Bayesian Confirmatory Factor Analysis

Chong Xing[1], Zack Roman[1], & Paul Johnson[1]

[1] Center for Research Methods and Data Analysis

2018

Goals of today's sessions

Today's sessions will focus on applying Bayesian framework to SEM analysis

- Reflect and integrate the conceptual knowledge learned from the previous sessions
- Begin to form a probabilistic view about Structural Equation Modeling
- Gain first-hand working experience estimating SEM models under Bayesian framework

Outline

1. Software Options for Bayesian SEM
   - BUGS
   - JAGS
   - Stan
   - Typical software workflow
2. Bayesian CFA Models
   - One-Factor Bayesian CFA
   - The Stan Code for Bayesian CFA
   - Two-Factor Bayesian CFA
   - Three-Factor Bayesian CFA
3. Discussion

Outline

Software options - BUGS

The following non-exhaustive software list shows programs we have used at CRMDA for estimating SEM models under Bayesian framework

BUGS **B**ayesian Inference **U**sing **G**ibbs **S**ampling (Spiegelhalter, Thomas, Best, & Lunn, 2003)

1. The main Bayesian software implementation prior to JAGS and Stan
2. Two variants - WinBUGS and OpenBUGS
3. Early (some current) Bayesian SEM research relies heavily on BUGS (e.g., Song & Lee, 2012)
4. MCMC samplers -
    1. Gibbs sampler (Geman & Geman, 1984)
    2. Metropolis-Hastings sampler (Hastings, 1970; Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953)
5. Software development has stopped; last release version (2010) - WinBUGS 1.4.3

Software options - JAGS

JAGS **J**ust **A**nother **G**ibbs **S**ampler (Plumer, 2003)

1. A cross-platform Bayesian inference software (i.e., Windows, Mac, Linux)
2. Another popular software for SEM research (e.g., Merkle & Wang, 2018)
3. MCMC samplers -
    1. Gibbs sampler
    2. Metropolis-Hastings sampler
    3. Naturally treating missing data (i.e., as random variables)
4. Latest stable release version (2017) - 4.3.0

Software options - Stan

Stan **the current forefront** of Bayesian software development (Stan Development Team, 2017)

1. Led by Andrew Gelman (Gelman et al., 2014; Gelman & Hill, 2007) at the University of Columbia
2. MCMC sampler - NUTS (No-U-Turn Sampler; Hoffman & Gelman, 2014)
3. More demanding on coding/programing skills
4. Used for our CFA and SEM examples

Software workflow

- A model syntax file written in a Bayesian programming language (i.e., BUGS, JAGS, or Stan)
- An execution syntax written within a software environment (R for our demonstration)
  - Points to the location of the model syntax
  - Specify the global values of a model (e.g., number of observations/factors/indicators)
  - Prepare data to be used for model estimation
- R helper packages

  R2OpenBUGS for BUGS
  rjags for JAGS
  rstan for Stan

Outline

The matrix notation for one-factor CFA

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\epsilon}$$

- $\mathbf{y}$ a vector of observed variables/indicators
- $\boldsymbol{\mu}$ a vector of intercepts associated with the indicators
- $\boldsymbol{\Lambda}$ a matrix of factor loadings relating y(observed variables) to ω (latent factors)
- $\boldsymbol{\eta}$ a vector of factor scores per subject/participant
- $\boldsymbol{\epsilon}$ a vector of residual errors

The matrix notation for one-factor CFA ...

The demonstration data set (CILS; Portes & Rumbaut, 2012)

DSDR | DATA SHARING FOR DEMOGRAPHIC RESEARCH

ICPSR 20520

Children of Immigrants Longitudinal Study (CILS), 1991-2006

Alejandro Portes
Princeton University

Rubén G. Rumbaut
University of California-Irvine

Data Collection Instruments

The matrix notation for one-factor CFA ...

The four observed indicators for English proficiency

24- How well do you speak English?

1. Not at all ___	2. Not well ___	3. Well ___	4. Very well ___	V24 ___

25- How well do you understand English?

1. Not at all ___	2. Not well ___	3. Well ___	4. Very well ___	V25 ___

26- How well do you read English?

1. Not at all ___	2. Not well ___	3. Well ___	4. Very well ___	V26 ___

27- How well do you write English?

1. Not at all ___	2. Not well ___	3. Well ___	4. Very well ___	V27 ___

The matrix notation for one-factor CFA ...

```r
## Reading in the demonstration data
ddir <- "../../data"
dat <- readRDS(file.path(ddir, "cils-subset_integer.rds"))
## Listing the variable names
names(dat)
```

```
 [1] "id"           "sex_91"       "age_91"       "citizen_91"   "speakEng_91"  "underEng_91"
 [7] "readEng_91"   "writeEng_91"  "ses_1_91"     "ses_2_91"     "bilingual_91" "speakSec_91"
[13] "underSec_91"  "readSec_91"   "writeSec_91"  "discri_1_91"  "discri_2_91"  "discri_3_91"
[19] "discri_4_91"  "discri_5_91"  "discri_6_91"  "discri_91"    "depre_1_91"   "depre_2_91"
[25] "depre_3_91"   "depre_4_91"   "paren_1_91"   "paren_2_91"   "disc_7_91"    "sex_95"
[31] "citizen_95"   "speakEng_95"  "underEng_95"  "readEng_95"   "writeEng_95"  "ses_1_95"
[37] "ses_2_95"     "bilingual_95" "speakSec_95"  "underSec_95"  "readSec_95"   "writeSec_95"
[43] "discri_1_95"  "discri_2_95"  "discri_3_95"  "discri_4_95"  "discri_5_95"  "discri_6_95"
[49] "discri_95"    "depre_1_95"   "depre_2_95"   "depre_3_95"   "depre_4_95"   "paren_1_95"
[55] "paren_2_95"   "discri_7_95"  "bilingual_01" "speakSec_01"  "underSec_01"  "readSec_01"
[61] "writeSec_01"  "speakEng_01"  "underEng_01"  "readEng_01"   "writeEng_01"  "discri_01"
[67] "citizen_01"
```

```r
## Requesting summary statistics
## For the English Proficiency measures
varName.Eng91 <- c("speakEng_91", "underEng_91",
                   "readEng_91", "writeEng_91")
summary(dat[ , varName.Eng91])
```

The matrix notation for one-factor CFA ...

```
  speakEng_91      underEng_91       readEng_91     writeEng_91
 Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :1.000
 1st Qu.:4.000   1st Qu.:4.000   1st Qu.:3.00   1st Qu.:3.000
 Median :4.000   Median :4.000   Median :4.00   Median :4.000
 Mean   :3.733   Mean   :3.778   Mean   :3.68   Mean   :3.644
 3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.00   3rd Qu.:4.000
 Max.   :4.000   Max.   :4.000   Max.   :4.00   Max.   :4.000
 NA's   :15      NA's   :16      NA's   :16     NA's   :15
```

A one-factor model with four self-reported items assessing children of immigrants' English proficiency

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega} + \boldsymbol{\epsilon}$$

$$\begin{bmatrix} SpeakEng\_91 \\ underEng\_91 \\ readEng\_91 \\ writeEng\_91 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} + \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} + [EngProfi91] + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix}$$

Estimating a one-factor Bayesian CFA model in 3 steps

```r
## Loading the rstan package
library(rstan)

## To install the package
## Uncomment and run the following line
## install.packages("rstan")
```

```r
## Creating the data set for rstan
dat.Eng91.complete <- na.omit(dat[ , varName.Eng91])
```

```r
## Including the model information
## To the data object for rstan
data_for_stan <-
 list(N = nrow(dat.Eng91.complete), ## sample size
 k = ncol(dat.Eng91.complete),       ## number of indicator
 y = as.matrix(dat.Eng91.complete), ## observed responses
 n_xi = 1,                           ## number of factor(s)
 str_loading = c(1, 1, 1, 1))        ## loading structure
```

Estimating a one-factor Bayesian CFA model in 3 steps ...

```
## Step 1 - Using the ‘stanc()‘ function
## To translate Stan code to C++
## The tr_01 object contains C++ translation for cfa-01.stan
## The "cfa-01.stan" is included in the current directory
tr_01 <- stanc("cfa-01.stan")
```

```
## Step 2 - Using the ‘stan_model()‘ function
## To create an S4 class model object
## The so_01 is an S4 class model object created for tr_01
## Warnings will show up - move to the next step
## If the warnings are nonfatal
so_01  <- stan_model(stanc_ret = tr_01, verbose=FALSE)
```

Estimating a one-factor Bayesian CFA model in 3 steps ...

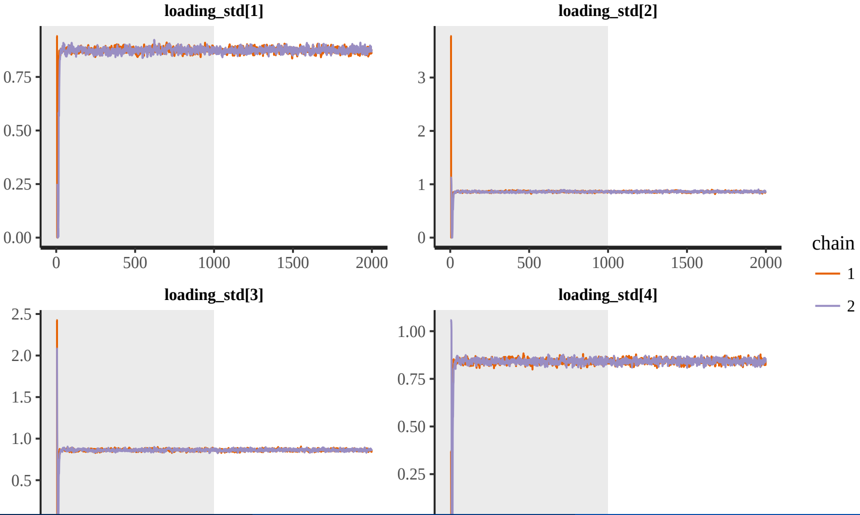
```r
## Step 3 - Using the `sampling()` function
## For MCMC sampling/estimation
post_01 <-
 sampling(object = so_01, ## the S4 model object
 data = data_for_stan,    ## the data input (an list object)
 chains = 2,              ## number of MCMC chains
 iter = 2000,             ## number of MCMC draws per chain
 warmup = 1000)           ## number of warmup draws per chain
## Saving the posterior draws to an .rds file
saveRDS(post_01, file.path(wdir, "post_01.rds"))
```

```r
## Reading the posterior draws back to R
post_01 <- readRDS(file.path(wdir, "post_01.rds"))
```

Summarizing a one-factor Bayesian CFA model results - Factor loadings

```
## Trace plot for visualizing MCMC convergence (loadings)
plot(post_01, plotfun = "trace", pars = c("loading_std"),
    inc_warmup = TRUE)
```

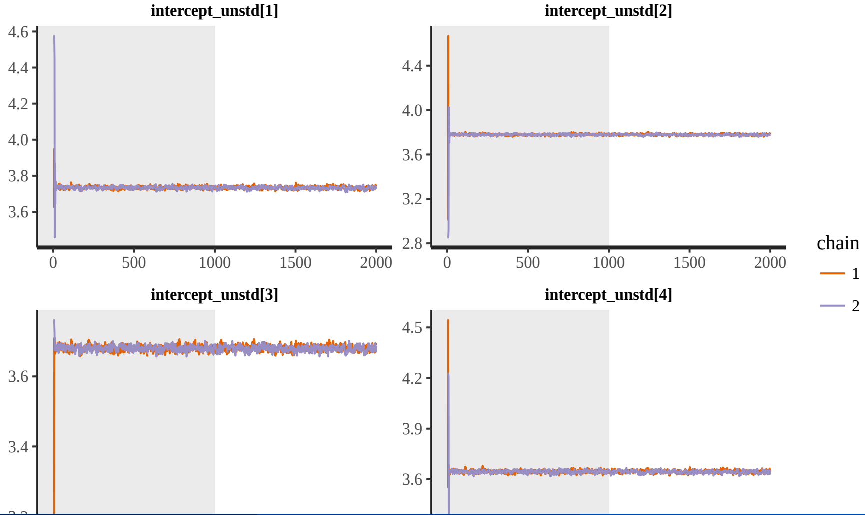
Summarizing a one-factor Bayesian CFA model results - Factor loadings ...



Summarizing a one-factor Bayesian CFA model results - Indicator intercepts

```
## Trace plot for visualizing MCMC convergence
## Of the indicator intercepts
plot(post_01, plotfun = "trace", pars = c("intercept_unstd"),
      inc_warmup = TRUE)
```

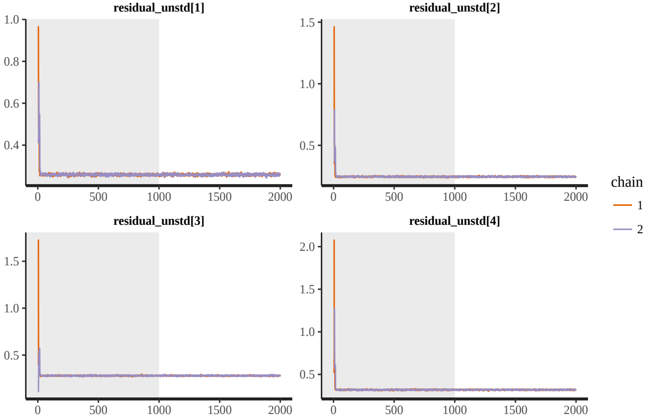
Summarizing a one-factor Bayesian CFA model results - Indicator intercepts ...



Summarizing a one-factor Bayesian CFA model results - Residuals

```
## Trace plot for visualizing MCMC convergence
## Of the indicator residuals (unique variances)
plot(post_01, plotfun = "trace", pars = c("residual_unstd"),
     inc_warmup = TRUE)
```

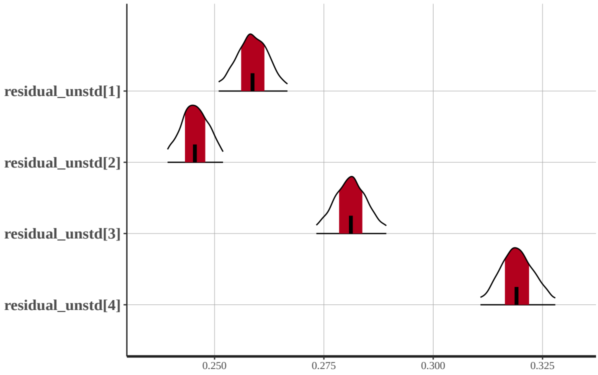
Summarizing a one-factor Bayesian CFA model results - Residuals ...



Summarizing a one-factor Bayesian CFA model results - Residuals ...

```
## Density plot for indicator residuals
plot(post_01, pars = "residual_unstd", show_density = TRUE,
      ci_level = 0.5)
```

Summarizing a one-factor Bayesian CFA model results - Residuals ...



Summarizing a one-factor Bayesian CFA model results - Residuals ...

```
## Descriptive table for summarizing
## The posterior distributions of the intercepts
print(post_01 , pars = "residual_unstd", probs = c(.025 , .975),
      digits = 2, mode = TRUE, use_cache = FALSE)
```

```
Inference for Stan model: cfa-01.
2 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=2000.

                  mean se_mean sd 2.5% 97.5% n_eff Rhat
residual_unstd[1] 0.26       0  0 0.25  0.27  1120    1
residual_unstd[2] 0.25       0  0 0.24  0.25  1124    1
residual_unstd[3] 0.28       0  0 0.27  0.29   951    1
residual_unstd[4] 0.32       0  0 0.31  0.33  1260    1

Samples were drawn using NUTS(diag_e) at Mon Jun  4 18:07:40 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

The Stan code for the CFA example

```
// Stan script for confirmatory factor analysis
// Adapted from Mike Lawrence's cfa.stan code on github
//
    https://gist.github.com/mike-lawrence/dd2435f290a567bd1fd03370ee6
data {
  int N;
    // N: number of subjects

  int k;
    // k: number of observed indicators

  matrix[N,k] y;
    // y: matrix of observed responses

  int n_xi;
    // n_factor: number of latent factor(s)

  int<lower=1,upper=n_xi> str_loading[k];
    // str_loading: factor loading structure }
transformed data {
  matrix[N, k] y_std;
```

The Stan code for the CFA example ...

```
    // y_std: standardized observed responses
  for(i in 1:k) {
    y_std[ , i] = (y[ , i] - mean(y[ , i])) / sd(y[ , i]);
  }
}
parameters {
  matrix[n_xi, N] normal01;
    // normal01 a helper variable for more
    // efficient non-centered multivariate
    // sampling of subj_facs

  cholesky_factor_corr[n_xi] factor_cor_helper;
    // factor_cor_helper: correlations
    // (on Cholesky factor scale) amongst
    // latent factors

  vector[k] intercept_std;
    // intercept_std: the mean for each observed indicator

  vector<lower=0>[k] residual_std;
    // residual_std: the unique factor (error) for each
       indicator
```

The Stan code for the CFA example ...

```
  vector<lower=0>[k] loading_std;
    // loading_std: how each indicator loads on to each factor
    // Must be postive for identifiability
}
transformed parameters {
  matrix[N,n_xi] factor_score;
    // factor_score: matrix of values for each
    // factor associated with each subject

  factor_score = transpose(factor_cor_helper * normal01);
    // the following calculation implies
    // that rows of subj_facs are sampled
    // from a multivariate normal
    // distribution with mean of 0 and
    // sd of 1 in all dimensions and a
    // correlation matrix of fac_cor
}
model {
  to_vector(normal01) ~ normal(0, 1);
    // normal01 must have normal(0,1) prior for
    // "non-centered" parameterization on the
```

The Stan code for the CFA example ...

```
    // multivariate distribution of latent factors

  factor_cor_helper ~ lkj_corr_cholesky(1);
    // flat prior on correlations

  intercept_std ~ normal(0, 1);
    // normal prior on y intercept

  residual_std ~ cauchy(0, 2.5);
    // normal prior on the unique factor for each indicator

  loading_std ~ normal(0, 1);
    // normal(0, 1) priors on factor loadings
  for(i in 1:k) {
    y_std[ , i] ~ normal(intercept_std[i] + factor_score[ ,
        str_loading[i]] * loading_std[i]
                                 , residual_std[i]);
      // each indicator as normal influenced
      // by its associated latent factor
  }
}
generated quantities {
```

The Stan code for the CFA example ...

```
corr_matrix[n_xi] factor_cor;
  // factor_cor: factor correlations

vector[k] intercept_unstd;
  // intercept on the original scale of an indicator

vector[k] residual_unstd;
  // residual on the original scale of an indicator

factor_cor =
    multiply_lower_tri_self_transpose(factor_cor_helper);
for(i in 1:k) {
  intercept_unstd[i] = intercept_std[i] * sd(y[ , i]) +
      mean(y[ , i]);
  residual_unstd[i] = residual_std[i] * sd(y[ , i]);
  }
}
```

Matrix notation for two-factor CFA

Adding a second factor: Second language proficiency

Let's talk about the language that you speak at your home:

49- Do you know a language other than English? 1. Yes___ 2. No___ V49 __|__

50- (If yes) What language is that? (If more than one, please list first the language you know best) V50a __|__

________________________________ V50b __|__

51- How well do you speak that language? (Or the foreign language that you know best)

1. Very little ___	2. Not well ___	3. Well ___	4. Very well ___	V51 __|__

52- How well do you understand that language?

1. Very little ___	2. Not well ___	3. Well ___	4. Very well ___	V52 __|__

53- How well do you read that language?

1. Very little ___	2. Not well ___	3. Well ___	4. Very well ___	V53 __|__

54- How well do you write that language?

1. Very little ___	2. Not well ___	3. Well ___	4. Very well ___	V54 __|__

Matrix notation for two-factor CFA ...

```r
## Requesting summary statistics
## For the second language proficiency measures
varName.Sec91 <-
 c("speakSec_91", "underSec_91", "readSec_91", "writeSec_91")
summary(dat[ , varName.Sec91])
```

```
 speakSec_91     underSec_91      readSec_91     writeSec_91
Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
1st Qu.:3.000   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:2.000
Median :3.000   Median :3.000   Median :3.000   Median :3.000
Mean   :3.096   Mean   :3.352   Mean   :2.645   Mean   :2.477
3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:3.000   3rd Qu.:3.000
Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000
NA's   :438     NA's   :438     NA's   :445     NA's   :449
```

Matrix notation for two-factor CFA ...

A two-factor CFA model relating English proficiency and second language proficiency (**Caution! This is not a causal model**)

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\epsilon}$$

$$\begin{bmatrix} speakEng\_91 \\ underEng\_91 \\ readEng\_91 \\ writeEng\_91 \\ speakSec\_91 \\ underSec\_91 \\ readSec\_91 \\ writeSec\_91 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \end{bmatrix} + \begin{bmatrix} \lambda_{11} & 0 \\ \lambda_{21} & 0 \\ \lambda_{31} & 0 \\ \lambda_{41} & 0 \\ 0 & \lambda_{52} \\ 0 & \lambda_{62} \\ 0 & \lambda_{72} \\ 0 & \lambda_{82} \end{bmatrix} + \begin{bmatrix} EngProfi91 \\ SecProfi91 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ \epsilon_7 \\ \epsilon_8 \end{bmatrix}$$

Estimating a two-factor Bayesian CFA model

```
## Creating the data set for rstan
dat.02 <- na.omit(dat[ , c(varName.Eng91, varName.Sec91)])
```

```
## Including the model information
## To the data object for rstan
data_for_stan_02 <-
 list(N = nrow(dat.02), ## sample size
 k = ncol(dat.02),      ## number of indicator
 y = as.matrix(dat.02), ## observed responses
 n_xi = 2,              ## number of factor(s)
 str_loading = c(1, 1, 1, 1,
                 2, 2, 2, 2)) ## loading structure
```

Estimating a two-factor Bayesian CFA model ...

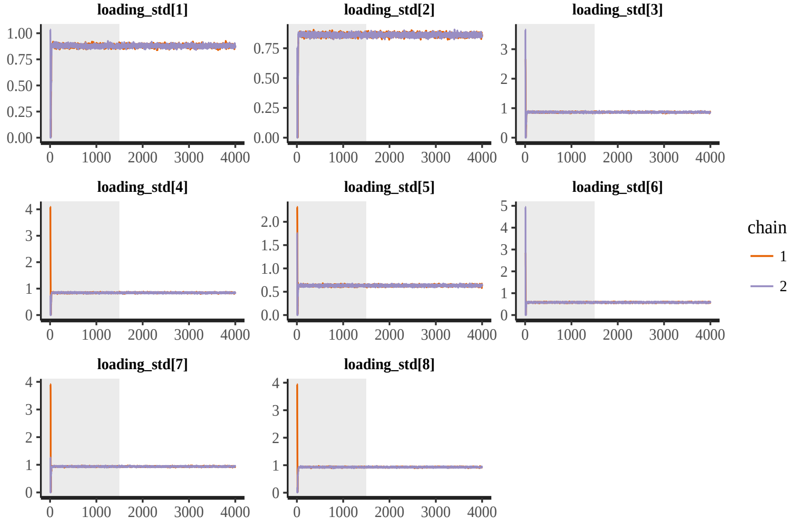
```
## Using the stan() function
## To estimate the model in 1 step
post_02 <-
 stan(file = "cfa-01.stan", ## the Stan syntax file
 data = data_for_stan_02,   ## data input for rstan
 chains = 2,                ## number of MCMC chains
 iter = 4000,               ## total number of MCMC draws per chain
 warmup = 1500)             ## number of warmup draws per chain
## Saving the posterior draws to an .rds file
saveRDS(post_02, file.path(wdir, "post_02.rds"))
```

Summarizing a two-factor Bayesian CFA model results - Factor loadings

```r
## Reading the posterior draws back to R
post_02 <- readRDS(file.path(wdir, "post_02.rds"))
```

```r
## Trace plot for visualizing MCMC convergence
## Factor loadings
plot(post_02, plotfun = "trace", pars = c("loading_std"),
    inc_warmup = TRUE)
```

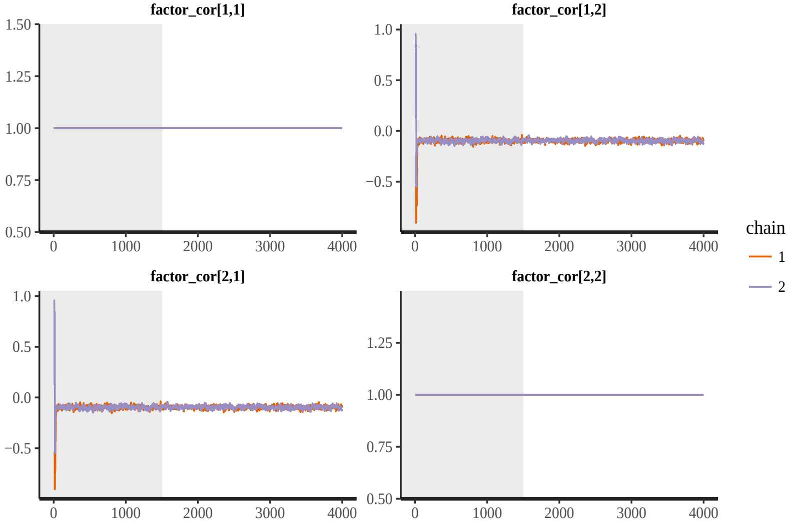
Summarizing a two-factor Bayesian CFA model results - Factor loadings ...



Summarizing a two-factor Bayesian CFA model results - Factor correlation

```
## Trace plot for visualizing MCMC convergence
## Factor correlation
plot(post_02, plotfun = "trace", pars = c("factor_cor"),
    inc_warmup = TRUE)
```

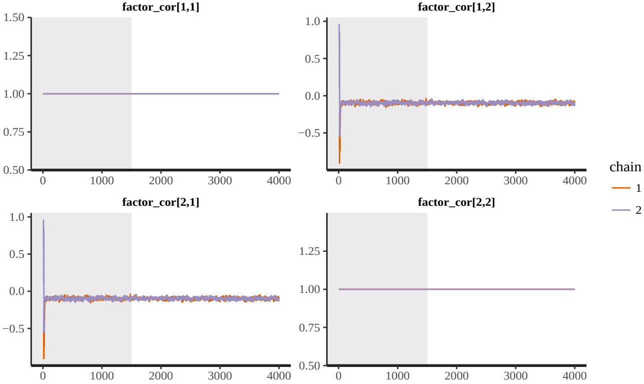
Summarizing a two-factor Bayesian CFA model results - Factor correlation ...



Summarizing a two-factor Bayesian CFA model results - Factor correlation ...

```
## Density plot for factor correlation
plot(post_02, pars = "factor_cor", show_density = TRUE,
    ci_level = 0.5)
```

Summarizing a two-factor Bayesian CFA model results - Factor correlation ... ...



Summarizing a two-factor Bayesian CFA model results - Factor correlation ... ...

```
## Descriptive table for summarizing the posterior
    distribution
print(post_02, pars = "factor_cor", probs = c(.025, .975),
      digits = 2, mode = TRUE, use_cache = FALSE)
```

```
Inference for Stan model: cfa-01.
2 chains, each with iter=4000; warmup=1500; thin=1;
post-warmup draws per chain=2500, total post-warmup draws=5000.

                mean se_mean   sd  2.5% 97.5% n_eff Rhat
factor_cor[1,1]  1.0       0 0.00  1.00  1.00  5000  NaN
factor_cor[1,2] -0.1       0 0.02 -0.13 -0.07   343 1.01
factor_cor[2,1] -0.1       0 0.02 -0.13 -0.07   343 1.01
factor_cor[2,2]  1.0       0 0.00  1.00  1.00  5000 1.00

Samples were drawn using NUTS(diag_e) at Tue Jun  5 10:36:09 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Matrix notation for three-factor CFA

How is self-reported depression related to linguistic competence?

Below is a list of feelings that people sometimes have. For each answer, how often have you felt this way during the past week?

		Rarely (less than once a week)	Some of the time (1 or 2 days a week)	Occasionally (3 or 4 days a week)	Most of the time (5 to 7 days a week)	
114-	I felt sad.	______	______	______	______	V114 _|_
115-	I could not get "going."	______	______	______	______	V115 _|_
116-	I did not feel like eating; my appetite was poor.	______	______	______	______	V116 _|_
117-	I felt depressed.	______	______	______	______	V117 _|_

Matrix notation for three-factor CFA ...

```
## Requesting summary statistics
## For the depression measures
varName.Depre91 <-
 c("depre_1_91", "depre_2_91", "depre_3_91", "depre_4_91")
summary(dat[ , varName.Sec91])
```

```
 speakSec_91     underSec_91      readSec_91      writeSec_91
Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
1st Qu.:3.000   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:2.000
Median :3.000   Median :3.000   Median :3.000   Median :3.000
Mean   :3.096   Mean   :3.352   Mean   :2.645   Mean   :2.477
3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:3.000   3rd Qu.:3.000
Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000
NA's   :438     NA's   :438     NA's   :445     NA's   :449
```

Matrix notation for three-factor CFA ...

A three-factor CFA model relating depression to English and second language proficiency

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\epsilon}$$

$$\begin{bmatrix} speakEng\_91 \\ underEng\_91 \\ readEng\_91 \\ writeEng\_91 \\ speakSec\_91 \\ underSec\_91 \\ readSec\_91 \\ writeSec\_91 \\ depre\_1\_91 \\ depre\_2\_91 \\ depre\_3\_91 \\ depre\_4\_91 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \\ \mu_{10} \\ \mu_{11} \\ \mu_{12} \end{bmatrix} + \begin{bmatrix} \lambda_{11} & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ \lambda_{41} & 0 & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & \lambda_{72} & 0 \\ 0 & \lambda_{82} & 0 \\ 0 & 0 & \lambda_{93} \\ 0 & 0 & \lambda_{10.3} \\ 0 & 0 & \lambda_{11.3} \\ 0 & 0 & \lambda_{12.3} \end{bmatrix} + \begin{bmatrix} Eng91 \\ Sec91 \\ Dep91 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ \epsilon_7 \\ \epsilon_8 \\ \epsilon_9 \\ \epsilon_{10} \\ \epsilon_{11} \\ \epsilon_{12} \end{bmatrix}$$

Estimating a three-factor Bayesian CFA model

```
## Creating the data set for rstan
dat.03 <- na.omit(dat[ , c(varName.Eng91, varName.Sec91,
    varName.Depre91)])
```

```
## Including the model information
## To the data object for rstan
data_for_stan_03 <-
 list(N = nrow(dat.03), ## sample size
 k = ncol(dat.03),      ## number of indicator
 y = as.matrix(dat.03), ## observed responses
 n_xi = 3,              ## number of factor(s)
 str_loading = c(1, 1, 1, 1,
                 2, 2, 2, 2,
                 3, 3, 3, 3)) ## loading structure
```

Estimating a three-factor Bayesian CFA model ...

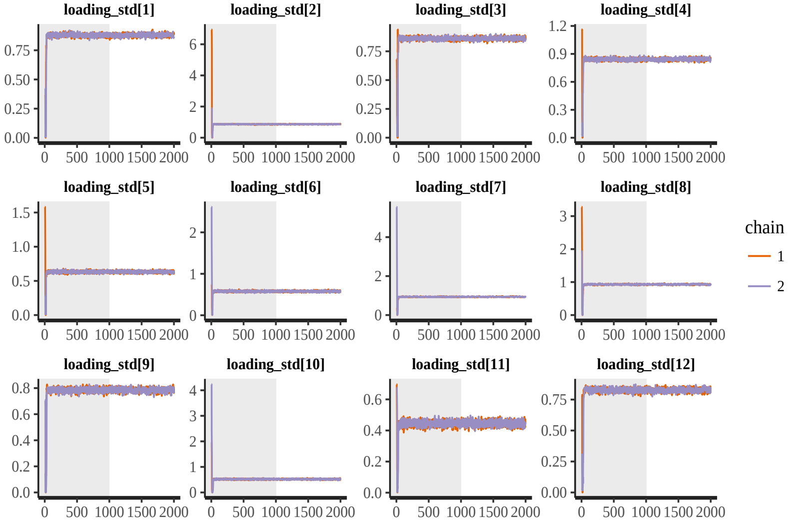
```
## Using the stan() function
## To estimate the model in 1 step
post_03 <-
 stan(file = "cfa-01.stan", ## the Stan syntax file
 data = data_for_stan_03,   ## data input for rstan
 chains = 2,                ## number of MCMC chains
 iter = 2000,               ## total number of MCMC draws per chain
 warmup = 1000)             ## number of warmup draws per chain
## Saving the posterior draws to an .rds file
saveRDS(post_03, file.path(wdir, "post_03.rds"))
```

Summarizing a three-factor Bayesian CFA model results - Factor loadings

```
## Reading the posterior draws back to R
post_03 <- readRDS(file.path(wdir, "post_03.rds"))
```

```
## Trace plot for visualizing MCMC convergence
## Factor loadings
plot(post_03, plotfun = "trace", pars = c("loading_std"),
    inc_warmup = TRUE)
```

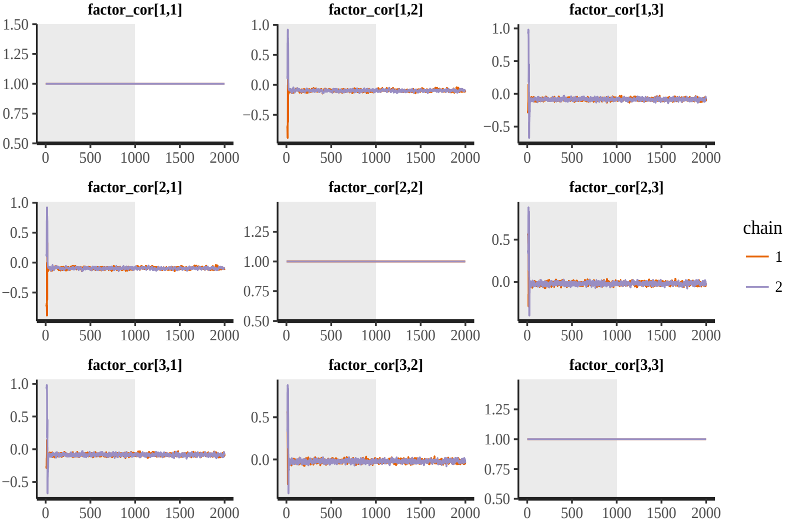
Summarizing a three-factor Bayesian CFA model results - Factor loadings ...



Summarizing a three-factor Bayesian CFA model results - Factor correlations

```r
## Trace plot for visualizing MCMC convergence
## Factor correlation
plot(post_03, plotfun = "trace", pars = c("factor_cor"),
    inc_warmup = TRUE)
```

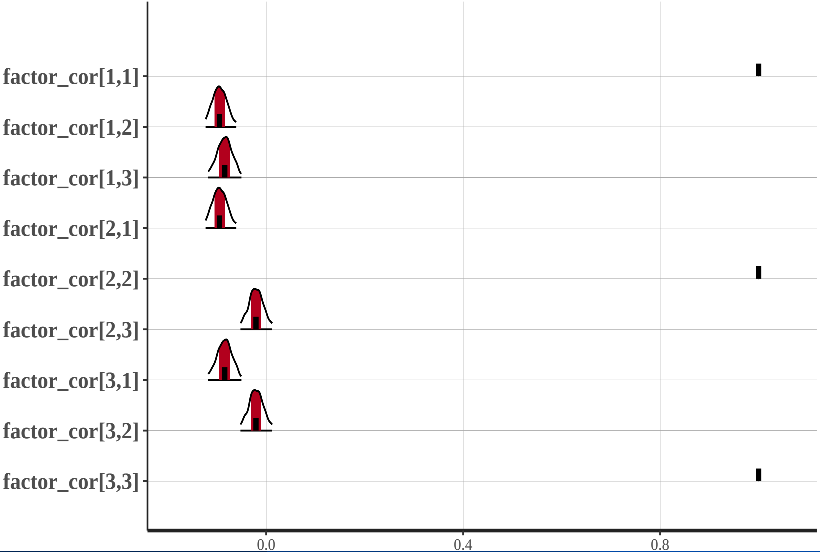
Summarizing a three-factor Bayesian CFA model results - Factor correlations ...



Summarizing a three-factor Bayesian CFA model results - Factor correlations ...

```
## Density plot for factor correlations
plot(post_03, pars = "factor_cor", show_density = TRUE,
    ci_level = 0.5)
```

Summarizing a three-factor Bayesian CFA model results - Factor correlations ... ...



Summarizing a three-factor Bayesian CFA model results - Factor correlations ...

```
## Descriptive table for summarizing the posterior
    distribution
print(post_03, pars = "factor_cor", probs = c(.025, .975),
      digits = 2, mode = TRUE, use_cache = FALSE)
```

```
Inference for Stan model: cfa-01.
2 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=2000.

                  mean se_mean   sd  2.5% 97.5% n_eff Rhat
factor_cor[1,1]   1.00       0 0.00  1.00  1.00  2000  NaN
factor_cor[1,2]  -0.09       0 0.02 -0.12 -0.06   110 1.02
factor_cor[1,3]  -0.08       0 0.02 -0.12 -0.05   351 1.00
factor_cor[2,1]  -0.09       0 0.02 -0.12 -0.06   110 1.02
factor_cor[2,2]   1.00       0 0.00  1.00  1.00  2000 1.00
factor_cor[2,3]  -0.02       0 0.02 -0.05  0.01   448 1.00
factor_cor[3,1]  -0.08       0 0.02 -0.12 -0.05   351 1.00
factor_cor[3,2]  -0.02       0 0.02 -0.05  0.01   448 1.00
factor_cor[3,3]   1.00       0 0.00  1.00  1.00  2000 1.00

Samples were drawn using NUTS(diag_e) at Tue Jun  5 11:46:14 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Outline

Limitations and Future Directions

1. Coding in Stan can be challenging - R packages can help translate lavaan style syntax to Stan or JAGS (e.g., Merkle's blavaan)
2. Missing data - need to be modeled separately (an open question for SEM researchers)
3. Ordinal data - still under development
4. Global model fit or model comparisons - ongoing debate

References

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). *Bayesian data analysis* (3rd ed.). Boca Raton, FL: CRC.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge:UK: Cambridge University.

Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, *6*, 721-741. doi:10.1016/b978-0-08-051581-6.50057-x

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, *57*, 97-109. doi:10.2307/2334940

References ...

Hoffman, M. D., & Gelman, A. (2014). The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, *15*, 1351-1381. Retrieved from http://www.jmlr.org/papers/volume15/hoffman14a/hoffman14a.pdf

Merkle, E. C., & Wang, T. (2018). Bayesian latent variable models for the analysis of experimental psychology data. *Psychonomic Bulletin and Review*, *25*, 256-270. doi:10.3758/s13423-016-1016-7

Metropolis, N., arianna W. Rosenbluth, Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087-1092. doi:10.2172/4390578

References ...

Plummer, M. (2003). Jags: A program for analysis of bayesian graphical models using gibbs sampling. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd international workshop on distributed statistical computing.* Retrieved from https://www.r-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf

Portes, A., & Rumbaut, R. G. (2012). *Children of immigrants longitudinal study (cils), 1991-2006 [Data file and code book]*. Retrieved from https://www.icpsr.umich.edu/icpsrweb/RCMD/studies/20520

Song, X.-Y., & Lee, S.-Y. (2012). *Basic and advanced bayesian structural equation modeling: With applications in the medical and behavioral sciences*. Chichester, UK: John Wiley and Sons.

Spiegelhalter, D., Thomas, A., Best, N., & Lunn, D. (2003). *WinBUGS user manual [Version 1.4]*. Retrieved from https://www.mrc-bsu.cam.ac.uk/wp-content/uploads/manual14.pdf

References ...

Stan Development Team. (2017). *Stan modeling language: User's guide and reference manual [Version 2.17.0]*. Retrieved from http://mc-stan.org/users/documentation/index.html

Session

```
sessionInfo()
```

```
R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04 LTS

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  base

other attached packages:
[1] rstan_2.17.3       StanHeaders_2.17.2 ggplot2_2.2.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.15     digest_0.6.15    grid_3.4.4       plyr_1.8.4       gtable_0.2.0
 [6] stats4_3.4.4     scales_0.5.0     pillar_1.1.0     rlang_0.1.6      lazyeval_0.2.1
[11] labeling_0.3     tools_3.4.4      munsell_0.4.3    compiler_3.4.4   inline_0.3.14
[16] colorspace_1.3-2 gridExtra_2.3    methods_3.4.4    tibble_1.4.2
```