

Package ‘robustDIF’

April 21, 2026

Type Package

Title Differential Item Functioning Using Robust Scaling

Version 0.2.0

Description Provides tools for testing differential item functioning (DIF) and differential test functioning (DTF) in two-group item response theory models. The package estimates robust scaling parameters via iteratively reweighted least squares with Tukey's bisquare loss, and supports Wald-type tests of item-level and test-level differences from robust scaling parameters. Inputs can be supplied directly from model parameter/covariance objects or extracted from fitted 'mirt' and 'lavaan' models. Methods are described in Halpin (2022) <[doi:10.48550/arXiv.2207.04598](https://doi.org/10.48550/arXiv.2207.04598)>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R(>= 3.5.0), Matrix

Imports mirt, lavaan

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Peter Halpin [aut, cre],
Kyle Nickodem [ctb],
James Eagle [ctb]

Maintainer Peter Halpin <peter.halpin@unc.edu>

Repository CRAN

Date/Publication 2026-04-21 18:42:07 UTC

Contents

.onLoad	2
a_fun	3

bsq_weight	3
delta_test	4
dif_test	4
d_fun	5
format_pars	6
get_lavaan_pars	6
get_mirt_pars	7
get_model_parms	7
get_starts	8
grad_a	9
grad_d	9
lts	10
psi	11
psi_prime	11
rdif	12
rdif.eg	13
rho	13
rho_grid	14
vcov_y	15
y_fun	15

Index	17
--------------	-----------

.onLoad	<i>Registers S3 methods at load time: - print for class "rdif" - plot for class "rdif" - summary for class "rdif"</i>
---------	---

Description

Registers S3 methods at load time: - print for class "rdif" - plot for class "rdif" - summary for class "rdif"

Usage

```
.onLoad(libname, pkgname)
```

Arguments

libname	Character string with the path to the package library.
pkgname	Character string with the package name.

Value

No return value, called for side effects when the package loads.

a_fun	<i>The R-DIF scaling function for item slopes.</i>
-------	--

Description

Computes the scaling function a_2/a_1 for item slopes (a) in groups $g = \{1, 2\}$

Usage

```
a_fun(mle, log = FALSE)
```

Arguments

mle	the output of get_model_parms
log	logical: return of $\log(a_2/a_1)$?

Value

The vector of scaling function values.

bsq_weight	<i>The bi-square weight function.</i>
------------	---------------------------------------

Description

If $\text{abs}(u) > k$, $\text{psi}(u) = 0$. Else, $\text{psi}(u) = (1 - (u/k)^2)^2$.

Usage

```
bsq_weight(u, k = 1.96)
```

Arguments

u	Can be a single value, vector, or matrix.
k	The tuning parameter. Can be a scalar or the same dimension as u.

Value

The bi-square psi-prime function.

delta_test	<i>Wald test of differential test functioning.</i>
------------	--

Description

A Wald test of the difference between the unweighted mean of the `y_fun` and robust scaling parameter from `rdif`. Called internally by `rdif`

Usage

```
delta_test(object, theta = NULL, k = NULL, fun = "d_fun3")
```

Arguments

<code>object</code>	either the output of <code>get_model_parms</code> or an <code>rdif</code> object from <code>rdif</code> .
<code>theta</code>	the estimated scaling parameter from <code>rdif</code> . Not needed when <code>object</code> is an <code>rdif</code> object.
<code>k</code>	the tuning parameter from <code>rdif</code> . Not needed when <code>object</code> is an <code>rdif</code> object.
<code>fun</code>	one of <code>c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3")</code> . See description for details.

Value

A `data.frame` that contains the output of the test.

Examples

```
#
mod <- rdif(mle = rdif.eg)
delta_test(object = rdif.eg, theta = mod$est, k = mod$k)
delta_test(mod)
```

dif_test	<i>Wald tests of differential item functioning (DIF).</i>
----------	---

Description

A Wald test of DIF on each item. Called internally by `rdif`

Usage

```
dif_test(object, theta = NULL, fun = "d_fun3")
```

Arguments

object	either the output of <code>get_model_parms</code> or an <code>rdif</code> object from <code>rdif</code> .
theta	the estimated scaling parameter from <code>rdif</code>
fun	one of <code>c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3")</code> . See description for details.

Value

A data.frame whose rows containing the results of the test for each item parameter.

Examples

```
mod <- rdif(mle = rdif.eg)
dif_test(object = rdif.eg, theta = mod$est)
dif_test(mod)
```

d_fun	<i>The R-DIF scaling functions for item intercepts / thresholds.</i>
-------	--

Description

Computes the scaling function to be used from the item thresholds (d) in groups = {1, 2}. The options are:

- type = 1: computes $d.fun1 = (d2 - d1)/a1$
- type = 2: computes $d.fun2 = (d2 - d1)/a2$
- type = 3: computes $d.fun3 = (d2 - d1)/\sqrt{\{(a1^2 + a2^2)/2\}}$

Usage

```
d_fun(mle, type = 3)
```

Arguments

mle	the output of <code>get_model_parms</code>
type	a number in 1:3 indicating which version of delta to compute. See description for details.

Details

Computes $(d2 - d1)/a$ for each threshold (d) of each item in groups $g = \{1, 2\}$. The parameter 'a' depends on the value of type:

Value

The vector of scaling function values.

format_pars	<i>Helper function used to format parameters estimates</i>
-------------	--

Description

Helper function used to format parameters estimates

Usage

```
format_pars(pars, names.vec, type)
```

Arguments

pars	numeric vector of item parameter estimates
names.vec	character vector item names
type	character; are pars from "lavaan" or "mplus"?

Value

data.frame of item parameter estimates

See Also

[robustDIF::get_lavaan_params()], [robustDIF::get_mplus_params()]

get_lavaan_pars	<i>Extract item parameter estimates and their covariance matrix from lavaan.</i>
-----------------	--

Description

Extract item parameter estimates and their covariance matrix from lavaan.

Usage

```
get_lavaan_pars(lavaan.object)
```

Arguments

lavaan.object	an object of class lavaan. Expected to be a 1-factor model estimated with <code>std.lv = TRUE</code> .
---------------	--

Value

A three-element list:

- vector of parameter names taking the form "item.parameter"
- list (one element per group) of vectors of item parameter estimates
- list (one element per group) of covariance matrices of item parameter estimates

get_mirt_pars	<i>Extract item parameter estimates and their covariance matrix from mirt.</i>
---------------	--

Description

Extract item parameter estimates and their covariance matrix from [mirt](#).

Usage

```
get_mirt_pars(mirt.object)
```

Arguments

mirt.object a [mirt](#) object of class SingleGroupClass or MultipleGroupClass. Expected to be a 1-factor model with SE = TRUE and itemtype of any combination of "2PL", "graded", or "gpcm".

Value

A three-element list:

- vector of parameter names taking the form "item.parameter"
- list (one element per group) of vectors of item parameter estimates
- list (one element per group) of covariance matrices of item parameter estimates

get_model_parms	<i>Extract and format item parameter estimates and their covariance matrix</i>
-----------------	--

Description

Takes a 1-factor model fit or list of 1-factor model fits from [mirt](#) or [cfa](#) and formats the item parameter estimates and their covariance matrix for use in other robustDIF functions.

Usage

```
get_model_parms(object)
```

Arguments

`object` model fit from a multigroup analysis or list of model fits for each group for a 1-factor model. See Details.

Details

The function takes a fitted 1-factor multigroup model or list of fitted 1-factor single group models. The factor must be standardized (i.e., variance = 1) and the covariance matrix be asymptotically correct. Currently, the function accepts:

- a `mirt` object of class `SingleGroupClass` or `MultipleGroupClass` with `SE = TRUE` (to return covariance matrix) and `itemtype` of any combination of "2PL", "graded", or "gpcm".
- a `lavaan` object estimated from `cfa` with `std.lv = TRUE`.

It is possible to use fits from other software with robustDIF functions, but the parameter estimates and their covariance matrices must be formatted identically to what is returned by `get_model_parms`. For details, see the documentation for the example dataset [rdif.eg](#).

Value

A three-element list:

- `par.names`: list with `internal` and `original` parameter names.
- `est`: list (one element per group) of data frames containing item parameters by row (`a1`, `d1`, `d2`, ...).
- `var.cov`: list (one element per group) of covariance matrices for the corresponding parameter vectors.

See Also

[rdif.eg](#)

`get_starts`

Compute starting values for [rdif](#).

Description

Compute starting values for [rdif](#).

Usage

```
get_starts(mle, fun = "d_fun3", alpha = 0.05)
```

Arguments

`mle` the output of [get_model_parms](#)

`fun` one of `c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3")`. See description for details.

`alpha` the desired false positive rate for flagging items with DIF.

Value

A vector containing the median of `y_fun`, the least trimmed squares estimate of location for `y_fun` with 50-percent trim rate, and the minimum of `rho_grid`.

grad_a	<i>The gradient matrix of <code>a_fun</code>.</i>
--------	---

Description

The gradient is taken with respect to the item parameters and organized to be conformable with `Matrix::bdiag(mle$var.cov)`. When evaluating the gradient under the null hypothesis of no DIF, the optional argument `theta` can be provided. It replaces the item-specific values of `a_fun` in the gradient computation.

Usage

```
grad_a(mle, theta = NULL, log = FALSE)
```

Arguments

<code>mle</code>	the output of <code>get_model_parms</code>
<code>theta</code>	(optional) the scaling parameter. Replaces item-specific values of alpha if provided.
<code>log</code>	logical: return of $\log(a_2/a_1)$?

Value

A matrix in which the columns are the gradient vectors of `a_fun`, for each item.

See Also

[a_fun](#)

grad_d	<i>The gradient matrix of <code>d_fun</code>.</i>
--------	---

Description

The gradient is taken with respect to the item parameters and organized to be conformable with `Matrix::bdiag(mle$var.cov)`. When evaluating the gradient under the null hypothesis of no DIF, the optional argument `theta` can be provided. It replaces the item-specific values of `d_fun` in the gradient computation.

Usage

```
grad_d(mle, theta = NULL, type = 3)
```

Arguments

mle	the output of get_model_parms
theta	(optional) the scaling parameter. Replaces item-specific values of d_fun if provided.
type	a number in 1:3 indicating which version of delta to compute. See description for details.

Value

A matrix in which the columns are the gradient vectors of [d_fun](#), for each item and threshold.

See Also

[d_fun](#)

lts

The least trimmed squares (LTS) estimate of location

Description

The least trimmed squares (LTS) estimate of location

Usage

```
lts(y, p = 0.5)
```

Arguments

y	a vector of data points.
p	the proportion of data points to trim.

Value

The LTS estimate of location of y.

See Also

[get_starts](#)

psi	<i>The bi-square psi function.</i>
-----	------------------------------------

Description

If $\text{abs}(u) > k$, $\text{psi}(u) = 0$. Else, $\text{psi}(u) = u(1 - (u/k)^2)^2$.

Usage

`psi(u, k = 1.96)`

Arguments

u	Can be a single value, vector, or matrix.
k	The tuning parameter. Can be a scalar or the same dimension as u.

Value

The bi-square psi function.

psi_prime	<i>The derivative of the bi-square psi function.</i>
-----------	--

Description

If $\text{abs}(u) > k$, $\text{psi_prime}(u) = 0$. Else, $\text{psi_prime}(u) = (1 - (u/k)^2)^2 - (2u/k)^2 (1 - (u/k)^2)$.

Usage

`psi_prime(u, k = 1.96)`

Arguments

u	Can be a single value, vector, or matrix.
k	The tuning parameter. Can be a scalar or the same dimension as u.

Value

The bi-square psi-prime function.

 rdif

Estimate IRT scale parameters using the robust DIF procedure.

Description

Estimation can be performed using iteratively re-weighted least squares (IRLS) or Newton-Raphson (NR). Currently, only IRLS is implemented. If `starting.value = "all"`, three starting values are computed: the median of `y_fun`, the least trimmed squares estimate of location for `y_fun` with 50-percent trim rate, and the minimum of `rho_grid`. The estimate is computed from each starting value, and the solution with the lowest value of the bi-square objective function is returned. If there are multiple solutions, they are stored in `other.solutions`.

Usage

```
rdif(
  mle,
  fun = "d_fun3",
  alpha = 0.05,
  starting.value = "all",
  tol = 1e-07,
  maxit = 100,
  method = "irls"
)
```

Arguments

<code>mle</code>	the output of <code>get_model_parms</code>
<code>fun</code>	one of <code>c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3")</code> . See description for details.
<code>alpha</code>	the desired false positive rate for flagging items with DIF.
<code>starting.value</code>	one of <code>c("med", "lts", "min_rho", "all")</code> or a numerical value to be used as the starting value. See description for details.
<code>tol</code>	convergence criterion for comparing subsequent values of estimate
<code>maxit</code>	maximum number of iterations
<code>method</code>	one of <code>c("irls", "newton")</code> . Currently, only IRLS is implemented.

Details

Implements M-estimation of an IRT scale parameter using the bi-square loss function. Also returns the bi-square weights for each item.

Value

An `rdif` object.

Examples

```
# Item intercepts, using the built-in example dataset "rdif.eg"
rdif(mle = rdif.eg, fun = "d_fun3")

# Item slopes
rdif(mle = rdif.eg, fun = "a_fun1")
```

rdif.eg

*Example data set for R-DIF functions.***Description**

A named list containing the maximum likelihood estimates and their estimated covariance matrix, for the 2PL IRT model fitted to 5 items in two independent groups. The first item has additive bias of .5 applied to the intercept only. The groups have a mean difference of .5 standard deviations on the latent trait. The variances of the latent trait are equal in each group.

Usage

```
rdif.eg
```

Format

A named list with 4 components:

par0 A data.frame or named list with par0\$a containing the item slopes and par0\$d containing the item intercepts, for the reference group.

par1 The item parameter estimates of the comparison groups. See par0 for formatting.

vcov0 The covariance matrix of par0, formatted as either a data.frame or matrix. The parameters should be organized by item, with the slope parameter coming first and the intercept parameter coming second (e.g., a.item1, d.item1, a.item2, d.item2, ...).

vcov1 The covariance matrix of par1. See vcov0 for formatting.

rho

*The bi-square rho function.***Description**

If $\text{abs}(u) > k$, $\rho(u) = 1$. Else, $\rho(u) = 1 - (1 - (u/k)^2)^3$.

Usage

```
rho(u, k = 1.96)
```

Arguments

- u Can be a single value, vector, or matrix.
- k The tuning parameter. Can be a scalar or the same dimension as u.

Value

The bi-square rho function.

rho_grid	<i>Compute a grid of bi-square Rho values</i>
----------	---

Description

Computes the objective function of the bi-square minimization problem in a location parameter, theta. The theta values are obtained internally by a grid search over the range of `y_fun`. Used for starting values and graphically diagnosing local solutions.

Usage

```
rho_grid(mle, fun = "d_fun3", alpha = 0.05, grid.width = 0.01)
```

Arguments

- mle the output of `get_model_parms`
- fun one of `c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3")`. See description for details.
- alpha the desired false positive rate for flagging items with DIF.
- grid.width the width of grid points.

Value

A named list with theta values and the corresponding Rho values.

vcov_y	<i>The covariance matrix of IRT scaling functions.</i>
--------	--

Description

When evaluating the covariance matrix under the null hypothesis of no DIF, the optional argument `theta` can be provided. It replaces the item-specific scaling functions in the gradient computation. Type should be the same as used in `y_fun`.

Usage

```
vcov_y(mle, theta = NULL, fun = "d_fun3")
```

Arguments

<code>mle</code>	the output of <code>get_model_parms</code>
<code>theta</code>	(optional) the scaling parameter. Replaces item-specific scaling functions if provided.
<code>fun</code>	one of <code>c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3")</code> . See description for details.

Value

The covariance matrix of `y_fun`.

See Also

[y_fun](#)

<code>y_fun</code>	<i>R-DIF scaling functions for item intercepts/thresholds and slopes.</i>
--------------------	---

Description

Computes the a scaling function using the item thresholds (`d`) and slopes (`a`) in groups = {0, 1}. The options are:

- `"a_fun1"`: computes a_2/a_1
- `"a_fun2"`: computes $\log(a_2/a_1)$
- `"d_fun1"`: computes $(d_2 - d_1)/a_1$ for each threshold
- `"d_fun2"`: computes $(d_2 - d_1)/a_2$ for each threshold
- `"d_fun3"`: computes $(d_2 - d_1)/\sqrt{(a_1^2 + a_2^2)/2}$ for each threshold

Usage

```
y_fun(mle, fun = "d_fun3")
```

Arguments

mle	the output of get_model_parms
fun	one of c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3"). See description for details.

Details

Computes the scaling function specified by fun, for each item.

Value

A vector of scaling function values.

Index

* datasets

rdif.eg, 13
.onLoad, 2

a_fun, 3, 9

bsq_weight, 3

cfa, 7, 8

d_fun, 5, 9, 10

delta_test, 4

dif_test, 4

format_pars, 6

get_lavaan_pars, 6

get_mirt_pars, 7

get_model_parms, 3–5, 7, 8–10, 12, 14–16

get_starts, 8, 10

grad_a, 9

grad_d, 9

lts, 10

mirt, 7, 8

psi, 11

psi_prime, 11

rdif, 4, 5, 8, 12

rdif.eg, 8, 13

rho, 13

rho_grid, 9, 12, 14

vcov_y, 15

y_fun, 4, 9, 12, 14, 15, 15