

Using measurementInvariance

Daniel Schulze

Version 0.3.1

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
- plyr
- 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.

All functions deal with unidimensional models only. The typical workflow is intended to be testMI() -> clusterItems() -> either partialMI() or modelAveraging(). However, the functions can also be used independently.

Continuous data (& partial MI)

Here we are taking data from the Holzinger-Swinefort (1939) example on cognitive tests. We will use the “Speed” 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 interested in establishing strong MI for the purpose of group comparison. The summary function gives an overview over model fit of sequentially tested MI models as well as their comparison.

```
speed <- c("t10", "t11", "t12", "t13")

testMIspeed <- testMI(items = speed,
                      group = "sex",
                      data = Data,
                      MIlevel = "strong")

summary(testMIspeed)
#> Two group model with:
#> Female Male total
#>    155   146   301
#>
#>
#> MI level          configural    weak strong
#> chi2              17.26   26.24  51.09
#> df                4       7    10
#> p                 0.002  0.000  0.000
#> CFI               0.950  0.928  0.846
#> RMSEA             0.148  0.135  0.165
#> RMSEA 90% lower   0.084  0.084  0.122
#> SRMR              0.038  0.065  0.086
#> diff chi2         8.796 26.244
#> diff df           3      3
#> diff p            0.032  0
#> diff CFI          -0.022 -0.082
#> diff RMSEA        -0.013  0.03
#> diff SRMR         0.027  0.021
#> Effect (DIF range) 0.688  0.565
#>
#> package          lavaan
#> estimator        MLR
#> item type        continuous
#> item missings     none
#> standardized     yes
#>
#> Use getModel() to access parameter estimates or to further process the results.
```

We identify (borderline) issues with weak and definite problems with strong MI (for cut-offs see Chen, 2007).

We proceed to identify item clusters for which MI holds. Most importantly, we need to tell the clusterItem()

function, for which parameter types there were issues with MI. In this case it's both loadings and difficulties (aka intercepts).

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.

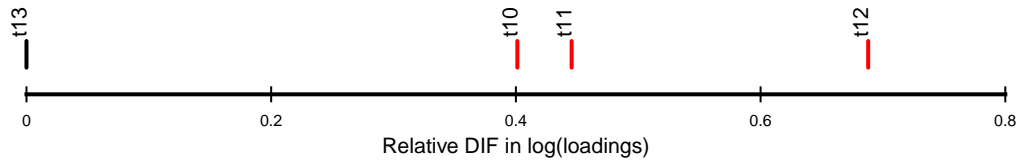
```
clusterItemsSpeed <- clusterItems(res_testMI = testMIspeed,
                                  clusterWhat = c("loadings", "difficulties"),
                                  method = "threshold",
                                  loadThreshold = 0.3,
                                  intThreshold = 0.5)

summary(clusterItemsSpeed,
         order = "clusters")
#> Clustering by threshold criterion with load threshold 0.5 and intercept threshold NA.
#>
#> 3 clusters found after clustering loadings and difficulties
#>      cluster
#> t13      1
#> t10      2
#> t11      2
#> t12      3
```

Three clusters are found. In order to better understand the clustering process, which takes two steps by first clustering item loadings and then item difficulties, a plot can be requested.

```
plotCluster(clusterItemsSpeed,
            showLegend = TRUE)
```

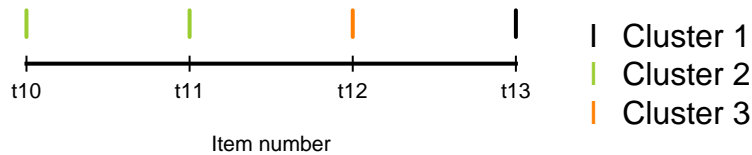
1st step: loading clusters



2nd clustering step: difficulties



Cluster summary



Alternatively to the threshold criterion, a significance test can be used as stopping criterion in the clustering.

```
clusterItemsSpeed_p <- clusterItems(res_testMI = testMIspeed,
                                     clusterWhat = c("loadings", "difficulties"),
                                     method = "sigTest",
                                     alphaValue = 0.05)

summary(clusterItemsSpeed_p,
        order = "clusters")
#> Clustering by sign. test with p of 0.05.
#>
#> 4 clusters found after clustering loadings and difficulties
#>   cluster
#> t13      1
#> t10      2
#> t11      3
#> t12      4
```

Here, four clusters are found. The items differ strong enough that they do not pair up at all according to the significance test.

After inspection of the items we might decide for going with cluster 2 of the threshold analysis as anchor items. Hence we call the `partialMI()` function. It directly displays the model estimates.

```
partialMI(res_clusterItems = clusterItemsSpeed,
          anchor = 2) # cluster number
#> lavaan 0.6-8 ended normally after 32 iterations
#>
#>   Estimator                               ML
```

```

#> Optimization method NLMINB
#> Number of model parameters 26
#> Number of equality constraints 4
#>
#> Number of observations per group:
#> Male 146
#> Female 155
#>
#> Model Test User Model:
#> Standard Robust
#> Test Statistic 22.273 20.489
#> Degrees of freedom 6 6
#> P-value (Chi-square) 0.001 0.002
#> Scaling correction factor 1.087
#> Yuan-Bentler correction (Mplus variant)
#> Test statistic for each group:
#> Male 12.298 11.313
#> Female 9.975 9.176
#>
#> Model Test Baseline Model:
#>
#> Test statistic 293.094 278.690
#> Degrees of freedom 12 12
#> P-value 0.000 0.000
#> Scaling correction factor 1.052
#>
#> User Model versus Baseline Model:
#>
#> Comparative Fit Index (CFI) 0.942 0.946
#> Tucker-Lewis Index (TLI) 0.884 0.891
#>
#> Robust Comparative Fit Index (CFI) 0.944
#> Robust Tucker-Lewis Index (TLI) 0.888
#>
#> Loglikelihood and Information Criteria:
#>
#> Loglikelihood user model (H0) -1551.952 -1551.952
#> Scaling correction factor 0.894
#> for the MLR correction
#> Loglikelihood unrestricted model (H1) -1540.816 -1540.816
#> Scaling correction factor 1.063
#> for the MLR correction
#>
#> Akaike (AIC) 3147.905 3147.905
#> Bayesian (BIC) 3229.461 3229.461
#> Sample-size adjusted Bayesian (BIC) 3159.690 3159.690
#>
#> Root Mean Square Error of Approximation:
#>
#> RMSEA 0.134 0.127
#> 90 Percent confidence interval - lower 0.078 0.072
#> 90 Percent confidence interval - upper 0.196 0.186
#> P-value RMSEA <= 0.05 0.010 0.014

```

```

#>
#> Robust RMSEA 0.132
#> 90 Percent confidence interval - lower 0.072
#> 90 Percent confidence interval - upper 0.197
#>
#> Standardized Root Mean Square Residual:
#>
#> SRMR 0.045 0.045
#>
#> 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|)
#> Factor =~
#> t10 (.p1.) 0.608 0.067 9.015 0.000
#> t11 (.p2.) 0.666 0.087 7.698 0.000
#> t12 0.815 0.115 7.086 0.000
#> t13 0.517 0.078 6.622 0.000
#>
#> Intercepts:
#> Estimate Std.Err z-value P(>|z|)
#> .t10 (.10.) -0.168 0.070 -2.391 0.017
#> .t11 (.11.) -0.200 0.077 -2.588 0.010
#> .t12 0.043 0.091 0.478 0.633
#> .t13 -0.047 0.073 -0.642 0.521
#> Factor 0.000
#>
#> Variances:
#> Estimate Std.Err z-value P(>|z|)
#> .t10 0.469 0.068 6.878 0.000
#> .t11 0.507 0.090 5.620 0.000
#> .t12 0.533 0.130 4.090 0.000
#> .t13 0.521 0.073 7.158 0.000
#> Factor 1.000
#>
#>
#> Group 2 [Female]:
#>
#> Latent Variables:
#> Estimate Std.Err z-value P(>|z|)
#> Factor =~
#> t10 (.p1.) 0.608 0.067 9.015 0.000
#> t11 (.p2.) 0.666 0.087 7.698 0.000
#> t12 0.635 0.113 5.601 0.000
#> t13 0.797 0.158 5.056 0.000
#>

```

```

#> Intercepts:
#>               Estimate Std.Err z-value P(>|z|)
#>   .t10      (.10.)  -0.168   0.070  -2.391   0.017
#>   .t11      (.11.)  -0.200   0.077  -2.588   0.010
#>   .t12                -0.407   0.110  -3.717   0.000
#>   .t13                -0.416   0.145  -2.870   0.004
#>   Factor                0.577   0.167   3.449   0.001
#>
#> Variances:
#>               Estimate Std.Err z-value P(>|z|)
#>   .t10                0.736   0.092   7.985   0.000
#>   .t11                0.552   0.092   6.020   0.000
#>   .t12                0.427   0.064   6.648   0.000
#>   .t13                0.593   0.105   5.636   0.000
#>   Factor                0.938   0.320   2.930   0.003
#> lavaan 0.6-8 ended normally after 32 iterations
#>
#> Estimator                      ML
#> Optimization method          NLMINB
#> Number of model parameters          26
#> Number of equality constraints         4
#>
#> Number of observations per group:
#>   Male                      146
#>   Female                    155
#>
#> Model Test User Model:
#>
#>               Standard      Robust
#> Test Statistic      22.273      20.489
#> Degrees of freedom         6         6
#> P-value (Chi-square)      0.001      0.002
#> Scaling correction factor      1.087
#>   Yuan-Bentler correction (Mplus variant)
#> Test statistic for each group:
#>   Male      12.298      11.313
#>   Female     9.975      9.176

```

Alternatively, we might have some source other than the clustering for finding anchor candidates. `partialMI()` also takes item names as anchors and does not only rely on `clusterItems()`. We can also put in all model information.

```

partialMI(items = speed,
  group = "sex",
  data = Data,
  MIlevel = "strong",
  partialMIwhat = c("loadings", "difficulties"), # equivalent to clusterWhat above
  anchor = c("t10", "t11")) # item names
#> lavaan 0.6-8 ended normally after 32 iterations
#>
#> Estimator                      ML
#> Optimization method          NLMINB
#> Number of model parameters          26
#> Number of equality constraints         4
#>

```

```

#> Number of observations per group:
#>   Male      146
#>   Female    155
#>
#> Model Test User Model:
#>
#>           Standard      Robust
#> Test Statistic      22.273    20.489
#> Degrees of freedom         6         6
#> P-value (Chi-square)      0.001    0.002
#> Scaling correction factor
#>   Yuan-Bentler correction (Mplus variant)
#> Test statistic for each group:
#>   Male      12.298    11.313
#>   Female      9.975     9.176
#>
#> Model Test Baseline Model:
#>
#> Test statistic      293.094    278.690
#> Degrees of freedom      12         12
#> P-value            0.000    0.000
#> Scaling correction factor
#>
#> User Model versus Baseline Model:
#>
#> Comparative Fit Index (CFI)      0.942    0.946
#> Tucker-Lewis Index (TLI)        0.884    0.891
#>
#> Robust Comparative Fit Index (CFI)      0.944
#> Robust Tucker-Lewis Index (TLI)        0.888
#>
#> Loglikelihood and Information Criteria:
#>
#> Loglikelihood user model (H0)      -1551.952    -1551.952
#> Scaling correction factor
#>   for the MLR correction
#> Loglikelihood unrestricted model (H1)      -1540.816    -1540.816
#> Scaling correction factor
#>   for the MLR correction
#>
#> Akaike (AIC)      3147.905    3147.905
#> Bayesian (BIC)      3229.461    3229.461
#> Sample-size adjusted Bayesian (BIC)      3159.690    3159.690
#>
#> Root Mean Square Error of Approximation:
#>
#> RMSEA      0.134    0.127
#> 90 Percent confidence interval - lower      0.078    0.072
#> 90 Percent confidence interval - upper      0.196    0.186
#> P-value RMSEA <= 0.05      0.010    0.014
#>
#> Robust RMSEA      0.132
#> 90 Percent confidence interval - lower      0.072
#> 90 Percent confidence interval - upper      0.197

```



```

#>
#> Standardized Root Mean Square Residual:
#>
#> SRMR 0.045 0.045
#>
#> 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|)
#> Factor =~
#> t10 (.p1.) 0.608 0.067 9.015 0.000
#> t11 (.p2.) 0.666 0.087 7.698 0.000
#> t12 0.815 0.115 7.086 0.000
#> t13 0.517 0.078 6.622 0.000
#>
#> Intercepts:
#> Estimate Std.Err z-value P(>|z|)
#> .t10 (.10.) -0.168 0.070 -2.391 0.017
#> .t11 (.11.) -0.200 0.077 -2.588 0.010
#> .t12 0.043 0.091 0.478 0.633
#> .t13 -0.047 0.073 -0.642 0.521
#> Factor 0.000
#>
#> Variances:
#> Estimate Std.Err z-value P(>|z|)
#> .t10 0.469 0.068 6.878 0.000
#> .t11 0.507 0.090 5.620 0.000
#> .t12 0.533 0.130 4.090 0.000
#> .t13 0.521 0.073 7.158 0.000
#> Factor 1.000
#>
#>
#> Group 2 [Female]:
#>
#> Latent Variables:
#> Estimate Std.Err z-value P(>|z|)
#> Factor =~
#> t10 (.p1.) 0.608 0.067 9.015 0.000
#> t11 (.p2.) 0.666 0.087 7.698 0.000
#> t12 0.635 0.113 5.601 0.000
#> t13 0.797 0.158 5.056 0.000
#>
#> Intercepts:
#> Estimate Std.Err z-value P(>|z|)
#> .t10 (.10.) -0.168 0.070 -2.391 0.017
#> .t11 (.11.) -0.200 0.077 -2.588 0.010

```

```

#>      .t12      -0.407    0.110   -3.717    0.000
#>      .t13      -0.416    0.145   -2.870    0.004
#>      Factor      0.577    0.167    3.449    0.001
#>
#> Variances:
#>           Estimate Std.Err  z-value  P(>|z|)
#>      .t10           0.736   0.092    7.985   0.000
#>      .t11           0.552   0.092    6.020   0.000
#>      .t12           0.427   0.064    6.648   0.000
#>      .t13           0.593   0.105    5.636   0.000
#>      Factor        0.938   0.320    2.930   0.003
#> lavaan 0.6-8 ended normally after 32 iterations
#>
#>      Estimator                      ML
#>      Optimization method          NLMINB
#>      Number of model parameters          26
#>      Number of equality constraints        4
#>
#>      Number of observations per group:
#>      Male                          146
#>      Female                        155
#>
#> Model Test User Model:
#>
#>           Standard      Robust
#>      Test Statistic      22.273    20.489
#>      Degrees of freedom         6         6
#>      P-value (Chi-square)      0.001    0.002
#>      Scaling correction factor      1.087
#>      Yuan-Bentler correction (Mplus variant)
#>      Test statistic for each group:
#>      Male                     12.298    11.313
#>      Female                    9.975     9.176

```

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.

```

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 ...

testMifims <- testMI(items = colnames(dataDich[1:7]),
  group = "country",
  data = dataDich,
  MIlevel = "strong",
  itemType = "dichotomous",
  dichModel = "2PL")

summary(testMifims)
#> Two group model with:
#> group1 group2 total
#> 500 500 1000
#>
#>
#> 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
#> Effect (DIF range) 1.158    2.259
#>
#> package          mirt
#> estimator         EM
#> item type         2PL
#> item missings     none
#>
#> Use getModel() to access parameter estimates or to further process the results.

```

Contrary to the first example, we find no issues with weak MI, but a clear violation of strong MI.

If we want to have a look at a specific model estimated by `testMI()`, we can 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`). Here, the weak MI model is requested.

```

resWeak <- getModel(testMifims,
  which = "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 difficulties (or thresholds in IRT terms), we find three item clusters.

```

clusterItemsFIMS <- clusterItems(res_testMI = testMifims,
                                clusterWhat = "difficulties",
                                method = "threshold",
                                intThreshold = 0.6)

summary(clusterItemsFIMS)
#> Clustering by threshold criterion with load threshold NA and intercept threshold NA.
#>
#> 3 clusters found after clustering difficulties
#>      cluster
#> M1PTI1      1
#> M1PTI2      1
#> M1PTI3      1
#> M1PTI12     2
#> M1PTI18     2
#> M1PTI14     3
#> M1PTI23     3

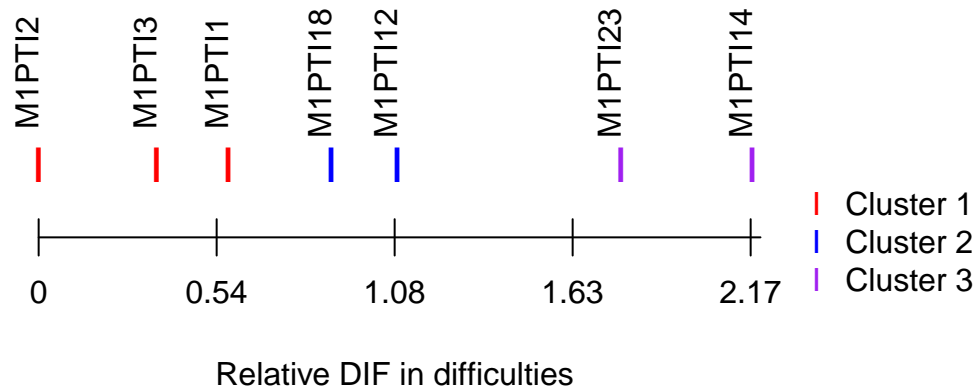
```

Again, the clusters can be plotted. In this case, it is only a single plot as only difficulties were clustered.

```

plotCluster(clusterItemsFIMS,
            showLegend = TRUE)

```



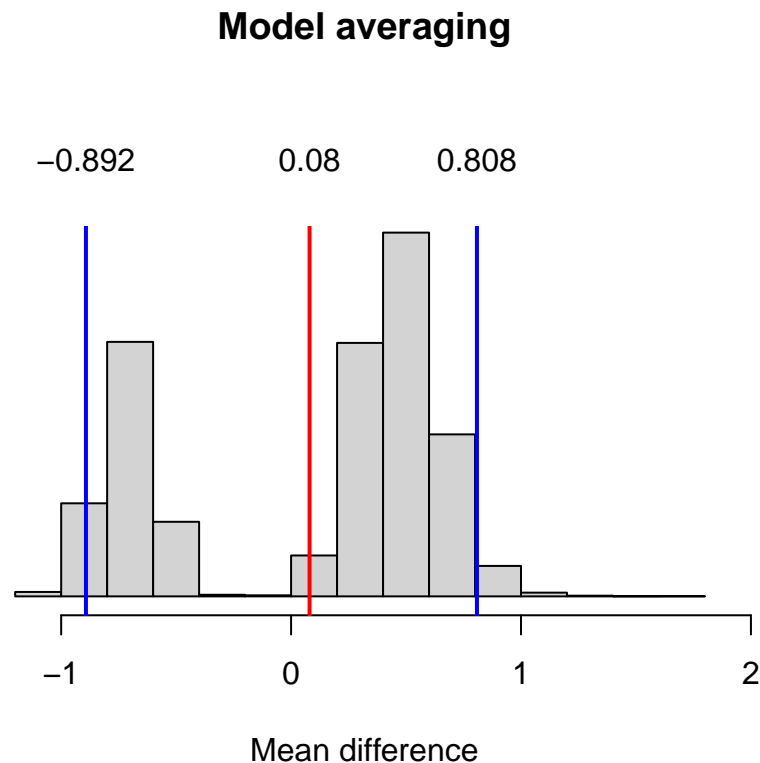
The relative DIF values which form the basis of the plot can be printed as well.

```
printDIF(clusterItemsFIMS)
#> Relative DIF of loadings in configural model, resulting in 1 cluster.
#> NULL
#>
#> Relative DIF of difficulties for loading cluster 1
#>      M1PTI1    M1PTI2    M1PTI3    M1PTI12    M1PTI14    M1PTI18
#> [1,] 0.0000000 0.5774560 0.2179689 -0.5153366 -1.5969640 -0.3141165
#> [2,] -0.5774560 0.0000000 -0.3594871 -1.0927925 -2.1744200 -0.8915725
#> [3,] -0.2179689 0.3594871 0.0000000 -0.7333055 -1.8149329 -0.5320854
#> [4,] 0.5153366 1.0927925 0.7333055 0.0000000 -1.0816274 0.2012201
#> [5,] 1.5969640 2.1744200 1.8149329 1.0816274 0.0000000 1.2828475
#> [6,] 0.3141165 0.8915725 0.5320854 -0.2012201 -1.2828475 0.0000000
#> [7,] 1.1969269 1.7743829 1.4148958 0.6815903 -0.4000371 0.8828104
#>      M1PTI23
#> [1,] -1.1969269
#> [2,] -1.7743829
#> [3,] -1.4148958
#> [4,] -0.6815903
#> [5,] 0.4000371
#> [6,] -0.8828104
#> [7,] 0.0000000
```

The 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(clusterItemsFIMS,
  weights = rep(1/3, 3),
  iter = 40000) # Runs long. Reduce for tests, if needed.
#> ### Model for cluster 1. Total time is estimated...
#>
#> ### Model for cluster 2. Estimated time at finish: 03:06
#>
#> ### Model for cluster 3. Estimated time at finish: 04:53
#>
#> Maximum Rhat (aka PSR): 1.01 [recommended: < 1.05]
#> Minimum effective sample size (aka N_eff): 253 [recommended: > 400]
#>
#> Mean difference in the latent variable after Bayesian model averaging:
#>
#> 2.5% cred. int. -0.892
#> mean          0.080
#> 97.5% cred. int. 0.808
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      -0.712   0.003 0.120 -0.955 -0.486  1941.887 1.001
#> variance_ratio       0.574   0.005 0.122  0.371  0.851   555.898 1.002
#> item_discrimination[1,1] 1.279   0.002 0.236  0.861  1.783 20695.568 1.000
#> item_discrimination[1,2] 1.370   0.012 0.453  0.638  2.409  1322.129 1.001
#> item_discrimination[2,1] 1.684   0.003 0.341  1.124  2.452 13260.053 1.000
#> item_discrimination[2,2] 3.233   0.028 1.000  1.692  5.611  1254.273 1.001
#> item_discrimination[3,1] 1.316   0.002 0.265  0.852  1.895 15634.337 1.000
#> item_discrimination[3,2] 2.002   0.018 0.671  0.950  3.546  1361.019 1.001
#> item_discrimination[4,1] 0.522   0.001 0.152  0.236  0.833 27221.575 1.000
#> item_discrimination[4,2] 0.768   0.006 0.311  0.244  1.474  2670.204 1.001
#> item_discrimination[5,1] 0.790   0.001 0.125  0.557  1.051  9004.670 1.000
#> item_discrimination[5,2] 0.790   0.001 0.125  0.557  1.051  9004.670 1.000
#> item_discrimination[6,1] 0.858   0.001 0.173  0.541  1.219 27392.224 1.000
#> item_discrimination[6,2] 3.398   0.039 1.202  1.655  6.376   941.860 1.001
#> item_discrimination[7,1] 1.539   0.008 0.299  1.031  2.193  1489.913 1.002
#> item_discrimination[7,2] 1.539   0.008 0.299  1.031  2.193  1489.913 1.002
#> item_difficulty[1,1]    -1.041   0.001 0.145 -1.343 -0.772 21814.657 1.000
#> item_difficulty[1,2]    -2.535   0.009 0.420 -3.476 -1.841  2284.293 1.000
#> item_difficulty[2,1]    -1.532   0.002 0.214 -2.003 -1.167 12615.913 1.000
#> item_difficulty[2,2]    -5.232   0.019 1.061 -7.749 -3.585  2984.421 1.000
#> item_difficulty[3,1]    -2.018   0.001 0.212 -2.473 -1.644 24106.979 1.000
#> item_difficulty[3,2]    -4.260   0.013 0.705 -5.872 -3.125  3092.466 1.000
#> item_difficulty[4,1]     0.797   0.000 0.104  0.597  1.004 59183.957 1.000
#> item_difficulty[4,2]     0.058   0.004 0.241 -0.479  0.461  3055.611 1.000
#> item_difficulty[5,1]    -0.192   0.001 0.097 -0.388 -0.008  6291.133 1.000
#> item_difficulty[5,2]    -0.192   0.001 0.097 -0.388 -0.008  6291.133 1.000
#> item_difficulty[6,1]    -0.447   0.001 0.108 -0.662 -0.240 36664.040 1.000
#> item_difficulty[6,2]    -3.513   0.029 1.067 -6.114 -1.984  1363.595 1.000
#> item_difficulty[7,1]    -2.342   0.004 0.235 -2.859 -1.936  3501.995 1.001
#> item_difficulty[7,2]    -2.342   0.004 0.235 -2.859 -1.936  3501.995 1.001

```

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): 253 [recommended: > 400]
#>
#> Mean difference in the latent variable after Bayesian model averaging:
#>
#> 2.5% cred. int. -0.796
#> mean           0.355
#> 97.5% cred. int. 0.836
plotAverage(bma2)

```


Model averaging

