

Some Modelling Issues



Problems

- Not always easy to get models that are easy to fit to data
- Some models are just awkward
- Others can be tweaked to improve the fitting
 - several techniques for this



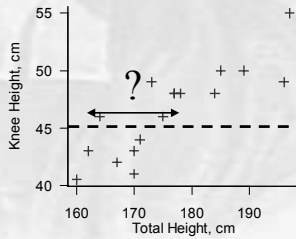
Missing Data

- Not all data sets are complete
- Missing data is a big problem
 - e.g. non-response in surveys
- Statistically, missing data can be dealt with in a Bayesian framework
 - treat missing data as another parameter
- Because of this, WinBugs can treat missing data easily

Knees (again)

- Knee and total height
- Suppose I know my knee height but not my total height

– 45cm



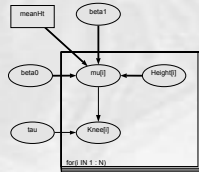
The Code

- Missing data is coded as NA
- Here we add it to the data:

```
list(N= 17, meanHt=176.1765, Knee=c(45, 41, 42, 40.5, 50, 48, 48, 48,
49, 48, 43, 46, 43, 46, 49, 44, 55, 50), Height=c(NA, 170, 167, 160,
189, 178, 177, 177, 196, 184, 162, 175, 170, 164, 173, 171, 197, 185))
```

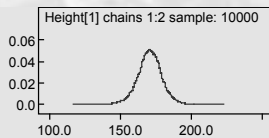
- Only difference in the DAG is that Height is now stochastic

– the missing observation has to be estimated



Results

- The posterior for Height



node	mean	sd	MC error	2.5%	median	97.5%	start	sample
Height[1]	170.7	8.843	0.09044	152.5	170.9	187.9	1001	10000

The Use

- This example was trivial
- In other cases, filling in missing data is useful
- e.g. 17 covariates, with hole in the data
 - don't need to throw away all the data with missing data
- May also need to consider the mechanisms by which data becomes missing
 - e.g. survival analysis
 - some patients outlive a clinical trial

Unbalanced Data

- Suppose we have this sort of data:
 - Group 1: **13.2**
 - Group 2: **12.3 14.1**
 - Group 3: **11.0 9.7 10.3 9.6**
- Data is unbalanced
 - “ragged array”
- How do we deal with this?

Unbalanced Data

- We could fill in the array:
 - Group 1: **13.2 NA NA NA**
 - Group 2: **12.3 14.1 NA NA**
 - Group 3: **11.0 9.7 10.3 9.6**

- Then the code is:

```
model {  
  for (i in 1:3) {  
    mu[i] ~ dunif(0, 100)  
    for (j in 1:4) { y[i, j] ~ dnorm(mu[i], 1) }  
  }  
}
```

Nested Indexing

- Filling in the array can be slow
 - missing values need to be estimated
 - missing values have are also in the likelihood
- As an alternative, we can use *nested indexing*
- For each observation, include an index for its group in the data
- This is quicker - we only loop through the data

Nested Indexing

- A better approach is to index the observations:

y: 13.2 12.3 14.1 11.0 9.7 10.3 9.6
person: 1 2 2 3 3 3 3

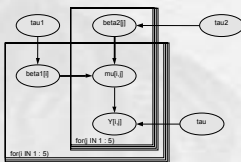
- And the code:

```
model {  
  for (i in 1:3) { mu[i] ~ dunif(0, 100) }  
  for (k in 1:7) { y[k] ~ dnorm(mu[person[k]], 1) }  
}
```

A (Silly) Model

- Suppose we have a model with two random effects:

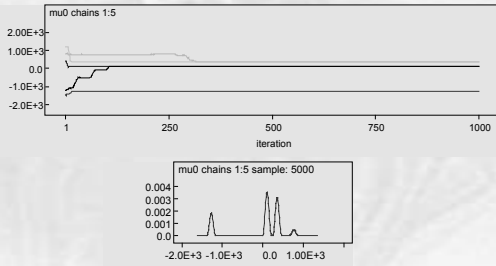
```
{ y[i] ~ dnorm(mu[i], tau)  
  mu[i] <- rand1 + rand2 }  
rand1 ~ dnorm(0, tau1)  
rand2 ~ dnorm(0, tau2)
```



- We can get the same value of $\mu[i]$ by:
 - setting rand1 to $\mu[i]$, and rand2 to 0, or
 - setting rand1 to 0, and rand2 to $\mu[i]$

The Results

- Run 5 chains, and get....



Identifiability

- In the previous model we could get the same likelihood in two ways
 - either rand1 or rand2 set to $\mu[i]$
- WinBugs can't choose between them
- The likelihood ($P(\gamma|\theta)$) is the same for both
- We say that they are *non-identifiable*
 - we can't identify them as unique
- Solution: change the model, or priors
