ESPECIFICAÇÕES/CONFIGURAÇÕES PARA A EQUALIZAÇÃO NO MIRT

Relatório do Projeto

Plano de Trabalho Complementar ao Plano CGMEB 1211853, que se refere a estudos e reuniões técnicas para subsidiar e orientar os procedimentos psicométricos que possibilitam a comparabilidade entre o Saeb e as avaliações estaduais.

Comissão de Assessoramento em Estatística e Psicometria do INEP

(Nomeação pela Portaria nº 722 de 30 de agosto de 2019)

Sumário

1. INTRODUÇÃO

Em avaliações educacionais e construção de indicadores de forma gerai, seja de nível socioeconômico ou similares, é fundamental que os resultados estejam em uma escala única na qual tantos os itens quantos as proficiências dos indivíduos sejam diretamente comparáveis. Colocar os resultados em uma mesma escala significa ***Equalizar***, equiparar, tornar comparável, e esta etapa pode ser realizada posterior ou conjuntamente à estimação dos parâmetros.

Especificamente em modelos da Teoria da Resposta ao Item (TRI), os métodos de estimação mais utilizados quando todos os parâmetros dos itens de uma única prova devem ser estimados representam o caso mais simples. No entanto, esta é apenas uma das possíveis situações que na prática podemos encontrar.

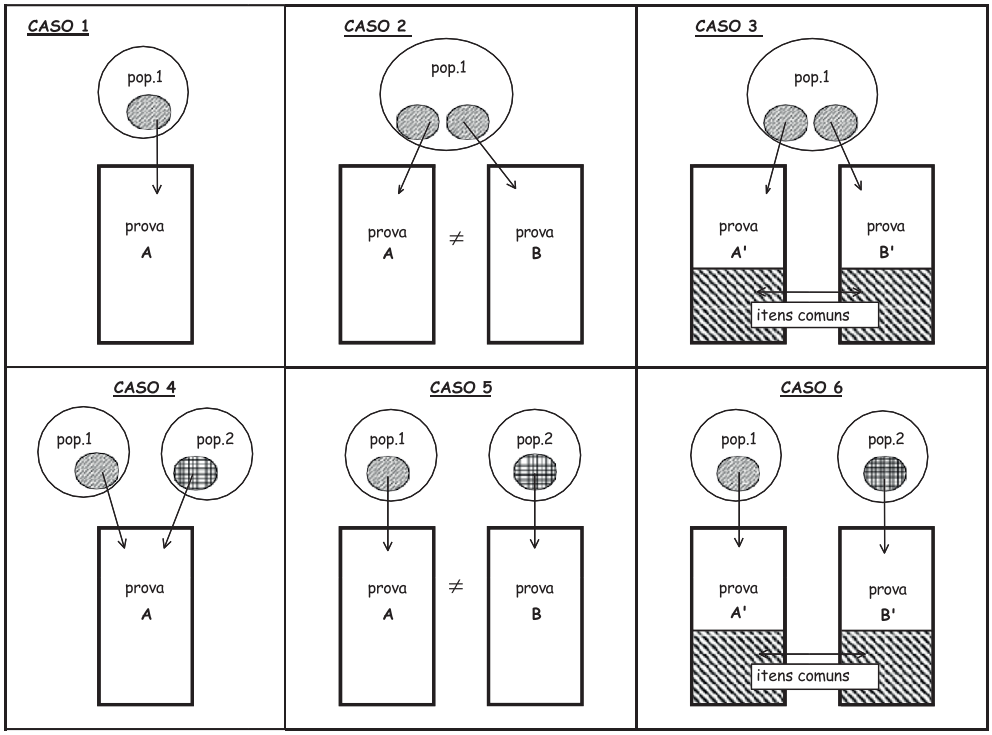
1.1. SITUAÇÕES DE EQUALIZAÇÃO

A seguir, listaremos os 6 casos possíveis, quanto ao número de grupos e de tipos de prova envolvidos. Esses casos estão esquematizados na Figura 1, exemplificando para apenas 2 populações (grupos) envolvidos, cabendo ressaltar que em muitas situações práticas há diversos cadernos de prova em cada população.

1. Um único grupo fazendo uma única prova.
2. Um único grupo, dividido em dois subgrupos, fazendo duas provas, totalmente distintas (nenhum item comum).
3. Um único grupo, dividido em dois subgrupos, fazendo duas provas, apenas parcialmente distintas, ou seja, com alguns itens comuns.
4. Dois grupos fazendo uma única prova.

5. Dois grupos fazendo duas provas, totalmente distintas (nenhum item comum).
6. Dois grupos fazendo duas provas, apenas parcialmente distintas, ou seja, com alguns itens comuns.

Figura 1. *Representação de 6 situações quanto ao número de grupos e tipos de provas*



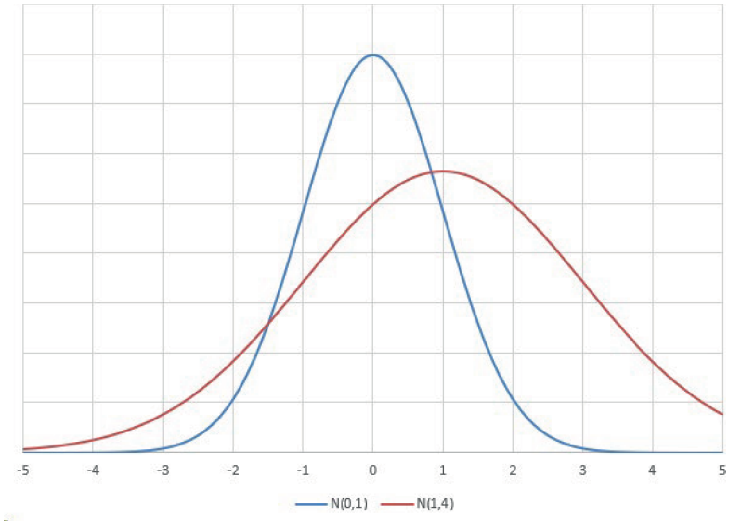
Fonte: Andrade, Tavares, Valle (2000)

1.2. POPULAÇÃO E ITENS

A proficiência de um indivíduo é uma Variável Latente, então é não observada diretamente, mas sim através de respostas a um teste na área da proficiência. Cada indivíduo de uma população terá sua proficiência, formando uma distribuição

estatística. Na Figura 2 temos duas distribuições normais, com médias 0 e 1, e desvio- padrão 1 e 2 (variância 4), respectivamente.

Figura 2. Ilustração de distribuições normais distintas



Fonte: Elaborado pelo autor

Pela Figura 2 vemos que um item com dificuldade média (0) para a população 1, com distribuição N(0,1), estará abaixo da dificuldade média (1) da população 2. É uma conclusão possível somente se tivermos uma escala única.

Durante o processo de estimação dos parâmetros dos itens, uma das populações é denominada População Referência, cuja distribuição será a N(0,1), salvo se houver outra especificação.

Existem dois tipos básicos de equalização: a equalização ***via população*** e a equalização via ***itens comuns***. Isto significa que há duas maneiras de colocar parâmetros, tanto de itens quanto de proficiências, numa mesma métrica: na primeira usamos o fato de que se um único grupo de respondentes é submetido a provas distintas, basta que todos os itens sejam calibrados conjuntamente para termos a garantia de que todos estarão na mesma métrica. Já na equalização via itens comuns, a

garantia de que as populações envolvidas terão seus parâmetros em uma única escala será dada pelos itens comuns entre as populações, que servirão de ligação entre elas.

1.3. DOIS GRUPOS FAZENDO DUAS PROVAS PARCIALMENTE DISTINTAS

A situação que merece mais atenção é o caso em que dois grupos são submetidos a duas provas diferentes, mas que têm alguns itens comuns. Temos assim um exemplo de equalização via itens comuns. Este caso representa o melhor exemplo do uso e da importância da equalização e sem dúvida, ilustra o maior avanço da TRI sobre a Teoria Clássica. O uso de itens comuns entre provas distintas aplicadas a populações distintas permite que todos os parâmetros estejam na mesma escala ao final dos processos de estimação, possibilitando comparações e a construção de “escalas do conhecimento” interpretáveis, que são de grande importância na área educacional. Importante ressaltar que apenas alguns dos itens, e não necessariamente a prova toda, fazem a ligação entre as duas populações envolvidas.

1.4. QUANDO ALGUNS ITENS SÃO NOVOS E OUTROS JÁ ESTÃO CALIBRADOS

Neste caso, temos itens “novos” e itens já calibrados, ou seja, desejamos calibrar alguns itens e manter os parâmetros de outros, que já foram calibrados anteriormente. Este também é uma situação que está tipicamente ligada `a criação de bancos de itens. Isto porque um banco de itens está continuamente em formação, ou seja, é bastante comum estarmos interessados em acrescentar novos itens ao conjunto que já se encontra no banco (assim como também é comum a retirada de itens do banco). Neste caso, o problema fundamental é garantir que os itens novos sejam calibrados na mesma métrica em que estão os outros itens do banco.

Na prática, este é um problema de solução mais complexa do que possa parecer em princípio. Isto porque é indispensável o uso de programas computacionais especificamente desenvolvidos para a análise de itens via TRI e esses

1.5. SOBRE O PROCESSO DE ESTIMAÇÃO

O método estatístico padrão de estimação dos parâmetros dos itens é a Máxima Verossimilhança Marginal (MVM) via Algoritmo EM (ver Andrade, Tavares & Valle, 2000). De forma controlar as mudanças das estimativas dos parâmetros dos itens, costuma-se adotar prioris para alguns parâmetros dos itens, particularmente o parâmetro de discriminação e parâmetro de acerto casual, pois ambos têm restrição de suporte ($a>0$ e c no intervalo [0,1]). Na estimação por MMV, adota-se distribuições estatísticas para as destruições das proficiências, de forma que as Equações de Estimação só terão os parâmetros dos itens para estimar numericamente. A esta etapa denomina-se Calibração (*Calibration*). Após estimados os parâmetros dos itens, seque a etapa de estimação das proficiências, denominada de *Scoring*.

2. EQUALIZAÇÃO EM SOFTWARES COMERCIAIS

Em softwares comerciais (proprietários) como BILOG-MG, talvez o mais tradicional, IRTPRO, PARSCALE, XCALIBRE, MULTILOG, dentre muitos outros, o processo de calibração é o MMV em duas etapas, e com prioris para controlar os saltos das estimativas dos parâmetros.

2.1. FORMATO DOS DADOS

O formato em que os vetores de resposta são disponibilizados no arquivo de dados é determinante para definição da sintaxe usada para calibração. No ***Formato Compacto***, para cada indivíduo devemos informar qual caderno foi respondido, enquanto no ***Formato Aberto*** haverá tantas colunas de resposta

quanto forem o número de itens. Para exemplificar, no SAEB-EM (Ensino Médio) cada estudante responderá a 2 cadernos de 13 itens, totalizando 26 itens em Matemática, de um total de 91 itens. No formato compacto o vetor de resposta terá tamanho 26, enquanto no formato aberto terá tamanho 91.

2.1. ESPECIFICAÇÃO DOS CADERNOS E GRUPOS

Restringindo ao BILOG-MG, a especificação de quais itens que compõem cada caderno é essencial para o formato compacto, e desnecessários ao formato aberto. No entanto, em ambos os casos é fundamental informar a qual grupo o indivíduo pertence.

2.2. FIXANDO ESTIMATIVAS DOS PARÂMETROS DOS ITENS

No BILOG-MG, a fixação (das estimativas) dos parâmetros dos itens é conduzida com o uso de prioris, cuja média corresponde ao valor a ser fixado, e adotando-se um desvio-padrão muito pequeno, como 10^{-3} ou 10^{-4}. Mesmo que seja feita a leitura de um arquivo externo, de extensão PRM, o procedimento continua o mesmo. Neste procedimento as estimativas finais não são necessariamente as mesmas iniciais, podendo apresentar uma variação mínima.

3. IMPLEMENTAÇÃO NO MIRT

O *mirt (Multidimentional Item Response Theory)* é um dos principais pacotes criados para o R (R Core Team, 2023), apresentado em Chalmers (2012), para calibração, scoring, equalização, e muitas outras de análise de dados educacionais.

3.1. COMANDO BÁSICO PARA CALIBRAÇÃO E SCORING

O comando básico do mirt para calibração e scoring quando temos apenas é:

mod = mirt(Dados, 1, itemtype = “3PL”, TOL = 1e-6, quadpts = 40, technical = list(NCYCLES = 500))

E os Argumentos da função são:

- **Dados**: Base de dados
- **3PL**: Modelo Logístico de 3 parâmetros, comum a todos os itens
- **TOL**: TOLerância, se nas etapas do processo de estimação a alteração absoluta máxima dentre todos os itens for abaixo de TOL o processo finaliza com indicação de Convergência.
- **quadpts**: Número de pontos de quadratura para aproximar a N(0,1)
- **NCYCLES**: Número de ciclos EM do processo iterativo.

Cabe ressaltar que o pacote mirt deverá estar **instalado** e **carregado** no R. O comando abaixo faz esta etapa dupla, se necessário: verifica se o mirt está instado, e se não estiver, ele instala; em qualquer caso, ele carrega o pacote.

if(!require(mirt)){install.packages("mirt"); library(mirt)

3.2. TIPOS DE ITEM

Podemos ter um teste com diversos tipos de item, tal como de de Múltipla Escolha e Resposta Construída, e precisará ser informado no formato:

Itemtype=c(“3PL”,”3PL”,”graded”,...)

No Manual do mirt temos a seguinte relação e descrição:

itemtype type of items to be modeled, declared as a vector for each item or a single value which will be recycled for each item. The NULL default assumes that the items follow a graded or 2PL structure, however they may be changed to the following:
• 'Rasch' - Rasch/partial credit model by constraining slopes to 1 and freely

estimating the variance parameters (alternatively, can be specified by applying equality constraints to the slope parameters in 'gpcm'; Rasch, 1960)
• '2PL', '3PL', '3PLu', and '4PL' - 2-4 parameter logistic model, where 3PL estimates the lower asymptote only while 3PLu estimates the upper asymptote only (Lord and Novick, 1968; Lord, 1980)
• '5PL' - 5 parameter logistic model to estimate asymmetric logistic response curves. Currently restricted to unidimensional models
• 'CLL' - complementary log-log link model. Currently restricted to unidimensional models
• 'ULL' - unipolar log-logistic model (Lucke, 2015). Note the use of this itemtype will automatically use a log-normal distribution for the latent traits
• 'graded' - graded response model (Samejima, 1969)
• 'grsm' - graded ratings scale model in the classical IRT parameterization (restricted to unidimensional models; Muraki, 1992)
• 'gpcm' and 'gpcmIRT' - generalized partial credit model in the slopeintercept and classical parameterization. 'gpcmIRT' is restricted to unidimensional models. Note that optional scoring matrices for 'gpcm' are available with the gpcm_mats input (Muraki, 1992)
• 'rsm' - Rasch rating scale model using the 'gpcmIRT' structure (unidimensional only; Andrich, 1978)
• 'nominal' - nominal response model (Bock, 1972)
• 'ideal' - dichotomous ideal point model (Maydeu-Olivares, 2006)
• 'ggum' - generalized graded unfolding model (Roberts, Donoghue, &Laughlin, 2000) and its multidimensional extension
• 'sequential' - multidimensional sequential response model (Tutz, 1990) in slope-intercept form
• 'Tutz' - same as the 'sequential' itemtype, except the slopes are fixed to 1 and the latent variance terms are freely estimated (similar to the 'Rasch' itemtype input)
• 'PC2PL' and 'PC3PL' - 2-3 parameter partially compensatory model. Note that constraining the slopes to be equal across items will reduce the model to Embretson's (a.k.a. Whitely's) multicomponent model (1980).
• '2PLNRM', '3PLNRM', '3PLuNRM', and '4PLNRM' - 2-4 parameter nested logistic model, where 3PLNRM estimates the lower asymptote only while 3PLuNRM estimates the upper asymptote only (Suh and Bolt, 2010)
• 'spline' - spline response model with the bs (default) or the ns function (Winsberg, Thissen, and Wainer, 1984)
• 'monopoly' - monotonic polynomial model for unidimensional tests for dichotomous and polytomous response data (Falk and Cai, 2016)
Additionally, user defined item classes can also be defined using the createItem Function

3.3. ESPECIFICAÇÃO DO MODELO

No mirt, podemos indicar (especificar) o modelo indicando as dimensões (fatores) envolvidas, estimativas iniciais, os itens a serem fixados (estimativas), prioris para controlar o processo iterativo, itens comuns entre grupos, dentre várias outras situações.

3.4. PRIORIS PARA CONTROLAR OS PARÂMETROS

O mirt tem uma sintaxe própria para especificar prioris de controle para os parâmetros, fornecida através de strings. Por exemplo, para 45 parâmetro do ENEM poderíamos adotar $c_i \sim Beta(4,16)$, cujo valor esperado é 4/(4+16) = 1/5 = 0,2, apropriado para 5 alternativas de resposta.

```
model <- 'F = 1-45
    PRIOR = (1-45, g, beta, 4, 16)'
pars <- mirt(Dados,1, model, itemtype = '3PL', pars = 'values')
```

3.5. MÚLTIPLOS GRUPOS (MG)

Quando temos mais de um grupo (digamos **K**), usamos a função do mirt denominada **multipleGroup**. Para usá-la, temos que ter a indicação do grupo na base de dados através do argumento *group*, bem como uma string informando a estrutura de ligação dos itens, com alguns sendo comuns aos grupos ou de parâmetros conhecidos.

A especificação do modelo, incluindo itens comuns. Por exemplo, podemos informar que os itens 1 a 20, dentre os 45 do teste, são iguais entre os grupos.

```
modelo <- mirt.model('F = 1-45
                      CONSTRAINB = (1-20, a1),(1-20, d), (1-20, g)
                      PRIOR = (1-40, a1, lnorm, -0.6550, 1.1445), (1-40, g, norm, -1.386, 0.5)')
```

Há outros quesitos muitos importantes no processo de equalização pelo mirt. No comando abaixo, o número de pontos de quadratura e o número de ciclos EM são proporcionais ao número de grupos.

```
invariance = c(names(Dados),'slopes', 'intercepts', 'free_var','free_means')

mod = multipleGroup(Dados, modelo, itemtype = "3PL", group = Grupos, invariance=invariance, TOL = 1e-6, quadpts = K*40, technical = list(NCYCLES = K*500))
```

No caso dos modelos logísticos de 1 a 5 parâmetros (5PL), a função de resposta é dada por:

$$P(U_{ij} = 1; \theta_j, a_i, b_i, c_i, \gamma_i, d_i) = c_i + (\gamma_i - c_i) \frac{1}{(1 + e^{-Da_i(\theta_j - b_i)})^{\delta_i}}$$

em que o parâmetro $\gamma_i \in (c_i, 1)$ representa a probabilidade de pessoas de alta proficiência não responderem corretamente ao item, enquanto o parâmetro $\delta_i \in$

(0,1] controla a assimetria da função de resposta do item. Quando γ_i = δ_i = 1 temos o 3PL.

3.6. FIXANDO PARÂMETROS DOS ITENS

O mirt tem uma forma muito consistente de fixação de parâmetros, usando os próprios valores na função de verossimilhança, sem alterá-los até o final. Antes de iniciar o processo de estimação, ele monta uma matriz em que constam todas as estimativas iniciais. Essa matriz pode ser obtida com o seguinte comando:

PRM = mirt(data,1, itemtype = "3PL", pars = 'values')

Figura 3. Ilustração da matriz de estimativas iniciais do mirt para o 3PL

	group	item	class	name	parnum	value	lbound	ubound	est	prior.type	prior_1	prior_2
1	all	MT1	dich	a1	1	0.85100000	-Inf	Inf	TRUE	none	NaN	NaN
2	all	MT1	dich	d	2	3.12687643	-Inf	Inf	TRUE	none	NaN	NaN
3	all	MT1	dich	g	3	0.15000000	0e+00	1	TRUE	none	NaN	NaN
4	all	MT1	dich	u	4	1.00000000	0e+00	1	FALSE	none	NaN	NaN
5	all	MT2	dich	a1	5	0.85100000	-Inf	Inf	TRUE	none	NaN	NaN
6	all	MT2	dich	d	6	3.64420486	-Inf	Inf	TRUE	none	NaN	NaN
7	all	MT2	dich	g	7	0.15000000	0e+00	1	TRUE	none	NaN	NaN
8	all	MT2	dich	u	8	1.00000000	0e+00	1	FALSE	none	NaN	NaN
9	all	MT3	dich	a1	9	0.85100000	-Inf	Inf	TRUE	none	NaN	NaN
10	all	MT3	dich	d	10	3.01660660	-Inf	Inf	TRUE	none	NaN	NaN
11	all	MT3	dich	g	11	0.15000000	0e+00	1	TRUE	none	NaN	NaN
12	all	MT3	dich	u	12	1.00000000	0e+00	1	FALSE	none	NaN	NaN
13	all	MT4	dich	a1	13	0.85100000	-Inf	Inf	TRUE	none	NaN	NaN
14	all	MT4	dich	d	14	3.26058726	-Inf	Inf	TRUE	none	NaN	NaN
15	all	MT4	dich	g	15	0.15000000	0e+00	1	TRUE	none	NaN	NaN
16	all	MT4	dich	u	16	1.00000000	0e+00	1	FALSE	none	NaN	NaN
17	all	MT5	dich	a1	17	0.85100000	-Inf	Inf	TRUE	none	NaN	NaN
18	all	MT5	dich	d	18	2.98499839	-Inf	Inf	TRUE	none	NaN	NaN
19	all	MT5	dich	g	19	0.15000000	0e+00	1	TRUE	none	NaN	NaN
20	all	MT5	dich	u	20	1.00000000	0e+00	1	FALSE	none	NaN	NaN

A seleção de quais parâmetros devem ser fixados deve ser feita alterando a coluna **est** da matriz para **FALSE.**

Vale ressaltar que o mirt adota o parâmetro intercepto, $d_i = -a_i b_i$, de forma que estes devem constar na matriz.

As prioris também podem ser especificados na matriz, mas serão usados apenas na estimação dos itens que serão efetivamente calibrados.

O comando para calibração utilizando a matriz alterada é:

mod.f = mirt(Dados,1, itemtype = "3PL", pars = PRM)

A seguir é apresentada uma função completa no R para fixação de parâmetros em um processo de Calibração e Equalização.

```
Estima_Fixos=function(data,  prm){

 cat("\nEstimativas de parametros a serem fixadas:\n"); print(prm); cat("\n")
 prm[,3]=-prm[,2]*prm[,3] # Construindo o INTERCEPTO para o MIRT

 PL=paste0(kPL,"PL")
 pars <- mirt(data,1, itemtype = PL, pars = 'values')

 #trocando os parametros do template do mirt pelos valores fixados, e FALSE para nao estimar
 k=1
 for (i in 1:It){
  ip=4*i-3
  if (i %in% FIXED) {
    lin.a=prm[k,1]*4-3; pars$value[lin.a]=prm[k,2];    pars$est[lin.a]=FALSE
    lin.b=prm[k,1]*4-2; pars$value[lin.b]=prm[k,3];    pars$est[lin.b]=FALSE
    lin.c=prm[k,1]*4-1; pars$value[lin.c]=prm[k,4];    pars$est[lin.c]=FALSE
    k=k+1; #cat("\nO item",k,"foi atualizado!\n")
  }
  else{ # (priori, media, desvio-padrao)
    pars[ip,10] = "lnorm" # Mesmo do BILOG-MG
    pars[ip,11:12] = c(1,exp(0.5)) # Mesmo do BILOG-MG

    pars[ip+1,10] = 'norm' # Mesmo do BILOG-MG
    pars[ip+1,11:12] = c(0,2) # Mesmo do BILOG-MG

    pars[ip+2,10] = 'norm' # No BILOG-MG e' Beta(5,17).
    pars[ip+2,11:12] = c(-1,.33) # No BILOG-MG e Beta(5,17) pela parametrização.

  }
 }

 pars.f = mirt(data,1, itemtype = PL, pars = pars)
 zeta=coef(pars.f, IRTpars = TRUE, simplify=TRUE)$items[,1:kPL] ## Parâmetros da Tabela
 colnames(zeta)=c("a","b","c","u")[1:kPL]

 cat("\nEstimativas dos parametros dos itens:\n"); print(round(zeta,3))

 return(zeta)
}
```

4. CONCLUSÕES E CONSIDERAÇÕES ESPECÍFICAS

A equalização no R precisa de cuidados adicionais quando comparada à equalização no BILOG-MG. Por outro lado, há efeitos tipo SHRINKAGE que são menores no mirt. Há diferentes forma de determinar restrições, usando especificação por string ou alterando diretamente na matriz associada.

O número de pontos de quadratura devem aumentar com o número de grupo, pois provavelmente teremos um intervalo maior a cobrir. O mesmo ocorre com o número de passos EM. Se esse cuidado não for tomado, o processo de equalização diminuirá a efetividade.

O tempo computacional é maior no mirt do que em aplicativos comerciais, aumentando bastante com o número de grupos de equalização.

BIBLIOGRAFIA

[1] Andrade, D.F., Tavares, H.R., Valle, R.C. (2000). Teoria da Resposta ao Item: Conceitos e Aplicações. Associação Brasileira de Estatística: São Paulo.

[2] Bennett, R.E, Ward, W. C, Rock, D. A & LaHart, C. (1990). Toward a framework for constructed-response items. New Jersay: Education Testing Service Princenton.

[3] Baker, F. B., Kim, Seock-Ho (2004). Item Response Theory - Parameter Estimation Techniques. New York: Marcel Dekker, Inc.

[4] Chalmers, R., P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. Journal of Statistical Software, 48(6), 1-29

[5] Embretson, S. E., & Reise, S. P. (2000). Item response theory for psychologists. Mahwah, NJ: Lawrence Erlbaum Associates.

[6] Hambleton, R. K. (1989). Principles and selected applications of item response theory. In R.L. Linn (Ed.), Educational Measurement. (3rd Ed.) (pp.147–200). New York: Macmillan Publishing Company.

[7] Hambleton, R. K, Swaminathan, H., & Rogers, H. J. (1991). Fundamentals of Item Response Theory. Newbury Park: Sage Publication.

[8] Klein, R. (2003). Utilização da Teoria da Resposta ao Item no Sistema Nacional de Avaliação da Educação Básica (SAEB). Revista Ensaio, n 40, v. 11, 283 – 296.

[9] Kline, P. (2000). (2nd Ed). The handbook of psychological testing.

[10] Longford, N. T.; Holland, P. W. & Thayer, D. T. Stability of the MH D-DIF Statistics Across Populations. In: Differential Item Functioning, Holland, P. W. & Wainer, H. (Eds.). Hillsdale, NJ: Lawrence Erlbaum, 1993.

[11] Lord, F.M., & Novick, M. R. (1968). Statistical theories of mental test scores. Reading, MA: Addition–Wesley Publishing Company.

[12] Lord, F.M. (1980). Applications of item response theory to practical testing problems. Hillsdale, NJ: Lawrence Erlbaum Associates.

[13] Van der Linden, W. J., & Hambleton, R. K. (Eds.). (1997). Handbook of modern item response theory. New York: Springer.

APÊNDICE A: MULTIPLEGROUP

Description

multipleGroup performs a full-information maximum-likelihood multiple group analysis for any combination of dichotomous and polytomous data under the item response theory paradigm using either Cai's (2010) Metropolis-Hastings Robbins-Monro (MHRM) algorithm or with an EM algorithm approach. This function may be used for detecting differential item functioning (DIF), thought the DIF function may provide a more convenient approach. If the grouping variable is not specified then the dentype input can be modified to fit mixture models to estimate any latent group components.

Usage

```
multipleGroup(
data,
model = 1,
group,
itemtype = NULL,
invariance = "",
method = "EM",
dentype = "Gaussian", ...
)
```

Arguments

data	a matrix or data.frame that consists of numerically ordered data, with missing data coded as NA
model	string to be passed to, or a model object returned from, mirt.model declaring how the global model is to be estimated (useful to apply constraints here)

group a character or factor vector indicating group membership. If a character vector is supplied this will be automatically transformed into a factor variable. As well, the first level of the (factorized) grouping variable will be treated as the "reference" group

itemtype can be same type of input as is documented in mirt, however may also be a ngroups by nitems matrix specifying the type of IRT models for each group, respectively. Rows of this input correspond to the levels of the group input. For mixture models the rows correspond to the respective mixture grouping variables to be constructed, and the IRT models should be within these mixtures

invariance a character vector containing the following possible options:

'free_mean' or 'free_means' freely estimate all latent means in all focal groups (reference group constrained to a vector of 0's)

'free_var', 'free_vars', 'free_variance', or 'free_variances' freely estimate all latent variances in focal groups (reference group variances all constrained to 1)

'slopes' to constrain all the slopes to be equal across all groups

'intercepts' to constrain all the intercepts to be equal across all groups, note for nominal models this also includes the category specific slope parameters

Additionally, specifying specific item name bundles (from colnames(data)) will constrain all freely estimated parameters in each item to be equal across groups. This is useful for selecting 'anchor' items for vertical and horizontal scaling, and for detecting differential item functioning (DIF) across groups

method a character object that is either 'EM', 'QMCEM', or 'MHRM' (default is 'EM'). See mirt for details

dentype type of density form to use for the latent trait parameters. Current options include all of the methods described in mirt, as well as

- 'mixture-#' estimates mixtures of Gaussian distributions, where the # placeholder represents the number of potential grouping variables (e.g., 'mixture-3' will estimate 3 underlying classes). Each class is assigned the group name MIXTURE_#, where # is the class number. Note that internally the mixture coefficients are stored as log values where the first mixture group coefficient is fixed at 0

... additional arguments to be passed to the estimation engine. See mirt for details and examples

Details

By default the estimation in multipleGroup assumes that the models are maximally independent, and therefore could initially be performed by sub-setting the data and running identical models with mirt and aggregating the results (e.g., log-likelihood). However, constrains may be automatically imposed across groups by invoking various invariance keywords. Users may also supply a list of parameter equality constraints to by constrain argument, of define equality constraints using the mirt.model syntax (recommended).

Value function returns an object of class MultipleGroupClass (MultipleGroupClass-class).

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Chalmers, R., P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software, 48*(6), 1-29. doi:10.18637/jss.v048.i06

See Also mirt, DIF, extract.group,

DRF

Examples

```
## Not run:

# single factor set.seed(12345) a <-
matrix(abs(rnorm(15,1,.3)), ncol=1) d <-
matrix(rnorm(15,0,.7),ncol=1) itemtype <-
rep('2PL', nrow(a)) N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2) group <-
c(rep('D1', N), rep('D2', N))

# marginal information itemstats(dat)

# conditional information itemstats(dat,
group=group)

mod_configural <- multipleGroup(dat, 1, group = group) #completely separate analyses
# limited information fit statistics
M2(mod_configural)

mod_metric <- multipleGroup(dat, 1, group = group, invariance=c('slopes')) #equal slopes
# equal intercepts, free variance and means mod_scalar2 <-
multipleGroup(dat, 1, group = group,
                                                invariance=c('slopes', 'intercepts', 'free_var','free_means'))
mod_scalar1 <- multipleGroup(dat, 1, group = group, #fixed means invariance=c('slopes', 'intercepts',
                                    'free_var'))
mod_fullconstrain <- multipleGroup(dat, 1, group = group, invariance=c('slopes', 'intercepts'))
extract.mirt(mod_fullconstrain, 'time') #time of estimation components

# optionally use Newton-Raphson for (generally) faster convergence in the M-step's mod_fullconstrain <-
multipleGroup(dat, 1, group = group, optimizer = 'NR', invariance=c('slopes', 'intercepts'))
extract.mirt(mod_fullconstrain, 'time') #time of estimation components
summary(mod_scalar2) coef(mod_scalar2,
simplify=TRUE) residuals(mod_scalar2)
plot(mod_configural) plot(mod_configural,
```

```
type = 'info') plot(mod_configural, type =
'trace')
plot(mod_configural, type = 'trace', which.items = 1:4)
itemplot(mod_configural, 2) itemplot(mod_configural, 2, type
= 'RE')

anova(mod_metric, mod_configural) #equal slopes only
anova(mod_scalar2, mod_metric) #equal intercepts, free variance and mean
anova(mod_scalar1, mod_scalar2) #fix mean anova(mod_fullconstrain, mod_scalar1)
#fix variance

# compared all at once (in order of most constrained to least) anova(mod_fullconstrain, mod_scalar2,
mod_configural)


# test whether first 6 slopes should be equal across groups values <- multipleGroup(dat, 1,
group = group, pars = 'values') values constrain <- list(c(1, 63), c(5,67), c(9,71), c(13,75),
c(17,79), c(21,83)) equalslopes <- multipleGroup(dat, 1, group = group, constrain =
constrain) anova(equalslopes, mod_configural)

# same as above, but using mirt.model syntax
newmodel <- ' F
      = 1-15
      CONSTRAINB = (1-6, a1)' equalslopes <- multipleGroup(dat,
newmodel, group = group) coef(equalslopes, simplify=TRUE)

#############
# vertical scaling (i.e., equating when groups answer items others do not) dat2 <- dat
dat2[group == 'D1', 1:2] <- dat2[group != 'D1', 14:15] <- NA head(dat2)
tail(dat2)

# items with missing responses need to be constrained across groups for identification nms <-
colnames(dat2)
mod <- multipleGroup(dat2, 1, group, invariance = nms[c(1:2, 14:15)])

# this will throw an error without proper constraints (SEs cannot be computed either) # mod <-
multipleGroup(dat2, 1, group)

# model still does not have anchors, therefore need to add a few (here use items 3-5) mod_anchor <-
multipleGroup(dat2, 1, group, invariance = c(nms[c(1:5, 14:15)], 'free_means', 'free_var'))
coef(mod_anchor, simplify=TRUE)

# check if identified by computing information matrix mod_anchor<-
multipleGroup(dat2,1,group,pars=mod2values(mod_anchor),TOL=NaN,SE=TRUE, invariance = c(nms[c(1:5,
14:15)], 'free_means', 'free_var'))
mod_anchor coef(mod_anchor)
coef(mod_anchor, printSE=TRUE)


##############
```

```
# DIF test for each item (using all other items as anchors) itemnames <-
colnames(dat)
refmodel <- multipleGroup(dat, 1, group = group, SE=TRUE, invariance=c('free_means', 'free_var',
                                        itemnames))

#loopoveritems(inpractice,runinparalleltoincreasespeed).Maybebettertouse?DIF estmodels <- vector('list',
ncol(dat)) for(i in 1:ncol(dat)) estmodels[[i]] <- multipleGroup(dat, 1, group = group, verbose = FALSE,
invariance=c('free_means', 'free_var', itemnames[-i]))
anova(refmodel, estmodels[[1]])
(anovas <- lapply(estmodels, function(x, refmodel) anova(refmodel, x), refmodel=refmodel))

# family-wise error control
p <- do.call(rbind, lapply(anovas, function(x) x[2, 'p'])) p.adjust(p, method = 'BH')

# same as above, except only test if slopes vary (1 df)
# constrain all intercepts estmodels <- vector('list', ncol(dat)) for(i in 1:ncol(dat)) estmodels[[i]] <-
multipleGroup(dat, 1, group = group, verbose = FALSE, invariance=c('free_means', 'free_var',
'intercepts', itemnames[-i]))

(anovas <- lapply(estmodels, function(x, refmodel) anova(refmodel, x), refmodel=refmodel))

# quickly test with Wald test using DIF()
mod_configural2 <- multipleGroup(dat, 1, group = group, SE=TRUE) DIF(mod_configural2, which.par =
c('a1', 'd'), Wald=TRUE, p.adjust = 'fdr')



##############
# Three group model where the latent variable parameters are constrained to
# be equal in the focal groups

set.seed(12345)
a <- matrix(abs(rnorm(15,1,.3)), ncol=1) d <-
matrix(rnorm(15,0,.7),ncol=1) itemtype <-
rep('2PL', nrow(a)) N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = .1, sigma = matrix(1.5)) dataset3 <- simdata(a, d,
N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2, dataset3) group <-
rep(c('D1', 'D2', 'D3'), each=N)

# marginal information itemstats(dat)

# conditional information itemstats(dat,
group=group)

model <- 'F1 = 1-15
          FREE[D2, D3] = (GROUP, MEAN_1), (GROUP, COV_11)
          CONSTRAINB[D2,D3] = (GROUP, MEAN_1), (GROUP, COV_11)'
```

```
mod <- multipleGroup(dat, model, group = group, invariance = colnames(dat)) coef(mod,
simplify=TRUE)



############# #
multiple factors

a <- matrix(c(abs(rnorm(5,1,.3)), rep(0,15),abs(rnorm(5,1,.3)), rep(0,15),abs(rnorm(5,1,.3))), 15, 3)
d <- matrix(rnorm(15,0,.7),ncol=1) mu <-
c(-.4, -.7, .1)
sigma <- matrix(c(1.21,.297,1.232,.297,.81,.252,1.232,.252,1.96),3,3)
itemtype <- rep('2PL', nrow(a)) N <-
1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = mu, sigma = sigma)
dat <- rbind(dataset1, dataset2) group <-
c(rep('D1', N), rep('D2', N))

# group models model
<- '
    F1 = 1-5
    F2 = 6-10
    F3 = 11-15'

# define mirt cluster to use parallel architecture if(interactive()) mirtCluster()

#EMapproach(notasaccuratewith3factors,butgenerallygoodforquickmodelcomparisons) mod_configural <-
multipleGroup(dat, model, group = group) #completely separate analyses mod_metric<-
multipleGroup(dat,model,group=group,invariance=c('slopes'))#equalslopes mod_fullconstrain<-
multipleGroup(dat,model,group=group,#equalmeans,slopes,intercepts invariance=c('slopes', 'intercepts'))

anova(mod_metric, mod_configural) anova(mod_fullconstrain,
mod_metric) # same as above, but with MHRM (generally more
accurate with 3+ factors, but slower) mod_configural <-
multipleGroup(dat, model, group = group, method = 'MHRM')
mod_metric<-
multipleGroup(dat,model,group=group,invariance=c('slopes'),method
='MHRM') mod_fullconstrain <- multipleGroup(dat, model, group =
group, method = 'MHRM', invariance=c('slopes', 'intercepts'))

anova(mod_metric, mod_configural) anova(mod_fullconstrain,
mod_metric)

#############
# polytomous item example set.seed(12345) a <-
matrix(abs(rnorm(15,1,.3)), ncol=1) d <-
matrix(rnorm(15,0,.7),ncol=1) d <- cbind(d, d-1,
d-2) itemtype <- rep('graded', nrow(a)) N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = .1, sigma = matrix(1.5))
```

```
dat <- rbind(dataset1, dataset2) group <-
c(rep('D1', N), rep('D2', N)) model <- 'F1 = 1-
15'

mod_configural <- multipleGroup(dat, model, group = group)
plot(mod_configural) plot(mod_configural, type = 'SE')
itemplot(mod_configural, 1) itemplot(mod_configural, 1, type = 'info')
plot(mod_configural, type = 'trace') # messy, score function typically better plot(mod_configural, type =
'itemscore')

fs <- fscores(mod_configural, full.scores = FALSE)
head(fs[["D1"]])
fscores(mod_configural, method = 'EAPsum', full.scores = FALSE)

# constrain slopes within each group to be equal (but not across groups) model2 <- 'F1 = 1-15
                CONSTRAIN = (1-15, a1)' mod_configural2 <-
multipleGroup(dat, model2, group = group) plot(mod_configural2, type =
'SE') plot(mod_configural2, type = 'RE') itemplot(mod_configural2, 10)

#############
## empirical histogram example (normal and bimodal groups)
set.seed(1234) a <- matrix(rlnorm(50, .2, .2)) d <- matrix(rnorm(50))
ThetaNormal <- matrix(rnorm(2000))
ThetaBimodal <- scale(matrix(c(rnorm(1000, -2), rnorm(1000,2)))) #bimodal
Theta <- rbind(ThetaNormal, ThetaBimodal) dat <- simdata(a, d, 4000,
itemtype = '2PL', Theta=Theta) group <- rep(c('G1', 'G2'), each=2000)
EH<-
multipleGroup(dat,1,group=group,dentype="empiricalhist",invarianc
e=colnames(dat)) coef(EH, simplify=TRUE)
plot(EH, type = 'empiricalhist', npts = 60)

# DIF test for item 1
EH1<-multipleGroup(dat,1,group=group,dentype="empiricalhist",invariance=colnames(dat)[-1])   anova(EH,   EH1)

#--------------------
# Mixture model (no prior group variable specified)

set.seed(12345) nitems <- 20 a1 <- matrix(.75,
ncol=1, nrow=nitems) a2 <- matrix(1.25,
ncol=1, nrow=nitems) d1 <-
matrix(rnorm(nitems,0,1),ncol=1) d2 <-
matrix(rnorm(nitems,0,1),ncol=1) itemtype <-
rep('2PL', nrow(a1))
N1 <- 500
N2 <- N1*2 # second class twice as large

dataset1 <- simdata(a1, d1, N1, itemtype) dataset2
<- simdata(a2, d2, N2, itemtype) dat <-
rbind(dataset1, dataset2) # group <- c(rep('D1',
N1), rep('D2', N2))
```

```
# Mixture Rasch model (Rost, 1990) models <-
'F1 = 1-20
                CONSTRAIN = (1-20, a1)' mod_mix <- multipleGroup(dat, models, dentype = 'mixture-
2', GenRandomPars = TRUE) coef(mod_mix, simplify=TRUE) summary(mod_mix) plot(mod_mix)
plot(mod_mix, type = 'trace') itemplot(mod_mix, 1, type = 'info')

head(fscores(mod_mix)) # theta estimates head(fscores(mod_mix, method = 'classify'))
# classification probability itemfit(mod_mix)

# Mixture 2PL model
mod_mix2 <- multipleGroup(dat, 1, dentype = 'mixture-2', GenRandomPars = TRUE)
anova(mod_mix, mod_mix2) coef(mod_mix2, simplify=TRUE) itemfit(mod_mix2)

# Compare to single group mod
<- mirt(dat) anova(mod,
mod_mix2)

########################################
# Zero-inflated 2PL IRT model
```

```
n <- 1000
nitems <-
20

a <- rep(2, nitems) d <- rep(c(-2,-1,0,1,2),
each=nitems/5) zi_p <- 0.2 # Proportion of people
in zero class

theta <- rnorm(n, 0, 1) zeros <-
matrix(0, n*zi_p, nitems)
nonzeros <- simdata(a, d, n*(1-zi_p), itemtype = '2PL',
                    Theta = as.matrix(theta[1:(n*(1-zi_p))]))
data <- rbind(nonzeros, zeros)

# define class with extreme theta but fixed item parameters
zi2PL <- "F = 1-20
          START [MIXTURE_1] = (GROUP, MEAN_1, -100), (GROUP, COV_11, .00001),
                              (1-20, a1, 1.0), (1-20, d, 0)
          FIXED [MIXTURE_1] = (GROUP, MEAN_1), (GROUP, COV_11),
                              (1-20, a1), (1-20, d)"

# define custom Theta integration grid that contains extreme theta + normal grid technical <-
list(customTheta = matrix(c(-100, seq(-6,6,length.out=61))))

# fit ZIM-IRT
zi2PL.fit <- multipleGroup(data, zi2PL, dentype = 'mixture-2', technical=technical) coef(zi2PL.fit,
simplify=TRUE)


## End(Not run)
```

MultipleGroupClass-class

Class "MultipleGroupClass"

Description

Defines the object returned from multipleGroup.

Slots

Call: function call

Data: list of data, sometimes in different forms

Options: list of estimation options

Fit: a list of fit information

Model: a list of model-based information

ParObjects: a list of the S4 objects used during estimation

OptimInfo: a list of arguments from the optimization process

numerical_deriv

Internals: a list of internal arguments for secondary computations (inspecting this object is generally not required) vcov: a matrix represented the asymptotic covariance matrix of the parameter estimates time: a data.frame indicating the breakdown of computation times in seconds

Methods coef

signature(object="MultipleGroupClass") print

signature(x="MultipleGroupClass") show

signature(object="MultipleGroupClass") anova

signature(object="MultipleGroupClass")

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Chalmers, R., P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software, 48*(6), 1-29. doi:10.18637/jss.v048.i06

mirt: Especificação do modelo

(disponível em https://rdrr.io/cran/mirt/man/mirt.model.html)

Specify model information

Description

The `mirt.model` function scans/reads user input to specify the confirmatory model. Item locations must be used in the specifications if no `itemnames` argument is supplied. This is called implicitly by estimation functions when a string is passed to the `model` argument.

Usage

```
mirt.model(
  input = NULL,
  itemnames = NULL,
  file = "",
  COV = NULL,
  quiet = TRUE,
  ...
)
```

Arguments

`input`	input for writing out the model syntax. Can either be a string declaration of class character or the so-called Q-matrix or class `matrix` that specifies the model either with integer or logical values. If the Q-matrix method is chosen covariances terms can be specified with the `COV` input
`itemnames`	a character vector or factor indicating the item names. If a data.frame or matrix object is supplied the names will be extracted using `colnames(itemnames)`. Supplying this input allows the syntax to be specified with the raw item names rather than item locations
`file`	a input specifying an external file that declares the input.
`COV`	a symmetric, logical matrix used to declare which covariance terms are estimated
`quiet`	logical argument passed to `scan()` to suppress console read message
`...`	additional arguments for `scan()`

Details

Factors are first named and then specify which numerical items they affect (i.e., where the slope is not equal to 0), separated either by commas or by - to indicate a range of items. Products between factors may be specified by enclosing the left hand term within brackets. To finish the declaration of a model simply enter a blank line with only a carriage return (i.e., the 'enter' or 'return' key), or instead read in an input version of the model syntax. The associated slopes throughout the package label these coefficients

as `a1, a2, ..., ak`, where the associated number is assigned according to the respective order of the defined factors.

For example, if the syntax were

```
"G = 1-10 F = 1-5 A = 6-10"
```

then the `G` factor would be assigned the slopes `a1` for each item, `F` assigned the slopes `a2`, and `A` assigned the slopes `a3`. The same principle applies to the `bfactor` function whereby the slopes are automatically included for the specific factors after the general factor structure has been assigned.

There is an optional keyword for specifying the correlation between relationships between factors called `COV`, and non-linear factor products can be included by enclosing the product combination on the left hand side of the declaration (e.g., `(F1*F1)` would create a quadratic factor for `F1`).

The keywords `CONSTRAIN, CONSTRAINB, PRIOR, FIXED, FREE, START, UBOUND, LBOUND` can be applied to specific sub-groups in multiple-group models by included square brackets before the = sign, where groups are separated by commas. For example, to apply within-group equality constraints to a group called "male", then specifying:

```
CONSTRAIN [male] = (1-5, a1)
```

is appropriate, while specifying the same constraints to the sub-groups "male" and "female" would appear as

```
CONSTRAIN [male, female] = (1-5, a1)
```

For all other groups in the multi-group model, these within-group equality constraints would not appear. Therefore, these bracketed group specifications are useful when modifying priors, starting values, between/within group equality constraints, and so on when the specifications for each sub-group may differ.

Finally, the keyword `GROUP` can be used to specify the group-level hyper-parameter terms, such as the means and variance of the default Gaussian distribution. For example, to set the starting value of the variance parameter (`COV_11`) to 1.5:

```
START = (GROUP, COV_11, 1.5)
```

COV

> Specify the relationship between the latent factors. Estimating a correlation between factors is declared by joining the two factors with an asterisk (e.g.,

F1*F2), or with an asterisk between three or more factors to estimate all the possible correlations (e.g., F1*F2*F3)

MEAN

A comma separated list specifying which latent factor means to freely estimate. E.g., `MEAN = F1, F2` will free the latent means for factors F1 and F2

CONSTRAIN

A bracketed, comma separated list specifying equality constrains between items. The input format is `CONSTRAIN = (items, ..., parameterName(s)), (items, ..., parameterName)`.

For example, in a single group 10-item dichotomous tests, using the default 2PL model, the first and last 5 item slopes (a1) can be constrained to be equal by using `CONSTRAIN = (1-5, a1), (6-10, a1)`, or some combination such as `CONSTRAIN = (1-3,4,5,a1), (6,7,8-10,a1)`.

When constraining parameters to be equal across items with different parameter names, a balanced bracketed vector must be supplied. E.g., setting the first slope for item 1 equal to the second slope in item 3 would be `CONSTRAIN = (1, 3, a1, a2)`

CONSTRAINB

A bracketed, comma separate list specifying equality constrains between groups. The input format is `CONSTRAINB = (items, ..., parameterName), (items, ..., parameterName)`.

For example, in a two group 10-item dichotomous tests, using the default 2PL model, the first 5 item slopes (a1) can be constrained to be equal across both groups by using `CONSTRAINB = (1-5, a1)`, or some combination such as `CONSTRAINB = (1-3,4,5,a1)`

PRIOR

A bracketed, comma separate list specifying prior parameter distributions. The input format is `PRIOR = (items, ..., parameterName, priorType, val1, val2), (items, ..., parameterName, priorType, val1, val2)`. For example, in a single group 10-item dichotomous tests, using the default 2PL model, defining a normal prior of N(0,2) for the first 5 item intercepts (d) can be defined by `PRIOR = (1-5, d, norm, 0, 2)`

Currently supported priors are of the form: `(items, norm, mean, sd)` for the normal/Gaussian, `(items, lnorm, log_mean, log_sd)` for log-normal, `(items, beta, alpha, beta)` for beta, and `(items, expbeta, alpha, beta)` for the beta distribution after applying the function `plogis` to the input value (note, this is

specifically for applying a beta prior to the lower-bound parameters in 3/4PL models)

LBOUND

A bracketed, comma separate list specifying lower bounds for estimated parameters (used in optimizers such as `L-BFGS-B` and `nlminb`). The input format is `LBOUND = (items, ..., parameterName, value), (items, ..., parameterName, value)`.

For example, in a single group 10-item dichotomous tests, using the 3PL model and setting lower bounds for the 'g' parameters for the first 5 items to 0.2 is accomplished with `LBOUND = (1-5, g, 0.2)`

UBOUND

same as LBOUND, but specifying upper bounds in estimated parameters

START

A bracketed, comma separate list specifying the starting values for individual parameters. The input is of the form `(items, ..., parameterName, value)`. For instance, setting the 10th and 12th to 15th item slope parameters (a1) to 1.0 is specified with `START = (10, 12-15, a1, 1.0)`

For more hands on control of the starting values pass the argument `pars = 'values'` through whatever estimation function is being used

FIXED

A bracketed, comma separate list specifying which parameters should be fixed at their starting values (i.e., not freely estimated). The input is of the form `(items, ..., parameterName)`. For instance, fixing the 10th and 12th to 15th item slope parameters (a1) is accomplished with `FIXED = (10, 12-15, a1)`

For more hands on control of the estimated values pass the argument `pars = 'values'` through whatever estimation function is being used

FREE

Equivalent to the `FIXED` input, except that parameters are freely estimated instead of fixed at their starting value

NEXPLORE

Number of exploratory factors to extract. Usually this is not required because passing a numeric value to the `model` argument in the estimation function will generate an exploratory factor analysis model, however if different start values, priors, lower and upper bounds, etc, are desired then this input can be used

Value

Returns a model specification object to be used in `mirt`, `bfactor`, `multipleGroup`, or `mixedmirt`

Author(s)

Phil Chalmers rphilip.chalmers@gmail.com and Alexander Robitzsch

References

Chalmers, R., P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software*, *48*(6), 1-29. \Sexpr[results=rd]{tools:::Rd_expr_doi("10.18637/jss.v048.i06")}

Examples

```
## Not run:

# interactively through the console (not run)
#model <- mirt.model()
#  F1 = 1,2,3,4-10
#  F2 = 10-20
# (F1*F2) = 1,2,3,4-10
# COV = F1*F2


# Or alternatively with a string input
s <- 'F1 = 1,2,3,4-10
      F2 = 10-20
      (F1*F2) = 1,2,3,4-10
      COV = F1*F2'
model <- mirt.model(s)

# strings can also be passed to the estimation functions directly,
#   which silently calls mirt.model(). E.g., using the string above:
# mod <- mirt(data, s)


# Q-matrix specification
Q <- matrix(c(1,1,1,0,0,0,0,0,0,1,1,1), ncol=2, dimnames = list(NULL, c('Factor1',
'Factor2')))
COV <- matrix(c(FALSE, TRUE, TRUE, FALSE), 2)
model <- mirt.model(Q, COV=COV)

## constrain various items slopes and all intercepts in single group model to be equal,
#   and use a log-normal prior for all the slopes
s <- 'F = 1-10
      CONSTRAIN = (1-3, 5, 6, a1), (1-10, d)
      PRIOR = (1-10, a1, lnorm, .2, .2)'
model <- mirt.model(s)


## constrain various items slopes and intercepts across groups for use in
multipleGroup(),
#  and constrain first two slopes within 'group1' to be equal
s <- 'F = 1-10
      CONSTRAIN = (1-2, a1)
      CONSTRAINB = (1-3, 5, 6, a1), (1-10, d)'
model <- mirt.model(s)


## specify model using raw item names
```

```
data(data.read, package = 'sirt')
dat <- data.read

# syntax with variable names
mirtsyn2 <- "
       F1 = A1,B2,B3,C4
       F2 = A1-A4,C2,C4
       MEAN = F1
       COV = F1*F1, F1*F2
       CONSTRAIN=(A2-A4,a2),(A3,C2,d)
       PRIOR = (C3,A2-A4,a2,lnorm, .2, .2),(B3,d,norm,0,.0001)"
# create a mirt model
mirtmodel <- mirt.model(mirtsyn2, itemnames=dat)
# or equivalently:
# mirtmodel <- mirt.model(mirtsyn2, itemnames=colnames(dat))

# mod <- mirt(dat , mirtmodel)


## End(Not run)
```