Generalized Measurement Models: Modeling Observed Polytomous Data

Lecture 4d

Today's Lecture Objectives

1. Show how to estimate unidimensional latent variable models with polytomous data*

- Also known as Polytomous Item Repsonse Theory (IRT) or Item Factor Analysis (IFA) models

2. Distributions appropriate for polytomous (discrete; data with lower/upper limits)

Example Data: Conspiracy Theories

Today's example is from a bootstrap resample of 177 undergraduate students at a large state university in the Midwest. The survey was a measure of 10 questions about their beliefs in various conspiracy theories that were being passed around the internet in the early 2010s. Additionally, gender was included in the survey. All items responses were on a 5- point Likert scale with:

1. Strongly Disagree
2. Disagree
3. Neither Agree or Disagree
4. Agree
5. Strongly Agree

Please note, the purpose of this survey was to study individual beliefs regarding conspiracies. The questions can provoke some strong emotions given the world we live in currently. All questions were approved by university IRB prior to their use.

Our purpose in using this instrument is to provide a context that we all may find relevant as many of these conspiracy theories are still prevalent today.

Conspiracy Theory Questions 1-5

Questions:

1. The U.S. invasion of Iraq was not part of a campaign to fight terrorism, but was driven by oil companies and Jews in the U.S. and Israel.
2. Certain U.S. government officials planned the attacks of September 11, 2001 because they wanted the United States to go to war in the Middle East.
3. President Barack Obama was not really born in the United States and does not have an authentic Hawaiian birth certificate.
4. The current financial crisis was secretly orchestrated by a small group of Wall Street bankers to extend the power of the Federal Reserve and further their control of the world's economy.
5. Vapor trails left by aircraft are actually chemical agents deliberately sprayed in a clandestine program directed by government officials.

Conspiracy Theory Questions 6-10

Questions:

6. Billionaire George Soros is behind a hidden plot to destabilize the American government, take control of the media, and put the world under his control.
7. The U.S. government is mandating the switch to compact fluorescent light bulbs because such lights make people more obedient and easier to control.
8. Government officials are covertly Building a 12-lane "NAFTA superhighway" that runs from Mexico to Canada through America’s heartland.
9. Government officials purposely developed and spread drugs like crack-cocaine and diseases like AIDS in order to destroy the African American community.
10. God sent Hurricane Katrina to punish America for its sins.

From Previous Lectures: CFA (Normal Outcomes)

For comparisons today, we will be using the model where we assumed each outcome was (conditionally) normally distributed:
For an item i the model is:

$$Y_{pi} = \mu_i + \lambda_i \theta_p + e_{p,i}; \quad e_{p,i} \sim N\left(0, \psi_i^2\right)$$

Recall that this assumption wasn't a good one as the type of data (discrete, bounded, some multimodality) did not match the normal distribution assumption

Plotting CFA (Normal Outcome) Results

Polytomous Data Distributions

Polytomous Data Characteristics

As we have done with each observed variable, we must decide which distribution to use

- To do this, we need to map the characteristics of our data on to distributions that share those characteristics

Our observed data:

- Discrete responses
- Small set of known categories: $1, 2, 3, 4, 5$
- Some observed item responses may be multimodal

Choice of distribution must match

- Be capable of modeling a small set of known categories
  - Discrete distribution
  - Limited number of categories (with fixed upper bound)
- Possible multimodality

Discrete Data Distributions

Stan has a list of distributions for bounded discrete data: **https://mc-stan.org/docs/functions-reference/bounded-discrete-distributions.html**

- Binomial distribution
  - Pro: Easy to use/code
  - Con: Unimodal distribution
- Beta-binomial distribution
  - Not often used in psychometrics (but could be)
  - Generalizes binomial distribution to have different probability for each trial
- Hypergeometric distribution
  - Not often used in psychometrics
- Categorical distribution (sometimes called multinomial)
  - Most frequently used
  - Base distribution for graded response, partial credit, and nominal response

models

- Discrete range distribution (sometimes called uniform)
  - Not useful–doesn’t have much information about latent variables

Binomial Distribution Models

Binomial Distribution Models

The binomial distribution is one of the easiest to use for polytomous items

- However, it assumes the distribution of responses are unimodal

Binomial probability mass function (i.e., pdf):

$$P(Y = y) = \binom{n}{y} p^k (1-p)^{(n-k)}$$

Parameters:

- n – "number of trials" (range: $n \in \{0, 1, \dots\}$)
- y – "number of successes" out of n "trials" (range: $y \in \{0, 1, \dots, n\}$)
- p – probability of "success" (range: $[0, 1]$)

Mean: np
Variance: $np(1-p)$

Adapting the Binomial for Item Response Models

Although it doesn't seem like our items fit with a binomial, we can actually use this distribution

- Item response: number of successes y_i
  - Needed: recode data so that lowest category is 0 (subtract one from each item)
- Highest (recoded) item response: number of trials n
  - For all our items, once recoded, $n_i = 4$ $(\forall i)$
- Then, use a link function to model each item's p_i as a function of the latent trait:

$$p_i = \frac{\exp\left(\mu_i + \lambda_i \theta_p)\right)}{1 + \exp\left(\mu_i + \lambda_i \theta_p)\right)}$$

Note:

- Shown with a logit link function (but could be any link)
- Shown in slope/intercept form (but could be discrimination/difficulty for

unidimensional items)

- Could also include asymptote parameters (c_i or d_i)

Binomial Item Repsonse Model

The item response model, put into the PDF of the binomial is then:

$$P(Y_{pi} \mid \theta_p) = \binom{n_i}{Y_{pi}} \left(\frac{\exp\left(\mu_i + \lambda_i \theta_p\right)\right)}{1 + \exp\left(\mu_i + \lambda_i \theta_p\right)\right)} \right)^{Y_{pi}} \left(1 - \left(\frac{\exp\left(\mu_i + \lambda_i \theta_p\right)\right)}{1 + \exp\left(\mu_i + \lambda_i \theta_p\right)\right)} \right) \right)^{(n_i - Y_{pi})}$$

Further, we can use the same priors as before on each of our item parameters

- μ_i: Normal prior $N(0, 1000)$
- λ_i Normal prior $N(0, 1000)$

Likewise, we can identify the scale of the latent variable as before, too:

- $\theta_p \sim N(0, 1)$

Estimating the Binomial Model in Stan

```
model {

  lambda ~ multi_normal(meanLambda, covLambda); // Prior for item discrimination/factor loadings
  mu ~ multi_normal(meanMu, covMu);             // Prior for item intercepts

  theta ~ normal(0, 1);                         // Prior for latent variable (with mean/sd specified)

  for (item in 1:nItems){
    Y[item] ~ binomial(maxItem[item], inv_logit(mu[item] + lambda[item]*theta));
  }

}
```

Here, the binomial function has two arguments:

- The first (`maxItem[item]`) is the number of "trials" n_i (here, our maximum score minus one)
- The second `inv_logit(mu[item] + lambda[item]*theta)` is the probability from our model (p_i)

The data `Y[item]` must be:

- Type: integer

- Range: 0 through `maxItem[item]`

Binomial Model `Parameters` Block

```
parameters {
  vector[nObs] theta;               // the latent variables (one for each person)
  vector[nItems] mu;                // the item intercepts (one for each item)
  vector[nItems] lambda;            // the factor loadings/item discriminations (one for each item)
}
```

No changes from any of our previous slope/intercept models

Binomial Model Data Block

```
data {
  int<lower=0> nObs;                              // number of observations
  int<lower=0> nItems;                            // number of items
  array[nItems] int<lower=0> maxItem;

  array[nItems, nObs] int<lower=0>  Y; // item responses in an array

  vector[nItems] meanMu;             // prior mean vector for intercept parameters
  matrix[nItems, nItems] covMu;      // prior covariance matrix for intercept parameters

  vector[nItems] meanLambda;         // prior mean vector for discrimination parameters
  matrix[nItems, nItems] covLambda;  // prior covariance matrix for discrimination parameters
}
```

Note:

- Need to supply `maxItem` (maximum score minus one for each item)
- The data are the same (integer) as in the binary/dichotomous items syntax

Preparing Data for Stan

```
# note: data must start at zero
conspiracyItemsBinomial = conspiracyItems
for (item in 1:ncol(conspiracyItemsBinomial)){
  conspiracyItemsBinomial[, item] = conspiracyItemsBinomial[, item] - 1
}

# check first item
table(conspiracyItemsBinomial[,1])
```

```
 0  1  2  3  4
49 51 47 23  7
```

```
# determine maximum value for each item
maxItem = apply(X = conspiracyItemsBinomial,
                MARGIN = 2,
                FUN = max)
```

Binomial Model Stan Call

```
# building standardized sum scores to initialize latent variable
sumScores = rowSums(conspiracyItems)
thetaInit = (sumScores - mean(sumScores))/sd(sumScores)

modelBinomial_samples = modelBinomial_stan$sample(
  data = modelBinomial_data,
  seed = 12112022,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 5000,
  iter_sampling = 5000,
  init = function() list(lambda = rnorm(n = nItems, mean = 10, sd = 2),
                         theta = rnorm(n = nObs, mean = thetaInit, sd = 0))
)
```

Binomial Model Results

```
# checking convergence
max(modelBinomial_samples$summary()$rhat, na.rm = TRUE)
```

```
[1] 1.001265
```

```
# item parameter results
print(modelBinomial_samples$summary(variables = c("mu", "lambda")) ,n=Inf)
```

```
# A tibble: 20 × 10
   variable     mean median    sd   mad     q5    q95  rhat ess_bulk ess_tail
   <chr>       <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>
 1 mu[1]      -0.840 -0.837 0.128 0.129 -1.05  -0.635  1.00    2409.    6027.
 2 mu[2]      -1.83  -1.82  0.216 0.212 -2.21  -1.49   1.00    2201.    4827.
 3 mu[3]      -1.99  -1.98  0.224 0.222 -2.38  -1.64   1.00    2193.    4891.
 4 mu[4]      -1.73  -1.72  0.210 0.210 -2.09  -1.40   1.00    2121.    4648.
 5 mu[5]      -2.00  -1.99  0.251 0.250 -2.43  -1.61   1.00    2102.    5043.
 6 mu[6]      -2.10  -2.09  0.246 0.243 -2.52  -1.71   1.00    2131.    5035.
 7 mu[7]      -2.44  -2.43  0.267 0.265 -2.90  -2.03   1.00    2471.    5576.
 8 mu[8]      -2.16  -2.15  0.246 0.242 -2.58  -1.78   1.00    2122.    4475.
 9 mu[9]      -2.40  -2.39  0.281 0.279 -2.89  -1.97   1.00    2307.    5761.
10 mu[10]     -3.97  -3.93  0.519 0.508 -4.89  -3.19   1.00    3188.    6378.
11 lambda[1]   1.11   1.11  0.147 0.147  0.884  1.37   1.00    4667.    7114.
12 lambda[2]   1.91   1.89  0.245 0.242  1.53   2.33   1.00    4071.    6885.
13 lambda[3]   1.92   1.91  0.257 0.252  1.53   2.37   1.00    4209.    4632.
14 lambda[4]   1.89   1.88  0.240 0.234  1.53   2.31   1.00    3599.    4405.
15 lambda[5]   2.28   2.26  0.285 0.277  1.85   2.78   1.00    3795.    5398.
16 lambda[6]   2.16   2.14  0.275 0.270  1.73   2.63   1.00    4022.    5340.
17 lambda[7]   2.14   2.12  0.299 0.294  1.68   2.66   1.00    3770.    5467.
18 lambda[8]   2.09   2.07  0.276 0.268  1.67   2.57   1.00    3952.    5307.
```

```
19 lambda[9]   2.36   2.34  0.319 0.319  1.87   2.90   1.00    3950.    5219.
20 lambda[10]  3.41   3.37  0.563 0.549  2.56   4.40   1.00    4459.    5240.
```

Option Characteristic Curves

ICC Plots

Investigating Latent Variable Estimates

Comparing Two Latent Variable Posterior Distributions

Comparing Latent Variable Posterior Mean and SDs

Comparing EAP Estimates with Sum Scores

Comparing Thetas: Binomial vs Normal

Categorical/Multinomial Distribution Models

Categorical/Multinomial Distribution Models

Although the binomial distribution is easy, it may not fit our data well

- Instead, we can use the categorical (sometimes called multinomial) distribution, with pmf (pdf):

$$P(Y = y) = \frac{n!}{y_1 ! \cdots y_C !} p_1^{y_1} \cdots p_C^{y_C}$$

Here:

- n: number of "trials"
- y_c : number of events in each of c categories ($c \in \{1, \ldots, C\}; \sum_c y_c = n$)
- p_c : probability of observing an event in category c

Adapting the Multinomial Distribution for Item Response Models

With some definitions, we can make a multinomial distribution into one we can use for polytomous item response models

- The number of "trials" is set to one for all items $n_i = 1$ $(\forall i)$
  - With $n_i = 1$, this is called the categorical distribution

$$P(Y_{pi} \mid \theta_p) = p_{i1}^{I(y_{pi}=1)} \cdots p_{iC_i}^{I(y_{pi}=C_i)}$$

- The number of categories is equal to the number of options on an item (C_i)
- The item response model is specified for the set of probabilities p_{ic}, with $\sum_c p_{ic} = 1$
  - Then, use a link function to model each item's set of p_{ic} as a function of the latent trait

Choices for Models for Probability of Each Category p_{ic}

The most-frequently used polytomous item response models all use the categorical distribution for observed data

- They differ in how the model function builds the conditional response probabilities
  - Graded response models: set of ordered intercepts (or thresholds) and a single loading
    - Called proportional odds models in categorical data analysis
  - Partial credit models: set of unordered difficulty parameters and a single loading
  - Nominal response models: set of unordered intercepts and set of loadings
    - Called generalized logit models in categorical data analysis
- Terminology note:
  - Terms graded response and partial credit come from educational measurement

- Data need not be graded response/partial credit to you

Graded Response Model

The graded response model is an ordered logistic regression model where:

$$P\left(Y_{ic} \mid \theta\right) = \begin{cases} 1 - P^*\left(Y_{i1} \mid \theta\right) & \text{if } c = 1 \\ P^*\left(Y_{ic-1} \mid \theta\right) - P^*\left(Y_{ic} \mid \theta\right) & \text{if } 1 < c < C_i \\ P^*\left(Y_{iC_i-1} \mid \theta\right) & \text{if } c = C_i \end{cases}$$

Where:

$$P^*\left(Y_{ic} \mid \theta\right) = \frac{\exp(\mu_{ic} + \lambda_i \theta_p)}{1 + \exp(\mu_{ic} + \lambda_i \theta_p)}$$

With:

- $C_i - 1$ Ordered intercepts: $\mu_1 > \mu_2 > \ldots > \mu_{C_i - 1}$

Estimating the Graded Response Model in Stan

```
model {

  lambda ~ multi_normal(meanLambda, covLambda); // Prior for item discrimination/factor loadings
  theta ~ normal(0, 1);                         // Prior for latent variable (with mean/sd specified)

  for (item in 1:nItems){
    thr[item] ~ multi_normal(meanThr[item], covThr[item]);            // Prior for item intercepts
    Y[item] ~ ordered_logistic(lambda[item]*theta, thr[item]);
  }

}
```

Notes:

- `ordered_logistic` is a built in Stan function that makes the model easy to implement
  - Instead of intercepts, however, it uses thresholds: $*{ic}* = -{ic} $
  - First argument is the linear predictor (the non-intercept portion of the model)
  - Second argument is the set of thresholds for the item
- The function expects the responses of Y to start at one and go to maxCategory (

Y_{pi} {1, , C_i})

- No transformation needed to data (unless some categories have zero observations)

Graded Response Model `parameters` Block

```
parameters {
  vector[nObs] theta;                // the latent variables (one for each person)
  array[nItems] ordered[maxCategory-1] thr; // the item thresholds (one for each item category minus
  vector[nItems] lambda;             // the factor loadings/item discriminations (one for each item)
}
```

Notes:

- Threshold parameters: `array[nItems] ordered[maxCategory-1] thr;`
  - Is an array (each item has maxCategory-1) parameters
  - Is of type `ordered`: Automatically ensures order is maintained

    $\tau_{i1} < \tau_{i2} < \ldots < \tau_{iC-1}$

Graded Response Model `generated quantities` Block

```
generated quantities{
  array[nItems] vector[maxCategory-1] mu;
  for (item in 1:nItems){
    mu[item] = -1*thr[item];
  }
}
```

We use generated quantities to convert threshold parameters into intercepts

Graded Response Model `data` block

```
data {
  int<lower=0> nObs;                              // number of observations
  int<lower=0> nItems;                            // number of items
  int<lower=0> maxCategory;

  array[nItems, nObs] int<lower=1, upper=5>  Y; // item responses in an array

  array[nItems] vector[maxCategory-1] meanThr;   // prior mean vector for intercept parameters
  array[nItems] matrix[maxCategory-1, maxCategory-1] covThr;      // prior covariance matrix for inte

  vector[nItems] meanLambda;         // prior mean vector for discrimination parameters
  matrix[nItems, nItems] covLambda;  // prior covariance matrix for discrimination parameters
}
```

Notes:

- The input for the prior mean/covariance matrix for threshold parameters is now an array (one mean vector and covariance matrix per item)

Graded Response Model Data Preparation

To match the array for input for the threshold hyperparameter matrices, a little data manipulation is needed

```
# item threshold hyperparameters
thrMeanHyperParameter = 0
thrMeanVecHP = rep(thrMeanHyperParameter, maxCategory-1)
thrMeanMatrix = NULL
for (item in 1:nItems){
  thrMeanMatrix = rbind(thrMeanMatrix, thrMeanVecHP)
}

thrVarianceHyperParameter = 1000
thrCovarianceMatrixHP = diag(x = thrVarianceHyperParameter, nrow = maxCategory-1)
thrCovArray = array(data = 0, dim = c(nItems, maxCategory-1, maxCategory-1))
for (item in 1:nItems){
  thrCovArray[item, , ] = diag(x = thrVarianceHyperParameter, nrow = maxCategory-1)
}
```

- The R array matches stan's array type

Graded Response Model Stan Call

```
# building standardized sum scores to initialize latent variable
sumScores = rowSums(conspiracyItems)
thetaInit = (sumScores - mean(sumScores))/sd(sumScores)

modelOrderedLogit_samples = modelOrderedLogit_stan$sample(
  data = modelOrderedLogit_data,
  seed = 121120221,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 5000,
  iter_sampling = 5000,
  init = function() list(lambda = rnorm(n = nItems, mean = 10, sd = 2),
                         theta = rnorm(n = nObs, mean = thetaInit, sd = 0))
)
```

Note: Using positive starting values for the λ parameters

Graded Response Model Results

```
# checking convergence
max(modelOrderedLogit_samples$summary()$rhat, na.rm = TRUE)
```

```
[1] 1.003616
```

```
# item parameter results
print(modelOrderedLogit_samples$summary(variables = c("lambda", "mu")) ,n=Inf)
```

```
# A tibble: 50 × 10
   variable       mean   median    sd   mad      q5     q95  rhat ess_bulk
   <chr>         <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>    <dbl>
 1 lambda[1]    2.13     2.12   0.290 0.287   1.68   2.63    1.00    4987.
 2 lambda[2]    3.07     3.05   0.438 0.434   2.40   3.83    1.00    5015.
 3 lambda[3]    2.59     2.56   0.392 0.383   2.00   3.28    1.00    5478.
 4 lambda[4]    3.11     3.08   0.432 0.420   2.45   3.86    1.00    4763.
 5 lambda[5]    5.01     4.95   0.785 0.756   3.84   6.41    1.00    4730.
 6 lambda[6]    5.04     4.98   0.771 0.745   3.88   6.39    1.00    4061.
 7 lambda[7]    3.24     3.21   0.495 0.495   2.49   4.11    1.00    5097.
 8 lambda[8]    5.34     5.26   0.873 0.825   4.06   6.92    1.00    4450.
 9 lambda[9]    3.16     3.12   0.490 0.488   2.42   4.02    1.00    5360.
10 lambda[10]   2.66     2.62   0.483 0.461   1.94   3.51    1.00    5942.
11 mu[1,1]      1.49     1.48   0.283 0.282   1.04   1.97    1.00    2680.
12 mu[2,1]      0.171    0.173  0.326 0.321  -0.367  0.706   1.00    1676.
13 mu[3,1]     -0.454   -0.446  0.302 0.298  -0.963  0.0257  1.00    1995.
14 mu[4,1]      0.345    0.343  0.332 0.328  -0.201  0.889   1.00    1692.
15 mu[5,1]      0.325    0.323  0.484 0.473  -0.473  1.13    1.00    1401.
16 mu[6,1]     -0.0421  -0.0301 0.491 0.481  -0.862  0.739   1.00    1369.
17 mu[7,1]     -0.818   -0.809  0.357 0.353  -1.42  -0.247   1.00    1739.
18 mu[8,1]     -0.392   -0.377  0.523 0.508  -1.28   0.434   1.00    1411.
```

```
19 mu[9,1]     -0.748   -0.738  0.350 0.346  -1.34  -0.191   1.00    1764.
20 mu[10,1]    -1.98    -1.95   0.392 0.380  -2.66  -1.38    1.00    2800.
21 mu[1,2]     -0.655   -0.651  0.254 0.254  -1.08  -0.242   1.00    2198.
22 mu[2,2]     -2.22    -2.20   0.392 0.382  -2.88  -1.61    1.00    2334.
23 mu[3,2]     -1.67    -1.66   0.331 0.325  -2.24  -1.16    1.00    2206.
```

Graded Response Model Option Characteristic Curves

Graded Response Model Item Characteristic Curves

Investigating Latent Variables

Comparing Two Latent Variable Posterior Distributions

Comparing Latent Variable EAP Estimates with Posterior SDs

Comparing Latent Variable EAP Estimates with Sum Scores

Comparing Latent Variable Posterior Means: GRM vs CFA

Comparing Latent Variable Posterior SDs: GRM vs Normal

Which Posterior SD is Larger: GRM vs. CFA

Comparing Thetas: GRM vs Binomial

Comparing Latent Variable Posterior SDs: GRM vs Binomial

Which Posterior SD is Larger: GRM vs. Binomial

Nominal Response Models (Generalized Logit Models)

Adapting the Multinomial Distribution for Item Response Models

With some definitions, we can make a multinomial distribution into one we can use for polytomous item response models

- The number of "trials" is set to one for all items $n_i = 1 \ (\forall i)$
  - With $n_i = 1$, this is called the categorical distribution

$$P(Y_{pi} \mid \theta_p) = p_{i1}^{I(y_{pi}=1)} \cdots p_{iC_i}^{I(y_{pi}=C_i)}$$

- The number of categories is equal to the number of options on an item (C_i)
- The item response model is specified for the set of probabilities p_{ic}, with $\sum_c p_{ic} = 1$
  - Then, use a link function to model each item's set of p_{ic} as a function of the latent trait

Nominal Response Model (Generalized Logit Model)

The nominal response model is an ordered logistic regression model where:

$$P\left(Y_{ic} \mid \theta\right) = \frac{\exp(\mu_{ic} + \lambda_{ic}\theta_p)}{\sum_{c=1}^{C_i} \exp(\mu_{ic} + \lambda_{ic}\theta_p)}$$

Where:

- One constraint per parameter (one of these options):
  - Sum to zero: $\sum_{c=1}^{C_i} \mu_{ic} = 0$ and $\sum_{c=1}^{C_i} \lambda_{ic} = 0$
  - Fix one category's parameters to zero: $\mu_{iC_i} = 0$ and $\lambda_{iC_i} = 0$

Estimating the NRM in Stan

```
model {

  vector[maxCategory] probVec;

  theta ~ normal(0,1);

  for (item in 1:nItems){
    initMu[item] ~ multi_normal(meanMu[item], covMu[item]);  // Prior for item intercepts
    initLambda[item] ~ multi_normal(meanLambda[item], covLambda[item]);  // Prior for item loadings

    for (obs in 1:nObs) {
      for (category in 1:maxCategory){
        probVec[category] = mu[item, category] + lambda[item, category]*theta[obs];
      }
      Y[item, obs] ~ categorical_logit(probVec);
    }
  }
}
```

- Probability vector is built category-by-category
  - Code is not vectorized (takes longer to run)
- `categorical_logit` model takes input, applies inverse logit function, evaluates categorical distribution pmf
- Set-to-zero constraints used:

- `initMu` and `initLambda` have only estimated values in them
- Will build `mu` and `lambda` in transformed parameters block

Nominal Response Model `parameters` Block

```
  parameters {
    array[nItems] vector[maxCategory - 1] initMu;
    array[nItems] vector[maxCategory - 1] initLambda;
    vector[nObs] theta;                   // the latent variables (one for each person)
  }
```

Set-to-zero constraints used:

- `initMu` and `initLambda` have only estimated values in them
- Will build `mu` and `lambda` in transformed parameters block

Nominal Response Model `transformed parameters` Block

```
transformed parameters {
  array[nItems] vector[maxCategory] mu;
  array[nItems] vector[maxCategory] lambda;

  for (item in 1:nItems){
    mu[item, 2:maxCategory] = initMu[item, 1:(maxCategory-1)];
    mu[item, 1] = 0.0; // setting one category's intercept to zero

    lambda[item, 2:maxCategory] = initLambda[item, 1:(maxCategory-1)];
    lambda[item, 1] = 0.0; // setting one category's lambda to zero

  }
}
```

Notes:

- Here, we set the first category's μ_{i1} and λ_{i1} to zero
- We use `mu` and `lambda` for the model itself (not `initMu` or `initLambda`)

Nominal Response Model `data` Block

```
data {
  int maxCategory;
  int nObs;
  int nItems;

  array[nItems, nObs] int Y;

  array[nItems] vector[maxCategory-1] meanMu;   // prior mean vector for intercept parameters
  array[nItems] matrix[maxCategory-1, maxCategory-1] covMu;      // prior covariance matrix for inter

  array[nItems] vector[maxCategory-1] meanLambda;       // prior mean vector for discrimination param
  array[nItems] matrix[maxCategory-1, maxCategory-1] covLambda;  // prior covariance matrix for discr

}
```

Almost the same as graded response model

- Lambda now needs an array

Nominal Response Model Data Preparation

```
# Data needs: successive integers from 1 to highest number (recode if not consistent)
maxCategory = 5

# data dimensions
nObs = nrow(conspiracyItems)
nItems = ncol(conspiracyItems)

# item threshold hyperparameters
muMeanHyperParameter = 0
muMeanVecHP = rep(muMeanHyperParameter, maxCategory-1)
muMeanMatrix = NULL
for (item in 1:nItems){
  muMeanMatrix = rbind(muMeanMatrix, muMeanVecHP)
}

muVarianceHyperParameter = 1000
muCovarianceMatrixHP = diag(x = muVarianceHyperParameter, nrow = maxCategory-1)
muCovArray = array(data = 0, dim = c(nItems, maxCategory-1, maxCategory-1))
for (item in 1:nItems){
  muCovArray[item, , ] = diag(x = muVarianceHyperParameter, nrow = maxCategory-1)
}

# item discrimination/factor loading hyperparameters
lambdaMeanHyperParameter = 0
lambdaMeanVecHP = rep(lambdaMeanHyperParameter, maxCategory-1)
lambdaMeanMatrix = NULL
for (item in 1:nItems){
  lambdaMeanMatrix = rbind(lambdaMeanMatrix, lambdaMeanVecHP)
}

lambdaVarianceHyperParameter = 1000
```

Nominal Response Model Stan Call

```
modelCategoricalLogit_samples = modelCategoricalLogit_stan$sample(
  data = modelOrderedLogit_data,
  seed = 121120222,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 1000,
  iter_sampling = 1000,
  init = function() list(theta=rnorm(nObs, mean=thetaInit, sd=0))
)
```

Nominal Response Model Results

```
# checking convergence
max(modelCategoricalLogit_samples$summary()$rhat, na.rm = TRUE)
```

```
[1] 1.005214
```

```
print(modelCategoricalLogit_samples$summary(variables = c("mu", "lambda")), n=Inf)
```

```
# A tibble: 100 × 10
   variable       mean  median    sd   mad      q5     q95   rhat ess_bulk
   <chr>         <dbl>   <dbl> <dbl> <dbl>   <dbl>   <dbl>  <dbl>    <dbl>
 1 mu[1,1]       0       0     0     0       0       0      NA          NA
 2 mu[2,1]       0       0     0     0       0       0      NA          NA
 3 mu[3,1]       0       0     0     0       0       0      NA          NA
 4 mu[4,1]       0       0     0     0       0       0      NA          NA
 5 mu[5,1]       0       0     0     0       0       0      NA          NA
 6 mu[6,1]       0       0     0     0       0       0      NA          NA
 7 mu[7,1]       0       0     0     0       0       0      NA          NA
 8 mu[8,1]       0       0     0     0       0       0      NA          NA
 9 mu[9,1]       0       0     0     0       0       0      NA          NA
10 mu[10,1]      0       0     0     0       0       0      NA          NA
11 mu[1,2]       1.10    1.08  0.296 0.290   0.626   1.61    1.00     1525.
12 mu[2,2]       0.390   0.387 0.254 0.252  -0.0174  0.818   1.00     2127.
13 mu[3,2]      -0.481  -0.484 0.274 0.270  -0.933  -0.0282  1.00     2144.
14 mu[4,2]       0.574   0.569 0.283 0.282   0.116   1.04    1.00     1992.
15 mu[5,2]       0.661   0.666 0.266 0.266   0.228   1.10    1.00     1731.
16 mu[6,2]       0.653   0.648 0.282 0.278   0.186   1.12    1.00     2017.
17 mu[7,2]      -0.302  -0.303 0.236 0.233  -0.692   0.0885  1.00     2130.
18 mu[8,2]       0.337   0.333 0.273 0.268  -0.113   0.795   1.00     1701.
19 mu[9,2]      -0.325  -0.331 0.239 0.237  -0.714   0.0774  1.00     2344.
```

```
20 mu[10,2]     -1.45   -1.44   0.263 0.263 -1.89   -1.04    1.00     2945.
21 mu[1,3]       0.970   0.966  0.306 0.306  0.469   1.48    1.00     1539.
22 mu[2,3]      -0.689  -0.686  0.374 0.363 -1.32   -0.0943  1.00     1674.
23 mu[3,3]      -0.197  -0.199  0.262 0.261 -0.631   0.231   0.999    2002.
```

Comparing EAP Estimates with Posterior SDs

Comparing EAP Estimates with Sum Scores

Comparing Thetas: NRM vs Normal:

Comparing Theta SDs: NRM vs Normal:

Which SDs are bigger: NRM vs. Normal?

Comparing Thetas: NRM vs GRM:

Comparing Theta SDs: NRM vs GRM:

Which SDs are bigger: NRM vs GRM?

Models With Different Types of Items

Models with Different Types of Items

Although often taught as one type of model that applies to all items, you can mix-and-match distributions

- Recall the posterior distribution of the latent variable θ_p
- For each person, the model (data) likelihood function can be constructed so that each item's conditional PDF is used:

$$f\left(\mathbf{Y}_p \mid \theta_p\right) = \prod_{i=1}^{I} f\left(Y_{pi} \mid \theta_p\right)$$

Here, $\prod_{i=1}^{I} f\left(Y_{pi} \mid \theta_p\right)$ can be any distribution that you can build

Wrapping Up

Wrapping Up

- There are many different models for polytomous data
  - Almost all use the categorical (multinomial with one trial) distribution
- What we say today is that the posterior SDs of the latent variables are larger with categorical models
  - Much more uncertainty compared to normal models
- What we will need is model fit information to determine what fits best

https://jonathantemplin.com/bayesian-psychometric-modeling-fall-2022/