# Flexible Rasch Mixture Models with Package psychomix

**Hannah Frick**
Universität Innsbruck

**Carolin Strobl**
Universität Zürich

**Friedrich Leisch**
Universität für Bodenkultur Wien

**Achim Zeileis**
Universität Innsbruck

**Abstract**

This introduction to the R package **psychomix** is a (slightly) modified version of Frick, Strobl, Leisch, and Zeileis (2012), published in the *Journal of Statistical Software*.

Measurement invariance is an important assumption in the Rasch model and mixture models constitute a flexible way of checking for a violation of this assumption by detecting unobserved heterogeneity in item response data. Here, a general class of Rasch mixture models is established and implemented in R, using conditional maximum likelihood estimation of the item parameters (given the raw scores) along with flexible specification of two model building blocks: (1) Mixture weights for the unobserved classes can be treated as model parameters or based on covariates in a concomitant variable model. (2) The distribution of raw score probabilities can be parametrized in two possible ways, either using a saturated model or a specification through mean and variance. The function `raschmix()` in the R package **psychomix** provides these models, leveraging the general infrastructure for fitting mixture models in the **flexmix** package. Usage of the function and its associated methods is illustrated on artificial data as well as empirical data from a study of verbally aggressive behavior.

*Keywords*: mixed Rasch model, Rost model, mixture model, **flexmix**, R.

## 1. Introduction

In item response theory (IRT), latent traits are usually measured by employing probabilistic models for responses to sets of items. One of the most prominent examples for such an approach is the Rasch model (Rasch 1960) which captures the difficulty (or equivalently easiness) of binary items and the respondent's trait level on a single common scale. Generally, a central assumption of most IRT models (including the Rasch model) is measurement invariance, i.e., that all items measure the latent trait in the same way for all subjects. If violated, measurements obtained from such a model provide no fair comparisons of the subjects. A typical violation of measurement invariance in the Rasch model is differential item functioning (DIF), see Ackerman (1992).

Therefore, assessing the assumption of measurement invariance and checking for DIF is crucial when establishing a Rasch model for measurements of latent traits. Hence, various approaches have been suggested in the literature that try to assess heterogeneity in (groups of) subjects either based on observed covariates or unobserved latent classes. If covariates are available, classical tests like the Wald or likelihood ratio test can be employed to compare model fits be-

tween some reference group and one or more focal groups (Glas and Verhelst 1995). Typically, these groups are defined by the researcher based on categorical covariates or arbitrary splits in either numerical covariates or the raw scores (Andersen 1972). More recently, extensions of these classical tests have also been embedded into a mixed model representation (Van den Noortgate and De Boeck 2005). Another recently suggested technique is to recursively define and assess groupings in a data-driven way based on all available covariates (both numerical and categorical) in so-called Rasch trees (Strobl, Kopf, and Zeileis 2011).

Heterogeneity occurring in latent classes only (i.e., not observed or captured by covariates), however, is typically addressed by mixtures of IRT models. Specifically, Rost (1990) combined a mixture model approach with the Rasch model. If any covariates are present, they can be used to predict the latent classes (as opposed to the item parameters themselves) in a second step (Cohen and Bolt 2005). More recently, extensions to this mixture model approach have been suggested that encompass this prediction, see Tay, Newman, and Vermunt (2011) for a unifying framework.

DIF cannot be separated from multidimensionality when items measure several different latent traits which also differ in (groups of) subjects (cf., e.g., Ackerman 1992, and the references therein). In this sense, mixtures of Rasch models can also be used to assess the assumption of unidimensionality, see Rijmen and De Boeck (2005) on the relationship of between-item multidimensional models and mixtures of Rasch models.

In this paper, we introduce the **psychomix** package for the R system for statistical computing (R Development Core Team 2011) that provides software for fitting a general and flexible class of Rasch mixture models along with comprehensive methods for model selection, assessment, and visualization. The package leverages the general and object-oriented infrastructure for fitting mixture models from the **flexmix** package (Leisch 2004; Grün and Leisch 2008), combining it with the function `RaschModel.fit()` from the **psychotools** package (Zeileis, Strobl, and Wickelmaier 2011) for the estimation of Rasch models. All packages are freely available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/`.

The reason for using `RaschModel.fit()` as opposed to other previously existing (and much more powerful and flexible) R packages for Rasch modeling – such as **ltm** (Rizopoulos 2006) or **eRm** (Mair and Hatzinger 2007) – is reduced computational complexity: `RaschModel.fit()` is intended to provide a "no frills" implementation of simple Rasch models, useful when refitting a model multiple times in mixtures or recursive partitions (see also Strobl *et al.* 2011).

While **psychomix** was under development, another R implementation of the Rost (1990) model became available in package **mRm** (Preinerstorfer and Formann 2011). As this builds on specialized C++ code, it runs considerably faster than the generic **flexmix** approach – however, it only covers this one particular type of model and offers fewer methods for specifying, inspecting, and assessing (fitted) models. In **psychomix**, both approaches are reconciled by optionally employing the **mRm** solution as an input to the **flexmix** routines.

In the following, we first briefly review both Rasch and mixture models and combine them in a general Rasch mixture framework (Section 2). Subsequently, the R implementation in **psychomix** is introduced (Section 3), illustrated by means of simulated data, and applied in practice to a study of verbally aggressive behavior (Section 4). Concluding remarks are provided in Section 5.

# 2. Rasch mixture models

In the following, we first provide a short introduction to the Rasch model, subsequently outline the basics of mixture models in general, and finally introduce a general class of Rasch mixture models along with the corresponding estimation techniques.

## 2.1. The Rasch model

Latent traits can be measured through a set of items to which binary responses are given. Success of solving an item or agreeing with it is coded as "1", while "0" codes the opposite response. The model suggested by Rasch (1960) uses the person's ability $\theta_i$ $(i = 1, \ldots, n)$ and the item's difficulty $\beta_j$ $(j = 1, \ldots, m)$ to model the response $y_{ij}$ of person $i$ to item $j$:

$$P(Y_{ij} = y_{ij}|\theta_i, \beta_j) \;=\; \frac{\exp\{y_{ij}(\theta_i - \beta_j)\}}{1 + \exp\{\theta_i - \beta_j\}}. \tag{1}$$

Under the assumption of independence – both across persons and items within persons (see Molenaar 1995b) – the likelihood for the whole sample $y = (y_{ij})_{n\times m}$ can be written as the product of the likelihood contributions from Equation 1 for all combinations of subjects and items. It is parametrized by the vector of all person parameters $\theta = (\theta_1, \ldots, \theta_n)^\top$ and the vector of all item parameters $\beta = (\beta_1, \ldots, \beta_m)^\top$ (see Equation 2). Since the probability of solving an item is modeled only through the logit of the difference between person ability and item difficulty, these parameters are on an interval scale with an arbitrary zero point. To obtain a fixed reference point, usually one item difficulty (e.g., the first $\beta_1$) or the sum of item difficulties is restricted to zero. See Fischer (1995) for details.

On the basis of the number of correctly solved items, the so-called "raw" scores $r_i = \sum_{j=1}^m y_{ij}$, the likelihood for the full sample can be factorized into a conditional likelihood of the item parameters $h(\cdot)$ and the score probabilities $g(\cdot)$ (Equation 3). Because the scores $r$ are sufficient statistics for the person parameters $\theta$, the likelihood of the item parameters $\beta$ conditional on the scores $r$ does not depend on the person parameters $\theta$ (Equation 4).

$$\begin{aligned} L(\theta, \beta) &= f(y|\theta, \beta) && (2)\\ &= h(y|r, \theta, \beta)g(r|\theta, \beta) && (3)\\ &= h(y|r, \beta)g(r|\theta, \beta). && (4) \end{aligned}$$

The conditional likelihood of the item parameters takes the form

$$h(y|r, \beta) \;=\; \prod_{i=1}^{n} \frac{\exp\{-\sum_{j=1}^m y_{ij}\beta_j\}}{\gamma_{r_i}(\beta)} \tag{5}$$

where $\gamma_{r_i}(\cdot)$ is the elementary symmetric function of order $r_i$, capturing all possible response patterns leading to a certain score (see Molenaar 1995a, for details).

There are several approaches to estimating the Rasch model: *Joint maximum likelihood (ML)* estimation of $\beta$ and $\theta$ is inconsistent, thus two other approaches have been established. Both are two-step approaches but differ in the way the person parameters $\theta$ are handled. For *marginal ML* estimation a distribution for $\theta$ is assumed and integrated out in $L(\theta, \beta)$, or equivalently in $g(r|\theta, \beta)$. In the *conditional ML* approach only the conditional likelihood of the

item parameters $h(y|r, \beta)$ from Equation 5 is maximized for estimating the item parameters. Technically, this is equivalent to maximizing $L(\theta, \beta)$ with respect to $\beta$ if one assumes that $g(r|\delta) = g(r|\theta, \beta)$ does not depend on $\theta$ or $\beta$, but potentially other parameters $\delta$.

In R, the **ltm** package (Rizopoulos 2006) uses the marginal ML approach while the **eRm** package (Mair and Hatzinger 2007) employs the conditional ML approach, i.e., uses and reports only the conditional part of the likelihood in the estimation of $\beta$. The latter approach is also taken by the `RaschModel.fit()` function in the **psychotools** package (Zeileis *et al.* 2011).

## 2.2. Mixture models

Mixture models are a generic approach for modeling data that is assumed to stem from different groups (or clusters) but group membership is unknown.

The likelihood $f(\cdot)$ of such a mixture model is a weighted sum (with prior weights $\pi_k$) of the likelihood from several components $f_k(\cdot)$ representing the different groups:

$$f(y_i) \;=\; \sum_{k=1}^{K} \pi_k f_k(y_i).$$

Generally, the components $f_k(\cdot)$ can be densities or (regression) models. Typically, all $K$ components $f_k(\cdot)$ are assumed to be of the same type $f(y|\xi_k)$, distinguished through their component-specific parameter vector $\xi_k$.

If variables are present which do not influence the components $f_k(\cdot)$ themselves but rather the prior class membership probabilities $\pi_k$, they can be incorporated in the model as so-called *concomitant variables* (Dayton and Macready 1988). In the psychometric literature, such covariates predicting latent information are also employed, e.g., by Tay *et al.* (2011) who advocate a unifying IRT framework that also optionally encompasses concomitant information (labeled *MM-IRT-C* for mixed-measurement IRT with covariates). To embed such concomitant variables $x_i$ into the general mixture model notation, a model for the component membership probability $\pi(k|x_i, \alpha)$ with parameters $\alpha$ is employed:

$$f(y_i|x_i, \alpha, \xi_1, \ldots, \xi_K) \;=\; \sum_{k=1}^{K} \pi(k|x_i, \alpha) f(y_i|\xi_k), \tag{6}$$

where commonly a multinomial logit model is chosen to parametrize $\pi(k|x_i, \alpha)$ (see e.g., Grün and Leisch 2008; Tay *et al.* 2011). Note that the multinomial model collapses to separate $\pi_k$ $(k = 1, \ldots, K)$ if there is only an intercept and no real concomitants in $x_i$.

## 2.3. Flavors of Rasch mixture models

When combining the general mixture model framework from Equation 6 with the Rasch model based on Equation 1, several options are conceivable for two of the building blocks. First, the component weights can be estimated via a separate parameter $\pi_k$ for each component or via a concomitant variable model $\pi(k|x_i, \alpha)$ with parameters $\alpha$. Second, the full likelihood function $f(y_i|\xi_k)$ of the components needs to be defined. If a conditional ML approach is adopted, it is clear that the conditional likelihood $h(y_i|r_i, \beta)$ from Equation 5 should be one part, but various choices for modeling the score probabilities are available. One option is

to model each score probability with its own parameter $g(r_i) = \psi_{r_i}$, while another (more parsimonious) option would be to adopt a parametric distribution of the score probabilities with fewer parameters (Rost and von Davier 1995).

Note that while for a single-component model, the estimates of the item parameters $\hat{\beta}$ are invariant to the choice of the score probabilities (as long as it is independent from $\beta$), this is no longer the case for a mixture model with $K \geq 2$. The estimation of the item parameters in the mixture is invariant given the weights $\pi(k|x_i, \alpha)$ but the weights and thus the estimates of $\beta$ may depend on the score distribution in a mixture model. (This dependency is introduced through the posterior probabilities calculated in the E-step of the algorithm that is explained Section 2.4.) Also note that the conditional ML approach employed here uses a model $g(r|\delta)$ directly on the score probabilities rather than a distribution on the person parameters $\theta$ as does the marginal ML approach.

*Rost's original parametrization*

One of these possible mixtures – the so-called "mixed Rasch model" introduced by Rost (1990) – is already well-established in the psychometric literature. It models the score probabilities through separate parameters $g(r_i) = \psi_{r_i}$ (under the restriction that they sum to unity) and does not employ concomitant variables. The likelihood of Rost's mixture model can thus be written as

$$f(y|\pi, \psi, \beta) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k h(y_i|r_i, \beta_k) \psi_{r_i,k}. \tag{7}$$

This particular parametrization is implemented in the R package **mRm** (Preinerstorfer and Formann 2011).

Since subjects who solve either none or all items (i.e., $r_i = 0$ or $m$, respectively) do not contribute to the conditional likelihood of the item parameters they cannot be allocated to any of the components in this parametrization. Hence, Rost (1990) proposed to remove those "extreme scorers" from the analysis entirely and fix the corresponding score probabilities $\psi_0$ and $\psi_m$ at 0. However, if one wishes to include these extreme scorers in the analysis, the corresponding score probabilities can be estimated through their relative frequency (across all components) and the remaining score probabilities within each component are rescaled to sum to unity together with those extreme score probabilities. Nevertheless, the extreme scorers still do not contribute to the estimation of the mixture itself.

*Other score distributions*

As noted by Rost and von Davier (1995), the disadvantage of this saturated model for the raw score probabilities is that many parameters need to be estimated ($K \times (m - 2)$, not counting potential extreme scorers) that are typically not of interest. To check for DIF, the item parameters are of prime importance while the raw score distribution can be regarded as a nuisance term. This problem can be alleviated by embedding the model from Equation 7 into a more general framework that also encompasses more parsimonious parametrizations. More specifically, a conditional logit model can be established

$$g(r|\delta) = \frac{\exp\{z_r^\top \delta\}}{\sum_{j=1}^{m-1} \exp\{z_j^\top \delta\}}, \tag{8}$$

containing some auxiliary regressors $z_i$ with coefficients $\delta$.

The saturated $g(r_i) = \psi_{r_i}$ model is a special case when constructing the auxiliary regressor from indicator/dummy variables for the raw scores $2, \ldots, m-1$: $z_i = (I_2(r_i), \ldots, I_{m-1}(r_i))^\top$. Then $\delta = (\log(\psi_2) - \log(\psi_1), \ldots, \log(\psi_{m-1}) - \log(\psi_1))^\top$ is a simple logit transformation of $\psi$.

As an alternative Rost and von Davier (1995) suggests a specification with only two parameters that link to mean and variance of the score distribution, respectively. More specifically, the auxiliary regressor is $z_i = (r_i/m, 4r_i(m - r_i)/m^2)^\top$ so that $\delta$ pertains to the vector of location and dispersion parameters of the score distribution. Thus, for $m > 4$ items, this parametrization is more parsimonious than the saturated model.

*General Rasch mixture model*

Combining all elements of the likelihood this yields a more general specification of the Rasch mixture model

$$f(y|\alpha, \beta, \delta) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi(k|x_i, \alpha)\ h(y_i|r_i, \beta_k)\ g(r_i|\delta_k) \tag{9}$$

with (a) the concomitant model $\pi(k|x_i, \alpha)$ for modeling component membership, (b) the component-specific conditional likelihood of the item parameters given the scores $h(y_i|r_i, \beta_k)$, and (c) the component-specific score distribution $g(r_i|\delta_k)$.

## 2.4. Parameter estimation

Parameter estimation for mixture models is usually done via the expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin 1977). It treats group membership as unknown and optimizes the complete-data log-likelihood including the group membership on basis of the observed values only. It iterates between two steps until convergence: estimation of group membership (E-step) and estimation of the components (M-step).

In the E-step, the posterior probabilities of each observation for the $K$ components is estimated through:

$$\hat{p}_{ik} \ = \ \frac{\hat{\pi}_k f(y_i|\hat{\xi}_k)}{\sum_{g=1}^{K} \hat{\pi}_g f(y_i|\hat{\xi}_g)} \tag{10}$$

using the parameter estimates from the previous iteration for the component weights $\pi$ and the model parameters $\xi$ which encompass $\beta$ and $\delta$. In the case of concomitant variables, the weights are $\hat{\pi}_{ik} = \pi(k|x_i, \hat{\alpha})$.

In the M-step, the parameters of the mixture are re-estimated with the posterior probabilities as weights. Thus, observations deemed unlikely to belong to a certain component have little influence on estimation within this component. For each component, the weighted ML estimation can be written as

$$\begin{aligned} \hat{\xi}_k \ &= \ \operatorname*{argmax}_{\xi_k} \sum_{i=1}^{n} \hat{p}_{ik} \log f(y_i|\xi_k) \quad (k = 1, \ldots, K) \\ &= \ \left\{ \operatorname*{argmax}_{\beta_k} \sum_{i=1}^{n} \hat{p}_{ik} \log h(y_i|r_i, \beta_k);\ \ \operatorname*{argmax}_{\delta_k} \sum_{i=1}^{n} \hat{p}_{ik} \log g(r_i|\delta_k) \right\} \end{aligned} \tag{11}$$

which for the Rasch model amounts to separately maximizing the weighted conditional log-likelihood for the item parameters and the weighted score log-likelihood.

The concomitant model can be estimated separately from the posterior probabilities:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{p}_{ik} \log(\pi(k|x_i, \alpha)), \tag{12}$$

where $\pi(k|x_i, \alpha)$ could be, e.g, a multinomial model.

Finally, note that the number of components $K$ is not a standard model parameter (because the likelihood regularity conditions do not apply) and thus it is not estimated through the EM algorithm. Either it needs to be chosen by the practitioner or by model selection techniques such as information criteria, as illustrated in the following examples.

# 3. Implementation in R

## 3.1. User interface

The function `raschmix()` can be used to fit the different flavors of Rasch mixture models described in Section 2.3: with or without concomitant variables in $\pi(k|x_i, \alpha)$, and with different score distributions $g(r_i|\delta_k)$ (saturated vs. mean/variance parametrization). The function's synopsis is

```
raschmix(formula, data, k, subset, weights, scores = "saturated",
  nrep = 3, cluster = NULL, control = NULL,
  verbose = TRUE, drop = TRUE, unique = FALSE, which = NULL,
  gradtol = 1e-6, deriv = "sum", hessian = FALSE, ...)
```

where the lines of arguments pertain to (1) data/model specification processed within `raschmix()`, (2) control arguments for fitting a single mixture model, (3) control arguments for iterating across mixtures over a range of numbers of components $K$, all passed to `stepFlexmix()`, and (4) control arguments for fitting each model component within a mixture (i.e., the M-step) passed to `RaschModel.fit()`. Details are provided below, focusing on usage in practice first.

A formula interface with the usual `formula`, `data`, `subset`, and `weights` arguments is used: The left-hand side of the formula sets up the response matrix $y$ and the right-hand side the concomitant variables $x$ (if any). The response may be provided by a single matrix or a set of individual dummy vectors, both of which may be contained in an optional data frame. Example usages are `raschmix(resp ~ 1, ...)` if the matrix `resp` is an object in the environment of the formula, typically the calling environment, or `raschmix(item1 + item2 + item3 ~ 1, data = d, ...)` if the `item*` vectors are in the data frame `d`. In both cases, `~ 1` signals that there are no concomitant variables – if there were, they could be specified as `raschmix(resp ~ conc1 + conc2, ...)`. As an additional convenience, the formula may be omitted entirely if there are no concomitant variables, i.e., `raschmix(data = resp, ...)` or alternatively `raschmix(resp, ...)`.

The `scores` of the model can be set to either `"saturated"` (see Equation 7) or `"meanvar"` for the mean/variance specification of Rost and von Davier (1995). Finally, the number of components $K$ of the mixture is specified through `k`, which may be a vector resulting in a mixture model being fitted for each element.

To control the EM algorithm for fitting the specified mixture models, `cluster` may optionally specify starting probabilities $\hat{p}_{ik}$ and `control` can set certain control arguments through a named list or an object of class "`FLXcontrol`". One of these control arguments named `minprior` sets the minimum prior probability for every component. If in an iteration of the EM algorithm, any component has a prior probability smaller then `minprior`, it is removed from the mixture in the next iteration. The default is `0`, i.e., avoiding such shrinkage of the model. If `cluster` is not provided, `nrep` different random initializations are employed, keeping only the best solution (to avoid local optima). Finally, `cluster` can be set to `"mrm"` in which case the fast C++ implementation from **mRm** (Preinerstorfer and Formann 2011) can be leveraged to generate optimized starting values. Again, the best solution of `nrep` runs of `mrm()` is used. Note that as of version 1.0 of **mRm** only the model from Equation 7 is supported in `mrm()`, resulting in suboptimal – but potentially still useful – posterior probabilities $\hat{p}_{ik}$ for any other model flavor.

Internally, `stepFlexmix()` is called to fit all individual mixture models and takes control arguments `verbose`, `drop`, and `unique`. If `k` is a vector, the whole set of models is returned by default but one may choose to select only the best model according to an information criterion. For example, `raschmix(resp, k = 1:3, which = "AIC", ...)` or `raschmix(resp ~ 1, data = d, k = 1:4, which = "BIC", ...)`.

The arguments `gradtol`, `deriv` and `hessian` are used to control the estimation of the item parameters in each M-step (Equation 11) carried out via `RaschModel.fit()`.

Function `raschmix()` returns objects of class "`raschmix`" or "`stepRaschmix`", respectively, depending on whether a single or multiple mixture models are fitted. These classes extend "`flexmix`" and "`stepFlexmix`", respectively, for more technical details see the next section. For standard methods for extracting or displaying information, either for "`raschmix`" directly or by inheritance, see Table 1 for an overview.

### 3.2. Internal structure

As briefly mentioned above, `raschmix()` leverages the **flexmix** package (Leisch 2004; Grün and Leisch 2008) and particularly its `stepFlexmix()` function for the estimation of (sets of) mixture models.

The **flexmix** package is designed specifically to provide the infrastructure for flexible mixture modeling via the EM algorithm, where the type of a mixture model is determined through the model employed in the components. In the estimation process, this component model definition corresponds to the definition of the M-step (Equation 11). Consequently, the **flexmix** package provides the framework for fitting mixture models by leveraging the modular structure of the EM algorithm. Provided with the right M-step, **flexmix** takes care of the data handling and iterating estimation through both E-step and M-step.

The M-step needs to be provided in the form of a `flexmix` driver inheriting from class "`FLXM`" (see Grün and Leisch 2008, for details). The **psychomix** package includes such a driver function: `FLXMCrasch()` relies on the function `RaschModel.fit()` from the **psychotools** package for estimation of the item parameters (i.e., maximization of the conditional likelihood from Equation 5) and adds different estimates of raw score probabilities depending on their parametrization. The driver can also be used directly with functions `flexmix()` and `stepFlexmix()`. The differences in model syntax and functionality for the classes of the resulting objects are illustrated in the Appendix A. As noted in the introduction, the reason for

| Function | Class | Description |
|---|---|---|
| `summary()` | "`raschmix`" | display information about the posterior probabilities and item parameters; returns an object of class "`summary.raschmix`" containing the relevant summary statistics (which has a `print()` method) |
| `parameters()` | "`raschmix`" | extract estimated parameters of the model for all or specified components, extract either parameters $\alpha$ of the concomitant model or item parameters $\beta$ and/or score parameters $\delta$ |
| `worth()` | "`raschmix`" | extract the item parameters $\beta$ under the restriction $\sum_{j=1}^{m} \beta_j = 0$ |
| `scoreProbs()` | "`raschmix`" | extract the score probabilities $g(r\|\delta)$ |
| `plot()` | "`raschmix`" | base graph of item parameter profiles in all or specified components |
| `xyplot()` | "`raschmix`" | lattice graph of item parameter profiles of all or specified components in a single or multiple panels |
| `histogram()` | "`raschmix`" | lattice rootogram or histogram of posterior probabilities |
| `print()` | "`stepFlexmix`" | simple printed display of number of components, log-likelihoods, and information criteria |
| `plot()` | "`stepFlexmix`" | plot information criteria against number of components |
| `getModel()` | "`stepFlexmix`" | select model according to either an information criterion or the number of components |
| `print()` | "`flexmix`" | simple printed display with cluster sizes and convergence |
| `clusters()` | "`flexmix`" | extract predicted class memberships |
| `posterior()` | "`flexmix`" | extract posterior class probabilities |
| `logLik()` | "`flexmix`" | extract fitted log-likelihood |
| `AIC()`; `BIC()` | "`flexmix`" | compute information criteria AIC, BIC |

Table 1: Methods for objects of classes "`raschmix`", "`flexmix`", and "`stepFlexmix`".

employing `RaschModel.fit()` rather than one of the more established Rasch model packages such as **eRm** or **ltm** is speed.

In the **flexmix** package, two fitting functions are provided. `flexmix()` is designed for fitting one model once and returns an object of class "`flexmix`". `stepFlexmix()` extends this so that either a single model or several models can be fitted. It also provides the functionality to fit each model repeatedly to avoid local optima.

When fitting models repeatedly, only the solution with the highest likelihood is returned. Thus, if `stepFlexmix()` is used to repeatedly fit a single model, it returns an object of class "`flexmix`". If `stepFlexmix()` is used to fit several different models (repeatedly or just once), it returns an object of class "`stepFlexmix`".

This principle extends to `raschmix()`: If it is used to fit a single model, the returned object is of class "`raschmix`". If used for fitting multiple models, `raschmix()` returns an object of

class "stepRaschmix". Both classes extend their **flexmix** counterparts.

### 3.3. Illustrations

For illustrating the flexible usage of `raschmix()`, we employ an artificial data set drawn from one of the three data generating processes (DGPs) suggested by Rost (1990) for the introduction of Rasch mixture models. All three DGPs are provided in the function `simRaschmix()` setting the `design` to `"rost1"`, `"rost2"`, or `"rost3"`, respectively. The DGPs contain mixtures of $K = 1$, and 2, and 3 components, respectively, all with $m = 10$ items.

The DGP `"rost1"` is intended to illustrate the model's capacity to correctly determine when no DIF is present. Thus, it includes only one latent class with item parameters $\beta^{(1)} = (2.7, 2.1, 1.5, 0.9, 0.3, -0.3, -0.9, -1.5, -2.1, -2.7)^\top$. (Rost originally used opposite signs to reflect item easiness parameters but since difficulty parameters are estimated by `raschmix()` the signs have been reversed.) The DGP `"rost2"` draws observations from two latent classes of equal sizes with item parameters of opposite signs: $\beta^{(1)}$ and $\beta^{(2)} = -\beta^{(1)}$, respectively (see Figure 2 for an example). Finally, for the DGP `"rost3"` a third component is added with item parameters $\beta^{(3)} = (-0.5, 0.5, -0.5, 0.5, -0.5, 0.5, -0.5, 0.5, -0.5, 0.5)^\top$. The prior probabilities for the latent classes with item parameters $\beta^{(1)}$, $\beta^{(2)}$, and $\beta^{(3)}$ are 4/9, 2/9, and 3/9 respectively. In all three DGPs, the person parameters $\theta$ are drawn from a discrete uniform distribution on $\{2.7,\ 0.9, -0.9, -2.7\}$, except for the third class of DGP `"rost3"` which uses only one level of ability, drawn from the before-mentioned set of four ability levels. In all DGPs, response vectors for 1800 subjects are initially drawn but the extreme scorers who solved either none or all items are excluded.

Here, a dataset from the second DGP is generated along with two artificial covariates `x1` and `x2`. Covariate `x1` is an informative binary variable (i.e., correlated with the true group membership) while `x2` is an uninformative continuous variable.

```
R> set.seed(1)
R> r2 <- simRaschmix(design = "rost2")
R> d <- data.frame(
+    x1 = rbinom(nrow(r2), prob = c(0.4, 0.6)[attr(r2, "cluster")], size = 1),
+    x2 = rnorm(nrow(r2))
+  )
R> d$resp <- r2
```

The Rost (1990) version of the Rasch mixture model – i.e., with a saturated score model and without concomitant variables – is fitted for one to three components. As no concomitants are employed in this model flavor, the matrix `r2` can be passed to `raschmix()` without formula:

```
R> set.seed(2)
R> m1 <- raschmix(r2, k = 1:3)
R> m1

Call:
raschmix(formula = r2, k = 1:3)

  iter converged k k0     logLik      AIC      BIC      ICL
```

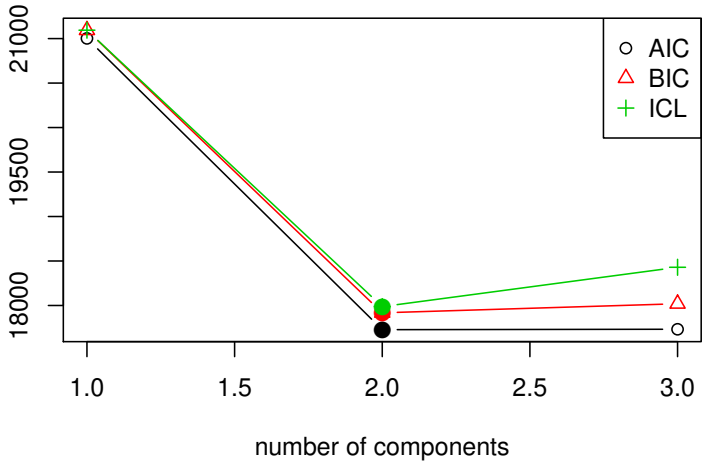


Figure 1: Information criteria for Rost's model with $K = 1, 2, 3$ components for the artificial scenario 2 data.

```
1    2      TRUE 1  1 -10484.227 21002.45 21094.39 21094.39
2    9      TRUE 2  2  -8829.038 17728.08 17917.35 17987.30
3  106      TRUE 3  3  -8813.362 17732.72 18019.34 18429.40
```

To inspect the results, the returned object can either be printed, as illustrated above, or plotted yielding a visualization of information criteria (see Figure 1). Both printed display and visualization show a big difference in information criteria across number of components $K$, with the minimum always being assumed for $K = 2$, thus correctly recovering the two latent classes constructed in the underlying DGP.

The values of the information criteria can also be accessed directly via the functions of the corresponding names. To select a certain model from a "`stepRaschmix`" object, the `getModel()` function from the **flexmix** package can be employed. The specification of `which` model is to be selected can either be an information criterion, or the number of components as a string, or the index of the model in the original vector `k`. In this particular case, `which = "BIC"`, `which = "2"`, and `which = 2` would all return the model with $K = 2$ components.

```
R> BIC(m1)

       1        2        3
21094.39 17917.35 18019.34

R> m1b <- getModel(m1, which = "BIC")
R> summary(m1b)

Call:
raschmix(formula = r2, k = 2)
```

```
       prior size post>0 ratio
Comp.1   0.5  819   1285 0.637
Comp.2   0.5  830   1301 0.638

Item Parameters:
           Comp.1     Comp.2
Item01 -2.5461200  2.6278031
Item02 -2.1053835  2.1250746
Item03 -1.6716294  1.3812228
Item04 -1.0293901  0.8535203
Item05 -0.2233486  0.3129260
Item06  0.2737782 -0.2937386
Item07  0.9561390 -0.8701059
Item08  1.5512468 -1.4464421
Item09  2.0670820 -2.0540182
Item10  2.7276257 -2.6362421

'log Lik.' -8829.038 (df=35)
AIC: 17728.08   BIC: 17917.35
```

To inspect the main properties of the model, `summary()` can be called. The information about the components of the mixture includes a priori component weights $\pi_k$ and sizes as well as the estimated item parameters $\hat{\beta}$ per component. Additionally, the fitted log-likelihood and the information criteria AIC and BIC are reported. As one of the item parameters in the Rasch model is not identified, a restriction needs to be applied to the item parameters. In the output of the `summary()` function, the item parameters of each component are scaled to sum to zero.

Two other functions, `worth()` and `parameters()`, can be used to access the item parameters. The sum restriction employed in the `summary()` output is also applied by `worth()`. Additionally, `worth()` provides the possibilities to select several or just one specific `component` and to transform item `difficulty` parameters to item easiness parameters. The function `parameters()` also offers these two options but restricts the first item parameter to be zero (rather than to restrict the sum of item parameters), as this restriction is used in the internal computations. Thus, for the illustrative dataset with 10 items, `parameters()` returns 9 item parameters, leaving out the first item parameter restricted to zero while `worth()` returns 10 item parameters summing to zero. The latter corresponds to the parametrization employed by Rost (1990) and `simRaschmix()`. For convenience reasons, the true parameters are attached to the simulated dataset as an attribute named `"difficulty"`. These are printed below and visualized in Figure 2 (left), showing that all item parameters are recovered rather well. Note that the ordering of the components in mixture models is generally arbitrary.

```
R> parameters(m1b, "item")

               Comp.1     Comp.2
item.Item01        NA         NA
item.Item02 0.4407365 -0.5027285
```

```
item.Item03 0.8744906 -1.2465803
item.Item04 1.5167298 -1.7742828
item.Item05 2.3227714 -2.3148771
item.Item06 2.8198982 -2.9215417
item.Item07 3.5022589 -3.4979090
item.Item08 4.0973667 -4.0742453
item.Item09 4.6132020 -4.6818213
item.Item10 5.2737457 -5.2640452
```

```
R> worth(m1b)
```

```
           Comp.1     Comp.2
Item01 -2.5461200  2.6278031
Item02 -2.1053835  2.1250746
Item03 -1.6716294  1.3812228
Item04 -1.0293901  0.8535203
Item05 -0.2233486  0.3129260
Item06  0.2737782 -0.2937386
Item07  0.9561390 -0.8701059
Item08  1.5512468 -1.4464421
Item09  2.0670820 -2.0540182
Item10  2.7276257 -2.6362421
```

```
R> attr(r2, "difficulty")
```

```
      [,1] [,2]
 [1,]  2.7 -2.7
 [2,]  2.1 -2.1
 [3,]  1.5 -1.5
 [4,]  0.9 -0.9
 [5,]  0.3 -0.3
 [6,] -0.3  0.3
 [7,] -0.9  0.9
 [8,] -1.5  1.5
 [9,] -2.1  2.1
[10,] -2.7  2.7
```

In addition to the item parameters, the `parameters()` function can also return the parameters of the `"score"` model and the `"concomitant"` model (if any). The type of parameters can be set via the `which` argument. Per default `parameters()` returns both item and score parameters.

A comparison between estimated and true class membership can be conducted using the `clusters()` function and the corresponding attribute of the data, respectively. As already noticeable from the item parameters, the first component of the mixture matches the second true group of the data and vice versa. This label-switching property of mixture models in general can also be seen in the cross-table of class memberships. We thus have 32 misclassifications among the 1649 observations.

```
R> table(model = clusters(m1b), true = attr(r2, "cluster"))
```

```
     true
model   1   2
    1  14 805
    2 812  18
```

For comparison, a Rasch mixture model with mean/variance parametrization for the score probabilities, as introduced in Section 2.3, is fitted with one to three components and the best BIC model is selected.

```
R> set.seed(3)
R> m2 <- raschmix(data = r2, k = 1:3, scores = "meanvar")
```

```
R> m2
```

```
Call:
raschmix(data = r2, k = 1:3, scores = "meanvar")

  iter converged k k0     logLik      AIC      BIC      ICL
1    2      TRUE 1  1 -10486.816 20995.63 21055.12 21055.12
2    9      TRUE 2  2  -8834.887 17715.77 17840.16 17910.24
3   70      TRUE 3  3  -8827.596 17725.19 17914.47 18467.25
```

```
R> m2b <- getModel(m2, which = "BIC")
```

As in the saturated version of the Rasch mixture model, all three information criteria prefer the two-component model. Thus, this version of a Rasch mixture model is also capable of recognizing the two latent classes in the data while using a more parsimonious parametrization with 23 instead of 35 parameters.

```
R> logLik(m2b)
```

```
'log Lik.' -8834.887 (df=23)
```

```
R> logLik(m1b)
```

```
'log Lik.' -8829.038 (df=35)
```

The estimated parameters of the distribution of the score probabilities can be accessed through `parameters()` while the full set of score probabilities is returned by `scoreProbs()`. The estimated score probabilities of the illustrative model are approximately equal across components and roughly uniform.

```
R> parameters(m2b, which = "score")
```

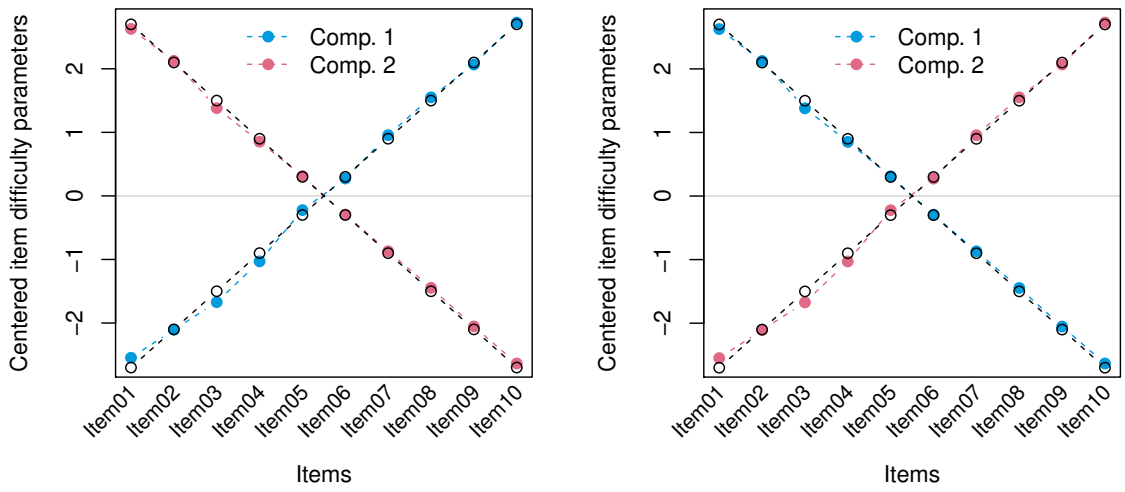


Figure 2: True (black) and estimated (blue/red) item parameters for the two model specifications, `"saturated"` (left) and `"meanvar"` (right), for the artificial scenario 2 data.

```
                    Comp.1     Comp.2
score.location     0.1655927  0.1001153
score.dispersion -0.1680513 -0.2514496
```

```
R> scoreProbs(m2b)
```

```
          Comp.1    Comp.2
 [1,] 0.0000000 0.0000000
 [2,] 0.1105324 0.1170100
 [3,] 0.1072126 0.1101524
 [4,] 0.1054000 0.1058038
 [5,] 0.1050205 0.1036919
 [6,] 0.1060587 0.1036871
 [7,] 0.1085569 0.1057891
 [8,] 0.1126178 0.1101268
 [9,] 0.1184119 0.1169719
[10,] 0.1261893 0.1267671
[11,] 0.0000000 0.0000000
```

The resulting item parameters for this particular data set are virtually identical to those from the saturated version, as can be seen in Figure 2.

To demonstrate the use of a concomitant variable model for the weights of the mixture, the two artificial variables `x1` and `x2` are employed. They are added on the right-hand side of the formula, yielding a multinomial logit model for the weights (only if `k = 2` or more components are specified).

```
R> set.seed(4)
R> cm2 <- raschmix(resp ~ x1 + x2, data = d, k = 1:3, scores = "meanvar")
```

The BIC is used to compare the models with and without concomitant variables and different number of components. The two true groups are recognized correctly with and without concomitant variables, while the model with concomitants manages to employ the additional information and reaches a somewhat improved model fit.

```
R> rbind(m2 = BIC(m2), cm2 = BIC(cm2))
```

```
           1        2        3
m2  21055.12 17840.16 17914.47
cm2 21055.12 17776.30 17865.63
```

While the likelihood ratio (LR) test cannot be employed to choose the number of components in a mixture model, it can be used to assess the concomitant variable model for a mixture model with a fixed number of components. Testing the 3-component model with concomitant variables against the 3-component model without concomitant variables yields a test statistic of 78.67 ($p < 0.001$).

As mentioned above, the parameters of the concomitant model can be accessed via the `parameters()` function, setting `which = "concomitant"`. The influence of the informative covariate `x1` is reflected in the large absolute coefficient while the estimated coefficient for the noninformative covariate `x2` is close to zero.

```
R> cm2b <- getModel(cm2, which = "BIC")
R> parameters(cm2b, which = "concomitant")
```

```
            1           2
(Intercept) 0 -0.45751346
x1          0  0.91232669
x2          0 -0.02908466
```

The corresponding estimated item parameters `parameters(cm2b, "item")` are not very different from the previous models (and are hence not shown here). This illustrative application shows that the inclusion of concomitant variables can provide additional information, e.g., that `x1` but not `x2` is associated with the class membership. Note also that this is picked up although a rather weak association was simulated here.

```
R> table(x1 = d$x1, clusters = clusters(cm2b))
```

```
   clusters
x1    1   2
  0 501 318
  1 325 505
```

## 4. Empirical application: Verbal aggression

The verbal aggression dataset (De Boeck and Wilson 2004) contains item response data from 316 first-year psychology students along with gender and trait anger (assessed by the Dutch adaptation of the state-trait anger scale) as covariates (Smits, De Boeck, and Vansteelandt 2004). The 243 women and 73 men responded to 24 items constructed the following way: Following the description of a frustrating situation, subjects are asked to agree or disagree with a possible reaction. The situations are described by the following four sentences:

- S1: A bus fails to stop for me.
- S2: I miss a train because a clerk gave me faulty information.
- S3: The grocery store closes just as I am about to enter.
- S4: The operator disconnects me when I had used up my last 10 cents for a call.

Each reaction begins with either "I want to" or "I do" and is followed by one of the three verbally aggressive reactions "curse", "scold", or "shout", e.g., "I want to curse", "I do curse", "I want to scold", or "I do scold".

For our illustration, we use only the first two sentences which describe situations in which the others are to blame. Extreme-scoring subjects agreeing with either none or all responses are removed.

```
R> data("VerbalAggression", package = "psychotools")
R> VerbalAggression$resp2 <- VerbalAggression$resp2[, 1:12]
R> va12 <- subset(VerbalAggression,
+    rowSums(resp2) > 0 & rowSums(resp2) < 12)
R> colnames(va12$resp2)

 [1] "S1WantCurse" "S1DoCurse"   "S1WantScold" "S1DoScold"
 [5] "S1WantShout" "S1DoShout"   "S2WantCurse" "S2DoCurse"
 [9] "S2WantScold" "S2DoScold"   "S2WantShout" "S2DoShout"
```

We fit Rasch mixture models with the mean/variance score model, one to four components, and with and without the two concomitant variables, respectively (the single component model being only fitted without covariates).

```
R> set.seed(1)
R> va12_mix1 <- raschmix(resp2 ~ 1, data = va12, k = 1:4, scores = "meanvar")
R> set.seed(2)
R> va12_mix2 <- raschmix(resp2 ~ gender + anger, data = va12, k = 1:4,
+    scores = "meanvar")
```

The corresponding BIC for all considered models can be computed by

```
R> rbind(BIC(va12_mix1), BIC(va12_mix2))
```

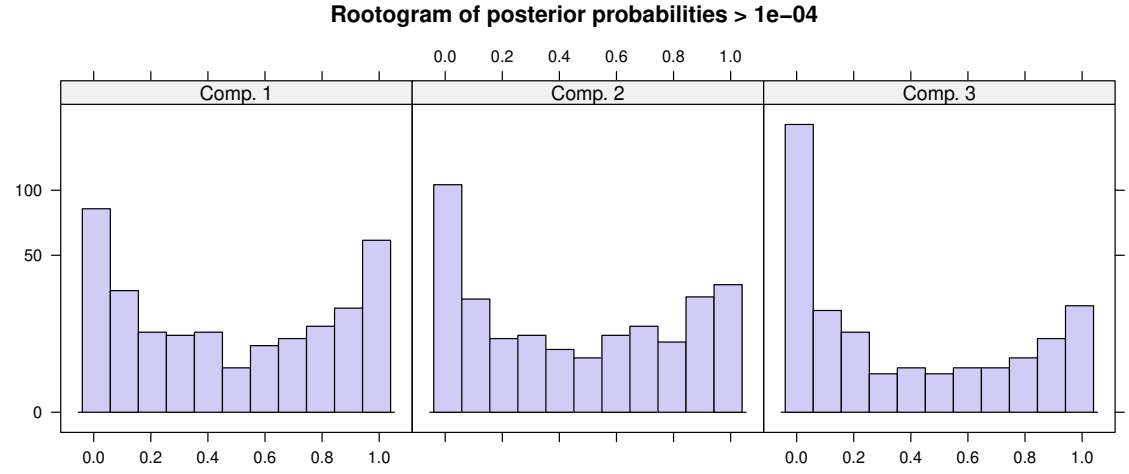


Figure 3: Rootogram of posterior probabilities in the 3-component Rasch mixture model on verbal aggression data.

```
             1        2        3        4
[1,] 3874.632 3857.549 3854.355 3889.782
[2,] 3874.632 3859.120 3854.824 3880.485
```

```
R> va12_mix3 <- getModel(va12_mix2, which = "3")
```

showing that three components are preferred regardless of whether or not concomitant variables are used. The difference in BIC between the models with and without concomitant variables is very small and the LR test yields a test statistic of 21.97 ($p < 0.001$), thus the 3-component model with concomitant variables is chosen.

The posterior probabilities for the three components can be visualized via `histogram(va12_mix3)` – by default using a square-root scale, yielding a so-called rootogram – as shown in Figure 3. In the ideal case, posterior probabilities of the observations for each component are either high or low, yielding a U-shape in all panels. In this case here, the components are separated acceptably well.

The item profiles in the three components can be visualized via `plot(va12_mix3)` or `xyplot(va12_mix3)` with the output of the latter being shown in Figure 4. The first six items are responses to the first sentence (bus), the remaining six refer to the second sentence (train). The six reactions are grouped in "want"/"do" pairs: first for "curse", then "scold", and finally "shout".

The first component displays a zigzag pattern which indicates that subjects in this component always find it easier or less extreme to "want to" react a certain way rather than to actually "do" react that way. In the other two components this want/do relationship is reversed, except for the shouting response (to either situation) and the scolding response to the train situation (S2) in the second component.

In the third component, there are no big differences in the estimated item parameters. Neither any situation (S1 or S2) nor any type of verbal response (curse, scold, or shout) is perceived as particularly extreme by subjects in this component. In components 1 and 2, the situation

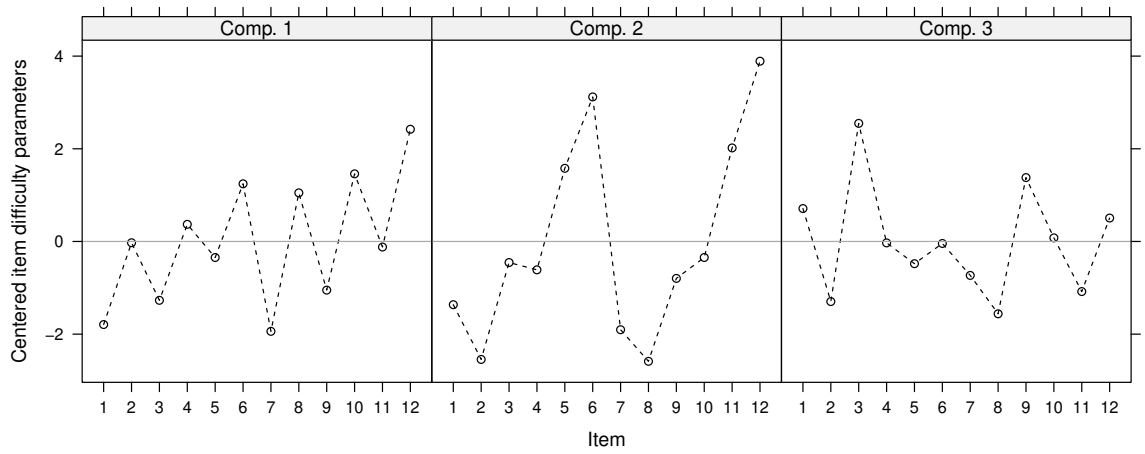


Figure 4: Item profiles for the 3-component Rasch mixture model on verbal aggression data. Items 1–6 pertain to situation S1 (bus), items 7–12 to situation S2 (train), each in the following order: want to curse, do curse, want to scold, do scold, want to shout, do shout.

is also not very relevant but subjects differentiate between the three verbal responses. This is best visible in component 2 where item difficulty is clearly increasing from response "curse" to response "shout". Thus, shouting is perceived as the most extreme verbal response while cursing is considered a comparably moderate response. In component 1 this pattern is also visible albeit not as prominently as in component 2.

The link between the covariates and the latent classes is described through the concomitant variable model:

```
R> parameters(getModel(va12_mix2, which = "3"), which = "concomitant")
```

```
            1           2         3
(Intercept) 0 -0.76273585 -3.674368
gendermale  0  1.66347276  1.415727
anger       0  0.01141244  0.126756
```

The absolute sizes of the coefficients reflect that there may be some association with gender but less with the anger score. However, as there is a slight increase in BIC compared to the model without concomitants, the association with the covariates appears to be relatively weak. In comparison to other approaches exploring the association of class membership with covariates *ex post* (e.g., as in Cohen and Bolt 2005), the main advantage of the concomitant variables model lies in the *simultaneous* estimation of the mixture and the influence of covariates.

## 5. Summary

Mixtures of Rasch models are a flexible means of checking measurement invariance and testing for differential item functioning. Here, we establish a rather general unifying conceptual framework for Rasch mixture models along with the corresponding computational tools in

the R package **psychomix**. In particular, this includes the original model specification of Rost (1990) as well as more parsimonious parametrizations (Rost and von Davier 1995), along with the possibility to incorporate concomitant variables predicting the latent classes (as in Tay *et al.* 2011).

The R implementation is based on the infrastructure provided by the **flexmix** package, allowing for convenient model specification and selection. The rich set of methods for **flexmix** objects is complemented by additional functions specifically designed for Rasch models, e.g., extracting different types of parameters in different transformations and visualizing the estimated component-specific item parameters in various ways. Optionally, speed gains can be obtained from utilizing the C++ implementation in the **mRm** package for selecting optimal starting values. Thus, **psychomix** provides a comprehensive and convenient toolbox for the application of Rasch mixture models in psychometric research practice.

## Acknowledgments

We are thankful to the participants of the "Psychoco 2011" workshop at Universität Tübingen for helpful feedback and discussions.

## References

Ackerman TA (1992). "A Didactic Explanation of Item Bias, Item Impact, and Item Validity from a Multidimensional Perspective." *Journal of Educational Measurement*, **29**(1), 67–91.

Andersen EB (1972). "A Goodness of Fit Test for the Rasch Model." *Psychometrika*, **38**(1), 123–140.

Cohen AS, Bolt DM (2005). "A Mixture Model Analysis of Differential Item Functioning." *Journal of Educational Measurement*, **42**(2), 133–148.

Dayton CM, Macready G (1988). "Concomitant-Variable Latent-Class Models." *Journal of the American Statistical Association*, **83**(401), 173–178.

De Boeck P, Wilson M (eds.) (2004). *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach*. Springer-Verlag, New York.

Dempster A, Laird N, Rubin D (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–38.

Fischer GH (1995). "Derivations of the Rasch Model." In Fischer and Molenaar (1995), chapter 2, pp. 15–38.

Fischer GH, Molenaar IW (eds.) (1995). *Rasch Models: Foundations, Recent Developments, and Applications*. Springer-Verlag, New York.

Frick H, Strobl C, Leisch F, Zeileis A (2012). "Flexible Rasch Mixture Models with Package **psychomix**." *Journal of Statistical Software*, **48**(7), 1–25. URL http://www.jstatsoft.org/v48/i07/.

Glas CAW, Verhelst ND (1995). "Testing the Rasch Model." In Fischer and Molenaar (1995), chapter 5, pp. 69–95.

Grün B, Leisch F (2008). "FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters." *Journal of Statistical Software*, **28**(4), 1–35. URL http://www.jstatsoft.org/v28/i04/.

Leisch F (2004). "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8), 1–18. URL http://www.jstatsoft.org/v11/i08/.

Mair P, Hatzinger R (2007). "Extended Rasch Modeling: The **eRm** Package for the Application of IRT Models in R." *Journal of Statistical Software*, **20**(9), 1–20. URL http://www.jstatsoft.org/v20/i09/.

Molenaar IW (1995a). "Estimation of Item Parameters." In Fischer and Molenaar (1995), chapter 3, pp. 39–51.

Molenaar IW (1995b). "Some Background for Item Response Theory and the Rasch Model." In Fischer and Molenaar (1995), chapter 1, pp. 3–14.

Preinerstorfer D, Formann AK (2011). "Parameter Recovery and Model Selection in Mixed Rasch Models." *British Journal of Mathematical and Statistical Psychology*, **65**(2), 251–262.

Rasch G (1960). *Probabilistic Models for Some Intelligence and Attainment Tests.* The University of Chicago Press.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Rijmen F, De Boeck P (2005). "A Relationship Between a Between-Item Multidimensional IRT Model and the Mixture Rasch Model." *Psychometrika*, **70**(3), 481–496.

Rizopoulos D (2006). "**ltm**: An R Package for Latent Variable Modeling and Item Response Theory Analyses." *Journal of Statistical Software*, **17**(5), 1–25. URL http://www.jstatsoft.org/v17/i05/.

Rost J (1990). "Rasch Models in Latent Classes: An Integration of Two Approaches to Item Analysis." *Applied Psychological Measurement*, **14**(3), 271–282.

Rost J, von Davier M (1995). "Mixture Distribution Rasch Models." In Fischer and Molenaar (1995), chapter 14, pp. 257–268.

Smits DJM, De Boeck P, Vansteelandt K (2004). "The Inhibition of Verbally Aggressive Behaviour." *European Journal of Personality*, **18**(7), 537–555.

Strobl C, Kopf J, Zeileis A (2011). "A New Method for Detecting Differential Item Functioning in the Rasch Model." *Working Paper 2011-01*, Working Papers in Economics and Statistics, Research Platform Empirical and Experimental Economics, Universität Innsbruck. URL http://EconPapers.RePEc.org/RePEc:inn:wpaper:2011-01.

Tay L, Newman DA, Vermunt JK (2011). "Using Mixed-Measurement Item Response Theory with Covariates (MM-IRT-C) to Ascertain Observed and Unobserved Measurement Equivalence." *Organizational Research Methods*, **14**(1), 147–176.

Van den Noortgate W, De Boeck P (2005). "Assessing and Explaining Differential Item Functioning Using Logistic Mixed Models." *Journal of Educational and Behavioral Statistics*, **30**(4), 443–464.

Zeileis A, Strobl C, Wickelmaier F (2011). ***psychotools****: Infrastructure for Psychometric Modeling*. R package version 0.1-1, URL http://CRAN.R-project.org/package=psychotools.

## A. Using the FLXMCrasch() driver directly with stepFlexmix()

The "FLXM" driver FLXMCrasch(), underlying the raschmix() function, can also be used directly with flexmix() or stepFlexmix(), respectively, via the model argument. To do so, essentially the same arguments as in raschmix() (see Section 3.1) can be used, however, they need to be arranged somewhat differently. The formula just specifies the item responses, while the concomitant variables need to be passed to the concomitant argument in a suitable "FLXP" driver (see Grün and Leisch 2008). Also some arguments, such as the scores distribution have to be specified in the FLXMCrasch() driver for the model argument. As an example, consider replication of the cm2 model fit on the artificial data from Section 3.3:

```
R> set.seed(4)
R> fcm2 <- stepFlexmix(resp ~ 1, data = d, k = 1:3,
+    model = FLXMCrasch(scores = "meanvar"),
+    concomitant = FLXPmultinom(~ x1 + x2))
```

Thus, there is some more flexibility but somewhat less convenience when using flexmix() or stepFlexmix() directly as opposed through the raschmix() interface. This is also reflected in the objects returned which are of class "flexmix" or "stepFlexmix", not class "raschmix" or "stepRaschmix", respectively. Thus, only the generic functions for those objects apply and not the additional ones specific to Rasch mixture models. In various cases, the methods are inherited or reused from **flexmix** and thus behave identically, e.g., for BIC() or getModel().

```
R> rbind(cm2 = BIC(cm2), fcm2 = BIC(fcm2))

            1       2        3
cm2  21055.12 17776.3 17865.63
fcm2 21055.12 17776.3 17865.63

R> fcm2b <- getModel(fcm2, which = "BIC")
```

For other methods, such as parameters(), the methods in **psychomix** offer more convenience. For example, the concomitant model coefficients can be accessed in the same way

```
R> cbind(parameters(cm2b, which = "concomitant"),
+    parameters(fcm2b, which = "concomitant"))

            1           2 1           2
(Intercept) 0 -0.45751346 0 -0.45751346
x1          0  0.91232669 0  0.91232669
x2          0 -0.02908466 0 -0.02908466
```

while the item and score parameters cannot be accessed separately (as it is possible for "raschmix" objects). They can only be accessed jointly as the "model" parameters. The method for "raschmix" objects also excludes non-identified or aliased parameters, e.g., the parameter for the achor item.

```
R> parameters(fcm2b, which = "model")
```

```
                Comp.1     Comp.2
item.Item01  0.0000000  0.0000000
item.Item02 -0.5088476  0.4457184
item.Item03 -1.2543141  0.8791793
item.Item04 -1.7796107  1.5171980
item.Item05 -2.3292419  2.3303544
item.Item06 -2.9354540  2.8248596
item.Item07 -3.5064344  3.4993600
item.Item08 -4.0819434  4.0918594
item.Item09 -4.6987642  4.6150243
item.Item10 -5.2710875  5.2645455
score1       0.1643237  0.1006912
score2      -0.1703023 -0.2488384
```

To sum up, some (convenience) functionality which is specific for Rasch mixture models is only available for objects of class "`raschmix`", e.g., the score probabilities via `scoreProbs()` or the item profiles via the default `plot()` method among others. On the other hand, functionality which is applicable to mixture models in general is inherited or preserved as part of the additional methods.

## Affiliation:

Hannah Frick, Achim Zeileis
Department of Statistics
Universität Innsbruck
Universitätsstr. 15
6020 Innsbruck, Austria
E-mail: Hannah.Frick@uibk.ac.at, Achim.Zeileis@R-project.org
URL: http://eeecon.uibk.ac.at/~frick/, http://eeecon.uibk.ac.at/~zeileis/

Carolin Strobl
Department of Psychology
Universität Zürich
Binzmühlestr. 14
8050 Zürich, Switzerland
E-mail: Carolin.Strobl@psychologie.uzh.ch
URL: http://www.psychologie.uzh.ch/fachrichtungen/methoden.html

Friedrich Leisch
Institute of Applied Statistics and Computing
Universität für Bodenkultur Wien
Peter Jordan-Str. 82
1180 Wien, Austria
E-mail: Friedrich.Leisch@R-project.org
URL: http://www.rali.boku.ac.at/friedrichleisch.html