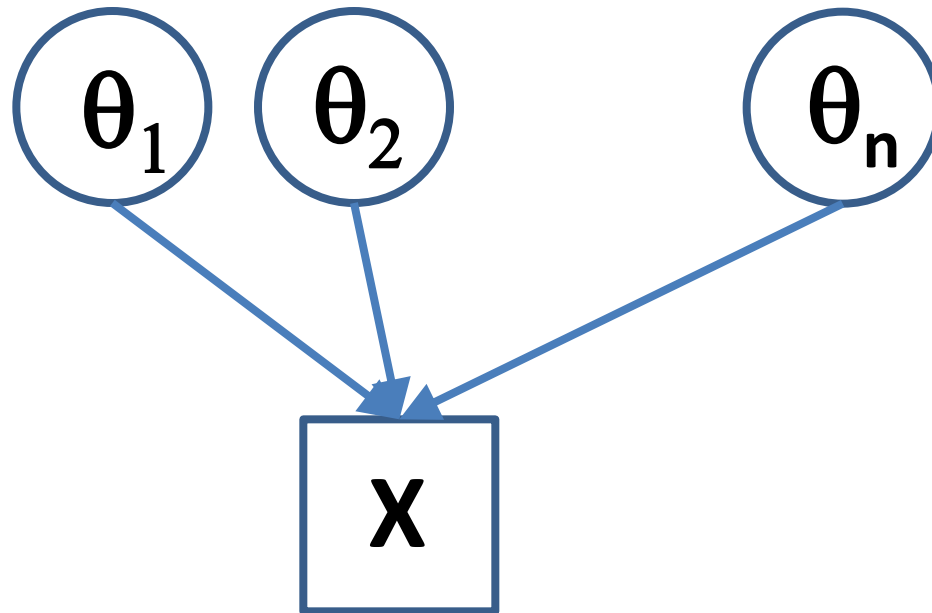
- What happens if some  $X$  are fixed,  $\theta$  is not?



- Unknown  $X$  will be dependent of known  $X$ . (we can learn from them).

# DAG

- What happens if  $X$  is fixed,  $\theta$  are not?



- Unknown  $\theta$  will be dependent on each other. E.g.  $X \sim N(\theta_1 + \theta_2, 1)$

# BUGS language

- **Declarative language: don't try to think procedural programming.**
- A logical definition can be expressed in several equivalent ways:

`X ~ dbin(theta,N)`

`theta ~ dunif(0,1); X <- 3; N <- 20`

**is same as** (assuming x is given as data)

`theta ~ dbeta(a,b); a <- X+1 ; b <- N-X+1; X <- 3; N <- 20`

(if x was not given as data, the latter would not be defined, and the former would produce predictive distribution for X & the prior for theta)

- Check syntax, load data, compile, load or generate inits, **run**.
- 'Save state' if you need to continue later from the same state → give these as inits.



# BUGS language

- **Always think DAGs**
- Can build DAGs with many levels, complex hierarchical modeling!
- A 'node' in a DAG can be either stochastic or deterministic
  - Stochastic:  $X \sim \text{dbin}(\text{theta}, N)$
  - Deterministic:  $Z \leftarrow \text{theta} * N$
- Does not matter in which order you write the declarations.

# Demo session with OpenBUGS

The screenshot displays the OpenBUGS software interface. The main window, titled 'untitled2', contains the following model specification:

```
model{  
X ~ dbin(theta,N)  
theta ~ dunif(0,1)  
}  
list(X=3,N=20)
```

Three tool windows are open:

- Specification Tool:** Contains buttons for 'check model', 'load data', 'compile', 'load inits', and 'gen inits'. It also has input fields for 'num of chains' (set to 1) and 'for chain' (set to 1).
- Update Tool:** Contains input fields for 'updates' (1000), 'refresh' (100), 'update', 'thin' (1), and 'iteration' (1000). It also has checkboxes for 'adapting' and 'over relax'.
- Sample Monitor Tool:** Shows the selected node 'theta'. It includes input fields for 'chains' (1 to 1), 'beg' (1), 'end' (10000000), and 'thin' (1). A 'percentiles' list is visible with values: 2.5, 5, 10, 25, median, 75, 90, 95, 97.5. It also features buttons for 'clear', 'set', 'stats', 'density', 'coda', 'diagnostics', 'trace', 'jump', 'bgr diag', 'history', 'accept', 'quantiles', and 'auto cor'.

At the bottom left of the screen, a status bar indicates '1000 updates took 0 s'. The Windows taskbar at the bottom shows various application icons and the system clock displaying '16:36 16.9.2012'.