# Latent Variable Scale Identification Methods

Lecture 4i

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

# Today’s Lecture Objectives

1. Show how to estimate the standard deviation of the latent variable

# 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.

# Model Setup Today

Today, we will revert back to the graded response model assumptions to discuss how to estimate the latent variable standard deviation

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

Where:

$$P\left(Y_{ic} > c \mid \theta\right) = \frac{\exp(-\tau_{ic} + \lambda_i \theta_p)}{1 + \exp(-\tau_{ic} + \lambda_i \theta_p)}$$

With:

- $C_i - 1$ Ordered thresholds: $\tau_1 < \tau_2 < \ldots < \tau_{C_i-1}$

We can convert thresholds to intercepts by multiplying by negative one: $\mu_c = -\tau_c$

# Scale Identification Methods

# Identification of Latent Traits, Part 1

Psychometric models require two types of identification to be valid:

1. Empirical Identification

- The minimum number of items that must measure each latent variable
- From CFA: three observed variables for each latent variable (or two if the latent variable is correlated with another latent variable)

Bayesian priors can help to make models with fewer items than these criteria suggest estimable

- The parameter estimates (item parameters and latent variable estimates) often have MCMC convergence issues and should not be trusted
- Use the CFA standard in your work

# Identification of Latent Traits, Part 2

Psychometric models require two types of identification to be valid:

2. Scale Identification (i.e., what the mean/variance is for each latent variable)

- The additional set of constraints needed to set the mean and standard deviation (variance) of the latent variables
- Two main methods to set the scale:
    - Marker item parameters
        - For variances: Set the loading/slope to one for one observed variable per latent variable
            - Can estimate the latent variable's variance (the diagonal of $\mathbf{\Sigma}_\theta$)
        - For means: Set the item intercept to one for one observed variable perlatent variable
            - Can estimate the latent variable's mean (in $\boldsymbol{\mu}_\theta$)
    - Standardized factors
        - Set the variance for all latent variables to one
        - Set the mean for all latent variables to zero
        - Estimate all unique off-diagonal correlations (covariances) in $\mathbf{\Sigma}_\theta$

# Marker Items for $\theta$ Standard Deviations

To estimate the standard deviation of $\theta$ (a type of empirical prior) * Set one loading/discrimination parameter to one

To estimate the mean of $\theta$:

- Set one threshold parameter to zero
- A bit more difficult to implement in Stan
- Skipped for today

Under both of these cases, the model/data likelihood is identified

- This provides what I call “strong identification” of the posterior distribution

I begin with a single $\theta$ as it is easier to show

- Multidimensional $\Theta$ comes after

# Stan's Model Block

```
model {

  initLambda ~ multi_normal(meanLambda, covLambda); // Prior for estimated item discrimination/factor loadings
  thetaSD ~ lognormal(thetaSDmean,thetaSDsd);               // Prior for theta standard deviation
  theta ~ normal(0, thetaSD);                       // Prior for latent variable (with sd specified)

  for (item in 1:nItems){
    thr[item] ~ multi_normal(meanThr[item], covThr[item]);             // Prior for item thresholds
    Y[item] ~ ordered_logistic(lambda[item]*theta, thr[item]);         // Item repsonse model (model/data likelihood)
  }


}
```

Notes:

- Here, we are only estimating the standard deviation of $\theta$
  - We will leave the mean at zero
  - We will use a log normal distribution for the SD (needs a mean and SD for hyperparameters)
- lambda`` inordered_logistic()function is different frominitLambda```
  - We need a transformed parameters block to set one loading to one

# Stan’s Parameters Block

```
parameters {
  vector[nObs] theta;                // the latent variables (one for each person)
  real<lower=0> thetaSD;

  array[nItems] ordered[maxCategory-1] thr; // the item thresholds (one for each item category minus one)
  vector[nItems-1] initLambda;            // the estimated factor loadings (number of items-1 for one marker item)
}
```

Notes:

- `thetaSD` has lower bound of zero
- `initLambda` is length `nItems-1` (one less as we set that one to one)

# Stan’s Transformed Parameters Block

```
transformed parameters{
  vector[nItems] lambda;   // the loadings that go into the model itself

  lambda[1] = 1.0;         // first loading on the factor is set to one for identification (marker item)
  lambda[2:(nItems)] = initLambda[1:(nItems-1)]; // rest of loadings are set to estimated values in initLambda
}
```

Notes:

- We set the first loading to one
  - All others are estimated
- Technically, any loading can be set to one
  - All are equivalent models based on likelihood
- If an item has very little relation to the latent variable, setting that item’s loading to one can cause estimation problems
  - Difficult to tell which item may have problems before the analysis

# Stan’s Data Block

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

  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 intercept parameters

  vector[nItems-1] meanLambda;         // prior mean vector for discrimination parameters
  matrix[nItems-1, nItems-1] covLambda;  // prior covariance matrix for discrimination parameters

  real thetaSDmean;  // prior mean hyperparameter for theta standard deviation (log normal distribution)
  real thetaSDsd; // prior sd hyperparameter for theta standard deviation (log normal distribution)
}
```

Notes:

- Here, we need to set mean/sd hyperparameters for the standard deviation of $\theta$

# R Data List

```r
modelGRM_markerItem_data = list(
  nObs = nObs,
  nItems = nItems,
  maxCategory = maxCategory,
  Y = t(conspiracyItems),
  meanThr = thrMeanMatrix,
  covThr = thrCovArray,
  meanLambda = lambdaMeanVecHP,
  covLambda = lambdaCovarianceMatrixHP,
  thetaSDmean = 0,
  thetaSDsd = 2
)
```

Notes:

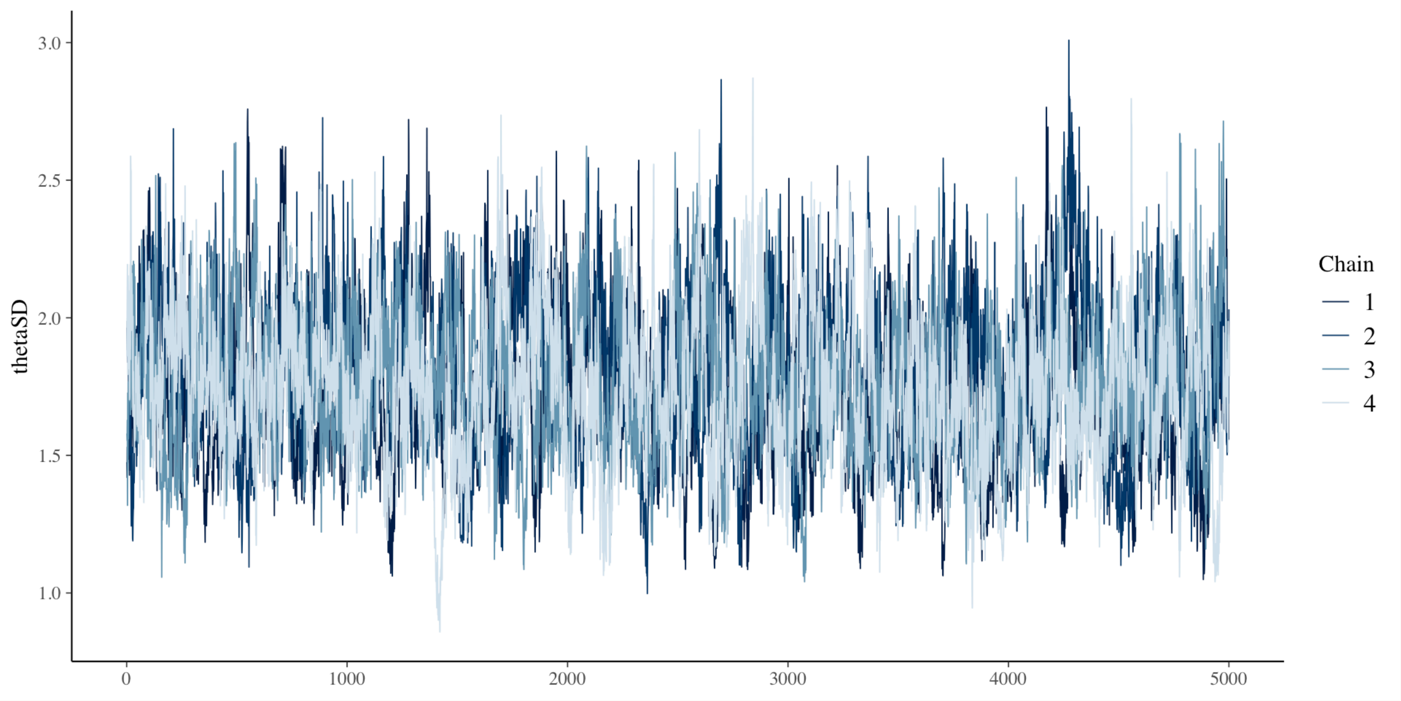
- Hyperparameter for location ($\mu$) of lognormal distribution is 0
- Hyperparameter for scale ($\sigma$) of lognormal distribution is 2
- Lognormal mean is $\exp\left(\mu + \frac{\sigma^2}{2}\right) = 7.39$
- Lognormal SD is $\sqrt{\left[\exp\left(\sigma^2\right) - 1\right] \exp\left(2\mu + \sigma^2\right)} = 54.1$

# Stan Results

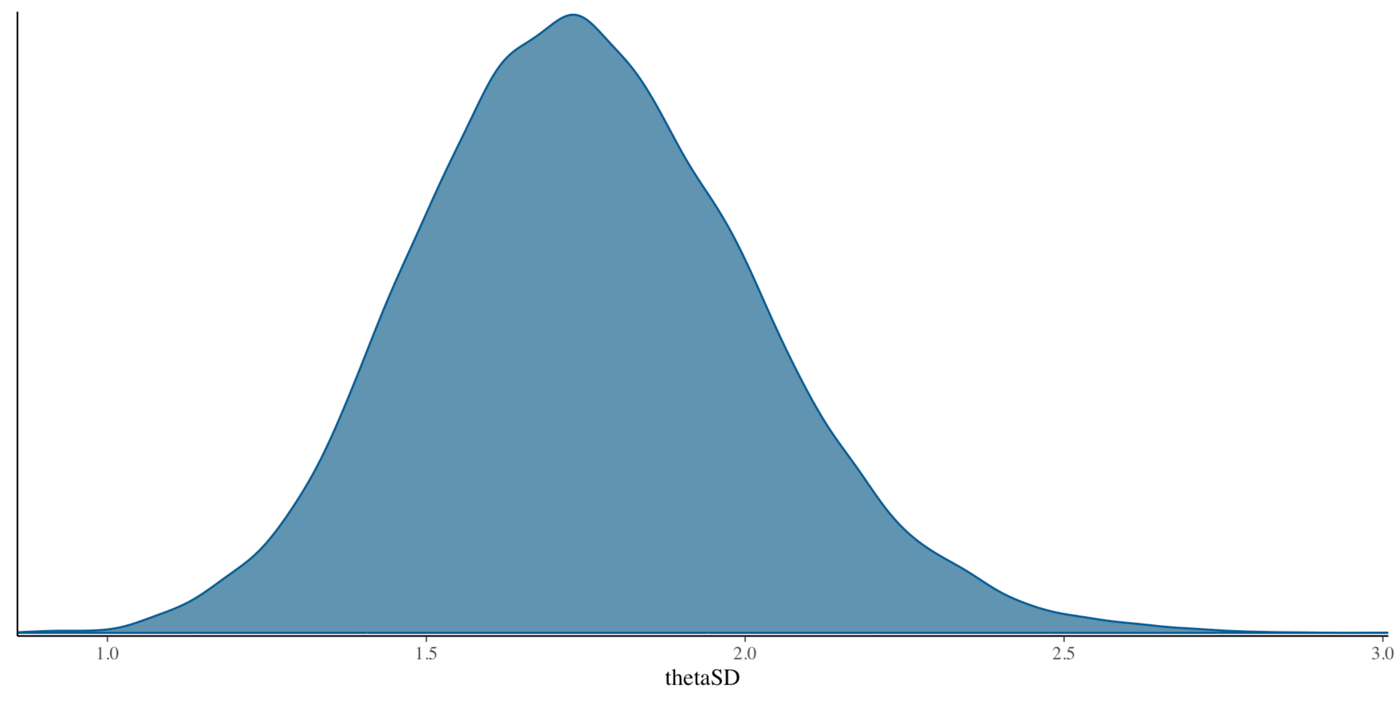
```
[1] 1.016827

# A tibble: 51 × 10
   variable        mean   median    sd   mad      q5     q95  rhat ess_bulk
   <chr>          <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>    <dbl>
 1 thetaSD      1.75      1.74   0.266 0.264   1.34    2.21   1.02     573.
 2 lambda[1]    1         1      0     0       1       1     NA         NA
 3 lambda[2]    1.69      1.66   0.307 0.296   1.25    2.25   1.01     874.
 4 lambda[3]    1.46      1.43   0.294 0.281   1.03    1.98   1.01     853.
 5 lambda[4]    1.72      1.69   0.318 0.301   1.27    2.30   1.01     858.
 6 lambda[5]    2.77      2.71   0.540 0.516   1.99    3.75   1.01     924.
 7 lambda[6]    2.81      2.75   0.556 0.532   2.02    3.81   1.01     926.
 8 lambda[7]    1.81      1.77   0.363 0.337   1.29    2.47   1.01     915.
 9 lambda[8]    2.98      2.91   0.614 0.572   2.10    4.10   1.01     984.
10 lambda[9]    1.75      1.72   0.345 0.332   1.26    2.37   1.01     901.
11 lambda[10]   1.49      1.46   0.330 0.312   1.02    2.10   1.01    1092.
12 mu[1,1]      1.40      1.40   0.257 0.255   0.992   1.83   1.00    4164.
13 mu[2,1]      0.205     0.209  0.320 0.315  -0.322   0.731  1.00    1872.
14 mu[3,1]     -0.434    -0.427  0.296 0.296  -0.932   0.0413 1.00    2143.
15 mu[4,1]      0.373     0.377  0.325 0.321  -0.165   0.909  1.00    1862.
16 mu[5,1]      0.379     0.380  0.476 0.467  -0.401   1.16   1.00    1532.
17 mu[6,1]      0.00864   0.0202 0.481 0.473  -0.790   0.783  1.00    1553.
18 mu[7,1]     -0.791    -0.779  0.356 0.351  -1.39   -0.229  1.00    1965.
19 mu[8,1]     -0.335    -0.316  0.518 0.505  -1.22    0.484  1.00    1489.
20 mu[9,1]     -0.718    -0.706  0.345 0.340  -1.31   -0.171  1.00    2120.
21 mu[10,1]    -1.96     -1.94   0.387 0.380  -2.64   -1.37   1.00    3292.
22 mu[1,2]     -0.568    -0.562  0.235 0.233  -0.962  -0.191  1.00    1530.
```

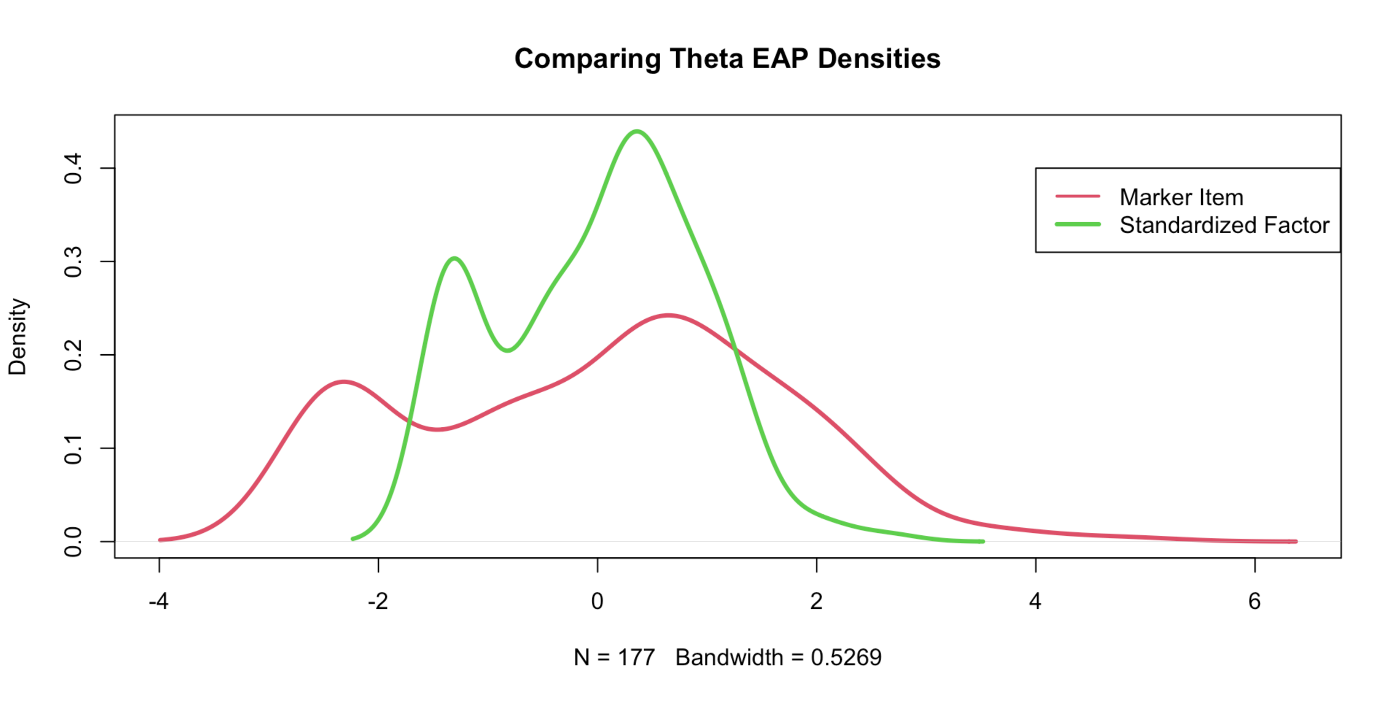
# SD $\theta$ Results



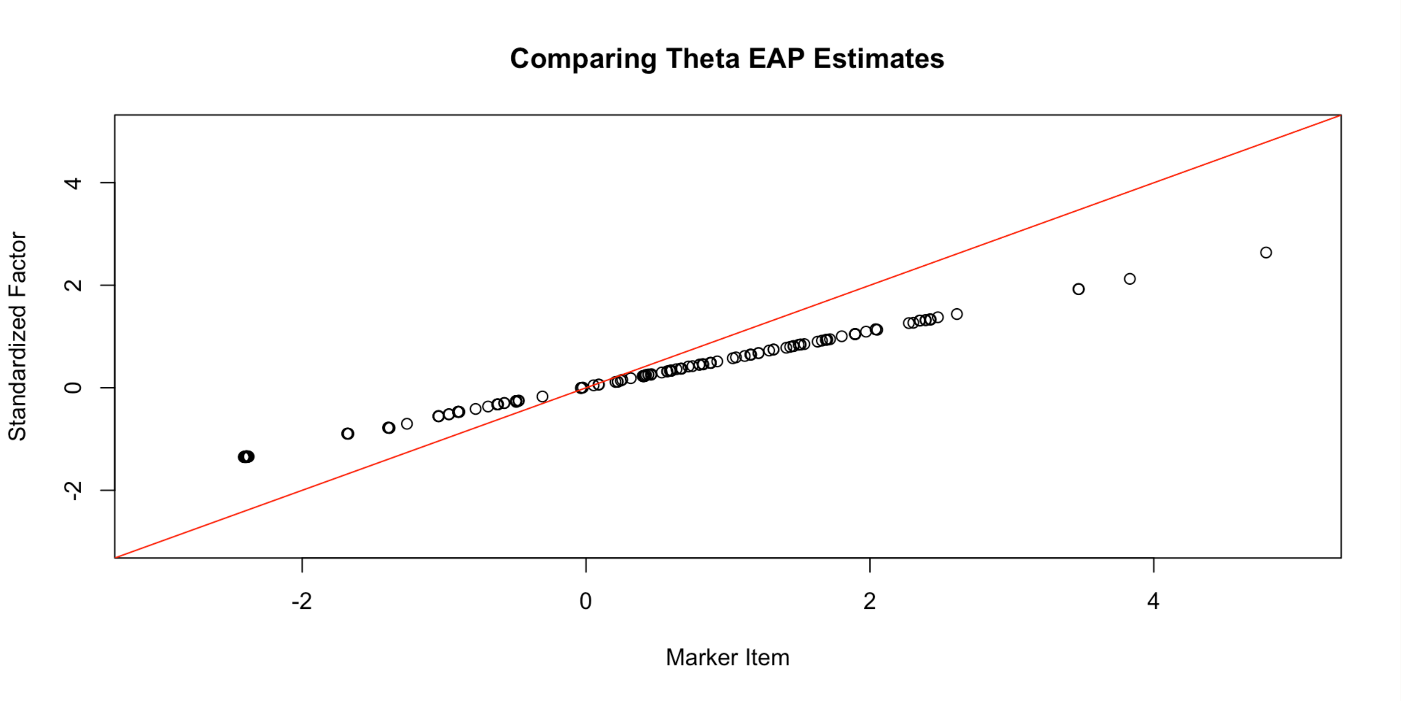
# SD $\theta$ Results



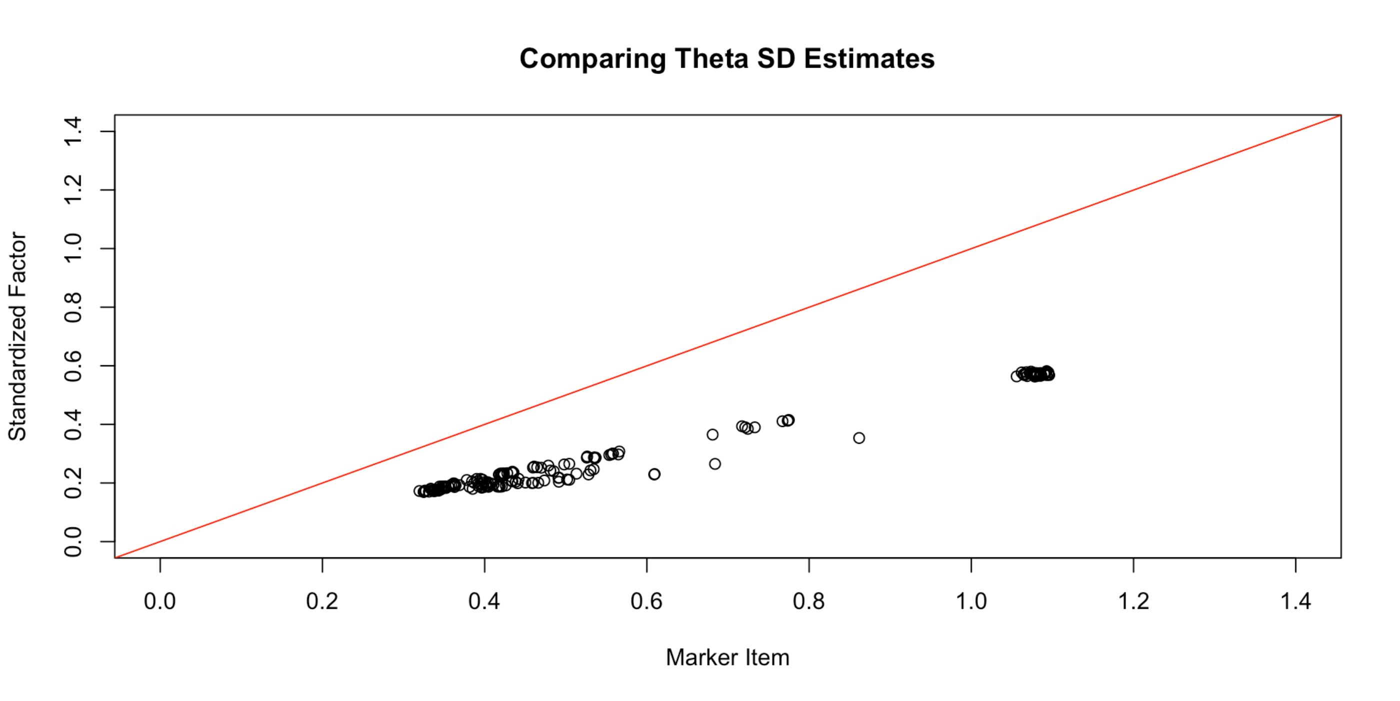
# Comparing $\theta$ EAP Estimates



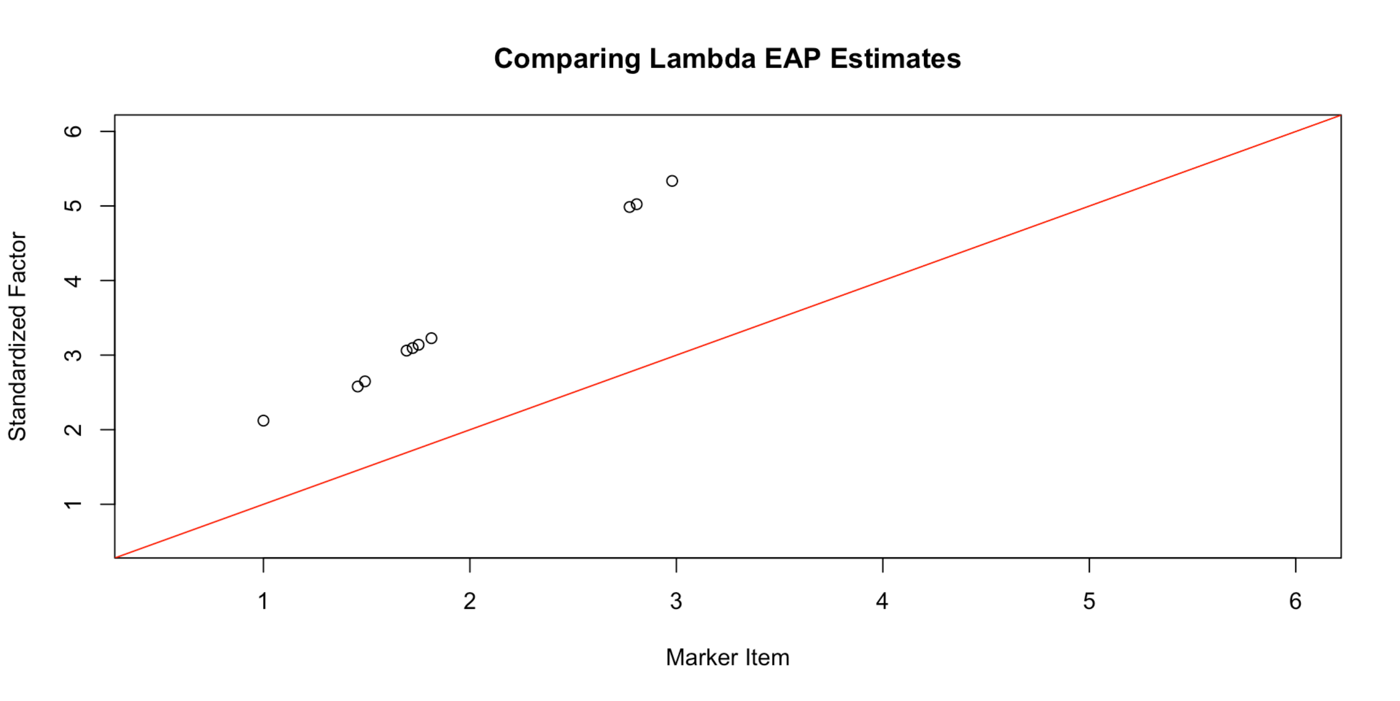
# Comparing $\theta$ EAP Estimates



# Comparing $\theta$ SD Estimates



# Comparing $\lambda$ EAP Estimates



# Marker Items for Multidimensional Models

# Marker Items for Multidimensional Models

We can also build marker items into multidimensional models

- A bit more tricky–we need to estimate the covariance matrix of $\theta$ now ($\Sigma_\theta$)
    - We accomplish this by pre- and post-multiplying the correlation matrix of $\theta$ by a diagonal matrix of the standard deviations of $\theta$

# Stan’s Model Block

```
model {

  matrix[nFactors, nFactors] thetaCovL;
  initLambda ~ multi_normal(meanLambda, covLambda);

  thetaCorrL ~ lkj_corr_cholesky(1.0);
  thetaSD ~ lognormal(sdThetaLocation,sdThetaScale);

  thetaCovL = diag_pre_multiply(thetaSD, thetaCorrL);
  theta ~ multi_normal_cholesky(meanTheta, thetaCovL);


  for (item in 1:nItems){
    thr[item] ~ multi_normal(meanThr[item], covThr[item]);
    Y[item] ~ ordered_logistic(thetaMatrix*lambdaMatrix[item,1:nFactors]', thr[item]);
  }


}
```

Notes:

- The `multi_normal_cholesky` function now uses the covariance matrix
- We calculate the covariance matrix using the `thetaCovL = diag_pre_multiply(thetaSD, thetaCorrL);;` line
- The SDs have a lognormal prior distribution for each

# Stan’s Parameters Block

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

  cholesky_factor_corr[nFactors] thetaCorrL;
  vector<lower=0>[nFactors] thetaSD;
}
```

Notes:

- We still have a correlation matrix estimated
- Now adding a vector of SDs

# Stan’s Transformed Data Block

```
transformed data{
  int<lower=0> nLoadings = 0;                                      // number of loadings in model
  array[nFactors] int<lower=0> markerItem = rep_array(0, nFactors);

  for (factor in 1:nFactors){
    nLoadings = nLoadings + sum(Qmatrix[1:nItems, factor]);
  }

  array[nLoadings, 4] int loadingLocation;                     // the row/column positions of each loading, plus marker sw

  int loadingNum=1;
  int lambdaNum=1;
  for (item in 1:nItems){
    for (factor in 1:nFactors){
      if (Qmatrix[item, factor] == 1){
        loadingLocation[loadingNum, 1] = item;
        loadingLocation[loadingNum, 2] = factor;
        if (markerItem[factor] == 0){
          loadingLocation[loadingNum, 3] = 1;     // ==1 if marker item, ==0 otherwise
          loadingLocation[loadingNum, 4] = 0;     // ==0 if not one of estimated lambdas
          markerItem[factor] = item;
        } else {
          loadingLocation[loadingNum, 3] = 0;
          loadingLocation[loadingNum, 4] = lambdaNum;
          lambdaNum = lambdaNum + 1;
        }
        loadingNum = loadingNum + 1;
      }
    }
  }
```

# Stan’s Transformed Data Block Notes

- Loading location now lists two additional columns
  - An indicator of whether or not to set value to one
  - An indicator of which loading in the loading vector is needed if loading is being estimated

# Stan’s Transformed Parameters Block

```
transformed parameters{
  matrix[nItems, nFactors] lambdaMatrix = rep_matrix(0.0, nItems, nFactors);
  matrix[nObs, nFactors] thetaMatrix;

  // build matrix for lambdas to multiply theta matrix

  for (loading in 1:nLoadings){
    if (loadingLocation[loading,3] == 1){
      lambdaMatrix[loadingLocation[loading,1], loadingLocation[loading,2]] = 1.0;
    } else {
      lambdaMatrix[loadingLocation[loading,1], loadingLocation[loading,2]] = initLambda[loadingLocation[loading,4]];
    }
  }

  for (factor in 1:nFactors){
    thetaMatrix[,factor] = to_vector(theta[,factor]);
  }

}
```

Notes:

- Here, we set the first item’s loading to one for each dimension
- We determine the location of each loading using the results from the transformed data block sorting the Q-matrix

# Stan’s Generated Quantities Block

```
generated quantities{
  array[nItems] vector[maxCategory-1] mu;
  corr_matrix[nFactors] thetaCorr;
  cholesky_factor_cov[nFactors] thetaCov_pre;
  cov_matrix[nFactors] thetaCov;

  for (item in 1:nItems){
    mu[item] = -1*thr[item];
  }

  thetaCorr = multiply_lower_tri_self_transpose(thetaCorrL);
  thetaCov_pre = diag_pre_multiply(thetaSD, thetaCorrL);
  thetaCov = multiply_lower_tri_self_transpose(thetaCov_pre);
}
```

Notes:

- Now we calculate the covariance matrix here, too

# Stan’s 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

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

  vector[nFactors] meanTheta;
  vector[nFactors] sdThetaLocation;
  vector[nFactors] sdThetaScale;
}
```

Notes:

- Adding hyperparameters for the SDs of *theta*
- No other differences from previous multidimensional model code

# R Data List

```r
thetaMean = rep(0, nFactors)
sdThetaLocation = rep(0, nFactors)
sdThetaScale = rep(.5, nFactors)

modelMultidimensionalGRM_markerItem_data = list(
  nObs = nObs,
  nItems = nItems,
  maxCategory = maxCategory,
  Y = t(conspiracyItems),
  nFactors = nFactors,
  Qmatrix = Qmatrix,
  meanThr = thrMeanMatrix,
  covThr = thrCovArray,
  meanLambda = lambdaMeanVecHP,
  covLambda = lambdaCovarianceMatrixHP,
  meanTheta = thetaMean,
  sdThetaLocation = sdThetaLocation,
  sdThetaScale = sdThetaScale
)
```

# Stan Results

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

```
[1] 1.050782
```

```r
# parameter results
print(modelMultidimensionalGRM_markerItem_samples$summary(variables = c("thetaSD", "thetaCov", "thetaCorr", "lambdaMatrix"
```

```
# A tibble: 70 × 10
   variable         mean    median      sd     mad      q5     q95  rhat ess_bulk
   <chr>           <dbl>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>    <dbl>
 1 thetaSD[1]      2.48    2.44e+0 0.371   0.354     1.92    3.20    1.02    203.
 2 thetaSD[2]      1.74    1.74e+0 0.255   0.260     1.34    2.16    1.01    201.
 3 thetaCov[1,…    6.29    5.96e+0 1.90    1.72      3.68   10.3     1.02    203.
 4 thetaCov[2,…    4.31    4.18e+0 1.04    0.941     2.83    6.36    1.02    176.
 5 thetaCov[1,…    4.31    4.18e+0 1.04    0.941     2.83    6.36    1.02    176.
 6 thetaCov[2,…    3.11    3.02e+0 0.908   0.902     1.79    4.69    1.01    201.
 7 thetaCorr[1…    1       1   e+0 0       0         1       1      NA        NA
 8 thetaCorr[2…    0.989   9.91e-1 0.00868 0.00773   0.972   0.998   1.03     48.9
 9 thetaCorr[1…    0.989   9.91e-1 0.00868 0.00773   0.972   0.998   1.03     48.9
10 thetaCorr[2…    1       1   e+0 0       0         1       1      NA        NA
11 lambdaMatri…    0       0       0       0         0       0      NA        NA
12 lambdaMatri…    1       1   e+0 0       0         1       1      NA        NA
13 lambdaMatri…    0       0       0       0         0       0      NA        NA
14 lambdaMatri…    0       0       0       0         0       0      NA        NA
15 lambdaMatri…    1.93    1.89e+0 0.392   0.368     1.37    2.65    1.01    247.
16 lambdaMatri…    0       0       0       0         0       0      NA        NA
17 lambdaMatri…    1.26    1.23e+0 0.254   0.256     0.887   1.70    1.00    321.
18 lambdaMatri…    2.08    2.02e+0 0.440   0.415     1.44    2.87    1.01    329.
19 lambdaMatri…    1.21    1.19e+0 0.231   0.220     0.871   1.63    1.00    413.
20 lambdaMatri…    0       0       0       0         0       0      NA        NA
21 lambdaMatri…    1       1   e+0 0       0         1       1      NA        NA
22 lambdaMatri…    0       0       0       0         0       0      NA        NA
```

# Wrapping Up

# Wrapping Up

This lecture showed how to set additional constraints to estimate the standard deviation of the latent variable(s)

- This is often called the structural model in psychometrics
- Bayesians call this an empirical prior
- We need parameter constraints to provide strong identification for the model/data likelihood