Package: elisr (via r-universe)

October 20, 2026

Type Package

Title Exploratory Likert Scaling

Version 0.1.1

Description An alternative to Exploratory Factor Analysis (EFA) for metrical data in R. Drawing on characteristics of classical test theory, Exploratory Likert Scaling (ELiS) supports the user exploring multiple one-dimensional data structures. In common research practice, however, EFA remains the go-to method to uncover the (underlying) structure of a data set. Orthogonal dimensions and the potential of overextraction are often accepted as side effects. As described in Müller-Schneider (2001) <doi:10.1515/zfsoz-2001-0404>), ELiS confronts these problems. As a result, 'elisr' provides the platform to fully exploit the exploratory potential of the multiple scaling approach itself.

License GPL (>= 3)

URL https://github.com/sbissantz/elisr

BugReports https://github.com/sbissantz/elisr/issues

Depends R (>= 4.0.0)

Imports stats (>= 4.0.0)

Suggests covr, devtools, knitr, pkgdown, psych, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en, de

LazyData true

RoxygenNote 7.1.1

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

Date/Publication 2021-05-13 11:10:57 UTC

RemoteUrl https://github.com/sbissantz/elisr

RemoteRef HEAD

RemoteSha 3faf47bf43fb2f0f791bc16327dd3dfb61791fe1

Contents

checks 2
disjoint 3
msdf 5
overlap 6
print.msdf 7
trust 8
utilities 9
workhorses 11

Index **13**

`checks` *Miscellaneous input validation*

Description

A set of test functions to ensure valid input and give helpful advice if it is not.

`check_df()` guarantees that `x` is an appropriate data frame for the analysis. That means: It verifies that `x` has less than two variables (a single item can't build a core), `x` has column names (used to pre-build `scls` in the overlapping process), if the column names are unique, and not of type `NA`. It throws an error if any of these requirements are not met. Additionally, it warns the user if the provided `colnames` are not unique or `NA`.

`check_sclvals()` tests whether `x` is a two element vector and throws an error if not. Integers are coerced to be of type `double`. Additionally, the function ensures that the first value is smaller than the second. Remember that checking for a two element vector implicitly secures that `x` is not `NULL` (because `NULL` is a logical constant of length '0').

`compare_sclvals()` makes sure that the `sclvals` set with `overlap()` are equal to those set with `disjoint()`. It throws an error if not.

`check_mrit()` guarantees that the input is a double vector of length one. Moreover, the function secures that the lower bound is unique and ranges between '0' and '1' (it throws an error if not). In addition, it warns a user pre-determining a fragment.

`check_ovlp()` safeguards that `x` is a character vector of length '1'. That means, it throws an error if not. Note that `switch()` within `disjoint()` and `overlap()` takes care of the input string itself. It throws an error when the given character doesn't match any available option.

`check_msdf()` guards against inputs that are not of type 'msdf'. It throws an error if not.

`check_neg()` verifies that the input is a logical constant of length 1 and not a missing value (this is necessary because objects of type `NA` are logical constants of length 1, too).

`check_comp()` examines the correlation matrix, `cor(df)`. It complains (throws an error) if no correlation in `cor(df)` is greater than the specified `mrit_min`.

Usage

```
check_df(x)

check_sclvals(x)

compare_sclvals(x, x_attr)

check_mrit(x)

check_ovlp(x)

check_msdf(x)

check_neg(x)

check_comp(x, mrit_min, use)
```

Arguments

x	some arbitrary input to be checked.
x_attr	a numeric vector of length 2 indicating the start- and endpoint of a scale.
mrit_min	a numeric constant of length 1 to specify the marginal corrected item-total correlation.
use	an optional string to specify how missing values enter the analysis. See use in cor for details.

Details

All functions are internal functions.

Value

All functions are called for their side effects. If there are no errors or warnings, no value is returned.

---

disjoint *Multiple scaling – the disjoint way*

Description

disjoint() returns a multiple, disjointedly scaled version of the specified data frame. This so called msdf sets up the building block for further analysis with overlap() (type ?overlap).

Usage

```
disjoint(
  df,
  mrit_min = 0.3,
  negative_too = FALSE,
  sclvals = NULL,
  use = "pairwise.complete.obs"
)
```

Arguments

df	a data frame (with more than two items and unique, non-NA column names).
mrit_min	a numeric constant of length 1 to specify the marginal corrected item-total correlation. Its value is in the range of 0-1. The default is set to .3.
negative_too	a logical constant indicating whether reversed items are included in the analysis. The default is set to FALSE.
sclvals	a numeric vector of length 2 indicating the start- and endpoint of a scale. Use something like c(min,max).
use	an optional string to specify how missing values enter the analysis. See use in cor for details. The default is set to pairwise.complete.obs.

Details

use clarifies how to set up a correlation matrix in the presence of missing values. In a typical scaling process this happens at least twice. First, when determining the core items (the two items in the correlation matrix with the highest linear relationship). Second, when an item is proposed for an emerging scale.

Note that disjoint() uses cor's default method pearson.

Value

A multiple scaled data frame (msdf).

References

Müller-Schneider, T. (2001). Multiple Skalierung nach dem Kristallisationsprinzip / Multiple Scaling According to the Principle of Crystallization. Zeitschrift für Soziologie, 30(4), 305-315. https://doi.org/10.1515/zfsoz-2001-0404

Examples

```
# Use only positive correlations
disjoint(mtcars, mrit_min = .4)

# Include negative correlations
disjoint(mtcars, mrit_min = .4, negative_too = TRUE, sclvals = c(1,7))

# Change the treatment of missing values
disjoint(mtcars, mrit_min = .4, use = "all.obs")
```

msdf *Create and test for a multiple scaled data frame*

Description

`new_msdf()` creates a multiple scaled data frame (`msdf`).

`is.msdf()` tests if an object is a multiple scaled data frame.

Usage

```
new_msdf(x = list(), method, mrit_min, negative_too, sclvals = NULL, df)

is.msdf(x)
```

Arguments

`x`	either a multiple scaled data frame (`new_msdf()`) or an arbitrary object to test (`is_msdf()`).
`method`	the method used to produce the multiple scaled data frame (either `disjoint()` or `overlap()`).
`mrit_min`	a numeric constant of length one to specify the marginal corrected item-total correlation. Its value is in the range of 0-1.
`negative_too`	a logical constant indicating whether reversed items are included in the analysis. The default is set to `FALSE`.
`sclvals`	a numeric vector of length two indicating the start- and endpoint of a scale.
`df`	the data frame to analyze.

Details

Objects of type ‘msdf‘ are for internal use only.

Value

`new_msdf()` returns a list of data frames with a few attributes that partially summarize the scaling process.

`is.msdf()` returns a logical vector of length one. `TRUE` indicates that the object is of type `msdf`.

overlap *Multiple scaling – the overlapping way*

Description

`overlap()` returns a overlapped version (either extended, or reversed, or both) of the specified `msdf`.

Usage

```
overlap(
  msdf,
  mrit_min = NULL,
  negative_too = FALSE,
  overlap_with = "fragment",
  sclvals = NULL,
  use = "pairwise.complete.obs"
)
```

Arguments

`msdf`	a multiple scaled data frame (built with `disjoint()`).
`mrit_min`	a numeric constant of length 1 to specify the marginal corrected item-total correlation. Its value is in the range of 0-1. The default is set to `.3`.
`negative_too`	a logical constant indicating whether reversed items are included in the analysis. The default is set to `FALSE`.
`overlap_with`	a string telling `overlap()` the set of items for the extension. To build up on all variables of a fragment use `fragment`, for the core-only option type `core`. The default is set to "fragment".
`sclvals`	a numeric vector of length 2 indicating the start- and endpoint of a scale. Use something like `c(min,max)`.
`use`	an optional string to specify how missing values enter the analysis. See `use` in `cor` for details. The default is set to `pairwise.complete.obs`.

Details

`use` clarifies how to set up a correlation matrix in the presence of missing values. In a typical scaling process this happens at least twice. First, when determining the core items (the two items in the correlation matrix with the highest linear relationship). Second, when an item is proposed for an emerging scale.

Note that `overlap()` uses `cor`'s default method `pearson`.

Value

A multiple scaled data frame (`msdf`).

References

Müller-Schneider, T. (2001). Multiple Skalierung nach dem Kristallisationsprinzip / Multiple Scaling According to the Principle of Crystallization. Zeitschrift für Soziologie, 30(4), 305-315. https://doi.org/10.1515/zfsoz-2001-0404

Examples

```
# Build a msdf
msdf <- disjoint(mtcars, mrit_min = .4)

# Use positive correlations (extend on fragments)
overlap(msdf, mrit_min = .6)

# Use positive correlations (extend on cores)
overlap(msdf, mrit_min = .6, overlap_with = "core")

# Include negative correlations
overlap(msdf, mrit_min = .7, negative_too = TRUE, sclvals = c(-3,3))

# Change the treatment of missing values
overlap(msdf, mrit_min = .6, use = "all.obs")
```

`print.msdf` *Print method for a multiple scaled data frame*

Description

The print method for a multiple scaled data frame. It summarizes the `msdf` using values of classical test theory. Note that every line in the output tags a new stage in the development process of each gradually emerging scale.

Usage

```
## S3 method for class 'msdf'
print(x, digits = 2, use = "pairwise.complete.obs", ...)
```

Arguments

`x`	a multiple scaled data frame (built with either `disjoint()` or `overlap()`).
`digits`	an integer constant to determine the number of printed digits. See `digits` in `options` for details.
`use`	an optional string to specify how missing values enter the analysis. See `use` in `cor` for details. The default is set to `pairwise.complete.obs`.
`...`	Additional arguments to the method which will be ignored.

Details

use clarifies how to set up a correlation matrix in the presence of missing values. In a typical scaling process this happens at least twice. First, when determining the core items (the two items in the correlation matrix with the highest linear relationship). Second, when an item is proposed for an emerging scale.

Value

A list of summary statistics: the marginal corrected item-total correlation (mrit), Cronbach's alpha (alpha), and the average correlation (rbar).

References

Müller-Schneider, T. (2001). Multiple Skalierung nach dem Kristallisationsprinzip / Multiple Scaling According to the Principle of Crystallization. Zeitschrift für Soziologie, 30(4), 305-315. https://doi.org/10.1515/zfsoz-2001-0404

---

trust *German General Social Survey's Trust Items*

Description

A subset of the German General Social Survey (ALLBUS) 2018 containing its trust items (F018).

Usage

```
trust
```

Format

Participants were asked about their trust in public institutions and organizations. '1' means they have absolutely no trust at all, whereas '7' represents a great deal of trust. The data frame has 3477 rows and 13 variables:

healserv Health service

fccourt German constitutional court

bundtag German Parliament

munadmin Municipal administration

judsyst Judicial system

tv Television

newsppr Newspapers

uni Universities and other institutes of higher education

fedgovt German government

police Police

polpati Political parties

eucomisn European Commission

eupalmnt European Parliament

Details

Please note that in comparison to the original data set participant order has been randomly rearranged to further ensure anonymity. This might lead to differences when trying to reproduce the results of the analysis with the original data set (mentioned below).

References

GESIS - Leibniz-Institut für Sozialwissenschaften (2019). German General Social Survey - ALLBUS 2018. GESIS Datenarchiv, Köln. ZA5272 Datenfile Version 1.0.0, https://doi.org/10.4232/1.13325.

Examples

```
# Use trust with disjoint
disjoint(trust, mrit_min = .4)
```

---

utilities *Miscellaneous utility functions*

Description

A set of utility functions to calculate characteristic values of classical test theory, reverse a variable, and extract the relevant bits from various objects.

`calc_rit()` calculates the corrected item-total correlation of a scale or fragment using a part-whole correction. Thus, the item itself is excluded in the calculation process.

`calc_rbar()` calculates the average correlation of a fragment or scale.

`calc_alpha()` calculates the internal consistency of a scale or fragment using Cronbach's alpha.

`rvrs_var()` reverses an item using the specified scaling values. It handles the following types of scales:

- ...-3 -2 -1 0 1 2 3..., e.g., `sclvals = c(-3, 3)`
- 0 1 2 3 4 5 6..., e.g., `sclvals = c(0, 7)`
- 1 2 3 4 5 6 7..., e.g., `sclcals = c(1, 7)`

`rvrs_note()` gets the full report of reversed variables and reports a unique list of them.

`extr_core()` is used to extract all pairs of core items from a fragment.

`extr_core_nms()` is used to extract the names of all pairs of core items from a given fragment.

`extreb_itms()` builds the counterpart of a fragment from the given item names. Therefore, the counterpart includes all variables that are not part of a fragment but which are mentioned in the specified data set.

`nme_msdf()` renames the components of a multiple scaled data frame. The naming scheme is `scl_n`. `scl` stands for 'scale' and `n` specifies the number of fragments or scales. For example, the first component is called `scl_1`.

Usage

```
calc_rit(scl, use)

calc_rbar(scl, use)

calc_alpha(scl, use)

rvrs_var(var, sclvals)

rvrs_note(msg, applicant)

extr_core(scl)

extr_core_nms(scl)

extreb_itms(df, itm_nms)

nme_msdf(x)
```

Arguments

scl	a scale within a multiple scaled data frame.
use	an optional string indicating how to deal with missing values, See use in cor for details.
var	a variable or item (often a column from a data frame).
sclvals	the start- and end point of a scale (specify: c(sp,ep)).
msg	a reverse message sent from either disjoint() or overlap().
applicant	the function which wants to leave messages disjoint() or overlap().
df	a data frame object.
itm_nms	the names of an item from a scale.
x	a multiple scaled data frame.

Details

All functions are internal functions.

Value

calc_rit() returns a numeric vector of length one. It is used to examine the coherence between item and overall score.

calc_rbar() returns a numeric vector of length one that reports the inter-item correlation.

calc_alpha() returns a numeric vector of length one which is used to assess the internal consistency of a scale.

rvrs_var() returns the reversed numeric vector using the above reversing scheme.

rvrs_note() is called for its side effects. It leaves a message when an item is reversed.

`extr_core()` returns a numeric vector of length two, which contains the two items with the highest correlation in a fragment or scale.

`extr_core_nms()` returns character vector of length two, which contains the names of the two items with the highest correlation in a fragment or scale.

`extrev_itms()` returns vector of length m minus two, where 'm' specifies the number of variables in a given data set. It contains the items of a scale's counterpart.

`nme_msdf()` returns a character vector that numbers each element of its input according to the above naming scheme.

`workhorses` *elisr's quadriga*

Description

The four workhorses inside `elisr`'s user functions `disjoint()` and `overlap()`.

`disj_pci()` is a loop which runs through the following steps: (1) Set up a (first) scale. (2) Find the two items with the highest positive correlation in the data set. (3) If the absolute value of this correlation is greater than the pre-specified lower bound (`mrit_min`), add up the two items to build the core of the emerging scale. (4) As long as the value of the correlation between the sum-score and a remaining item in the data frame is greater than `mrit_min`, flavor the scale with the appropriate item. (5) If there are at least two leftovers in the data frame that meet the inclusion criterion, start over again.

`disj_nci()` is almost identical to `disj_pci()`, though step (4) varies slightly from above. To take negative correlations into account, `disj_nci()` flavors the scale with appropriate item as long as the *absolute* value of the correlation between the sum-score and a remaining items in the data frame is greater than `mrit_min`.

`ovlp_pci()` takes a disjointedly built scale fragment and tries to extend it with those items in the data set, which are not yet built into the fragment (aka., its counterpart). Because `ovlp_pci()` does this for every disjointedly built scale fragment it is a multiple one-dimensional extension of `disj_pci()`.

The only difference to `ovlp_pci()` is that `ovlp_nci()` can handle reversed items. The extension algorithm remains almost the same; `ovlp_nci()` flavors *each* scale fragment with appropriate items from its counterpart as long as the *absolute* value of the correlation between the sum-score and a remaining item is greater than `mrit_min`. Thus, it is a multiple one-dimensional extension of `disj_nci()`:

Usage

```
disj_pci(df, mrit_min, use)

disj_nci(df, mrit_min, sclvals, use)

ovlp_pci(msdf, mrit_min, overlap_with, use)

ovlp_nci(msdf, mrit_min, overlap_with, sclvals, use)
```

Arguments

df	a data frame object.
mrit_min	a numerical constant to specify the marginal corrected item total correlation. The value must be in the range of 0-1.
use	an optional string to specify how missing values will enter the analysis. See use in cor for details.
sclvals	a numerical vector of length 2 indicating the start- and endpoint of a scale.
msdf	a multiple scaled data frame (built with disjoint()).
overlap_with	a string telling overlap() the set of items for the extension. To build up on all variables of a fragment use fragment, for the core-only option type core. The default is set to "fragment".

Details

All functions are internal functions.

The use argument specifies how to set up a correlation matrix in the presence of missing values. In a typical scaling process this happens at least twice. First, when determining the core items (the two items in the correlation matrix with the highest linear relationship). Second, when an item is proposed for an emerging scale.

Note that all functions use cor's default method pearson.

Value

disj_pci() and disj_nci() both return a list of data frames which result from applying the above-mentioned algorithm.

ovlp_pci() and ovlp_nci() often return an *extended* a list of data frames.

References

Müller-Schneider, T. (2001). Multiple Skalierung nach dem Kristallisationsprinzip / Multiple Scaling According to the Principle of Crystallization. Zeitschrift für Soziologie, 30(4), 305-315. https://doi.org/10.1515/zfsoz-2001-0404

Index

∗ **datasets**
    trust, 8

calc_alpha *(*utilities*)*, 9
calc_rbar *(*utilities*)*, 9
calc_rit *(*utilities*)*, 9
check_comp *(*checks*)*, 2
check_df *(*checks*)*, 2
check_mrit *(*checks*)*, 2
check_msdf *(*checks*)*, 2
check_neg *(*checks*)*, 2
check_ovlp *(*checks*)*, 2
check_sclvals *(*checks*)*, 2
checks, 2
compare_sclvals *(*checks*)*, 2
cor, *3, 4*, *6, 7*, *10*, *12*

disj_nci *(*workhorses*)*, 11
disj_pci *(*workhorses*)*, 11
disjoint, 3

extr_core *(*utilities*)*, 9
extr_core_nms *(*utilities*)*, 9
extreb_itms *(*utilities*)*, 9

is.msdf *(*msdf*)*, 5

msdf, 5

new_msdf *(*msdf*)*, 5
nme_msdf *(*utilities*)*, 9

options, *7*
overlap, 6
ovlp_nci *(*workhorses*)*, 11
ovlp_pci *(*workhorses*)*, 11

print.msdf, 7

rvrs_note *(*utilities*)*, 9
rvrs_var *(*utilities*)*, 9

trust, 8

utilities, 9

workhorses, 11