Class 3: More complex forms of missing data

Andrew Parnell
andrew.parnell@mu.ie

Maynooth University
National University of Ireland Maynooth

https://andrewcparnell.github.io/mda_course

In this class ...

- Revision of last week
- A bit more on Bayesian models and output
- Some different types of missingness: longitudinal data analysis and time series analysis
- Not missing at random models: pattern mixture models and selection models

A reminder of terms

- Missing completely at random (**MCAR**) means the cause of the missingness was completely unrelated to the data.
- Missing at random (**MAR**) means that the missingness only depends on the observed data. Given the observed information, the data are MCAR.
- Not missing at random (**NMAR** or MNAR) means the missingness depends on unobserved data that we do not have.

If you are in an MCAR or MAR situation where the parameters ψ and θ are not linked, the missingness is known as **ignorable**

Missing data analysis mathematical notation

Let:

- Y be the variables we are interested in, as a matrix of n observations and p variables
- Y_{obs} the observed data
- Y_{mis} the missing data
- M an $n \times p$ matrix that defines which observations/variables are missing, with $m_{ij} = 1$ if observation i and variable j are missing, and 0 otherwise
- ψ some parameters governing the missing data mechanism

Now:

- MCAR means

$$P(m = 1 | Y_{\text{obs}}, Y_{\text{mis}}, \psi) = P(m = 1 | \psi)$$

- MAR means

$$P(m = 1 | Y_{\text{obs}}, Y_{\text{mis}}, \psi)$$

$$= P(m = 1 | Y_{\text{obs}}, \psi)$$

- NMAR means

$$P(m = 1 | Y_{\text{obs}}, Y_{\text{mis}}, \psi)$$

depends on Y_{mis}

Reminder of JAGS code

```
model_code ='
model {
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dnorm(intercept + slope*x[i], residual_sd^-2)
  }
  # Priors
  intercept ~ dnorm(0,10^-2)
  slope ~ dnorm(0,10^-2)
  residual_sd ~ dunif(0,10)
}
'
```

Running a JAGS model

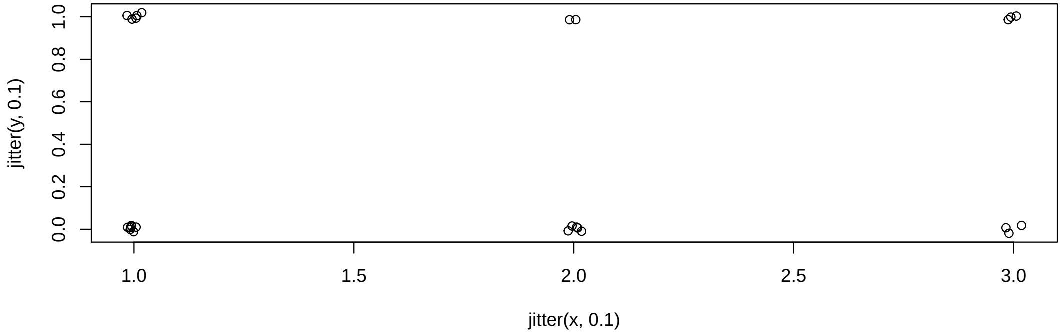
```r
library(R2jags)
mod_par =  c("intercept",
             "slope",
             "residual_sd")
model_run = jags(data =
                   list(N = length(y),
                        y = y, x = x),
                 parameters.to.save =
                   mod_par,
                 model.file =
                   textConnection(model_code))
```

JAGS for binary logistic regression

```
model_code = '
model
{
  # Likelihood
  for (i in 1:N) {
    y[i] ~ dbern(p[i])
    logit(p[i]) = alpha + beta * x[i]
  }
  # Priors
  alpha ~ dnorm(0, 5^-2)
  beta ~ dnorm(0, 5^-2)
}
'
```

Example for missing data

```
library(mice)
y = as.integer(is.na(nhanes$chl))
x = nhanes$age
plot(jitter(x, 0.1),jitter(y, 0.1))
```

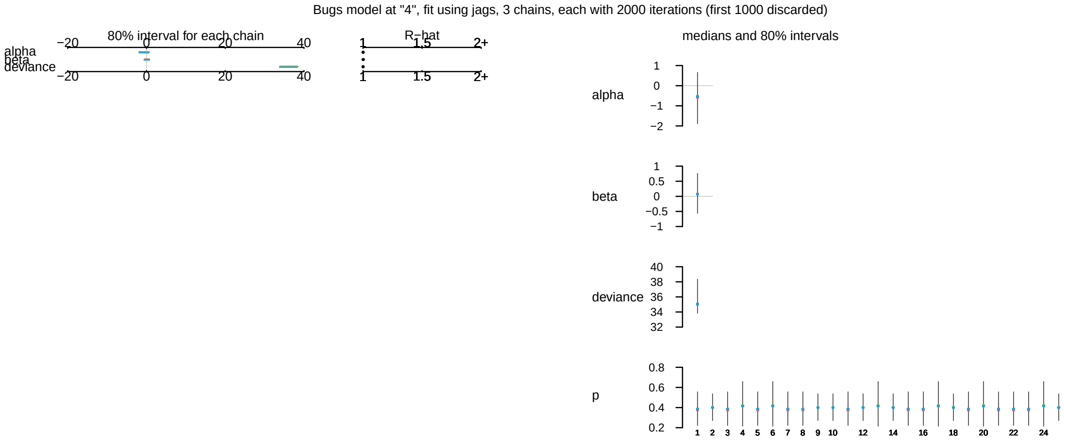


Running the JAGS model

```
library(R2jags)
mod_par =  c("p", "alpha", "beta")
model_run = jags(data =
                   list(N = length(y),
                        y = y, x = x),
                 parameters.to.save =
                   mod_par,
                 model.file =
                   textConnection(model_code))
```

Looking at the output

```r
plot(model_run)
```



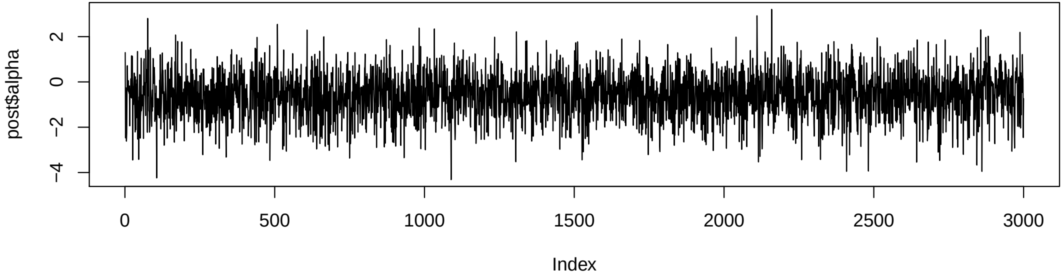
A quick note on JAGS output

- There are three key parts of the algorithm that affect how good the posterior samples are:
  1. The starting values you chose. If you chose bad starting values, you might need to discard the first few thousand iterations. This is known as the *burn-in* period
  2. The way you choose your new parameter values. If they are too close to the previous values the MCMC might move too slowly so you might need to *thin* the samples out by taking e.g. every 5th or 10th iteration
  3. The total number of iterations you choose. Ideally you would take millions but this will make the run time slower

Plotting the output

You can plot the iterations for all the parameters with `traceplot`, or for just one with e.g.

```
post = model_run$BUGSoutput$sims.list
plot(post$alpha, type = 'l')
```



A good trace plot will show no patterns or runs, and will look like it has a stationary mean and variance

How many chains?

- Beyond increasing the number of iterations, thinning, and removing a burn-in period, JAGS automatically runs *multiple chains*
- This means that they start the algorithm from 3 or 4 different sets of starting values and see if each *chain* converges to the same posterior distribution
- If the MCMC algorithm has converged then each chain should have the same mean and variance.
- JAGS reports the `Rhat` value, which is close to 1 when all the chains match
- It's about the simplest and quickest way to check convergence. If you get `Rhat` values above 1.1, run your MCMC for more iterations

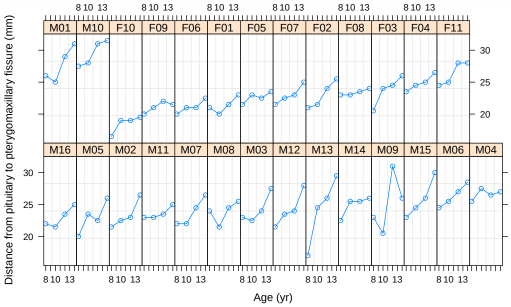
Missingness in longitudinal data

```r
library(nlme)
head(Orthodont)
```

```
## Grouped Data: distance ~ age | Subject
##   distance age Subject  Sex
## 1     26.0   8     M01 Male
## 2     25.0  10     M01 Male
## 3     29.0  12     M01 Male
## 4     31.0  14     M01 Male
## 5     21.5   8     M02 Male
## 6     22.5  10     M02 Male
```

These data are distance measures from x-ray images of the skull. We also have age, sex, and repeated individual measurements

```r
plot(Orthodont)
```



A simple model for these data

Fit a mixed effects (hierarchical) model of the form:

$$\text{distance}_{ij} \sim N(\alpha_j + \beta \times (\text{age}_{ij} - \bar{\text{age}}), \sigma^2)$$

$$\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2)$$

where:

- distance_{ij} is the distance measurement for observation i for individual j
- age_{ij} is the age associated observation i for individual j
- α_j is a random intercept term for each individual j with some overall mean μ_j and a variability between individuals σ_α

A JAGS model

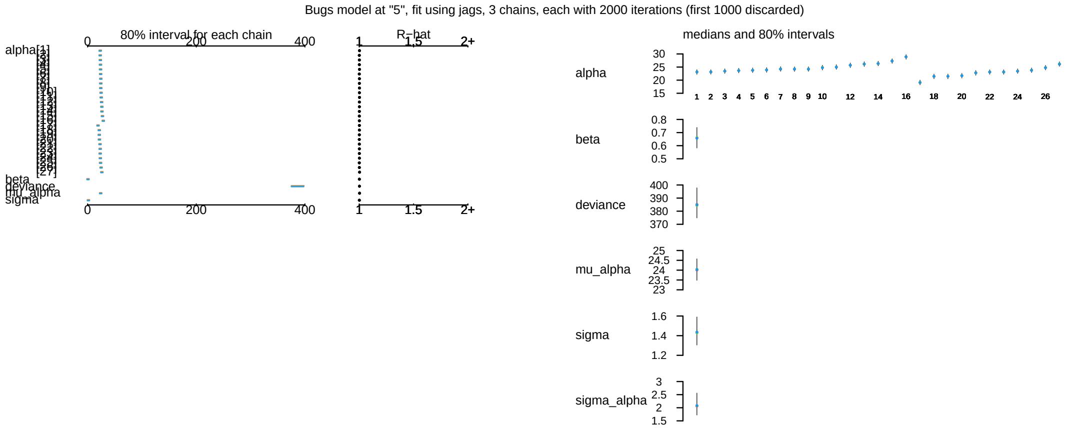
```
model_code = '
model
{
  # Likelihood
  for (i in 1:N) {
    y[i] ~ dnorm(alpha[person[i]] + beta * (age[i] - mean(age)), sigma^-2)
  }
  # Prior for intercept
  for(j in 1:N_people) {
    alpha[j] ~ dnorm(mu_alpha, sigma_alpha^-2)
  }
  # Priors on other parameters
  mu_alpha ~ dnorm(0, 100^-2)
  beta ~ dnorm(0, 10^-2)
  sigma ~ dgamma(1,1)
  sigma_alpha ~ dgamma(1,1)
}
```

Running this model

```r
mod_par =  c("mu_alpha", "alpha", "beta", "sigma", "sigma_alpha")
model_run = jags(data =
                   list(N = nrow(Orthodont),
                        N_people = length(unique(Orthodont$Subject)),
                        y = Orthodont$distance,
                        age = Orthodont$age,
                        person = Orthodont$Subject),
                 parameters.to.save = mod_par,
                 model.file = textConnection(model_code))
```

Output

```r
plot(model_run)
```



What if some of the data are missing?

- Some distance measurements or ages might be missing. (Subject being missing seems unlikely but possible)
- These might just be individual values (*item non-response* or *analysis dropout*), possibly MCAR or MAR
- We might get **treatment discontinuation** where some subjects no longer take part. This gets us back to a **monotone missingness pattern**
- We might get an alternative form of dropout if growth is irregular and subjects are removed form the study - this might be an example of MNAR
- All of these are pretty easy to model in JAGS (especially if MAR)

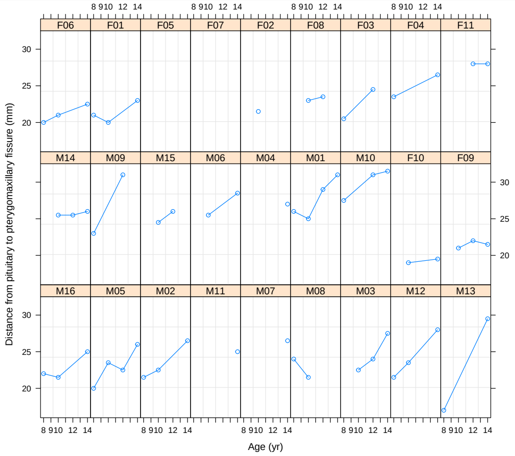
Adding missing values

Set a large number of the values across the data set to missing

```
set.seed(123)
Orthodont2 = Orthodont
Orthodont2$distance[sample(1:
    nrow(Orthodont2), 40)] = NA
Orthodont2$age[sample(1:
    nrow(Orthodont2), 20)] = NA
```

These data are distance measures from x-ray images of the skull. We also have age, sex, and repeated individual measurements

```
plot(Orthodont2)
```



New model

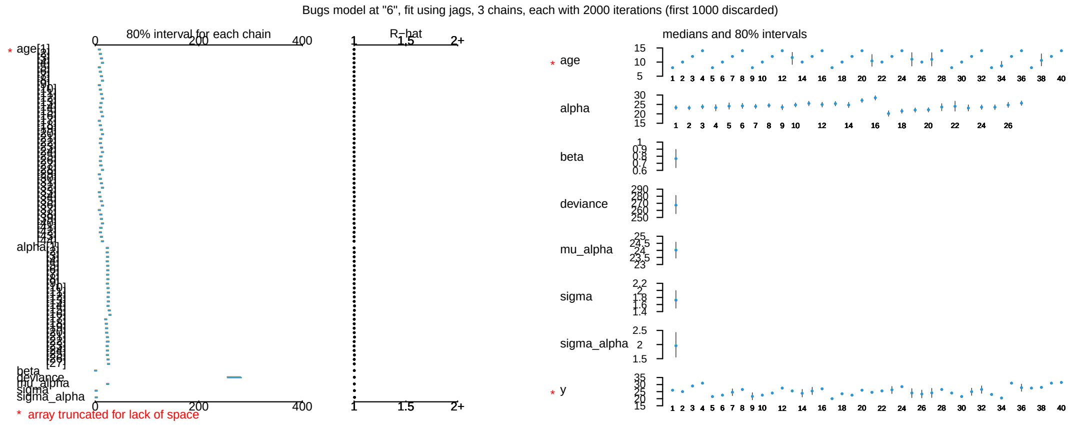
```
model_code = '
model
{
  # Likelihood
  for (i in 1:N) {
    y[i] ~ dnorm(alpha[person[i]] + beta * (age[i] - mean(age)), sigma^-2)
  }
  # Prior for intercept
  for(j in 1:N_people) {
    alpha[j] ~ dnorm(mu_alpha, sigma_alpha^-2)
  }
  # Prior for missing values
  for(k in 1:N_miss_age) {
    age[miss[k]] ~ dunif(min_age, max_age)
  }
  # Priors on other parameters
  mu_alpha ~ dnorm(0, 100^-2)
  beta ~ dnorm(0, 10^-2)
  sigma ~ dgamma(1,1)
  sigma_alpha ~ dgamma(1,1)
```

Running the model

```r
mod_par =  c("mu_alpha", "alpha", "beta", "sigma",
             "sigma_alpha", "y", "age")
model_run = jags(data =
                   list(N = nrow(Orthodont2),
                        N_people = length(unique(Orthodont2$Subject)),
                        N_miss_age = sum(is.na(Orthodont2$age)),
                        miss = which(is.na(Orthodont2$age)),
                        y = Orthodont2$distance,
                        age = Orthodont2$age,
                        min_age = min(Orthodont2$age, na.rm = TRUE),
                        max_age = max(Orthodont2$age, na.rm = TRUE),
                        person = Orthodont2$Subject),
                 parameters.to.save = mod_par,
                 model.file = textConnection(model_code))
```

Output

```
plot(model_run)
```



Missing data in time series models

- In a time series analysis we usually have a model of the form:

  $$y_t = f(y_t^-, X_t) + \epsilon_t$$

  where y_t^- represents previous values of the time series and X represents additional covariates
- The response y_t might be multivariate
- Missingness might occur in y, in X, or even in t
- Some tricks:
  - If you have many missing observations you could move to a continuous time series model
  - JAGS (and Stan) will usually work fine if there are missing values in the time series without any change to the model
  - With missing X values the approach on the previous slides will work fine
  - With missing t values this starts to get fiddly, but it's still possible provided you can put a prior distribution on t

The imputeTS package

- There is a nice R package for **single** imputation of time series called `imputeTS`
- It will impute the missing values in y_t via various methods:
    - `na_kalman` Missing Value Imputation by Kalman Smoothing
    - `na_locf` Missing Value Imputation by Last Observation Carried Forward
    - `na_ma` Missing Value Imputation by Weighted Moving Average
    - `na_seadec` Seasonally Decomposed Missing Value Imputation
    - `na_seasplit` Seasonally Splitted Missing Value Imputation
- Running the stochastic methods multiple times might allow for multiple imputation
- More details here: https://steffenmoritz.github.io/imputeTS/

Back to MNAR

- For MNAR data we need to build a model that accounts for the missingness mechanism as well as the prediction model
- There are two popular approaches **selection models** and **pattern-mixture models**
- The two approaches correspond to the way in which we factor the joint distribution
- For regression models, we now need to consider the full data set $y = (Y_{\text{obs}}, Y_{\text{mis}})$ so we are interested in finding the likelihood:

$$p(y, M|\theta, \psi, x) = \prod_{i=1}^{n} p(y_i, m_i|\theta, \psi, x_i)$$

Selection and pattern mixture models

- We are going to factor this model two different ways:
- The *Selection* model way:

$$p(y_i, m_i|\theta, \psi, x_i) = p(y_i|x_i, \theta) \times p(m_i|x_i, y_i, \psi)$$

Here the first term is just a standard regression model, and the second term is a binary regression which has y as a covariate (with x) in the model. **Both models have to be fitted at the same time**

- The *Pattern mixture* model way:

$$p(y_i, m_i|\theta, \psi, x_i) = p(y_i|x_i, m_i, \theta) \times p(m_i|x_i)$$

Now the first distribution is a regression model that is dependent on whether the data are missing or not, and the second is just a very simple logistic regression model based on covariates x

Issues with pattern mixture models

Here's a simple example. Suppose we define a pattern mixture model as:

$$y_i | m_i, x_i, \theta \sim N(\alpha^{(m_i)} + \beta^{(m_i)} x_i, (\sigma^{(m_i)})^2)$$

$$m_i | x_i, \psi \sim Binom(1, p_i),\ \text{logit}(p_i) = \gamma + \delta x_i$$

This means that there are different regression parameters for whether the data is missing or not. Of course, the problem is that you do not have any data y_i when the data is missing ($m_i = 1$)!!!

- **We need extra assumptions to fit this model**

Issues with selection models

Here's the same example as a selection model

$$y_i | x_i, \theta \sim N(\alpha + \beta x_i, \sigma^2)$$

$$m_i | x_i, y_i, \psi \sim Binom(1, p_i), \ \text{logit}(p_i) = \gamma + \delta x_i + \omega y_i$$

This means that the coefficients of the binary missingness indicator will also be hard to estimate if there is a high degree of missingness in y_i

(The probit version of this model is often known as a Heckman model)

Some notes on pattern mixture and selection models

- These models can extend to multivariate y values exactly as in the multiple imputation scenarios we met before.
- You can create hybrid versions of the two (called pattern-set mixture models) but this gets fiddly
- It's called a pattern mixture model because it's essentially a mixture of two regression models (one for the missing and one for the not missing data)
- I tend to find selection models easier to understand (and fit). Also if $\omega = 0$ in the previous slide then you end up back at MAR!
- It's hard to compare between the two approaches in most real-world scenarios

Some simplifications

- A simpler version of the pattern mixture model is:

$$y_i | m_i, x_i, \theta \sim N(\alpha^{(m_i)} + \beta x_i, \sigma^2)$$

$$m_i | x_i, \psi \sim Binom(1, p_i),\ \text{logit}(p_i) = \gamma + \delta x_i$$

  $\ldots$ so only a changing intercept between missing and non-missing data
- This means that a simple statistic such as $\alpha^{(m_1)} - \alpha^{(m_0)}$ will tell you how the missing data are shifted up or down compared to the observed data
- Little and Rubin argue that the pattern mixture model is better because the interpretation of the parameter ω in the selection model is harder, and because imputed data values are easier to simulate from

Including extra assumptions

These models often need extra assumptions to fit well. These might include:

- Extra (smaller) data sets on missing values, such as following up non-response
- Imposing restrictions on the model parameters
- Being Bayesian, and putting prior distributions on the parameters

Bayesian approaches to pattern mixture models

- A common approach is to put a prior distribution on the regression parameters for the missing observations
- In the simpler intercept only example we have:

$$y_i | m_i, x_i, \theta \sim N(\alpha^{(m_i)} + \beta x_i, \sigma^2)$$

- A good prior distribution might be:

$$\alpha^{(m_1)} \sim N(\alpha^{(m_0)}, \sigma_\alpha^2)$$

- This means it is centered around the non-missing intercept
- The parameter σ_α might also have a prior distribution
- It may also be set to be a function of $\alpha^{(m_0)}$, e.g. $\sigma_\alpha = k\alpha^{(m_0)}$

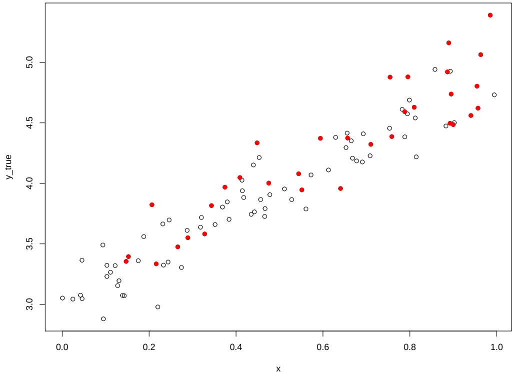
Example: a selection model in JAGS

```
model_code = '
model
{
  # Likelihood
  for (i in 1:N) {
    y[i] ~ dnorm(alpha + beta * x[i], sigma^-2)
    m[i] ~ dbern(p[i])
    logit(p[i]) = gamma + omega * (y[i] - mean(y))
  }
  # Priors for regression model
  alpha ~ dnorm(0, 100^-2)
  beta ~ dnorm(0, 100^-2)
  sigma ~ dgamma(1, 1)
  # Priors for missingness model
  gamma ~ dnorm(0, 5^-2)
  omega ~ dnorm(0, 5^-2)
}
```

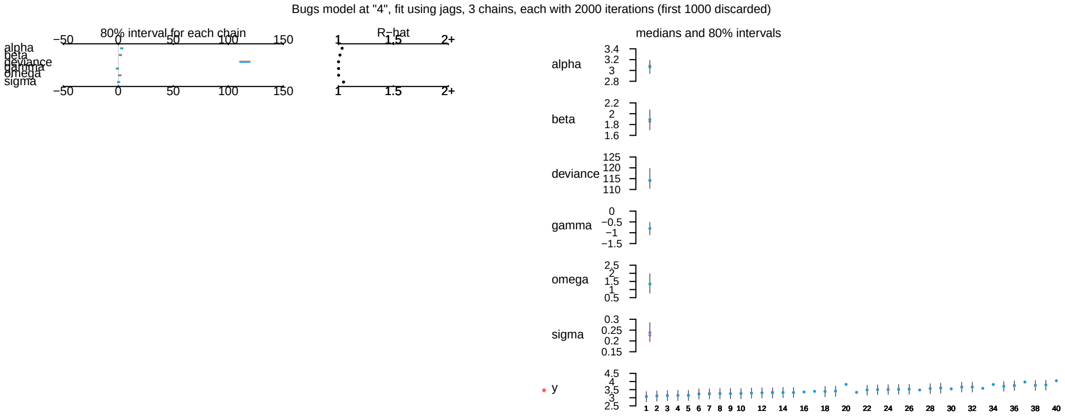
Some selection model NMAR data

```r
set.seed(123)
N = 100
x = sort(runif(N))
alpha = 3
beta = 2
sigma = 0.2
y_true = rnorm(N, alpha + beta * x,
               sigma)
gamma = -0.5
omega = 1
m = rbinom(N, 1, plogis(gamma +
      omega * (y_true -
               mean(y_true))))
y = y_true
y[m==0] = NA
```

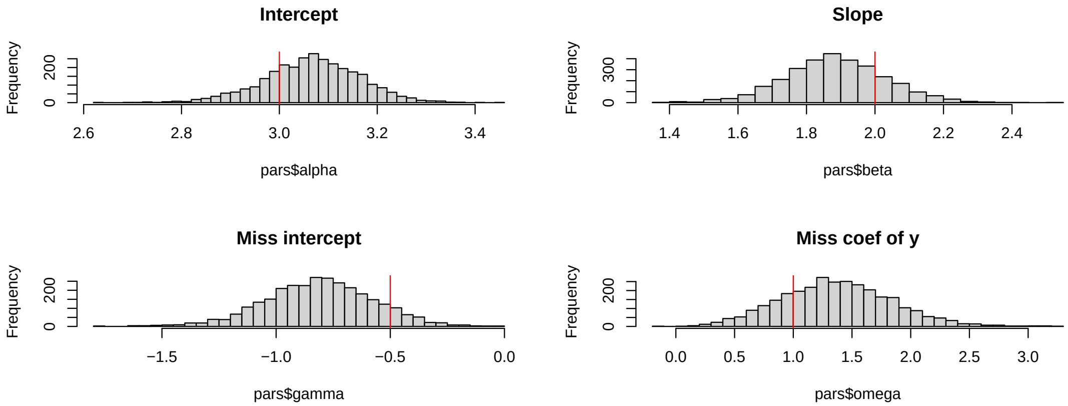
```r
plot(x, y_true)
points(x, y, col = 'red', pch = 19)
```



Output


Bugs model at "4", fit using jags, 3 chains, each with 2000 iterations (first 1000 discarded)
80% interval for each chain
R-hat
medians and 80% intervals
alpha
beta
deviance
gamma
omega
sigma
* y
−50
0
50
100
150
1
1.5
2+

How well did it estimate the model parameters



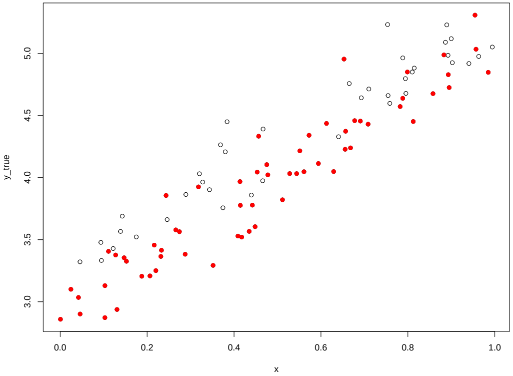
And now a pattern-mixture model

```
model_code = '
model
{
  # Likelihood
  for (i in 1:N) {
    # Note: miss[i] = 1 if not-missing and miss[i] = 2 if missing
    y[i] ~ dnorm(alpha[miss[i]] + beta * x[i], sigma^-2)
    m[i] ~ dbern(p[i])
    logit(p[i]) = gamma + delta * (x[i] - mean(x))
  }
  # Priors for regression model
  alpha[1] ~ dnorm(0, 100^-2)
  alpha[2] ~ dnorm(alpha[1], sigma_alpha^-2)
  beta ~ dnorm(0, 100^-2)
  sigma ~ dgamma(1, 1)
  sigma_alpha ~ dgamma(1, 1)
  # Priors for missingness model
  gamma ~ dnorm(0, 5^-2)
  delta ~ dnorm(0, 5^-2)
```

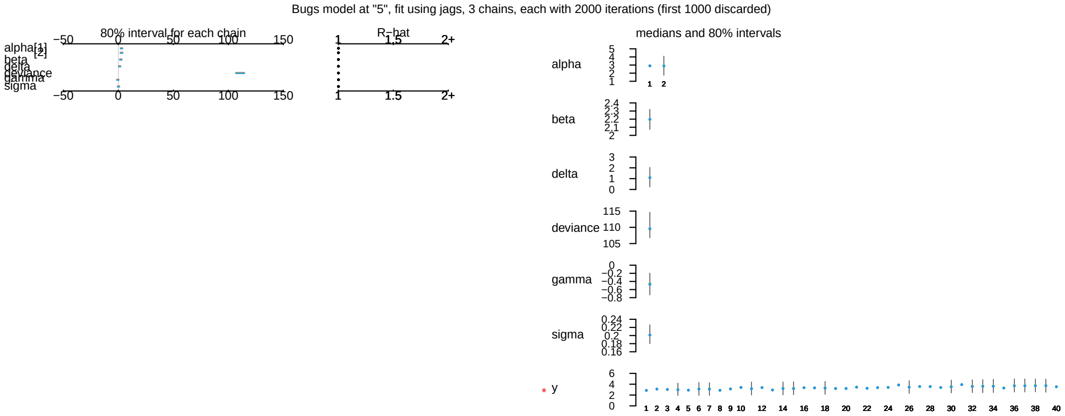
Some pattern mixture model NMAR data

```r
set.seed(123)
N = 100
x = sort(runif(N))
gamma = -0.5
delta = 1
m = rbinom(N, 1, plogis(gamma +
      delta * (x - mean(x)))) + 1
alpha = c(3,3.3)
beta = 2
sigma = 0.2
y_true = rnorm(N, alpha[m] +
          beta * x, sigma)
y = y_true
y[m==2] = NA
```

```r
plot(x, y_true)
points(x, y, col = 'red', pch = 19)
```

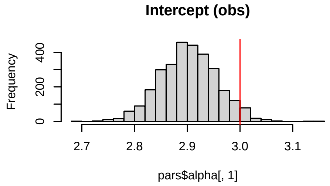


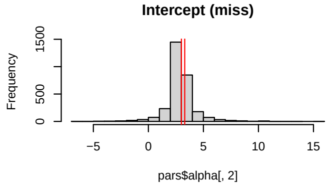
Output

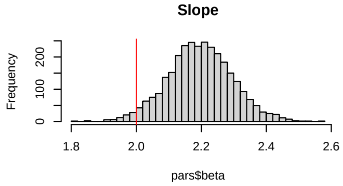


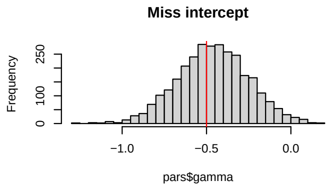
Bugs model at "5", fit using jags, 3 chains, each with 2000 iterations (first 1000 discarded)
80% interval for each chain
R-hat
medians and 80% intervals
alpha[1]
[2]
beta
delta
deviance
gamma
sigma
−50 0 50 100 150
1 1.5 2+
alpha
beta
delta
deviance
gamma
sigma
* y

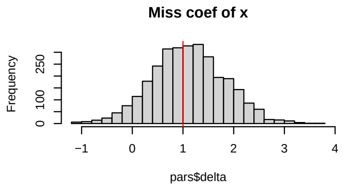
How well did it estimate the model parameters











Non-ignorable missing data in `mice`

- The `mice` package also allows for some NMAR type multiple imputation
- There is another package called `miceMNAR` that specialises in this
- It requires you to specify the regression-type model simultaneously with the imputation
- As before, it uses the fully conditional specification type model
- There are several different functions including `mice.impute.ri` and `mice.impute.mnar.logreg`
- Most of the functions seem to contain examples
- This is very new (only implemented in 2020)

Summary

- Missing not at random data is harder to fit because it usually requires some extra assumptions about the missingness mechanism and makes the model more complicated
- We have met the two main types; selection models and pattern mixture models
- They differ in how they decompose the likelihood, and how the results can be interpreted
- All these models can be fitted in JAGS and Stan; a version can be fitted in `mice`