Bayesian latent variable modeling in education research

Ed Merkle

NCME 2024 Training Session

Acknowledgments

- *blavaan* has been funded by Institute of Education Sciences Grant R305D210044, and it has seen many contributors and collaborators over time! It would not be possible without Yves Rosseel.

Introduction

- Goals of this session
  - Discuss the major ideas underlying Bayesian latent variable models
  - See how these ideas work in practice
  - Discuss some advanced issues and workflows

Introduction

- How will we meet those goals?
  - Emphasize the “raw data” part of the models
  - Model real datasets that are easily obtainable
  - Include lots of code, so that you can work through concepts in the future

Introduction

- What is not covered: R background, highly exotic psychometrics models.
- Also, we will not spend time assessing MCMC convergence. See here for details on how it works in *blavaan*.
- Also, there is much more checking and summary we could do with every model that we discuss. We are time limited.
- I apologize now for discussing what you already know, and for assuming you know what you don't know!

Introduction

- Outline
  - Latent variable model overview
  - Bayesian methods overview
  - Case studies: CFA, IRT, two-level
  - Recommendations, workflows
  - Advanced topics, requests

Model Overview

Model Overview

- In psychometrics and education, the latent variable in "latent variable models" is typically a person's unobserved trait.
- Let's follow (some) tradition and call it θ.
- While we cannot observe θ directly, we can observe some other variable that serves as a proxy for θ. Let's call that other variable y.

Linear

- Linear equations are everywhere in statistics. So too in psychometrics.
- For traditional models, we assume that our observed variable is a noisy, linearly function of θ.
- For now, let's write it as

$$y = \beta_0 + \beta_1\theta + e$$

Linear

$$y = \beta_0 + \beta_1 \theta + e$$

- This is a model of one observed variable, but we typically have multiple observed variables.
- If θ were observed, this would be a regression model.
- We left out subscripts. So we could say that this is the model of a single person on a single observed variable.

Subscripts

Let's add subscripts for person i and variable j

$$y_{ij} = \beta_{0j} + \beta_{1j}\theta_i + e_{ij}$$

- The subscripts on β_0 and β_1 show that the intercept and slope are unique to an observed variable j.

Matrix

- It is customary to collect all the observed variables for person i in one vector. And collect the intercepts for observed variables in one vector, and similarly for the slopes. Then the j subscripts disappear:

$$\boldsymbol{y}_i = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \theta_i + \boldsymbol{e}_i$$

where bold represents a vector (or a matrix, though on matrices are on this slide)

Matrix

- We may also be interested in measuring multiple latent traits per person. In this case, we also have a vector for θ:

$$\boldsymbol{y}_i = \beta_0 + \beta_1 \boldsymbol{\theta}_i + \boldsymbol{e}_i$$

- Then β_1 becomes a matrix.

Models

$$\beta_0 + \beta_1 \boldsymbol{\theta}_i + \boldsymbol{e}_i$$

- From this simple linear equation, we can obtain many traditional psychometric models:
  - Observed variables are continuous: Factor analysis, SEM
  - Observed variables are binary or ordinal: 2-parameter logistic model, graded response model, generalized partial credit model.
  - In the latter case, the linear equation does not directly predict the observed variables. The linear equation predicts (a function of) the probability that a particular person assumes a particular category of a particular variable.

Models

- The previous slide implies that β_0 and β_1 have different names in different situations. (and also different Greek letters!)
  - β_0: Intercept, mean, difficulty, easiness
  - β_1: Loading, discrimination
- We have also ignored the fact that one latent variable can be predictive of a second latent variable. This is what SEM is about, and it requires a second linear equation.

SEM

- The *structural* equation of SEM:

$$\boldsymbol{\theta}_i = \boldsymbol{\alpha}_0 + \boldsymbol{\alpha}_1 \boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i$$

- This can look crazy because $\boldsymbol{\theta}_i$ appears on both sides of the equation!
- The key is to realize that no single element of $\boldsymbol{\theta}_i$ can predict itself. Each element can predict other elements, making this just another linear equation.

Model Summary

- So far, we have seen that the traditional psychometric models involve linear relationships between the latent variables $\boldsymbol{\theta}$ and (functions of) the observed variables $\boldsymbol{y}$. And in SEM, we can have linear relationships between different elements of $\boldsymbol{\theta}$.
- We have also emphasized that these are models of raw data. Some presentations begin with models of covariance matrices instead of raw data. I find that intuition is lost when we begin with covariance matrices.

Bayesian Introduction

Model Estimation

- Traditionally, models are estimated via Maximum Likelihood. Idea:
  - The model defines a *likelihood function*. Inputs to the function are data and model parameters. Output is a single number ("the likelihood").
  - Different parameter values produce different numbers as output.
  - We seek the parameter values that output the largest number, given our data.

Model Estimation

- How does this happen? Think about climbing a mountain.
  - Start somewhere on the mountain (start with some parameter values).
  - Figure out which way is up.
  - Take a step in the upward direction (refine your parameter values).
  - Continue these steps until you reach the peak.

Bayesian Estimation

- The same model likelihood is involved in Bayesian estimation. But now we additionally take into account our prior expectations/beliefs about model parameters.
- These expectations are encoded as *prior distributions*. They are sort of a generalization of maximum likelihood:
  - Maximum likelihood estimation: No/flat prior beliefs, anything can happen.
  - Bayesian estimation: Prior beliefs could be flat, but they can also be informative.

Bayesian Estimation

- "Prior expectations sound subjective, and I don't want a subjective statistical analysis."
  - If it involves real data, some subjective decisions are already being made.
  - You might not know what parameter values to expect, but you probably know what you *do not* expect. There is where prior distributions can be helpful.

Identification

- Latent variables have no inherent location or scale.
  - It is customary to fix each latent variable's mean to 0 and variance to 1. (or, sometimes, fix a loading to 1 instead of the variance.)
  - These *identification constraints* can influence our prior beliefs about parameter values. We will keep this in mind as we work through examples.
  - Sometimes, the identification constraints change the stated prior distribution. See this hyperlink.

Bayesian Estimation

- Beyond prior distributions, Bayesian model estimation procedures usually differ from Maximum Likelihood:
  - Maximum likelihood is seeking the top of the mountain (of the likelihood)
  - Bayesian estimation typically surveys the full mountain, as opposed to only finding the top. This is accomplished via Markov chain Monte Carlo.

MCMC

- Markov chain Monte Carlo: Draw samples of parameters from the posterior distribution.
  - Parameter values that are more likely will tend to be drawn more often.
  - If we draw many samples, we can produce accurate summaries of the posterior distribution.
  - But we need *many* samples, and this can use lots of computer memory!

MCMC

- Specific flavors of MCMC include Gibbs sampling, Metropolis Hastings sampling, and Hamiltonian Monte Carlo.
- These can differ in speed, efficiency, and flexibility.
- The MCMC methods are best compared by visualization:

https://chi-feng.github.io/mcmc-demo/app.html

Steps

- Steps that we will use in our Bayesian case studies to come:
  - Set prior distributions, making use of prior predictive checks.
  - Estimate model via MCMC.
  - Do posterior predictive checks and other model criticisms.
  - Summarize key results.

Case 1: CFA

CFA

- For our first case study, we will do a CFA of item response data.
- I might get banned from the NCME conference, but I will apply a model of continuous data to 0/1 item responses!
- Why? It will be easier to see differences between the CFA and the item response model that comes next.

Data

- Data: Responses of 565 Austrian students to 7 mathematics items from PISA 2009.
- Conveniently included in the *sirt* package:

```
data("data.pisaMath", package = "sirt")

dat <- data.pisaMath$data
```

Data

- Obtaining response patterns

```
patts <- with(dat, paste0(M192Q01, M406Q01, M423Q01, M496Q01, M564Q01,
                          M571Q01, M603Q01))
```

Data

```
summary( as.factor(patts) )
```

```
## 1111111 0000000 0010000 0111111 1111011 0010001 0011000 0011001 0011111 1101111 1111101
##      51      24      22      13      13      12      12      12      12      12      12
## 0010100 1011111 0011011 0011100 1011011 1110111 1111110 0010010 0111011 1010011 1111001
##      11      11       9       9       9       9       9       8       8       8       8
## 0000001 0000100 0000110 0011010 0110100 1010000 1010001 0001000 0010101 0010110 0011110
##       7       7       7       7       7       7       7       7       6       6       6       6
## 1010111 1011000 1011001 0000010 0100111 0110011 0110110 1010100 1111010 0010011 0100100
##       6       6       6       5       5       5       5       5       5       4       4
## 0101110 0111000 0111010 1001111 1010101 1010110 1011110 1110001 1111100 0001001 0001100
##       4       4       4       4       4       4       4       4       4       3       3
## 0001110 0001111 0101011 0110001 0110111 0111100 0111110 1000101 1110100 1110101 1111000
##       3       3       3       3       3       3       3       3       3       3       3
## 0000111 0001011 0100010 0100101 0101111 0110000 0110010 0111101 1000001 1000011 1000111
##       2       2       2       2       2       2       2       2       2       2       2
## 1001011 1010010 1011010 1011100 1100111 1101010 1101011 1110010 1110011 0000011 0000101
##       2       2       2       2       2       2       2       2       2       1       1
## 0001010 0010111 0011101 0100000 0100011 0100110 0101000 0101010 0101101 0110101 0111001
##       1       1       1       1       1       1       1       1       1       1       1
## (Other)
##      14
```

Model

- We are fitting a 1-factor CFA that assumes continuous responses!
- Parameters:
  - Intercepts (1 for each item).
  - Slopes (1 loading for each item).
  - Residual standard deviations (1 for each item).
  - Factor mean fixed to 0, factor variance fixed to 1.

Model

- These are all binary items scored as correct/incorrect. What do we already know about the parameters?
  - Intercepts: Should be between 0 and 1 (related to percent correct).
  - Slopes: Between 0 and 1 (person parameters are like a z-score, and they are predicting a number between 0 and 1).
  - Residual standard deviations: No larger than 1.

Priors

- Based on this knowledge, some "mildly informative" prior distributions:
  - Intercepts: Normal(.5, .5) $\leftarrow$ second number is SD
  - Slopes: Normal(.5, .25)
  - Residual SD: Gamma(1, 1)

Priors

- How did I come up with Gamma(1, 1)?
- One approach: we refresh yourself about properties of the gamma distribution, which provides intuition for setting prior distribution parameters.
- But I typically summarize random numbers:

```
summary( rgamma(1e5, 1, 1) )
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  0.000009  0.286489  0.692068  0.996326  1.380861 10.338650
```

Priors

- We will now specify these prior distributions in blavaan.
- Greek letters correspond to LISREL parameters. For us: nu is intercepts, lambda is loadings, theta is residual standard deviations.

```
mypriors <- dpriors(nu = "normal(.5, .5)", lambda = "normal(.5, .25)",
                    theta = "gamma(1, 1)[sd]")
```

- Correspondence between Greek letters and parameters is found at this hyperlink.

Priors

- Now that we have priors, let's generate data from these priors to see whether the data are plausible.

- We will use `prisamp = TRUE` in blavaan:

```
## specifying my model:
m1 <- ' f1 =~ M192Q01 + M406Q01 + M423Q01 + M496Q01 + M564Q01 + M571Q01 + M603Q01 '

## drawing prior samples (100 for each of three chains):
m1pri <- bcfa(m1, data = dat, burnin = 100, sample = 100, std.lv = TRUE, prisamp = TRUE,
              dp = mypriors)
```

```
##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000195 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.95 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: There aren't enough warmup iterations to fit the
## Chain 1:          three stages of adaptation as currently configured.
## Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
```

Priors

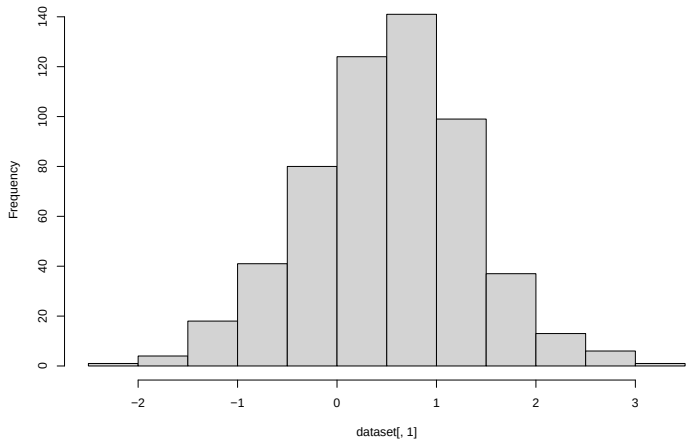
- Now we can generate data from these prior samples. The command below gives us 300 datasets, each with 565 individuals and 7 variables.

```
pridat <- sampleData(m1pri, simplify = TRUE)
```

Priors

- Histogram of the first variable from one generated dataset:

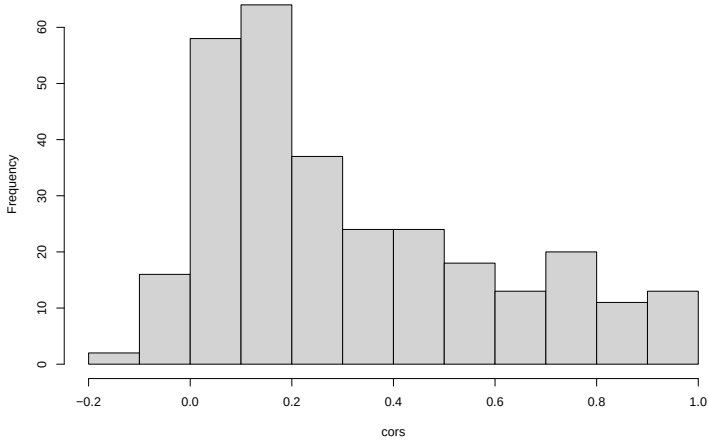
```
dataset <- pridat[[ 1 ]]
hist(dataset[, 1], main = "")
```



Priors

- We can also obtain a histogram of correlations between the first two variables. But now we have to summarize across datasets.

```
cors <- sapply(pridat, function(x) cor( x[,1], x[,2] ))
hist(cors, main = "")
```



Priors

- From these summaries, we see what "mildly informative" means: variables are close to being between 0 and 1, and correlations are on the positive side.
- If we wanted to, we could make revisions to keep the data closer to $(0, 1)$. The current dataset is somewhat large, so it might not matter.
- But sensitivity analysis is also worthwhile!

Estimation

- We now estimate the model by removing `prisamp = TRUE` from our previous command, and running for longer:

```
m1est <- bcfa(m1, data = dat, burnin = 1000, sample = 1000, std.lv = TRUE,
              dp = mypriors)

##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000204 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.04 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

Estimation I

```
summary(m1est)

## blavaan 0.5.3.1251 ended normally after 1000 iterations
##
##   Estimator                                         BAYES
##   Optimization method                               MCMC
##   Number of model parameters                          14
##
##   Number of observations                             565
##
##   Statistic                                 MargLogLik         PPP
##   Value                                      -2657.623       0.010
##
## Parameter Estimates:
##
##
## Latent Variables:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   f1 =~
##     M192Q01           0.267    0.026    0.215    0.317    0.999 normal(.5, .25)
##     M406Q01           0.277    0.025    0.228    0.324    0.999 normal(.5, .25)
##     M423Q01           0.103    0.022    0.061    0.145    0.999 normal(.5, .25)
##     M496Q01           0.236    0.025    0.186    0.285    1.000 normal(.5, .25)
##     M564Q01           0.180    0.026    0.128    0.230    0.999 normal(.5, .25)
##     M571Q01           0.254    0.026    0.204    0.304    0.999 normal(.5, .25)
##     M603Q01           0.249    0.025    0.198    0.298    0.999 normal(.5, .25)
##
## Variances:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .M192Q01           0.180    0.014    0.155    0.208    0.999 gamma(1, 1)[sd]
```

Estimation II

```
##    .M406Q01          0.173    0.013    0.149    0.199    0.999 gamma(1, 1)[sd]
##    .M423Q01          0.183    0.011    0.163    0.206    0.999 gamma(1, 1)[sd]
##    .M496Q01          0.195    0.014    0.169    0.222    1.000 gamma(1, 1)[sd]
##    .M564Q01          0.220    0.014    0.194    0.249    0.999 gamma(1, 1)[sd]
##    .M571Q01          0.186    0.014    0.161    0.215    1.000 gamma(1, 1)[sd]
##    .M603Q01          0.188    0.013    0.164    0.216    1.000 gamma(1, 1)[sd]
##     f1               1.000
```

Estimation Notes

- The intercepts do not appear in the output. They are not needed for estimation of this model (the sample covariance matrix is a sufficient statistic), but can be obtained by estimating with the argument `meanstructure = TRUE`.
- The third item stands out as having a lower loading than the others.
- The posterior predictive p-value is low, suggesting not-so-good model fit.

Fit Indices

- Bayesian versions of traditional fit indices are available (Garnier-Villareal & Jorgensen, 2020).

```
res <- blavFitIndices(m1est, pD = "loo")

summary(res)

##
## Posterior summary statistics and highest posterior density (HPD) 90% credible intervals for devm-b
##
##                 EAP Median   MAP    SD lower upper
## BRMSEA        0.046  0.045 0.045 0.005 0.037 0.053
## BGammaHat     0.988  0.989 0.989 0.003 0.985 0.993
## adjBGammaHat 0.983  0.984 0.984 0.004 0.978 0.989
## BMc           0.979  0.980 0.981 0.005 0.973 0.987
```

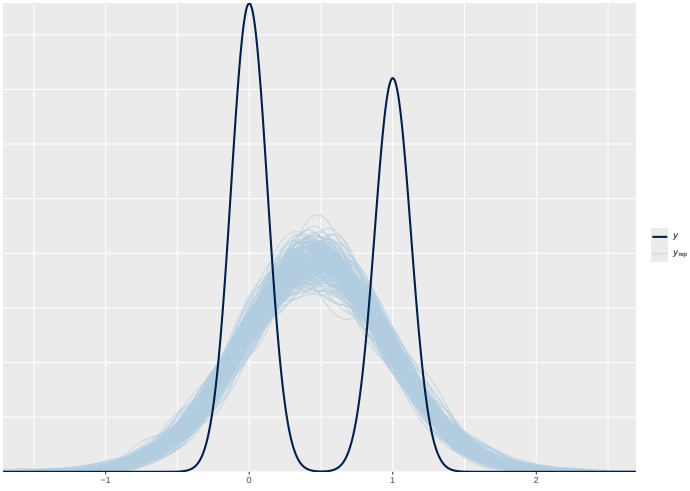
Posterior Checks

- While global statistics of model fit are helpful, targeted posterior checks can provide detailed information about fit:
  - Generate data from your posterior distribution, summarize
  - Summarize your observed data in the same way
  - Create visual comparisons
- Your checks can be anything, so long as you can code it! In the following slides, we show some examples using the *bayesplot* package. Some extra commands will help us arrange data in the required format.

Posterior Checks I

```
## First item: observed data vs posterior predictions
postdat <- sampleData(m1est, nrep = 1000, simplify = TRUE)

y1rep <- t( sapply(postdat, function(x) x[,1]) )

ppc_dens_overlay(y = blavInspect(m1est, 'data')[, 1], yrep = y1rep[1:200, ])
```

Posterior Checks II



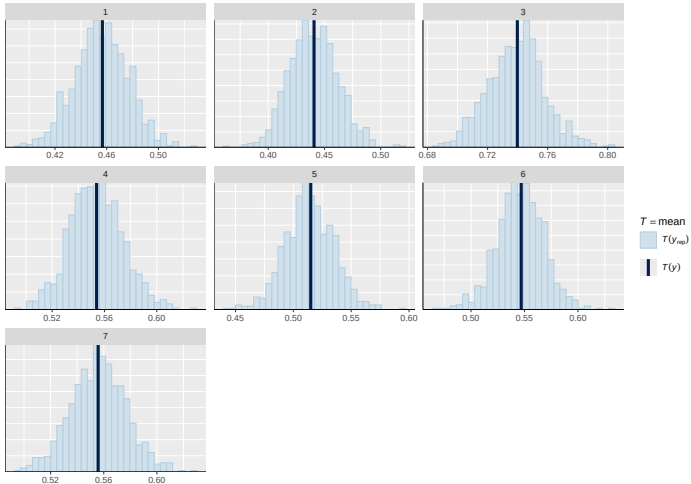
Posterior Checks

- The previous graph shows that our observed data are $0/1$, but the model predicts continuous data.
- The model predictions are sort of similar to percent correct, except they go below 0 and above 1.

Posterior Checks I

```
## Percent correct vs posterior predictions, all items
obsdat <- blavInspect(m1est, 'data')

yrep <- lapply(1:7, function(i) t( sapply(postdat, function(x) x[, i]) ))
yrep <- do.call("cbind", yrep)

ppc_stat_grouped(y = as.numeric(obsdat), yrep = yrep, group = rep(1:7, each = nobs(m1est)))
```

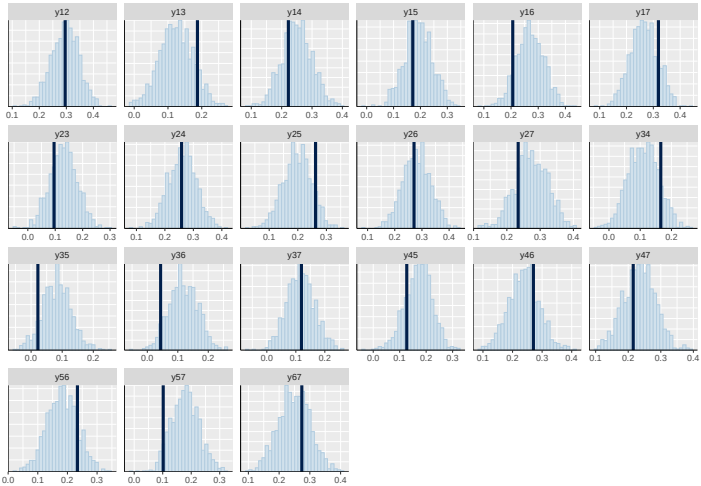
Posterior Checks II



Posterior Checks I

```
## Observed correlations vs posterior correlations, all pairs of items
cormat <- cor(obsdat)
obscor <- cormat[ lower.tri(cormat) ]

correp <- lapply(postdat, cor)
correp <- lapply(correp, function(x) x[ lower.tri(x) ])
correp <- do.call("rbind", correp)

yobs <- as.numeric(obscor)

grp <- as.factor( unlist( sapply(1:6, function(i) paste0("y", i, (i+1):7)) ) )
ppc_stat_grouped(y = yobs, yrep = correp, group = grp, facet_args = list(nrow = 4)) +
  theme(legend.position = "none")
```

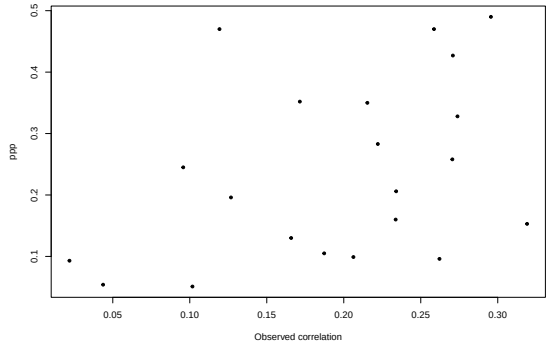
Posterior Checks II



Posterior check

- The graph below shows that the model has trouble capturing small observed correlations.

```
ppps <- sapply(1:21, function(i) min(mean(correp[ ,i] < yobs[i]), mean(correp[, i] > yobs[i])))

plot(obscor, ppps, xlab = "Observed correlation", ylab = "ppp", pch = 20)
```



Summary

- So:
  - Our overall model fit is not the best, also not the worst.
  - The model had trouble predicting weak correlations. This makes sense, because a one-factor model is generally about predicting relationships between all observed variables.
  - But we treated our binary data as continuous! Next, we see what happens when we treat the data as categorical.

Case 2: IRT

IRT

- Now, we use an item response model that is specifically developed for binary responses.
- Before: We were predicting our 0/1 data on a continuous scale. Model predictions were like probabilities, except they could go below 0 and above 1.
- Now: Same model, except it is predicting the z-score associated with the probability of being correct. The z-scores can be negative or positive, so it is no problem to have model predictions below 0 or above 1.

Model

- The types of model parameters are basically the same as before:
  - Intercept (threshold): Larger numbers mean you are less likely to be correct. So sometimes called *item difficulty*.
  - Slope: How do person parameters influence chance (z-score) of being correct?
  - Residual standard deviations: Fixed to 1, because variance of 0/1 data is determined by the mean.
  - Factor mean fixed to 0, factor variance fixed to 1.

Model

- Note: IRT modelers will often use an equivalent difficulty/discrimination parameterization, instead of slope/intercept! It is possible to obtain one set of parameters from the other.
  - Slope/intercept: $-\beta_{0j} + \beta_{1j}\theta_i$
  - Discrimination/difficulty: $\alpha_{1j}(\theta_i - \delta_{0j})$

Model

- Also note: IRT modelers will often predict the log-odds of being correct, instead of the z-score of being correct!
- This is the difference between the logit link function (log odds) and the probit link function (z-score).
- So our model could be called a two-parameter probit, vs a two-parameter logistic.
- Probit parameter estimates can be converted to logit estimates, and vice versa. (see, e.g., McDonald, 1999)

Priors

- When setting priors, it helps to remember that we are predicting z-scores. It would be unusual to observe a number outside of $(-3, 3)$. So some initial prior distributions could be:
  - Intercepts: Normal(0, 1) $\leftarrow$ second number is SD
  - Slopes: Normal(1, .5)

Priors

- Like before, we specify the prior distributions in blavaan. Intercepts are now in tau.

```
mypriors <- dpriors(tau = "normal(0, 1.5)", lambda = "normal(1, .5)")
```

Priors

- Like before, we generate parameters from the priors:

```
## the model is the same as before:
m2 <- ' f1 =~ M192Q01 + M406Q01 + M423Q01 + M496Q01 + M564Q01 + M571Q01 + M603Q01 '

## drawing prior samples (100 for each of three chains):
m2pri <- bcfa(m2, data = dat, burnin = 100, sample = 100, std.lv = TRUE, prisamp = TRUE,
              dp = mypriors, ordered = TRUE)

##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001222 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 12.22 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: There aren't enough warmup iterations to fit the
## Chain 1:          three stages of adaptation as currently configured.
## Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 1:          the given number of warmup iterations:
## Chain 1:            init_buffer = 15
## Chain 1:            adapt_window = 75
## Chain 1:            term_buffer = 10
```

Priors

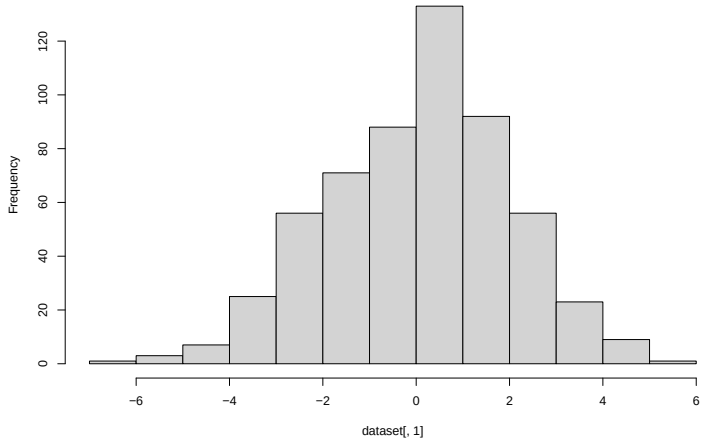
- Now we can generate data from the priors. The `type = "link"` argument says that we want to generate z-scores associated with the probability of being correct, as opposed to the original 0/1 data:

```
pridat <- sampleData(m2pri, simplify = TRUE, type = "link")
```

Priors

- Histogram of the first variable from one generated dataset:

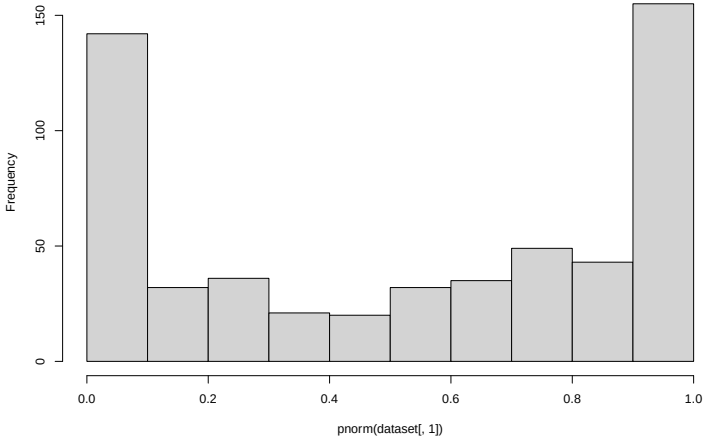
```
dataset <- pridat[[ 1 ]]
hist(dataset[, 1], main = "")
```



Priors

- Translated to probabilities of being correct:

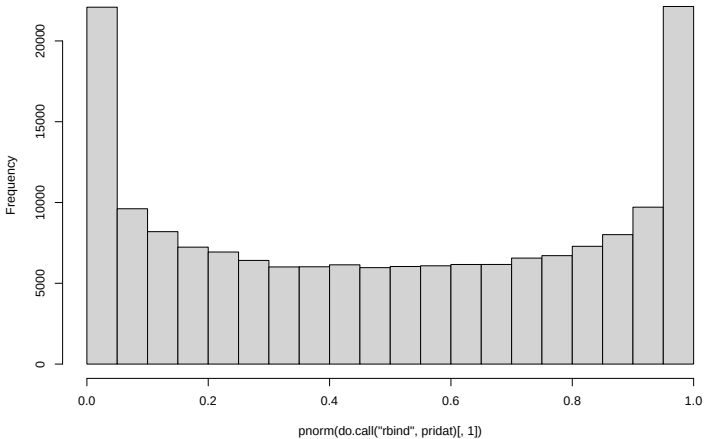
```
hist(pnorm( dataset[, 1] ), main = "")
```



Priors

- The previous histogram produced data where people had extreme chances of being correct. Does it hold across all the generated datasets?

```
hist(pnorm( do.call("rbind", pridat)[, 1] ), main = "")
```



Priors

- If the items are good, then we would expect test-takers to have a probability of being correct that is not too close to 0 or 1. So we might revise our priors to not produce such extreme probabilities.
  - Intercept: z-score associated with probability that the average test-taker gets the item correct. Say that the probability ranges from around .2 to .8, which are z-scores of $\pm$.84. Normal(0, .28)
  - Slope: For two test-takers whose proficiencies differ by 1 SD, what is the expected increase in probability correct? On probit scale, an increase of 1 could go from 16% chance to 50% chance, which is large. So Normal(.4, .2)

Priors

- The discussion on the previous slide leads to

```
mypriors <- dpriors(tau = "normal(0, .28)", lambda = "normal(.4, .2)")
```

Posteriors

- We claim that our prior distributions reflect general expectations, but are only mildly informative.
- But we can also do a sensitivity analysis to explore how our priors influence results.
- So we fit two models, one with our informative priors and one without.

```
m2fit <- bcfa(m2, data = dat, std.lv = TRUE, ordered = TRUE, dp = mypriors)
```

```
##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001441 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 14.41 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  501 / 1500 [ 33%]  (Sampling)
## Chain 1: Iteration:  650 / 1500 [ 43%]  (Sampling)
```

Posteriors

```
m2nifit <- bcfa(m2, data = dat, std.lv = TRUE, ordered = TRUE)
```

```
##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001442 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 14.42 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  501 / 1500 [ 33%]  (Sampling)
## Chain 1: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 1: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 1: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 1: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 1: Iteration: 1250 / 1500 [ 83%]  (Sampling)
## Chain 1: Iteration: 1400 / 1500 [ 93%]  (Sampling)
## Chain 1: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 48.051 seconds (Warm-up)
## Chain 1:                47.998 seconds (Sampling)
```

Posteriors I

```
summary(m2fit)

## blavaan 0.5.3.1251 ended normally after 1000 iterations
##
##   Estimator                                         BAYES
##   Optimization method                               MCMC
##   Number of model parameters                          14
##
##   Number of observations                             565
##
##   Statistic                                 MargLogLik         PPP
##   Value                                             NA       0.274
##
## Parameter Estimates:
##
##   Parameterization                                Theta
##
## Latent Variables:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   f1 =~
##     M192Q01           0.770    0.095    0.588    0.962    1.003 normal(.4, .2)
##     M406Q01           0.808    0.100    0.626    1.011    1.006 normal(.4, .2)
##     M423Q01           0.328    0.073    0.192    0.473    1.006 normal(.4, .2)
##     M496Q01           0.666    0.085    0.503    0.834    1.000 normal(.4, .2)
##     M564Q01           0.488    0.079    0.343    0.654    1.002 normal(.4, .2)
##     M571Q01           0.721    0.095    0.540    0.913    1.008 normal(.4, .2)
##     M603Q01           0.699    0.091    0.531    0.887    1.001 normal(.4, .2)
##
## Intercepts:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
```

Posteriors II

```
##    .M192Q01           0.000
##    .M406Q01           0.000
##    .M423Q01           0.000
##    .M496Q01           0.000
##    .M564Q01           0.000
##    .M571Q01           0.000
##    .M603Q01           0.000
##     f1                0.000
##
## Thresholds:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##     M192Q01|t1         0.140    0.064    0.011    0.263    1.002 normal(0, .28)
##     M406Q01|t1         0.191    0.067    0.059    0.323    1.000 normal(0, .28)
##     M423Q01|t1        -0.647    0.060   -0.765   -0.533    1.005 normal(0, .28)
##     M496Q01|t1        -0.156    0.062   -0.284   -0.041    1.000 normal(0, .28)
##     M564Q01|t1        -0.038    0.057   -0.150    0.070    1.000 normal(0, .28)
##     M571Q01|t1        -0.138    0.063   -0.262   -0.019    1.000 normal(0, .28)
##     M603Q01|t1        -0.161    0.062   -0.282   -0.043    1.001 normal(0, .28)
##
## Variances:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .M192Q01           1.000
##    .M406Q01           1.000
##    .M423Q01           1.000
##    .M496Q01           1.000
##    .M564Q01           1.000
##    .M571Q01           1.000
##    .M603Q01           1.000
##     f1                1.000
##
```

Posteriors III

```
## Scales y*:
##                  Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##     M192Q01         0.792
##     M406Q01         0.778
##     M423Q01         0.950
##     M496Q01         0.832
##     M564Q01         0.899
##     M571Q01         0.811
##     M603Q01         0.820
```

Sensitivity

- Comparing informative priors to default priors, the informative priors appear to be no worse by cross-validation metrics:

```
fitMeasures(m2fit)
```

```
##       npar       logl        ppp        bic        dic      p_dic       waic     p_waic
##     14.000  -2493.324      0.274   5075.339   5011.459     12.405   5011.045     11.959
##    se_waic      looic      p_loo     se_loo margloglik
##     43.440   5011.081     11.977     43.440         NA
```

```
fitMeasures(m2nifit)
```

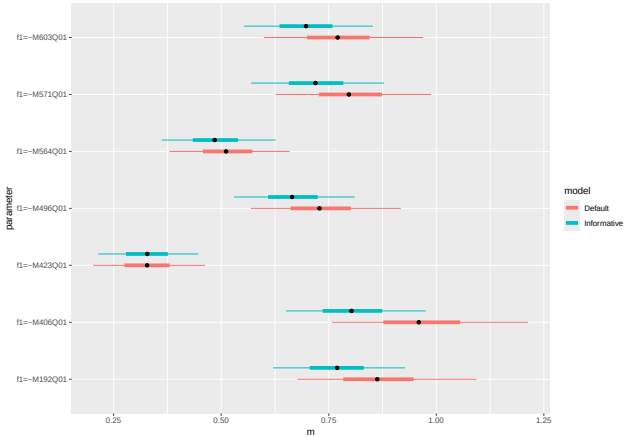
```
##       npar       logl        ppp        bic        dic      p_dic       waic     p_waic
##     14.000  -2491.693      0.311   5072.076   5010.987     13.801   5011.410     14.157
##    se_waic      looic      p_loo     se_loo margloglik
##     47.249   5011.454     14.180     47.249         NA
```

Sensitivity I

- Comparing loadings under the two models (other parameters are more similar):

```
## related to code at https://github.com/stan-dev/bayesplot/issues/232
combined <- rbind(plot(m2fit, 1:7, 'intervals_data', showplot = FALSE),
                  plot(m2nifit, 1:7, 'intervals_data', showplot = FALSE))
combined$model <- rep(c("Informative", "Default"), each = nrow(combined)/2)

pos <- position_nudge(y = ifelse(combined$model == "Default", 0, 0.2))

ggplot(combined, aes(x = m, y = parameter, color = model)) +
  geom_linerange(aes(xmin = l, xmax = h), position = pos, linewidth = 2)+
  geom_linerange(aes(xmin = ll, xmax = hh), position = pos)+
  geom_point(position = pos, color="black")
```

Sensitivity II



Sensitivity

- The graph shows that the informative priors have shrunk some loadings toward zero.
- I am not generally concerned about this because the loadings are notoriously difficult to estimate (like estimating an interaction between person and item, as opposed to a main effect). It is doubtful that shrinkage will hurt the model's generalizability.
- The WAIC and LOO metrics also support the idea that informative priors could provide better generalization. (If you don't know these, they have similar goals to AIC or BIC.)
- But you might also consider the purpose of the model.

Model Checking

- The posterior predictive p-value indicates reasonable model fit. We can dig deeper with customized posterior predictive checks via `ppmc()`.
- There is a good deal of flexibility here because you can write custom R functions involving a blavaan object.
- Bonifay & Depaoli (2021) describe use of the item-total correlation: empirical correlation between each item and the sum of remaining items, vs posterior distribution of the same correlation.

Model Checking

- Define a function that computes item-total correlations, then send to `ppmc()`:

```
it_tot <- function(fit) {
  tmpdata <- fit@Data@X[[1]]
  sapply(1:ncol(tmpdata),
         function(i) cor(tmpdata[,i], rowSums(tmpdata[,-i])))
}

itt2 <- ppmc(m2fit, discFUN = it_tot)
```

Model Checking

- These ppps indicate that the observed item-total correlations fall inside the posterior predictive distributions. Some columns indicate no variability because the observed item-total correlation is a function of data only (as opposed to function of data and model parameters).

```
summary(itt2)
```

```
##
## Posterior summary statistics and highest posterior density (HPD) 95% credible intervals for the posterior distribution of realized
##
##
##     EAP Median   MAP SD lower upper PPP_sim_GreaterThan_obs PPP_sim_LessThan_obs
## 1 0.415  0.415 0.414  0 0.415 0.415                   0.130                0.870
## 2 0.424  0.424 0.423  0 0.424 0.424                   0.137                0.863
## 3 0.176  0.176 0.176  0 0.176 0.176                   0.522                0.478
## 4 0.368  0.368 0.368  0 0.368 0.368                   0.235                0.765
## 5 0.264  0.264 0.264  0 0.264 0.264                   0.518                0.482
## 6 0.386  0.386 0.386  0 0.386 0.386                   0.230                0.770
## 7 0.371  0.371 0.371  0 0.371 0.371                   0.282                0.718
```

Model Comparison

- We *cannot* easily compare this IRT model to the CFA that we fit to the same data:
  - The CFA treats the data as continuous; the model likelihood involves a normal density.
  - The IRT likelihood involves probabilities of discrete categories.
  - Densities are not directly comparable to probabilities.

Summary

- We estimated our Bayesian IRT model as a factor analysis of discrete data.
  - We could call it either item factor analysis or item response model: estimation methods sometimes differ for frequentist models, but they are the same thing in Bayesian modeling.
  - Parameters remain similar to those from CFA of continuous data. We are now predicting probit(P(correct)), whereas the CFA was predicting continuous data that only assumed values of 0 and 1.
  - We saw some model checks and comparisons that are especially flexible in *blavaan*, because we can use the R universe to define functions for posterior checks.

Case 3: Explanatory IRT

Explanatory IRT

- The previous section covered a traditional Bayesian IRT model. That model has many uses, including:
  - Item selection
  - Adaptive testing
  - Scoring/estimating person parameters

Explanatory IRT

- The previous IRT model might be unsatisfying if we want to understand *why* people responded in the way that they did:
  - Why is item 1 more difficult than item 2?
  - Why is person 1 more proficient than person 2?
- We can start to address these questions by including extra covariates in the model. Then we might call our model an *explanatory* item response model (see De Boeck & Wilson, 2004).

Explanatory IRT

- Some possibilities:
  - Instead of fixing average person proficiency to 0, use covariates to predict average proficiency.
  - Decompose item difficulties into effects associated with specific item attributes.
  - Person-by-item interactions, getting at differential item functioning.

Explanatory IRT

- Here, we expand on our previous model by adding person covariates: female (0/1) and SES (centered/scaled).
- The model provides information about how the two covariates are predictive of a person's proficiency across the 7 items.
- Our outcome variable (proficiency) is unobserved. Bayesian methods can help us characterize uncertainty in the estimated relationships between observed covariates and unobserved outcome.

Model

- From a *blavaan* point of view, including covariates is a simple addition to the previous model syntax.

```
m3 <- ' f1 =~ M192Q01 + M406Q01 + M423Q01 + M496Q01 + M564Q01 + M571Q01 + M603Q01
        f1 ~ female + hisei + female:hisei '
```

Model

- We now use `bsem()` because the model includes regressions involving latent variables (though `bcfa()` still works!).
- Also, `fixed.x = TRUE` treats our covariates as fixed (as in traditional regression).
- Also, `save.lvs = TRUE` will facilitate model checking.

Model

```
m3fit <- bsem(m3, data = dat, std.lv = TRUE, ordered = TRUE, dp = mypriors,
              fixed.x = TRUE, save.lvs = TRUE)
```

```
##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.002825 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 28.25 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  501 / 1500 [ 33%]  (Sampling)
## Chain 1: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 1: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 1: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 1: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 1: Iteration: 1250 / 1500 [ 83%]  (Sampling)
## Chain 1: Iteration: 1400 / 1500 [ 93%]  (Sampling)
## Chain 1: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 66.111 seconds (Warm-up)
```

Results I

```
summary(m3fit)

## blavaan 0.5.3.1253 ended normally after 1000 iterations
##
##   Estimator                                       BAYES
##   Optimization method                              MCMC
##   Number of model parameters                         17
##
##   Number of observations                            565
##
##   Statistic                                 MargLogLik         PPP
##   Value                                             NA       0.061
##
## Parameter Estimates:
##
##   Parameterization                               Theta
##
## Latent Variables:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   f1 =~
##     M192Q01           0.758    0.092    0.585    0.943    1.005 normal(.4, .2)
##     M406Q01           0.747    0.092    0.577    0.933    1.000 normal(.4, .2)
##     M423Q01           0.312    0.067    0.183    0.446    1.003 normal(.4, .2)
##     M496Q01           0.629    0.083    0.477    0.796    1.007 normal(.4, .2)
##     M564Q01           0.424    0.071    0.287    0.564    1.000 normal(.4, .2)
##     M571Q01           0.685    0.086    0.521    0.858    1.002 normal(.4, .2)
##     M603Q01           0.647    0.082    0.495    0.816    1.002 normal(.4, .2)
##
## Regressions:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
```

Results II

```
##   f1 ~
##     female           -0.202    0.107   -0.412    0.014    1.002   normal(0,10)
##     hisei             0.442    0.080    0.288    0.602    0.999   normal(0,10)
##     female:hisei     -0.199    0.111   -0.421    0.013    0.999   normal(0,10)
##
## Intercepts:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .M192Q01            0.000
##    .M406Q01            0.000
##    .M423Q01            0.000
##    .M496Q01            0.000
##    .M564Q01            0.000
##    .M571Q01            0.000
##    .M603Q01            0.000
##    .f1                 0.000
##
## Thresholds:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##     M192Q01|t1         0.089    0.075   -0.060    0.235    1.002 normal(0, .28)
##     M406Q01|t1         0.139    0.074   -0.000    0.289    1.003 normal(0, .28)
##     M423Q01|t1        -0.670    0.063   -0.794   -0.550    1.001 normal(0, .28)
##     M496Q01|t1        -0.199    0.070   -0.342   -0.066    1.001 normal(0, .28)
##     M564Q01|t1        -0.069    0.060   -0.186    0.046    1.000 normal(0, .28)
##     M571Q01|t1        -0.181    0.071   -0.323   -0.042    1.002 normal(0, .28)
##     M603Q01|t1        -0.206    0.071   -0.350   -0.069    0.999 normal(0, .28)
##
## Variances:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .M192Q01            1.000
##    .M406Q01            1.000
```

Results III

```
##    .M423Q01          1.000
##    .M496Q01          1.000
##    .M564Q01          1.000
##    .M571Q01          1.000
##    .M603Q01          1.000
##    .f1               1.000
##
## Scales y*:
##                   Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##     M192Q01          0.776
##     M406Q01          0.781
##     M423Q01          0.948
##     M496Q01          0.829
##     M564Q01          0.911
##     M571Q01          0.806
##     M603Q01          0.822
```

Results

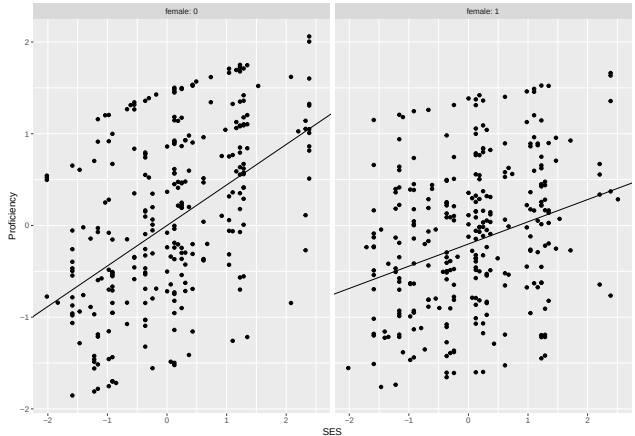
- The regression estimates imply a negative association between female and proficiency, and a positive association between SES and proficiency.
- The posterior interval for the interaction overlaps with 0 but is mostly negative.
- The model is estimating two regression lines, one for female and one for non-female:
  - Non-female: intercept 0, slope 0.442
  - Female: intercept -0.202, slope $0.442 + (-0.199)$
  - (these are posterior means, there is also uncertainty in the estimates!)

Results I

- Graph of posterior mean regression lines, and posterior mean latent variables:

```
lvmeans <- blavInspect(m3fit, 'lvmeans')
dat$f1pred <- lvmeans[, 1]

regwts <- coef(m3fit)[grep("^f1~", names(coef(m3fit)))]
regdf <- cbind.data.frame(female = c(0, 1), int = c(0, regwts[1]), slp = c(regwts[2], sum(regwts[2:3])))

ggplot(dat, aes(x = hisei, y = f1pred)) + geom_point() + geom_abline(data = regdf, aes(slope = slp, intercept = int)) +
  facet_wrap( ~ female, labeller = label_both) + xlab("SES") + ylab("Proficiency")
```

Results II

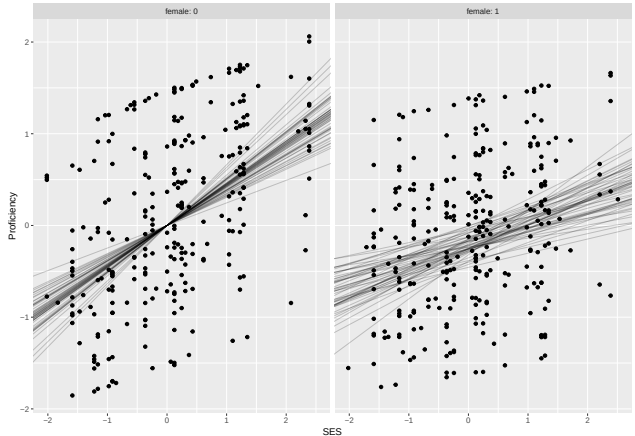


Results I

- Same graph, but show regression lines from 50 posterior samples:

```
p <- ggplot(dat, aes(x = hisei, y = f1pred)) + geom_point() + facet_wrap( ~ female, labeller = label_both) +
  xlab("SES") + ylab("Proficiency")

samps <- do.call("rbind", blavInspect(m3fit, 'mcmc'))
ndraws <- 50
regdf <- cbind.data.frame(female = rep(c(0, 1), ndraws), int = rep(0, ndraws * 2), slp = rep(0, ndraws * 2))
draws <- sample(1:nrow(samps), ndraws)
for (i in 1:length(draws)) {
  regwts <- samps[draws[i], grep("^f1~", colnames(samps))]
  regdf$int[i * 2] <- regwts[1]
  regdf$slp[(i - 1)*2 + 1:2] <- c(regwts[2], sum(regwts[2:3]))
}

p + geom_abline(data = regdf, aes(slope = slp, intercept = int), alpha=.2)
```

Results II



Results

- The previous graph shows an odd characteristic: the line for non-female is constrained to go through (0,0)
  - This is the intercept for non-females.
  - This intercept is fixed to 0, because it identifies the latent variable.
  - This does not cause as many problems as one may expect, because the 0 point on the latent variable is arbitrary.
  - But if we "center" female, we can obtain visualizations that are less unusual.

```
dat$female <- dat$female - .5
```

Model

- Re-estimating the model:

```
m4fit <- bsem(m3, data = dat, std.lv = TRUE, ordered = TRUE, dp = mypriors,
              fixed.x = TRUE, save.lvs = TRUE)

##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.002033 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 20.33 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  501 / 1500 [ 33%]  (Sampling)
## Chain 1: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 1: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 1: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 1: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 1: Iteration: 1250 / 1500 [ 83%]  (Sampling)
## Chain 1: Iteration: 1400 / 1500 [ 93%]  (Sampling)
## Chain 1: Iteration: 1500 / 1500 [100%]  (Sampling)
```

Results I

```
summary(m4fit)

## blavaan 0.5.3.1253 ended normally after 1000 iterations
##
##   Estimator                                         BAYES
##   Optimization method                               MCMC
##   Number of model parameters                          17
##
##   Number of observations                             565
##
##   Statistic                                 MargLogLik         PPP
##   Value                                             NA       0.058
##
## Parameter Estimates:
##
##   Parameterization                               Theta
##
## Latent Variables:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   f1 =~
##     M192Q01           0.768    0.097    0.594    0.974    1.011 normal(.4, .2)
##     M406Q01           0.750    0.091    0.577    0.940    1.003 normal(.4, .2)
##     M423Q01           0.314    0.067    0.183    0.448    1.003 normal(.4, .2)
##     M496Q01           0.626    0.081    0.473    0.787    1.006 normal(.4, .2)
##     M564Q01           0.427    0.067    0.296    0.559    1.000 normal(.4, .2)
##     M571Q01           0.695    0.090    0.527    0.882    1.001 normal(.4, .2)
##     M603Q01           0.652    0.082    0.493    0.819    1.001 normal(.4, .2)
##
## Regressions:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
```

Results II

```
##   f1 ~
##     female           -0.241    0.114   -0.458   -0.019    1.001   normal(0,10)
##     hisei             0.339    0.055    0.229    0.447    0.999   normal(0,10)
##     female:hisei     -0.199    0.108   -0.409    0.014    1.000   normal(0,10)
##
## Intercepts:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .M192Q01           0.000
##    .M406Q01           0.000
##    .M423Q01           0.000
##    .M496Q01           0.000
##    .M564Q01           0.000
##    .M571Q01           0.000
##    .M603Q01           0.000
##    .f1                0.000
##
## Thresholds:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##     M192Q01|t1        0.156    0.067    0.026    0.287    1.000 normal(0, .28)
##     M406Q01|t1        0.207    0.067    0.078    0.342    1.000 normal(0, .28)
##     M423Q01|t1       -0.645    0.060   -0.763   -0.532    1.003 normal(0, .28)
##     M496Q01|t1       -0.143    0.061   -0.262   -0.020    1.000 normal(0, .28)
##     M564Q01|t1       -0.030    0.056   -0.140    0.082    1.000 normal(0, .28)
##     M571Q01|t1       -0.126    0.064   -0.245   -0.001    1.000 normal(0, .28)
##     M603Q01|t1       -0.149    0.061   -0.267   -0.031    1.000 normal(0, .28)
##
## Variances:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .M192Q01           1.000
##    .M406Q01           1.000
```

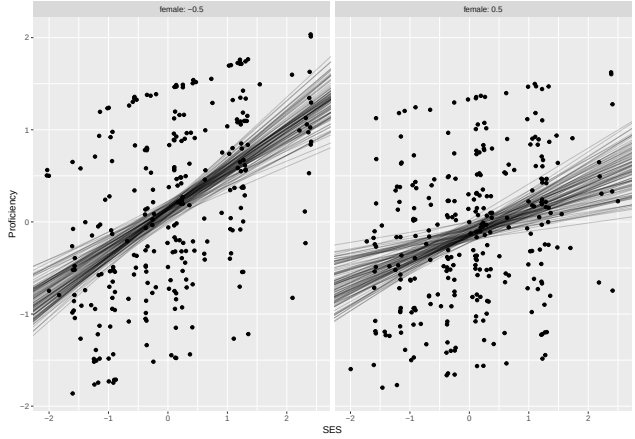
Results III

```
##    .M423Q01          1.000
##    .M496Q01          1.000
##    .M564Q01          1.000
##    .M571Q01          1.000
##    .M603Q01          1.000
##    .f1               1.000
##
## Scales y*:
##                   Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##     M192Q01          0.772
##     M406Q01          0.779
##     M423Q01          0.948
##     M496Q01          0.830
##     M564Q01          0.909
##     M571Q01          0.802
##     M603Q01          0.820
```

Results I

```
lvmeans <- blavInspect(m4fit, 'lvmeans')
dat$f1pred <- lvmeans[, 1]

p <- ggplot(dat, aes(x = hisei, y = f1pred)) + geom_jitter() + facet_wrap( ~ female, labeller = label_both) +
  xlab("SES") + ylab("Proficiency")

samps <- do.call("rbind", blavInspect(m4fit, 'mcmc'))
ndraws <- 100
regdf <- cbind.data.frame(female = rep(c(-.5, .5), ndraws), int = rep(0, ndraws * 2), slp = rep(0, ndraws * 2))
draws <- sample(1:nrow(samps), ndraws)
for (i in 1:length(draws)) {
  regwts <- samps[draws[i], grep("^f1~", colnames(samps))]
  regdf$int[(i - 1)*2 + 1:2] <- regwts[1] * c(-.5, .5)
  regdf$slp[(i - 1)*2 + 1:2] <- regwts[2] + regwts[3] * c(-.5, .5)
}

p + geom_abline(data = regdf, aes(slope = slp, intercept = int), alpha=.2)
```

Results II



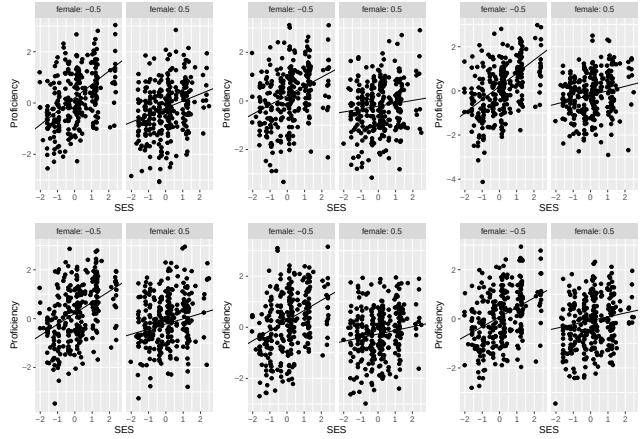
Results

- The previous graph represents uncertainty in the model's regression parameters.
- But there is also uncertainty in the latent variables! This uncertainty is more difficult to capture visually.
- One possibility: separate scatterplots for each posterior sample (which includes latent variables).

Results I

```
## each panel represents one posterior sample
library("patchwork")

lvs <- do.call("rbind", blavInspect(m4fit, 'lvs'))
ndraws <- 6
draws <- sample(1:nrow(samps), ndraws)

ps <- vector("list", length(draws))

for (i in 1:ndraws) {
  dat$f1pred <- lvs[draws[i], ]

  regwts <- samps[draws[i], grep("^f1~", colnames(samps))]
  regdf <- cbind.data.frame(female = c(-.5, .5), int = regwts[1] * c(-.5, .5), slp = regwts[2] + regwts[3] * c(-.5, .5))

  ps[[i]] <- ggplot(dat, aes(x = hisei, y = f1pred)) + geom_jitter() +
    geom_abline(data = regdf, aes(slope = slp, intercept = int)) +
    facet_wrap( ~ female, labeller = label_both) +
    xlab("SES") + ylab("Proficiency")
}

Reduce("+", ps)
```

Results II



Summary

- The *blavaan* framework facilitates extension of traditional IRT models to explanatory IRT models.
- The graphical capabilities of R help us find modeling problems/issues that may be difficult or impossible to find by staring at text output.
- In turn, these attributes help us to best characterize the implications of the model for the observed data.

Case 4: Multilevel SEM

Multilevel SEM

- Goals of this section:
  - Consider two-level SEM as a model of data, vs a model of a covariance matrix
  - Consider model checking in light of the data model
  - See how it works with real data

Basics

Multilevel SEM

- Why use two-level SEM?
  - We are doing SEM, but observations (people) are nested within a grouping variable.
  - For example, many variables are collected on each student, and students are nested within schools.
  - Now we can separate covariance between schools from covariance within schools, and have a model on each.

Two-level SEM

- Two-level SEM comes in at least two flavors, "random intercept" and "random slope".
- The terms "random intercept" and "random slope" are confusing (to me) because you (= I) would expect them to be the same as univariate multilevel models. But they are generally different.
- Also, the term "two-level" is not necessarily what you would expect. Multilevel people might call it "three-level" (multiple responses within person within school).

Two-level SEM

- Let y_{ijk} be the score of person i in school k on variable j.

- It is easy to see the random intercept when you write the model like this:

$$\begin{aligned} \boldsymbol{y}_{ik} &= \boldsymbol{\nu}_k + \boldsymbol{\Lambda}\boldsymbol{\eta}_{ik} + \boldsymbol{e}_{ik} \\ \boldsymbol{\eta}_{ik} &= \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{\eta}_{ik} + \boldsymbol{\epsilon}_{ik} \\ \boldsymbol{\nu}_k &= \boldsymbol{\nu}_0 + \boldsymbol{u}_k \end{aligned}$$

- We can view $\boldsymbol{u}_k$ as a vector of random effects for school k. Each observed variable j will have a separate random effect for each school k. We can jointly model the collection of random efects for school k.

Two-level SEM

- Let $\boldsymbol{u}_k$ be the vector of intercept random effects for school k.
- The easiest thing is to do is model it as an unrestricted multivariate normal.
- This can be problematic when there are many observed variables. For example, if there are 8 observed variables, then unrestricted MVN adds 36 covariance parameters to estimate.

Two-level SEM

- Instead of unrestricted multivariate normal, It is common to model $\boldsymbol{u}_k$ using a second structural equation model of schools.

- Now we have school latent variables influencing the value of the school's intercept. It is another full model on top of what we already had.

$$\boldsymbol{u}_k = \boldsymbol{\nu}^c + \boldsymbol{\Lambda}^c \underline{\underline{\boldsymbol{\eta}}}_k + \boldsymbol{e}_k^c$$
$$\underline{\underline{\boldsymbol{\eta}}}_k = \boldsymbol{\alpha}^c + \boldsymbol{B}^c \underline{\underline{\boldsymbol{\eta}}}_k + \boldsymbol{\epsilon}_k^c$$

Two-level SEM

- Altogether

$$\boldsymbol{y}_{ik} = \boldsymbol{\nu}_k + \boldsymbol{\Lambda}\,\text{👤}_i + \boldsymbol{e}_i$$
$$\text{👤}_i = \boldsymbol{\alpha} + \boldsymbol{B}\,\text{👤}_i + \boldsymbol{\epsilon}_i$$

$$\boldsymbol{\nu}_k = \boldsymbol{\nu}_0 + \boldsymbol{u}_k$$
$$\boldsymbol{u}_k = \boldsymbol{\nu}^c + \boldsymbol{\Lambda}^c\,\text{🏛}_k + \boldsymbol{e}^c_k$$
$$\text{🏛}_k = \boldsymbol{\alpha}^c + \boldsymbol{B}^c\,\text{🏛}_k + \boldsymbol{\epsilon}^c_k$$

Flavors

- For random intercept models, the school-level model neatly separates from the student-level model.
- For random slope models, we get normal random variables multiplied by other normal random variables, which makes the likelihood evaluation difficult. This leads to fewer software options for reliably estimating the models.

Estimation

- The frequentist version of the model integrates out latent variables, as in the one-level case.
- But this can be problematic because, once you integrate out latent variables, all observed variables in a school are correlated.
- So you end up with a multivariate normal likelihood of high dimension.

Estimation

- Brief example: Say 20 students are observed in each of 200 schools. Each student is measured on 4 variables.
- So across all 20 students in a school, we have 80 observed variables.
- If we integrate out latent variables, our model likelihood involves an 80-dimensional multivariate normal. If your model forces you to obtain inverses and determinants of 80×80 matrices, you do not have a happy life.

Estimation

- This problem was addressed since at least the late 80s (e.g., McDonald & Goldstein, 1989 and others). For this example, we can re-express the 80-dimensional likelihood in a way that involves only 4×4 matrices.

- Idea: Instead of raw data, compute many descriptive statistics, including sample means within each school, sample means across schools, covariances within each school, covariances between schools, number of students within each school. The likelihood can be written as a function of all these things.

- A nice, detailed discussion of these results is in Rosseel (2021).

Example

Example

- PISA 2003 data, student and school predictors of math achievement.
  - 9,729 students in 359 schools
  - Student variables: enjoyment of math (4 indicators), math achievement, perception of teacher support
  - School variable: teacher enthusiasm (3 indicators)
  - The data were used by Rockwood (2020), https://osf.io/pxz5s/

Model

- *blavaan* model specification:

```
m5 <- '
    level: within
        fw =~ enjoy1 + enjoy2 + enjoy3 + enjoy4
        math ~ fw + support

    level: between
        fenth =~ enth1 + enth2 + enth3
        fenj =~ enjoy1 + enjoy2 + enjoy3 + enjoy4
        math ~ fenth + fenj + support
'
```

Model

- What does it mean?
  - Enjoyment of math is latent at the student and school level. The school has an influence on all students' math enjoyments, separately from an individual student's enjoyment.
  - Teacher enthusiasm variables are measured once per school. So the latent variable is at the school level.
  - Student perception of support is split into a student component and school component. "latent mean centering"
  - We regress latent variables, along with student perception of support, on math achievement.

Priors

- Once again, we seek mildly informative prior distributions. It helps to know more about the variables:
  - `math`: generally assumes values between 0 and 10
  - `enjoy`, `enth` variables: ratings from 1 to 4
  - `support`: scaled and centered around 0

Priors

- We will identify the model by fixing one loading per latent variable to 1, and we expect all loadings to be positive. So the prior for free loadings will be Normal(1, .5).
- Based on our variables, it would be surprising if any variances were much higher than 1. So Gamma(.5, .5) on SDs.
- Also based on our variables, it would be surprising if regression weights were much larger than 2. And we want to regularize towards 0. So Normal(0, 1).

```
mypris <- dpriors(lambda = "normal(1, .5)", theta = "gamma(.5, .5)[sd]",
                  psi = "gamma(.5, .5)[sd]", beta = "normal(0, 1)")
```

Estimation

- Estimation is similar to before, with addition of the `cluster` argument:

```
m5fit <- bsem(m5, data = dat, dp = mypris, cluster = "school", save.lvs = TRUE)
```

```
## blavaan NOTE: two-level models are new, please report bugs!
## https://github.com/ecmerkle/blavaan/issues
##
##
## SAMPLING FOR MODEL 'stanmarg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.003565 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 35.65 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  501 / 1500 [ 33%]  (Sampling)
## Chain 1: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 1: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 1: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 1: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 1: Iteration: 1250 / 1500 [ 83%]  (Sampling)
```

Results I

```
summary(m5fit)

## blavaan 0.5.3.1253 ended normally after 1000 iterations
##
##   Estimator                                      BAYES
##   Optimization method                             MCMC
##   Number of model parameters                        38
##
##   Number of observations                          9729
##   Number of clusters [school]                      359
##
##   Statistic                                 MargLogLik         PPP
##   Value                                     -70123.647       0.348
##
## Parameter Estimates:
##
##
##
## Level 1 [within]:
##
## Latent Variables:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   fw =~
##     enjoy1            1.000
##     enjoy2            0.900    0.011    0.879    0.921    1.000     normal(1, .5)
##     enjoy3            1.162    0.013    1.137    1.186    1.001     normal(1, .5)
##     enjoy4            0.871    0.012    0.849    0.894    1.000     normal(1, .5)
##
## Regressions:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
```

Results II

```
##   math ~
##     fw                0.665    0.025    0.616    0.712    0.999      normal(0, 1)
##     support          -0.102    0.016   -0.134   -0.070    1.000      normal(0, 1)
##
## Intercepts:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .enjoy1             0.000
##    .enjoy2             0.000
##    .enjoy3             0.000
##    .enjoy4             0.000
##    .math               0.000
##     fw                 0.000
##
## Variances:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .enjoy1             0.252    0.005    0.242    0.261    1.000 gamma(.5, .5)[sd]
##    .enjoy2             0.213    0.004    0.206    0.221    0.999 gamma(.5, .5)[sd]
##    .enjoy3             0.156    0.004    0.147    0.164    1.000 gamma(.5, .5)[sd]
##    .enjoy4             0.315    0.005    0.304    0.325    1.000 gamma(.5, .5)[sd]
##    .math               2.168    0.032    2.105    2.232    1.000 gamma(.5, .5)[sd]
##     fw                 0.465    0.010    0.444    0.486    1.000 gamma(.5, .5)[sd]
##
##
## Level 2 [school]:
##
## Latent Variables:
##                     Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   fenth =~
##     enth1               1.000
##     enth2               0.983    0.114    0.778    1.232    1.001     normal(1, .5)
```

Results III

```
##     enth3             0.624    0.085    0.460    0.793    1.001     normal(1, .5)
##   fenj =~
##     enjoy1            1.000
##     enjoy2            0.862    0.088    0.690    1.040    1.000     normal(1, .5)
##     enjoy3            1.085    0.095    0.909    1.300    1.002     normal(1, .5)
##     enjoy4            0.946    0.100    0.761    1.150    1.000     normal(1, .5)
##
## Regressions:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   math ~
##     fenth             0.374    0.113    0.159    0.608    1.001      normal(0, 1)
##     fenj              1.777    0.452    0.909    2.689    1.002      normal(0, 1)
##     support          -0.516    0.131   -0.766   -0.267    1.000      normal(0, 1)
##
## Covariances:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##   fenth ~~
##     fenj              0.006    0.005   -0.004    0.016    0.999         beta(1,1)
##
## Intercepts:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .enth1             2.931    0.029    2.874    2.988    1.000      normal(0,32)
##    .enth2             3.093    0.027    3.040    3.145    1.000      normal(0,32)
##    .enth3             3.134    0.028    3.078    3.187    1.000      normal(0,32)
##    .enjoy1            2.061    0.012    2.039    2.085    1.002      normal(0,32)
##    .enjoy2            1.873    0.011    1.852    1.894    1.003      normal(0,32)
##    .enjoy3            2.172    0.012    2.148    2.196    1.002      normal(0,32)
##    .enjoy4            2.557    0.012    2.534    2.580    1.002      normal(0,32)
##    .math              4.868    0.043    4.781    4.951    1.000      normal(0,10)
##     fenth             0.000
```

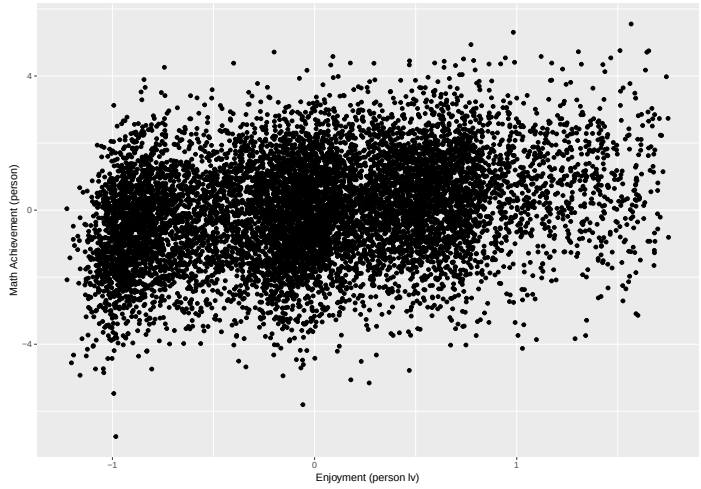
Results IV

```
##     fenj              0.000
##
## Variances:
##                    Estimate  Post.SD pi.lower pi.upper     Rhat    Prior
##    .enth1             0.126    0.020    0.086    0.164    1.003 gamma(.5, .5)[sd]
##    .enth2             0.082    0.018    0.045    0.117    1.002 gamma(.5, .5)[sd]
##    .enth3             0.228    0.018    0.195    0.264    1.000 gamma(.5, .5)[sd]
##    .enjoy1            0.001    0.001    0.000    0.004    1.000 gamma(.5, .5)[sd]
##    .enjoy2            0.004    0.001    0.002    0.006    1.000 gamma(.5, .5)[sd]
##    .enjoy3            0.003    0.001    0.000    0.005    1.007 gamma(.5, .5)[sd]
##    .enjoy4            0.007    0.002    0.004    0.010    1.000 gamma(.5, .5)[sd]
##    .math              0.413    0.044    0.332    0.507    1.001 gamma(.5, .5)[sd]
##     fenth             0.169    0.027    0.120    0.226    1.003 gamma(.5, .5)[sd]
##     fenj              0.019    0.004    0.012    0.026    1.000 gamma(.5, .5)[sd]
```

Within Relationship I

```
## Visualize person-level relationship between enjoyment and achievement:
lvmeans <- blavInspect(m5fit, "lvmeans")
dat$lvmeans <- lvmeans[, 1]

schmath <- with(dat, tapply(math, school, mean))
dat$schmath <- schmath[match( dat$school, as.numeric(names(schmath)) )]
dat$math_within <- dat$math - dat$schmath

ggplot(dat, aes(x = lvmeans, y = math_within)) + geom_point() + xlab("Enjoyment (person lv)") +
  ylab("Math Achievement (person)")
```

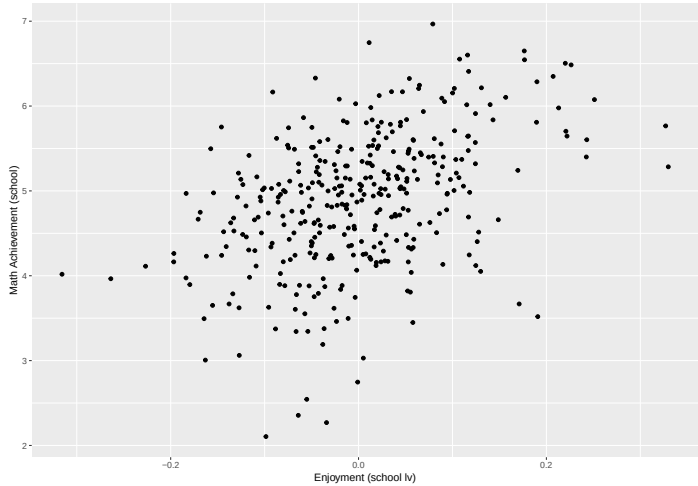
Within Relationship II



Between Relationship I

```
## Visualize school-level relationship betwen enjoyment and achievement:
lv2 <- blavInspect(m5fit, "lvmeans", level = 2)

## ensure the school means are ordered in the same way that blavaan orders clusters
schmath <- schmath[ match(blavInspect(m5fit, 'cluster.id'), as.numeric(names(schmath))) ]

l2dat <- cbind.data.frame(schmath, lv2)
ggplot(l2dat, aes(x = fenj, y = schmath)) + geom_point() + xlab("Enjoyment (school lv)") +
  ylab("Math Achievement (school)")
```

Between Relationship II



Summary

- Our modeling strategies transferred over to two-level SEM, with few changes.
- If we are careful, we can use our posterior samples to visualize results that are typically only summarized in tables.
- Checks and summaries from earlier examples are also applicable!

General Recommendations & Commentary

Workflow

- Some general workflow recommendations:
  - Use frequentist models as quick checks of syntax/model ideas.
  - Consider your prior distributions, and do prior predictive checks.
  - Arguments `burnin = 100, sample = 100` is sufficient for rough results of many *blavaan* models (`target = "stan"` only).
  - Consider meaningful summaries of how the posterior distribution corresponds to the observed data, based on the goals of your model.

Workflow

- Related to the previous slide: tailor models and summaries to your substantive goals.
  - Use traditional SEM as a lauching point for tailored models and tailored model checks.
  - Posterior samples allow us to summarize uncertainty in most quantities of interest (e.g., item-total correlations).
  - Recipes can be helpful, but also limiting. Some coding skills can take you far.

Limitations

- The examples presented here "just worked." We continue to improve *blavaan* to "just work" on more datasets. Some models/datasets currently won't work so well, including:
  - Observed variables that assume large values, with no consideration of priors (lack of model convergence).
  - Large amounts of missing data, large numbers of observed variables (slow).
  - Inclusion of reverse-coded items: must carefully consider whether priors on loadings match sign constraints on loadings.

Advanced Topics

Advanced Topics

- Speeding up Stan
- Extending *blavaan* models
- Future developments

Speed

- MCMC can be very slow. For different types of MCMC, the speed issue manifests itself differently.
  - Gibbs, Metropolis Hastings: Need to draw a huge number of samples due to autocorrelation of posterior samples.
  - Hamiltonian: Fewer samples are often needed (less autocorrelation), but gradient computations (autodiff) also required.

Speed

- Focusing on Stan and Hamiltonian Monte Carlo, we can see speed and efficiency improvements by reducing the number of model parameters.
- This is especially relevant for SEM:
  - In traditional MCMC estimation, the latent variables are model parameters. This leads to conditional independence and univariate likelihoods, as opposed to multivariate likelihoods.
  - But every person in the data has unique latent variables, so we are adding hundreds of model parameters. This can be burdensome.

Speed

- This means we can speed up Stan by specifying models that look similar to frequentist models:
  - Use of sufficient statistics: instead of looping over every person, evaluate one likelihood that involves sample mean and sample covariance matrix.
  - Likelihoods that are marginal over latent variables.
  - For sampling latent variables: derive conditional distribution of latents given observed, sample in generated quantities block.

Limitations

- Limitation to this approach: difficulty in extending the model.
  - For example, if our latent variables are not multivariate normal, then we cannot easily marginalize over the latent variables.
  - Also, non-normal observed variables are not easily incorporated.
  - So there is a tradeoff between fast estimation of traditional models, vs easy extension to non-traditional models.

Extensions

- But it is possible to extend *blavaan* models:
  - Use `mcmcfile = TRUE` to export the Stan (or JAGS) model and data.
  - Easy Stan changes include prior distributions, and parameter constraints.
  - For other changes, consider `target = "jags"`. The model is tailored to your requested model and samples latent variables.

Future Developments

- Near term additions: Structural after measurement approach, refining two-level SEM.
- Longer term: Two-level ordinal, random slopes.
- General principle: No new features that cannot be easily tested. (i.e., I don't want to wait hours to see whether a model worked)

Questions

- Questions/comments

Thank you!

In R:

```
install.packages("blavaan")
```

blavaan website:

https://ecmerkle.github.io/blavaan/

References I

Bonifay, W., & Depaoli, S. (2021). Model evaluation in the presence of categorical data: Bayesian model checking as an alternative to traditional methods. *Prevention Science*, *24*(3), 467–479. http://doi.org/10.1007/s11121-021-01293-w

De Boeck, P., & Wilson, M. (2004). *Explanatory item response models: A generalized linear and nonlinear approach*. New York: Springer-Verlag.

Garnier-Villareal, M., & Jorgensen, T. D. (2020). Adapting fit indices for Bayesian structural equation modeling: Comparison to maximum likelihood. *Psychological Methods*, *25*, 46–70.

McDonald, R. P. (1999). *Test theory: A unified treatment*. Mahwah, NJ: Erlbaum.

McDonald, R. P., & Goldstein, H. (1989). Balanced versus unbalanced designs for linear structural relations in two level data. *British Journal of Mathematical and Statistical Psychology*, *42*, 215–232.

Merkle, Edgar C., Ariyo, O., Winter, S. D., & Garnier-Villarreal, M. (2023). Opaque prior distributions in bayesian latent variable models. *Methodology*, *19*(3), 228–255. http://doi.org/10.5964/meth.11167

References II

Merkle, E. C., Fitzsimmons, E., Uanhoro, J., & Goodrich, B. (2021). Efficient Bayesian structural equation modeling in Stan. *Journal of Statistical Software*, *100*(6), 1–22. Retrieved from https://doi.org/10.18637/jss.v100.i06

Merkle, E. C., Furr, D., & Rabe-Hesketh, S. (2019). Bayesian comparison of latent variable models: Conditional versus marginal likelihoods. *Psychometrika*, *84*, 802–829.

Merkle, E. C., & Rosseel, Y. (2018). blavaan: Bayesian structural equation models via parameter expansion. *Journal of Statistical Software*, *85*(4), 1–30.

Rockwood, N. J. (2020). Maximum likelihood estimation of multilevel structural equation models with random slopes for latent covariates. *Psychometrika*, *85*(2), 275–300. http://doi.org/10.1007/s11336-020-09702-9

Rosseel, Y. (2021). Evaluating the observed log-likelihood function in two-level structural equation modeling with missing data: From formulas to R code. *Psych*, *3*(2), 197–232. http://doi.org/10.3390/psych3020017