Package: robustDIF (via r-universe)

May 28, 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>.

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

URL https://peterhalpin.github.io/robustDIF/

Config/pak/sysreqs cmake make libuv1-dev libx11-dev

Repository https://peterhalpin.r-universe.dev

Date/Publication 2026-04-23 17:05:48 UTC

RemoteUrl https://github.com/peterhalpin/robustdif

RemoteRef HEAD

RemoteSha fdabf378379e4bf2dc658ca13d4cc50370e90205

Contents

`.onLoad` *Registers S3 methods at load time: - print for class "rdif" - plot for class "rdif" - summary for class "rdif"*

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 *The R-DIF scaling function for item slopes.*

Description

Computes the scaling function `a2/a1` 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(a2/a1)?

Value

The vector of scaling function values.

bsq_weight *The bi-square weight function.*

Description

If `abs(u) > k` , `psi(u) = 0`. Else, `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.

d_fun *The R-DIF scaling functions for item intercepts / thresholds.*

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 `get_model_parms`
`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.

delta_test *Wald test of differential test functioning.*

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

object	either the output of get_model_parms or an rdif object from rdif.
theta	the estimated scaling parameter from rdif. Not needed when object is an rdif object.
k	the tuning parameter from rdif. Not needed when object is an rdif object.
fun	one of c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3"). 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 *Wald tests of differential item functioning (DIF).*

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 get_model_parms or an rdif object from rdif.
theta	the estimated scaling parameter from rdif
fun	one of c("a_fun1", "a_fun2", "d_fun1", "d_fun2", "d_fun3"). 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)
```

format_pars *Helper function used to format parameters estimates*

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 *Extract item parameter estimates and their covariance matrix from* lavaan.

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 std.lv = TRUE.

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` *Extract item parameter estimates and their covariance matrix from* `mirt`.

Description

Extract item parameter estimates and their covariance matrix from `mirt`.

Usage

```
get_mirt_pars(mirt.object, cluster = NULL)
```

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"`.
`cluster`	optional cluster-ID vector used to compute a cluster-robust covariance matrix using Oakes bread, empirical score outer products aggregated by cluster, and a CR1 finite-sample correction.

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 *Extract and format item parameter estimates and their covariance matrix*

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, cluster = NULL)
```

Arguments

object	model fit from a multigroup analysis or list of model fits for each group for a 1-factor model. See Details.
cluster	optional clustering IDs used to compute a cluster-robust covariance matrix for mirt fits. For MultipleGroupClass, provide an atomic vector of length equal to the number of observations. For a list of SingleGroupClass fits, provide a list of cluster-ID vectors (one per model).

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.

When cluster is supplied for mirt fits, the covariance matrix is computed using a cluster-robust sandwich estimator with Oakes bread and cluster-summed empirical scores. A CR1 finite-sample correction is applied: (G/(G-1)) * ((N-1)/(N-p)), where G is the number of clusters, N is the number of observations, and p is the number of free parameters. 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 *The gradient matrix of* a_fun.

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

mle	the output of get_model_parms
theta	(optional) the scaling parameter. Replaces item-specific values of alpha if provided.
log	logical: return of log(a2/a1)?

Value

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

See Also

a_fun

---

grad_d *The gradient matrix of* d_fun.

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 — *The bi-square psi function.*

Description

If abs(u) > k , psi(u) = 0. Else, 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 — *The derivative of the bi-square psi function.*

Description

If `abs(u) > k`, `psi_prime(u) = 0`. Else, `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 `other.solutions`.

Usage

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

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.
starting.value	one of c("med", "lts", "min_rho", "all") or a numerical value to be used as the starting value. See description for details.
tol	convergence criterion for comparing subsequent values of estimate
maxit	maximum number of iterations
method	one of c("irls", "newton"). 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 `abs(u) > k` , `rho(u) = 1`. Else, `psi(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 *Compute a grid of bi-square Rho values*

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 *The covariance matrix of IRT scaling functions.*

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

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

Value

The covariance matrix of y_fun.

See Also

y_fun

y_fun *R-DIF scaling functions for item intercepts/thresholds and slopes.*

Description

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

- "a_fun1": computes a2/a1
- "a_fun2": computes log(a2/a1)
- "d_fun1": computes (d2 - d1)/a1 for each threshold
- "d_fun2": computes (d2 - d1)/a2 for each threshold
- "d_fun3": computes (d2 - d1)/sqrt((a1^2 + a2^2)/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