Modeling Construct Change Over Time Amidst Potential Changes in Construct Measurement: A Longitudinal Moderated Factor Analysis Approach

Supplemental Analysis Code

Modeling Construct Change Over Time Amidst Potential Changes in Construct Measurement: A Longitudinal Moderated Factor Analysis Approach

Supplemental Analysis Code

Here we provide code used in the numerical example and the empirical data analysis. We annotated blocks of code and discussed the workflow associated with the longitudinal MNLFA model.

Numerical Example

The Mplus code for fitting a multiple-group second-order growth curve model is shown below. DIF from time and binary covariate "site of study" are specified through different parameter names between timepoints or groups. The sum of growth intercept variance and residual variance is constrained to the population true value 0.76.

```
TITLE: 4 timepoints total variance 1000 sample
DATA: FILE IS example-wide.dat;
VARIABLE:
  NAMES ARE
  id studyeff V1T1 V2T1 V3T1 V4T1 V5T1 V6T1
  V7T1 V8T1 V9T1 V10T1 V11T1 V12T1
  V1T2 V2T2 V3T2 V4T2 V5T2 V6T2 V7T2 V8T2 V9T2 V10T2 V11T2 V12T2
  V1T3 V2T3 V3T3 V4T3 V5T3 V6T3 V7T3 V8T3 V9T3 V10T3 V11T3 V12T3
  V1T4 V2T4 V3T4 V4T4 V5T4 V6T4 V7T4 V8T4 V9T4 V10T4 V11T4 V12T4;
  USEVARIABLES ARE
  studyeff V1T1 V2T1 V3T1 V4T1 V5T1 V6T1 V7T1 V8T1 !V9T1 V10T1
  V1T2 V2T2 V3T2 V4T2 V5T2 V6T2 V7T2 V8T2
  V1T3 V2T3 V3T3 V4T3 V5T3 V6T3 V7T3 V8T3
  V1T4 V2T4 V3T4 V4T4 V5T4 V6T4 V7T4 V8T4;
  CATEGORICAL ARE
  V1T1 V2T1 V3T1 V4T1 V5T1 V6T1 V7T1 V8T1 !V9T1 V10T1
  V1T2 V2T2 V3T2 V4T2 V5T2 V6T2 V7T2 V8T2
  V1T3 V2T3 V3T3 V4T3 V5T3 V6T3 V7T3 V8T3
  V1T4 V2T4 V3T4 V4T4 V5T4 V6T4 V7T4 V8T4;  !V9T2 V10T2;
```

```
MISSING IS .;
CLASSES = site(2);
KNOWNCLASS = site (studyeff= -1 1);
ANALYSIS:
ESTIMATOR = ML;
LINK = Logit; TYPE = Mixture; ALGORITHM = Integration;
  MODEL:
  %OVERALL%
  time1 by V1T1*1 (1)
           V2T1 (L2t1)
           V3T1 (L3t1)
           V4T1*1.7 (L4t1)
           V5T1 (L5)
           V6T1*2.5 (3)
           V7T1*1 (4)
           V8T1*1.3 (5);
  time2 by V1T2* (1)
           V2T2 (L2t2)
           V3T2 (L3t2)
           V4T2*1.83 (L4t2)
           V5T2 (L5)
           V6T2 (3)
           V7T2 (4)
           V8T2 (5);
  time3 by V1T3* (1)
           V2T3 (L2t3)
           V3T3 (L3t3)
           V4T3*1.97 (L4t3)
           V5T3 (L5)
           V6T3 (3)
           V7T3 (4)
           V8T3 (5);
  time4 by V1T4* (1)
```

```
            V2T4 (L2t4)
            V3T4 (L3t4)
            V4T4 (L4t4)
            V5T4 (L5)
            V6T4 (3)
            V7T4 (4)
            V8T4 (5);
  time1-time4*0.41 (sigma2);
  [time1@0 time2@0 time3@0 time4@0];

  int slp | time1@-1.5 time2@-0.5 time3@0.5 time4@1.5;
  int with slp (tau01);
  int*0.35 (tau00);
  [int@-0.06];
  slp*0.07 (tau11);
  [slp*0.4];

  ! Thresholds;
  [V1T1$1*0.5 V1T2$1*0.5 V1T3$1*0.5 V1T4$1*0.5] (2);
  [V2T1$1] (n2t1);
  [V2T2$1] (n2t2);
  [V2T3$1] (n2t3);
  [V2T4$1] (n2t4);
  [V3T1$1] (n3t1);
  [V3T2$1] (n3t2);
  [V3T3$1] (n3t3);
  [V3T4$1] (n3t4);
  [V4T1$1*2.20] (n4t1);
  [V4T2$1*1.87] (n4t2);
  [V4T3$1*1.53] (n4t3);
  [V4T4$1*1.53] (n4t4);
  [V5T1$1] (n5);
  [V5T2$1] (n5);
```

```
[V5T3$1] (n5);
[V5T4$1] (n5);
[V6T1$1*2.5 V6T2$1*2.5 V6T3$1*2.5 V6T4$1*2.5] (7);
[V7T1$1*0.5 V7T2$1*0.5 V7T3$1*0.5 V7T4$1*0.5] (8);
[V8T1$1*0.9 V8T2$1*0.9 V8T3$1*0.9 V8T4$1*0.9] (9);

%site#2%
time1 by V2T1 (L2t1s)
         V3T1 (L3t1s)
         V5T1 (L5s);
time2 by V2T2 (L2t2s)
         V3T2 (L3t2s)
         V5T2 (L5s);
time3 by V2T3 (L2t3s)
         V3T3 (L3t3s)
         V5T3 (L5s);
time4 by V2T4 (L2t4s)
         V3T4 (L3t4s)
         V5T4 (L5s);
int with slp (tau01s);
int*0.35 (tau00s);
[int*0.06](alpha0s);
slp*0.07 (tau11s);
[slp*0.5](alpha1s);

! Thresholds;
[V2T1$1] (n2t1s);
[V2T2$1] (n2t2s);
[V2T3$1] (n2t3s);
[V2T4$1] (n2t4s);
[V3T1$1] (n3t1s);
[V3T2$1] (n3t2s);
[V3T3$1] (n3t3s);
```

```
    [V3T4$1] (n3t4s);
    [V5T1$1] (n5s);
    [V5T2$1] (n5s);
    [V5T3$1] (n5s);
    [V5T4$1] (n5s);

     MODEL CONSTRAINT:
        tau00 = .76 - sigma2;
OUTPUT: TECH1 TECH3;
```

The longitudinal MNLFA model is fit in R with *rstan*. If necessary, readers should become familiar with the `rstan` functions and syntax first. The code uses long-formatted data. It first set up data input for *rstan*, which includes objects such as the number of observations, items, individuals, timepoints, a pattern of DIF, and some parameter prior values. These data objects are listed in a Stan input object `fa.data`. The code also included parameter initial values used to sample independent MCMC chains. To re-use this model code for other datasets, readers can arrange the data into a similar format and provide the corresponding data objects (those listed in `fa.data`) for the model.

```
###########impact and dif replicating analysis in mplus#################
library("Rcpp")
library("rstan")
options(mc.cores = 4)
rstan_options(auto_write = TRUE)
library(data.table)
library(tidyr)
library(dplyr)
bcdat = read.table("example-long.dat", header = F)
colnames(bcdat) = unlist(strsplit(c("id age studyeff eta i1 i2 i3 i4 i5 i6
    i7 i8 i9 i10 i11 i12"), "[ ]+"))
bclong = bcdat %>% filter(age %in% c(-1.5, -0.5, 0.5)) %>% pivot_longer(
   cols = !c('id', 'age' , 'studyeff', 'eta'), names_to = c('item'), names
```

```
   _pattern = 'i([0-9]*)', values_to = 'resp' ) %>%
  filter(item %in% paste0(1:8,"")) #only first 8 items to match mplus
bclong$time = as.numeric(as.factor(bclong$age)) #create wave variable
path = "simout3"

Nobs = nrow(bclong)
P = length(unique(bclong$item))
Ni = length(unique(bclong$id))
D = length(unique(bclong$time))
person = bclong$id; itm = as.numeric(bclong$item); time = as.numeric(
   bclong$time)
Xtv = as.matrix(bclong$age) #time-related dif effect
Xf = as.matrix(bclong$studyeff) #time-invariant impact and dif effect

NFpreds = ncol(Xf); Ntvpreds = ncol(Xtv);
Y = bclong$resp
#dif pattern; col1 is time-varying dif effect, col2 is time-invariant dif
   effect
Ldf <- matrix(c(0,1,1,1,0,0,0,0,
                0,1,1,0,1,0,0,0), nrow = P, ncol = 2) #dif pattern; col1
                   is time-varying dif effect, col2 is time-invariant dif
                   effect
Mtv = sum(Ldf[,1:Ntvpreds])
Mf = sum(Ldf[,(Ntvpreds+1):(Ntvpreds+NFpreds)])

fa.data <-list(Nobs=Nobs,P=P,Ni=Ni,D=D,person=person, itm=itm, time=time,
   NFpreds=NFpreds,Ntvpreds=Ntvpreds,Y=Y,Xf=Xf,Xtv=Xtv,Ldf=Ldf,Mtv=Mtv,Mf=
   Mf,sigma_l=2, sigma_nu=3, sigma_di=2, sigma_cor=2, sigma_f=1)
init_cat = function() {
  init.values<-list(
    Lp = runif(P,0,.1), Np = runif(P,0,.1),
    L_diff = runif(Mf,0,.1), L_diftv = runif(Mtv,0,.1),
    n_diff = runif(Mf,0,.1), n_diftv = runif(Mtv,0,.1),
```

```
    b_mu = matrix(runif(2*NFpreds,0,.05), nrow = 2),
    b_phi = matrix(runif(2*NFpreds,0,.05), nrow = 2),
    mu_slp = runif(1,0,.1),
    phi_slp = runif(1,0,.1),
    phi_int = runif(1,0,.1))
  return(init.values);
}
```

Next the Stan code for the longitudinal MNLFA is written as strings and complied in R. The first two blocks set up the data input and initialized model parameters.

```
    #compile the model
stan_m<- stan_model(model_code = "
data {
  int<lower=1> Nobs;             // number of total observations
  int<lower=1> P;                // number of unique items not considering
      different item sets for now?
  int<lower=1> Ni;                // number of individuals
  int<lower=1> D;                // number of time points
  int<lower=1, upper=Ni> person[Nobs];  // person index
  int<lower=1, upper=P> itm[Nobs];       // item index in the data
  int<lower=1, upper=D> time[Nobs];      // time index in the data
  real age[Nobs];        // age values in the data
  int<lower=0> NFpreds;           // number of time-invariant predictors
  int<lower=0> Ntvpreds;           // number of time-varying predictors
  int Y[Nobs];               // response vector long format
  matrix[Nobs, NFpreds] Xf;      // time-invariant dif and impact
     predictor matrix
  matrix[Nobs, Ntvpreds] Xtv;      // time-varying dif and impact
     predictor matrix
  matrix[P,2] Ldf;               // confirmatory dif pattern; col1 time-
     varying, col2 invariant
  int<lower=1> Mtv;                // number of time-varying dif parameter
     , continuous not expanded over time
```

```
  int<lower=1> Mf;                // number of time-invariant dif
     parameter
  real<lower=0> sigma_l;
  real<lower=0> sigma_nu;
  real<lower=0> sigma_cor;
  real<lower=0> sigma_f;
  real<lower=0> sigma_di;
}
parameters {
  vector<lower=0>[P] Lp;         // loading baseline vector
  vector<offset=0, multiplier=sigma_di>[Mf] L_diff;       // time-
     invariant loading dif; known pattern and 1 variable for now
  vector<offset=0, multiplier=sigma_di>[Mtv] L_diftv;       // time-
     varying continuous loading dif; known pattern and 1 variable for now
  vector<offset=0, multiplier=sigma_nu>[P] Np;   // intercept baseline
     vector
  vector<offset=0, multiplier=sigma_di>[Mf] n_diff;       // time-
     invariant intercept dif; known pattern and 1 variable for now
  vector<offset=0, multiplier=sigma_di>[Mtv] n_diftv;       // time-
     varying continuous intercept dif; known pattern and 1 variable for
     now
  real<offset=0, multiplier=sigma_f> mu_slp; // growth factor slope mean
  real<lower=0> phi_int; // factor diagonal sd
  real<lower=0> phi_slp; // factor diagonal sd
  matrix<offset=0, multiplier=sigma_f>[2,NFpreds] b_mu; // growth factor
     mean impact
  matrix<offset=0, multiplier=sigma_f>[2,NFpreds] b_phi; // factor sd
     impact
  vector[NFpreds] b_phicor; //impact coefficient on phi_cor
  real phicorr; //unconstrained factor correlation coef
  matrix[2,Ni] fac_dist; // helper for non-centered sampling
  matrix[D,Ni] fac_eti_raw; // person and time-specific errors
}
```

The next block designated model identification constraints and assigned vectors DIF parameters to DIF matrices according to the given pattern of DIF. These DIF matrices will be multiplied with predictors.

```
transformed parameters{
   vector[P] Ldiff;   // loading time-invariant dif vector;
   vector[P] Ldiftv; // loading time-varying dif vector;
   vector[P] Ndiff;   // intercept time-invariant dif vector;
   vector[P] Ndiftv; // intercept time-varying dif vector;
   vector[2] mu_eta = [0, mu_slp]';
   vector<lower=0>[2] phi_eta = [phi_int, phi_slp]';
   matrix[D,Ni] fac_eti; // time-specific errors
   real<lower=0> eti_sd;
   eti_sd = sqrt(0.76 - phi_int^2); //sum of intercept and residual
      variance constraint
   fac_eti = fac_eti_raw * eti_sd;
   Ldiff = rep_vector(0,P);
   Ldiftv = rep_vector(0,P); // to be multiplied by person-specific age
   // assign dif to vectors
   {
    int temp;
    temp = 0;
     for (i in 1:P){
        if (Ldf[i,2]==1){
        temp = temp + 1;
        Ldiff[i] = L_diff[temp];}
     }
     temp = 0;
     for (i in 1:P){
        if (Ldf[i,1]==1){
        temp = temp + 1;
        Ldiftv[i] = L_diftv[temp];}
     }
```

```
   }
   Ndiff = rep_vector(0,P);
   Ndiftv = rep_vector(0,P);
   {
    int temp;
    temp = 0;
     for (i in 1:P){
        if (Ldf[i,2]==1){
        temp = temp + 1;
        Ndiff[i] = n_diff[temp];}
     }
     temp = 0;
     for (i in 1:P){
        if (Ldf[i,1]==1){
        temp = temp + 1;
        Ndiftv[i] = n_diftv[temp];}
     }
   }
}
```

The next block set up model prior distributions and the growth models. Distributions of growth factors were first specified in the `fac_gr` object. Then time-specific factor of individuals (initialized as a factor score matrix object) are specified, and the measurement model parameters and related covariate effects are specified. Finally, some parameters of interest such as group-specific growth variance matrices are produced in the `generated quantities` block using the estimated model parameters.

```
model {
  matrix[2,Ni] fac_gr; // growth factor individual scores
  matrix[2,Ni] fac_gr_helper; // helper for fac gr
  matrix[D,Ni] fac_scor; //individual time-specific scores
  real phi_corr;  //impact correlation
  matrix[2,2] phi_mat;
```

```
Lp ~ normal(0,sigma_l);
Np ~ normal(0,sigma_nu);
L_diff ~ normal(0,sigma_di);
L_diftv ~ normal(0,sigma_di);
n_diff ~ normal(0,sigma_di);
n_diftv ~ normal(0,sigma_di);
to_vector(b_mu) ~ normal(0,sigma_f);
to_vector(b_phi) ~ normal(0,sigma_f);
b_phicor ~ normal(0,1);
phicorr ~ normal(0,1);

mu_slp ~ normal(0,sigma_f);
phi_int ~ normal(0,sigma_f);
phi_slp ~ normal(0,sigma_f);
to_vector(fac_dist) ~ normal(0,1);
to_vector(fac_eti_raw) ~ normal(0,1);
phi_mat[1,1] = 1;
phi_mat[2,2] = 1;
{
int k; // unique person index
int itemi;
k = 1;
for (i in 1:Nobs){
  if (person[i] == k){
    phi_corr = 1 - 2 / (exp( 2* (phicorr + Xf[i,] * b_phicor ) ) + 1); /
       /fisher z transform
    phi_mat[2,1] = phi_corr;
    phi_mat[1,2] = phi_corr;
    fac_gr_helper[,k] <- diag_pre_multiply(phi_eta .* exp(b_phi * Xf[i,]
       '), cholesky_decompose(phi_mat)) * fac_dist[,k]; // no
       correlation impact yet
    fac_gr[,k] <- mu_eta + b_mu * Xf[i,]' + fac_gr_helper[,k];
    k = k + 1; }
```

```
    }
  }
  {
  int timei;
  int itemi;
  int persi;
  real agei;
  //The likelihood
  for(j in 1:Nobs){
    timei = time[j]; //time index for other parameters
    agei = age[j];   // age values for coefficients
    itemi = itm[j];
    persi = person[j];
    fac_scor[timei,persi] = fac_gr[1,persi] + fac_gr[2,persi]*agei + fac_
        eti[timei, persi];
    Y[j] ~ bernoulli_logit(Np[itemi]+Ndiff[itemi]*Xf[j,1]+Ndiftv[itemi]*
        agei + (Lp[itemi] + Ldiff[itemi]*Xf[j,1] + Ldiftv[itemi]*agei) *
        fac_scor[timei,persi]);
    }  // time-varying impact without re can be added here;
  }
}
generated quantities{
  cov_matrix[2] phi_var2;
  cov_matrix[2] phi_var1;
  corr_matrix[2] phi_mat1;
  corr_matrix[2] phi_mat2;
  real phi_corr1;
  real phi_corr2;
  phi_mat1[1,1] = 1;
  phi_mat1[2,2] = 1;
  phi_corr1 = 1 - 2 / (exp( 2* (phicorr + b_phicor[1] ) ) + 1); //fisher z
      transform
  phi_mat1[2,1] = phi_corr1;
```

```
  phi_mat1[1,2] = phi_corr1;
  phi_mat2[1,1] = 1;
  phi_mat2[2,2] = 1;
  phi_corr2 = 1 - 2 / (exp( 2* (phicorr - b_phicor[1] ) ) + 1); //check 2
     cov mat, not needed at the end
  phi_mat2[2,1] = phi_corr2;
  phi_mat2[1,2] = phi_corr2;
  phi_var1 = quad_form_diag(phi_mat1, phi_eta .* exp(to_vector(b_phi)));
  phi_var2 = quad_form_diag(phi_mat2, phi_eta .* exp( -1 * to_vector(b_phi
     )));
}
", verbose = F)
```

The complete model syntax is wrapped inside the `stan_model` function to be compiled. It is also possible to save the syntax in a separate text file. Compiling and running all of the code above in R using *rstan* produces a model object. Then we can run the MCMC sampling procedure and produce model estimates. The code below obtains model estimates and some diagnostic information. This numerical example used a 4-core Xeon E5-series CPU on a remote computing cluster. It completed 1500 iterations in about 1.5 hours.

```
stan_f <- sampling(stan_m,
                   data=fa.data,
                   pars=c("Lp","Np", "Ldiff", "Ldiftv","Ndiff", "Ndiftv",
                      "mu_eta", "b_mu", "b_phi", "phi_var1","phi_var2", "
                      phi_eta", "phicorr", "b_phicor", "eti_sd"),
                   chains = 4, iter = 1500, init = init_cat, cores = 4,
                   control = list(adapt_delta=0.86, max_treedepth = 13),
                   seed = 199)

fit_summary <- as.data.frame(summary(stan_f,pars=c("Lp","Np", "Ldiff", "
   Ldiftv","Ndiff", "Ndiftv", "mu_eta", "b_mu", "b_phi", "phi_var1","phi_
   var2", "phi_eta", "phicorr",  "b_phicor","eti_sd"), probs = c
   (0.025,0.5, 0.975))$summary)
```

```
sampler_params <- get_sampler_params(stan_f, inc_warmup = FALSE)
chain_div <- sapply(sampler_params, function(x) sum(x[, "divergent__"]))
chain_tree <- sapply(sampler_params, function(x) max(x[, "treedepth__"]) <
    13)
rhat1 <- sum(fit_summary$Rhat > 1.1, na.rm = T)
rhat01 <- sum(fit_summary$Rhat > 1.01, na.rm = T)
```

Empirical Example

The empirical example uses Add Health public self-weighting core sample. The data used to generate the sample is extracted from three `.rda` objects from the public dataset and processed with the R code below.

```
#########data prep##########
library(dplyr)
library(tidyr)
load('21600-0001-Data.rda')
load('21600-0005-Data.rda')
load('21600-0008-Data.rda')

mergedat = da21600.0001 %>% left_join(da21600.0005, by = 'AID') %>% left_
   join(da21600.0008, by = 'AID')
mergedat$H1GI1Y <- as.numeric(gsub("\\([0-9]+\\) +", "", mergedat$H1GI1Y))
    #birth year; keep full years
mergedat$IYEAR <- as.numeric(gsub("\\([0-9]+\\) +", "", mergedat$IYEAR)) #
   interview year; keep full years
mergedat$IYEAR2 <- as.numeric(gsub("\\([0-9]+\\) +", "", mergedat$IYEAR2))
mergedat$IYEAR3 <- as.numeric(gsub("\\([0-9]+\\) +", "", mergedat$IYEAR3))
mergedat$BIO_SEX <- as.numeric(mergedat$BIO_SEX) - 1 #0 is male 1 is
   female
mergedat$SMP01 <- as.numeric(gsub("^\\(([0-9]+)\\).+$", "\\1", mergedat$
   SMP01)) #core self-weighting sample; 0 is no 1 is yes
mergedat$SMP01_2 <- as.numeric(gsub("^\\(([0-9]+)\\).+$", "\\1", mergedat$
   SMP01_2)) #core self-weighting sample; 0 is no 1 is yes
```

```
#find self-weighting core data, with age gender not missing
coredat = mergedat %>% filter_at(vars(SMP01, SMP01_2), any_vars(. == 1))
   %>% mutate(age_1 = IYEAR - H1GI1Y, age_2 = IYEAR2 - H1GI1Y, age_3 =
   IYEAR3 - H1GI1Y, na.rm = F) %>%
  filter_at( vars(BIO_SEX), all_vars(!is.na(.)) ) %>%  filter( age_1 < 20
     ) #some age above 19 https://www.who.int/health-topics/adolescent-
     health#tab=tab_1
#select delinquicy items; may change to negative affect later
set.seed(235)
coredat_use = coredat %>% select(age_1, age_2, age_3, BIO_SEX,
                                 H1DS1, H1DS2, H1DS3, H1DS8, H1DS9,
   H1DS10, H1DS12, H1DS13, H1DS15, H2DS1, H2DS2, H2DS3, H2DS6, H2DS7,
   H2DS8,  H2DS10,  H2DS11, H2DS12, H3DS1,H3DS2, H3DS3,  H3DS5,   H3DS6)
   %>%
  filter_at(vars(H1DS1:H3DS6), any_vars(!is.na(.))) %>% #keep rows that
     have at least one value
  mutate_at(vars(H1DS1:H3DS6), ~recode(., "(0) (0) Never" = 0, .default=1)
      )
  #mutate_at(vars(H1DS1:H3DS6), ~recode(., "(0) (0) Never" = 0, .default
     =1) ) %>%  slice_sample(n=500)

coredat_long = coredat_use %>% rename(graffiti_1=H1DS1, dmgprop_1=H1DS2,
   lie_1=H1DS3, car_1=H1DS8, stealm_1=H1DS9, house_1=H1DS10, selldrg_1=
   H1DS12, steall_1=H1DS13, unruly_1=H1DS15, #non-violent delinquency
   items
                                    graffiti_2=H2DS1, dmgprop_2=H2DS2,
                                       lie_2=H2DS3, car_2=H2DS6, stealm_
                                       2=H2DS7, house_2=H2DS8,  selldrg_
                                       2=H2DS10, steall_2=H2DS11, unruly
                                       _2=H2DS12, #same items as wave 1
                                    dmgprop_3=H3DS1, stealm_3=H3DS2,
                                       house_3=H3DS3,  selldrg_3=H3DS5,
```

```
                                                steall_3=H3DS6) %>% mutate(ID =
                                               row_number()) %>%
  pivot_longer( !c(BIO_SEX, ID), names_to = c(".value", "wave"), names_sep
      = "_")

longdat = coredat_long %>%  pivot_longer(graffiti:unruly, names_to = "
   question", values_to = "resp") %>%
  mutate(question = factor(question, levels = c( "graffiti", "dmgprop",  "
     lie", "car","stealm", "house", "selldrg", "steall", "unruly")) ) %>%
  mutate(item = as.numeric(question)) %>% na.omit()

write.csv(longdat, "longdat.csv", row.names = F)
```

Next, a longitudinal MNLFA where all DIF is regularized with SSP is fitted to the data. We save the results for subsequent DIF evaluation. We repeat the same process as the numerical example above. Note that age is first centered and transformed with interpretable values. We also choose penalty hyperprior values so that the model can be identified without divergence during estimation. Increasing `gamma_a` gives stronger mean shrinkage, while increasing `gamma_b` produces higher variance on the Laplace prior scale parameters and may weaken the shrinkage effect. Levels of shrinkage necessary depends on the magnitudes of covariate DIF effects being regularized and may vary across different datasets.

```
######age sex agesq ageint age2int data 3 timepoint delinquicy full obs
   ##########
library("Rcpp")
library("rstan")
options(mc.cores = 4)
rstan_options(auto_write = TRUE)
library(tidyr)
library(dplyr)
library(data.table)
```

```
longdat = read.csv("longdat.csv") %>% mutate(age2 = (age - 17.5)/6, agesq
   = age2^2, agsx = age2*BIO_SEX, ag2sx=agesq*BIO_SEX)

path = "sspAH10"
Nobs = nrow(longdat)
P = length(unique(longdat$item))
Ni = length(unique(longdat$ID))
D = length(unique(longdat$wave))
person = longdat$ID; itm = as.numeric(longdat$item); time = as.numeric(
   longdat$wave)
Xtv = as.matrix(cbind(longdat$age2, longdat$agesq, longdat$agsx, longdat$
   ag2sx)) #time-related effect age agesq age*sex
Xf = as.matrix(longdat$BIO_SEX) #time-invariant effect

NFpreds = ncol(Xf); Ntvpreds = ncol(Xtv);
Y = longdat$resp
Ldf <- matrix(rep(1, P*(NFpreds+Ntvpreds)), nrow = P, ncol = NFpreds+
   Ntvpreds) #dif pattern; col1 2 3 is time-varying dif effect, col4 is
   time-invariant dif effect
Mtv1 = sum(Ldf[,1]) #main effect age DIF
Mtv2 = sum(Ldf[,2:Ntvpreds]) #interaction and squared DIF
Mf = sum(Ldf[,(Ntvpreds+1):(Ntvpreds+NFpreds)])

fa.data <-list(Nobs=Nobs,P=P,Ni=Ni,D=D,person=person, itm=itm, time=time,
   NFpreds=NFpreds,Ntvpreds=Ntvpreds,Y=Y,
               Xf=Xf,Xtv=Xtv,Ldf=Ldf,Mtv1=Mtv1,Mtv2=Mtv2,Mf=Mf,sigma_l=2,
                  sigma_nu=2, sigma_di=2, sigma_f=1.5, gamma_a = 4000,
                  gamma_b = 200)
init_cat = function() {
  init.values<-list(
    Lp = runif(P,1,1.5), Np = runif(P,-2,-1.5),
    L_diff = runif(Mf,0,.1), L_diftv = runif(Mtv1,0,.1), L_diftv2 = runif(
       Mtv2,0,.1),
```

```
    n_diff = runif(Mf,0,.1), n_diftv = runif(Mtv1,0,.1), n_diftv2 = runif(
       Mtv2,0,.1),
    b_mu = matrix(runif(2*NFpreds,0,.05), nrow = 2),
    b_phi = matrix(runif(2*NFpreds,0,.05), nrow = 2),
    b_mu_asq = runif(1,0,.1), b_mu_asqint = runif(1,0,.1), mu_slp = runif
       (1,.5,.51), phi_slp = runif(1,.5,.51),
    phi_int = runif(1,.5,.51))
  return(init.values);
}
```

The following blocks of code defines the main Bayesian MNLFA model with SSP selection. Note that the lasso penalty and inclusion probability parameters are added to the raw DIF parameters to form the SSP DIF estimators in the `transformed parameters` block, and the DIF pattern is assumed to be a matrix of 1s. The inclusion parameters for each DIF covariate are sequentially assigned to their corresponding DIF parameters.

```
stan_m<- stan_model(model_code = "
data {
  int<lower=1> Nobs;             // number of total observations
  int<lower=1> P;                // number of unique items not considering
      different item sets for now?
  int<lower=1> Ni;                // number of individuals
  int<lower=1> D;                // number of time points
  int<lower=1, upper=Ni> person[Nobs];  // person index
  int<lower=1, upper=P> itm[Nobs];       // item index in the data
  int<lower=1, upper=D> time[Nobs];      // time index in the data
  int<lower=0> NFpreds;           // number of time-invariant predictors
  int<lower=0> Ntvpreds;           // number of time-varying predictors
  int Y[Nobs];               // response vector long format
  matrix[Nobs, NFpreds] Xf;      // time-invariant dif and impact
     predictor matrix
  matrix[Nobs, Ntvpreds] Xtv;      // time-varying dif and impact
     predictor matrix
```

```
  matrix[P,(NFpreds+Ntvpreds)] Ldf;              // confirmatory dif
     pattern; col1 2 time-varying, col3 invariant
  int<lower=1> Mtv1;                 // number of time-varying dif
     parameter main effect
  int<lower=1> Mtv2;                 // number of time-varying dif
     parameter squared/interaction term
  int<lower=1> Mf;                // number of time-invariant dif
     parameter
  real<lower=0> sigma_l;
  real<lower=0> sigma_nu;
  real<lower=0> sigma_di;
  real<lower=0> sigma_f;
  real gamma_a;
  real gamma_b;
}
parameters {
  vector<lower=0>[P] Lp;         // loading baseline vector
  real<lower=0> lambda_lf;
  real<lower=0> lambda_nf;
  real<lower=0> lambda_lt1;
  real<lower=0> lambda_nt1;
  real<lower=0> lambda_lt2;
  real<lower=0> lambda_nt2;
  vector[P] Np;          // intercept baseline vector
  vector[Mf] L_diff_raw;       // time-invariant loading dif;
  vector[Mtv1] L_diftv1_raw;       // time-varying continuous loading dif
  vector[Mtv2] L_diftv2_raw;       // time-varying continuous loading dif
     non main effect
  vector[Mf] n_diff_raw;       // time-invariant intercept dif; known
     pattern and 1 variable for now
  vector[Mtv1] n_diftv1_raw;       // time-varying continuous intercept
     dif
  vector[Mtv2] n_diftv2_raw;       // time-varying continuous intercept
```

```
    dif non main effect;
  vector<lower=0, upper=1>[P] pi_ff; //by item length is P, by DIF length
    is sum of Ms
  vector<lower=0, upper=1>[P] pi_tv1;
  vector<lower=0, upper=1>[P] pi_tv2;
  vector<lower=0, upper=1>[P] pi_tv3;
  vector<lower=0, upper=1>[P] pi_tv4;
  real mu_slp; // growth factor slope mean
  real<lower=0> phi_int; // factor diagonal sd
  real<lower=0> phi_slp; // factor diagonal sd
  matrix[2,NFpreds] b_mu; // growth factor mean impact
  matrix[2,NFpreds] b_phi; // factor sd impact
  vector[NFpreds] b_phicor; //impact coefficient on phi_cor
  real phicorr; //unconstrained factor correlation coef
  real b_mu_asq; // impact from tvc age squared
  real b_mu_asqint; // impact from tvc age squared interaction

  matrix[2,Ni] fac_dist; // helper for non-centered sampling, in place of
    fac_scor re
  matrix[D,Ni] fac_eti_raw; // person and time-specific errors; assume
    equal timepoints for now
  real eti_sd2; // error variance impact
}
transformed parameters{
  vector[Mf] L_diff;       // time-invariant loading dif;
  vector[Mtv1] L_diftv1;       // time-varying continuous loading dif
  vector[Mtv2] L_diftv2;       // time-varying continuous loading dif non
    main effect
  vector[Mf] n_diff;       // time-invariant intercept dif; known pattern
    and 1 variable for now
  vector[Mtv1] n_diftv1;       // time-varying continuous intercept dif
  vector[Mtv2] n_diftv2;       // time-varying continuous intercept dif
    non main effect;
```

```
vector[P] Ldiff;   // loading time-invariant dif vector;
matrix[P, Ntvpreds] Ldiftv; // loading time-varying dif vector
vector[P] Ndiff;   // intercept time-invariant dif vector
matrix[P, Ntvpreds] Ndiftv; // intercept time-varying dif vector
vector[2] mu_eta = [0, mu_slp]';
vector<lower=0>[2] phi_eta = [phi_int, phi_slp]';

Ldiff = rep_vector(0,P);
Ldiftv = rep_matrix(0,P,Ntvpreds);

L_diff = L_diff_raw ./ lambda_lf;
L_diftv1 = L_diftv1_raw ./ lambda_lt1;
L_diftv2 = L_diftv2_raw ./ lambda_lt2;
n_diff = n_diff_raw ./ lambda_nf;
n_diftv1 = n_diftv1_raw ./ lambda_nt1;
n_diftv2 = n_diftv2_raw ./ lambda_nt2;
// assign dif to vectors
{
 int temp;
 temp = 0;
  for (i in 1:P){
     if (Ldf[i,(Ntvpreds+1)]==1){
     temp = temp + 1;
     Ldiff[i] = L_diff[temp] .* pi_ff[temp];}
  }
  temp = 0;
  for (i in 1:P){
     if (Ldf[i,1]==1){
     temp = temp + 1;s
     Ldiftv[i,1] = L_diftv1[temp] .* pi_tv1[i];} //age
  }
  temp = 0;  // reset temp again here for diftv2 non main effects
  for (i in 1:P){
```

```
      if (Ldf[i,2]==1){
      temp = temp + 1;
      Ldiftv[i,2] = L_diftv2[temp] .* pi_tv2[i];} //agesq
   }
   for (i in 1:P){
      if (Ldf[i,3]==1){
      temp = temp + 1;
      Ldiftv[i,3] = L_diftv2[temp] .* pi_tv3[i];} //ageint
   }
   for (i in 1:P){
      if (Ldf[i,4]==1){
      temp = temp + 1;
      Ldiftv[i,4] = L_diftv2[temp] .* pi_tv4[i];} //age2int
   }
  }
 Ndiff = rep_vector(0,P);
 Ndiftv = rep_matrix(0,P,Ntvpreds);
 {
  int temp;
  temp = 0;
   for (i in 1:P){
      if (Ldf[i,(Ntvpreds+1)]==1){
      temp = temp + 1;
      Ndiff[i] = n_diff[temp] .* pi_ff[temp];}
   }
   temp = 0;
   for (i in 1:P){
      if (Ldf[i,1]==1){
      temp = temp + 1;
      Ndiftv[i,1] = n_diftv1[temp] .* pi_tv1[i];} //age
   }
   temp = 0;  // reset temp again here for diftv2 non main effects
   for (i in 1:P){
```

```
        if (Ldf[i,2]==1){
        temp = temp + 1;
        Ndiftv[i,2] = n_diftv2[temp] .* pi_tv2[i];} //agesq
     }
     for (i in 1:P){
        if (Ldf[i,3]==1){
        temp = temp + 1;
        Ndiftv[i,3] = n_diftv2[temp] .* pi_tv3[i];} //ageint
     }
     for (i in 1:P){
        if (Ldf[i,4]==1){
        temp = temp + 1;
        Ndiftv[i,4] = n_diftv2[temp] .* pi_tv4[i];} //age2int
     }
   }
}
```

For the actual model, note that we include separated residual error SD parameters for female and male groups. We also included effects of age-squared and its interaction with sex on growth factor means. The regularized model took about 7 hours to complete within a $N = 1000$ subset of data and almost two days on the complete dataset using a 4-core Xeon E5-series CPU on a remote computing cluster. After fitting the regularized model, we examine the inclusion probability parameter estimates. We identify DIF effects whose inclusion estimates are above our chosen inclusion threshold (0.8).

```
model {
  matrix[2,Ni] fac_gr; // growth factor individual scores
  matrix[2,Ni] fac_gr_helper; // helper for fac gr
  matrix[D,Ni] fac_scor; //individual time-specific scores
  real phi_corr;  //impact correlation
  matrix[2,2] phi_mat;

  Lp ~ normal(1,sigma_l);
```

```
Np ~ normal(-2,sigma_nu);
lambda_lt1 ~ gamma(gamma_a,gamma_b);
lambda_nt1 ~ gamma(gamma_a,gamma_b);
lambda_lt2 ~ gamma(gamma_a,gamma_b);
lambda_nt2 ~ gamma(gamma_a,gamma_b);
lambda_lf ~ gamma(gamma_a,gamma_b);
lambda_nf ~ gamma(gamma_a,gamma_b);
L_diff_raw ~ double_exponential(0, 1);
n_diff_raw ~ double_exponential(0, 1);
L_diftv1_raw ~ double_exponential(0, 1);
n_diftv1_raw ~ double_exponential(0, 1);
L_diftv2_raw ~ double_exponential(0, 1);
n_diftv2_raw ~ double_exponential(0, 1);
to_vector(b_mu) ~ normal(0,sigma_f);
to_vector(b_phi) ~ normal(0,sigma_f);
b_mu_asq ~ normal(0,sigma_f);
b_mu_asqint ~ normal(0,sigma_f);
b_phicor ~ normal(0,1);
phicorr ~ normal(0,1);

mu_slp ~ normal(0,sigma_di);
phi_int ~ normal(0,sigma_di);
phi_slp ~ normal(0,sigma_di);
eti_sd2 ~ normal(0,1);
to_vector(fac_dist) ~ normal(0,1);
to_vector(fac_eti_raw) ~ normal(0,1);
pi_ff ~ beta(0.5,0.5);
pi_tv1 ~ beta(0.5,0.5);
pi_tv2 ~ beta(0.5,0.5);
pi_tv3 ~ beta(0.5,0.5);
pi_tv4 ~ beta(0.5,0.5);

phi_mat[1,1] = 1;
```

```
phi_mat[2,2] = 1;
{
int k; // unique person index
int itemi;
k = 1;
for (i in 1:Nobs){
  if (person[i] == k){
    phi_corr = 1 - 2 / (exp( 2* (phicorr + Xf[i,] * b_phicor ) ) + 1); /
       /fisher z transform
    phi_mat[2,1] = phi_corr;
    phi_mat[1,2] = phi_corr;
    fac_gr_helper[,k] <- diag_pre_multiply(phi_eta .* exp(b_phi * Xf[i,]
       ' ), cholesky_decompose(phi_mat)) * fac_dist[,k]; // no
       correlation impact yet
    fac_gr[,k] <- mu_eta + b_mu * Xf[i,]' + fac_gr_helper[,k];
    k = k + 1; }
  }
}

{
int timei;
int itemi;
int persi;
real agei;
real asq;
real asqint;
//The likelihood
for(j in 1:Nobs){
  timei = time[j]; //time index for other parameters
  agei = Xtv[j,1];   // age values for coefficients
  asq = Xtv[j,2];   // age squared
  asqint = Xtv[j,4];   // age2 sex interaction; age sex interaction
     included in b_mu[2];
```

```
    itemi = itm[j];
    persi = person[j];
    fac_scor[timei,persi] = fac_gr[1,persi] + fac_gr[2,persi]*agei + b_mu_
       asq*asq + b_mu_asqint*asqint + fac_eti_raw[timei, persi] * exp(eti_
       sd2*Xf[j,1]); // * exp(eti_sd2*Xf[j,1]);
    Y[j] ~ bernoulli_logit(Np[itemi]+Ndiff[itemi]*Xf[j,1]+Ndiftv[itemi, ]*
       Xtv[j,]' + (Lp[itemi] + Ldiff[itemi]*Xf[j,1] + Ldiftv[itemi, ]*Xtv[
       j,]') * fac_scor[timei,persi]);
    }
  }
}
generated quantities{
  cov_matrix[2] phi_var2;
  cov_matrix[2] phi_var1;
  corr_matrix[2] phi_mat1;
  corr_matrix[2] phi_mat2;
  real phi_corr1;
  real phi_corr2;
  real<lower=0> eti_var2;

  phi_mat1[1,1] = 1;
  phi_mat1[2,2] = 1;
  phi_corr1 = 1 - 2 / (exp( 2* (phicorr ) ) + 1); //fisher z transform
  phi_mat1[2,1] = phi_corr1;
  phi_mat1[1,2] = phi_corr1;

  phi_mat2[1,1] = 1;
  phi_mat2[2,2] = 1;
  phi_corr2 = 1 - 2 / (exp( 2* (phicorr + b_phicor[1] ) ) + 1);
  phi_mat2[2,1] = phi_corr2;
  phi_mat2[1,2] = phi_corr2;

  phi_var1 = quad_form_diag(phi_mat1, phi_eta * 1);
```

```
  phi_var2 = quad_form_diag(phi_mat2, phi_eta .* exp( 1 * to_vector(b_phi)
     ));
  eti_var2 = (1*exp(eti_sd2*1))^2;
}   ", verbose = F)
stan_f <- sampling(stan_m,
                   data=fa.data,
                   pars=c("Lp","Np", "Ldiff", "Ldiftv","Ndiff", "Ndiftv",
                      "mu_eta", "b_mu", "b_phi", "b_phi_asq", "phi_var1","
                      phi_var2", "phi_eta", "phicorr", "b_phicor", "
                      lambda2_lt1", "lambda2_lt2", "lambda2_lf", "lambda2_
                      nt1", "lambda2_nt2", "lambda2_nf", "pi_ff", "pi_tv1"
                      , "pi_tv2", "pi_tv3"),
                   chains = 4, iter = 3000, init = init_cat, cores = 4,
                   control = list(adapt_delta=0.89, max_treedepth = 13),
                   seed = 90)

fit_summary <- as.data.frame(summary(stan_f, pars=c("Lp","Np", "Ldiff", "
   Ldiftv","Ndiff", "Ndiftv", "mu_eta", "b_mu", "b_phi", "b_phi_asq", "phi
   _var1","phi_var2", "phi_eta", "phicorr", "b_phicor", "lambda2_lt1", "
   lambda2_lt2", "lambda2_lf",  "lambda2_nt1", "lambda2_nt2", "lambda2_nf"
   , "pi_ff", "pi_tv1", "pi_tv2", "pi_tv3"), probs = c(0.025,0.5, 0.975))$
   summary)
sampler_params <- get_sampler_params(stan_f, inc_warmup = FALSE)
chain_div <- sapply(sampler_params, function(x) sum(x[, "divergent__"]))

save.image("ssp1l-at.RData")

library(posterior)
stan_fdraws <- as_draws_df(stan_f)
print(summarise_draws(stan_fdraws), n=180)
summarise_draws(subset_draws(stan_fdraws, variable = c("^pi_ff","^pi_tv"),
    regex = T)) %>% filter(mean >0.8)
```

Next we re-estimate a longitudinal MNLFA with the detected DIF patterns. R code used to set the DIF pattern and prior values are shown below. The subsequent *rstan model* compilation and result summary is similar to the one shown in the numerical example and is omitted from here. In our experience, changing the way age is scaled or not scaling age may result in non-convergence for the same model. When age or other continuous covariate has a large SD, true DIF effects in the original data may be much smaller than the DIF effects in the data rescaled to a smaller SD. As a result, priors that are informative (scaled close to the true magnitude of the target parameter) for model fitting in the rescaled data may not be informative enough in the original data to produce converged solutions. This combined with the complexity of the current model causes the model to be less easy to converge when the scale of age increased from the penalized model.

```
longdat = read.csv("longdat") %>% mutate(age2 = (age - 17.5)/6, agesq =
   age2^2, agsx = age2*BIO_SEX)
path = "lssAH-at"
Nobs = nrow(longdat)
P = length(unique(longdat$item))
Ni = length(unique(longdat$ID))
D = length(unique(longdat$wave))
person = longdat$ID; itm = as.numeric(longdat$item); time = as.numeric(
   longdat$wave)
Xtv = as.matrix(cbind(longdat$age2,longdat$agesq, longdat$agsx)) #time-
   varying effect age agesq age*sex
Xf = as.matrix(longdat$BIO_SEX) #time-invariant effect

NFpreds = ncol(Xf); Ntvpreds = ncol(Xtv);
Y = longdat$resp
Ldf <- matrix(c(0,1,1,1,1,0,1,0,1, #col1 is age; col2 agesq col3 agesex
                0,1,1,0,0,0,1,0,0,
                0,0,0,0,0,0,0,0,1,
                0,1,1,0,0,0,1,0,1), nrow = P, ncol = 4) #dif pattern; col1
```

```
                    2 3 is time-varying dif effect, col4 is time-invariant
                    dif effect
Mtv1 = sum(Ldf[,1:(Ntvpreds-2)]) #main effect age DIF
Mtv2 = sum(Ldf[,(Ntvpreds-1):Ntvpreds]) #interaction and squared DIF
Mf = sum(Ldf[,(Ntvpreds+1):(Ntvpreds+NFpreds)])

fa.data <-list(Nobs=Nobs,P=P,Ni=Ni,D=D,person=person, itm=itm, time=time,
   NFpreds=NFpreds,Ntvpreds=Ntvpreds,Y=Y, age2int=age2int,
               Xf=Xf,Xtv=Xtv,Ldf=Ldf,Mtv1=Mtv1,Mtv2=Mtv2,Mf=Mf,sigma_l
                  =1.5, sigma_nu=1.5, sigma_di=1,sigma_cor=1.5, sigma_f
                  =1.5)
init_cat = function() {
  init.values<-list(
    Lp = runif(P,1,1.5), Np = runif(P,-1,0),
    L_diff = runif(Mf,0,.1), L_diftv = runif(Mtv1,0,.1), L_diftv2 = runif(
       Mtv2,0,.1),
    n_diff = runif(Mf,0,.1), n_diftv = runif(Mtv1,0,.1), n_diftv2 = runif(
       Mtv2,0,.1),
    b_mu = matrix(runif(2*NFpreds,0,.05), nrow = 2),
    b_phi = matrix(runif(2*NFpreds,0,.05), nrow = 2),
    b_mu_asq = runif(1,0,.1), b_mu_asqint = runif(1,0,.1), mu_slp = runif
       (1,0,.1), phi_slp = runif(1,0,.1),
    phi_int = runif(1,0,.1))
  return(init.values);
}
#compile the model
stan_m<- stan_model(file = 'AH.stan', verbose = F)
stan_f <- sampling(stan_m,
                   data=fa.data,
                   pars=c("Lp","Np", "Ldiff", "Ldiftv","Ndiff", "Ndiftv",
                      "mu_eta", "b_mu", "b_phi", "b_mu_asq", "b_mu_asqint"
                      , "eti_var2", "phi_var1","phi_var2", "phi_eta", "
                      phicorr", "b_phicor", "fac_gr"),
```

```
                    chains = 4, warmup=1000, iter=2500, init = init_cat,
                       cores = 4,
                    control = list(adapt_delta=0.86, max_treedepth = 13),
                    seed = 198)
```