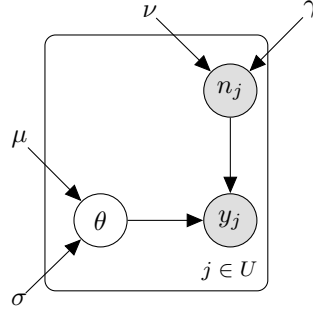


Modeling Profiles and Recommendation List

Modeling the Profile

We model user profiles with a hierarchical model summarized below:



Input Data

Each user has a profile; all we need is the (known-gender) profile size (n_j) and the number of female authors (y_j). J is the number of users.

```
int<lower=0> J;
int<lower=5> n[J];
int<lower=0> y[J];
```

The smallest profile size is 5, by experimental design; therefore, want a shifted profile data piece.

```
int<lower=0> shiftN[J];
for (i in 1:J) {
    shiftN[i] = n[i] - 5;
}
```

Modeling Profile Sizes

In order to realistically do predictive modeling, our model must model user profile sizes as being drawn from a distribution. A Poisson distribution has too much (and uncontrollable) variance, so we will use a negative binomial with mean ν and dispersion γ . Further, since we have a minimum size of 5, we will apply the distribution to the shifted values.

We tested several different ways of modeling profile sizes, and a negative binomial on shifted sizes had the best fit. This means that ν will not quite be the true mean.

Parameter definitions:

```
real<lower=0> nMean;  
real<lower=0> nDisp;
```

And the likelihood model:

```
shiftN ~ neg_binomial_2(nMean, nDisp);
```

We'll place a vague prior on both the mean and dispersion.

```
nMean ~ exponential(0.001);  
nDisp ~ exponential(0.001);
```

Modeling Profile Observations

Following a hierarchical model, each user has a profile bias θ_j . We model the observed author distribution with a binomial:

$$y_j \sim \text{Binomial}(n_j, \theta_j)$$

```
y ~ binomial(n, theta);
```

Modeling Profile Bias

The common way to model the distribution of θ_j is with a Beta distribution, as in Gelman. However, a logit-normal is more computationally efficient, and will be more conceptually consistent when we go to do a regression. Also, we tested both a beta model and a logit-normal model, and found the logit-normal model to have moderately better fit.

Therefore, we will model bias as the logit transform of a normal variable with parameters μ and σ^2 :

$$\begin{aligned}\theta'_j &= \text{logit}(\theta_j) \\ \theta'_j &\sim \text{Normal}(\mu, \sigma)\end{aligned}$$

So we define our parameters;

```
real mu;  
real<lower=0> sigma;  
vector[J] nTheta;
```

Some transformed parameters:

```
vector<lower=0,upper=1>[J] theta;
```

Their transformation:

```
theta = inv_logit(nTheta);
```

And then a likelihood model:

```
nTheta ~ normal(mu, sigma);
```

Finally, we will place vague priors over μ and σ :

```
mu ~ normal(0, 100);
```

```
sigma ~ exponential(0.001);
```

Generating Predictions

Finally, in order to compare posterior predictions with observed data, we want to draw samples from the fitted model. We can do this along with the fitting, with the following generator code.

```
real nThetaP;  
real<lower=0, upper=1> thetaP;  
int<lower=5> nP = 0;  
int<lower=0> yP;  
  
nThetaP = normal_rng(mu, sigma);  
thetaP = inv_logit(nThetaP);  
nP = neg_binomial_2_rng(nMean, nDisp) + 5;  
yP = binomial_rng(nP, thetaP);
```

The Model

For diagnostic purposes, we want to fit just the profiles, before we try to fit the recommendation lists too.

```
// This file is generated from models.md  
data {  
    @profile_data  
}  
transformed data {  
    @profile_xdata  
    @profile_data_transforms  
}  
parameters {  
    @profile_params  
}  
transformed parameters {  
    @profile_xparams  
    @profile_transforms  
}  
model {
```

```

    // prior distribution
    @profile_prior
    // likelihood model
    @profile_model
  }
  generated quantities {
    @profile_predict_vars
    @profile_predict_model
  }

```

Recommendation Lists

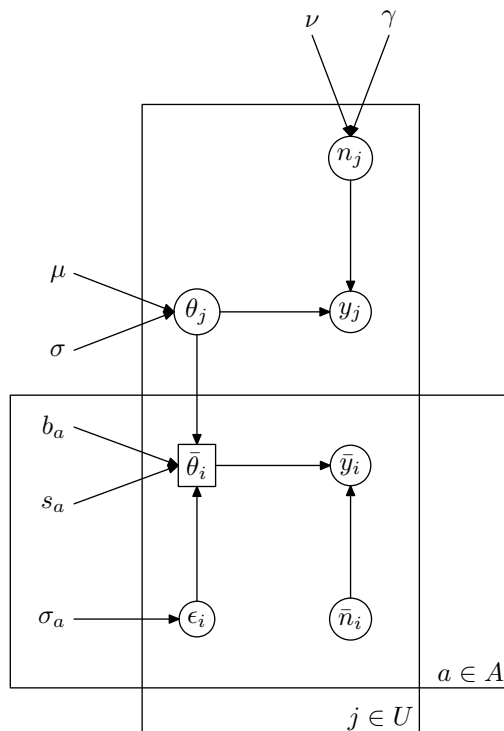


Figure 1: Plate diagram for profile and recommendation lists

Input Data

We need a way to represent recommendation lists in the system. I would prefer to use a matrix; however, not every user has recommendations from every algorithm, so it would be an incomplete matrix; Stan does not like incomplete data.

Therefore we represent each list as a data point, and include its algorithm.

We need the sizes - the number of algorithms and lists:

```
int<lower=0> A;
int<lower=0> NL;
```

And we need the data itself - the recommender user \bar{u}_i (**ru**), the recommender algorithm \bar{a}_i (**ra**), the recommendation (known author) list size \bar{n}_i (**rn**), and the number of female authors \bar{y}_i (**ry**).

```
int<lower=1,upper=J> ru[NL];
int<lower=1,upper=A> ra[NL];
int<lower=0> rn[NL];
int<lower=0> ry[NL];
```

We also want the observed proportion of female authors.

```
vector<lower=0,upper=1>[NL] rp;
vector[NL] rll;

rp = (to_vector(ry) + 1) ./ (to_vector(rn) + 2);
rll = logit(rp);
```

Modeling Recommender Response

Again, we will use a binomial to model the distribution of observed female proportions in recommender output lists, such that:

$$\bar{y}_i \sim \text{Binomial}(\bar{n}_i, \bar{\theta}_i)$$

$$\bar{\theta}'_i = \text{logit}(\bar{\theta}_i)$$

We want to just write this:

```
ry ~ binomial(rn, thetaR);
```

However, due to our consistency accomodation, we can't. See the next section for details on that.

The key to relating recommender output to user profile properties is through a linear regression in the logit space, the space in which user biases are taken to be normally distributed. We do this relation in terms of a recommender baseline bias b_a , response slope s_a , and variance σ_a^2 . This is defined by:

$$\bar{\theta}'_i = b_a + s_a \theta'_{u_i} + \epsilon_i$$

$$\epsilon_i \sim \text{Normal}(0, \sigma_a)$$

So we need a noise parameter:

```
vector[NL] rbias;
vector[NL] noiseR;
```

And our recommender response parameters:

```
vector[A] recB;
vector[A] recS;
vector<lower=0>[A] recV;
```

And then we need to connect these to the user profile θ s and the recommender properties.

```
rbias = recB[ra] + recS[ra] .* nTheta[ru];
noiseR = rll - rbias;
```

Noise is normal:

```
noiseR ~ normal(0, recV[ra]);
```

And finally we put vague priors on our recommender response parameters. These will be vague.

```
recB ~ normal(0, 100);
recS ~ normal(0, 100);
recV ~ exponential(0.001);
```

Predicting Lists

We now want to draw a synthetic user, so that we can examine posterior predictive distributions. We have already written the code to do this for a new user; we now generate their lists.

```
vector[A] biasP;
vector[A] noiseP;
vector[A] thetaRP;

biasP = recB + recS * nThetaP;
for (a in 1:A) {
  noiseP[a] = normal_rng(0, recV[a]);
}
thetaRP = inv_logit(noiseP + biasP);
```

The Model

Now we want to put all of this together into a model.

```
// This file is generated from models.md
data {
  @profile_data
  @list_data
```

```

}
transformed data {
    @profile_xdata
    @list_xdata
    @profile_data_transforms
    @list_data_transforms
}
parameters {
    @profile_params
    @list_params
}
transformed parameters {
    @profile_xparams
    @list_xparams
    @profile_transforms
    @list_transforms
}
model {
    // profile prior distribution
    @profile_prior
    // rec list priors
    @list_priors

    // profile likelihood model
    @profile_model

    // rec list likelihood model
    @list_model
}
generated quantities {
    @profile_predict_vars
    @list_predict_vars
    @profile_predict_model
    @list_predict_model
}

```