Using measurementInvariance

Daniel Schulze

Version 0.3

Package overview

measurementInvariance is an R package dedicated to sound measurement invariance (MI) analysis, focusing on issues of establishing partial MI. It subsumes SEM and IRT models by importing from lavaan and mirt respectively.

Imported packages:

- lavaan
- mirt
- Ckmeans.1d.dp
- reshape2
- msm
- blavaan (only needed in Bayesian part)
- rstan (only needed in Bayesian part)

Function overview

There are 4 main functions:

- testMI(): Global MI tests
- clusterItems(): Under violations of MI, find clusters (subsets) of items, for which MI holds
- partialMI(): Use a chosen item cluster as anchor
- modelAveraging(): When not choosing an item cluster, apply Bayesian model averaging to reflect information from several competing partial MI models.

The typical work flow is intended to be testMI() -> clusterItems() -> either partialMI() or modelAveraging().

Table 1: Current state of testMI()

	2 groups	> 2 groups	longit. (2)	cont. covar
cont. items	X	X		
dich. items: factor	X	X		
dich. items: Rasch	X	X		
dich. items: 2PL	X	X		
ordinal				

Table 2: Current state of clusterItems()

	2 groups	> 2 groups	longitudinal (2)	cont. covar
cont. items	X			

	2 groups	> 2 groups	longitudinal (2)	cont. covar
dich. items: factor	X			
dich. items: Rasch	X			
dich. items: 2PL	X			
ordinal				

Table 3: Current state of partialMI()

	2 groups	> 2 groups	longitudinal (2)	cont. covar	multidim.
cont. items	X				X
dich. items: factor	X				X
dich. items: Rasch	X				X
dich. items: 2PL	X				X
ordinal					

Table 4: Current state of modelAveraging()

	2 groups	> 2 groups	longitudinal (2)	cont. covar	multidim.
cont. items	X				
dich. items: factor					
dich. items: Rasch	X				
dich. items: 2PL	X				
ordinal					

Continuous data (& partial MI)

Here we are taking data from the Holzinger-Swinefort (1939) example on cognitive tests. We will use the "Speed" and and the "Math" items with gender as a grouping variable, for which MI is to be tested.

```
suppressMessages(library(MBESS))
data(HS)
Data <- HS[, c("t10_addition", "t11_code", "t12_counting_groups_of_dots", # speed items
               "t13_straight_and_curved_capitals",
               "t20_deduction", "t21_numerical_puzzles", "t22_problem_reasoning", # math items
               "t23_series_completion", "t24_woody_mccall", "sex")]
colnames(Data) <- sub("_.*", "", colnames(Data)) # shorten variable names for convenience
str(Data)
#> 'data.frame':    301 obs. of  10 variables:
#>  $ t10: int  78 87 75 69 85 100 108 78 104 95 ...
#>  $ t11: int  74 84 49 65 63 92 65 80 52 74 ...
#>  $ t12: int  115 125 78 106 126 133 124 103 93 91 ...
#>  $ t13: int  229 285 159 175 213 270 175 132 265 157 ...
#>  $ t20: int  3 -3 -3 -2 29 9 18 15 12 33 ...
#>  $ t21: int  14 13 9 10 15 2 10 9 15 8 ...
#>  $ t22: int  34 21 18 22 19 16 19 22 18 25 ...
#>  $ t23: int  5 1 7 6 4 10 3 18 17 8 ...
#>  $ t24: int  24 12 20 19 20 22 15 24 18 16 ...
#>  $ sex: Factor w/ 2 levels "Female","Male": 2 1 1 2 1 1 2 1 1 1 ...
```

Assume, we are ultimately interested in the correlation of the two factors Speed and Math. We are thus interested in establishing weak MI. Set up a model just like in lavaan. The function testMI sets up models for all factor separately. We identify (borderline) issues with weak MI for Speed, while it holds for Math. (for cut offs see Chen, 2007)

```
model <- "Speed =~ t10 + t11 + t12 + t13
          Math =~ t20 + t21 + t22 + t23 + t24"

res_testMI <- testMI(model,
                     group = "sex",
                     data = Data,
                     MIlevel = "weak")
#> Input is a cross-sectional model with 2 groups and 2 factors fitted seperately.
summary(res_testMI)
#> Two group model with:
#>  Female Male total
#>     155  146   301
#>
#>                        Speed             Math
#> MI level           configural   weak configural   weak
#> chi2                    17.26  26.24       8.92  16.74
#> df                          4      7         10     14
#> p                       0.002  0.000      0.540  0.270
#> CFI                     0.950  0.928      1.000  0.993
#> RMSEA                   0.148  0.135      0.000  0.036
#> RMSEA 90% lower         0.084  0.084      0.000  0.000
#> SRMR                    0.038  0.065      0.023  0.052
#> diff chi2                      8.796              8.21
#> diff df                            3                 4
#> diff p                         0.032             0.084
#> diff CFI                      -0.022            -0.007
#> diff RMSEA                    -0.013             0.036
#> diff SRMR                      0.027              0.03
#>
#> package                lavaan
#> estimator                 MLR
#> item type          continuous
#> item missings            none
#> standardized              yes
#>
#> Use getModel() to access parameter estimates or to further process the results.
```

Proceed to identify item clusters for which MI holds. First, item clustering is done by setting a threshold in loading difference that is not to be surpassed by the items of a specific cluster. The smaller the threshold, the more homogeneous the items of a cluster become when compared across groups.

```
res_clusterItems <- clusterItems(res_testMI,
                                 MIholding = "Speed configural Math weak",
                                 method = "threshold",
                                 loadThreshold = 0.3)
#> Skipping clustering for Math as MI level already holds.
summary(res_clusterItems,
        order = "clusters")
#> Clustering by threshold criterion with load threshold 0.3 and intercept threshold NA.
#>
```

```
#> Factor: Speed   2 clusters found (configural -> weak).
#>     cluster
#> t13       1
#> t10       2
#> t11       2
#> t12       2
#> Factor: Math   (not clustered)
#>     cluster
#> t20       1
#> t21       1
#> t22       1
#> t23       1
#> t24       1
```

Alternatively, a significance test can be used, which yields the same result in this example.

```
res_clusterItems2 <- clusterItems(res_testMI,
                                  MIholding = "Speed configural Math weak",
                                  method = "sigTest",
                                  pValue = 0.05)
#> Skipping clustering for Math as MI level already holds.
summary(res_clusterItems2,
        order = "clusters")
#> Clustering by sign. test with p of 0.05 .
#>
#> Factor: Speed   2 clusters found (configural -> weak).
#>     cluster
#> t13       1
#> t10       2
#> t11       2
#> t12       2
#> Factor: Math   (not clustered)
#>     cluster
#> t20       1
#> t21       1
#> t22       1
#> t23       1
#> t24       1
```

After inspection of the items we might decide for going with cluster 2 of the Speed items as anchor items. Hence we call the partialMI function:

```
partialMI(res_clusterItems,
          "Speed 2 Math 1") # insert the factor names followed by the cluster label
#> lavaan 0.6-8 ended normally after 40 iterations
#>
#>   Estimator                                         ML
#>   Optimization method                           NLMINB
#>   Number of model parameters                        60
#>   Number of equality constraints                    17
#>
#>   Number of observations per group:
#>     Male                                           146
#>     Female                                         155
#>
```

```
#> Model Test User Model:
#>                                               Standard      Robust
#>   Test Statistic                               163.718     170.587
#>   Degrees of freedom                                65          65
#>   P-value (Chi-square)                           0.000       0.000
#>   Scaling correction factor                                  0.960
#>        Yuan-Bentler correction (Mplus variant)
#>   Test statistic for each group:
#>     Male                                        77.250      80.491
#>     Female                                      86.468      90.096
#>
#> Model Test Baseline Model:
#>
#>   Test statistic                               856.233     864.037
#>   Degrees of freedom                                72          72
#>   P-value                                        0.000       0.000
#>   Scaling correction factor                                  0.991
#>
#> User Model versus Baseline Model:
#>
#>   Comparative Fit Index (CFI)                    0.874       0.867
#>   Tucker-Lewis Index (TLI)                       0.861       0.852
#>
#>   Robust Comparative Fit Index (CFI)                         0.871
#>   Robust Tucker-Lewis Index (TLI)                            0.857
#>
#> Loglikelihood and Information Criteria:
#>
#>   Loglikelihood user model (H0)              -3463.543   -3463.543
#>   Scaling correction factor                                  0.749
#>       for the MLR correction
#>   Loglikelihood unrestricted model (H1)      -3381.684   -3381.684
#>   Scaling correction factor                                  0.994
#>       for the MLR correction
#>
#>   Akaike (AIC)                                7013.086    7013.086
#>   Bayesian (BIC)                              7172.492    7172.492
#>   Sample-size adjusted Bayesian (BIC)         7036.120    7036.120
#>
#> Root Mean Square Error of Approximation:
#>
#>   RMSEA                                          0.100       0.104
#>   90 Percent confidence interval - lower         0.081       0.085
#>   90 Percent confidence interval - upper         0.120       0.123
#>   P-value RMSEA <= 0.05                          0.000       0.000
#>
#>   Robust RMSEA                                               0.102
#>   90 Percent confidence interval - lower                     0.083
#>   90 Percent confidence interval - upper                     0.121
#>
#> Standardized Root Mean Square Residual:
#>
#>   SRMR                                           0.084       0.084
```

```
#>
#> Parameter Estimates:
#>
#>   Standard errors                             Sandwich
#>   Information bread                           Observed
#>   Observed information based on                Hessian
#>
#>
#> Group 1 [Male]:
#>
#> Latent Variables:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>   Speed =~
#>     t10     (.p1.)    0.626    0.068    9.254    0.000
#>     t11     (.p2.)    0.717    0.081    8.832    0.000
#>     t12     (.p3.)    0.682    0.115    5.928    0.000
#>     t13               0.545    0.071    7.716    0.000
#>   Math =~
#>     t20     (.p5.)    0.518    0.069    7.453    0.000
#>     t21     (.p6.)    0.633    0.065    9.725    0.000
#>     t22     (.p7.)    0.601    0.064    9.417    0.000
#>     t23     (.p8.)    0.712    0.062   11.553    0.000
#>     t24     (.p9.)    0.610    0.058   10.507    0.000
#>
#> Covariances:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>   Speed ~~
#>     Math              0.641    0.083    7.745    0.000
#>
#> Intercepts:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>    .t10     (.22.)   -0.069    0.073   -0.944    0.345
#>    .t11     (.23.)   -0.081    0.086   -0.942    0.346
#>    .t12     (.24.)   -0.087    0.081   -1.070    0.285
#>    .t13     (.25.)   -0.083    0.076   -1.098    0.272
#>    .t20     (.26.)    0.026    0.073    0.354    0.723
#>    .t21     (.27.)    0.050    0.071    0.705    0.481
#>    .t22     (.28.)    0.047    0.073    0.641    0.522
#>    .t23     (.29.)    0.051    0.076    0.667    0.505
#>    .t24     (.30.)    0.037    0.070    0.522    0.602
#>     Speed             0.000
#>     Math              0.000
#>
#> Variances:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>    .t10               0.505    0.068    7.382    0.000
#>    .t11               0.485    0.090    5.383    0.000
#>    .t12               0.644    0.126    5.101    0.000
#>    .t13               0.493    0.066    7.411    0.000
#>    .t20               0.766    0.098    7.811    0.000
#>    .t21               0.473    0.081    5.805    0.000
#>    .t22               0.740    0.096    7.728    0.000
#>    .t23               0.503    0.072    6.966    0.000
```

```
#>     .t24              0.536    0.065    8.238    0.000
#>      Speed            1.000
#>      Math             1.000
#>
#>
#> Group 2 [Female]:
#>
#> Latent Variables:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>    Speed =~
#>      t10      (.p1.)    0.626    0.068    9.254    0.000
#>      t11      (.p2.)    0.717    0.081    8.832    0.000
#>      t12      (.p3.)    0.682    0.115    5.928    0.000
#>      t13                0.840    0.175    4.813    0.000
#>    Math =~
#>      t20      (.p5.)    0.518    0.069    7.453    0.000
#>      t21      (.p6.)    0.633    0.065    9.725    0.000
#>      t22      (.p7.)    0.601    0.064    9.417    0.000
#>      t23      (.p8.)    0.712    0.062   11.553    0.000
#>      t24      (.p9.)    0.610    0.058   10.507    0.000
#>
#> Covariances:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>    Speed ~~
#>      Math               0.579    0.168    3.436    0.001
#>
#> Intercepts:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>     .t10      (.22.)   -0.069    0.073   -0.944    0.345
#>     .t11      (.23.)   -0.081    0.086   -0.942    0.346
#>     .t12      (.24.)   -0.087    0.081   -1.070    0.285
#>     .t13      (.25.)   -0.083    0.076   -1.098    0.272
#>     .t20      (.26.)    0.026    0.073    0.354    0.723
#>     .t21      (.27.)    0.050    0.071    0.705    0.481
#>     .t22      (.28.)    0.047    0.073    0.641    0.522
#>     .t23      (.29.)    0.051    0.076    0.667    0.505
#>     .t24      (.30.)    0.037    0.070    0.522    0.602
#>      Speed              0.222    0.159    1.401    0.161
#>      Math              -0.127    0.141   -0.901    0.368
#>
#> Variances:
#>                    Estimate  Std.Err  z-value  P(>|z|)
#>     .t10                0.738    0.092    8.014    0.000
#>     .t11                0.551    0.103    5.339    0.000
#>     .t12                0.474    0.071    6.679    0.000
#>     .t13                0.587    0.115    5.116    0.000
#>     .t20                0.612    0.092    6.676    0.000
#>     .t21                0.625    0.108    5.778    0.000
#>     .t22                0.449    0.072    6.267    0.000
#>     .t23                0.377    0.064    5.923    0.000
#>     .t24                0.618    0.077    7.976    0.000
#>      Speed              0.826    0.271    3.044    0.002
#>      Math               1.221    0.256    4.777    0.000
```

```
#> lavaan 0.6-8 ended normally after 40 iterations
#>
#>   Estimator                                         ML
#>   Optimization method                           NLMINB
#>   Number of model parameters                        60
#>   Number of equality constraints                    17
#>
#>   Number of observations per group:
#>     Male                                           146
#>     Female                                         155
#>
#> Model Test User Model:
#>                                               Standard      Robust
#>   Test Statistic                               163.718     170.587
#>   Degrees of freedom                                65          65
#>   P-value (Chi-square)                           0.000       0.000
#>   Scaling correction factor                                  0.960
#>        Yuan-Bentler correction (Mplus variant)
#>   Test statistic for each group:
#>     Male                                        77.250      80.491
#>     Female                                      86.468      90.096
```

After establishing weak (partial) MI, it is now admissible to compare the latent correlation of Speed and Math between the groups.

Dichotomous data (& Bayesian model averaging)

The package provides good support for dichotomous data models via categorical SEM models in lavaan and IRT models in mirt. Here we have a second example from the FIMS study testing mathematical ability where a 2PL IRT model is applied. Our goal is to compare the latent means of two countries (1 Australia - 2 Japan). We thus need to establish strong MI. We find no issues with weak MI, but clear violation of strong MI.

```
suppressMessages(library(TAM))
data("data.fims.Aus.Jpn.scored")
# choosing only a subset of items and a subset of the sample to keep a lid on
# the computation times of Bayesian analyses
dataDich <- data.fims.Aus.Jpn.scored[c(1:500, 5801:6300), c(2, 3, 4, 8, 9, 11, 15, 16)]
str(dataDich)
#> 'data.frame':    1000 obs. of  8 variables:
#>  $ M1PTI1 : num  1 0 1 1 1 1 0 1 0 0 ...
#>  $ M1PTI2 : num  0 1 0 1 1 1 0 1 0 1 ...
#>  $ M1PTI3 : num  1 1 1 1 1 1 1 0 1 1 ...
#>  $ M1PTI12: num  0 0 0 1 0 0 1 0 0 0 ...
#>  $ M1PTI14: num  0 1 0 0 1 0 1 1 1 0 ...
#>  $ M1PTI18: num  0 0 1 0 1 1 0 1 0 1 ...
#>  $ M1PTI23: num  1 1 1 0 1 0 0 1 0 1 ...
#>  $ country: int  1 1 1 1 1 1 1 1 1 1 ...

res_testMI <- testMI(items = colnames(dataDich[1:7]),
                     group = "country",
                     data = dataDich,
                     MIlevel = "strong",
                     dich = T,
                     dichModel = "2PL")
```

```
#> Input is a cross-sectional model with 2 groups and 1 factor.
summary(res_testMI)
#> Two group model with:
#>  group1 group2 total
#>     500    500  1000
#>
#>                     Factor
#> MI level        configural     weak   strong
#> AIC               7503.048 7499.910 7593.687
#> SABIC             7551.536 7538.008 7621.394
#> BIC               7640.466 7607.881 7672.211
#> M2                  44.040   52.092  154.823
#> df                      28       35       42
#> p                    0.028    0.032    0.000
#> RMSEA                0.024    0.022    0.052
#> RMSEA 90% lower      0.008    0.007    0.043
#> CFI                  0.963    0.961    0.741
#> diff chi2                     8.862  105.776
#> diff df                           6        6
#> diff p                        0.181        0
#> diff RMSEA                   -0.002     0.03
#> diff CFI                     -0.002    -0.22
#>
#> package             mirt
#> estimator             EM
#> item type            2PL
#> item missings       none
#>
#> Use getModel() to access parameter estimates or to further process the results.
```

If you want to have a look at a specific model estimated by testMI(), use getModel(). This prints a model summary. Additionally, any method from the core package (in this case: mirt) can be applied to the resulting object (e.g. coef, itemfit).

```
resWeak <- getModel(res_testMI,
                    "Factor",
                    "weak")
#> mirt model estimates
#>
#> Full-information item factor analysis with 1 factor(s).
#> Converged within 1e-04 tolerance after 57 EM iterations.
#> mirt version: 1.33.2
#> M-step optimizer: nlminb
#> EM acceleration: Ramsay
#> Number of rectangular quadrature: 61
#> Latent density type: Gaussian
#>
#> Information matrix estimated with method: Oakes
#> Second-order test: model is a possible local maximum
#> Condition number of information matrix =  17.69478
#>
#> Log-likelihood = -3727.955
#> Estimated parameters: 29
#> AIC = 7499.91; AICc = 7500.946
```

```
#> BIC = 7607.881; SABIC = 7538.008
#>               stats
#> M2           52.092
#> df           35.000
#> p             0.032
#> RMSEA         0.022
#> RMSEA_5       0.007
#> RMSEA_95      0.034
#> SRMSR.group1  0.045
#> SRMSR.group2  0.050
#> TLI           0.953
#> CFI           0.961
#>
#>
#> Parameter Estimates:
#>
#> $group1
#> $items
#>            a1      d g u
#> M1PTI1  1.029  0.970 0 1
#> M1PTI2  1.699  1.548 0 1
#> M1PTI3  1.190  1.950 0 1
#> M1PTI12 0.503 -0.787 0 1
#> M1PTI14 0.541  0.353 0 1
#> M1PTI18 1.128  0.487 0 1
#> M1PTI23 1.328  2.049 0 1
#>
#> $means
#> F1
#>  0
#>
#> $cov
#>    F1
#> F1  1
#>
#>
#> $group2
#> $items
#>            a1      d g u
#> M1PTI1  1.029  1.665 0 1
#> M1PTI2  1.699  2.821 0 1
#> M1PTI3  1.190  2.864 0 1
#> M1PTI12 0.503 -0.607 0 1
#> M1PTI14 0.541 -0.548 0 1
#> M1PTI18 1.128  0.869 0 1
#> M1PTI23 1.328  1.548 0 1
#>
#> $means
#> F1
#>  0
#>
#> $cov
#>       F1
```

```
#> F1 0.949
```

Applying clusterItems() with a threshold for intercepts (or tresholds in IRT terms), we find three item clusters.

```
res_clusterItems <- clusterItems(res_testMI,
                                 MIholding = "weak",
                                 method = "threshold",
                                 intThreshold = 0.6)
summary(res_clusterItems)
#> Clustering by threshold criterion with load threshold 0.6 and intercept threshold NA.
#>
#> Factor: Factor   3 clusters found (weak -> strong).
#>         cluster
#> M1PTI1        1
#> M1PTI2        1
#> M1PTI3        1
#> M1PTI12       2
#> M1PTI18       2
#> M1PTI14       3
#> M1PTI23       3
```

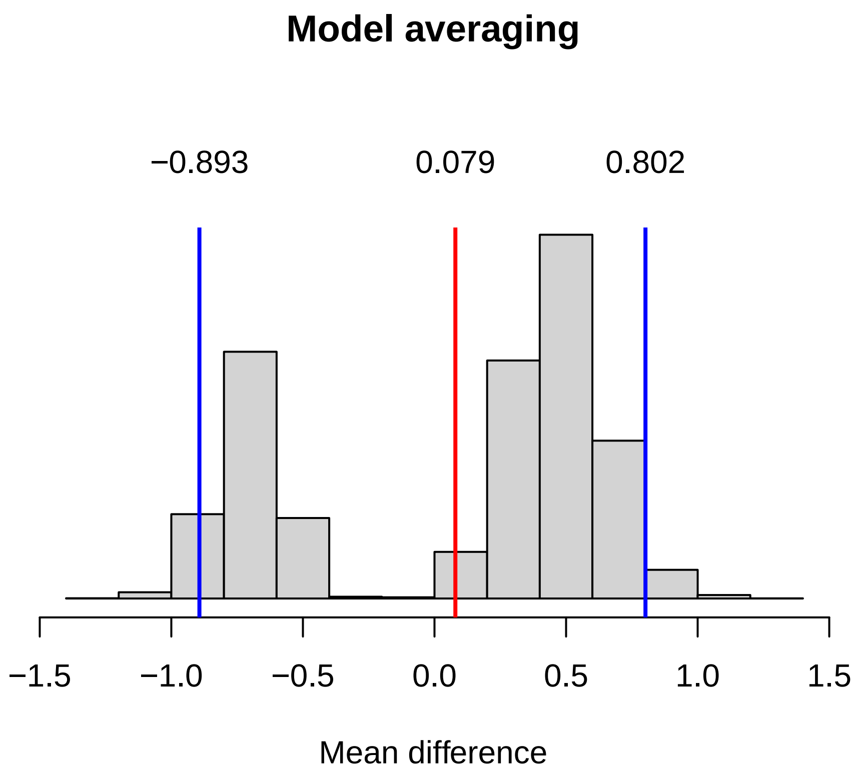
These three clusters can be subject to Bayesian model averaging which returns an averaged mean difference of the two countries. Here we will assume a completely naive weighting scheme by equal weights for all clusters. The resulting averaged mean difference is shown in a plot which illustrates the vastly different results depending on the chosen anchor set. One cluster yields an inverse result compared to the other two clusters, leveling out a mean difference on average.

```
bma <- modelAveraging(res_clusterItems,
                      weights = rep(1/3, 3),
                      iter = 40000) # Runs long. Reduce for tests, if needed.
#> ### Model for cluster 1. Total time is estimated...
#> Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. S
#> http://mc-stan.org/misc/warnings.html#bfmi-low
#> Warning: Examine the pairs() plot to diagnose sampling problems
#>
#> ### Model for cluster 2. Estimated time at finish: 03:14
#> recompiling to avoid crashing R session
#> Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. S
#> http://mc-stan.org/misc/warnings.html#bfmi-low

#> Warning: Examine the pairs() plot to diagnose sampling problems
#>
#> ### Model for cluster 3. Estimated time at finish: 04:54
#> recompiling to avoid crashing R session
#> Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. S
#> http://mc-stan.org/misc/warnings.html#bfmi-low

#> Warning: Examine the pairs() plot to diagnose sampling problems
#>
#> Maximum Rhat (aka PSR): 1.01 [recommended: < 1.05]
#> Minimum effective sample size (aka N_eff): 457 [recommended: > 400]
#>
#> Mean difference in the latent variable after Bayesian model averaging:
#>
#>  2.5% cred. int. -0.893
```

```
#> mean                0.079
#> 97.5% cred. int.   0.802
plotAverage(bma)
```



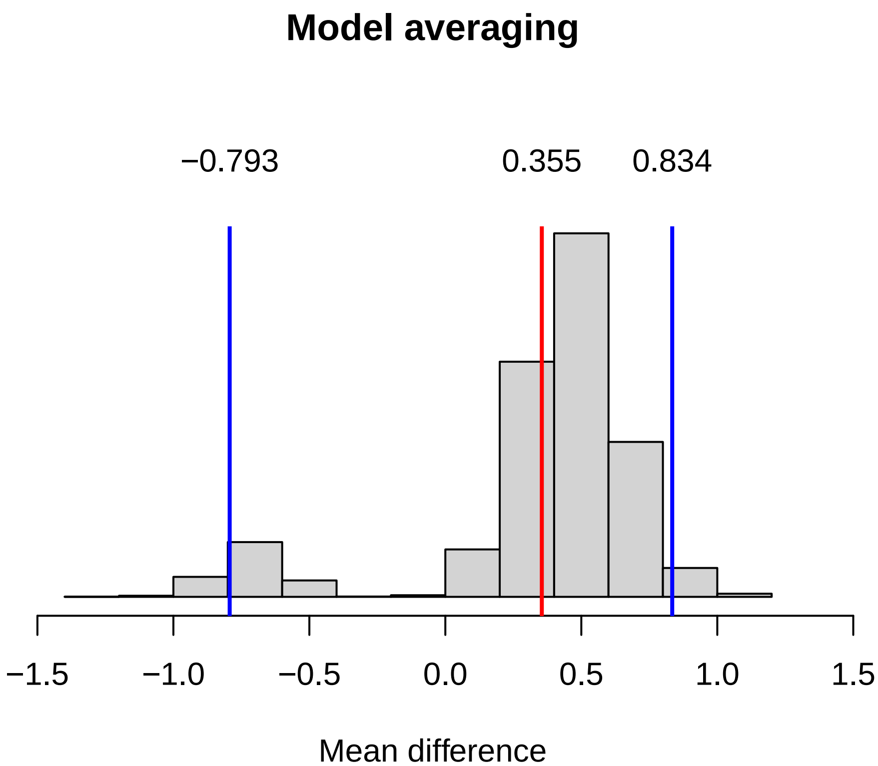
A specific partial model from the Bayesian analysis can be accessed by getModel. We can see that it is cluster three whose items yield a negative mean difference.

```
getModel(bma, which = 3) # e.g. for cluster 3 as anchor
#>                           mean se_mean    sd   2.5%  97.5%     n_eff  Rhat
#> mean_difference         -0.713   0.003 0.123 -0.973 -0.484  1506.658 1.002
#> variance_ratio           0.580   0.006 0.124  0.367  0.854   491.923 1.000
#> item_discrimination[1,1]  1.278   0.002 0.237  0.862  1.791 18625.014 1.000
#> item_discrimination[1,2]  1.365   0.014 0.455  0.642  2.403  1113.398 1.000
#> item_discrimination[2,1]  1.695   0.003 0.352  1.125  2.491 10715.492 1.000
#> item_discrimination[2,2]  3.218   0.031 1.009  1.679  5.610  1056.200 1.000
#> item_discrimination[3,1]  1.317   0.002 0.268  0.849  1.903 16390.965 1.000
#> item_discrimination[3,2]  1.986   0.019 0.665  0.937  3.521  1178.270 1.000
#> item_discrimination[4,1]  0.521   0.001 0.153  0.230  0.831 21710.366 1.000
#> item_discrimination[4,2]  0.763   0.006 0.308  0.243  1.449  2505.059 1.000
#> item_discrimination[5,1]  0.788   0.001 0.123  0.559  1.043  7739.547 1.000
#> item_discrimination[5,2]  0.788   0.001 0.123  0.559  1.043  7739.547 1.000
#> item_discrimination[6,1]  0.858   0.001 0.173  0.541  1.216 23854.291 1.000
#> item_discrimination[6,2]  3.371   0.044 1.239  1.633  6.446   789.389 1.001
#> item_discrimination[7,1]  1.532   0.009 0.301  1.013  2.192  1136.131 1.001
#> item_discrimination[7,2]  1.532   0.009 0.301  1.013  2.192  1136.131 1.001
#> item_difficulty[1,1]    -1.042   0.001 0.145 -1.342 -0.774 22878.936 1.000
```

```
#> item_difficulty[1,2]     -2.533   0.009 0.419 -3.479 -1.848  2403.277 1.000
#> item_difficulty[2,1]     -1.539   0.002 0.220 -2.030 -1.171 10955.159 1.000
#> item_difficulty[2,2]     -5.225   0.019 1.065 -7.733 -3.590  3082.859 1.000
#> item_difficulty[3,1]     -2.021   0.001 0.215 -2.485 -1.643 23378.296 1.000
#> item_difficulty[3,2]     -4.248   0.012 0.694 -5.823 -3.119  3187.013 1.000
#> item_difficulty[4,1]      0.796   0.000 0.104  0.597  1.004 59781.893 1.000
#> item_difficulty[4,2]      0.063   0.004 0.237 -0.464  0.466  3505.962 1.000
#> item_difficulty[5,1]     -0.191   0.001 0.094 -0.382 -0.012  5595.862 1.000
#> item_difficulty[5,2]     -0.191   0.001 0.094 -0.382 -0.012  5595.862 1.000
#> item_difficulty[6,1]     -0.447   0.001 0.108 -0.662 -0.239 38348.632 1.000
#> item_difficulty[6,2]     -3.491   0.030 1.079 -6.125 -1.982  1317.420 1.000
#> item_difficulty[7,1]     -2.337   0.004 0.234 -2.851 -1.937  3543.864 1.000
#> item_difficulty[7,2]     -2.337   0.004 0.234 -2.851 -1.937  3543.864 1.000
```

Other weights can be applied fast to an already estimated modelAveraging object:

```
bma2 <- modelAveraging(res_modelAveraging = bma,
                       weights = c(0.45, 0.45, 0.1))
#>
#> Maximum Rhat (aka PSR): 1.01 [recommended: < 1.05]
#> Minimum effective sample size (aka N_eff): 457 [recommended: > 400]
#>
#> Mean difference in the latent variable after Bayesian model averaging:
#>
#>  2.5% cred. int. -0.793
#> mean              0.355
#> 97.5% cred. int.  0.834
plotAverage(bma2)
```


Model averaging
−0.793
0.355
0.834
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Mean difference