



Mokken Scale Analysis in R

L. Andries van der Ark
Tilburg University

Abstract

Mokken scale analysis (MSA) is a scaling procedure for both dichotomous and polytomous items. It consists of an item selection algorithm to partition a set of items into Mokken scales and several methods to check the assumptions of two nonparametric item response theory models: the monotone homogeneity model and the double monotonicity model. First, we present an R package **mokken** for MSA and explain the procedures. Second, we show how to perform MSA in R using test data obtained with the Adjective Checklist.

Keywords: nonparametric item response theory, Mokken scale analysis, monotonicity, R.

1. Introduction

In this paper an R package ([R Development Core Team 2007](#)), called **mokken**, for Mokken scale analysis (MSA) is discussed. MSA is a scaling technique for ordinal data and mainly used for scaling test and questionnaire data. MSA is closely related to nonparametric item response theory (IRT) models which imply ordinal measurement. MSA consists of two parts: (1) an automated selection algorithm which partitions a set of ordinal variables (from here on called items) into scales (called Mokken scales) satisfying criteria related to nonparametric IRT models and possibly leaving some items unselected, and (2) methods to investigate assumptions of nonparametric IRT models.

The paper provides a short summary of the main concepts in MSA but is by no means exhaustive. For a more thorough discussion of nonparametric IRT, MSA, and data analysis strategies we refer to the following literature. Nonparametric IRT and MSA for dichotomous item scores were developed by [Mokken \(1971\)](#); also see [Mokken and Lewis, 1982](#)) and extended to polytomous items scores by [Molenaar \(1991, 1997\)](#). [Sijtsma and Molenaar \(2002\)](#) gave an overview of nonparametric IRT and MSA, and provided many references and examples. [Meijer and Baneke \(2004\)](#) demonstrated how MSA can be used preliminary to parametric IRT.

Currently available software for MSA are a commercial package called **MSP5** for Windows (Molenaar and Sijtsma 2000) and a Stata module (Weesie 1999).

The remainder of the paper is organized as follows. Section 2 discusses nonparametric IRT models and several methods to check model assumptions. Section 3 discusses the functions in **mokken**. Section 4 gives a demonstration of MSA by applying the functions in **mokken** to personality test data.

2. Mokken scale analysis

2.1. Nonparametric IRT models

Suppose a test or a questionnaire contains a set of items which are numbered $1, \dots, J$ and indexed by j . For convenience, but without loss of generality, suppose that each item has $m+1$ ordered answer categories. Let X_j denote the score on item j with realization $x_j = 0, 1, \dots, m$. If $m = 1$ the item is called dichotomous; if $m > 1$ the item is called polytomous. The sum score is defined as $X_+ = \sum_{j=1}^J X_j$. In IRT it is assumed that a (possibly multidimensional) latent trait θ triggers the item responses. It is also assumed that the ordering of the scores of each item reflects the hypothesized ordering on θ . Expression $X_j \geq x$ is called an *item step* (Sijtsma and Molenaar 2002, p. 122) and $P(X_j \geq x|\theta)$ is called the *item step response function*. Because $P(X_j \geq 0|\theta) = 1$ for all θ , the relation between item j and θ is characterized by m item step response functions: $P(X_j \geq 1|\theta), \dots, P(X_j \geq m|\theta)$. For dichotomous items the item step response function reduces to $P(X = 1|\theta)$.

Four assumptions define the two most popular nonparametric IRT models.

Unidimensionality : θ is a unidimensional latent variable;

Local independence : $P(X_1 = x_1, \dots, X_J = x_J|\theta) = \prod_{j=1}^J P(X_j = x_j|\theta)$;

Latent monotonicity : $P(X_j \geq x|\theta_a) \leq P(X_j \geq x|\theta_b)$, for all $\theta_a < \theta_b$ for $j = 1, \dots, J$; $x = 1, \dots, m$; and

Nonintersection : if for a fixed value θ_0 $P(X_i \geq x|\theta_0) \geq P(X_j \geq y|\theta_0)$ then $P(X_i \geq x|\theta) \geq P(X_j \geq y|\theta)$ for all θ . This is true for all pairs of items $i \neq j$ and for all pairs of item scores x, y .

Local independence implies that the item responses only depend on θ . Latent monotonicity means that the item step response functions are nondecreasing functions of θ . (Latent monotonicity is usually referred to as monotonicity, but we prefer the term latent monotonicity to distinguish it from manifest monotonicity that it introduced later on.) Nonintersection means that the item step response functions do not intersect.

The assumptions unidimensionality, local independence, and latent monotonicity define the most general nonparametric IRT model: the monotone homogeneity model (Mokken 1971) also known as the nonparametric graded response model (Hemker, Sijtsma, Molenaar, and Junker 1997). Assumptions unidimensionality, local independence, latent monotonicity, and nonintersection define the double monotonicity model (Mokken 1971). Several other nonparametric IRT models have been proposed (see van der Ark 2001, for an overview). All popular

unidimensional parametric IRT models, such as the Rasch model (Rasch 1960), the two- and three-parameter logistic model (Birnbaum 1968), the graded response model (Samejima 1969), also assume unidimensionality, local independence, and latent monotonicity. Therefore, investigation of the assumptions of nonparametric IRT models is also useful when parametric IRT models are used. In addition, parametric IRT models assume that the item step response functions have a parametric functional form.

Nonparametric IRT models have the following measurement properties. For dichotomous items, the monotone homogeneity model implies stochastic ordering of θ by X_+ (known under the acronym SOL), i.e.,

$$P(\theta > a | X_+ = L) \geq P(\theta > a | X_+ = K) \text{ for all } a \text{ and for all } K < L$$

(Hemker, Sijtsma, Molenaar, and Junker 1996; also see, Grayson 1988; Huynh 1994). Because the monotone homogeneity model is the most general IRT model, SOL also holds for other popular IRT models for dichotomous item scores. In general, SOL does not hold for IRT models for polytomous item scores (Hemker *et al.* 1997) but for most models violations are rare if the number of items exceeds five (van der Ark 2005). For dichotomous items, the double monotonicity model allows an invariant ordering of the items on θ . For polytomous items this is not the case. Sijtsma and Junker (1996) and Sijtsma and Hemker (1998) provided more details on item ordering.

2.2. Scalability coefficients

For each pair of items, there is an item-pair scalability coefficient H_{ij} ; $i, j = 1, \dots, J$ (Molenaar 1991). Let $\text{COV}(X_i, X_j)$ be the covariance between X_i and X_j , and let $\text{COV}(X_i, X_j)^{\max}$ be the maximum covariance between X_i and X_j given the marginal distributions of X_i and X_j . If the variance of the scores on item i and item j are both positive, then H_{ij} is the normed covariance between the item scores:

$$H_{ij} = \frac{\text{COV}(X_i, X_j)}{\text{COV}(X_i, X_j)^{\max}}. \quad (1)$$

If X_i or X_j have zero variance, H_{ij} can still be computed (Molenaar 1991) but Equation 1 is no longer true. In MSA, items belonging to the same Mokken scale should have positive item-pair scalability coefficients.

For each item, there is an item scalability coefficient H_j ; $j = 1, \dots, J$ (Molenaar 1991). Let $R_{-j} = X_+ - X_j$; R_{-j} is called the *rest score*. Let $\text{COV}(X_j, R_{-j})$ be the covariance between X_j and R_{-j} , and let $\text{COV}(X_j, R_{-j})^{\max}$ be the maximum covariance between X_j and R_{-j} given the marginal distributions of X_j and R_{-j} . If X_j and R_{-j} both have positive variance, then H_j is the normed covariance between the item score and the rest score:

$$H_j = \frac{\text{COV}(X_j, R_{-j})}{\text{COV}(X_j, R_{-j})^{\max}}. \quad (2)$$

In MSA, items belonging to the same Mokken scale should have an item scalability coefficient greater than some positive lower bound c . As a rule of thumb $c > .3$ (Sijtsma and Molenaar 2002). van Abswoude, van der Ark, and Sijtsma (2004) argued that H_j can be interpreted in a similar way as the discrimination parameters in parametric IRT.

For the entire set of items, there is a test scalability coefficient H :

$$H = \frac{\sum_{j=1}^J \text{COV}(X_j, R_{-j})}{\sum_{j=1}^J \text{COV}(X_j, R_{-j})^{\max}}.$$

If $H = 1$ the test data follow a perfect Guttman scalogram. Mokken (1971) proposed the following rules of thumb for H . A scale is considered weak if $.3 \leq H < .4$, a scale is considered moderate if $.4 \leq H < .5$, and a scale is considered strong if $H > .5$.

Assumptions unidimensionality, local independence, and latent monotonicity imply the following testable restrictions on the scalability coefficients (Sijtsma and Molenaar 2002, Theorem 4.3): $0 \leq H_{ij} \leq 1$, for all $i \neq j$; $0 \leq H_j \leq 1$, for all j ; and $0 \leq H \leq 1$. For an extensive discussion of the relationship between the scalability coefficients, we refer to Sijtsma and Molenaar (2002, Chapter 4) and van Abswoude *et al.* (2004).

2.3. Automated item selection algorithm

An important part of MSA is the partitioning of a set of items into Mokken scales and possibly a set of unscalable items. Mokken (1971, p. 184) defined a Mokken scale as a set of dichotomously scored items for which, for a suitably chosen positive lower bound c , all inter-item covariances are strictly positive and $H_j \geq c > 0$. This definition can be readily generalized to polytomously scored items. Partitioning a set of items into Mokken scales is done using an automated item selection algorithm (Mokken 1971, pp. 190–193) described in detail by Sijtsma and Molenaar (2002, Chapter 5) and van Abswoude *et al.* (2004). Two parameters, lower bound c and nominal significance level α , have to be specified by the researcher. Lower bound c defines the minimum value of coefficients H_j in the Mokken scale. The recommended default value is $c = .3$ (Molenaar and Sijtsma 2000). Parameter α is the nominal significance level of the inequality tests used in the automated item selection algorithm and its recommended default value is .05.

2.4. Investigation of latent monotonicity

Manifest monotonicity is an observable property of the test data, and defined as

$$P(X_j \geq x | R_{-j} = s) \geq P(X_j \geq x | R_{-j} = r) \text{ for all } j, x, s > r. \quad (3)$$

Junker and Sijtsma (2000) showed that for dichotomous items latent monotonicity implies manifest monotonicity. For polytomous items, some counterexamples have been found (Junker and Sijtsma 2000) but Molenaar and Sijtsma (2000) assume that in practice, also for polytomous items, manifest monotonicity is a valid test of latent monotonicity.

A first practical issue when using manifest monotonicity to investigate latent monotonicity is that the number of respondents having $R_{-j} = r$ may be too small for an accurate estimation of $P(X_j \geq x | R_{-j} = r)$. This is solved by grouping respondents with adjacent rest scores until the size of the rest score group is greater a preset criterion called *minsize* (Molenaar and Sijtsma 2000, pp. 67-70). In fact, (3) becomes

$$P(X_j \geq x | R_{-j} \in \{s_1, \dots, s_n\}) \geq P(X_j \geq x | R_{-j} \in \{r_1, \dots, r_m\}) \text{ for all } j, x, s_1 > r_m,$$

where s_1, \dots, s_n and r_1, \dots, r_m are n and m consecutive integers, respectively. For the default method to construct rest score groups see [Molenaar and Sijtsma \(2000, p. 67\)](#). It is advised to investigate latent monotonicity several times using different values for `minsize`.

A second practical issue is that some violations of manifest monotonicity may be too small to be relevant. Therefore, only violations greater than `minvi` (default value is `.03`) are reported and for each reported violation a significance test at level $\alpha = .05$ (without Bonferroni correction) is computed ([Molenaar and Sijtsma 2000, p. 67](#)).

2.5. Investigation of nonintersection

[Molenaar and Sijtsma\(2000, pp. 74–88\)](#) describe three methods to investigate nonintersection: method `pmatrx`, method `restscore`, and method `restsplit` (method `restsplit` is not included in `mokken`). All three methods are a special case of the following implication. If local independence holds, then nonintersection,

$$P(X_i \geq x|\theta) \geq P(X_j \geq y|\theta) \text{ for all } \theta \quad (4)$$

implies manifest property

$$P(X_i \geq x|W = w) \geq P(X_j \geq y|W = w) \text{ for all } w, \quad (5)$$

where W is a manifest variable independent of X_i and X_j .

Method pmatrx

Method `pmatrx` was proposed by [Mokken \(1971, pp. 180–182\)](#). Manifest variable W in (5) is the dichotomized score on item k . Mokken showed that if (4) holds then

$$P(X_i \geq x, X_k \geq z) \geq P(X_j \geq y, X_k \geq z) \text{ for } z = 1, \dots, m; i \neq k; j \neq k, \quad (6)$$

and

$$P(X_i < x, X_k < z) \leq P(X_j < y, X_k < z) \text{ for } z = 1, \dots, m; i \neq k; j \neq k. \quad (7)$$

The joint probabilities $P(X_i \geq x, X_k \geq z)$ are collected in a $Jm \times Jm$ matrix called the $P(++)$ matrix. The rows and columns of the $P(++)$ matrix correspond to the Jm item steps ordered in popularity. The first row and column of the $P(++)$ matrix correspond to the least popular item step, the last row and column correspond to the most popular item step. Entries in the $P(++)$ matrix pertaining to the same item (i.e. $P(X_j \geq x, X_j \geq z)$ $j = 1, \dots, J; x = 1, \dots, m; z = 1, \dots, m$) are not considered. The $P(++)$ matrix can be used to test nonintersection in the following way. Equation (6) is equivalent to a $P(++)$ matrix that has nondecreasing entries both rowwise and columnwise.

Similarly, the $P(--)$ matrix contains the joint probabilities $P(X_i < x, X_j < y)$. The rows and columns of the $P(--)$ matrix correspond to the same ordered item steps as the rows and columns of the $P(++)$ matrix. Entries $P(X_j < x, X_j < z)$ ($j = 1, \dots, J; x = 1, \dots, m; z = 1, \dots, m$), i.e. entries pertaining to the same item, are not considered. Equation (7) is equivalent to a $P(--)$ matrix that has nonincreasing entries both rowwise and columnwise.

Method restscore

A second choice for W in (5) is rest score $R_{-i,-j} = X_+ - X_i - X_j$ (note that X_+ is not an adequate choice because it depends on X_i and X_j). Equation 5 then becomes

$$P(X_i \geq x|R_{-i,-j} = r) \geq P(X_j \geq y|R_{-i,-j} = r) \text{ for all } r.$$

Method `restscore` investigates nonintersection for each pair of items.

Similar practical issues as discussed in Section 2.4 apply to method `restscore`. First, the number of respondents having $R_{-i,-j} = r$ may be too small for an accurate estimation of $P(X_i \geq x | R_{-i,-j} = r)$ and rest score groups must be formed (Molenaar and Sijtsma 2000, pp. 74–78).

Second, some violations may be too small to be relevant. Therefore, only violations greater than `minvi` (default value is .03) are reported and for each reported violation a significance test at level $\alpha = .05$ (without Bonferroni correction) is computed (Molenaar and Sijtsma 2000, p. 78).

3. Description of the functions in `mokken`

The package `mokken` contains five principal functions. Except for the graphics, the function names and the output in `mokken` are similar to function names and output in the package `MSP5` for Windows (Molenaar and Sijtsma 2000). The graphics in `mokken` differ substantially from the graphics provided by `MSP5` for Windows. The functions in `mokken` were tested on several real and simulated data sets. In all occasions the results were identical to results obtained with `MSP5` for Windows.

1. `coefH`

Description: returns a list of the scalability coefficients (Section 2.2).

Usage: `coefH(X)`

Required arguments: **X**: matrix or data frame of numeric data containing the responses of N respondents to J items. Missing values are not allowed.

Value:

Hij: matrix containing item-pair scalability coefficients H_{ij} ,

Hj: vector containing item scalability coefficients H_j

H: scalability coefficient H .

Details: Does not work if any of the item scores has a variance equal to zero. Such items should not be used in a test and should be removed from the data frame.

2. `search.normal`

Description: returns a vector of J integers indicating the Mokken scale to which an item belongs (Section 2.2).

Usage: `search.normal(X, lowerbound = .3, alpha = .05)`

Required arguments: **X**: matrix or data frame of numeric data containing the responses of N respondents to J items. Missing values are not allowed.

Optional arguments: **lowerbound** and **alpha** are scalars (see Section 2.3)

Value: An indicator vector of length J . Each entry refers to an item. Items with same integer belong to the same Mokken scale. A zero indicates an unscalable item. If n is the largest integer, then n Mokken scales were found.

Details: The number of Mokken scales cannot exceed $J/2$.

3. `check.monotonicity`

Description: Returns the results from the investigation of latent monotonicity.

Usage: `check.monotonicity(X, minvi = .03, minsize = default.minsize)`

Required arguments: **X**: matrix or data frame of numeric data containing the responses of N respondents to J items. Missing values are not allowed.

Optional arguments:

minvi: minimum size of a violation that is reported (Molenaar and Sijtsma 2000, p. 71).

minsize: minimum size of a rest score group. By default `minsize = N/10` if $N \geq 500$; `minsize = N/5` if $250 \geq N < 500$; and `minsize = max(N/3, 50)` if $N < 250$ (Molenaar and Sijtsma 2000, p. 72).

Value: Returns an object of class `monotonicity.class` containing

results: A list with as many components as there are items. Each component itself is also a list containing the results of the check of manifest monotonicity. See Section 4 or (Molenaar and Sijtsma 2000, pp. 66–74) for more detailed information;

I.labels: the item labels,

Hi: the item scalability coefficients H_j (2); and

m: the number of answer categories.

Details: S3 methods are available so `summary` and `plot` can be used for objects of class `monotonicity.class`. Let `MC` be an object of class `monotonicity.class`.

`summary(MC)` returns a matrix with a summary of the results of the investigation of latent monotonicity (See Section 4 for an example).

`plot(MC, items = all)` returns a graph (See Section 4 for an example).

items: vector containing the numbers of the items for which the results are depicted graphically. By default the results for all items are depicted.

4. `check.pmatrix`

Description: computes the $P(++)$ matrix, the $P(--)$ matrix, and auxiliary information.

Usage: `check.pmatrix(X, minvi = .03)`

Required arguments: **X**: matrix or data frame of numeric data containing the responses of N respondents to J items. Missing values are not allowed.

Optional arguments: **minvi**: minimum size of a violation that is reported (Molenaar and Sijtsma 2000, p. 71).

Value: returns an object (of class `pmatrix.class`) containing

Ppp: the $P(++)$ matrix,

Pmm: the $P(--)$ matrix,

I.item: vector indicating to which items the rows and column the $P(++)$ matrix belong.

I.step: the labels of the item steps in order of popularity,

I.labels: the item labels,

Hi: the item scalability coefficients H_j (2), and

minvi: the value of minvi.

Details: The output is often numerous. S3 methods are available so **summary** and **plot** can be used for objects of class `pmatrix.class`. Let **PC** be an object of class `pmatrix.class`.

summary(PC) returns a list with two components. The first component contains a summary of the P(++) matrix per item and the second component contains a summary of the P(--) matrix per item. See Section 4 for an example.

plot(PC, items = all, pmatrix=both) returns a graphic display of the results of the investigation of nonintersection using method `pmatrix` (See Section 4 for an example).

items: vector containing the numbers of the items for which the results are depicted graphically. By default the results for all items are depicted.

pmatrix has values "ppp", "pmm", and "both"; If **pmatrix**="ppp", then the P(++) matrix is plotted, if **pmatrix**="pmm", then the P(--) matrix is plotted, if **pmatrix**="both", then both the P(++) matrix and P(--) matrix are plotted.

No c-axis labels are provided in the plot if the number of item steps is greater than 10.

5. `check.restscore`

Description: Returns the results from the investigation of nonintersection using method `restscore`.

Usage: `check.restscore(X, minvi = .03, minsize = default.minsize)`

Required arguments: **X**: matrix or data frame of numeric data containing the responses of N respondents to J items. Missing values are not allowed.

Optional arguments:

minvi: minimum size of a violation that is reported (Molenaar and Sijtsma 2000, p. 71).

minsize: minimum size of a rest score group. By default $\text{minsize} = N/10$ if $N \geq 500$; $\text{minsize} = N/5$ if $250 \geq N < 500$; and $\text{minsize} = \max(N/3, 50)$ if $N < 250$ (Molenaar and Sijtsma 2000, p. 72).

Value: returns an object (of class `restscore.class`) containing

results: A list with as many components as there are item pairs. Each component itself is also a list containing the results of the check of nonintersection using method `restscore`. See Section 4 or (Molenaar and Sijtsma 2000, pp. 74–78) for more detailed information;

I.labels: the item labels,

Hi: the item scalability coefficients H_j (2); and

m: the number of answer categories.

Details: the output is often numerous because **results** is a list of $J(J - 1)/2$ components. The procedure can be slow for large numbers of items. S3 methods are

available so `summary` and `plot` can be used for objects of class `restscore.class`. Let `RC` be an object of class `restscore.class`.

`summary(RC)` Returns a matrix with a summary of the results of the checks of nonintersection using method `restscore`.

`plot(RC, item.pairs = all)` returns a graphic display of the results of the investigation of nonintersection using method `restscore` (See Section 4 for an example).

`item.pairs`: vector containing the numbers of the item pairs for which the results are depicted graphically. For example, `item.pairs = 1` prints the results for items 1 and 2, `item.pairs = 2` prints the results for items 1 and 3, `item.pairs = J` prints the results for items 1 and J , and `item.pairs = J+1` prints the results for items 2 and 3. By default the results for all item pairs are depicted.

4. Example

4.1. The data

MSA was performed on Adjective Checklist (Gough and Heilbrun 1980) data, `acl`, which are available in `mokken`. The data (Vorst 1992) contain the scores of 433 students from the University of Amsterdam on 218 items from a Dutch version of the Adjective Checklist. Each item is an adjective with five ordered answer categories (0 = completely disagree, 1 = disagree, 2 = neither agree nor disagree, 3 = agree, 4 = completely agree). Form each adjective, the respondents must consider to what degree it describes their personality, and mark the answer category that fits best to this description.

Initially, the number of items in the Dutch version of the Adjective Checklist was larger than 218. Oosterveld (1989) suggested constructing 22 scales of 10 items each, and ignoring the remaining adjectives. Two items were not included in the administered test leaving two scales with 9 items. 77 of the 218 items that constitute the ten scales were negatively worded. The negatively worded items are indicated by an asterisk in the `dimnames` and their item scores were reversed. 296 out of the 94,394 responses (.03%) were missing; item scores were imputed for the missing scores using method Two-Way with Error (for details on this imputation method see Bernaards and Sijtsma, 2000) applied to each scale separately, yielding a completed 433×218 data matrix. Table 1 gives an overview of the 22 scales of the Adjective Checklist and their items.

4.2. Scalability coefficients

The first scale is denoted Commuality and contains the adjectives Reliable, Honest, Unscrupulous*, Deceitful*, Unintelligent*, Obnoxious*, Thankless*, Unfriendly*, Dependable, and Cruel*. The scalability coefficients of this scale are obtained by

```
R> acl.com <- acl[,1:10]
R> coefH(acl.com)
```

Scale	Items	Scale	Items
Communality	1-10	Change	111-119
Achievement	11-20	Succorance	120-129
Dominance	21-30	Abasement	130-139
Endurance	31-40	Deference	140-149
Order	41-50	Personal Adjustment	151-159
Intelligence	51-60	Ideal Self	160-169
Nurturance	61-70	Critical parent	170-179
Affiliation	71-80	Nurturant parent	180-189
Exhibition	81-90	Adult	190-199
Autonomy	91-100	Free Child	200-209
Aggression	101-110	Adapted Child	210-218

Table 1: An overview of the 22 scales of the Adjective Checklist and their items

The output shows that all H_{ij} s are positive satisfying the first criterion of a Mokken scale. Seven out of ten H_j s are less than the lower bound $c = .3$ which violates the second criterion of a Mokken scale; especially the item-scalability coefficient for Unintelligent* is low ($H_j = .11$). The scalability coefficient for the entire scale, H , equals $.26$ which is too low for the qualification “weak scale”.

4.3. Investigation of latent monotonicity

Investigating latent monotonicity in the scale Communality can be done using the following three commands.

```
R> monotonicity.com <- check.monotonicity(acl.com)
R> summary(monotonicity.com)
R> plot(monotonicity.com)
```

All computations are done in the function `check.monotonicity`; the most important component of the resulting object `monotonicity.com` is `results`, which is a list with as many components as there are items in the scale (i.e. 10), and each component itself is also a list with four components. These results are summarized by `summary` and visualized by `plot`.

For example, `monotonicity.com$results[[5]]` is a list containing information on Item 5:

1. `monotonicity.com$results[[5]][[1]]` is the label of the item: Unintelligent*.
2. `monotonicity.com$results[[5]][[2]]` is the following matrix (rounded):

Group	Lo	Hi	N	F0	F1	F2	F3	F4	Mean	P(X>=1)	P(X>=2)	P(X>=3)	P(X>=4)
1	12	25	89	0	3	20	28	38	3.13	1.00	0.97	0.74	0.43
2	26	28	93	0	1	18	41	33	3.14	1.00	0.99	0.80	0.35
3	29	31	113	1	1	10	44	57	3.37	0.99	0.98	0.89	0.50
4	32	36	138	0	2	9	43	84	3.51	1.00	0.99	0.92	0.61

For each rest score group (Group) it shows the minimum rest score (Lo), the maximum rest score (Hi), the size of the rest score group (N), the frequency distribution of the

item scores (F_0, \dots, F_4), the mean item score (**Mean**), and the proportion of respondents obtaining at least scores 1, 2, 3, and 4 ($P(X \geq 1), \dots, P(X \geq 4)$). It should be noted that the last column (first and second row) shows a violation of manifest monotonicity that is greater than 0.03.

3. `monotonicity.com$results[[5]][[3]]` is a summary of the violations of manifest monotonicity in item 5:

	#ac	#vi	#vi/#ac	maxvi	sum	sum/#ac	zmax	group	group	#zsig
P(X >=1)	6	0	0.00	0.00	0.00	0.00	0.00	0	0	0
P(X >=2)	6	0	0.00	0.00	0.00	0.00	0.00	0	0	0
P(X >=3)	6	0	0.00	0.00	0.00	0.00	0.00	0	0	0
P(X >=4)	6	1	0.17	0.07	0.07	0.01	0.85	1	2	0
Total	24	1	0.04	0.07	0.07	0.00	0.85	0	0	0

For each item step it gives the number of active pairs (**#ac**, if there are four rest score groups, there are $4 \times 3 \times \frac{1}{2}$ active pairs); the number of violations of manifest monotonicity (**#vi**, note that only violations greater than `minvi` are reported); the average number of violations of manifest monotonicity per active pair (**#vi/#ac**), the largest violation of manifest monotonicity (**maxvi**); the sum of violations of manifest monotonicity (**sum**); the average violation per active pair (**sum/#ac**); the two rest score groups that are involved in the largest violation of manifest monotonicity (**group**, equals zero if there were no violations); and the number of violations that are significantly greater than zero.

4. `monotonicity.com$results[[5]][[4]]` gives the values of `minsize` and `minvi` which were 0.03 and 86, respectively.

For more details see [Molenaar and Sijtsma \(2000, pp. 66–74\)](#).

The result of `summary(monotonicity.com)` is a matrix showing for each item a summary of the checks of manifest monotonicity.

	ItemH	#ac	#vi	#vi/#ac	maxvi	sum	sum/#ac	zmax	#zsig
reliable	0.30	24	0	0.00	0.00	0.00	0	0.00	0
honest	0.27	24	0	0.00	0.00	0.00	0	0.00	0
unscrupulous*	0.24	24	0	0.00	0.00	0.00	0	0.00	0
deceitful*	0.32	24	0	0.00	0.00	0.00	0	0.00	0
unintelligent*	0.12	24	1	0.04	0.07	0.07	0	0.85	0
obnoxious*	0.29	24	0	0.00	0.00	0.00	0	0.00	0
thankless*	0.25	24	0	0.00	0.00	0.00	0	0.00	0
unfriendly*	0.31	24	0	0.00	0.00	0.00	0	0.00	0
dependable	0.30	24	0	0.00	0.00	0.00	0	0.00	0
cruel*	0.25	24	0	0.00	0.00	0.00	0	0.00	0

It shows that there is only one (nonsignificant) violation of manifest monotonicity in the scale. This violation occurred for the item `Unintelligent*`.

The results are visualized using the `plot` function. The page that pertains to the item `Unintelligent*` is shown in [Figure 1](#) (can be obtained as a separate file by

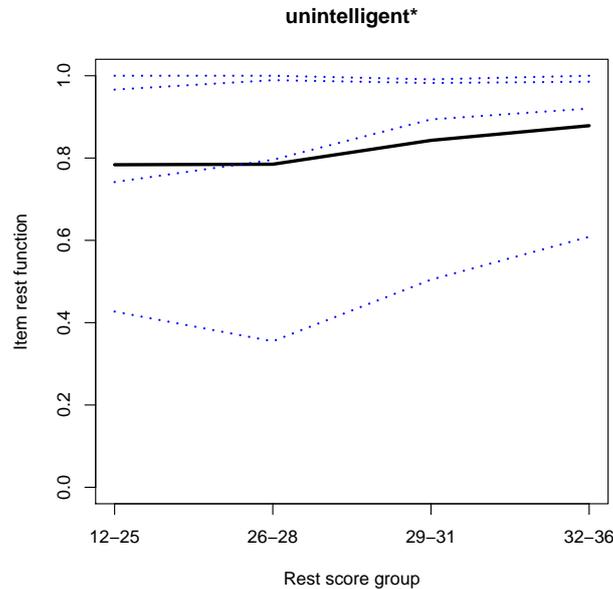


Figure 1: Visualization of the check of manifest monotonicity for item 5 (Unintelligent*)

`plot(acl.com, items=5)`.) Figure 1 shows the estimated item step response functions $P(X_5 \geq x | R_{-5})$ for $x = 1, \dots, 4$ (dashed lines) and the estimated mean response function $\frac{1}{m}E(X_5 | R_{-5})$ (solid line). The violation of manifest monotonicity is shown by the decrease between $P(X_5 \geq 4 | R_{-5} \in \{12, 13, \dots, 25\})$ and $P(X_5 \geq 4 | R_{-5} \in \{26, 27, 28\})$ (lowest dashed line).

4.4. Investigation of nonintersection

Investigating nonintersection in the scale Commuality can be done using methods `pmatrix` and `restscore`.

Method pmatrix

Method `pmatrix` can be applied using the following three commands.

```
R> pmatrix.com <- check.pmatrix(acl.com)
R> summary(pmatrix.com)
R> plot(pmatrix.com)
```

Function `check.pmatrix` computes the $P(++)$ matrix (`pmatrix.com$Ppp`) and the $P(--)$ matrix (`pmatrix.com$Pmm`). The size of the two matrices ($Jm \times Jm = 40 \times 40$) is too large to be useful for inspecting violations of nonintersection. Function `summary` reduces the quantity of the output.

Function `summary` produces summaries of both the $P(++)$ and the $P(--)$ matrix. The summary of the $P(++)$ matrix is

	ItemH	#ac	#vi	#vi/#ac	maxvi	sum	sum/#ac
reliable	0.30	144	6	0.001	0.08	0.39	0.003
honest	0.27	144	11	0.000	0.05	0.43	0.003
unscrupulous*	0.24	144	0	0.000	0.00	0.00	0.000
deceitful*	0.32	144	0	0.000	0.00	0.00	0.000
unintelligent*	0.12	144	0	0.000	0.00	0.00	0.000
obnoxious*	0.29	144	0	0.000	0.00	0.00	0.000
thankless*	0.25	144	1	0.000	0.05	0.05	0.000
unfriendly*	0.31	144	1	0.000	0.04	0.04	0.000
dependable	0.30	144	10	0.000	0.05	0.35	0.002
cruel*	0.25	144	1	0.000	0.03	0.03	0.000

For each item the output shows: the scalability coefficient H_j (ItemH, Section 2.2), the number of active pairs (#ac), the number of violations greater than minvi (#vi), the average number of violations per active pair (#vi/#ac), the maximum violation (maxvi), the sum of the violations greater than minvi (sum), and sum/#ac. The output shows that items Reliable, Honest, Thankless*, Unfriendly*, and Dependable* have some violations greater than 0.03. The violations are relatively small. A similar matrix is provided to summarize the results of the P(--) matrix.

Function `plot` produces two graphs for each item j . In one graph the lines represent the rows of the P(++) matrix pertaining to item j which should be nondecreasing if nonintersection holds, In the other graph the lines represent the rows of the P(--) matrix pertaining to item j which should be nonincreasing if nonintersection holds.

Figure 2 displays the four rows in the P(++) matrix pertaining to item 5 (Unintelligent*). On the horizontal axis, the $(10 - 1)4 = 36$ item steps of the remaining 9 items are displayed in ascending order of popularity; i.e. the tick on the extreme left of the horizontal axis is the least popular item step $X_2 \geq 4$, and the tick on the extreme right of the horizontal axis is the most popular item step $X_1 \geq 1$. Let I_1, \dots, I_{36} denote the 36 ordered item steps. The upper line in the graph connects $P(X_5 \geq 1, I_1), \dots, P(X_5 \geq 1, I_{36})$, the next line in the graph displays $P(X_5 \geq 2, I_1), \dots, P(X_5 \geq 2, I_{36})$, etc. A nonincreasing line indicates a violation of nonintersection. Figure 2 shows that there are a few very small violations (less than 0.03) in the rows of the P(++) matrix pertaining to item 5 (Unintelligent*). A similar graph is provided for the P(--) matrix, where a nondecreasing line indicates a violation of nonintersection.

Method `restscore`

Method `restscore` can be applied using the following three commands.

```
R> restscore.com <- check.restscore(acl.com)
R> summary(restscore.com)
R> plot(restscore.com)
```

The most important component of `restscore.com` is `results`, which is a list with as many components as there are item pairs, i.e. $10 \times 9 \times .5 = 45$; each component itself is also a list. For example, `restscore.com$results[[31]]` pertaining to items 5 and 6 is the following list:

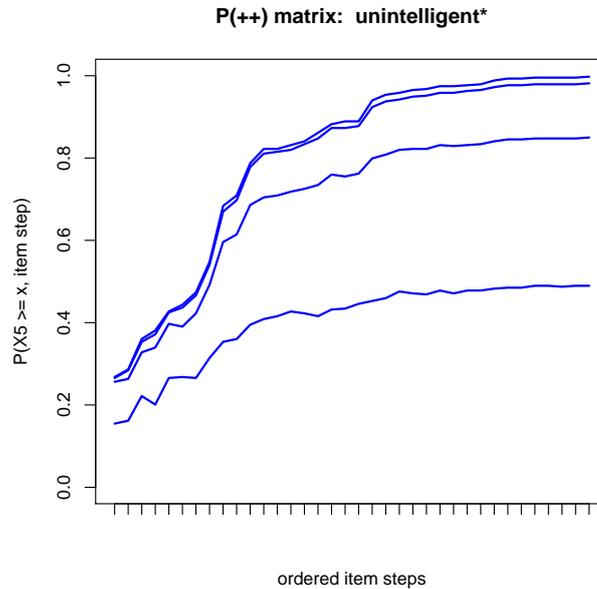


Figure 2: Visualization of the check of nonintersection using method `pmatrix` for item 5 (Unintelligent*)

`restscore.com$results[[31]][[1]]` is the label of both items: `Unintelligent*` and `Obnoxious*`.

`restscore.com$results[[31]][[2]]` is the following matrix:

Group	Lo	Hi	N	E(X5)	E(X6)	P(X5>=1)	P(X5>=2)	P(X5>=3)	P(X5>=4)	P(X6>=1)
1	9	22	88	3.16	2.64	1.00	0.97	0.76	0.43	1.00
2	23	25	98	3.11	3.12	1.00	0.99	0.79	0.34	1.00
3	26	27	88	3.38	3.33	0.99	0.99	0.88	0.52	1.00
4	28	32	159	3.50	3.62	1.00	0.98	0.92	0.60	0.99
						P(X6>=2)	P(X6>=3)	P(X6>=4)		
						0.92	0.58	0.14		
						0.99	0.88	0.26		
						1.00	0.89	0.44		
						0.99	0.94	0.69		

It shows for each of the four restscore groups (Group) the minimum rest score (Lo); the maximum rest score (Hi); the size of the rest group (N); the mean scores of item 5 ($E(X5)$) and item 6 ($E(X6)$), and the probabilities of obtaining at least a score x on items 5 and 6, for $x = 1, \dots, 4$ ($P(X5 \geq 1), \dots, P(X6 \geq 4)$). Note that there are two violations greater than `minvi`: $P(X_5 \geq 3 | R_{-5,-6})$ and $P(X_6 \geq 3 | R_{-5,-6})$, and $P(X_5 \geq 4 | R_{-5,-6})$ and $P(X_6 \geq 4 | R_{-5,-6})$ intersect.

`restscore.com$results[[31]][[3]]` shows the violations of restscore for each pair of estimated item step response functions:

		#ac	#vi	#vi/#ac	maxvi	sum	sum/#ac	zmax	#zsig
P(X5>=1) P(X6>=1)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=1) P(X6>=2)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=1) P(X6>=3)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=1) P(X6>=4)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=2) P(X6>=1)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=2) P(X6>=2)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=2) P(X6>=3)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=2) P(X6>=4)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=3) P(X6>=1)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=3) P(X6>=2)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=3) P(X6>=3)		3	1	0.33	0.09	0.09	0.03	1.61	0
P(X5>=3) P(X6>=4)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=4) P(X6>=1)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=4) P(X6>=2)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=4) P(X6>=3)		3	0	0.00	0.00	0.00	0.00	0.00	0
P(X5>=4) P(X6>=4)		3	1	0.33	0.09	0.09	0.03	1.83	1
E(X5) E(X6)		3	1	0.33	0.12	0.12	0.04	NA	NA
Total		48	2	0.04	0.09	0.19	0.00	1.83	1

As was noted earlier, two estimated item step response functions showed violations greater than \min_{vi} ; only the violation by $P(X_5 \geq 4|R_{-5,-6})$ and $P(X_6 \geq 4|R_{-5,-6})$ is significant. $E(X_5|R_{-5,-6})$ and $E(X_6|R_{-5,-6})$ also intersect, but no tests are provided for this violation.

`summary(restscore.com)` gives the following output.

	ItemH	#ac	#vi	#vi/#ac	maxvi	sum	sum/#ac	zmax	#zsig
reliable	0.30	432	7	0.02	0.09	0.31	0	1.43	0
honest	0.27	432	5	0.01	0.07	0.25	0	1.25	0
unscrupulous*	0.24	416	6	0.01	0.11	0.42	0	1.46	0
deceitful*	0.32	400	8	0.02	0.09	0.40	0	1.22	0
unintelligent*	0.12	416	14	0.03	0.11	0.86	0	1.99	2
obnoxious*	0.29	432	7	0.02	0.11	0.52	0	1.83	1
thankless*	0.25	432	7	0.02	0.08	0.38	0	1.14	0
unfriendly*	0.31	432	8	0.02	0.11	0.48	0	1.99	1
dependable	0.30	432	9	0.02	0.09	0.48	0	1.22	0
cruel*	0.25	432	3	0.01	0.04	0.12	0	0.67	0

For each item it shows scalability coefficient H_j (ItemH), the total number of active pairs (#ac), the total number of violations (#vi), the average number of violations per active pair (#vi/#ac), the maximum violation (maxvi), the sum of all violations (sum), the average violation per active pair (sum/#ac), the maximum test statistic (zmax), and the number of significant violations (#zsig). For example, item 5 (Unintelligent*) has most violations (14), two were significant. These significant violations occurred in item pairs (5,6) and (5,8). With respect to nonintersection, item 5 would be a good candidate to be removed from the item set.

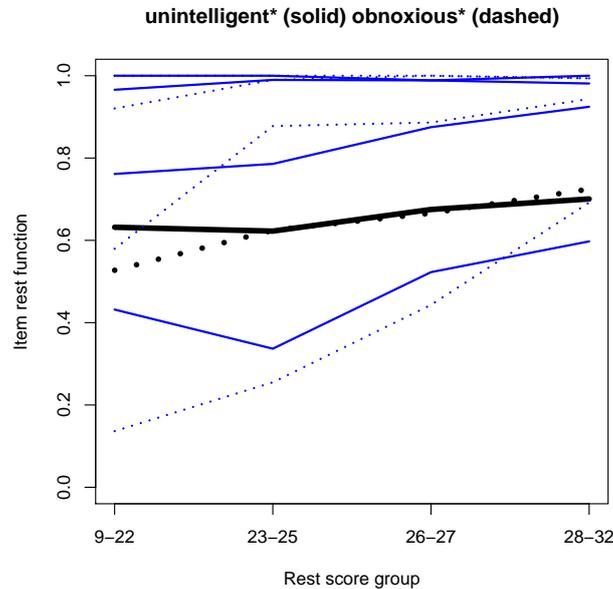


Figure 3: Visualization of the check of nonintersection using method restscore for items 4 (Deceitful*) and 5 (Unintelligent*)

Function `plot(restscore.com)` visualizes the results of investigating nonintersection using method restscore. As an example, the results for items 5 (Unintelligent*) and 6 (Obnoxious*) are depicted in Figure 3. The thick solid line represents $E(X_5|R_{-5,-6})$, and the thick dashed line represents $E(X_6|R_{-5,-6})$ (both lines scaled by a factor $\frac{1}{4}$). The violation found in `restscore.com$results[[31]][[3]]` is also shown in the graph because the thick solid line and the thick dashed line intersect. The thin solid lines represent the estimated item step response functions of item 5 (Unintelligent*) and the thin dashed lines represent the estimated item step response functions of item 6 (Obnoxious*). Figure 3 clearly shows the two violations found in `restscore.com$results[[31]][[3]]` as intersections of a solid and a dashed line.

4.5. Automated item selection algorithm

The automated item selection algorithm can be applied to scale Commuality as follows:

```
R> scale.com <- search.normal(ac1.com)
```

and results in the following vector `scale.com` (`t(scale.com)` is displayed).

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    0    1    0    2    2    2    1    2
```

hence, using the default lower bound $c = .3$ results in two Mokken scales. One Mokken scale contains the adjectives Dependable, Reliable, Honest, and Deceitful* (indicated by a 1 in `scale.com`), and the other Mokken scale contains the adjectives Cruel*, Unfriendly*,

Obnoxious*, and Thankless* (indicated by a 2 in `scale.com`). The items Unscrupulous* and Unintelligent* are unscalable (indicated by a 0 in `scale.com`).

Computing the scalability coefficients for the first scale can be done as follows

```
R> coefH(ac1.com[,scale.com==1])
```

Similarly, investigating latent monotonicity for the first Mokken scale is done by

```
R> monotonicity.com.1 <- check.monotonicity(ac1.com[,scale.com==1])
R> summary(monotonicity.com.1)
R> plot(monotonicity.com.1)
```

Investigating latent monotonicity for the second scale and checking nonintersection are done in a similar way.

Acknowledgments

Thanks are due to Harrie C. M. Vorst for providing the ACL data set, Anton L. M. de Vries for providing the item labels in English, Rudy Ligtoet for testing **mokken**, and Patrick Mair and an anonymous reviewer for providing very useful comments on the package and the paper, respectively.

References

- Bernaards CA, Sijtsma K (2000). "Influence of Imputation and EM Methods on Factor Analysis When Item Nonresponse in Questionnaire Data Is Nonignorable." *Multivariate Behavioral Research*, **35**, 321–364.
- Birnbaum A (1968). "Some Latent Trait Models." In FM Lord, MR Novick (eds.), "Statistical Theories of Mental Test Scores," pp. 397–424. Addison-Wesley, Reading, MA.
- Gough HG, Heilbrun AB (1980). *The Adjective Check List, Manual 1980 Edition*. Consulting Psychologists Press, Palo Alto, CA.
- Grayson DA (1988). "Two-Group Classification in Latent Trait Theory: Scores With Monotone Likelihood Ratio." *Psychometrika*, **53**, 383–392.
- Hemker BT, Sijtsma K, Molenaar IW, Junker BW (1996). "Polytomous IRT Models and Monotone Likelihood Ratio of the Total Score." *Psychometrika*, **61**, 679–693.
- Hemker BT, Sijtsma K, Molenaar IW, Junker BW (1997). "Stochastic Ordering Using the Latent Trait and the Sum Score in Polytomous IRT Models." *Psychometrika*, **62**, 331–347.
- Huynh H (1994). "A New Proof for Monotone Likelihood Ratio for the Sum of Independent Bernoulli Random Variables." *Psychometrika*, **59**, 77–79.
- Junker BW, Sijtsma K (2000). "Latent and Manifest Monotonicity." *Applied Psychological Measurement*, **24**, 65–81.

- Meijer RR, Baneke JJ (2004). “Analyzing Psychopathology Items: A Case for Nonparametric Item Response Theory Modeling.” *Psychological Methods*, **9**, 354–368.
- Mokken RJ (1971). *A Theory and Procedure of Scale Analysis*. De Gruyter, Berlin, Germany.
- Mokken RJ, Lewis C (1982). “A Nonparametric Approach to the Analysis of Dichotomous Responses.” *Applied Psychological Measurement*, **6**, 417–430.
- Molenaar IW (1991). “A Weighted Loevinger H-Coefficient Extending Mokken Scaling to Multicategory Items.” *Kwantitatieve Methoden*, **12**(37), 97–117.
- Molenaar IW (1997). “Nonparametric Models for Polytomous Responses.” In WJ van der Linden, RK Hambleton (eds.), “Handbook of Modern Item Response Theory,” pp. 369–380. Springer-Verlag, New York.
- Molenaar IW, Sijtsma K (2000). *User’s Manual MSP5 for Windows*. IEC ProGAMMA, Groningen, The Netherlands.
- Oosterveld P (1989). “ACL: verkorting en unidimensionaliteit [ACL: Reduction and Unidimensionality].” Unpublished manuscript, University of Amsterdam.
- Rasch G (1960). *Probabilistic Models for Some Intelligence and Attainment Tests*. Nielsen and Lydiche, Copenhagen, Denmark.
- R Development Core Team (2007). *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/>.
- Samejima F (1969). “Estimation of Latent Ability Using a Response Pattern of Graded Scores.” *Psychometrika Monograph*, **17**.
- Sijtsma K, Hemker BT (1998). “Nonparametric Polytomous IRT Models for Invariant Item Ordering, With Results for Parametric Models.” *Psychometrika*, **63**, 183–200.
- Sijtsma K, Junker BW (1996). “A Survey of Theory and Methods of Invariant Item Ordering.” *British Journal of Mathematical and Statistical Psychology*, **49**, 79–105.
- Sijtsma K, Molenaar IW (2002). *Introduction to Nonparametric Item Response Theory*. Sage, Thousand Oaks, CA.
- van Abswoude AAH, van der Ark LA, Sijtsma K (2004). “A Comparative Study of Test Data Ddimensionality Assessment Procedures Under Nonparametric IRT Models.” *Applied Psychological Measurement*, **28**, 3–24.
- van der Ark LA (2001). “Relationships and Properties of Polytomous Item Response Theory Models.” *Applied Psychological Measurement*, **25**, 273–282.
- van der Ark LA (2005). “Stochastic Ordering of the Latent Trait by the Sum Score Under Various Polytomous IRT Models.” *Psychometrika*, **70**, 283–304.
- Vorst HCM (1992). [Responses to the Adjective Checklist] Unpublished raw data.

Weesie J (1999). “**MOKKEN**: Stata module: Mokken scale analysis.” *Software Components RePEc:boc:bocode:sjw31*, RePEc EconPapers. URL <http://econpapers.repec.org/software/bocbocode/sjw31.htm>.

Affiliation:

L. Andries van der Ark
Department of methodology and Statistics
Tilburg University
P. O. Box 90153, 5000 LE, Tilburg, The Netherlands
E-mail: a.vdark@uvt.nl
URL: <http://spitswww.uvt.nl/~avdrark/>