

Package ‘ccpsyc’

July 25, 2026

Type Package

Title Methods for Cross-Cultural Psychology

Version 0.2.8

Description Combines multiple functions that automate and simplify methods commonly employed in cross-cultural psychology, providing a unified analysis approach for measurement invariance testing, effect sizes for differential item functioning, factor congruence and multi-group reliability. Methods follow Fischer and Karl (2019) [<doi:10.3389/fpsyg.2019.01507>](https://doi.org/10.3389/fpsyg.2019.01507) and Gunn, Grimm and Edwards (2020) [<doi:10.1080/10705511.2019.1689507>](https://doi.org/10.1080/10705511.2019.1689507).

License GPL-3

URL <https://github.com/Jo-Karl/ccpsyc>

BugReports <https://github.com/Jo-Karl/ccpsyc/issues>

Depends R (>= 4.0.0)

Imports dplyr, lavaan, magrittr, MCMCpack, psych, RcppAlgos, readr, stats, tibble, tidyr

Suggests knitr, readxl, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxxygenNote 7.3.2

NeedsCompilation no

Author Johannes Karl [aut, cre]

Maintainer Johannes Karl <johannes.a.karl@gmail.com>

Repository CRAN

Date/Publication 2026-07-24 22:30:02 UTC

Contents

boot_inv_eff	2
calculate_dif_indices	3
clearing_fa	5
dMACS	6
equival	7
example	8
format_boot_inv_eff	9
invariance_eff	9
lavTestScore.clean	10
mg_rel_table	11
multi_group_eff	12
pancultural	13
prost	13
release_bonferroni	14
splitgroup	15
Index	16

boot_inv_eff	<i>Bootstrapped pairwise differences in psychometric function of groups.</i>
--------------	--

Description

Bootstrapped pairwise differences in psychometric function of groups.

Usage

```
boot_inv_eff(  
  n,  
  n_sample,  
  df,  
  items,  
  group,  
  eff_sizes = c("SDI2", "UDI2", "WSDI", "WUDI", "dmacs"),  
  seed = 2711  
)
```

Arguments

n	Number of bootstraps
n_sample	Number of participants to sample
df	Data to resample
items	Items to resample for the model as vector of strings
group	String variable indicating grouping variable
eff_sizes	Effect sizes to be returned
seed	Seed for replicability

Value

Returns a dataframe with the bootstrapped effect sizes based on the `invariance_eff` function in this package for two country comparisons.

Examples

```
two_country <- dplyr::filter(example, country %in% c("NZ" , "BRA"))
boot_inv_eff(n = 10,
             n_sample = 200, df = two_country, group = "country",
             items = paste0("voice",1:3, "m"))
```

`calculate_dif_indices` *Calculate Differential Item Functioning Indices (This will slowly replace invariance_eff)*

Description

Computes Signed and Unsigned Differential Item Functioning (SDIF/UDIF) indices along with differential MACS (dMACS) for multi-group confirmatory factor analysis.

Usage

```
calculate_dif_indices(
  df,
  items,
  group,
  nodewidth = 0.01,
  intercept_fix = 1,
  lower_lv = -10,
  upper_lv = 10,
  reference_group = NULL,
  focal_group = NULL,
  convergence_check = TRUE,
  ...
)
```

Arguments

<code>df</code>	A data frame containing the item responses and grouping variable
<code>items</code>	Character vector of item column names to include in the analysis
<code>group</code>	Character string specifying the grouping variable column name
<code>nodewidth</code>	Numeric value for numerical integration step size. Default is 0.01. Tips for setting nodewidth: • Smaller values (0.001-0.01) = Higher accuracy, slower computation • Larger values (0.02-0.1) = Lower accuracy, faster computation

	• For most applications, 0.01 provides good balance • Use smaller values (0.005) for high-stakes decisions • Use larger values (0.02-0.05) for exploratory analyses
intercept_fix	Integer specifying which item intercept to fix to 0 for identification. Default is 1
lower_lv	Numeric value for lower bound of latent variable range. Default is -10. Tips for setting lower_lv: • Should cover the practical range of latent variable values • Too narrow = Missing important DIF effects in tails • Too wide = Unnecessary computation • For typical applications: -4 to -6 covers ~99.9% of normal distribution • For extreme groups or highly variable data: -8 to -12 • Consider your data's latent variable distribution from initial CFA
upper_lv	Numeric value for upper bound of latent variable range. Default is 10. Tips for setting upper_lv: • Should mirror lower_lv for symmetric coverage • For typical applications: +4 to +6 covers ~99.9% of normal distribution • For extreme groups or highly variable data: +8 to +12 • Ensure range covers both groups' latent variable distributions • Check factor score ranges from preliminary analysis to guide choice
reference_group	Character string or numeric value specifying the reference group. If NULL, uses the first level of the grouping variable
focal_group	Character string or numeric value specifying the focal group. If NULL, uses the second level of the grouping variable
convergence_check	Logical indicating whether to check model convergence. Default is TRUE
...	Additional arguments passed to lavaan::cfa()

Details

Computational Considerations: The computational time is approximately proportional to: $(\text{upper\_lv} - \text{lower\_lv}) / \text{nodewidth} * \text{number\_of\_items}$

For a typical analysis with 10 items:

- Fast setting: range -4 to 4, nodewidth 0.02, about 4,000 calculations
- Balanced setting: range -6 to 6, nodewidth 0.01, about 12,000 calculations
- Precise setting: range -8 to 8, nodewidth 0.005, about 32,000 calculations

Value

A data frame with columns:

items	Item names
SDI2	Signed Differential Item Functioning index for focal group

UDI2	Unsigned Differential Item Functioning index for focal group
WSDI	Weighted Signed Differential Item Functioning index
WUDI	Weighted Unsigned Differential Item Functioning index
dmacs	Differential MACS values

Examples

```
two_country <- dplyr::filter(example, country %in% c("NZ", "BRA"))
calculate_dif_indices(
  df = two_country,
  items = paste0("voice", 1:3, "m"),
  group = "country",
  nodewidth = 0.05,
  lower_lv = -6,
  upper_lv = 6
)
```

clearing_fa	<i>Function to quickly organize and clear psych factor loadings</i>
-------------	---

Description

Function to quickly organize and clear psych factor loadings

Usage

```
clearing_fa(
  psych_fa,
  cutoff = 0.4,
  dbl_dist = 0.2,
  key_file = NULL,
  cleaned = TRUE
)
```

Arguments

psych_fa	Output from the psych package, can be either fa or principal with at least two dimensions
cutoff	Desired cutoff below which loadings are omitted defaults to .40
dbl_dist	Desired distance between highest and second highest loading for an item to remove double loadings, defaults to .20
key_file	Optional: Either a .csv or .xlsx file with at least two columns: 1 labeled item containing the item labels as in the data frame, 2 a column labeled wording containing the item wording.
cleaned	If true (default), only the cleaned solution with a message for descriptive stats are returned. If false the function returns a list of data frames one cleaned and one showing all in-between steps

Value

clean This column contains the assignment after removing NAs and double loadings
dir This column contains the direction (positive or negative) of the highest loading.

Examples

```
library(psych)
fa_solution <- fa(example[c(paste0("help", 1:6, "m"), c(paste0("voice", 1:5, "m")))], nfactors = 2)
clearing_fa(fa_solution)
```

dMACS	<i>Computes dMACS</i>
-------	-----------------------

Description

Computes dMACS

Usage

```
dMACS(fit.cfa, group1, group2)
```

Arguments

- fit.cfa Lavaan output object with two groups and a single factor.
- group1 String for first group in the grouping factor
- group2 String for second group in the grouping factor

Value

Returns dMACS for each item.

Examples

```
two_country <- dplyr::filter(example, country %in% c("NZ", "BRA"))
model <- "voice =~ voice1m + voice2m + voice3m"
fit <- lavaan::cfa(model, data = two_country, group = "country", std.lv = TRUE)
dMACS(fit, group1 = "BRA", group2 = "NZ")
```

equival	<i>One-step equivalence testing The function allows for a simple one step test of configural, metric, and scalar equivalence between multiple groups.</i>
---------	---

Description

One-step equivalence testing The function allows for a simple one step test of configural, metric, and scalar equivalence between multiple groups.

Usage

```
equival(x, dat, group, standart_lv = TRUE, orthog = TRUE, estim = "MLM")
```

Arguments

x	CFA model identical to models provided to lavaan.
dat	A data frame or tibble containing the raw data for the specified model.
group	A character string that indicates the column of dat that contains the grouping variable. e.g "country"
standart_lv	A boolean that indicates whether the latent variables should be standardised.
orthog	A boolean that indicates whether the latent variables should be orthogonal.
estim	A string indicating the estimator to be used MLM for complete data and MLR for incomplete data. Defaults to MLM

Value

Returns a data frame with the fit indices for each model and delta values comparing the different levels of equivalence. **For a step by step interpretation see.**

Examples

```
model <- "voice =~ voice1m + voice2m + voice3m
         help =~ help1m + help2m + help3m"
equival(x = model, dat = example, group = "country")
```

example

Help and Voice Behavior in different countries

Description

Help and Voice Behavior in different countries

Usage

example

Format

A data frame with 5201 rows and 13 variables:

country Country of sample

help1m First Help Item

help2m Second Help Item

help3m Third Help Item

help4m Fourth Help Item

help5m Fifth Help Item

help6m Sixth Help Item

help7m Seventh Help Item

voice1m First Voice Item

voice2m Second Voice Item

voice3m Third Voice Item

voice4m Fourth Voice Item

voice5m Fifth Voice Item

...

Source

<https://www.frontiersin.org/articles/10.3389/fpsyg.2019.01507/full>

format_boot_inv_eff *Improving boot effectsizes output*

Description

Improving boot effectsizes output

Usage

```
format_boot_inv_eff(x)
```

Arguments

x The output of a bootstrapped invariance effect call

Value

A formatted tibble with all effect sizes reported by boot_inv_eff from this package and significant determined by 95% CIs either crossing 0 or .30

Examples

```
two_country <- dplyr::filter(example, country %in% c("NZ" , "BRA"))
boot_inv_eff_result <- boot_inv_eff(n = 10,
                                   n_sample = 200, df = two_country, group = "country",
                                   items = paste0("voice",1:3, "m"))
format_boot_inv_eff(boot_inv_eff_result)
```

invariance_eff *Invariance Effect Sizes*

Description

Invariance Effect Sizes

Usage

```
invariance_eff(
  df,
  items,
  group,
  nodewidth = 0.01,
  intercept_fix = 1,
  lowerLV = -10,
  upperLV = 10,
  ...
)
```

Arguments

df	Multi-group dataframe
items	vector of items for the target construct
group	string defining grouping variable
nodewidth	Steps tested
intercept_fix	Which item should have a fixed intercept defaults to the first item
lowerLV	Lower range of latent variable tested
upperLV	Upper range of latent variable tested
...	Passes on to lavaan CFA functions

Value

Returns a dataframe with a row for each item comprising the uni-factorial solution and one column for each invariance effect size. A detailed interpretation of each effect size is provided in [Gunn et al. \(2019\)](#).

lavTestScore.clean	<i>Get more comprehensible output from lavTestScore</i>
--------------------	---

Description

Get more comprehensible output from lavTestScore

Usage

```
lavTestScore.clean(lavaan.fit, ndigit = 3, ...)
```

Arguments

lavaan.fit	Model fitted with lavaan
ndigit	Defines the rounding
...	Arguments passed to lavTestScore

Value

Returns a dataframe which contains one row for each constrained parameter in the model together with a chi-square test indicating whether the parameter significantly differs between groups. This is a cleaned version identical to [lavTestScore](#).

Author(s)

Maksim Rudnev

mg_rel_table	<i>Multi-group reliability table</i>
--------------	--------------------------------------

Description

Multi-group reliability table

Usage

```
mg_rel_table(
  df_s,
  measure_list,
  group,
  digitn = 3,
  conf.level = 0.95,
  seed = NULL
)
```

Arguments

df_s	The full dataframe with all groups and items.
measure_list	A named list of vectors containing the item names. The format should be <code>list(measure_name1 = c('Item1', 'Item2', 'Item3'), measure_name2 = c('Item1', 'Item2', 'Item3'))</code>
group	Grouping variable in the dataset as string for example "country"
digitn	Controls the amount of digits shown in the output
conf.level	Confidence level used for the alpha and omega intervals.
seed	Retained for backward compatibility and no longer used. The confidence intervals are computed analytically rather than by bootstrapping, so the results are fully reproducible without a seed.

Details

Cronbach's alpha is reported with a normal-theory confidence interval, McDonald's omega is estimated from a single-factor congeneric model fitted with [cfa](#) and reported with a delta-method confidence interval, the glb column reports the maximum split-half reliability (Guttman's lambda 4) from [splitHalf](#) and coefficient H follows Hancock and Mueller (2001). Measures that cannot be estimated for a given group, for instance because a scale has fewer than three items or a model fails to converge, are returned as NA.

Value

Returns a formatted dataframe with the reliability of all constructs by group

Examples

```
mg_rel_table(  
  df_s = example,  
  measure_list = list(voice = paste0("voice", 1:3, "m")),  
  group = "country"  
)
```

multi_group_eff	<i>Pairwise Effect sizes of similarities and difference in the psychometric structure between multiple groups</i>
-----------------	---

Description

Pairwise Effect sizes of similarities and difference in the psychometric structure between multiple groups

Usage

```
multi_group_eff(  
  df,  
  group,  
  items,  
  eff_sizes = c("SDI2", "UDI2", "WSDI", "WUDI", "dmacs")  
)
```

Arguments

- df Multi-Group data frame
- group String variable indicating the grouping variable
- items Vector of strings indicating items for the uni-factorial construct
- eff_sizes Effect sizes to be returned

Value

The function returns a list of dataframes with the first reporting the averaged results per item and the second reporting the pairwise comparisons.

Examples

```
example_s <- dplyr::filter(example, country %in% c("NZ", "BRA"))  
multi_group_eff(df = example, group = "country", items = paste0("voice",1:3, "m"))
```

pancultural	<i>Creating a Pan-Cultural Loading Matrix</i>
-------------	---

Description

Creating a Pan-Cultural Loading Matrix

Usage

```
pancultural(df, group, nfactors)
```

Arguments

df	A data frame contains the variables for the exploratory factor analysis and the grouping variable.
group	The name of the column tht cointains the grouping supplied as a string.
nfactors	The number of factors expected.

Value

returns a Pan-Cultural loading matrix.

Examples

```
pancultural(example, "country", 5)
```

prost	<i>Procrustes rotation function, returning Tucker's Phi</i>
-------	---

Description

Procrustes rotation function, returning Tucker's Phi

Usage

```
prost>Loading, norm, rotated = FALSE, digits = 2)
```

Arguments

loading	A correlation matrix to be rotated towards a target
norm	A correlation matrix that is the goal of the rotation
rotated	A TRUE/FALSE operator indicating if the rotated matrix should be returned in addition to Tucker's Phi
digits	The number of digits to be displayed in the output, defaults to 2

Value

Returns Tuckers Phi evaluating the congruence of the loading matrix to the normative matrix

release_bonferroni	<i>Examining chisquare improvement if paths are unconstrained. The function returns the paths to be unconstrained based on chisquare change. Adjusted P-values are calculated based on iterative Bonferroni corrections.</i>
--------------------	--

Description

Examining chisquare improvement if paths are unconstrained. The function returns the paths to be unconstrained based on chisquare change. Adjusted P-values are calculated based on iterative Bonferroni corrections.

Usage

```
release_bonferroni(lavaan.fit, ndigit = 3, exp_p = 0.05, ...)
```

Arguments

lavaan.fit	Model fitted with lavaan
ndigit	Number of digits to round chi and p to
exp_p	Expected p-value
...	Arguments passed to lavTestScore

Value

Returns a dataframe representing a Bonferroni corrected version of [lavTestScore.clean](#).

Author(s)

Maksim Rudnev

splitgroup	<i>Split by groups</i>
------------	------------------------

Description

Split by groups

Usage

```
splitgroup(df, group, named = FALSE, name.list = NA)
```

Arguments

df	Dataframe
group	Variable from the dataset that defines the groups
named	TRUE/FALSE argument wheter the resulting list should be named
name.list	Supply a list of names same length as number of groups

Value

Returns a list of dataframes with only the selected groups

Index

- * **datasets**
 - example, [8](#)
- boot_inv_eff, [2](#)
- calculate_dif_indices, [3](#)
- cfa, [11](#)
- clearing_fa, [5](#)
- dMACS, [6](#)
- equival, [7](#)
- example, [8](#)
- format_boot_inv_eff, [9](#)
- invariance_eff, [9](#)
- lavTestScore, [10](#)
- lavTestScore.clean, [10](#), [14](#)
- mg_rel_table, [11](#)
- multi_group_eff, [12](#)
- pancultural, [13](#)
- prost, [13](#)
- release_bonferroni, [14](#)
- splitgroup, [15](#)
- splitHalf, [11](#)