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.
  - $p(\theta|X) = p(X \mid N,\theta)\, p(\theta)\, /c$
  - To compute the posterior, we define $p(X \mid N,\theta)$ and $p(\theta)$. And we set a value for X.
  - We do not need to define or solve c!

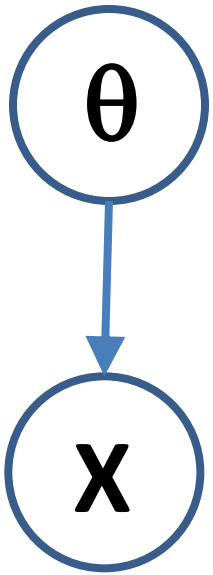
WinBUGS/OpenBUGS

- **BUGS –language:**

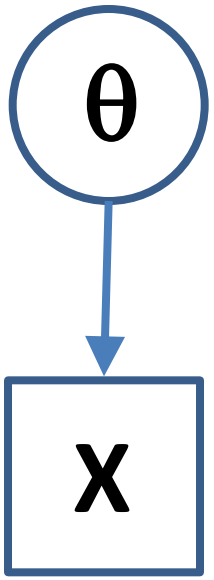
```
model{
X ~ dbin(theta,N)
theta ~ dunif(0,1)  # or  dbeta(a,b)
}
list(X=3,N=20)
```

Directed Acyclic Graphs: DAG

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



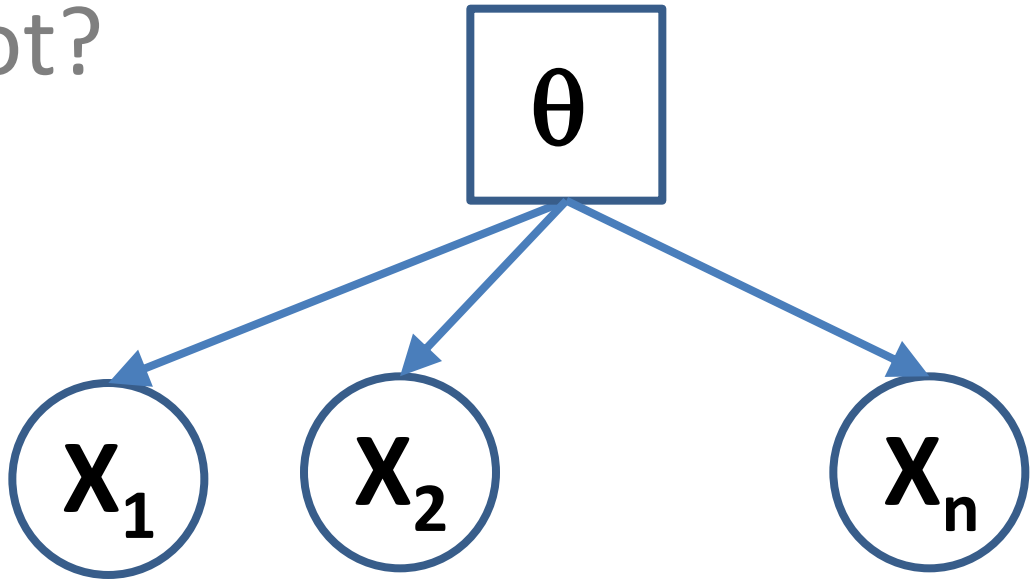
When X is unknown



When X is observed (fixed) as data

DAG

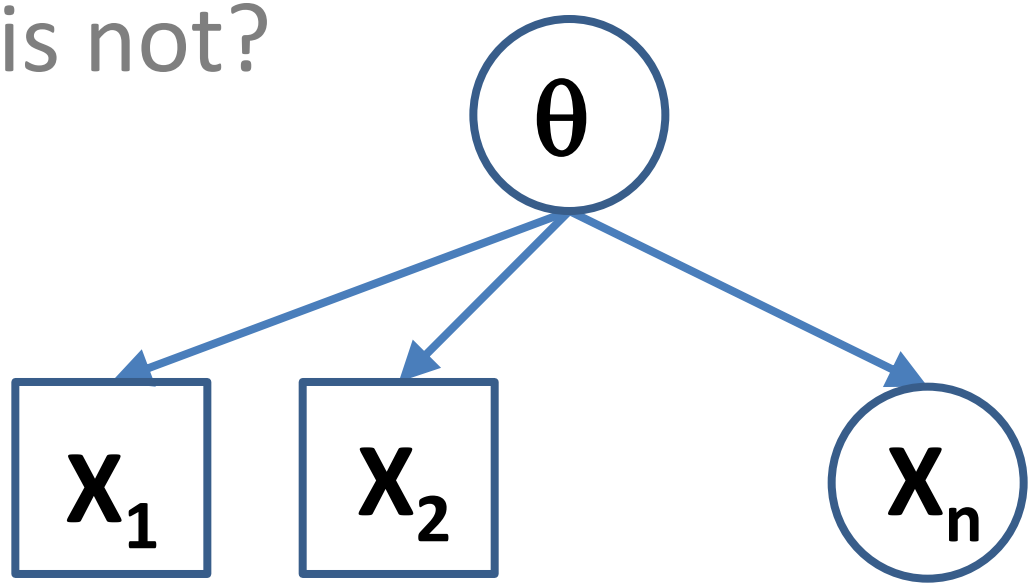
- What happens if θ is fixed, X are not?



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

DAG

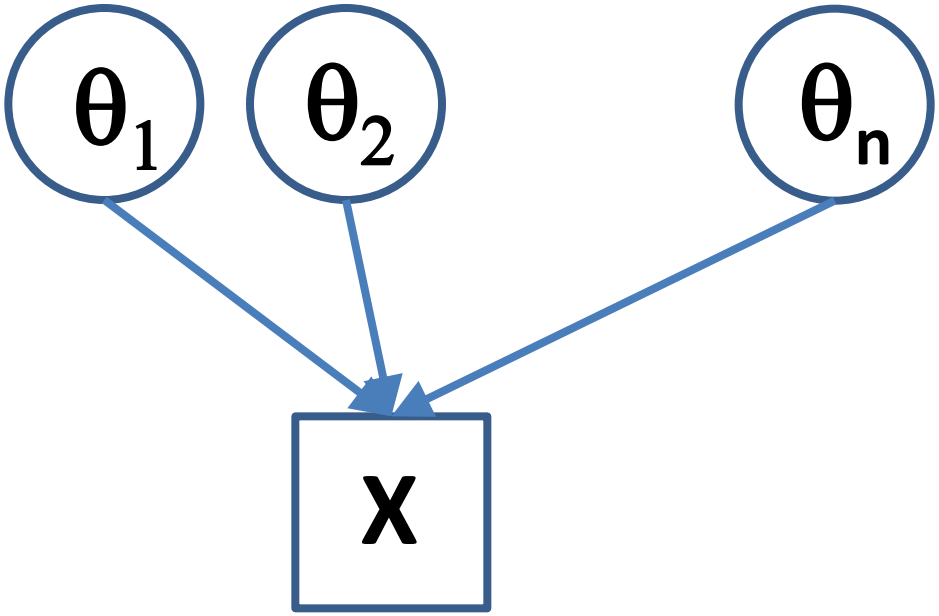
- What happens if some X are fixed, θ is not?



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

DAG

- What happens if X is fixed, θ are not?



- Unknown θ 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 ~ dbin(theta,N)
  - Deterministic: Z <- theta*N
- Does not matter in which order you write the declarations.

Demo session with OpenBUGS