Class 6: Hierarchical generalised linear models

Andrew Parnell
andrew.parnell@mu.ie

Maynooth University
National University of Ireland Maynooth

Learning outcomes:

- Understand the modelling implications of moving from linear to hierarchical generalised linear models (HGLMs)
- Know some of the different versions of Hierarchical GLMs
- Be able to fit HGLMS in JAGS
- Be able to expand and summarise fitted models

From LMs to HGLMs

- Reminder: a hierarchical model has prior distributions on the parameters which depend on further parameters
- A generalised linear model is one in which the probability distribution is not normal, and a link function serves to match the mean of the distribution to the covariates
- Within this framework, we can borrow the ideas from the previous class to create hierarchical GLMs
- We will go through four examples: binomial-logit, Poisson, robust regression, and ordinal regression

Example 1: binomial-logit

- In class 2, we met the Binomial-logit model for binary data:

  $$y_i \sim Bin(1, p_i), logit(p_i) = \alpha + \beta(x_i - \bar{x})$$

  Here $logit(p_i)$ is the link function equal to $\log\left(\frac{p_i}{1-p_i}\right)$ and transforms the bounded probabilities into an unbounded space

- If we have non-binary data we just change the likelihood:

  $$y_i \sim Bin(N_i, p_i), logit(p_i) = \alpha + \beta(x_i - \bar{x})$$

- In a hierarchical version of this model, we vary the *latent parameters* α and β and give them prior distributions

The swiss willow tit data

```
swt = read.csv('../data/swt.csv')
head(swt)
```

```
##   rep.1 rep.2 rep.3 c.2 c.3 elev forest dur.1 dur.2 dur.3 length alt
## 1     0     0     0   0   0  420      3   240    58    73    6.2 Low
## 2     0     0     0   0   0  450     21   160    39    62    5.1 Low
## 3     0     0     0   0   0 1050     32   120    47    74    4.3 Med
## 4     0     0     0   0   0 1110     35   180    44    71    5.4 Med
## 5     0     0     0   0   0  510      2   210    56    73    3.6 Low
## 6     0     0     0   0   0  630     60   150    56    73    6.1 Low
```

A hierarchical model

- Suppose we want to fit a model on the sum $y_i =$ `rep.1 + rep.2 + rep.3`:

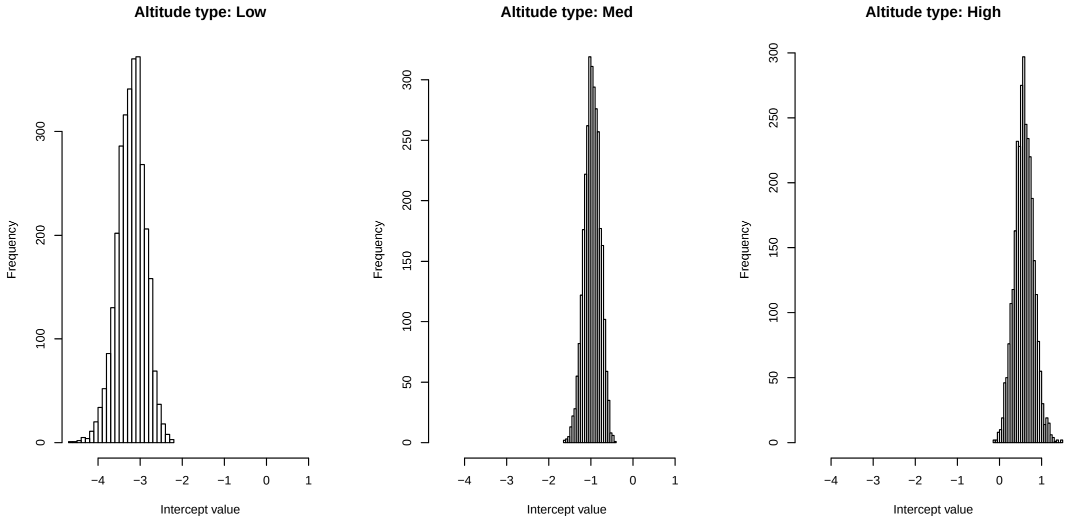
  $$y_i \sim Bin(N_i, p_i), logit(p_i) = \alpha_{\text{altitude}_i} + \beta_{\text{altitude}_i}(x_i - \bar{x})$$

  where x_i is the percentage of forest cover
- What prior distributions should we use for α and β?
- Useful side note: A value of 10 on the logit scale leads to a probability of about 1, and a value of -10 leads to a probability of about 0 (you can test this by typing `inv.logit(10)`) so I wouldn't expect the value of $logit(p_i)$ to ever get much bigger than 10 or smaller than -10
- I have no idea whether we are more likely to find these birds in high percentage forest or low, so I'm happy to think that β might be around zero, and be positive or negative. Forest cover ranges from 0 to 100 so that suggests that β is very unlikely to be bigger than 0.1 or smaller than -0.1. Perhaps $\beta \sim N(0, 0.1^2)$ is a good prior
- It looks to me like the intercept is very unlikely to be outside the range (-10, 10) so perhaps $\alpha \sim N(0, 5^2)$ is appropriate

JAGS code

```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dbin(p[i], N_exp[i])
    logit(p[i]) <- alpha[alt[i]] + beta[alt[i]]* (x[i] - mean(x))
  }
  # Priors
  for(j in 1:N_alt) {
    alpha[j] ~ dnorm(mu_alpha, sigma_alpha^-2)
    beta[j] ~ dnorm(mu_beta, sigma_beta^-2)
  }
  mu_alpha ~ dnorm(0, 5^-2)
  mu_beta ~ dnorm(0, 0.1^-2)
  sigma_alpha ~ dt(0, 5^-2, 1)T(0,)
  sigma_beta ~ dt(0, 5^-2, 1)T(0,)
}
'
```

Model fit - intercepts

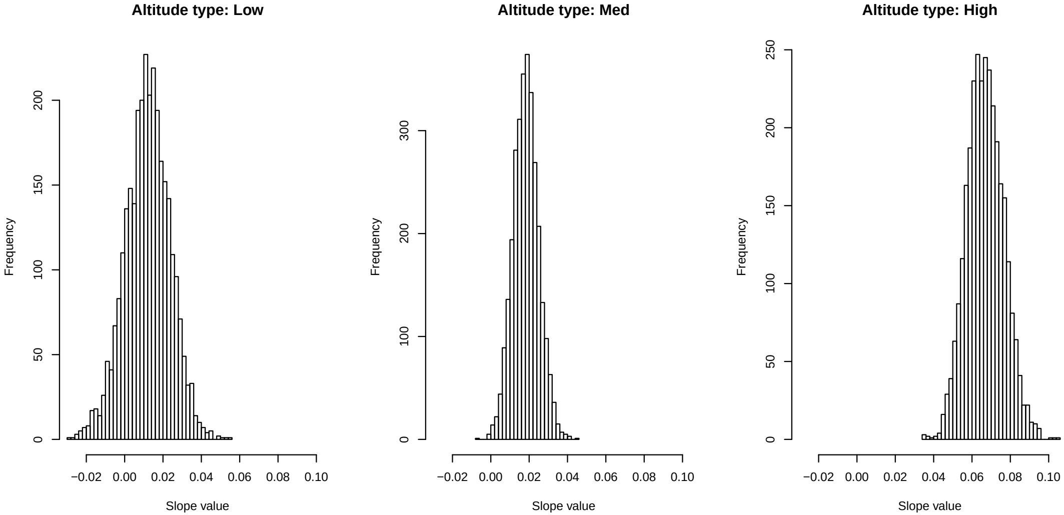


Altitude type: Low

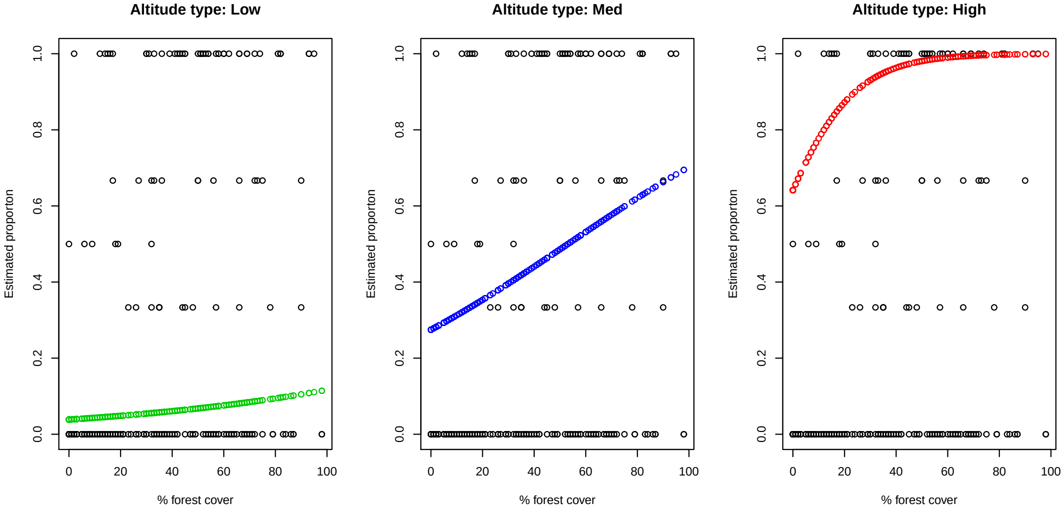
Altitude type: Med

Altitude type: High

Model fit - Slopes



Model fit - estimated mean proportions



Type 2: Poisson HGLMs

- For a Poisson distribution there is no upper bound on the number of counts
- We just change the likelihood (to Poisson) and the link function (to log):

$$y_i \sim Po(\lambda_i), \log(\lambda_i) = \alpha + \beta(x_i - \bar{x}))$$

- We can now add our hierarchical layers into α and β, or...
- Another way we can add an extra layer is by giving $\log(\lambda_i)$ a probability distribution rather than setting it to a value
- This is a way of introducing *over-dispersion*, i.e. saying that the data are more variable than that expected by a standard Poisson distribution with our existing covariates

An over-dispersed model

- The over-dispersed model looks like:

$$y_i \sim Po(\lambda_i), \log(\lambda_i) \sim N(\alpha + \beta(x_i - \bar{x}), \sigma^2)$$

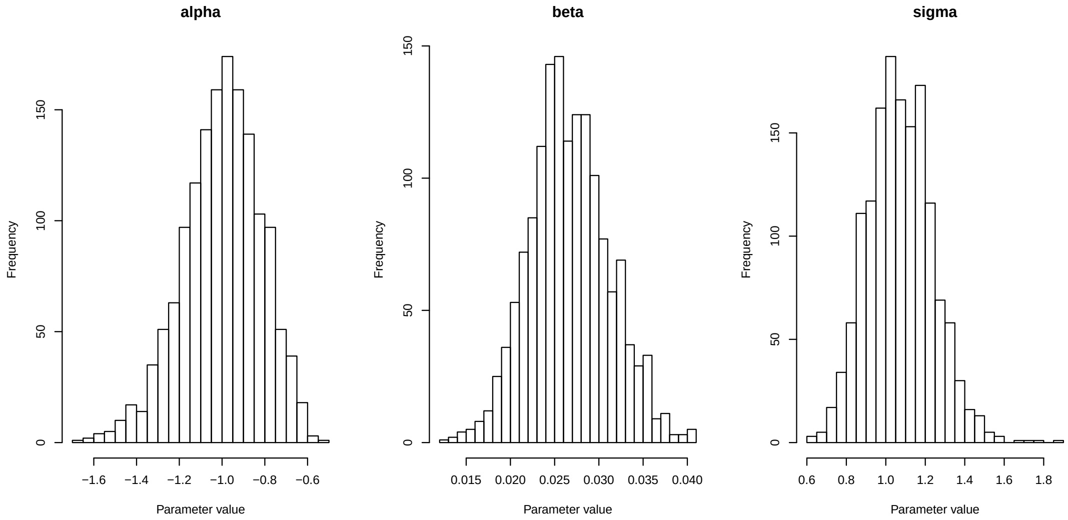
  where σ is the over-dispersion parameter

- We now need to estimate prior distributions for α, β, and σ
- We will use the SWT data again, but pretend that we didn't know that they had gone out N times looking for the birds

JAGS code for OD Poisson

```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dpois(exp(log_lambda[i]))
    log_lambda[i] ~ dnorm(alpha + beta * (x[i] - mean(x)),
        sigma^-2)
  }
  alpha ~ dnorm(0, 5^-2)
  beta ~ dnorm(0, 0.1^-2)
  sigma ~ dt(0, 5^-2, 1)T(0,)
}
'
```

Model run


alpha
Frequency
0
50
100
150
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6
Parameter value
beta
Frequency
0
50
100
150
0.015
0.020
0.025
0.030
0.035
0.040
Parameter value
sigma
Frequency
0
50
100
150
0.6
0.8
1.0
1.2
1.4
1.6
1.8
Parameter value

Notes about OD Poisson model

- The way to think about OD models is via the data generating process.
- We could compare this model to one without over dispersion via DIC (or if time, cross validation). We should also compute a posterior predictive distribution for full comparison
- In general, the parameter values (i.e. alpha and beta) tend to be more uncertain when you add in over dispersion
- Also in the data set is a variable called `dur` which represents how long they spent looking for the birds. This could be added in as an offset via the likelihood:

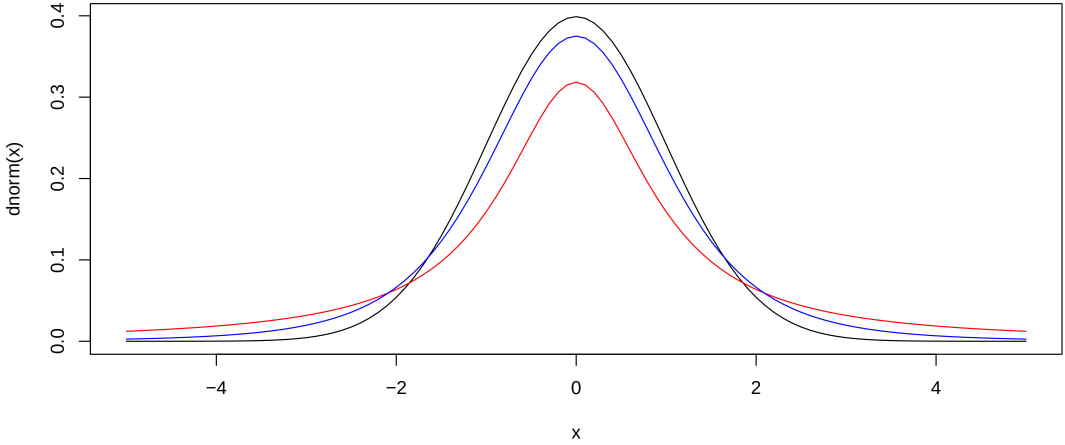
```
y[i] ~ dpois(dur[i] * exp(log_lambda[i]))
```

Type 3: *t*-distributed HGLMs

- How do Bayesians deal with outliers?
- A common view is that we should delete these observations before we run the model, but what if we can't find a reason for doing so
- A good Bayesian will include outliers as part of the model.
- One way of doing this is by switching from a normal distribution to a *t*-distribution

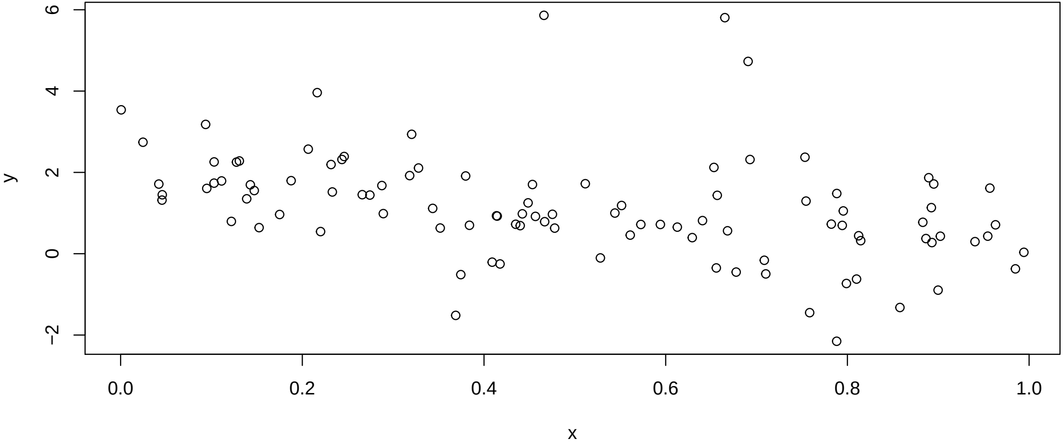
Normal vs *t*

```r
curve(dnorm, from = -5, to = 5)
curve(dt(x, df = 1), add = TRUE, col = 'red')
curve(dt(x, df = 4), add = TRUE, col = 'blue')
```



Polluted data

- Suppose we had some data which looked like this:



There are a few observations here which look a bit odd

JAGS code for a *t*-model

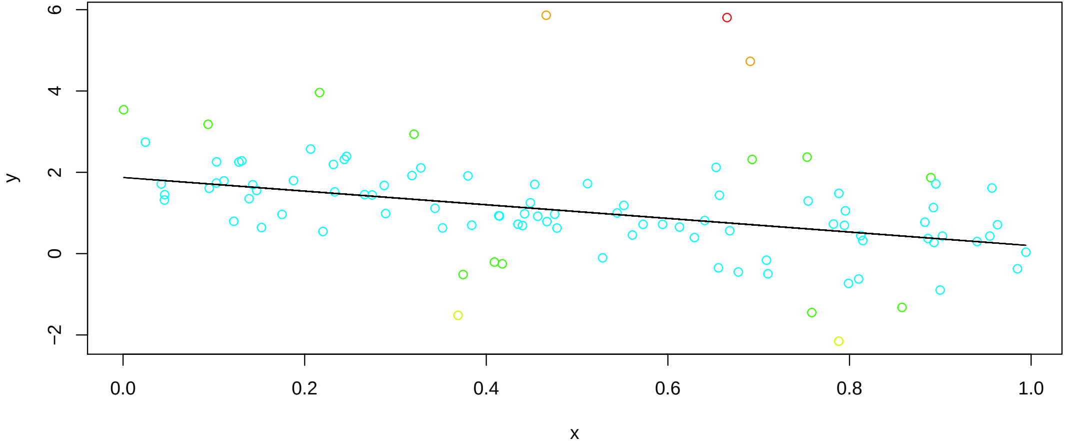
```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dt(alpha + beta * (x[i] - mean(x)),
                sigma^-2, df[i])
    df[i] ~ dcat(p)
  }
  alpha ~ dnorm(0, 1^-2)
  beta ~ dnorm(0, 1^-2)
  sigma ~ dt(0,1,1)T(0,)
}
'
```

Fitting the model

```r
jags_run = jags(data = list(N = N,
                            p = rep(1,10)/10,
                            y = y,
                            x = x),
                parameters.to.save = c('alpha',
                                       'beta',
                                       'df'),
                model.file = textConnection(jags_code))
```

Output from the model

```
dfs = jags_run$BUGSoutput$median$df
pars = jags_run$BUGSoutput$mean
cols = rainbow(10)
plot(x, y, col = cols[dfs])
lines(x, as.numeric(pars$alpha) +
        as.numeric(pars$beta)*(x - mean(x)))
```



Prior distributions on the degrees of freedom

- Here I've set a prior distribution on the degrees of freedom parameter to be a categorical distribution with probabilities 0.1 for df $= 1, 2, \ldots, 10$
- Smaller values of `df` mean that a data point is more likely to be an outlier
- The categorical distribution automatically looks up the right `df` value for each probability
- This model is impossible to fit in Stan, because it contains a discrete parameter

Type 4: Ordinal data HGLMs

- Often we have a response variable which is ordinal, e.g. disagree, neutral, agree, etc
- There are lots of different (and complicated) ways to model such data
- Perhaps the easiest is to think of it as a hierarchical model with ‘cut-points’ on a latent linear regression

An ordinal model example

- Suppose $y_i = \{\text{disagree, neutral, agree}\}$ and we make it dependent on a latent continuous variable z_i, so that :

$$y_i = \begin{cases} \text{agree} & \text{if } z_i > 0.5 \\ \text{neutral} & \text{if } -0.5 < z_i \leq 0.5 \\ \text{disagree} & \text{if } z_i \leq -0.5 \end{cases}$$

- We then give z_i a prior distribution, e.g. $N(\beta_0 + \beta_1 x_i, \sigma^2)$

Fitting ordinal models in JAGS

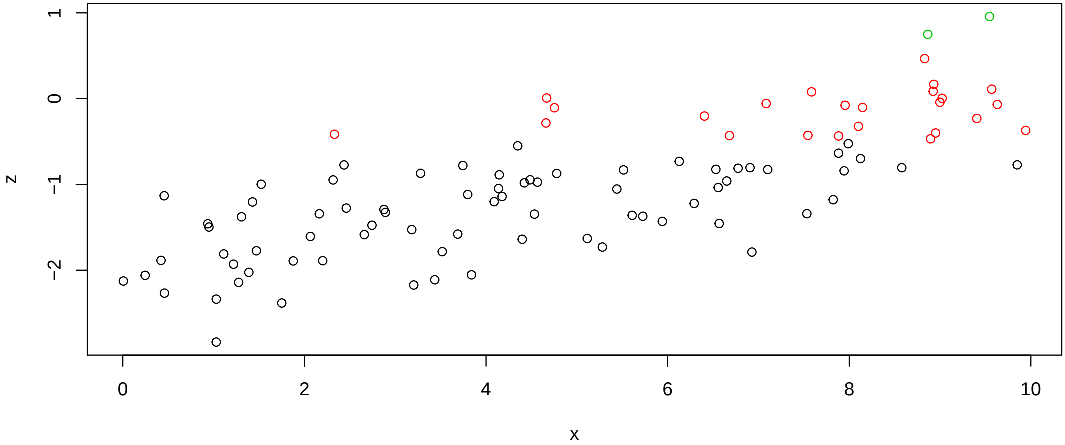
```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    z[i] ~ dnorm(alpha + beta * (x[i] - mean(x)),
                    sigma^-2)
    y[i] ~ dinterval(z[i], cuts)
  }
  alpha ~ dnorm(0, 100^-2)
  beta ~ dnorm(0, 100^-2)
  sigma ~ dt(0, 10^-2, 1)T(0, )
}
'
```

Simulating some example data

```r
N = 100
alpha = -1
beta = 0.2
sigma = 0.51
set.seed(123)
x = runif(N, 0, 10)
cuts = c(-0.5, 0.5)
z = rnorm(N, alpha + beta * (x - mean(x)), sigma)
y = findInterval(z, cuts)
```

Simulated data - plot

```
plot(x, z, col = y + 1)
```



Fitting in JAGS - needs initial values

```
jags_inits = function() {
  z = runif(N, -0.5, 0.5)
  z[y==0] = runif(sum(y==0), -1, -0.5)
  z[y==2] = runif(sum(y==2), 0.5, 1)
  return(list(z = z))
}
jags_run = jags(data = list(N = N,
                            y = y,
                            x = x,
                            cuts = cuts),
                inits = jags_inits,
                parameters.to.save = c('alpha',
                                       'beta',
                                       'sigma'),
                model.file = textConnection(jags_code))
```

Output

```r
print(jags_run)
```

```
## Inference for Bugs model at "7", fit using jags,
##  3 chains, each with 2000 iterations (first 1000 discarded)
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## alpha     -1.093   0.181 -1.531 -1.188 -1.068 -0.962 -0.815 1.052    46
## beta       0.221   0.052  0.135  0.186  0.215  0.250  0.345 1.044    51
## sigma      0.559   0.128  0.365  0.468  0.538  0.627  0.859 1.027    81
## deviance   0.000   0.000  0.000  0.000  0.000  0.000  0.000 1.000     1
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 0.0 and DIC = 0.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Summary

- We have now seen a number of different types of hierarchical GLM
- Many of the ideas of hierarchical linear models transfer over, but we can explore richer behaviour with hierarchical GLMs
- These have all used the normal, binomial or Poisson distribution at the top level, and have allowed for over-dispersion, robustness, and ordinal data, to name just three