Missing Data

Lecture 4g

https://jonathantemplin.com/bayesian-psychometric-modeling-fall-2024/

Today's Lecture Objectives

1. Show how to estimate models where data are missing

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.

Missing Data in Stan

Missing Data in Stan

If you ever attempted to analyze missing data in Stan, you likely received an error message:

```
Error: Variable 'Y' has NA values.
```

That is because, as a default, Stan does not model missing data

- Instead, we have to get Stan to work with the data we have (the values that are not missing)
  - That does not mean remove cases where any observed variables are missing

Example Missing Data

To make things a bit easier, I'm only turning one value into missing data (the first person's response to the first item)

```
# Import data ==========================================================================

conspiracyData = read.csv("conspiracies.csv")
conspiracyItems = conspiracyData[,1:10]

# make some cases missing for demonstration:
conspiracyItems[1,1] = NA
```

Note: All code will work with as much missing as you have

- Observed variables do have to have some values that are not missing (by definition!)

Stan Syntax: Multidimensional Model

We will use the syntax from the last lecture–for multidimensional measurement models with ordinal logit (graded response) items

The Q-matrix this time, will be a single column vector (one dimension)

```
      [,1]
 [1,]    1
 [2,]    1
 [3,]    1
 [4,]    1
 [5,]    1
 [6,]    1
 [7,]    1
 [8,]    1
 [9,]    1
[10,]    1
```

Stan Model Block

```
model {

  lambda ~ multi_normal(meanLambda, covLambda);
  thetaCorrL ~ lkj_corr_cholesky(1.0);
  theta ~ multi_normal_cholesky(meanTheta, thetaCorrL);


  for (item in 1:nItems){
    thr[item] ~ multi_normal(meanThr[item], covThr[item]);
    Y[item, observed[item, 1:nObserved[item]]] ~ ordered_logistic(thetaMatrix[observed[item, 1:nObserved[item]],]*lambdaMa
  }


}
```

Notes:

- Big change is in `Y`:
  - Previous: `Y[item]`
  - Now: `Y[item, observed[item, 1:nObserved[item]]]`
    - The part after the comma is a list of who provided responses to the item (input in the data block)
- Mirroring this is a change to `thetaMatrix[observed[item, 1:nObserved[item]],]`
  - Keeps only the latent variables for the persons who provided responses

Stan Data Block

```
data {

  // data specifications  ============================================================
  int<lower=0> nObs;                          // number of observations
  int<lower=0> nItems;                        // number of items
  int<lower=0> maxCategory;       // number of categories for each item
  array[nItems] int nObserved;
  array[nItems, nObs] array[nItems] int observed;

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

  // loading specifications  ============================================================
  int<lower=1> nFactors;                                  // number of loadings in the model
  array[nItems, nFactors] int<lower=0, upper=1> Qmatrix;

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

  vector[nFactors] meanTheta;
}
```

Data Block Notes

- Two new arrays added:
  - `array[nItems] int nObserved;`: The number of observations with non-missing data for each item
  - `array[nItems, nObs] array[nItems] int observed;`: A listing of which observations have non-missing data for each item
    - Here, the size of the array is equal to the size of the data matrix
    - If there were no missing data at all, the listing of observations with non-missing data would equal this size
- Stan uses these arrays to only model data that are not missing
  - The values of observed serve to select only cases in Y that are not missing

Building Non-Missing Data Arrays

To build these arrays, we can use a loop in R:

```
observed = matrix(data = -1, nrow = nrow(conspiracyItems), ncol = ncol(conspiracyItems))
nObserved = NULL
for (variable in 1:ncol(conspiracyItems)){
  nObserved = c(nObserved, length(which(!is.na(conspiracyItems[, variable]))))
  observed[1:nObserved[variable], variable] = which(!is.na(conspiracyItems[, variable]))
}
```

For the item that has the first case missing, this gives:

```
nObserved[1]
```

```
[1] 176
```

```
observed[, 1]
```

```
  [1]   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
 [19]  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37
 [37]  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55
 [55]  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73
 [73]  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91
 [91]  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109
[109] 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
[127] 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
[145] 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163
[163] 164 165 166 167 168 169 170 171 172 173 174 175 176 177  -1
```

The item has 176 observed responses and one missing

Array Indexing

We can use the values of `observed[,1]` to have Stan only select the corresponding data points that are non-missing

To demonstrate, in R, here is all of the data for the first item

```
conspiracyItems[,1]
```

```
  [1] NA  3  4  2  2  1  4  2  3  2  1  3  2  1  2  2  2  2  3  2  1  1  1  1  4
 [26]  4  4  3  4  3  3  1  2  1  2  3  1  2  1  2  1  1  2  2  4  3  3  1  1  4
 [51]  1  2  1  3  1  1  2  1  4  2  2  2  1  5  3  2  3  3  1  3  2  1  2  1  1
 [76]  2  3  4  3  3  2  2  1  3  3  1  2  3  1  4  2  1  2  5  5  2  3  1  3  2
[101]  3  5  2  4  1  3  3  4  3  2  2  4  3  3  4  3  2  2  1  1  3  4  3  2  1
[126]  4  3  2  2  3  4  5  1  5  3  1  3  3  3  2  2  1  1  3  1  3  3  4  1  1
[151]  4  1  4  2  1  1  1  2  2  1  4  2  1  2  4  1  2  5  3  2  1  3  3  3  2
[176]  3  3
```

And here, we select the non-missing using the index values in `observed`:

```
conspiracyItems[observed[1:nObserved, 1], 1]
```

```
  [1] 3 4 2 2 1 4 2 3 2 1 3 2 1 2 2 2 2 3 2 1 1 1 1 4 4 4 3 4 3 3 1 2 1 2 3 1 2
 [38] 1 2 1 1 2 2 4 3 3 1 1 4 1 2 1 3 1 1 2 1 4 2 2 2 1 5 3 2 3 3 1 3 2 1 2 1 1
 [75] 2 3 4 3 3 2 2 1 3 3 1 2 3 1 4 2 1 2 5 5 2 3 1 3 2 3 5 2 4 1 3 3 4 3 2 2 4
[112] 3 3 4 3 2 2 1 1 3 4 3 2 1 4 3 2 2 3 4 5 1 5 3 1 3 3 3 2 2 1 1 3 1 3 3 4 1
[149] 1 4 1 4 2 1 1 1 2 2 1 4 2 1 2 4 1 2 5 3 2 1 3 3 3 2 3 3
```

The values of `observed[1:nObserved, 1]` lead to only using the non-missing data

Change Missing NA to Nonsense Values

Finally, we must ensure all data into Stan have no NA values

- My recommendation: Change all NA values to something that cannot be modeled
  - I am picking -1 here: It cannot be used with the ordered_logit() likelihood
- This ensures that Stan won't model the data by accident
  - But, we must remember this if we are using the data in other steps (like PPMC)

```
# Fill in NA values in Y
Y = conspiracyItems
for (variable in 1:ncol(conspiracyItems)){
  Y[which(is.na(Y[,variable])),variable] = -1
}
```

15

Running Stan

With our missing values denoted, we then run Stan as we have previously

```
modelOrderedLogit_data = list(
  nObs = nObs,
  nItems = nItems,
  maxCategory = maxCategory,
  nObserved = nObserved,
  observed = t(observed),
  Y = t(Y),
  nFactors = ncol(Qmatrix),
  Qmatrix = Qmatrix,
  meanThr = thrMeanMatrix,
  covThr = thrCovArray,
  meanLambda = lambdaMeanVecHP,
  covLambda = lambdaCovarianceMatrixHP,
  meanTheta = thetaMean
)


modelOrderedLogit_samples = modelOrderedLogit_stan$sample(
  data = modelOrderedLogit_data,
  seed = 191120221,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 2000,
  iter_sampling = 2000,
  init = function() list(lambda=rnorm(nItems, mean=5, sd=1))
)
```

Stan Results

```
[1] 1.004752
```

```
# A tibble: 50 × 10
   variable      mean   median    sd   mad       q5    q95  rhat ess_bulk ess_tail
   <chr>        <dbl>    <dbl> <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>    <dbl>
 1 lambda[1]   2.02     2.00   0.269 0.262   1.60     2.49   1.00    2413.    4538.
 2 lambda[2]   2.88     2.85   0.390 0.389   2.27     3.55   1.00    1666.    3991.
 3 lambda[3]   2.40     2.38   0.344 0.342   1.87     3.00   1.00    2683.    4438.
 4 lambda[4]   2.91     2.89   0.390 0.382   2.32     3.59   1.00    1781.    3763.
 5 lambda[5]   4.32     4.28   0.602 0.595   3.40     5.38   1.00    1843.    3279.
 6 lambda[6]   4.19     4.17   0.562 0.561   3.32     5.16   1.00    2002.    3757.
 7 lambda[7]   2.90     2.87   0.425 0.413   2.25     3.66   1.00    2476.    3430.
 8 lambda[8]   4.16     4.12   0.562 0.548   3.31     5.15   1.00    1545.    3167.
 9 lambda[9]   2.90     2.88   0.424 0.422   2.26     3.64   1.00    2401.    3674.
10 lambda[1…   2.43     2.40   0.415 0.409   1.80     3.15   1.00    2844.    4579.
11 mu[1,1]     1.87     1.86   0.295 0.289   1.41     2.39   1.00    1110.    3809.
12 mu[2,1]     0.817    0.807  0.304 0.301   0.340    1.34   1.00     648.    1974.
13 mu[3,1]     0.102    0.101  0.264 0.259  -0.328    0.548  1.00     742.    1829.
14 mu[4,1]     0.992    0.978  0.317 0.315   0.486    1.52   1.00     610.    1473.
15 mu[5,1]     1.30     1.29   0.424 0.419   0.635    2.01   1.00     554.    1716.
16 mu[6,1]     0.954    0.948  0.403 0.401   0.309    1.63   1.00     533.    1327.
17 mu[7,1]    -0.105   -0.108  0.298 0.296  -0.590    0.389  1.00     638.    1678.
18 mu[8,1]     0.691    0.679  0.399 0.391   0.0550   1.37   1.00     496.    1106.
19 mu[9,1]    -0.0571  -0.0552 0.300 0.291  -0.540    0.429  1.00     634.    1829.
20 mu[10,1]   -1.35    -1.34   0.308 0.303  -1.87    -0.862  1.00     857.    2389.
21 mu[1,2]    -0.190   -0.190  0.238 0.238  -0.576    0.205  1.00     834.    1955.
22 mu[2,2]    -1.53    -1.52   0.324 0.320  -2.08    -1.02   1.00     815.    2413.
```

Likelihoods with Missing Data

Likelihoods with Missing Data

The way we have coded Stan enables estimation by effectively skipping over cases that were missing

- This means our likelihood functions are slightly different

For the parameters of an item i, the previous model/data likelihood was:

Now, we must alter the PDF so that missing data do not contribute:

This also applies to the likelihood for a person's as any missing items are skipped:

Ramifications of Skipping Missing Data

It may seem somewhat basic to simply skip over missing cases (also called list-wise deletion), but:

- Such methods meet the assumptions of missing at random data
  - Missing data are related to some of the observed data

It is a stronger method for analysis than case-wise deletion (removing a person fully)

- Case-wise deletion assumes the data are missing completely at random
  - No relation to any observed data
  - Less likely to hold in missing data than MAR

Moreover, the methods we implemented in Stan are equivalent to those implemented in maximum likelihood algorithms

- The likelihood function is the same

Wrapping Up

Wrapping Up

Today, we showed how to skip over missing data in Stan

- Slight modifications needed to syntax
- Assumes missing at random

Of note, we could (but didn't) also build models for missing data in Stan

- Using the transformed parameters block

Finally, Stan's missing data methods are quite different from JAGS

- JAGS imputes any missing data at each step of a Markov chain using Gibbs sampling