Package 'klausuR'

April 4, 2022

Type Package

Title Multiple Choice Test Evaluation

Description A set of functions designed to quickly generate results of a multiple choice test. Generates detailed global results, lists for anonymous feedback and personalised result feedback (in LaTeX and/or PDF format), as well as item statistics like Cronbach's alpha or disciminatory power. 'klausuR' also includes a plugin for the R GUI and IDE RKWard, providing graphical dialogs for its basic features. The respective R package 'rkward' cannot be installed directly from a repository, as it is a part of RKWard. To make full use of this feature, please install RKWard from <https://rkward.kde.org> (plugins are detected automatically). Due to some restrictions on CRAN, the full package sources are only available from the project homepage.

Depends R (>= 2.9.0), xtable

Imports methods, graphics, tools, utils, stats, grDevices, psych

Enhances rkward

URL https://reaktanz.de/?c=hacking&s=klausuR

BugReports https://github.com/unDocUMeantIt/klausuR/issues

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

Version 0.12-14

Date 2022-04-04

RoxygenNote 7.1.2

Collate '00_class_01_klausuR.test.R' '00_class_02_klausuR.R' '00_class_03_klausuR.answ.R' '00_class_04_klausuR.mult.R' '01_method_plot.klausuR.R' '01_method_show.klausuR.R' '01_method_sort.klausuR.R' 'compare.R' 'grand.table.R' 'klausuR-internal.R' 'klausuR-internal_LaTeX_report.R' 'klausuR-package.R' 'klausur.R' 'klausur.data.R' 'klausur.gen.R' 'klausur.gen.corr.R' 'klausur.gen.marks.R' 'klausur.mufo.R' 'klausur.report.R' 'nret.rescale.R' 'nret.translator.R'

NeedsCompilation no

Author m.eik michalke [aut, cre]

Maintainer m.eik michalke <meik.michalke@hhu.de>

Repository CRAN

Date/Publication 2022-04-04 21:40:12 UTC

R topics documented:

klausuR-package . . . 2
antworten . . . 3
antworten.mufo . . . 4
compare . . . 5
grand.table . . . 7
klausur . . . 8
klausuR,-class . . . 13
klausuR.answ,-class . . . 14
klausur.data . . . 15
klausur.gen . . . 18
klausur.gen.corr . . . 19
klausur.gen.marks . . . 20
klausur.mufo . . . 21
klausuR.mult-class . . . 24
klausur.report . . . 24
klausuR.test,-class . . . 27
nret.rescale . . . 28
nret.translator . . . 29
plot . . . 31
show,klausuR-method . . . 32
sort . . . 33
spss.data . . . 34

Index **35**

`klausuR-package` *Multiple Choice Test Evaluation*

Description

A set of functions designed to quickly generate results of a multiple choice test. Generates detailed global results, lists for anonymous feedback and personalised result feedback (in LaTeX and/or PDF format), as well as item statistics like Cronbach's alpha or disciminatory power. 'klausuR' also includes a plugin for the R GUI and IDE RKWard, providing graphical dialogs for its basic features. The respective R package 'rkward' cannot be installed directly from a repository, as it is a part of RKWard. To make full use of this feature, please install RKWard from <https://rkward.kde.org> (plugins are detected automatically). Due to some restrictions on CRAN, the full package sources are only available from the project homepage.

Details

The DESCRIPTION file:

Package:	klausuR
Type:	Package
Version:	0.12-14
Date:	2022-04-04
Depends:	R (>= 2.9.0), xtable
Enhances:	rkward
Encoding:	UTF-8
License:	GPL (>= 3)
LazyLoad:	yes
URL:	https://reaktanz.de/?c=hacking&s=klausuR

Author(s)

NA

Maintainer: NA

See Also

Useful links:

- https://reaktanz.de/?c=hacking&s=klausuR
- Report bugs at https://github.com/unDocUMeantIt/klausuR/issues

---

`antworten` *Sample dataset: antworten*

Description

Sample data of a multiple choice test.

Usage

```
data(antworten)
```

Format

A data.frame with 20 observations of the following 37 variables.

`No` Serial number for each subject (numeric vector)

`Name` Family name (character vector)

`FirstName` First name (character vector)

`MatrNo` Matriculation number (numeric vector)

`Pseudonym` Pseudonym for anonymous feedback (character)

`Item01` **to** `Item30` given answers, only one answer per item (numeric vector)

`Item31` **to** `Item32` given answers, several answers per item (numeric vector)

Details

This data was anonymised and Items 31 and 32 modified, to demonstrate items with multiple answers as well.

Source

The data was taken from a final exam test in differential psychology at the Institute for Experimental Psychology of the Heinrich-Heine-University Düsseldorf, summer term 2009.

Examples

```
data(antworten)
```

`antworten.mufo` *Sample dataset: antworten.mufo*

Description

Sample data of a multiple choice test with two test forms.

Usage

```
data(antworten.mufo)
```

Format

A data.frame with 20 observations of the following 38 variables.

`No` Serial number for each subject (numeric vector)

`Name` Family name (character vector)

`FirstName` First name (character vector)

`MatrNo` Matriculation number (numeric vector)

`Pseudonym` Pseudonym for anonymous feedback (character)

`Item01` **to** `Item30` given answers, only one answer per item (numeric vector)

`Item31` **to** `Item32` given answers, several answers per item (numeric vector)

`Form` Test form (character vector)

And a matrix called `corr.key` with indices for all items in the two test forms.

Details

The data is actually the same as `antworten`. But in this object the items of the last 10 test subjects were reversed, hence this is test form "b". As a convenience, a matrix called `corr.key` is also included. This data can be used to test `klausur.mufo`.

Source

The data was taken from a final exam test in differential psychology at the Institute for Experimental Psychology of the Heinrich-Heine-University Düsseldorf, summer term 2009.

Examples

```
data(antworten.mufo)
```

`compare` *Comparison of data sets*

Description

The function `compare` will take two data.frames (or objects of class `klausuR.answ-class`) and compare them for equality. This is useful to check for typos before you calculate the results with `klausur`. If you need to type in the given answers by hand, errors easily occur, so it is advisable to input all data at least twice (perhaps by different persons) and check for differences with this function, which can then be corrected by looking up the original answer in the test.

Usage

```
compare(
  set1,
  set2,
  select = NULL,
  ignore = NULL,
  new.set = FALSE,
  rename = c(),
  trim = FALSE,
  id = list(No = "No", Name = c("FirstName", "Name"))
)
```

Arguments

`set1, set2`	The data sets to be compared. Can be two data.frames or objects of class `klausuR.answ-class`. If the latter, their slots `id` and `items` will be compared.
`select`	A vector with variables that should be compared, all others are omitted. At least all the values given in `id` are needed for the output! If `NULL`, all variables are examined.
`ignore`	A vector with variables that should be dropped from both sets. See also `select`.

new.set	Logical. If TRUE, a data.frame of the compared sets is returned, with all unequal cells set to NA.
rename	A named vector defining if variables in set1 and set2 need to be renamed into the klausuR name scheme. Accepts elements named No, Name, FirstName, MatrNo, Pseudonym and Form. The values of these elements represent the variable names of the input data.
trim	Logical. Indicates wheter whitespace in character variables should be trimmed.
id	A named list of character vectors to help identify differing cases in the input data. The element names of this list will become column names in the generated output table, their values define the respective column names of the input data. If a value has more than one element, they will be collapsed into one string for the output.

Details

If you don't want to compare all variables but only a subset, you can use the select option (see examples below). But be careful with this, at least all the values given in id are needed to produce the output table.

If new.set=TRUE, a new data.frame will be returned, that is identical in both sets compared, but all dubious values will be replaced by NA.

Value

If new.set=FALSE, a data.frame of the differences, if found (if not, just a message is returned). Otherwise returns a combined data.frame (see details).

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

klausur

Examples

```
## Not run:
data(antworten)

# create some differences
antworten2 <- antworten[-3, -7]
antworten2[4,6] <- NA
antworten2[8,8:10] <- antworten2[8,8:10] + 1

# default comparison
compare(antworten, antworten2)

# compare only variables 1 to 12
compare(antworten, antworten2, select=c(1:12))
```

```
# omit variables 3 to 8 and create a new set called "antworten.comp"
# from the results
antworten.comp <- compare(antworten, antworten2, select=-c(3:8), new.set=TRUE)

## End(Not run)
```

`grand.table` *Export results to a table*

Description

Try this function to combine results from an evaluated test into a matrix and export it to a table, e.g. to import in other software products. It can be particularily helpful for ET/NRET coded tests, if you want to compare the results of different valid scoring functions.

Usage

```
grand.table(
  NR.res = NULL,
  NRET.res = NULL,
  NRETp.res = NULL,
  ET.res = NULL,
  rescale = TRUE,
  file = NULL,
  csv2 = TRUE,
  encoding = "CP1252",
  ...
)
```

Arguments

`NR.res`	An object of class `klausuR` which was evaluated according to the NR scoring policy. Usual MC tests must be given as `NR.res`, too.
`NRET.res`	An object of class `klausuR` which was evaluated according to the NRET scoring policy.
`NRETp.res`	An object of class `klausuR` which was evaluated according to the NRET+ scoring policy.
`ET.res`	An object of class `klausuR` which was evaluated according to the ET scoring policy.
`rescale`	Logical, whether ET/NRET scaled results should be rescaled by `nret.rescale`.
`file`	A character string giving a file name to save to. If `NULL`, no file will be written.
`csv2`	Logical. If `FALSE`, `write.csv` will be used instead of `write.csv2`.
`encoding`	Encoding of the exported table.
`...`	Additional options for `write.csv`/`write.csv2`.

Value

A data.frame.

Note

For obvious reasons At least one of `NR.res`, `NRET.res`, `NRETp.res` or `ET.res`, or any combination of those, must be specified.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

`klausur` *Evaluate multiple choice tests*

Description

The function `klausur` expects an object of class `klausuR.answ-class`, containing some identification data on all subjects and their answers to the test items, a vector with the correct answers, and optionally a vector with marks assigned to the points achieved. It will compute global test results as well as some item analysis (including Cronbach's alpha, discriminatory power and Lienert's selection index of the test items), and anonymous feedback for the test subjects.

Usage

```
klausur(
  data,
  marks = NULL,
  mark.labels = NULL,
  items = NULL,
  wght = NULL,
  score = "solved",
  matn = NULL,
  na.rm = TRUE,
  cronbach = TRUE,
  item.analysis = TRUE,
  sort.by = "Name",
  maxp = NULL
)
```

Arguments

`data`	An object of class `klausuR.answ-class`.
`marks`	A vector assigning marks to points achieved (see details). Alternatively, set it to `"suggest"` to let `klausur.gen.marks` calculate suggestions under the assumption of normal distribution. If `NULL`, this value must be set in the `data` object.

mark.labels	If marks="suggest", use these as the marks you want to give.
items	Indices of a subset of variables in data to be taken as items.
wght	A vector with weights for each item (named also according to Item###). If NULL, the value from the data object will be used.
score	Specify the scoring policy, must be one of "solved" (default), "partial", "liberal", "pick-n", "NR", "ET", "NRET", or "NRET+".
matn	A matriculation number of a subject, to receive detailed results for that subject.
na.rm	Logical, whether cases with NAs should be ignored in data. Defaults to TRUE.
cronbach	Logical. If TRUE, Cronbach's alpha will be calculated.
item.analysis	Logical. If TRUE, some usual item statistics like difficulty, discriminatory power and distractor analysis will be calculated.
sort.by	A character string naming the variable to sort the results by. Set to c() to skip any re-ordering. If cronbach is TRUE, too, it will include the alpha values if each item was deleted.
maxp	Optional numeric value, if set will be forced as the maximum number of points achievable. This should actually not be needed, if your test has no strange errors. But if for example it later turns out you need to adjust one item because it has two instead of one correct answers, this option can become handy in combination with "partial" scoring and item weights.

Details

For details on the ecpected data structure refer to klausur.data,

Scoring functions

In combination with multiple (correct) answers for certain items you can specify one of six scoring policies via the score-parameter. If you set it to something othe than "solved", as the names may suggest you **allow partially given answers** under the condition that the test subject didn't check more alternatives than there are correct ones (that is, if you checked four alternatives where three correct ones were possible, you're out):

- "solved" Multiple Choice: Check all correct alternatives. This is the default, which means that one will only get any points for an item if the answer was 100% correct (that is, all or nothing).
- "partial" Multiple Choice: Check all correct alternatives, allow partially given answers, but none of the distractors must be checked.
- "liberal" Multiple Choice: Check all correct alternatives, allow partially given answers, even distractors can be checked.
- "pick-n" Multiple Choice: Check all correct alternatives, allow partially given answers, even distractors can be checked. Difference to "liberal" is that you will also get points for unchecked distractors.
- "ET" Elimination Testing: In contrast to the usual MC procedure, eliminate/strike all *wrong* alternatives.
- "NRET" Number Right Elimination Testing: Like ET+MC, eliminate/strike all *wrong* alternatives *and* check the correct one.

- "NRET+" Number Right Elimination Testing, more strict: Like NRET, but if more alternatives are checked right than there are right anwers, it will automatically yield to 0 points for that item.
- "NR" Number Right: The usual MC scoring, but works with ET/NRET data. It was implemented for completeness, e.g. to compare results of different scoring techniques.

An example for "solved", "partial" and "liberal": If an item has five answer alternatives, the correct answer is "134" and a subject checked "15", "solved" will give no point (because "15" is not equal to "134"), as will "partial" (because "5" is wrong), but "liberal" will give 1/3 (because "1" is correct), and "pick-n" will give 2/5 (because "1" was correctly checked and "2" correctly unchecked).

(Number Right) Elimination Testing

Note that "ET", "NRET"/"NRET+" and "NR" will disable wght as of now, and **need the data in different format** than the other scoring functions (see klausur.data for details). klausur will evaluate each answer individually and sum up the points for each item. The alternative-wise evaluations will be documented in the trfls slot of the results. Therefore, in these cases that matrix is not boolean, but more complex. For each item and each subject, a character string represents the evaluated answer alternatives, with the following elements:

- P True positive: Alternative was checked as right and is right. *Points: +1 (+ constant)*
- p False positive: Alternative was checked as right but is wrong. *Points: 0 (+ constant)*
- N True negative: Alternative was checked as wrong and is wrong. *Points: +1 (+ constant)*
- n False negative: Alternative was checked as wrong but is right. *Points: -(alternatives-1) (+ constant)*
- 0 Missing: Alternative wasn't checked at all. *Points: 0 (+ constant)*
- * Error: Alternative was checked both wrong and right. *Points NRET+: 0 (+ constant); NR scores 1 point if this was the correct alternative, ET 1 point if it hit a wrong one, and NRET sums up the points for both the positive and negative answer (all + constant)*

An example: If we have an item with four alternatives, and the third one is right (i.e., "--+-"), and a test subject considered the first alternative to be correct and eliminated all others (i.e., "+---"), it would be evaluated as "pNnN", that is 0+1-3+1=-1 point, not considering the constant. As you can see, it would be possible to end up with a negative sum of points. If you consider how in the end a mark will be assigned to the achieved points, this would be a problem, because a vactor cannot have negative indices. To circumvent this issue, klausuR automatically adds a constant to all results, so that the worst possible result is not negative but 0. This constant is simply (alternatives-1), i.e. 3 for the example. In other words, if our test had 10 such items, the results minus 30 would be equivalent to scoring without that constant. You can use nret.rescale to remove the constant from the results afterwards.

Marks

The **assigned marks** are expected to be in a certain format as well (see klausur.data for details), as long as you don't want klausur to suggest them itself. If you want to let klausuR make a suggestion, set marks="suggest", and klausur.gen.marks kicks in and takes either the mark.labels you have defined here or will ask you step by step. See the documentation of that function for details. To see the suggested result in detail, have a look at the slot marks of the returned object.

To calculate Cronbach's alpha and item analysis methods from the package psych are used. Lienert's selction index ("Selektionskennwert") aims to consider both discriminatory power (correlation of an item with the test results) and difficulty to determine the quality of an item. It is defined as

$$S = \frac{r_{it}}{2 \times \sqrt{Difficulty \times (1 - Difficulty)}}$$

Item analysis also includes item discrimination.

Value

An object of class klausuR-class with the following slots.

results	A data.frame with global results
answ	A data.frame with all given answers
corr	A vector with the correct answers
wght	A vector with the weights of items
points	A data.frame with resulting points given for the answers
marks	A vector with assignments of marks to achieved score
marks.sum	A more convenient matrix with summary information on the defined marks
trfls	A data.frame of TRUE/FALSE values, whether a subject was able to solve an item or not
anon	A data.frame for anonymous feedback
mean	A table with mean, median and quartiles of the test results
sd	Standard deviation of the test results
cronbach	Internal consistency, a list of three elements "alpha", "ci" (confidence interval 95%) and "deleted" (alpha if item was removed)
item.analysis	A data.frame with information on difficulty, discriminatory power, discriminant factor and Lienert's selection index of all items.
distractor.analysis	A list with information on the selected answer alternatives for each individual item (only calculated if item.analysis=TRUE). Also lists the discriminatory power of each alternative, being the point-biserial (a.k.a Pearson) correlation of it with the global outcome.
misc	Anything that was stored in the misc slot of the input data.

Not all slots are shown by default (refer to show and plot).

Note

klausur allows some tweaks that are probably not as useful as they seem. For instance, having items with more than one correct answer doesn't necessarily yield more diagnostic information, allowing for those being answered partially adds to that, and binding marks blindly to a normal distribution can give quite unfair test results! In addition, please do **always check a sample of the results** to make sure no errors accurred.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

klausur.data, klausur.report, compare, klausur.gen, klausur.gen.marks, klausur.gen.corr, plot

Examples

```
data(antworten)

# vector with correct answers:
richtig <- c(Item01=3, Item02=2, Item03=2, Item04=2, Item05=4,
 Item06=3, Item07=4, Item08=1, Item09=2, Item10=2, Item11=4,
 Item12=4, Item13=2, Item14=3, Item15=2, Item16=3, Item17=4,
 Item18=4, Item19=3, Item20=5, Item21=3, Item22=3, Item23=1,
 Item24=3, Item25=1, Item26=3, Item27=5, Item28=3, Item29=4,
 Item30=4, Item31=13, Item32=234)

# vector with assignement of marks:
notenschluessel <- c()
# scheme of assignments: marks[points_from:to] <- mark
notenschluessel[0:12]  <- 5.0
notenschluessel[13:15] <- 4.0
notenschluessel[16:18] <- 3.7
notenschluessel[19:20] <- 3.3
notenschluessel[21]    <- 3.0
notenschluessel[22]    <- 2.7
notenschluessel[23]    <- 2.3
notenschluessel[24]    <- 2.0
notenschluessel[25:26] <- 1.7
notenschluessel[27:29] <- 1.3
notenschluessel[30:32] <- 1.0

# now combine all test data into one object of class klausur.answ
data.obj <- klausur.data(answ=antworten, corr=richtig, marks=notenschluessel)

# if that went well, get the test results
klsr.obj <- klausur(data.obj)

# to try pick-n scoring, we must also define all distractors
falsch <- c(Item01=1245, Item02=1345, Item03=1345, Item04=1345, Item05=1235,
 Item06=1245, Item07=1235, Item08=2345, Item09=1345, Item10=1345, Item11=1235,
 Item12=1235, Item13=1345, Item14=1245, Item15=1345, Item16=1245, Item17=1235,
 Item18=1235, Item19=1245, Item20=1234, Item21=1245, Item22=1245, Item23=2345,
 Item24=1245, Item25=2345, Item26=1245, Item27=1234, Item28=1245, Item29=1235,
 Item30=1235, Item31=245, Item32=15)

data.obj <- klausur.data(answ=antworten, corr=richtig, wrong=falsch,
      marks=notenschluessel)
klsr.obj <- klausur(data.obj, score="pick-n")
```

```
############################
 # example for an NRET test
############################
# load sampla data in SPSS format
data(spss.data)
# define correct answers
spss.corr <- c(
   item01=2, item02=3, item03=3, item04=3, item05=2,
   item06=2, item07=3, item08=1, item09=1, item10=2)

# convert into klausuR type coding
klausuR.data <- nret.translator(spss.data, spss="in")
klausuR.corr <- nret.translator(spss.corr, spss="in", corr=TRUE,
  num.alt=3, spss.prefix=c(corr="item"))
# now create the data object; "Nickname" must be renamed
data.obj <- klausur.data(answ=klausuR.data, corr=klausuR.corr,
  rename=c(Pseudonym="Nickname"))

 # finally, the test can be evaluated, using the scoring functions available
NRET.results <- klausur(data.obj, marks="suggest", mark.labels=11, score="NRET")
NRETplus.results <- klausur(data.obj, marks="suggest", mark.labels=11, score="NRET+")
NR.results <- klausur(data.obj, marks="suggest", mark.labels=11, score="NR")
ET.results <- klausur(data.obj, marks="suggest", mark.labels=11, score="ET")
```

`klausuR,-class` *Class klausuR*

Description

This class is used for objects that are returned by `klausur`.

Slots

`results` A data.frame with global results

`answ` A data.frame with all given answers

`corr` A vector with the correct answers

`wght` A vector with the weights of items

`points` A data.frame with resulting points given for the answers

`marks` A vector with assignments of marks to achieved score

`marks.sum` A more convenient matrix with summary information on the defined marks

`trfls` A data.frame of TRUE/FALSE values, whether a subject was able to solve an item or not

`anon` A data.frame for anonymous feedback

`mean` A table with mean, median and quartiles of the test results

`sd` Standard deviation of the test results

`cronbach` Internal consistency, a list of three elements "alpha", "ci" (confidence interval 95%) and "deleted" (alpha if item was removed)

`item.analysis` A data.frame with information on difficulty, discriminant power, discriminant factor and selection index of all items.

`distractor.analysis` A list with information on the selected answer alternatives for each individual item.

`test` Currently an empty placeholder. Planned to hold the actual test items in future releases.

`misc` Anything that was stored in the `misc` slot of the input data.

`klausuR.answ,-class` *Class klausuR.answ*

Description

This class is used for objects needed by `klausur`. They contain all relevant data to calculate test results.

Slots

`corr` Contains three elements:

- `corr` The correct answers to each item.
- `corr.key` An optional data.frame or matrix for test with multiple test forms, indicating the positions of all items (columns) in all forms (rows). Must have a column called `Form` (like in `id`), and the item columns must follow the usual name scheme `Item###`. `NULL` if not needed.
- `wrong` For pick-n scoring, this is the inverse of `corr`, i.e., all wrong item alternatives.

`id` Contains the columns `No`, `Name`, `FirstName`, `MatrNo`, `Pseudonym` and `Form`.

`items` Contains a copy of `id$MatrNo` and all answers to the test items (one item per column).

`score` Contains three elements:

- `marks` The assigned marks for achieved points (`NULL` if none)
- `wght` Weights for each item (`NULL` if none)
- `maxp` Optional, to force a certain maximum points value (`NULL` if none)

`test` Currently an empty placeholder. Planned to hold the actual test items in future releases.

`misc` Any additional data you'd like to be stored along with `id` and `items`, e.g. table data from/for other software products. Won't be used for anything.

Note

The slots , `id`, `items` and `misc`, must have the same number of rows and contain copies of the colum `MatrNo` for identification.

`klausur.data` *A function to create data objects with given and correct answers to a test.*

Description

`klausur.data` automatically parses the variable names in answto decide **which variables are actual test items**, if they are named according to the given scheme `Item###`. To help in constructing a data.frame with correct column names one can call the `klausur.gen` utility to generate an empty data object of a given number of items and test subjects.

Usage

```
klausur.data(
  answ,
  corr,
  items = NULL,
  marks = NULL,
  wght = NULL,
  corr.key = NULL,
  rename = c(),
  dummies = c(),
  disc.misc = FALSE,
  na.rm = TRUE,
  item.prefix = c(),
  sort.by = "Name",
  maxp = NULL,
  wrong = NULL,
  keep.cases = NULL,
  recode.na = 0
)
```

Arguments

answ	A `data.frame` which has to include at least these variables: `No`, `Name`, `FirstName`, `MatrNo`, as well as `Pseudonym` (optional) and variables for the answered items (according to the scheme `Item###`, where ### is a number with leading zeros, if needed).
corr	A vector with the correct answers to all items in `answ` (named also according to `Item###`).
items	Indices of a subset of variables in `answ` to be taken as items.
marks	A vector assigning marks to points achieved (see details). Leave `NULL` if not available.
wght	A vector with weights for each item (named also according to `Item###`). Leave `NULL` if not available.

corr.key	If test has several test forms: A data.frame or matrix indicating the positions of all items (columns) in all forms (rows). Must have a column called Form (like answ), and the item columns must follow the explained name scheme Item###. NULL if not needed.
rename	A named vector defining if variables in answ need to be renamed into the klausuR name scheme. Accepts elements named No, Name, FirstName, MatrNo, Pseudonym and Form. The values of these elements represent the variable names of the input data.
dummies	A vector of dummy variables to be created, e.g. if you don't need/want actual data in the id slot. Can include "No", "Name", "FirstName", "MatrNo" and "Pseudonym". Columns will just be filled with increasing integers.
disc.misc	Logical. If TRUE, left over columns from answ will not be stored in slot misc but silently discarded.
na.rm	Logical, whether cases with NAs should be ignored in answ. Defaults to TRUE.
item.prefix	A named character vector with two optional elements, item and corr, defining the name prefix used for the items in the test data and the vector with correct answers, respectively. Defaults to item="Item" and corr="Item".
sort.by	A character string naming the variable to sort the answ data by. Set to c() to skip any re-ordering.
maxp	Optional numeric value, if set will be forced as the maximum number of points achievable. This should actually not be needed, if your test has no strange errors. But if for example it later turns out you need to adjust one item because it has two instead of one correct answers, this option can become handy in combination with "partial" scoring and item weights.
wrong	If you want full pick-n scoring: A vector similar to corr, but this time listing all alternatives that are wrong.
keep.cases	A vector of MatrNo values, if you want to prevent these cases from being dropped even if they contain missing data. If not NULL, missing values in all test items are replaced by the value given to recode.na, before na.rm is evaluated.
recode.na	A value to replace missing data with in all cases specified by keep.cases. Ignored if keep.cases=NULL.

Details

If you have **items with multiple correct answers** you can easily code these as one single item: All alternatives a subject has marked should be combined to a single value without spaces. The vector with correct answers will have to be coded accordingly, of course. An example: If someone marked the first, third and fourth answer, you would code this as "134". See klausur.gen.corr for a helpful function to create such an answer vector. Internally klausur checks for equality of given answers and correct values, that is, it will only give that person a point if the correct answer was coded as "134" as well.

Data for (Number Right) Elimination Testing

If your test is to be evaluated according to elimination testing (ET), number right elimination testing (NRET) or number right (NR, which is actually multiple choice) scoring, the data has to be in a different format: In contrast to the usual MC procedure, ET items are answered by eliminating all

alternatives a subject considers *wrong*; in an NRET test subjects are asked to eliminate all wrong alternatives *and* mark the one they consider the correct answer. That is, for both scoring functions, you need to know for each answer alternative whether a subject saw it as right, wrong or was not sure and left it open.

In this implementation, these answers are to be coded as a plus sign "+" (right answer), a minus sign "-" (wrong answer) or a zero "0" (missing). If you need to code errors (like both "right" and "wrong" have been marked), use the asterisk "*" for these cases. All answers to **one item** belong into **one column**. E.g., if you have four answer alternatives, a subject thought the second one to be the correct answer and eliminated the rest, you'd have to code this item as "-+--". The same is true for the vector of correct answers, of course.

Marks

The **assigned marks** are expected to be in a certain format as well, as long as you don't want `klausur` to suggest them itself. Just create an empty vector to start with (say `your.marks <-c()`) and fill it according to the scheme `your.marks[<points from>:<points to>] <-<mark>`. For example: Should one get a 1.7 if in sum 27 to 30 points were achieved, you'd assign these points as indices to the vector with `your.marks[27:30] <-"1.7"` (see example section below). It is crucial to assign marks to the whole range of points that can be achieved in the test. On the other hand, it's irrelevant wheter you assign decimal marks as in the example, only integer values, a 15 marks scheme or whatever. The convenience function `klausur.gen.marks` can assist you in creating such a valid vector.

Value

An object of class `klausuR.answ-class`.

Examples

```
data(antworten)

# vector with correct answers:
richtig <- c(Item01=3, Item02=2, Item03=2, Item04=2, Item05=4,
 Item06=3, Item07=4, Item08=1, Item09=2, Item10=2, Item11=4,
 Item12=4, Item13=2, Item14=3, Item15=2, Item16=3, Item17=4,
 Item18=4, Item19=3, Item20=5, Item21=3, Item22=3, Item23=1,
 Item24=3, Item25=1, Item26=3, Item27=5, Item28=3, Item29=4,
 Item30=4, Item31=13, Item32=234)

# vector with assignement of marks:
notenschluessel <- c()
# scheme of assignments: marks[points_from:to] <- mark
notenschluessel[0:12]  <- 5.0
notenschluessel[13:15] <- 4.0
notenschluessel[16:18] <- 3.7
notenschluessel[19:20] <- 3.3
notenschluessel[21]    <- 3.0
notenschluessel[22]    <- 2.7
notenschluessel[23]    <- 2.3
notenschluessel[24]    <- 2.0
notenschluessel[25:26] <- 1.7
notenschluessel[27:29] <- 1.3
```

```
notenschluessel[30:32] <- 1.0

# now combine all test data into one object of class klausur.answ
data.obj <- klausur.data(answ=antworten, corr=richtig, marks=notenschluessel)

# if that went well, get the test results
klsr.obj <- klausur(data.obj)
```

`klausur.gen` *Generate empty data sets for the use with klausur*

Description

Generate empty data sets for the use with klausur

Usage

```
klausur.gen(items = NULL, obs = 1, items.char = FALSE)
```

Arguments

`items`	An integer declaring the number of items to be created
`obs`	Integer, numer ob observations
`items.char`	Logical, will the answers be coded as characters or integer numbers (default)?

Value

A data.frame containing the variables "No", "Name", "FirstName", "MatrNo", "Pseudonym", "Form" and the number of items as needed.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

`klausur`, `compare`, `klausur.gen.marks`, `klausur.gen.corr`

Examples

```
antworten2 <- klausur.gen(items=20,obs=40)
```

klausur.gen.corr *A function to generate a vector with correct answers*

Description

Create a vector of correct answers to be used by klausur.

Usage

```
klausur.gen.corr(answ = NULL, items.char = FALSE, test.forms = 1)
```

Arguments

answ	Either an object with item names in klausuR scheme (see klausur), e.g. your observation data, or an integer representing the maximum score of the test. If NULL, you will be asked for the maximum score.
items.char	Logical, will the answers be coded as characters or integer numbers (default)?
test.forms	An integer value specifying how many parallel test forms are available

Details

By default answers are expected to be numeric values. You can change that to character with items.char=TRUE.

The parameter answ is quite versatile. You can just feed it your observation data, if it complies with the naming scheme for items (Item###, see also klausur.gen), and klausur.gen.corr will use all of its items automatically. Or you assign the number of items directly as an integer value. If you leave answ=NULL, you will be asked for the number of items.

Value

A numeric or character vector (depending on the parameter items.char).

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

klausur

Examples

```
## Not run:
richtig <- klausur.gen.corr(answ=antworten)

## End(Not run)
```

`klausur.gen.marks` *Generate mark assignments*

Description

Create a vector of marks to be used by `klausur`.

Usage

```
klausur.gen.marks(
  mark.labels = NULL,
  answ = NULL,
  wght = NULL,
  suggest = list(mean = NULL, sd = NULL),
  minp = 0
)
```

Arguments

`mark.labels`	Either a vector with labels (names) for all marks that should be assigned to certain test scores, or one of the arguments 6, 11, 16, "DIHK", "UK", "USA" or "A" (see Details). If NULL, you will be asked to type in labels.
`answ`	Either an object with item names in klausuR scheme (see `klausur`), e.g. your observation data, or an integer representing the maximum score of the test. If NULL, you will be asked for the maximum score.
`wght`	A vector with weights for each item. If NULL, the number of items is used as maximum score, that is, each item gives a point.
`suggest`	A list with the elements `mean` and `sd`. If both are not NULL, this function will suggest marks for achieved points assuming normal distribution. That is, "mean" and "sd" should be set to the corresponding values of the test's results.
`minp`	An integer value, in case there is a minimum score no-one can fall below (which can happen, e.g., with ET/NRET scoring and different numbers of answer alternatives). Should be left as is in most cases.

Details

If `mark.labels` is set to one of the arguments 6, 11, 16, "DIHK", "UK", "USA" or "A", often used schemes for marks will be used as a preset. In case of `mark.labels=6`, marks will go from 1 (best) to 6 (failed), as widely used in German schools. In case of `mark.labels=11`, marks will range from 1.0 (best) to 5.0 (failed), with the marks 1 through 3 being split into three decimal steps .0, .3 and .7, as is often used in academic institutions. In case of `mark.labels=16`, marks will be a range of points from 15 (best) to 0 (failed), as often used in German gymnasiums. If `mark.labels="A"`, marks A to F are given.

For the other cases some more probably useful assumptions are being made, which percentage of achieved points leads to which mark. If `mark.labels="DIHK"`, marks will be 1 through 6, and calculated according to usual standards of the Deutsche Industrie- und Handelskammer (1 > 92%, 2

> 81%, 3 > 67%, 4 > 50%, 5 > 30%, 6 below that). If `mark.labels="UK"`, marks are A > 90%, B > 65%, C > 35%, D > 10% and E below that, and for `mark.labels="USA"` it's A > 90%, B > 80%, C > 70%, D > 60% and F below that. Please note that the percentages indicate individual test results and not "the best X percent of the sample". If you'd rather use your own system, either declare it as a vector, or leave as NULL, and you'll be asked (be sure to **begin with the worst** mark!).

The parameter `answ` is quite versatile as well. You can just feed it your observation data, if it complies with the naming scheme for items (`Item###`, see also `klausur.gen`), and `klausur.gen.marks` will calculate the maximum score automatically. Or you assign the maximum directly as an integer value.

Another feature can be toggled with the parameter `suggest`. If you feed it with the mean and standard deviation values of your test's results, marks are automatically assigned to the achieved score under the assumption of normal distribution. Please understand that the naming "suggest" is not an accident! This is only a suggestion, please review it, tweak it, revise it, until it fits your needs. However, this feature can directly be called by `klausur`.

Value

A character vector.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

`klausur`

Examples

```
## Not run:
notenschluessel <- klausur.gen.marks(mark.labels=11,answ=antworten)

## End(Not run)
```

`klausur.mufo` *Evaluate multiple choice tests with several test forms*

Description

This function can be used to evaluate tests that have several test forms. Please be aware that its results only make sense if each test form uses the same items, only in a different order.

Usage

```
klausur.mufo(
  data,
  marks = NULL,
  mark.labels = NULL,
  items = NULL,
  wght = NULL,
  score = "solved",
  matn = NULL,
  na.rm = TRUE,
  cronbach = TRUE,
  item.analysis = TRUE
)
```

Arguments

data	An object of class klausuR.answ-class.
marks	A vector assigning marks to points achieved (see details). Alternatively, set it to "suggest" to let klausur.gen.marks calculate suggestions under the assumption of normal distribution. If NULL, this value must be set in the data object.
mark.labels	If marks="suggest", use these as the marks you want to give.
items	Indices of a subset of variables in answ to be taken as items.
wght	A vector with weights for each item (named also according to Item###). If NULL, the value from the data object will be used.
score	Specify the scoring policy, must be one of "solved" (default), "partial", "liberal", "NR", "ET", "NRET", or "NRET+".
matn	A matriculation number of a subject, to receive detailed results for that subject.
na.rm	Logical, whether cases with NAs should be ignored in data. Defaults to TRUE.
cronbach	Logical. If TRUE, Cronbach's alpha will be calculated.
item.analysis	Logical. If TRUE, some usual item statistics like difficulty and discriminatory power will be calculated. If cronbach is TRUE, too, it will include the alpha values if each item was deleted.

Details

Firstly, klausur.mufo will compute partial results for each parallel form, and in the end combine these to global results. Cronbach alpha and item analysis will be calculated for all subjects accordingly, therefore the test items of all tests will be re-ordered to fit the order of the first given test form (this does not apply to the partial results).

The parameters are mostly the same as those for klausur. However, in the data object the slot corr must also contain corr.key, to communicate the order of items in each test form, and the slot id needs one additional variable called Form.

An example: You have prepared a test in two different parallel forms "A" an "B", So in addition to the variables in data@id you need to create a variable called Form, to document which test subject was given which test form. Since form "B" holds the same items as form "A", only in a different

order, we only need to define these positions and we're done. Therefore `corr.key` must be a matrix or data.frame, again with a column called "Form", one column for each item, and one row of data for each test form. That is, you'd need one row for test form "A" and one for test form "B", giving an index for each item where it is placed in the form. For "A" this is simply ascending numbers from 1 to how many questions you asked, but for row "B" each number indicates at which position an item of "A" is to be found. See the example below.

Value

An object of class `klausuR.mult-class` with the following slots.

`forms`	A character vector naming all test forms
`results.part`	A list of objects of class `klausuR`, holding all partial results
`results.glob`	An object of class `klausuR` with the global results

Not all slots are shown by default (refer to `show`).

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

`klausur`, `klausur.data`

Examples

```
# this will create the data.frame "antworten.mufo"
# and the matrix "corr.key"
data(antworten.mufo)

# vector with correct answers:
richtig <- c(Item01=3, Item02=2, Item03=2, Item04=2, Item05=4,
 Item06=3, Item07=4, Item08=1, Item09=2, Item10=2, Item11=4,
 Item12=4, Item13=2, Item14=3, Item15=2, Item16=3, Item17=4,
 Item18=4, Item19=3, Item20=5, Item21=3, Item22=3, Item23=1,
 Item24=3, Item25=1, Item26=3, Item27=5, Item28=3, Item29=4,
 Item30=4, Item31=13, Item32=234)

# vector with assignement of marks:
notenschluessel <- c()
# scheme of assignments: marks[points_from:to] <- mark
notenschluessel[0:12]  <- 5.0
notenschluessel[13:15] <- 4.0
notenschluessel[16:18] <- 3.7
notenschluessel[19:20] <- 3.3
notenschluessel[21]    <- 3.0
notenschluessel[22]    <- 2.7
notenschluessel[23]    <- 2.3
notenschluessel[24]    <- 2.0
notenschluessel[25:26] <- 1.7
notenschluessel[27:29] <- 1.3
```

```
notenschluessel[30:32] <- 1.0

# now combine all test data into one object of class klausur.answ
mufo.data.obj <- klausur.data(answ=antworten.mufo, corr=richtig, marks=notenschluessel,
      corr.key=corr.key)
# expect some warnings here, because some items have no variance
# in their subtest results, hence item analysis fails on them
klsr.mufo.obj <- klausur.mufo(mufo.data.obj)
```

`klausuR.mult-class` *Class klausuR.mult*

Description

This class is used for objects that are returned by `klausur.mufo`.

Slots

`forms` A vector with the names of all test forms.

`results.part` A list with the partial results of each test form

`results.glob` An object of class klausuR-class with overall results

`klausur.report` *Generate individual reports on multipe choice test results*

Description

`klausur.report` takes (at least) an object of class klausuR (or klausuR.mult) and a matriculation number to generate personal test results in LaTeX and/or PDF format.

Usage

```
klausur.report(
  klsr,
  matn,
  save = FALSE,
  pdf = FALSE,
  path = NULL,
  file.name = "matn",
  hist = list(points = FALSE, marks = FALSE),
  hist.merge = list(),
  hist.points = "hist_points.pdf",
  hist.marks = "hist_marks.pdf",
  descr = list(title = NULL, name = NULL, date = NULL),
  marks.info = list(points = FALSE, percent = FALSE),
```

```
  lang = "en",
  alt.candy = TRUE,
  anon.glob.file = "anon.tex",
  decreasing = TRUE,
  sort.by = "Points",
  NRET.legend = FALSE,
  table.size = "auto",
  merge = FALSE,
  quiet = FALSE,
  fancyhdr = TRUE
)
```

Arguments

klsr	An object of class klausuR or klausuR.mult. To create reports from more than one object with the same configuration, you can also give them in one list here, which will cause the function to call itself recursively.
matn	Matriculation number, "all" (produces individuall documents for all subjects), "anon" (produces anonymous feedback) or "glob" (produces a global results document).
save	Logical: If TRUE, files are saved to disk (scheme: "path/matn.tex").
pdf	Logical: If TRUE, LaTeX reports will be converted to PDF automatically, using texi2dvi. If save is FALSE, a temporary directory is used, that is only the PDF files will be saved.
path	Path for save and hist files.
file.name	File name scheme for the reports, either "matn" (matriculation number) or "name" (name and firstname).
hist	A list with the logical elements points and marks: If TRUE, the reports will include histograms of the distribution of points and/or marks. The needed PDF files will be created by plot and saved as well. (see path, hist.points and hist.marks).
hist.merge	If you need/want to combine results from several klausuR class objects for the histograms, provide them all in a list here.
hist.points	File name for the histogram of points.
hist.marks	File name for the histogram of marks.
descr	Details on the test: List with the elements title (title of the test), name (your name) and date.
marks.info	A list with the logical elements points and percent: If TRUE, the reports will include a table showing how marks were assigned to points achieved and/or percent solved, respectively.
lang	Set to "de" for reports in German, English is the default.
alt.candy	If TRUE, a comma will be inserted for items with multiple alternatives ("235" becomes "2, 3, 5" in the printout)
anon.glob.file	If matn="anon" or matn="glob", you can specify a filename for this particular report.

decreasing	Logical, whether sorting of output should be done increasing or decreasing (only relevant for matn="anon" or matn="glob").
sort.by	Character string naming a variable to sort the results by. Defaults to "Marks" (only relevant for matn="anon" or matn="glob").
NRET.legend	Logical, If ET/NRET data is reported, you can demand a legend in the table caption by setting this to true.
table.size	Character string to shrink the tables, must be one of "auto", "normalsize", "small", "footnotesize", "scriptsize" or "tiny". The default table.size="auto" tries to decide between "normalsize" and "footnotesize" to avoid pages with only one or two rows. If that fails, try to manually set the size.
merge	Logical, if TRUE no individual PDFs will be saved, but one large file with all reports. Uses the "pdfpages" package, and only useful if pdf=TRUE as well.
quiet	Logical, if TRUE no feedback messages on the current status are given.
fancyhdr	Logical, if TRUE additional information is printed in the header and footer of the LaTeX/PDF files.

Details

The report contains, next to the individual results, a table with all given and correct answers (using xtable), as well as nice histograms showing the distribution of the test results (points and/or marks are supportet). If the matriculation numer is set to "all", reports for all subjects are produced. Setting it to "anon" will get you a printable version of the anonymized results.

By default output is sent to standard out. To save them to disk in LaTeX format a "save" parameter is provided. Alternatively, the reports can be converted to PDF format as well. klausur.report is calling texi2dvi from the tools package for that.

If the object is of class klausuR.mult, only the global results for tests with several test forms are evaluated. In case you'd rather like reports on each test form, call klausur.report with the single slots from that object accordingly.

Value

One or several LaTeX and/or PDF documents. If defined two histograms will be plotted.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

klausur, xtable, texi2dvi

Examples

```
data(antworten)

# vector with correct answers:
richtig <- c(Item01=3, Item02=2, Item03=2, Item04=2, Item05=4,
```

```
 Item06=3, Item07=4, Item08=1, Item09=2, Item10=2, Item11=4,
 Item12=4, Item13=2, Item14=3, Item15=2, Item16=3, Item17=4,
 Item18=4, Item19=3, Item20=5, Item21=3, Item22=3, Item23=1,
 Item24=3, Item25=1, Item26=3, Item27=5, Item28=3, Item29=4,
 Item30=4, Item31=13, Item32=234)

# vector with assignement of marks:
notenschluessel <- c()
# scheme of assignments: marks[points_from:to] <- mark
notenschluessel[0:12]  <- 5.0
notenschluessel[13:15] <- 4.0
notenschluessel[16:18] <- 3.7
notenschluessel[19:20] <- 3.3
notenschluessel[21]    <- 3.0
notenschluessel[22]    <- 2.7
notenschluessel[23]    <- 2.3
notenschluessel[24]    <- 2.0
notenschluessel[25:26] <- 1.7
notenschluessel[27:29] <- 1.3
notenschluessel[30:32] <- 1.0

data.obj <- klausur.data(answ=antworten, corr=richtig, marks=notenschluessel)
klsr.obj <- klausur(data.obj)

## Not run:
klausur.report(klsr=klsr.obj, matn="all", descr=list(title="Klausur Tatort",
  name="Dr. T. Aeter", date="24.09.2010"))

## End(Not run)
```

`klausuR.test,-class` *Class klausuR.test*

Description

This class is currently an empty placeholder. In future releases, it is planned to contain the actual test items in a list like format.

Slots

`items` Empty dummy.

`nret.rescale` *Rescale test evaluation results*

Description

By default `klausur` adds a constant to results of ET/NRET type tests. This ensures that the minimum value of points can't fall below zero. If you'd rather like to see the results without this constant, i.e. results can be negative, you can rescale them with this function.

Usage

```
nret.rescale(
  res.obj,
  score = "NRET",
  points = TRUE,
  percent = TRUE,
  marks = TRUE
)
```

Arguments

`res.obj`	An object of class `klausuR` with results of an ET/NRET coded test.
`score`	Either `"NR"`, `"ET"`, `"NRET"` or `"NRET+"`, defining the scoring function used.
`points`	Logical, whether point values should be rescaled.
`percent`	Logical, whether the percentage of received points should be rescaled, so that 0 points are 0 percent.
`marks`	Logical, whether the assigned marks should be rescaled to fit the rescaled points (probably a good idea). However, this will removed the attached mark assignment vecor, since its indices can't be negative.

Value

An object of class `klausuR` with rescaled results.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

nret.translator *Convert NRET/ET data between klausuR and other software*

Description

This function should help to interchange answer data between R and other statistical software packages – especially SPSS, but it's probably useful for other products as well.

Usage

```
nret.translator(
  dat,
  items = NULL,
  spss = "out",
  corr = FALSE,
  num.alt = NULL,
  klausuR.alt = c(is.true = "+", is.false = "-", missing = "0", err = "*"),
  spss.alt = c(is.true = "2", is.false = "1", missing = "0", err = "3"),
  rm.old.vars = TRUE,
  items.only = FALSE,
  klausuR.prefix = c(),
  spss.prefix = c()
)
```

Arguments

dat	A data.frame, the object to convert.
items	Optional vector defining the columns to convert. If NULL, the function will try to autodetect Items: klausuR type items are expected to be named "ItemXXX", with XXX indicating the item number, SPSS type items "itemXXXaYY", with XXX indicating the item number and YY the number of the answer alternative.
spss	Either "in" or "out", depending on the direction of conversion.
corr	Logical. Set to TRUE if dat is a vector with the correct answers. If corr=TRUE and spss="in", you must also set num.alt accordingly!
num.alt	A numeric value definig the number of answer alternatives for each item. Can be a vector, if items have different numbers of options. If it is shorter than the number of items, it will be repeated for all items.
klausuR.alt	A named vector defining the codes for klausuR type of answers.
spss.alt	A named vector defining the codes for SPSS type of answers.
rm.old.vars	Logical. If TRUE, the converted columns will not be returned. Only relevant if corr=FALSE.
items.only	Logical. If TRUE, only the converted columns will be returned. Only relevant if corr=FALSE.

`klausuR.prefix` A named character vector with two optional elements, `item` and `corr`, defining the name prefix used for the items in the test data and the vector with correct answers, respectively. Defaults to `item="Item"` and `corr="Item"`.

`spss.prefix` Like `klausuR.prefix`, but for the SPSS data. Defaults to `item="item"` and `corr="corr"`.

Details

`klausur` expects data in a special format if it should be evaluated according to (Number Right) Elimination Testing (NRET/ET), only one variable per item. Other software products might not be able to process this rather condensed format. In that case, you will most likely need several variables for each item, i.e. one per answer alternative. Adding to that, the coding of answers is by default done with "+", "-", "0" and "*" in `klausuR`, again a solution that might confuse other products.

This function translates data in both directions, and does also convert vectors giving the correct answer. The latter will turn a `klausuR` type answer string into a number indicating the correct alternative (and the other way round). This means that it will only work if there's exactly one valid answer to each item. If you convert towards SPSS, the resulting list will also include SPSS syntax to define variables respectively.

Value

If `corr=FALSE`, a data.frame with more or less columns (depending on `rm.old.vars` and `items.only`). If `corr=TRUE`, returns a named vector if `spss="in"` and a list if `spss="out"` (containing SPSS syntax in the element `syntax` and also a named vector, called `answ`).

Note

The conversion is done on an object basis, that is, `nret.translator` will not open or write files, but take and return R objects. The function should ignore any other columns/variables in the object.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

`klausur`

Examples

```
## Not run:
# from SPSS to R
data(spss.data)
klausuR.data <- nret.translator(spss.data, spss="in")
spss.corr <- c(
   item01=2, item02=3, item03=3, item04=3, item05=2,
   item06=2, item07=3, item08=1, item09=1, item10=2)
klausuR.corr <- nret.translator(spss.corr, spss="in", corr=TRUE, num.alt=3)
```

```
# from R to SPSS
spss.data <- nret.translator(klausuR.data)
spss.corr <- nret.translator(klausuR.corr, corr=TRUE, num.alt=3)
# if you find the syntax useful
cat(spss.corr$syntax, file="~/somewhere/NRET.sps")

## End(Not run)
```

`plot` *Plot methods for S4 objects of class klausuR and klausuR.mult*

Description

These plot methods are beeing called by `klausur.report`. If `x` is of class `klausuR.mult`, only the global results will be plotted. Should you rather like plots on each test form, call `plot` with the single slots from that object accordingly.

Usage

```
plot(x, y, ...)

## S4 method for signature 'klausuR,missing'
plot(x, marks = FALSE, sd.lines = FALSE, plot.normal = TRUE,
      na.rm = TRUE, ...)

## S4 method for signature 'klausuR.mult,missing'
plot(x, marks = FALSE, sd.lines = FALSE,
      plot.normal = TRUE, ...)
```

Arguments

`x`	An S4 object of class `klausuR` or `klausuR.mult`
`y`	From the generic `plot` function, ignored for klausuR class objects.
`...`	Any other argument suitable for plot()
`marks`	Logical, whether the histogram should show the distribution of points (default) or marks
`sd.lines`	Logical, whether standard deviation lines should be plotted
`plot.normal`	Logical, whether normal distribution should be plotted (according to mean and Sd of the results)
`na.rm`	Logical, whether NA values should be ignored. Defaults to TRUE, because plotting would fail otherwise

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

klausur, klausur.mufo, klausur.report

Examples

```
data(antworten)

# vector with correct answers:
richtig <- c(Item01=3, Item02=2, Item03=2, Item04=2, Item05=4,
 Item06=3, Item07=4, Item08=1, Item09=2, Item10=2, Item11=4,
 Item12=4, Item13=2, Item14=3, Item15=2, Item16=3, Item17=4,
 Item18=4, Item19=3, Item20=5, Item21=3, Item22=3, Item23=1,
 Item24=3, Item25=1, Item26=3, Item27=5, Item28=3, Item29=4,
 Item30=4, Item31=13, Item32=234)

# vector with assignement of marks:
notenschluessel <- c()
# scheme of assignments: marks[points_from:to] <- mark
notenschluessel[0:12]  <- 5.0
notenschluessel[13:15] <- 4.0
notenschluessel[16:18] <- 3.7
notenschluessel[19:20] <- 3.3
notenschluessel[21]    <- 3.0
notenschluessel[22]    <- 2.7
notenschluessel[23]    <- 2.3
notenschluessel[24]    <- 2.0
notenschluessel[25:26] <- 1.7
notenschluessel[27:29] <- 1.3
notenschluessel[30:32] <- 1.0

data.obj <- klausur.data(answ=antworten, corr=richtig, marks=notenschluessel)
klsr.obj <- klausur(data.obj)
plot(klsr.obj, marks=TRUE)
```

---

show,klausuR-method *Show methods for S4 objects of classes klausuR and klausuR.mult*

Description

Print a nice summary of the slots results, anon, cronbach and item.analysis. If object is of class klausuR.mult, the global results for tests with several test forms are evaluated.

Usage

```
## S4 method for signature 'klausuR'
show(object)

## S4 method for signature 'klausuR.mult'
show(object)
```

Arguments

object	An object of class klausuR or klausuR.mult

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

klausur, klausur.mufo

Examples

```
## Not run:
klausur(data.obj)

# multiple test forms
klausur.mufo(data.obj, marks=notenschluessel)

## End(Not run)
```

---

sort *Sort method for S4 objects of class klausuR*

Description

Returns the given object, with global results and anonymized results sorted by the given variable.

Usage

```
sort(x, decreasing = FALSE, ...)

## S4 method for signature 'klausuR'
sort(x, decreasing = FALSE, sort.by = c())
```

Arguments

x	An object of class klausuR
decreasing	Logical, whether sorting should be sone increasing or decreasing.
...	Additional arguments.
sort.by	An optional character string naming a variable to sort the results by. Defaults to c(), i.e. no re-ordering.

Author(s)

m.eik michalke <meik.michalke@uni-duesseldorf.de>

See Also

klausur

Examples

```
## Not run:
klsr.obj <- klausur(data.obj)
sort(klsr.obj, sort.by="Points")

## End(Not run)
```

---

spss.data *Sample dataset: spss.data*

Description

Sample data of a test with NRET data.

Usage

```
data(spss.data)
```

Format

A data.frame with 17 observations of the following 35 variables.

`No` Serial number for each subject (numeric vector)

`MatrNo` Matriculation number (character vector)

`FirstName` First name (character vector)

`Name` Family name (character vector)

`Nickname` Pseudonym for anonymous feedback, needs to be renamed (character vector)

`item01a1` **to** `item10a3` Given answers, three answers for each item, i.e. one for each alternative (numeric vector)

Details

This data was anonymised and combined from originally 51 observations. See nret.translator on details how the answers are coded.

Source

The data was taken from a final exam test in diagnostic methods at the Institute for Experimental Psychology of the Heinrich-Heine-University Düsseldorf, summer term 2011.

Examples

```
data(spss.data)
```

Index

∗ **IO**
  klausur.report, 24
∗ **classes**
  klausuR,-class, 13
  klausuR.answ,-class, 14
  klausuR.mult-class, 24
  klausuR.test,-class, 27
∗ **datagen**
  klausur.gen, 18
∗ **datasets**
  antworten, 3
  antworten.mufo, 4
  spss.data, 34
∗ **file**
  klausur.report, 24
∗ **methods**
  plot, 31
  show,klausuR-method, 32
  sort, 33
∗ **misc**
  grand.table, 7
  klausur, 8
  klausur.mufo, 21
  nret.rescale, 28
  nret.translator, 29
∗ **plot**
  plot, 31
∗ **utilities**
  compare, 5
  klausur.gen.corr, 19
  klausur.gen.marks, 20

antworten, 3
antworten.mufo, 4

compare, 5, *12*, *18*
corr.key *(*antworten.mufo*)*, 4

data.frame, *15*

grand.table, 7

klausuR *(*klausuR-package*)*, 2
klausur, *5, 6*, 8, *13, 14*, *18–23*, *26*, *30*, *32–34*
klausuR,-class, 13
klausuR-class *(*klausuR,-class*)*, 13
klausuR-package, 2
klausuR.answ,-class, 14
klausuR.answ-class
        *(*klausuR.answ,-class*)*, 14
klausur.data, *9, 10*, *12*, 15, *23*
klausur.gen, *12*, *15*, 18, *19*, *21*
klausur.gen.corr, *12*, *16*, *18*, 19
klausur.gen.marks, *8*, *10*, *12*, *17, 18*, 20, *22*
klausur.mufo, 21, *24*, *32, 33*
klausuR.mult,-class
        *(*klausuR.mult-class*)*, 24
klausuR.mult-class, 24
klausur.report, *12*, 24, *31, 32*
klausuR.test,-class, 27
klausuR.test-class
        *(*klausuR.test,-class*)*, 27

nret.rescale, *7*, *10*, 28
nret.translator, 29, *34*

plot, *11, 12*, *25*, 31
plot,-methods *(*plot*)*, 31
plot,klausuR,missing,ANY-method *(*plot*)*,
        31
plot,klausuR,missing-method *(*plot*)*, 31
plot,klausuR-method *(*plot*)*, 31
plot,klausuR.mult,missing,ANY-method
        *(*plot*)*, 31
plot,klausuR.mult,missing-method
        *(*plot*)*, 31
plot,klausuR.mult-method *(*plot*)*, 31
psych, *11*

show, *11*, *23*
show,-methods *(*show,klausuR-method*)*, 32
show,klausuR-method, 32

show,klausuR.mult-method
(show,klausuR-method), 32
sort, 33
sort,-methods (sort), 33
sort,klausuR-method (sort), 33
sort.klausuR,klausuR-method (sort), 33
spss.data, 34

texi2dvi, *25*, *26*

xtable, *26*