

Package ‘DAKS’

June 6, 2016

Version 2.1-3

Date 2016-06-05

Title Data Analysis and Knowledge Spaces

Description Functions and an example dataset for the psychometric theory of knowledge spaces. This package implements data analysis methods and procedures for simulating data and quasi orders and transforming different formulations in knowledge space theory. See `package?DAKS` for an overview.

Author Ali Uenlue [aut, cre], Anatol Sargin [aut]

Maintainer Ali Uenlue <ali.uenlue@tum.de>

Depends R (>= 3.3.0), relations, sets

LazyLoad yes

LazyData yes

License GPL (>= 2)

URL <http://www.meb.edu.tum.de>

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-06 13:14:35

R topics documented:

DAKS-package	2
corr_iita	3
hasse	5
iita	6
imp2state	8
ind_gen	9
mini_iita	11
ob_counter	12
orig_iita	14
pattern	15
pisa	17

pop_iita	18
pop_variance	20
print.iita	22
print.pat	23
print.popiita	24
print.summpopiita	25
print.ztest	26
simu	27
state2imp	29
summary.iita	30
summary.popiita	31
variance	32
z_test	34

Index	36
--------------	-----------

DAKS-package

Data Analysis and Knowledge Spaces: The R Package DAKS

Description

The package **DAKS** implements three inductive item tree analysis algorithms for deriving quasi orders from binary data, the original, corrected, and minimized corrected algorithms. It provides functions for computing population and estimated asymptotic variances of the *diff* fit measures, and for switching between test item and knowledge state representations. Other features are a Hasse diagram drawing device, a data and quasi order simulation tool based on a finite mixture latent variable model, and a function for computing response pattern and knowledge state frequencies.

Details

Package: **DAKS**
Type: Package
Version: 2.1-3
Date: 2016-06-05
License: GPL (>= 2)

Knowledge space theory is a recent psychometric test theory based on combinatorial mathematical structures (order and lattice theory); see *Doignon and Falmagne (1999)*. Solvability dependencies between dichotomous test items play an important role in knowledge space theory. Utilizing hypothesized dependencies between items, knowledge space theory has been successfully applied for the computerized, adaptive assessment and training of knowledge. For instance, see the ALEKS system, a fully automated math tutor on the Internet (<http://www.aleks.com/>).

The package **DAKS** is implemented based on the S3 system. It comes with a namespace and consists of the following functions (all functions are external, there are no internal functions): [corr_iita](#), [hasse](#), [iita](#), [imp2state](#), [ind_gen](#), [mini_iita](#), [ob_counter](#), [orig_iita](#), [pattern](#), [pop_iita](#), [pop_variance](#), [print.iita](#), [print.pat](#), [print.popiita](#), [print.summpopiita](#), [print.ztest](#), [simu](#), [state2imp](#), [summary.iita](#), [summary.popiita](#), [variance](#), and [z_test](#). There is an empir-

ical dataset, [pisa](#), accompanying the package **DAKS**. This dataset is part of the 2003 Programme for International Student Assessment (PISA; <http://www.pisa.oecd.org/>).

Author(s)

Anatol Sargin, Ali Uenlue

Maintainer: Ali Uenlue <ali.uenlue@tum.de>

References

Doignon, J.-P. and Falmagne, J.-C. (1999) *Knowledge Spaces*. Berlin, Heidelberg, and New York: Springer-Verlag.

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Schrepp, M. (1999) On the empirical construction of implications between bi-valued test items. *Mathematical Social Sciences*, **38**, 361–375.

Schrepp, M. (2003) A method for the analysis of hierarchical dependencies between items of a questionnaire. *Methods of Psychological Research*, **19**, 43–79.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

corr_iita

Corrected Inductive Item Tree Analysis

Description

corr_iita performs the corrected inductive item tree analysis procedure and returns the corresponding *diff* values.

Usage

```
corr_iita(dataset, A)
```

Arguments

dataset a required data frame or matrix consisting of binary, 1 or 0, numeric data.

A a required list of competing quasi orders (surmise relations), for instance obtained from a call to [ind_gen](#).

Details

Corrected inductive item tree analysis is a data analysis method for deriving knowledge structures (more precisely, surmise relations) from binary data. Details on this procedure can be found in [iita](#). The set of competing quasi orders is passed via the argument `A`, so any selection set of quasi orders can be used.

The set of competing quasi orders must be a list of objects of the class `set`. These objects (quasi orders) consist of 2-tuples (i, j) of the class `tuple`, where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

The data must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments `dataset` and `A` are of required types, `corr_iita` returns a named list of the following components:

<code>diff.value</code>	a vector of the <i>diff</i> values corresponding to the competing quasi orders in <code>A</code> .
<code>error.rate</code>	a vector of the error rates corresponding to the competing quasi orders in <code>A</code> .

Note

The function `iita` can be used to perform one of the three inductive item tree analysis procedures (including the corrected inductive item tree analysis method) selectively. Whereas for the function `corr_iita` a selection set of competing quasi orders has to be passed via the argument `A` manually, `iita` automatically generates a selection set from the data using the inductive generation procedure implemented in `ind_gen`.

The latter approach using `iita` is common so far, in knowledge space theory, where the inductive data analysis methods have been utilized for exploratory derivations of surmise relations from data. The function `corr_iita`, on the other hand, can be used to select among surmise relations for instance obtained from querying experts or from competing psychological theories.

Author(s)

Anatol Sargin, Ali Uenlue

References

- Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.
- Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[orig_iita](#) for original inductive item tree analysis; [mini_iita](#) for minimized corrected inductive item tree analysis; [iita](#), the interface that provides the three inductive item tree analysis methods under one umbrella; [pop_variance](#) for population asymptotic variances of *diff* coefficients;

[variance](#) for estimated asymptotic variances of *diff* coefficients; [pop_iita](#) for population inductive item tree analysis. See also [DAKS-package](#) for general information about this package.

Examples

```
ind <- ind_gen(ob_counter(pisa))
corr_iita(pisa, ind)
```

hasse

Hasse Diagram of Surmise Relation

Description

hasse plots the Hasse diagram of a surmise relation (more precisely, of its corresponding quotient set).

Usage

```
hasse(imp, items)
```

Arguments

imp	a required object of class set representing the set of implications, for instance obtained from a call to iita .
items	a required numeric giving the number of items of the domain taken as basis for imp.

Value

If the arguments `imp` and `items` are of required types, `hasse` produces a plot, and returns a list of the equally informative items.

Note

The function `hasse` is not capable of plotting equally informative items. This is why equally informative items are returned in a list.

A set of implications, an object of the class [set](#), consists of 2-tuples (i, j) of the class [tuple](#), where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

Author(s)

Anatol Sargin, Ali Uenlue

References

Doignon, J.-P. and Falmagne, J.-C. (1999) *Knowledge Spaces*. Berlin, Heidelberg, and New York: Springer-Verlag.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[iita](#), the interface that provides the three inductive item tree analysis methods under one umbrella. See also [DAKS-package](#) for general information about this package.

Examples

```
## requires the package Rgraphviz from Bioconductor
## users must have installed Graphviz on their computers
## Not run:
hasse(iita(pisa, v = 2)$implications, 5)

## End(Not run)
```

iita

Inductive Item Tree Analysis

Description

`iita` can be used to perform one of the three inductive item tree analysis algorithms (original, corrected, and minimized corrected) selectively.

Usage

```
iita(dataset, v)
```

Arguments

<code>dataset</code>	a required data frame or matrix consisting of binary, 1 or 0, numeric data.
<code>v</code>	a required numeric giving the inductive item tree analysis algorithm to be performed; $v = 1$ (minimized corrected), $v = 2$ (corrected), and $v = 3$ (original).

Details

The three inductive item tree analysis algorithms are exploratory methods for extracting quasi orders (surmise relations) from data. In each algorithm, competing binary relations are generated (in the same way for all three versions), and a fit measure (differing from version to version) is computed for every relation of the selection set in order to find the quasi order that fits the data best. In all three algorithms, the idea is to estimate the numbers of counterexamples for each quasi order, and to find, over all competing quasi orders, the minimum value for the discrepancy between the observed

and expected numbers of counterexamples. The three data analysis methods differ in their choices of estimates for the expected numbers of counterexamples. (For an item pair (i, j) , the number of subjects solving item j but failing to solve item i , is the corresponding number of counterexamples. Their response patterns contradict the interpretation of (i, j) as ‘mastering item j implies mastering item i .’) The algorithms are described in the paper about the **DAKS** package by *Uenlue and Sargin (2010)*, and in the paper by *Sargin and Uenlue (2009)*.

`iita` calls `ind_gen` for constructing the set of competing quasi orders according to the inductive generation procedure. Subject to the selected version to be performed, `iita` computes the discrepancies between observed and expected numbers of counterexamples under each relation, and finds a quasi order with the minimum discrepancy (*diff*) value.

A set of implications, an object of the class `set`, consists of 2-tuples (i, j) of the class `tuple`, where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

The data must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments `dataset` and `v` are of required types, `iita` returns a named list consisting of the following five components:

<code>diff</code>	a vector giving the <i>diff</i> values corresponding to the (inductively generated) competing quasi orders.
<code>implications</code>	an object of class <code>set</code> representing the solution quasi order (with smallest <i>diff</i> value) under the selected algorithm.
<code>error.rate</code>	a value giving the estimated error rate corresponding to the best fitting quasi order.
<code>selection.set.index</code>	a numeric giving the index of the solution quasi order in the selection set.
<code>v</code>	the version used; <code>v = 1</code> (minimized corrected), <code>v = 2</code> (corrected), and <code>v = 3</code> (original).

Note

The function `iita` can be used to perform one of the three inductive item tree analysis procedures selectively. Whereas for the functions `orig_iita`, `corr_iita`, `mini_iita` selection sets of competing quasi orders have to be passed via an argument manually, `iita` automatically generates a selection set from the data using the inductive generation procedure implemented in `ind_gen`.

The latter approach using `iita` is common so far, in knowledge space theory, where the inductive data analysis methods have been utilized for exploratory derivations of surmise relations from data. The functions `orig_iita`, `corr_iita`, `mini_iita`, on the other hand, can be used to select among surmise relations for instance obtained from querying experts or from competing psychological theories.

Author(s)

Anatol Sargin, Ali Uenlue

References

- Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.
- Schrepp, M. (1999) On the empirical construction of implications between bi-valued test items. *Mathematical Social Sciences*, **38**, 361–375.
- Schrepp, M. (2003) A method for the analysis of hierarchical dependencies between items of a questionnaire. *Methods of Psychological Research*, **19**, 43–79.
- Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[orig_iita](#) for original inductive item tree analysis; [corr_iita](#) for corrected inductive item tree analysis; [mini_iita](#) for minimized corrected inductive item tree analysis; [ind_gen](#) for inductive generation procedure; [pop_variance](#) for population asymptotic variances of *diff* coefficients; [variance](#) for estimated asymptotic variances of *diff* coefficients; [z_test](#) for one- and two-sample Z-tests; [pop_iita](#) for population inductive item tree analysis. See also [DAKS-package](#) for general information about this package.

Examples

```
iita(pisa, v = 1)
iita(pisa, v = 3)
```

imp2state

Transformation from Implications to Knowledge States

Description

imp2state transforms a set of implications (ought to be a surmise relation) to the corresponding set of knowledge states (the quasi ordinal knowledge space).

Usage

```
imp2state(imp, items)
```

Arguments

- imp** a required object of class [set](#) representing the set of implications, for instance obtained from a call to [iita](#).
- items** a required numeric giving the number of items of the domain taken as basis for **imp**.

Value

If the arguments `imp` and `items` are of required types, `imp2state` returns a matrix consisting of ones or zeros (the quasi ordinal knowledge space), in which each row represents the 1/0-pattern of a knowledge state.

Note

For any set of implications the returned knowledge structure is a quasi ordinal knowledge space. In case of a surmise relation this is Birkhoff's theorem. For details refer to *Doignon and Falmagne (1999, Theorem 1.49)*.

A set of implications, an object of the class `set`, consists of 2-tuples (i, j) of the class `tuple`, where a 2-tuple (i, j) is interpreted as 'mastering item j implies mastering item i .'

Author(s)

Anatol Sargin, Ali Uenlue

References

Doignon, J.-P. and Falmagne, J.-C. (1999) *Knowledge Spaces*. Berlin, Heidelberg, and New York: Springer-Verlag.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[state2imp](#) for transformation from knowledge states to implications. See also [DAKS-package](#) for general information about this package.

Examples

```
x <- iita(pisa, v = 1)
imp2state(x$implications, ncol(pisa))
```

ind_gen

Inductive Generation Procedure

Description

`ind_gen` generates inductively a set of competing quasi orders.

Usage

```
ind_gen(b)
```

Arguments

b a required matrix of the numbers of counterexamples for all pairs of items, for instance obtained from a call to `ob_counter`.

Value

If the argument `b` is of required type, `ind_gen` returns a list of the inductively generated quasi orders.

Note

The function `iita` calls `ind_gen` for constructing the set of competing quasi orders according to the inductive generation procedure.

The set of competing quasi orders is a list of objects of the class `set`. These objects (quasi orders) consist of 2-tuples (i, j) of the class `tuple`, where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Schrepp, M. (1999) On the empirical construction of implications between bi-valued test items. *Mathematical Social Sciences*, **38**, 361–375.

Schrepp, M. (2003) A method for the analysis of hierarchical dependencies between items of a questionnaire. *Methods of Psychological Research*, **19**, 43–79.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

`ob_counter` for computation of numbers of counterexamples; `iita`, the interface that provides the three inductive item tree analysis methods under one umbrella; `z_test` for one- and two-sample Z-tests. See also `DAKS-package` for general information about this package.

Examples

```
ob <- ob_counter(pisa)
ind_gen(ob)
```

 mini_iita

 Minimized Corrected Inductive Item Tree Analysis

Description

mini_iita performs the minimized corrected inductive item tree analysis procedure and returns the corresponding *diff* values.

Usage

```
mini_iita(dataset, A)
```

Arguments

dataset	a required data frame or matrix consisting of binary, 1 or 0, numeric data.
A	a required list of competing quasi orders (surmise relations), for instance obtained from a call to ind_gen .

Details

Minimized corrected inductive item tree analysis is a data analysis method for deriving knowledge structures (more precisely, surmise relations) from binary data. Details on this procedure can be found in [iita](#). The set of competing quasi orders is passed via the argument A, so any selection set of quasi orders can be used.

The set of competing quasi orders must be a list of objects of the class [set](#). These objects (quasi orders) consist of 2-tuples (i, j) of the class [tuple](#), where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

The data must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments dataset and A are of required types, corr_iita returns a named list of the following components:

diff.value	a vector of the <i>diff</i> values corresponding to the competing quasi orders in A.
error.rate	a vector of the error rates corresponding to the competing quasi orders in A.

Note

The function [iita](#) can be used to perform one of the three inductive item tree analysis procedures (including the minimized corrected inductive item tree analysis method) selectively. Whereas for the function mini_iita a selection set of competing quasi orders has to be passed via the argument A manually, iita automatically generates a selection set from the data using the inductive generation procedure implemented in [ind_gen](#).

The latter approach using `iita` is common so far, in knowledge space theory, where the inductive data analysis methods have been utilized for exploratory derivations of surmise relations from data. The function `mini_iita`, on the other hand, can be used to select among surmise relations for instance obtained from querying experts or from competing psychological theories.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

`orig_iita` for original inductive item tree analysis; `corr_iita` for corrected inductive item tree analysis; `iita`, the interface that provides the three inductive item tree analysis methods under one umbrella; `pop_variance` for population asymptotic variances of *diff* coefficients; `variance` for estimated asymptotic variances of *diff* coefficients; `pop_iita` for population inductive item tree analysis. See also [DAKS-package](#) for general information about this package.

Examples

```
ind <- ind_gen(ob_counter(pisa))
mini_iita(pisa, ind)
```

ob_counter

Computation of Numbers of Counterexamples

Description

`ob_counter` computes from a dataset for all item pairs the corresponding numbers of counterexamples.

Usage

```
ob_counter(dataset)
```

Arguments

`dataset` a required data frame or matrix consisting of binary, 1 or 0, numeric data.

Details

For an item pair (i, j) , the number of subjects solving item j but failing to solve item i , is the corresponding number of counterexamples. Their response patterns contradict the interpretation of (i, j) as ‘mastering item j implies mastering item i .’

The data must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the argument `dataset` is of required type, `ob_counter` returns a matrix of the numbers of counterexamples for all pairs of items.

Note

The function `ind_gen` can be used to inductively generate from the returned matrix of the numbers of counterexamples a set of quasi orders.

The function `iita` calls `ob_counter`.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

`ind_gen` for inductive generation procedure using numbers of counterexamples; `iita`, the interface that provides the three inductive item tree analysis methods under one umbrella. See also `DAKS-package` for general information about this package.

Examples

```
ob_counter(pisa)
```

orig_iita

Original Inductive Item Tree Analysis

Description

orig_iita performs the original inductive item tree analysis procedure and returns the corresponding *diff* values.

Usage

```
orig_iita(dataset, A)
```

Arguments

dataset	a required data frame or matrix consisting of binary, 1 or 0, numeric data.
A	a required list of competing quasi orders (surmise relations), for instance obtained from a call to ind_gen .

Details

Original inductive item tree analysis is a data analysis method for deriving knowledge structures (more precisely, surmise relations) from binary data. Details on this procedure can be found in [iita](#). The set of competing quasi orders is passed via the argument A, so any selection set of quasi orders can be used.

The set of competing quasi orders must be a list of objects of the class [set](#). These objects (quasi orders) consist of 2-tuples (i, j) of the class [tuple](#), where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

The data must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments dataset and A are of required types, corr_iita returns a named list of the following components:

diff.value	a vector of the <i>diff</i> values corresponding to the competing quasi orders in A.
error.rate	a vector of the error rates corresponding to the competing quasi orders in A.

Note

The function [iita](#) can be used to perform one of the three inductive item tree analysis procedures (including the original inductive item tree analysis method) selectively. Whereas for the function orig_iita a selection set of competing quasi orders has to be passed via the argument A manually, iita automatically generates a selection set from the data using the inductive generation procedure implemented in [ind_gen](#).

The latter approach using `iita` is common so far, in knowledge space theory, where the inductive data analysis methods have been utilized for exploratory derivations of surmise relations from data. The function `orig_iita`, on the other hand, can be used to select among surmise relations for instance obtained from querying experts or from competing psychological theories.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

`corr_iita` for corrected inductive item tree analysis; `mini_iita` for minimized corrected inductive item tree analysis; `iita`, the interface that provides the three inductive item tree analysis methods under one umbrella; `pop_variance` for population asymptotic variances of *diff* coefficients; `variance` for estimated asymptotic variances of *diff* coefficients; `pop_iita` for population inductive item tree analysis. See also [DAKS-package](#) for general information about this package.

Examples

```
ind <- ind_gen(ob_counter(pisa))
orig_iita(pisa, ind)
```

pattern

Frequencies of Response Patterns and Knowledge States

Description

`pattern` computes the absolute frequencies of the (occurring) response patterns, and optionally, the absolute frequencies of a collection of specified knowledge states in a dataset.

Usage

```
pattern(dataset, n = 5, P = NULL)
```

Arguments

dataset	a required data frame or matrix consisting of binary, 1 or 0, numeric data.
n	an optional numeric, with default $n = 5$, giving the n highest frequencies and their corresponding response patterns to be returned.
P	an optional matrix of ones and zeros giving the knowledge states to be used. The default $P = \text{NULL}$ corresponds to no knowledge states being specified.

Details

This function can be used to retrieve information about how often response patterns and knowledge states occur in a dataset. The argument n refers to response patterns, not knowledge states, and in particular is independent of specifications of the argument P . If `pattern` is called without specifying n explicitly, the response patterns with the five highest frequencies are returned (along with their frequencies). If n is specified, the response patterns with the n highest frequencies are returned (along with their frequencies). If n is larger than the number of different response patterns in the dataset, n is set the number of different response patterns.

The knowledge states are represented as 1/0-patterns and are the rows of the argument matrix P . The matrix P must contain only ones and zeros, which encode whether or not an item belongs to a knowledge state, respectively. If P is not specified, `pattern` only returns information about response patterns (as described previously).

The data must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments `dataset`, n , and P are of required types, `pattern` returns a named list consisting of the following three components:

<code>response.patterns</code>	an array giving the response patterns (with the n highest frequencies) and their absolute frequencies in dataset.
<code>states</code>	a matrix of the knowledge states and their absolute frequencies in dataset. Each row represents a knowledge state, the last column gives the frequencies of the states. If $P = \text{NULL}$, the component <code>states</code> is <code>NULL</code> .
<code>n</code>	a numeric giving the number of response patterns that are returned.

Note

Although `pattern` is intended for use with dichotomous data, it also works with polytomously scored items.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[ob_counter](#) for computation of numbers of counterexamples; [simu](#) for data simulation tool; [iita](#), the interface that provides the three inductive item tree analysis methods under one umbrella. See also [DAKS-package](#) for general information about this package.

Examples

```
pattern(pisa, n = 3)
pattern(pisa)
```

pisa

Programme for International Student Assessment (PISA) Data

Description

The accompanying binary dataset is part of the empirical 2003 Programme for International Student Assessment (PISA) data. It contains the item responses by 340 German students on a 5-item dichotomously scored mathematical literacy test.

Usage

```
pisa
```

Format

The `pisa` data frame consists of 340 rows and 5 columns, representing the response patterns of the students to the test items. Each number, an integer, 1 or 0, encodes a correct or incorrect response, respectively.

Note

The dataset `pisa` was obtained after dichotomizing the original multiple-choice or open format test data. Wording of the test items used in the assessment is not known (not available publicly).

Source

OECD Programme for International Student Assessment (PISA; <http://www.pisa.oecd.org/>)

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

See [DAKS-package](#) for general information about this package.

pop_iita

Population Inductive Item Tree Analysis

Description

pop_iita can be used to perform one of the three inductive item tree analysis algorithms (original, corrected, and minimized corrected) in population quantities (in a known population) selectively.

Usage

```
pop_iita(imp, ce, lg, items, dataset = NULL, A = NULL, v)
```

Arguments

imp	a required object of class <code>set</code> representing the underlying set of implications (assumed to be a quasi order), for instance obtained from a call to <code>iita</code> .
ce	a required numeric giving the probability for a careless error.
lg	a required numeric giving the probability for a lucky guess.
items	a required numeric giving the number of items of the domain taken as basis for imp.
dataset	an optional data frame or matrix consisting of binary, 1 or 0, numeric data. The default <code>dataset = NULL</code> corresponds to no dataset being used.
A	an optional list of competing quasi orders (surmise relations).
v	a required numeric giving the inductive item tree analysis algorithm to be performed, in population quantities; $v = 1$ (minimized corrected), $v = 2$ (corrected), and $v = 3$ (original).

Details

The three inductive item tree analysis algorithms are exploratory methods for extracting quasi orders (surmise relations) from data. In each algorithm, competing binary relations are generated (in the same way for all three versions), and a fit measure (differing from version to version) is computed for every relation of the selection set in order to find the quasi order that fits the data best. In all three algorithms, the idea is to estimate the numbers of counterexamples for each quasi order, and to find, over all competing quasi orders, the minimum value for the discrepancy between the observed and expected numbers of counterexamples. The three data analysis methods differ in their choices of estimates for the expected numbers of counterexamples. For details see [iita](#). The algorithms are described in the paper about the **DAKS** package by *Uenlue and Sargin (2010)*, and in the paper by *Sargin and Uenlue (2009)*.

Compared to [iita](#), the function `pop_iita` implements the three inductive item tree analysis algorithms in population, **not** sample, quantities. The argument `imp` must give a quasi order, and equipped with the error probabilities `ce` and `lg`, it is considered a special case of the basic local independence model (*Doignon and Falmagne, 1999*). The latter then is considered as the underlying population model. If `dataset = NULL` a set of competing quasi orders is constructed based on a population analog of the inductive generation procedure implemented in sample quantities in [ind_gen](#). If a dataset is specified explicitly, that data are used to generate the set of competing quasi orders based on the sample version of the inductive generation procedure.

A set of implications, an object of the class `set`, consists of 2-tuples (i, j) of the class `tuple`, where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

The data (in `dataset`) must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments `imp`, `ce`, `lg`, `items`, `dataset`, `A`, and `v` are of required types, `pop_iita` returns a named list consisting of the following five components:

<code>pop.diff</code>	a vector giving the population <i>diff</i> values corresponding to the (inductively generated) competing quasi orders (subject to selected version that was performed).
<code>pop.matrix</code>	a matrix of all possible response patterns and their corresponding population occurrence probabilities.
<code>error.pop</code>	a vector of the population γ rates corresponding to the (inductively generated) competing quasi orders (subject to selected version that was performed).
<code>selection.set</code>	a list of the (inductively generated) competing quasi orders.
<code>v</code>	the version used; $v = 1$ (minimized corrected), $v = 2$ (corrected), and $v = 3$ (original).

Note

The single careless error `ce` and lucky guess `lg` probabilities are assumed to be constant over all items. The most general case that can be specified thus includes two error probabilities, which are the same for all items.

The sample *diff* coefficients of the three inductive item tree analysis algorithms can be transformed into maximum likelihood estimators, by division through the square of sample size. These transformed *diff* coefficients are considered in population quantities. The γ rates are the algorithms' specific estimates of the postulated response error probability.

Population and estimated asymptotic variances of the maximum likelihood estimators *diff* are implemented in the functions `pop_variance` and `variance`, respectively.

Author(s)

Anatol Sargin, Ali Uenlue

References

Doignon, J.-P. and Falmagne, J.-C. (1999) *Knowledge Spaces*. Berlin, Heidelberg, and New York: Springer-Verlag.

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

`pop_variance` for population asymptotic variances of *diff* coefficients; `variance` for estimated asymptotic variances of *diff* coefficients; `simu` for data simulation tool; `ind_gen` for (sample) inductive generation procedure; `iita`, the interface that provides the three (sample) inductive item tree analysis methods under one umbrella. See also `DAKS-package` for general information about this package.

Examples

```
x <- simu(3, 10000, ce = 0.05, lg = 0.05, delta = 0.12)
y <- iita(x$dataset, v = 2)
z <- pop_iita(x$implications, 0.05, 0.05, 3, x$dataset, v = 2)

## similar sample and population diff values are obtained
(y$diff) / (10000^2)
z
```

pop_variance

Population Asymptotic Variance

Description

`pop_variance` computes the population (exact) asymptotic variances of the maximum likelihood estimators *diff*, assuming a multinomial probability distribution on the set of all response patterns.

Usage

```
pop_variance(pop_matrix, imp, error_pop, v)
```

Arguments

pop_matrix	a required matrix of all possible response patterns and their corresponding population occurrence probabilities, for instance obtained from a call to pop_iita .
imp	a required object of class set representing the set of implications (ought to be a quasi order) for which <i>diff</i> is computed, for instance obtained from a call to pop_iita .
error_pop	a required numeric giving the γ rate to be used for computing <i>diff</i> , for instance obtained from a call to pop_iita .
v	a required numeric giving the inductive item tree analysis algorithm to be performed, in population quantities; $v = 1$ (minimized corrected) and $v = 2$ (corrected).

Details

Subject to the selected version to be performed, `pop_variance` computes the population asymptotic variance of the maximum likelihood estimator *diff*, which here is formulated for the relation specified in `imp` and for the γ rate in `error_pop`. This population variance is obtained using the delta method, which requires calculating the Jacobian matrix of the *diff* coefficient and the inverse of the expected Fisher information matrix for the multinomial distribution with cell probabilities as specified in `pop_matrix`.

A set of implications, an object of the class [set](#), consists of 2-tuples (i, j) of the class [tuple](#), where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

Value

If the arguments `pop_matrix`, `imp`, `error_pop`, and `v` are of required types, `pop_variance` returns a numeric giving the population asymptotic variance of the maximum likelihood estimator *diff* (formulated for the relation in `imp` and the γ rate in `error_pop`).

Note

The current version of the package **DAKS** does not support computing population asymptotic variances for the original inductive item tree analysis algorithm; population asymptotic variances can be calculated only for the corrected and minimized corrected algorithms.

The sample *diff* coefficients of the three inductive item tree analysis algorithms can be transformed into maximum likelihood estimators, by division through the square of sample size. These transformed *diff* coefficients are considered in population quantities. The γ rates are the algorithms’ specific estimates of the postulated response error probability.

Estimated asymptotic variances of the maximum likelihood estimators *diff* are implemented in the function [variance](#).

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[variance](#) for estimated asymptotic variances of *diff* coefficients; [pop_iita](#) for population inductive item tree analysis; [ind_gen](#) for (sample) inductive generation procedure; [iita](#), the interface that provides the three (sample) inductive item tree analysis methods under one umbrella. See also [DAKS-package](#) for general information about this package.

Examples

```
## Not run:
x <- simu(5, 100, 0.05, 0.05, delta = 0.15)
y <- pop_iita(x$implications, 0.05, 0.05, 5, x$dataset, v = 2)
pop_variance(y$pop.matrix,
             y$selection.set[[which(y$pop.diff == min(y$pop.diff))]],
             y$error.pop[which(y$pop.diff == min(y$pop.diff))], v = 2)

## End(Not run)
```

print.iita

Print Method for Objects of Class iita

Description

S3 method to print objects of class `iita`.

Usage

```
## S3 method for class 'iita'
print(x, ...)
```

Arguments

`x` a required object of class `iita`, obtained from a call to the function `iita`.
`...` further arguments to be passed to or from other methods.

Details

Prints the main results from inductive item tree analysis algorithms.

Value

If the argument `x` is of required type, `print.iita` prints the set of implications.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[iita](#), the interface that provides the three (sample) inductive item tree analysis methods under one umbrella. See also [DAKS-package](#) for general information about this package.

print.pat

Print Method for Objects of Class pat

Description

S3 method to print objects of class `pat`.

Usage

```
## S3 method for class 'pat'
print(x, ...)
```

Arguments

`x` a required object of class `pat`, obtained from a call to the function [pattern](#).
`...` further arguments to be passed to or from other methods.

Details

Prints the main results from inductive item tree analysis algorithms.

Value

If the argument `x` is of required type, `print.pat` prints the response patterns with the n highest frequencies in the dataset and optionally returns the absolute frequencies of specified knowledge states in the dataset.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An **R** package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

See also [DAKS-package](#) for general information about this package.

print.popiita

Print Method for Objects of Class popiita

Description

S3 method to print objects of class popiita.

Usage

```
## S3 method for class 'popiita'  
print(x, ...)
```

Arguments

x a required object of class popiita, obtained from a call to the function [pop_iita](#).
... further arguments to be passed to or from other methods.

Details

Prints the main results from inductive item tree analysis algorithms in population values.

Value

If the argument x is of required type, print.popiita prints a vector of population *diff* values, a vector of population error rates, and the quasi order with minimum population *diff* value.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[pop_iita](#). See also [DAKS-package](#) for general information about this package.

print.summpopiita	<i>Print Method for Objects of Class summpopiita</i>
-------------------	--

Description

S3 method to print objects of class summpopiita.

Usage

```
## S3 method for class 'summpopiita'  
print(x, ...)
```

Arguments

x	a required object of class summpopiita, obtained from a call to the function summary.popiita .
...	further arguments to be passed to or from other methods.

Details

Prints the main results from inductive item tree analysis algorithms in population values.

Value

If the argument x is of required type, `print.summpopiita` prints a vector of population *diff* values, a vector of population error rates, the population matrix, and the obtained selection set.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[pop_iita](#). See also [DAKS-package](#) for general information about this package.

print.ztest

Print Method for Objects of Class ztest

Description

S3 method to print objects of class ztest.

Usage

```
## S3 method for class 'ztest'
print(x, ...)
```

Arguments

x a required object of class ztest, obtained from a call to the function [z_test](#).
 ... further arguments to be passed to or from other methods.

Details

Prints the main results from inductive item tree analysis algorithms.

Value

If the argument x is of required type, print.ztest prints the Z- and p-value, the alternative hypothesis, the confidence interval, and the sample estimates.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376– 392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[z_test](#), the function for performing a Z-test. See also [DAKS-package](#) for general information about this package.

 simu

Data and Quasi Order Simulation Tool

Description

simu can be used to simulate binary, of type 1/0, data using a basic local independence model. The number of items, the sample size, and two parameters for the careless error and lucky guess probabilities can be set explicitly. The underlying combinatorial structure used for simulating the data can either be specified manually or is generated randomly.

Usage

```
simu(items, size, ce, lg, imp = NULL, delta)
```

Arguments

items	a required numeric giving the number of items of the domain taken as basis for the simulation.
size	a required numeric giving the number of response patterns to be simulated (the sample size).
ce	a required numeric giving the probability for a careless error.
lg	a required numeric giving the probability for a lucky guess.
imp	an optional object of class set representing the underlying set of implications (assumed to be a quasi order) used for simulating the data, for instance obtained from a call to iita . The default <code>imp = NULL</code> corresponds to generating the quasi order used for simulating the data randomly.
delta	a required (if <code>imp = NULL</code>) numeric giving the probability for adding an item pair to the randomly generated quasi order (reflexive pairs are always included a priori).

Details

The function `simu` simulates data using a special case of the basic local independence model, which is a fundamental restricted latent class model in knowledge space theory (*Doignon and Falmagne, 1999*). The single careless error `ce` and lucky guess `lg` probabilities are assumed to be constant over all items. The most general case that can be specified thus includes two error probabilities at each item, the same two rates for all items. The general form of the basic local independence model allows for varying careless error and lucky guess rates from item to item (not identifiable in general, however).

If a quasi order is specified in `imp` explicitly, Birkhoff's theorem is used to derive its corresponding quasi ordinal knowledge space, which is equipped with the error probabilities `ce` and `lg` to give the

basic local independence model used for simulating the data. If `imp = NULL`, the underlying quasi order is generated randomly as follows. All reflexive pairs are added to the relation. The constant specified in `delta` is utilized as the probability for adding each of the remaining non-reflexive item pairs to the relation. The transitive closure of this relation is computed, and the resulting quasi order is then the relation underlying the simulation.

A set of implications, an object of the class `set`, consists of 2-tuples (i, j) of the class `tuple`, where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

The simulated dataset contains only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments `items`, `size`, `ce`, `lg`, `imp`, and `delta` are of required types, `simu` returns a named list consisting of the following three components:

<code>dataset</code>	a matrix of binary, 1 or 0, numeric data.
<code>implications</code>	an object of class <code>set</code> representing the underlying set of implications (assumed to be a quasi order) used for simulating the data. If <code>imp = NULL</code> , this is the quasi order that was randomly generated; otherwise identical to the set of implications specified in the argument <code>imp</code> .
<code>states</code>	a matrix consisting of ones or zeros (the quasi ordinal knowledge space), in which each row represents the 1/0-pattern of a knowledge state. This is the knowledge structure corresponding to the set of implications specified in <code>implications</code> .

Note

To pass a quasi order as the argument `imp` to `simu` it may be more convenient to transform from knowledge states to implications using the function `state2imp`.

The probability specified in `delta` does not necessarily correspond to the ratio of implications in the randomly generated quasi order, because the transitive closure is formed after having added item pairs. In *Sargin and Uenlue (2009)* a normal sampling scheme for drawing `delta` values using $\mu = 0.16$ and $\sigma = 0.06$ for nine items has been proposed. This sampling scheme provides far better representative samples of quasi orders than sampling `delta` values uniformly from the unit interval.

Author(s)

Anatol Sargin, Ali Uenlue

References

- Doignon, J.-P. and Falmagne, J.-C. (1999) *Knowledge Spaces*. Berlin, Heidelberg, and New York: Springer-Verlag.
- Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.
- Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[state2imp](#) for transformation from knowledge states to implications; [imp2state](#) for transformation from implications to knowledge states; [pop_iita](#) for population inductive item tree analysis; [iita](#), the interface that provides the three (sample) inductive item tree analysis methods under one umbrella. See also [DAKS-package](#) for general information about this package.

Examples

```
simu(7, 20, 0.1, 0.1, delta = 0.15)
```

state2imp	<i>Transformation from Knowledge States to Implications</i>
-----------	---

Description

state2imp transforms a set of knowledge states (ought to be a quasi ordinal knowledge space) to the corresponding set of implications (the surmise relation).

Usage

```
state2imp(P)
```

Arguments

P a required matrix of ones and zeros giving the knowledge states to be used. Each row represents the 1/0-pattern of a knowledge state.

Value

If the argument P is of required type, state2imp returns an object of class [set](#) (the surmise relation) representing the set of implications.

Note

For any set of knowledge states the returned binary relation is a surmise relation. In case of a quasi ordinal knowledge space this is Birkhoff's theorem. For details refer to *Doignon and Falmagne (1999, Theorem 1.49)*.

A set of implications, an object of the class [set](#), consists of 2-tuples (i, j) of the class [tuple](#), where a 2-tuple (i, j) is interpreted as 'mastering item j implies mastering item i .'

Author(s)

Anatol Sargin, Ali Uenlue

References

Doignon, J.-P. and Falmagne, J.-C. (1999) *Knowledge Spaces*. Berlin, Heidelberg, and New York: Springer-Verlag.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[imp2state](#) for transformation from implications to knowledge states. See also [DAKS-package](#) for general information about this package.

Examples

```
## an arbitrary matrix of knowledge states is defined
x <- matrix(0, nrow = 5, ncol = 3)
x[1, ] <- c(0, 0, 0)
x[2, ] <- c(0, 0, 1)
x[3, ] <- c(0, 1, 0)
x[4, ] <- c(0, 1, 1)
x[5, ] <- c(1, 1, 1)

state2imp(x)
```

summary.iita

Summary Method for Objects of Class iita

Description

S3 method to summarize objects of class `iita`.

Usage

```
## S3 method for class 'iita'
summary(object, ...)
```

Arguments

`object` a required object of class `iita`, obtained from a call to the function `iita`.
`...` further arguments to be passed to or from other methods.

Details

Summarizes the main results from inductive item tree analysis algorithms.

Value

If the argument `object` is of required type, `summary.iita` returns the vector of *diff* values, the quasi order, the error rate, and the index in the selection set of the quasi order with minimum *diff* value.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[iita](#), the interface that provides the three (sample) inductive item tree analysis methods under one umbrella. See also [DAKS-package](#) for general information about this package.

summary.popiita

Summary Method for Objects of Class popiita

Description

S3 method to summarize objects of class `popiita`.

Usage

```
## S3 method for class 'popiita'
summary(object, ...)
```

Arguments

`object` a required object of class `popiita`, obtained from a call to the function [pop_iita](#).
`...` further arguments to be passed to or from other methods.

Details

Summarizes the main results from inductive item tree analysis algorithms in population values.

Value

If the argument `object` is of required type, `summary.popiita` returns the results obtained from [pop_iita](#).

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[pop_iita](#). See also [DAKS-package](#) for general information about this package.

variance	<i>Estimated Asymptotic Variance</i>
----------	--------------------------------------

Description

`variance` computes estimated asymptotic variances of the maximum likelihood estimators *diff* from data, assuming a multinomial probability distribution on the set of all response patterns.

Usage

```
variance(dataset, imp, v)
```

Arguments

<code>dataset</code>	a required data frame or matrix consisting of binary, 1 or 0, numeric data.
<code>imp</code>	a required object of class <code>set</code> representing the set of implications (ought to be a quasi order) for which <i>diff</i> is computed, for instance obtained from a call to <code>simu</code> .
<code>v</code>	a required numeric giving the inductive item tree analysis algorithm to be performed; <code>v = 1</code> (minimized corrected) and <code>v = 2</code> (corrected).

Details

Subject to the selected version to be performed, `variance` computes a consistent estimator for the population asymptotic variance of the maximum likelihood estimator *diff*, which here is formulated for the relation specified in `imp` and for the data in `dataset`. This estimated asymptotic variance is obtained using the delta method, which requires calculating the Jacobian matrix of the *diff* coefficient and the inverse of the expected Fisher information matrix for the multinomial distribution on the set of all response patterns. In the expression for the exact asymptotic variance, the true parameter vector of multinomial probabilities is estimated by its corresponding maximum likelihood estimate (vector of the relative frequencies of the response patterns).

A set of implications, an object of the class `set`, consists of 2-tuples (i, j) of the class `tuple`, where a 2-tuple (i, j) is interpreted as ‘mastering item j implies mastering item i .’

The data must contain only ones and zeros, which encode solving or failing to solve an item, respectively.

Value

If the arguments `dataset`, `imp`, and `v` are of required types, `variance` returns a numeric giving the estimated asymptotic variance of the maximum likelihood estimator *diff* (formulated for the relation in `imp` and the data in `dataset`).

Note

The current version of the package **DAKS** does not support computing estimated asymptotic variances for the original inductive item tree analysis algorithm; population asymptotic variances can be estimated only for the corrected and minimized corrected algorithms.

The two types of estimators for the population asymptotic variances of the *diff* coefficients obtained using the expected Fisher information matrix on the one hand, and the observed Fisher information matrix on the other, yield the same result, in the case of the multinomial distribution. Since computation based on expected Fisher information is faster, this is implemented in `variance`.

The sample *diff* coefficients of the three inductive item tree analysis algorithms can be transformed into maximum likelihood estimators, by division through the square of sample size. These transformed *diff* coefficients are considered in sample and population quantities.

Population (exact) asymptotic variances of the maximum likelihood estimators *diff* are implemented in the function `pop_variance`.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

`pop_variance` for population asymptotic variances of *diff* coefficients; `pop_iita` for population inductive item tree analysis; `iita`, the interface that provides the three (sample) inductive item tree analysis methods under one umbrella; `z_test` for one- and two-sample Z-tests. See also `DAKS-package` for general information about this package.

Examples

```
x <- simu(5, 100, 0.05, 0.05, delta = 0.15)
variance(x$dataset, x$implications, v = 2)
```

z_test

*One- and Two-Sample Z-Tests for diff Values***Description**

z_test performs one- and two-sample Z-tests for the *diff* values.

Usage

```
z_test(dataset, imp, imp_alt = NULL,
        alternative = c("two.sided", "less", "greater"), mu = 0,
        conf.level = 0.95, v)
```

Arguments

dataset	a required data frame or matrix consisting of binary, 1 or 0, numeric data.
imp	a required object of class <code>set</code> representing the set of implications (ought to be a quasi order).
imp_alt	an optional set of implications, representing the alternative quasi order.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "less", or "greater".
mu	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
conf.level	confidence level of the interval.
v	a required numeric giving the inductive item tree analysis algorithm to be performed; $v = 1$ (minimized corrected) and $v = 2$ (corrected).

Details

This function performs a Z-test for the *diff* values of one or two quasi orders.

Value

If the arguments are of required types, z_test returns a named list consisting of the following seven components:

Z.value	the value of the Z-statistic.
p.value	the p-value for the test.
conf	a confidence interval for the mean appropriate to the specified alternative hypothesis.

diff_value	the corresponding <i>diff</i> values for the used quasi orders according to the specified method.
alternative	a character string specifying the alternative hypothesis.
mu	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
conf.level	the level of the confidence interval.

Note

The current version of the package **DAKS** does not support performing a Z-test for the original inductive item tree analysis algorithm.

Author(s)

Anatol Sargin, Ali Uenlue

References

Sargin, A. and Uenlue, A. (2009) Inductive item tree analysis: Corrections, improvements, and comparisons. *Mathematical Social Sciences*, **58**, 376–392.

Uenlue, A. and Sargin, A. (2010) **DAKS**: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, **37**(2), 1–31. URL <http://www.jstatsoft.org/v37/i02/>.

See Also

[iita](#), the interface that provides the three (sample) inductive item tree analysis methods under one umbrella; [variance](#) for estimated asymptotic variances of *diff* coefficients. See also [DAKS-package](#) for general information about this package.

Examples

```
sel_set<-ind_gen(ob_counter(pisa[, 1:3]))
z_test(pisa[, 1:3], sel_set[[2]], sel_set[[3]], v = 1)
```

Index

- *Topic **datagen**
 - simu, 27
- *Topic **datasets**
 - pisa, 17
- *Topic **distribution**
 - simu, 27
- *Topic **graphs**
 - hasse, 5
- *Topic **hplot**
 - hasse, 5
- *Topic **htest**
 - pop_iita, 18
 - pop_variance, 20
 - print.iita, 22
 - print.pat, 23
 - print.popiita, 24
 - print.summpopiita, 25
 - print.ztest, 26
 - summary.iita, 30
 - summary.popiita, 31
 - variance, 32
 - z_test, 34
- *Topic **manip**
 - corr_iita, 3
 - iita, 6
 - imp2state, 8
 - ind_gen, 9
 - mini_iita, 11
 - ob_counter, 12
 - orig_iita, 14
 - pattern, 15
 - pop_iita, 18
 - print.iita, 22
 - print.pat, 23
 - print.popiita, 24
 - print.summpopiita, 25
 - print.ztest, 26
 - state2imp, 29
 - summary.iita, 30
 - summary.popiita, 31
 - variance, 32
 - z_test, 34
- *Topic **math**
 - imp2state, 8
 - pop_variance, 20
 - print.iita, 22
 - print.pat, 23
 - print.popiita, 24
 - print.summpopiita, 25
 - print.ztest, 26
 - simu, 27
 - state2imp, 29
 - summary.iita, 30
 - summary.popiita, 31
 - variance, 32
 - z_test, 34
- *Topic **models**
 - corr_iita, 3
 - iita, 6
 - imp2state, 8
 - ind_gen, 9
 - mini_iita, 11
 - orig_iita, 14
 - pop_iita, 18
 - pop_variance, 20
 - print.iita, 22
 - print.pat, 23
 - print.popiita, 24
 - print.summpopiita, 25
 - print.ztest, 26
 - simu, 27
 - state2imp, 29
 - summary.iita, 30
 - summary.popiita, 31
 - variance, 32
 - z_test, 34
- *Topic **multivariate**
 - pop_variance, 20

- print.iita, 22
- print.pat, 23
- print.popiita, 24
- print.summpopiita, 25
- print.ztest, 26
- simu, 27
- summary.iita, 30
- summary.popiita, 31
- variance, 32
- z_test, 34
- *Topic **package**
 - DAKS-package, 2
- *Topic **print**
 - hasse, 5
- *Topic **univar**
 - corr_iita, 3
 - iita, 6
 - mini_iita, 11
 - ob_counter, 12
 - orig_iita, 14
 - pattern, 15
 - pop_iita, 18
 - pop_variance, 20
 - print.iita, 22
 - print.pat, 23
 - print.popiita, 24
 - print.summpopiita, 25
 - print.ztest, 26
 - summary.iita, 30
 - summary.popiita, 31
 - variance, 32
 - z_test, 34
- *Topic **utilities**
 - imp2state, 8
 - ob_counter, 12
 - pattern, 15
 - pop_variance, 20
 - print.iita, 22
 - print.pat, 23
 - print.popiita, 24
 - print.summpopiita, 25
 - print.ztest, 26
 - simu, 27
 - state2imp, 29
 - summary.iita, 30
 - summary.popiita, 31
 - variance, 32
 - z_test, 34
- corr_iita, 2, 3, 7, 8, 12, 15
- DAKS-package, 2
- hasse, 2, 5
- iita, 2, 4–6, 6, 7, 8, 10–15, 17–20, 22, 23, 27, 29–31, 33, 35
- imp2state, 2, 8, 29, 30
- ind_gen, 2–4, 7, 8, 9, 10, 11, 13, 14, 19, 20, 22
- mini_iita, 2, 4, 7, 8, 11, 15
- ob_counter, 2, 10, 12, 13, 17
- orig_iita, 2, 4, 7, 8, 12, 14
- pattern, 2, 15, 23
- pisa, 3, 17
- pop_iita, 2, 5, 8, 12, 15, 18, 21, 22, 24–26, 29, 31–33
- pop_variance, 2, 4, 8, 12, 15, 20, 20, 33
- print.iita, 2, 22
- print.pat, 2, 23
- print.popiita, 2, 24
- print.summpopiita, 2, 25
- print.ztest, 2, 26
- set, 4, 5, 7–11, 14, 18, 19, 21, 27–29, 32–34
- simu, 2, 17, 20, 27, 32
- state2imp, 2, 9, 28, 29, 29
- summary.iita, 2, 30
- summary.popiita, 2, 25, 31
- tuple, 4, 5, 7, 9–11, 14, 19, 21, 28, 29, 33
- variance, 2, 5, 8, 12, 15, 20–22, 32, 35
- z_test, 2, 8, 10, 26, 27, 33, 34