Package 'exams'

March 17, 2017

Version 2.2-1

Date 2017-03-16

Title Automatic Generation of Exams in R

Description Automatic generation of exams based on exercises in Markdown or LaTeX format, possibly including R code for dynamic generation of exercise elements.
Exercise types include single-choice and multiple-choice questions, arithmetic problems, string questions, and combinations thereof (cloze). Output formats include standalone files (PDF, HTML, Docx, ODT, ...), Moodle XML, QTI 1.2 (for OLAT/OpenOLAT), QTI 2.1, Blackboard, ARSnova, and TCExam. In addition to fully customizable PDF exams, a standardized PDF format is provided that can be printed, scanned, and automatically evaluated.

Depends R (>= 3.0.0)

Imports stats, graphics, grDevices, tools, utils

Suggests base64enc, knitr, parallel, png, RCurl, RJSONIO, rmarkdown, tth

License GPL-2 | GPL-3

URL http://exams.R-Forge.R-project.org/

NeedsCompilation no

Author Achim Zeileis [aut, cre],
Bettina Gruen [aut],
Friedrich Leisch [aut],
Nikolaus Umlauf [aut],
Niels Smits [ctb],
Mirko Birbaumer [ctb],
Dominik Ernst [ctb]

Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>

Repository CRAN

Date/Publication 2017-03-16 23:49:09 UTC

R topics documented:

`exams` *Generation of Simple Exams*

Description

Old (version 1) interface for Sweave-based automatic generation of exams including multiple choice questions and arithmetic problems. Now it is recommended to use the (version 2) interface `exams2pdf`.

Usage

```
exams(file, n = 1, nsamp = NULL, dir = NULL, template = "plain",
  inputs = NULL, header = list(Date = Sys.Date()), name = NULL,
  quiet = TRUE, edir = NULL, tdir = NULL, control = NULL)
```

Arguments

file	character. A specification of a (list of) exercise files, for details see below.
n	integer. The number of copies to be compiled from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
dir	character. The output directory, this has to be set if n is greater than 1 (or template is longer than 1).
template	character. A specification of a LaTeX template. The package currently provides "exam", "solution", "plain". For details see below.
inputs	character. Names of files that are needed as inputs during LaTeX compilation (e.g., style files, headers). Either the full path must be given or the file needs to be in edir.
header	list. A list of further options to be passed to the LaTeX files.
name	character. A name prefix for resulting exercises, by default chosen based on template.
quiet	logical. Should output be suppressed when calling Sweave and texi2dvi.
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
control	A list of control arguments for the appearance of multiple choice results (see 'Details').

Details

exams is the old (version 1) interface for Sweave-based generation of PDF exams. It is only provided for backward compatibility and is superseded by the far more flexible function exams2pdf.

exams generates exams from lists of Sweave source files by: (1) running Sweave on each exercise, (2) including the resulting LaTeX files in a template, (3) running texi2dvi on the template, and (4) storing the resulting PDF file in an output dir (or displaying it interactively).

Each exercise in an exam is essentially a standalone Sweave source file that exams knows (almost) nothing about, it just calls Sweave (n times). The only exception is some meta-information which is passed by means of four commands back to exams. The commands are '\extype' (which may be 'mchoice' or 'num'), '\exsolution' (e.g., 3.124 for a numeric solution and 10010 for a multiple choice solution), '\exstring' (containing a human-readable string with the solution), and '\extol' (a tolerance for numeric solutions).

The specification in file should be either of form "foo" or equivalently "foo.Rnw", where the file "foo.Rnw" should either be in the local directory, the edir directory or in the exercises directory of the package. file can either be a simple vector or a list of vectors. In the latter case, exercises are chosen randomly within each list element. For example, the specification file = list(c("a", "b"), "xyz") will result in an exam with two exercises: the first exercise is chosen randomly between "a" and "b" while "xyz" is always included as the second exercise.

The template is a (vector of) specification(s) of LaTeX templates. It can be "foo" or equivalently "foo.tex" where "foo.tex" should either be in the local directory (or provided with the full path) or in the tex directory of the package. It should specify where in the template the exercises are included, using the markup '\exinput{exercises}'. Additionally, it may contain '\exinput{questionnaire}' and '\exinput{header}'. template can also be a vector, then for each of the n runs several output files (one for each template) are created.

The name prefix for each file is by default the base name of the corresponding template but can also be changed via name.

exams creates the PDF files and stores them in an output directory together with the solution meta information as 'metainfo.rda' (see also below). If only a single PDF is created (currently the default), dir may be NULL and it is only displayed on the screen.

The argument control is specified by a named list with elements mchoice.print and mchoice.symbol. The element mchoice.print is used for specifying the characters used for printing. It is again a named list where element True gives the (five) characters used for printing when the answer is correct and False if the answer is wrong. The symbol used for the questionnaire output in the final PDF file is defined by mchoice.symbol which is vector with elements True and False.

Value

An object of class "exams_metainfo" is returned invisibly. It is a list of length n, containing a list of meta informations for each exercise:

mchoice	logical. Is the exercise a multiple choice exercise?
length	integer. Length of solution.
solution	either a logical vector (for multiple choice) or numeric vector (for arithmetic problems).
string	character. A human-readable version of the solution.

References

Gruen B, Zeileis A (2009). Automatic Generation of Exams in R. *Journal of Statistical Software*, **29**(10), 1–14. http://www.jstatsoft.org/v29/i10/.

See Also

exams2pdf, Sweave, texi2dvi, mchoice2string

Examples

```
## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## define an exams (= list of exercises)
myexam <- list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  "scatterplot",
```

```
  "relfreq"
)

if(interactive()) {
## compile a single random exam (displayed on screen)
sol <- exams(myexam)
sol
}

## generate multiple exams (stored in output directory)
odir <- tempfile()
sol <- exams(myexam, n = 3, dir = odir, template = c("exam", "solution"))
sol

## inspect solution for a particular exam
print(sol, 3)

if(interactive()) {
## modify control argument for printing
mymchoice.control <- list(mchoice.print = list(True = LETTERS[1:5], False = "_"))
sol <- exams("boxplots", template = "solution",
  control = mymchoice.control)
sol
}
```

---

exams2arsnova *Generation of Exam/Quiz Sessions in ARSnova Format*

Description

Interface for generating interactive sessions in the JSON format of the audience response system ARSnova.

Usage

```
exams2arsnova(file, n = 1L, dir = ".",
  name = "R/exams", sname = NULL, qname = NULL,
  quiet = TRUE, resolution = 100, width = 4, height = 4, svg = FALSE, encoding = "",
  url = "https://arsnova.eu/api", sessionkey = NULL, jsessionid = NULL,
  active = TRUE, votingdisabled = FALSE, showstatistic = FALSE, showanswer = FALSE,
  abstention = TRUE, variant = "lecture", ssl.verifypeer = TRUE,
  fix_choice = FALSE, ...)

make_exams_write_arsnova(url = "https://arsnova.eu/api", sessionkey = NULL,
  jsessionid = NULL, name = "R/exams", sname = NULL, qname = NULL,
  active = TRUE, votingdisabled = FALSE, showstatistic = FALSE, showanswer = FALSE,
  abstention = TRUE, variant = "lecture", ssl.verifypeer = TRUE, fix_choice = FALSE)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file.
dir	character. The default is either display on the screen or the current working directory.
name	character. A name prefix for resulting exercises and RDS file.
sname	character. A vector of length 2 with the session name (maximum of 50 characters) and its abbreviation (maximum of 8 characters). Ignored if the sessionkey of an existing session is supplied and otherwise copied by default from name.
qname	character. A vector of names for each question/exercise in file. By default, the name is used.
quiet	logical. Should output be suppressed when calling xweave and texi2dvi.
resolution, width, height	numeric, passed to xweave.
svg	logical. Should graphics be rendered in SVG or PNG (default)?
encoding	character, passed to xweave.
url,sessionkey,jsessionid	character specifying (1) the base URL of the ARSnova API, (2) the 8-digit ARSnova session key, (3) the JSESSIONID cookie of an active ARSnova session. If all are provided all questions are imported directly into the existing ARSnova session. Otherwise, a JSON import file is generated.
active	logical. Should the question be active (i.e., released for students) or locked?
votingdisabled	logical. Should voting be disabled?
showstatistic	logical. Should statistics be shown?
showanswer	logical. Should answers be shown?
abstention	logical. Are abstentions allowed?
variant	character. Should the question be a "lecture" or a "preparation" questions?
ssl.verifypeer	logical. Should SSL certificates be validated when connecting via https?
fix_choice	logical. Should math markup be removed in single and multiple choice lists? (This is may be needed for older ARSnova versions where math markup is rendered in the question itself but not the choice list.)
...	arguments passed on to xexams.

Details

exams2arsnova generates exams in the JSON format for ARSnova using xexams. It proceeds by (1) calling xweave on each exercise, (2) reading the resulting LaTeX code, (3) transforming the LaTeX code to Markdown, and (4) embedding the Markdown code into the JSON format for ARSnova (and optionally imports it into a running ARSnova session).

For steps (1) and (2) the standard drivers in xexams are used. For step (3) a suitable transformation function is set up on the fly using make_exercise_transform_pandoc. For step (4) a simple writer function is set up on the fly that embeds the transformed Markdown code into a hard-coded JSON

template using toJSON and either writes a single JSON file for each exam or imports these directly into an ARSnova session.

When url, sessionkey, and jsessionid are all supplied, curlPerform is used to import tall questions directly into the existing ARSnova session. Otherwise, a file is written to the disk and then needs to be imported manually into an ARSnova server. This file is either a JSON file for a whole new session (if sessionkey is NULL, the default) or a CSV file with the questions only.

Value

A list of exams as generated by xexams is returned invisibly.

Examples

```
## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## Not run:
## exams2arsnova can either create text files with JSON data
exams2arsnova("tstat2")

## or directly post this to an active ARSnova session (for which the
## server URL, the 8-digit session key, and the JSESSIONID cookie are needed)
exams2arsnova("tstat2", url = "https://arsnova.eu/api",
  sessionkey = "49061284", jsessionid = "A5BEFDA4141816BB425F2204A602E4B3")

## End(Not run)
```

---

exams2blackboard *Generation of Exams in Blackboard Format*

Description

Automatic generation of exams in Blackboard format (which is partially based on QTI 1.2).

Usage

```
exams2blackboard(file, n = 1L, nsamp = NULL, dir = ".",
  name = NULL, quiet = TRUE, edir = NULL,
  tdir = NULL, sdir = NULL, verbose = FALSE,
  resolution = 100, width = 4, height = 4, encoding = "",
  num = NULL, mchoice = NULL,
  schoice = mchoice, string = NULL, cloze = NULL,
  template = "blackboard",
  pdescription = "This is an item from an item pool.",
  tdescription = "This is today's test.",
  pinstruction = "Please answer the following question.",
  tinstruction = "Give an answer to each question.",
```

```
  maxattempts = 1, zip = TRUE, points = NULL,
  eval = list(partial = TRUE, negative = FALSE),
  base64 = FALSE, converter = NULL,
  ...)

make_itembody_blackboard(rtiming = FALSE, shuffle = FALSE,
  rshuffle = shuffle, minnumber = NULL, maxnumber = NULL,
  defaultval = NULL, minvalue = NULL, maxvalue = NULL,
  cutvalue = NULL, enumerate = TRUE, digits = NULL,
  tolerance = is.null(digits), maxchars = 12,
  eval = list(partial = TRUE, negative = FALSE),
  qti12 = FALSE)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
dir	character. The default is the current working directory.
name	character. A name prefix for resulting exercises and ZIP file.
quiet	logical. Should output be suppressed when calling xweave?
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.
verbose	logical. Should information on progress of exam generation be reported?
resolution, width, height	numeric. Options for rendering PNG graphics passed to xweave.
encoding	character, passed to xweave.
num	function or named list applied to numerical (i.e., type num) questions. If num is a function, num will be used for generating the item body of the question, see function make_itembody_blackboard(). If num is a named list, these arguments will be passed to function make_itembody_blackboard().
mchoice, schoice, string, cloze	function or named list applied to multiple choice, single choice, string, and cloze questions (i.e., type mchoice, schoice, string, and cloze), respectively. See argument num for more details.
template	character. The IMS QTI 1.2 or 2.1 template that should be used. Currently, the package provides "blackboard.xml".

pdescription	character. Description (of length 1) of the item pool (i.e., the set of copies).
tdescription	character. Description (of length 1) of the overall assessment (i.e., exam).
pinstruction	character. Instruction (of length 1) for the item pool (i.e., set of copies).
tinstruction	character. Instruction (of length 1) for the overall assessment (i.e., exam).
maxattempts	integer. The maximum attempts for one question, may also be set to `Inf`.
zip	logical. Should the resulting XML file (plus supplements) be zipped?
points	integer. How many points should be assigned to each exercise? Note that this argument overules any exercise points that are provided within an `"\expoints{}"` tag in the `.Rnw` file. The vector of points supplied is expanded to the number of exercises in the exam.
eval	named list, specifies the settings for the evaluation policy, see function `exams_eval`.
base64	logical. Should images be embedded using Base 64 coding? Argument `base64` may also be a character vector of file endings that should be Base 64 encoded, e.g. `base64 = c("png", "rda")` will only encode PNG images and binary `.rda` files.
converter, ...	arguments passed on to `make_exercise_transform_html`. The default for `converter` is set to `"ttm"` unless there are Rmd exercises in `file` where `"pandoc"` is used.
rtiming, shuffle, rshuffle, minnumber, maxnumber, defaultval, minvalue, maxvalue	arguments used for IMS QTI 1.2 item construction, for details see the XML specification (see IMS Global Learning Consortium, Inc. 2012), especially Section 4.
cutvalue	numeric. The cutvalue at which the exam is passed.
enumerate	logical. Insert potential solutions in enumerated list?
digits	integer. How many digits should be used for `num` exercises?
tolerance	logical. Should tolerance intervals be used for checking if the supplied `num` answer/number is correct? The default is to use tolerance intervals if `digits = NULL`.
maxchars	numeric. Lower bound for the number of characters in fill-in-blank fields. The actual number of characters is selected as the maximum number of characters of this value and the actual solution.
qti12	logical. For reverse compability to plain QTI 1.2 XML format.

Details

Blackboard employs an XML format that essentially uses the Question & Test Interoperability (QTI) standard, version 1.2, see IMS Global Learning Consortium, Inc. (2012). However, as this deviates from the plain QTI 1.2 standard in several places, the `exams2qti12` cannot be used directly. Instead, `exams2blackboard` is a new interface that is likely to be improved in future versions.

`exams2blackboard` produces a `.zip` file that may be uploaded into Blackboard. This includes the final XML file of the exam/assessment as well as possible supplement folders that include images, data sets etc. used for the exam. After uploading the test into Blackboard, the material will appear under 'Course Tools': the test will be available in 'Tests', and each pool within the test will also appear in 'Pools'.

exams2blackboard proceeds by (1) calling xweave on each exercise, (2) reading the resulting LaTeX code, (3) transforming the LaTeX code to HTML, and (4) embedding the HTML code in a XML file using Blackboard's QTI standards for assessments and question items. For steps (1) and (2) the standard drivers in xexams are used. In step (3), a suitable transformation function is set up on the fly using make_exercise_transform_html, see also the details section in exams2html. For step (4), the function will cycle through all questions and exams to generate the final XML file in the Blackboard QTI standard. Therefore, each question will be included in the XML as one section. The replicates of each question will be written as question items of the section.

The function uses the XML template for Blackboard's QTI standards for assessments and items to generate the exam (per default, this is the XML file blackboard.xml provided in the xml folder of this package). The assessment template must provide one section including one item. exams2blackboard will then use the single item template to generate all items, as well as the assessment and section specifications set within the template.

The default template will generate exams/assessments that sample one replicate of a question/item for each section. The usual procedure in exam/assessment generation would be to simply copy \& paste the XML template of the package and adapt it to the needs of the user. Note that all specifiers that have a leading ## in the XML template will be replaced by suitable code in exams2blackboard and should always be provided in the template. I.e., the user may add additional tags to the XML template or modify certain specifications, like the number of replicates/items that should be sampled for each section etc.

Per default, the individual question/item bodies are generated by function make_itembody_blackboard, i.e., make_itembody_blackboard checks the type of the question and will produce suitable XML code. Note that for each question type, either the arguments of make_itembody_blackboard may be set within num, mchoice, schoice and string in exams2blackboard, by providing a named list of specifications that should be used, or for each questiontype, a function that produces the item body XML code may be provided to num, mchoice, schoice and string. E.g., mchoice = list(shuffle = TRUE) will force only multiple choice questions to have a shuffled answerlist.

Note that in Blackboard cloze items are not officially supported, and hence this type of item is not supported in the current version of exams2blackboard either. It is currently investigated if a workaround may be implemented to allow for cloze items.

Value

exams2blackboard returns a list of exams as generated by xexams.

make_itembody_blackboard returns a function that generates the XML code for the itembody tag in Blackboard's version of the IMS QTI 1.2 format.

References

Blackboard, Inc. (2016). *Blackboard Help: Question types*. http://en-us.help.blackboard.com/Learn/Instructor/Tests_Pools_Surveys/040_Question_Types

IMS Global Learning Consortium, Inc. (2012). *IMS Question & Test Interoperability: ASI XML Binding Specification Final Specification Version 1.2*. http://www.imsglobal.org/question/qtiv1p2/imsqti_asi_bindv1p2.html

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. http://www.jstatsoft.org/v58/i01/.

See Also

exams2qti12

Examples

```
## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## define an exams (= list of exercises)
myexam <- list(
  "boxplots",
  "ttest",
  "anova",
  "scatterplot",
  "relfreq"
)

## output directory
mydir <- tempdir()

## generate .zip with Blackboard exam in temporary directory
exams2blackboard(myexam, n = 3, dir = mydir)
dir(mydir)
```

---

exams2html *Generation of Exams in HTML Format*

Description

Automatic generation of exams in HTML format.

Usage

```
exams2html(file, n = 1L, nsamp = NULL, dir = ".", template = NULL,
  name = NULL, quiet = TRUE, edir = NULL, tdir = NULL, sdir = NULL, verbose = FALSE,
  question = "<h4>Question</h4>", solution = "<h4>Solution</h4>",
  mathjax = FALSE, resolution = 100, width = 4, height = 4, svg = FALSE,
  encoding = "", converter = NULL, ...)

make_exercise_transform_html(converter = c("ttm", "tth", "pandoc", "tex2image"),
  base64 = TRUE, ...)

make_exams_write_html(template = "plain", name = NULL,
```

```
  question = "<h4>Question</h4>", solution = "<h4>Solution</h4>",
  mathjax = FALSE)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
dir	character specifying the output directory (default: current working directory). If only a single HTML file is produced and no dir is explicitly specified, the file is displayed in the browser rather than saved in dir.
template	character. A specification of a HTML template. The default is to use the "plain.html" file provided in the package unless there are Rmd exercises in file. Then, "plain8.html" is used (which expects UTF-8 encoding as used by pandoc).
name	character. A name prefix for resulting exercises.
quiet	logical. Should output be suppressed when calling xweave?
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.
verbose	logical. Should information on progress of exam generation be reported?
question	character or logical. Should the question be included in the HTML output? If question is a character it will be used as a header for resulting questions. Argument question may also be a vector that controls the output for the templates.
solution	character or logical, see argument question.
mathjax	logical. Should the JavaScript from http://www.MathJax.org/ be included for rendering mathematical formulas?
resolution, width, height	numeric. Options for rendering PNG (or SVG) graphics passed to xweave.
svg	logical. Should graphics be rendered in SVG or PNG (default)?
encoding	character, passed to xweave.
base64	logical. Should images be embedded using Base 64 coding? Argument base64 may also be a character vector of file endings that should be Base 64 encoded, e.g. base64 = c("png", "rda") will only encode PNG images and binary .rda files.
converter, ...	arguments passed on to make_exercise_transform_html. The default for converter is set to "ttm" unless there are Rmd exercises in file where "pandoc" is used.

Details

exams2html generates exams in a very simple HTML format using xexams. It proceeds by (1) calling xweave on each exercise, (2) reading the resulting LaTeX code, (3) transforming the LaTeX code to HTML, and (4) embedding the HTML code in a template (a simple and plain template is used by default).

For steps (1) and (2) the standard drivers in xexams are used.

For step (3) a suitable transformation function is set up on the fly using make_exercise_transform_html. This transforms the LaTeX code in question/questionlist and solution/solutionlist by leveraging one of four functions: ttm produces HTML with MathML for mathematical formulas, tth produces plain HTML that aims to emulate mathematical formulas, pandoc_convert employs pandoc offering different options for handling formulas, and tex2image runs LaTeX and turns the result into a single image. In all cases, images can either be stored in supplementary files or embedded directly in Base 64 coding.

For step (4) a simple writer function is set up on the fly that embeds the transformed HTML code into a template and writes a single HTML file for each exam.

Value

exams2html returns a list of exams as generated by xexams.

make_exercise_transform_html returns a function that is suitable for being supplied as driver$transform to xexams.

make_exams_write_html returns a function that is suitable for being supplied as driver$write to xexams.

References

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. http://www.jstatsoft.org/v58/i01/.

See Also

xexams, ttm, tth, pandoc_convert, tex2image, browseURL

Examples

```
## load package and enforce par(ask = FALSE)
options(device.ask.default = FALSE)

if(interactive()) {
## compile a single random exam (displayed in the browser)
exams2html(list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  "scatterplot",
  "relfreq"
))
```

```
## examples with different locales (UTF-8, ISO-8859-15)
## using special characters (Euro and Pound symbol, German umlaut)
if(!identical(Sys.getlocale(), "C")) {
## UTF-8
exams2html("currency8", encoding = "utf8", template = "plain8")

## ISO Latin 9 (aka ISO-8859-15)
exams2html("currency9", encoding = "latin9", template = "plain9")
}


## various versions of displaying mathematical formulae

## via MathML (displayed correctly in MathML-aware browsers, e.g. Firefox)
exams2html("tstat")

## via MathML + MathJax (should work in all major browsers,
## note the display options you get when right-clicking on the formulas
## in the browser)
exams2html("tstat", mathjax = TRUE)

## via plain HTML (works in all browsers but with inferior formatting)
exams2html("tstat", converter = "tth")

## via HTML with embedded picture (works in all browsers but
## is slow and requires LaTeX and ImageMagick)
## Not run:
exams2html("tstat", converter = "tex2image")

## End(Not run)
}
```

`exams2lops` *Generation of Exams in LOPS Exam Server Format (WU Wien)*

Description

Automatic generation of exams in LOPS exam server format (WU Wien).

Usage

```
exams2lops(file, n = 1L, nsamp = NULL, dir = ".", name = NULL,
  quiet = TRUE, edir = NULL, tdir = NULL, sdir = NULL, verbose = FALSE,
  solution = TRUE, doctype = NULL, head = NULL, resolution = 100, width = 4,
 height = 4, svg = FALSE,  encoding = "", converter = "tex2image", base64 = FALSE,
  auto_scramble = TRUE, ...)

make_exams_write_lops(name = NULL, auto_scramble = TRUE, ...)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
dir	character. The default is the current working directory.
name	character. A name prefix for resulting exercises.
quiet	logical. Should output be suppressed when calling xweave?
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.
verbose	logical. Should information on progress of exam generation be reported?
solution	logical. Should the solution be included in the HTML output?
doctype	character vector with a DOCTYPE tag for the HTML page. By default HTML4 is employed.
head	character vector for the head tag. By default a simple header is employed, setting the font to Arial.
resolution, width, height	numeric. Options for rendering PNG (or SVG) graphics passed to xweave.
svg	logical. Should graphics be rendered in SVG or PNG (default)?
encoding	character, passed to xweave.
converter	character. Workhorse function for transforming LaTeX code to HTML.
base64	logical. Should images be embeddeg using Base 64 coding?
auto_scramble	logical. Should answers be scrambled automaticall?
...	arguments passed on to make_exercise_transform_html.

Details

exams2lops will produce a .zip file that may be uploaded. It proceeds by (1) calling xweave on each exercise, (2) reading the resulting LaTeX code, (3) transforming the LaTeX code to HTML, and (4) embedding the HTML code in a XML file using the LOPS exam server XML format (WU Wien).

For steps (1) and (2) the standard drivers in xexams are used. In step (3), a suitable transformation function is set up on the fly using make_exercise_transform_html, see also the details section in exams2html.

For step (4) a simple writer function is set up on the fly that embeds the transformed HTML code into the final XML files for each question and the exam.

Note that in make_exams_write_lops only multiple and single choice questions are supported at the moment, since the LOPS exam server XML format (WU Wien) is used to generate printed versions for large scale multiple choice exams. In addition, only images of the question/questionlist/solution/solutionlist should be generated, since the server has only minimum support for e.g. MathML markup used to produce mathematical formulas.

Value

exams2lops returns a list of exams as generated by xexams.

make_exams_write_lops eturns a function that generates the XML code for the question in LOPS exam server format (WU Wien).

References

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. http://www.jstatsoft.org/v58/i01/.

See Also

xexams, ttm, tth, tex2image, make_exercise_transform_html,

Examples

```
## Not run:
## output directory
mydir <- tempdir()

## generate the exam
exams2lops(c("scatterplot", "boxplots"), dir = mydir)
dir(mydir)

## End(Not run)
```

---

exams2moodle *Generation of Exams in Moodle XML Format*

Description

Automatic generation of exams in Moodle XML format.

Usage

```
exams2moodle(file, n = 1L, nsamp = NULL, dir = ".",
  name = NULL, quiet = TRUE, edir = NULL,
  tdir = NULL, sdir = NULL, verbose = FALSE,
  resolution = 100, width = 4, height = 4, svg = FALSE, encoding = "",
  iname = TRUE, stitle = NULL,
```

```
  testid = FALSE, zip = FALSE, num = NULL, mchoice = NULL,
  schoice = mchoice, string = NULL, cloze = NULL,
  points = NULL, rule = NULL, pluginfile = TRUE,
  converter = NULL, ...)

make_question_moodle(name = NULL, solution = TRUE,
  shuffle = FALSE, penalty = 0, answernumbering = "abc",
  usecase = FALSE, cloze_mchoice_display = "MULTICHOICE",
  truefalse = c("True", "False"), enumerate = TRUE, abstention = NULL,
  eval = list(partial = TRUE, negative = FALSE, rule = "false2"),
  essay = NULL)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
dir	character. The default is the current working directory.
name	character. A name prefix for resulting exercises and ZIP file.
quiet	logical. Should output be suppressed when calling xweave?
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.
verbose	logical. Should information on progress of exam generation be reported?
resolution, width, height	numeric. Options for rendering PNG (or SVG) graphics passed to xweave.
svg	logical. Should graphics be rendered in SVG or PNG (default)?
encoding	character, passed to xweave.
iname	logical. Should the exam name be included in the path in the <category> tag in the final XML file? This option may be useful when questions should be added to certain already existing question banks, i.e. iname = TRUE will include the exam name by $course$/ExamName/.
stitle	character. For the questions specified in argument file, additional section titles may be set. The section titles will then be added to the <category> tag in the final XML file (see also argument iname), i.e. the section name for each question will be written to $course$/ExamName/SectionName. Note that section names may also be provided in the \exsection{} tag in the .Rnw file of the question. However, section names that are specified in stitle will overwrite \exsection{} tags. stitle may also include NA, e.g. stitle = c("Exercise 1", NA, "Exercise 3")

`testid` logical. Should an unique test id be added to the exam `name`.

`zip` logical. Should the resulting XML file be zipped?

`num` function or named list applied to numerical (i.e., type `num`) questions. If `num` is a function, `num` will be used for generating the item body of the question, see function `make_itembody_qti12()`. If `num` is a named list, these arguments will be passed to function `make_itembody_qti12()`.

`mchoice, schoice, string, cloze` function or named list applied to multiple choice, single choice, string, and cloze questions (i.e., type `mchoice`, `schoice`, `string`, and `cloze`), respectively. See argument `num` for more details.

`points` integer. How many points should be assigned to each exercise? Note that this argument overules any exercise points that are provided within an `"\expoints{}"` tag in the `.Rnw` file. The vector of points supplied is expanded to the number of exercises in the exam.

`rule` character specifying which rule to use for negative partial credits. see function `exams_eval`. Note that the default using cloze exercises is `rule = "none"`.

`pluginfile` logical. Should supplements be included in the Moodle XML file via Moodle's Pluginfile mechanism? This is the default but may not work with older versions of Moodle (<2.5). If set to `FALSE` supplements like graphics and data are included as data URIs.

`solution` logical. Should the question solution, if available, be added in the question XML?

`shuffle` For `mchoice` and `schoice` exercises, if set to `TRUE` will force Moodle to additionally shuffle the provided answer list.

`penalty` numeric. Specifies the penalty tag for a question.

`answernumbering` character. Specifies how choice questions should be numbered.

`usecase` logical. Should string questions be case sensitive or not.

`cloze_mchoice_display` character. In `cloze` type questions, the user may set the visual appearance of choice questions in either a drop down menue (default) `"MULTICHOICE"`, vertical alignment `"MULTICHOICE_V"` or horizontal `"MULTICHOICE_H"`.

`truefalse` character of length 2. For single choice answers in `cloze` questions, the user may specify the possible options shown.

`enumerate` logical. In `cloze` questions, if set to `TRUE`, the answerlist and solutionlist will be enumerated.

`abstention` character or logical. Should an explicit abstention option be added in single/multiple choice exercises? The character text specified is used for an extra button in Moodle which (when selected) always leads to zero points.

`eval` named list, specifies the settings for the evaluation policy, see function `exams_eval`.

`essay` logical. Should `string` questions be rendered into Moodle `shortanswer` or `essay` questions? The default is to use `shortanswer` unless either `essay=TRUE` or the exercise's metainformation is set to `essay`.

`converter, ...` arguments passed on to `make_exercise_transform_html`. The default for `converter` is set to `"ttm"` unless there are Rmd exercises in `file` where `"pandoc"` is used.

Details

exams2moodle produces an XML file that may be uploaded into Moodle. It proceeds by (1) calling xweave on each exercise, (2) reading the resulting LaTeX code, (3) transforming the LaTeX code to HTML, and (4) embedding the HTML code in a XML file using the Moodle standards for exams/quizzes.

For steps (1) and (2) the standard drivers in xexams are used. In step (3), a suitable transformation function is set up on the fly using make_exercise_transform_html, see also the details section in exams2html.

For step (4), the function will cycle through all questions and exams to generate the final XML file in Moodle standard. The structure of the resulting XML file is such that one category will be set for the exam/quiz using the exam/quiz name (or this category may be suppressed (i.e., not included in the XML) by setting iname = FALSE), followed by one category/section for each question, while the replicates of each question will be included in the corresponding category/section. Note that category/section names may also be provided in the \exsection{} tag in the .Rnw files, or within argument stitle in exams2moodle. This may be useful when questions should automatically be added to already existing Moodle question banks. (See also the argument descriptions above.)

The XML code for each question is then generated using function make_question_moodle. Note that for each question type, either the arguments of make_question_moodle may be set within num, mchoice, schoice, string and cloze in exams2moodle, by providing a named list of specifications that should be used, or for each questiontype, a function that produces the question XML code may be provided to num, mchoice, schoice, string and cloze. E.g., to suppress the solution for numeric questions one may set num = list(solution = FALSE).

When specifying cloze exercises, two approaches are possible: Either a answerlist with all questions is provided within the question or, alternatively, the answer fields can be placed anywhere in the question text. For the latter, the strings ##ANSWER1##, ##ANSWER2##, etc., have to be used, see the exercises "boxhist2.Rnw" and "fourfold2.Rnw" for illustration and Appendix C in Zeileis et al. (2014) for further details.

To fix the width of numeric answer fields withing cloze exercises (in order not to convey any clues about the length of the correct solution), the \exextra[numwidth] metainformation command can be used in the .Rnw exercise. For example, it can be set to \exextra[numwidth,logical]{TRUE}, \exextra[numwidth,numeric]{5}, or \exextra[numwidth,character]{100.0}.

In order to generate free text questions in moodle one may specify extra parameters via \exextra. Currently the following options are supported:

- essay: logical. Enables the essay function.
- format: character. Type of text field (one of: plain, editor, editorfilepicker monospaced noinline)
- required: logical. Whether an answer is required.
- attachments: numeric. How many attachments can be uploaded.
- attachmentsrequired: numeric. The number of required attachments.

Value

exams2moodle returns a list of exams as generated by xexams.

make_question_moodle returns a function that generates the XML code for the question in Moodle's XML standard.

References

Dougiamas M, et al. (2015). *Moodle, Version 2.8*. http://moodle.org/.

MoodleDocs (2015). *Moodle XML Format*. http://docs.moodle.org/en/Moodle_XML

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. http://www.jstatsoft.org/v58/i01/.

See Also

xexams, ttm, tth, tex2image, make_exercise_transform_html,

Examples

```
## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## define an exams (= list of exercises)
myexam <- list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  c("scatterplot", "boxhist"),
  "relfreq"
)

## output directory
mydir <- tempdir()

## generate moodle quiz in temporary directory
## using a few customization options
exams2moodle(myexam, n = 3, dir = mydir,
  num = list(solution = FALSE),
  mchoice = list(shuffle = TRUE)
)
dir(mydir)
```

---

exams2nops *Generation of Written Exams for Automatic Evaluation*

Description

Generation of exams in PDF format that can be printed, scanned, and evaluated automatically.

Usage

```
exams2nops(file, n = 1L, dir = NULL, name = NULL,
  language = "en", title = "Exam", course = "",
  institution = "R University", logo = "Rlogo.png", date = Sys.Date(),
  replacement = FALSE, intro = NULL, blank = NULL, duplex = TRUE, pages = NULL,
  usepackage = NULL, header = NULL, encoding = "", startid = 1L,
  points = NULL, showpoints = FALSE, samepage = FALSE,
  twocolumn = FALSE, reglength = 7L, ...)

make_nops_template(n, replacement = FALSE, intro = NULL, blank = NULL,
  duplex = TRUE, pages = NULL, file = NULL, nchoice = 5, encoding = "",
  samepage = FALSE, twocolumn = FALSE, reglength = 7L)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file (in exams2nops) and the number of exercises per exam (in make_nops_template), respectively.
dir	character. The default is either display on the screen or the current working directory.
name	character. A name prefix for resulting exams and RDS file.
language	character. Path to a DCF file with a language specification. Currently, the package ships: English ("en"), Dutch ("nl"), French ("fr"), German ("de"), Hungarian ("hu"), Italian ("it"), Romanian ("ro"), Portuguese ("pt"), Spanish ("es").
title	character. Title of the exam, e.g., "Introduction to Statistics".
course	character. Optional course number, e.g., "101".
institution	character. Name of the institution at which the exam is conducted.
logo	character. Path to a logo image. If the logo is not found, it is simply omitted.
date	character or "Date" object specifying the date of the exam.
replacement	logical. Should a replacement exam sheet be included?
intro	character with LaTeX code for optional introduction text on the first page of the exam.
blank	integer. Number of blank pages to be added at the end. (Default is chosen to be half of the number of exercises.) If pages is specified, blank can also be a vector of length two with blank pages before and after the extra pages, respectively.
duplex	logical. Should blank pages be added after the title page (for duplex printing)?
pages	character. Path(s) to additional PDF pages to be included at the end of the exam (e.g., formulary or distribution tables).
usepackage	character. Names of additional LaTeX packages to be included.
header	list. A list of further options to be passed to the LaTeX files.
encoding	character, passed to xweave.
startid	integer. Starting ID for the exam numbers (defaults to 1).

points	integer. How many points should be assigned to each exercise? Note that this argument overules any exercise points that are provided within the expoints tags of the exercise files (if any). The vector of points supplied should either have length 1 or the number of exercises in the exam.
showpoints	logical. Should the PDF show the number of points associated with each exercise (if specified in the Rnw/Rmd exercise or in points)?
samepage	logical. Should the itemized question lists be forced to be on the same page?
twocolumn	logical. Should a two-column layout be used?
reglength	integer. Number of digits in the registration ID. The default is 7 and it can be increased up to 10.
...	arguments passed on to exams2pdf.
nchoice	character. The number of choice alternatives per exercise.

Details

exams2nops is a convenience interface for exams2pdf with a dynamically generated title page which can be printed, scanned with nops_scan and evaluated automatically by nops_eval. It is originally intended for single- and multiple choice (schoice/mchoice) questions only but has also some limited support for open-ended (string) questions.

The exam sheet consists of various sections where information is either printed our filled in by the students. The section with personal data is just for human readers, it is not read automatically. The registration number has to be filled in in digits and also marked with corresponding crosses where only the latter is read automatically. The exam ID/type/scrambling are printed directly into the PDF and read automatically after scanning. Note that the font in the PDF must not be modified for the reading step to work reliably. (A sans-serif font is used and hence the sfmath LaTeX package is also used - if it is installed.) The questions can have up to five alternatives which have to answered by the students. The crosses are read automatically where both empty and completely filled boxes are regarded as not crossed.

The examples below show how PDF exams can be generated along with an RDS file with (serialized) R data containing all meta-information about the exam. The PDFs can be printed out for conducting the exam and the exam sheet from the first page then needs to be scanned into PDF or PNG images. Then the information from these scanned images can be read by nops_scan, extracting information about the exam, the participants, and the corresponding answers (as described above). The ZIP file produced by nops_scan along with the RDS of the exam meta-information and a CSV file with participant information can then be used by nops_eval to automatically evaluate the whole exam and producing HTML reports for each participant. See nops_eval for a worked example.

Currently, up to three open-ended string questions can also be included. These do not generate boxes on the first exam sheet but instead a second exam sheet is produced for these open-ended questions. It is assumed that a human reader reads these open-ended questions and then assigns points by marking boxes on this separate sheet. Subsequently, this sheet can also be read by nops_scan.

Value

A list of exams as generated by xexams is returned invisibly.

Examples

```
## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## define an exam (= list of exercises)
myexam <- list(
  "tstat2.Rnw",
  "ttest.Rnw",
  "relfreq.Rnw",
  "anova.Rnw",
  c("boxplots.Rnw", "scatterplot.Rnw"),
  "cholesky.Rnw"
)

if(interactive()) {
## compile a single random exam (displayed on screen)
exams2nops(myexam, duplex = FALSE, language = "de")
}

## create multiple exams on the disk (in a
## temporary directory)
mydir <- tempdir()

## generate NOPS exam in temporary directory
set.seed(403)
ex1 <- exams2nops(myexam, n = 2, dir = mydir)
dir(mydir)

## use a few customization options: different
## university/logo and language/title
## with a replacement sheet but for non-duplex printing
set.seed(403)
ex2 <- exams2nops(myexam, n = 2,
  institution = "Universit\"at Innsbruck",
  name = "uibk", logo = "uibk-logo-bw.png",
  title = "Klausur", language = "de",
  replacement = TRUE, duplex = FALSE)
dir(mydir)
```

---

exams2pandoc *Generation of Exams via Pandoc*

Description

Automatic generation of exams via pandoc, by default in docx format.

Usage

```
exams2pandoc(file, n = 1L, nsamp = NULL, dir = ".",
  name = "pandoc", type = "docx", template = "plain.tex",
  question = "Question", solution = "Solution",
  header = list(Date = Sys.Date()), inputs = NULL, options = NULL,
 quiet = TRUE, resolution = 100, width = 4, height = 4, svg = FALSE, encoding = "",
  edir = NULL, tdir = NULL, sdir = NULL, verbose = FALSE, points = NULL, ...)
```

Arguments

`file`	character. A specification of a (list of) exercise files.
`n`	integer. The number of copies to be compiled from `file`.
`nsamp`	integer. The number(s) of exercise files sampled from each list element of `file`. Sampling without replacement is used if possible. (Only if some element of `nsamp` is larger than the length of the corresponding element in `file`, sampling with replacement is used.)
`dir`	character specifying the output directory (default: current working directory). If only a single HTML file is produced and no `dir` is explicitly specified, the file is displayed in the browser rather than saved in `dir`.
`name`	character. A name prefix for resulting exercises.
`type`	character. The file type to convert to using pandoc. The default is `"docx"` (but other choices are also supported, e.g., `"odt"`, `"html"`, `"markdown"` etc.).
`template`	character. A specification of a template in either LaTeX, HTML, or Markdown format. The default is to use the `"plain.tex"` file provided but an alternative `"plain.html"` is also available.
`question`	character or logical. Should the question be included in the output? If `question` is a character it will be used as a header for resulting questions.
`solution`	character or logical, see argument `question`.
`header`	list. A list of named character strings (or functions generating such) to be substituted in the `template`.
`inputs`	character. Names of files that are needed as inputs for the `template` (e.g., images, headers). Either the full path must be given or the file needs to be in `edir`.
`options`	character. A string of options to be passed on to `pandoc_convert`.
`quiet`	logical. Should output be suppressed when calling `xweave`?
`resolution, width, height`	numeric. Options for rendering PNG (or SVG) graphics passed to `xweave`.
`svg`	logical. Should graphics be rendered in SVG or PNG (default)?
`encoding`	character, passed to `xweave`.
`edir`	character specifying the path of the directory in which the files in `file` are stored (see also below).
`tdir`	character specifying a temporary directory, by default this is chosen via `tempdir`. Note that this is cleaned up and potentially temporary files are deleted.

sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.
verbose	logical. Should information on progress of exam generation be reported?
points	integer. How many points should be assigned to each exercise? Note that this argument overules any exercise points that are provided within the expoints tags of the exercise files (if any). The vector of points supplied should either have length 1 or the number of exercises in the exam.
...	currently not used.

Details

exams2pandoc can generate exams in various output formats (by default docx) using xexams and pandoc_convert. It proceeds by (1) calling xweave on each exercise, (2) reading the resulting LaTeX or Markdown code, (3) transforming the code to the markup of some exam template (either LaTeX, HTML, or Markdown), (4) embedding the code in a template and converting it to the desired output format using pandoc.

For steps (1) and (2) the standard drivers in xexams are used.

For step (3) a suitable transformation function is set up on the fly using make_exercise_transform_pandoc. Depending on which format the template uses (LaTeX or HTML or Markdown) the transformation may or may not be trivial.

For step (4) all exercises are inserted into the template (and also replacing certain additional tags from header) and then pandoc_convert is used to convert to the desired output format (one file for each exam). In principle, all output types of pandoc are supported, but most of them have not been tested. (The main motivation for exams2pandoc was the generation of "docx" or "odt" files.)

Value

exams2pandoc returns a list of exams as generated by xexams.

See Also

xexams, pandoc_convert

Examples

```
## load package and enforce par(ask = FALSE)
options(device.ask.default = FALSE)

## define an exams (= list of exercises)
myexam <- list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  c("scatterplot", "boxhist"),
  "relfreq"
)

## output directory
mydir <- tempdir()
```

```
## compile two docx and odt versions each
set.seed(1090)
exams2pandoc(myexam, n = 2, dir = mydir, type = "docx")
set.seed(1090)
exams2pandoc(myexam, n = 2, dir = mydir, type = "odt")
```

---

exams2pdf *Generation of Exams in PDF Format*

Description

Automatic generation of exams in PDF format.

Usage

```
exams2pdf(file, n = 1L, nsamp = NULL, dir = ".", template = NULL,
  inputs = NULL, header = list(Date = Sys.Date()), name = NULL,
  control = NULL, encoding = "", quiet = TRUE, transform = NULL,
  edir = NULL, tdir = NULL, sdir = NULL, verbose = FALSE,
  points = NULL, ...)

make_exams_write_pdf(template = "plain", inputs = NULL,
  header = list(Date = Sys.Date()), name = NULL, quiet = TRUE,
  control = NULL)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
dir	character specifying the output directory (default: current working directory). If only a single PDF file is produced and no dir is explicitly specified, the file is displayed on the screen rather than saved in dir.
template	character. A specification of a LaTeX template. The package currently provides "exam", "solution", "plain", among others. The default is to use the "plain.tex" file unless there are Rmd exercises in file for which "plain8.tex" is used. For further details see below.
inputs	character. Names of files that are needed as inputs during LaTeX compilation (e.g., style files, headers). Either the full path must be given or the file needs to be in edir.

header	list. A list of further options to be passed to the LaTeX files.
name	character. A name prefix for resulting exercises, by default chosen based on template.
control	A list of control arguments for the appearance of multiple choice results (see details).
encoding	character, passed to xweave.
quiet	logical. Should output be suppressed when calling xweave and texi2dvi.
transform	function. An optional transform driver passed to xexams (by default no transformation is used).
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.
verbose	logical. Should information on progress of exam generation be reported?
points	integer. How many points should be assigned to each exercise? Note that this argument overules any exercise points that are provided within the expoints tags of the exercise files (if any). The vector of points supplied should either have length 1 or the number of exercises in the exam.
...	further arguments passed on to xweave.

Details

exams2pdf is a more flexible re-implementation of the old (version 1) exams function (Gruen and Zeileis 2009), using the new extensible xexams framework (Zeileis et al. 2014). A detailed introduction is provided in vignette("exams", package = "exams"), also pointing out relative advantages of the new interface.

exams2pdf proceeds by using make_exams_write_pdf to set up a custom driver$write function on the fly before calling xexams. This custom driver combines each exams with the desired template (and inputs etc.) and then calls texi2dvi on the resulting LaTeX file to produce PDF output. For a single exam (n = 1) the resulting PDF is displayed on screen (unless dir is explicitly specified) while for n > 1 the PDF files are stored in the output directory dir.

Value

exams2pdf returns a list of exams as generated by xexams.

make_exams_write_pdf returns a function that is suitable for being supplied as driver$write to xexams.

References

Gruen B, Zeileis A (2009). Automatic Generation of Exams in R. *Journal of Statistical Software*, **29**(10), 1–14. http://www.jstatsoft.org/v29/i10/.

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. http://www.jstatsoft.org/v58/i01/.

See Also

xexams, exams, texi2dvi

Examples

```
## load package and enforce par(ask = FALSE)
options(device.ask.default = FALSE)

if(interactive()) {
## compile a single random exam (displayed on screen)
exams2pdf(list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  "scatterplot",
  "relfreq"
))
}
```

---

exams2qti12 *Generation of Exams in IMS QTI 1.2 and 2.1 Format*

Description

Automatic generation of exams in IMS QTI 1.2 and 2.1 format.

Usage

```
exams2qti12(file, n = 1L, nsamp = NULL, dir = ".",
  name = NULL, quiet = TRUE, edir = NULL,
  tdir = NULL, sdir = NULL, verbose = FALSE,
  resolution = 100, width = 4, height = 4, svg = FALSE, encoding = "",
  num = NULL, mchoice = NULL,
  schoice = mchoice, string = NULL, cloze = NULL,
  template = "qti12", duration = NULL,
  stitle = "Exercise", ititle = "Question",
  adescription = "Please solve the following exercises.",
  sdescription = "Please answer the following question.",
  maxattempts = 1, cutvalue = 0, solutionswitch = TRUE,
  zip = TRUE, points = NULL,
  eval = list(partial = TRUE, negative = FALSE),
  converter = NULL, xmlcollapse = FALSE, ...)
```

```
exams2qti21(file, n = 1L, nsamp = NULL, dir = ".",
  name = NULL, quiet = TRUE, edir = NULL,
  tdir = NULL, sdir = NULL, verbose = FALSE,
  resolution = 100, width = 4, height = 4, svg = FALSE, encoding  = "",
  num = NULL, mchoice = NULL,
  schoice = mchoice, string = NULL, cloze = NULL,
  template = "qti21", duration = NULL,
  stitle = "Exercise", ititle = "Question",
  adescription = "Please solve the following exercises.",
  sdescription = "Please answer the following question.",
  maxattempts = 1, cutvalue = 0, solutionswitch = TRUE,
  zip = TRUE, points = NULL,
  eval = list(partial = TRUE, negative = FALSE),
  converter = NULL, base64 = TRUE, mode = "hex", ...)

make_itembody_qti12(rtiming = FALSE, shuffle = FALSE,
  rshuffle = shuffle, minnumber = NULL, maxnumber = NULL,
  defaultval = NULL, minvalue = NULL, maxvalue = NULL,
  cutvalue = NULL, enumerate = TRUE, digits = NULL,
  tolerance = is.null(digits), maxchars = 12,
  eval = list(partial = TRUE, negative = FALSE),
  fix_num = TRUE)

make_itembody_qti21(shuffle = FALSE, defaultval = NULL,
  minvalue = NULL, maxvalue = NULL, enumerate = TRUE,
  digits = NULL, tolerance = is.null(digits), maxchars = 12,
  eval = list(partial = TRUE, negative = FALSE),
  solutionswitch = TRUE)
```

Arguments

file	character. A specification of a (list of) exercise files.
n	integer. The number of copies to be compiled from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
dir	character. The default is the current working directory.
name	character. A name prefix for resulting exercises and ZIP file.
quiet	logical. Should output be suppressed when calling xweave?
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.

verbose — logical. Should information on progress of exam generation be reported?

resolution, width, height — numeric. Options for rendering PNG (or SVG) graphics passed to xweave.

svg — logical. Should graphics be rendered in SVG or PNG (default)?

encoding — character, passed to xweave.

num — function or named list applied to numerical (i.e., type num) questions. If num is a function, num will be used for generating the item body of the question, see function make_itembody_qti12() (or make_itembody_qti21()). If num is a named list, these arguments will be passed to function make_itembody_qti12() (or make_itembody_qti21() using exams2qti21()).

mchoice, schoice, string, cloze — function or named list applied to multiple choice, single choice, string, and cloze questions (i.e., type mchoice, schoice, string, and cloze), respectively. See argument num for more details.

template — character. The IMS QTI 1.2 or 2.1 template that should be used. Currently, the package provides "qti12.xml" and "qti21.xml".

duration — integer. Set the duration of the exam in minutes.

stitle — character. A title that should be used for the sections. May be a vector of length 1 to use the same title for each section, or a vector containing different section titles.

ititle — character. A title that should be used for the assessment items. May be a vector of length 1 to use the same title for each item, or a vector containing different item titles. Note that the maximum of different item titles is the number of sections/questions that are used for the exam.

adescription — character. Description (of length 1) for the overall assessment (i.e., exam).

sdescription — character. Vector of descriptions for each section.

maxattempts — integer. The maximum attempts for one question, may also be set to Inf.

cutvalue — numeric. The cutvalue at which the exam is passed.

solutionswitch — logical. Should the question/item solutionswitch be enabled? In OLAT this means that the correct solution is shown after an incorrect solution was entered by an examinee (i.e., this is typically only useful if maxattempts = 1).

zip — logical. Should the resulting XML file (plus supplements) be zipped?

points — integer. How many points should be assigned to each exercise? Note that this argument overules any exercise points that are provided within an "\expoints{}" tag in the .Rnw file. The vector of points supplied is expanded to the number of exercises in the exam.

eval — named list, specifies the settings for the evaluation policy, see function exams_eval.

base64 — logical. Should images be embedded using Base 64 coding? Argument base64 may also be a character vector of file endings that should be Base 64 encoded, e.g. base64 = c("png", "rda") will only encode PNG images and binary .rda files.

mode — character. See function tth.

`rtiming, shuffle, rshuffle, minnumber, maxnumber, defaultval, minvalue, maxvalue`
: arguments used for IMS QTI 1.2 item construction, for details see the XML specification (see IMS Global Learning Consortium, Inc. 2012), especially Section 4. Genereting IMS QTI 2.1 items using `exams2qti21()` the arguments have similar meaning.

`enumerate`
: logical. Insert potential solutions in enumerated list?

`digits`
: integer. How many digits should be used for `num` exercises?

`tolerance`
: logical. Should tolerance intervals be used for checking if the supplied `num` answer/number is correct? The default is to use tolerance intervals if `digits = NULL`.

`maxchars`
: numeric. Lower bound for the number of characters in fill-in-blank fields. The actual number of characters is selected as the maximum number of characters of this value and the actual solution.

`fix_num`
: logical. This is a special flag to enable/force the display of the correct solutions for numeric exercises/answers as well as to obtain results when archiving tests. Note that this is a workaround, which works e.g. within OLAT.

`converter`
: character. Argument passed on to `make_exercise_transform_html`. The default for `converter` is set to `"ttm"` unless there are Rmd exercises in `file` where `"pandoc"` is used.

`xmlcollapse`
: logical or character. Should line breaks be collapsed in the XML code. If `TRUE` everything is collapsed with spaces (`" "`) but other collapse characters could be supplied.

`...`
: further arguments passed on to `make_exercise_transform_html`.

Details

The Question & Test Interoperability (QTI) is an international XML standard for specifying e-learning tests (IMS Global Learning Consortium, Inc. 2012ab). The standard evolved over various versions with the first release culminating in the QTI 1.2 standard and the stable version of the second release currently at QTI 2.1. While both versions share many similarities, they differ in many details. Hence, separate functions `exams2qti12` and `exams2qti21` are provided. The former has already been thoroughly tested and the latter is still in beta testing stage and might change in future releases.

`exams2qti12` produces a `.zip` file that may be uploaded (e.g. in OLAT). This includes the final XML file of the exam/assessment as well as possible supplement folders that include images, data sets etc. used for the exam. It proceeds by (1) calling `xweave` on each exercise, (2) reading the resulting LaTeX code, (3) transforming the LaTeX code to HTML, and (4) embedding the HTML code in a XML file using the IMS QTI 1.2 standards for assessments and question items.

For steps (1) and (2) the standard drivers in `xexams` are used. In step (3), a suitable transformation function is set up on the fly using `make_exercise_transform_html`, see also the details section in `exams2html`.

For step (4), the function will cycle through all questions and exams to generate the final XML file in IMS QTI 1.2 standard. Therefore, each question will be included in the XML as one section. The replicates of each question will be written as question items of the section.

The function uses the XML template for IMS QTI 1.2 assessments and items to generate the exam (per default, this is the XML file `qti12.xml` provided in the `xml` folder of this package). The

assessment template must provide one section including one item. exams2qti12 will then use the single item template to generate all items, as well as the assessment and section specifications set within the template.

The default template will generate exams/assessments that sample one replicate of a question/item for each section. The usual procedure in exam/assessment generation would be to simply copy \& paste the XML template of the package and adapt it to the needs of the user. Note that all specifiers that have a leading ## in the XML template will be replaced by suitable code in exams2qti12 and should always be provided in the template. I.e., the user may add additional tags to the XML template or modify certain specifications, like the number of replicates/items that should be sampled for each section etc.

Per default, the individual question/item bodies are generated by function make_itembody_qti12, i.e. make_itembody_qti12 checks the type of the question and will produce suitable XML code. Note that for each question type, either the arguments of make_itembody_qti12 may be set within num, mchoice, schoice, string and cloze in exams2qti12, by providing a named list of specifications that should be used, or for each questiontype, a function that produces the item body XML code may be provided to num, mchoice, schoice, string and cloze. E.g., mchoice = list(shuffle = TRUE) will force only multiple choice questions to have a shuffled answerlist.

Note that in OLAT/OpenOLAT num exercises are not officially supported but in fact work correctly. The only drawback is that in certain settings the correct solution is not shown at the end of the assessment (although it is used for all internal computations). Therefore, two workarounds are implemented. Either fix_num can be set to TRUE (default), then a fix is added by double-checking the result, or digits can be set to a fixed value (e.g., digits = 2). In the latter case, the num exercise is represented by a string. Then the answer must be provided exactly to the decimal places specified (e.g., if the exact solution is 16.4562, then the correct answer in the test will be "16.46", i.e., a character string of 5 characters).

Generating exams/assessment in IMS QTI 2.1 format using exams2qti21() and make_itembody_qti21() is performed in a similar way as described above. Note that the IMS QTI 2.1 generators are still work in progress. The generated XML files have been validated using the IMS validator provided at http://membervalidator.imsglobal.org/qti/ (when it was still freely available). Furthermore, a selection of generated exams/assessments has been tested using the ONYX Editor and Player, see https://www.onyx-editor.de/.

Value

exams2qti12 and exams2qti21 return a list of exams as generated by xexams.

make_itembody_qti12 and make_itembody_qti21 return a function that generates the XML code for the itembody tag in IMS QTI 1.2 and IMS QTI 2.1 format.

References

IMS Global Learning Consortium, Inc. (2012a). *IMS Question & Test Interoperability: ASI XML Binding Specification Final Specification Version 1.2*. http://www.imsglobal.org/question/qtiv1p2/imsqti_asi_bindv1p2.html

IMS Global Learning Consortium, Inc. (2012b). *IMS Question & Test Interoperability (QTI) XSD Binding Version 2.1 Final*. http://www.imsglobal.org/question/qtiv2p1/imsqti_bindv2p1.html

BPS Bildungsportal Sachsen GmbH (2014). *ONYX Testsuite*. http://www.bps-system.de/cms/en/products/onyx-testsuite/

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. http://www.jstatsoft.org/v58/i01/.

See Also

xexams, ttm, tth, tex2image, make_exercise_transform_html,

Examples

```
## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## define an exams (= list of exercises)
myexam <- list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  c("scatterplot", "boxhist"),
  "relfreq"
)

## output directory
mydir <- tempdir()

## generate .zip with QTI 1.2 exam in temporary directory
## using a few customization options
exams2qti12(myexam, n = 3, dir = mydir,
  maxattempts = 3,
  num = list(digits = 1),
  mchoice = list(shuffle = TRUE, enumerate = FALSE)
)
dir(mydir)
```

---

exams2tcexam *Generation of Exams in TCExam Format*

Description

Interface for generating exams in TCExam format.

Usage

```
exams2tcexam(file, n = 1L, nsamp = NULL, dir = ".",
  name = NULL, quiet = TRUE, edir = NULL, tdir = NULL, sdir = NULL, verbose = FALSE,
  resolution = 100, width = 4, height = 4, svg = FALSE, encoding = "", points = NULL,
```

```
  modulename = name, subjectname = name, subjectdescription = NULL, timer = 0,
fullscreen = FALSE, inlineanswers = FALSE, autonext = FALSE, shuffle = FALSE,
 lang = "en", date = Sys.time(), zip = FALSE, converter = NULL, ...)
```

Arguments

`file`	character. A specification of a (list of) exercise files.
`n`	integer. The number of copies to be compiled from `file`.
`nsamp, quiet, edir, tdir, sdir, verbose`	arguments passed to `xexams`.
`dir`	character specifying the output directory path. The default is the current working directory.
`name`	character. A name prefix for resulting XML file.
`resolution, width, height, svg, encoding`	arguments passed to `xweave`.
`points`	numeric. Number of points for the questions.
`modulename`	character. Module name.
`subjectname`	character. Subject name.
`subjectdescription`	character. Subject description.
`timer`	numeric. Number of seconds for each question.
`fullscreen`	logical. Should the question be shown in full-screen mode?
`inlineanswers`	logical. Should the question list be presented inline?
`autonext`	logical. Automatically advance to the next item?
`shuffle`	logical. Should the question list of schoice/mchoice answers be shuffled (or kept fixed)?
`lang`	character. Two-letter indicator of the language.
`date`	character or `"Date"` object specifying the date of the exam.
`zip`	logical. Should the resulting XML file be zipped?
`converter, ...`	arguments passed on to `make_exercise_transform_html`. The default for `converter` is set to `"ttm"` unless there are Rmd exercises in `file` where `"pandoc"` is used.

Details

`exams2tcexam` generates XML exams that can be imported into the TCExam software of Asuni (2012). Currently, the subset of HTML(-like) commands that is supported in TCExam is rather limited, e.g., tables and figures cannot be directly included.

Value

A list of exams as generated by `xexams` is returned invisibly.

References

Asuni (2012). *TCExam: Computer-Based Assessment Software*. http://tcexam.org/.

Examples

```
## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## Not run:
## exams2tcexam creates a single XML file
exams2tcexam("tstat2", n = 2)

## End(Not run)
```

---

exams_eval *Auxiliary Tools for Evaluating Exams*

Description

Generation various helper functions for evaluating exams.

Usage

```
exams_eval(partial = TRUE, negative = FALSE,
  rule = c("false2", "false", "true", "all", "none"))
```

Arguments

partial	logical. Should single/multiple-choice answers be evaluated as a whole pattern (partial = FALSE) or should partial credits be assigned to each of the choices (partial = TRUE)?
negative	logical or numeric. Handling of negative points for an exercise, for details see below.
rule	character specifying which rule to use for negative partial credits.

Details

The function exams_eval is a convenience wrapper for specifying various types of evaluation policies. It returns a set of auxiliary functions that may be useful in the evaluation of exams.

Exercises of types "num" or "string" can essentially be just correct or wrong. In the former case they will give 100 percent of all points, in the latter either 0 percent or some negative percentage can be assigned. If negative percentages are used (e.g., negative = 0.25), then it needs to be distinguished between solved incorrectly and not attempted to solve (which should yield 0 percent).

However, for multiple-choice answers the evaluation policy can either pertain to the answer pattern as a whole (which can be correct or wrong, see above) or it can employ a partial credit strategy. In

the latter case, each selected correct choice will yield the fraction 1/ncorrect of points. When an incorrect choice is selected, it should lead to negative points. Five strategies are currently implemented: `"false"` uses 1/nwrong while `"false2"` uses 1/max(nwrong, 2); `"true"` uses 1/ncorrect (so that each wrong selection cancels one correct selection); `"all"` uses 1 (so that a single wrong selection cancels all correct selections); and `"none"` uses 0 (so that wrong selections have no effect at all). When aggregating the partial percentages, the overall points can become negative. By setting `negative` a lower bound can be set: `negative = TRUE` sets no bound while `negative = FALSE` sets the bound to zero. Any other numeric value could be set as well, e.g., `negative = 0.25`.

The functions returned by `exams_eval` internally just distinguish between `num`, `string`, and `mchoice` answers. Thus, if evaluations for `schoice` or `cloze` exercises are required, these have to be built by appropriately reusing the building blocks for `num`/`string`/`mchoice`. For example, the components of `cloze` exercises have to be evaluated individually and then aggregated as desired. Or, if a distinction between `mchoice` and `schoice` regarding partial credits is needed, one evaluation has to be set up with `partial = TRUE` and the other with `partial = FALSE`. Different evaluations for different item types may be set as in: `exams2qti12(..., eval = eval1, schoice = list(eval = eval2))`. Then `eval = eval1` is used as the default for all exercise types except `schoice` where `eval = eval2` is used.

Thus, `exams_eval` might not give the complete finished evaluation policy for an entire exam but supplies the most important building blocks for setting this up "by hand". Internally, `exams_eval` is also used by `exams2moodle`, `exams2qti12` and `exams2blackboard` for writing the evaluation specifications in the respective XML specifications.

Value

`exams_eval` returns a list with the input parameters `partial`, `negative`, and `rule` along with the following functions:

`checkanswer`	function with arguments (`correct`, `answer`, and `tolerance = 0`. It checks whether `answer` (sufficiently) matches `correct` or not. It returns `1` for correct, `-1` for wrong and `0` for not attempted. In case of `partial = TRUE`, the functions returns a vector for multiple-choice questions.
`pointvec`	function with argument `correct = NULL`. It computes the vector of points for correct and wrong answers, respectively.
`pointsum`	function with arguments (`correct`, `answer`, and `tolerance = 0`. It computes the overall number of points.

See Also

`exams2moodle`, `exams2qti12`, `exams2blackboard`

Examples

```
## binary evaluation policy with solutions being either correct
## or wrong: partial = FALSE, negative = FALSE
ee <- exams_eval(partial = FALSE, negative = FALSE)

## points that can be achieved are 0/1
ee$pointvec()
```

```
## checkanswer() returns 1 for correct, -1 for incorrect and 0 for missing answer
ee$checkanswer(1.23, 1.23)
ee$checkanswer(1.23, "1.23")
ee$checkanswer(1.23, "1,23")
ee$checkanswer(1.23, 1.24)
ee$checkanswer(1.23, 1.24, tolerance = 0.01)
ee$checkanswer(1.23, NA)
ee$checkanswer(1.23, NULL)
ee$checkanswer(1.23, "")

## similarly for logical (mchoice/schoice) answers
## (which allows either string or logical specification)
ee$checkanswer("10000", "10000")
ee$checkanswer(c(TRUE, FALSE, FALSE, FALSE, FALSE), c(TRUE, FALSE, FALSE, FALSE, FALSE))
ee$checkanswer(c(TRUE, FALSE, FALSE, FALSE, FALSE), "10000")
ee$checkanswer("10000", "01000")
ee$checkanswer("10000", "11000")

## and analogously for strings
ee$checkanswer("foo", "foo")
ee$checkanswer("foo", "bar")
ee$checkanswer("foo", "")

## obtain points achieved
ee$pointsum("10000", "10000")
ee$pointsum("10000", "01000")
ee$pointsum("10000", "00000")
ee$pointsum("10000", NA)

## ------------------------------------------------------------
## evaluation policy with -25% penalty for wrong answers
ee <- exams_eval(partial = FALSE, negative = -0.25)

## points that can be achieved are 1/-0.25 (or zero)
ee$pointvec()

## obtain points achieved
ee$pointsum("10000", "10000")
ee$pointsum("10000", "01000")
ee$pointsum("10000", "00000")
ee$pointsum("10000", NA)
ee$pointsum(1.23, 1.23)
ee$pointsum(1.23, 2.34)
ee$pointsum(1.23, NA)
ee$pointsum(1.23, 1.24)
ee$pointsum(1.23, 1.24, tolerance = 0.1)

## ------------------------------------------------------------
## default evaluation policy with partial points
## (but without negative points overall)
ee <- exams_eval()

## points that can be achieved are 1/3 (1/#true)
```

```
## or -1/2 (1/#false)
ee$pointvec("10101")

## obtain points achieved
ee$pointsum("10101", "10101")
ee$pointsum("10101", "10100")
ee$pointsum("10101", "11100")
ee$pointsum("10101", "01010")
ee$pointsum("10101", "00000")

## show individual answer check
ee$checkanswer("10101", "10101")
ee$checkanswer("10101", "10100")
ee$checkanswer("10101", "11100")
ee$checkanswer("10101", "01010")
ee$checkanswer("10101", "00000")

## numeric/string answers are not affected by partial=TRUE
ee$checkanswer(1.23, 1.23)
ee$pointsum(1.23, 1.23)
ee$checkanswer(1.23, 2.34)
ee$pointsum(1.23, 2.34)

## ------------------------------------------------------------
## evaluation policy with partial points
## (and with up to -25% negative points overall)
ee <- exams_eval(partial = TRUE, negative = -0.25)

## points that can be achieved are 1/3 (1/#true)
## or -1/2 (1/#false)
ee$pointvec("10101")

## obtain points achieved
ee$pointsum("10101", "10101")
ee$pointsum("10101", "01010")
ee$pointsum("10101", "00000")

## show individual answer check
ee$checkanswer("10101", "10101")
ee$checkanswer("10101", "10100")
ee$checkanswer("10101", "11100")
ee$checkanswer("10101", "01010")
ee$checkanswer("10101", "00000")

## numeric/string answers are not affected by partial=TRUE
ee$pointsum(1.23, 1.23)
ee$pointsum(1.23, 2.34)
```

`exams_skeleton` *Generate Skeleton for Exams Directory/Script*

Description

Generate a directory structure which contains 'demo-*.R' scripts along with directories containing all available demonstation exercise '.Rnw' or '.Rmd' files and necessary template files (LaTeX, HTML, or XML).

Usage

```
exams_skeleton(dir = ".",
  type = c("num", "schoice", "mchoice", "string", "cloze"),
  writer = c("exams2html", "exams2pdf", "exams2moodle",
    "exams2qti12", "exams2qti21", "exams2arsnova", "exams2nops"),
  markup = "latex", absolute = FALSE, encoding = "")
```

Arguments

dir	character with path to directory. The default is the current working directory.
type	character vector indicating types of exercises that should be included in the 'demo.R' script. By default an example for each type of exercise is included.
writer	character vector indicating the exams2xyz writer functions that should be included in the 'demo.R' script. By default an example for each type of writer is included.
markup	character vector indicating whether the example exercises use "latex" markup (.Rnw files) or "markdown" markup (.Rmd files).
absolute	logical. Should the paths in the 'demo.R' script be absolute? The default is to use relative paths.
encoding	character specifying the encoding to be used in the exams2xyz writer functions.

Details

exams_skeleton (or equivalently exams.skeleton) creates a directory with several 'demo-*.R' scripts illustrating the use of the various exams2xyz interfaces. Subdirectories with copies of all demonstration exercise .Rnw or .Rmd files and templates for different output formats (LaTeX, HTML, or XML) are also created.

This should provide a starting point for users wishing to start their own collection of exercises with **exams**.

The encoding is not used by default. In principle, it can be set to any value that Sweave can work with in the current locale. If set to "UTF-8" (or "utf8"), or "ISO-8859-1" (or "latin1"), or "ISO-8859-15" (or "latin9"), the LaTeX and/or HTML templates are adapted accordingly. For other encodings the templates may need further touch-ups.

Value

exams_skeleton returns a list of character vectors with the demo scripts invisibly.

See Also

exams2html, exams2pdf, exams2moodle, exams2qti12, exams2qti21, exams2arsnova, exams2nops

Examples

```
## output directory (replace this with "/path/to/your/directory")
mydir <- file.path(tempdir(), "myexam")

## create exams skeleton with absolute paths in demo.R
exams_skeleton(dir = mydir, absolute = TRUE)

## look at created files
dir(mydir)
dir(mydir, recursive = TRUE)

## now open demo-all.R or any of the other demo-*.R scripts in your
## favorite R code editor and run the examples...
```

`fmt` *Auxiliary Formatting Functions*

Description

Auxiliary functions for displaying numeric elements in exercises.

Usage

```
fmt(x, digits = 2L, zeros = digits < 4L)

round2(x, digits = 0)

char_with_braces(x)

num_to_tol(x, reltol = 0.0002, min = 0.01, digits = 2)

## S3 method for class 'matrix'
toLatex(object, skip = FALSE, fix = getOption("olat_fix"),
  escape = TRUE, ...)
```

Arguments

`x`	numeric vector.
`digits`	integer. Digits that should be used for rounding.
`zeros`	logical. Should trailing zeros be added?
`reltol`	numeric. Relative tolerance (relative to correct solution `x`).
`min`	numeric. Minimum absolute tolerance.
`object`	matrix.
`skip`	logical. Should an additional skip be added between rows?
`fix`	logical. Should an additional empty column be added between all columns? This is a workaround for OLAT that collapses spaces between columns in MathML.

escape	logical. Should LaTeX commands be escaped (as appropriate for Sweave) or not (as appropriate for knit)?
...	currently not used.

Details

Various functions that help displaying numerical results in exercises:

The function fmt rounds and adds trailing zeros (by default if digits is lower than 4).

The function round2 does what is known in German as kaufmaennisches Runden (rounding away from zero for trailing 5s).

The function char_with_braces adds parentheses for negative elements (in order to facilitate their display in equations).

The function num_to_tol (or equivalently num2tol) computes the absolute tolerance based on a numeric solution x and a relative tolerance reltol.

The toLatex method sets up a matrix array with parentheses.

Examples

```
## emulate how students round
## (rather than using the round-to-even strategy R employs)
round2(c(0.005, 0.015), digits = 2)
round(c(0.005, 0.015), digits = 2)

## this is also employed internally in the fmt() formatting function
fmt(c(0.005, 0.015))

## the main purpose of fmt() is that some numeric result can be displayed
## both at high accuracy and then at the rounding that students should do
## (e.g., with 2 or 3 digits)
sol <- runif(1)
fmt(sol, 6)
fmt(sol, 2)

## but fmt() also assures showing a very high numer of significant digits
## (up to 12)
sol <- 123456 + sol
sol
fmt(sol, 6)
fmt(sol, 2)

## and fmt() also takes care of adding trailing zeros (if digits < 4)
fmt(1)
fmt(1, digits = 3)
fmt(1, digits = 6)

## char_with_braces() is for adding parentheses, e.g., before constructing a sum
paste(char_with_braces(-2:2), collapse = " + ")

## for including a matrix in a LaTeX formula
x <- matrix(1:4, ncol = 2)
```

```
toLatex(x)
toLatex(x, skip = TRUE)

## compute absolute tolerances:
## minimum is 0.01
num_to_tol(1)
## but can be larger for larger solutions
num_to_tol(100)
```

---

include_supplement *Copy (Static) Supplement Files for Inclusion in Exercises*

Description

Copy (static) files (e.g., graphics, data sets, etc.) for inclusion as supplements in an exercise.

Usage

```
include_supplement(file, dir = NULL, recursive = FALSE)
```

Arguments

file	character. A (vector of) file name(s).
dir	character. The directory where file can be found. If used within the code chunks of exercises, the default is to use the directory in which the exercises are stored.
recursive	logical. Should also sub-directories of dir be searched for file?

Details

Usually, supplement files are created dynamically within an exercise, e.g., data is simulated and then plotted or stored in a file etc. However, sometimes an exercises wants to include a static supplement file that is available in some directory on the system. Then, the include_supplement is a convenience function that copies such a file from its directory into the supplements of an exercise. Then it can be included/referenced as usual in the question/solution text.

Examples

```
## The "Rlogo" exercise uses a static image which is provided
## within the "exams" package.
if(interactive()) {
exams2html("Rlogo.Rnw")
}
```

match_exams_call *Query Last xexams/exams2xyz Call*

Description

Query the last call made to xexams (typically through some exams2xyz interface).

Usage

```
match_exams_call(which = 1L, deparse = TRUE)
```

Arguments

which	integer. Specifies the hierarchy level at which the exams2xyz call should be extracted.
deparse	logical. Should only the deparsed function name be computed (or the entire call)?

Details

The function match_exams_call is useful for determining within an exercise which exams2xyz interface is used in order to behave slightly differently, e.g., for PDF vs. HTML output.

This feature only works from R 3.2.0 onwards.

Examples

```
## call exams2nops
tdir <- tempdir()
exams2nops("tstat2.Rnw", dir = tdir)
match_exams_call()

## exams2nops called exams2pdf called xexams:
match_exams_call(which = NULL)

## get full exams2nops call
match_exams_call(deparse = FALSE)

## but note that convenience wrappers etc. are included
e2n <- function(...) exams2nops(...)
e2n("tstat2.Rnw", dir = tdir)
match_exams_call(which = NULL)
```

`matrix_to_schoice` *Generate Single- and Multiple-Choice Question Lists for Matrix Solutions*

Description

Functions for generating single- and multiple-choice question lists for a matrix solution. (Optimized for integer matrices.)

Usage

```
matrix_to_schoice(x, y = NULL, lower = FALSE, name = "a",
  delta = 0.5, digits = 0)

matrix_to_mchoice(x, y = NULL, lower = FALSE, name = "a",
  comparisons = c("==", "<", ">", "<=", ">="))

det_to_schoice(x, y = NULL, range = NULL, delta = 0.5, digits = 0)
```

Arguments

`x`	matrix (correct result).
`y`	numeric vector (optional) with (potentially) wrong solutions/comparisons.
`lower`	logical. Should only elements from the lower triangle be assessed?
`name`	character. Base name for matrix elements.
`delta`	numeric. Minimal distance between solutions.
`digits`	integer. Digits that should be displayed.
`comparisons`	character. Vector of logical comparisons that should be employed.
`range`	numeric vector of length 2 (optional) with range of random wrong solutions.

Details

The function `matrix_to_schoice` (or equivalently `matrix2schoice`) can be used for generating a single-choice question list for a correct result matrix `x`. One element is picked randomly from the matrix and chosen to be the correct solution. Other values from the observed absolute range are used as wrong solutions by default (if `y` does not provide an alternative list of potential solutions).

The function `matrix_to_mchoice` (or equivalently `matrix2mchoice`) can be used for generating a multiple-choice question list for a correct result matrix `x`. Each item from the question list is a logical comparison of one matrix element with a comparion value. By default the comparisons are picked randomly from the observed absolute range (unless `y` specifies a different list of comparisons).

The function `det_to_schoice` (or equivalently `det2schoice`) can be used for generating a single-choice question list for the determinant of a 2x2 matrix. It has been optimized for matrices with single-digit integer elements. It may not yield very balanced random solutions for other scenarios.

Value

`matrix_to_schoice/matrix2schoice` returns a list with the following components:

`index`	numeric vector with matrix index of the correct solution chosen.
`name`	character with LaTeX code for the correct matrix element chosen.
`solutions`	a logical vector of length 5 indicating the `correct` solution,
`questions`	a character vector of length 5 with question list.

`matrix_to_mchoice/matrix2mchoice` returns a list with the following components:

`solutions`	a logical vector of length 5 indicating the `correct` solution,
`questions`	a character vector of length 5 with question list.
`explanations`	a character vector of length 5 with explanations why the solutions are correct or wrong.

`det_to_schoice/det2schoice` returns a list with the following components:

`solutions`	a logical vector of length 5 indicating the `correct` solution,
`questions`	a character vector of length 5 with question list.

See Also

`num_to_schoice`

Examples

```
A <- matrix(c(-9, 0, 5, -2), ncol = 2)
matrix_to_schoice(A)
matrix_to_mchoice(A)
det_to_schoice(A)

B <- matrix(1:9, ncol = 3)
matrix_to_schoice(B)
matrix_to_mchoice(B)
```

---

mchoice2string *Convenience Functions for Exam Formatting*

Description

A collection of convenience functions for formatting in exam generation that can be used for switching between suitable logical/text/numeric representations of multiple choice solutions.

Usage

```
mchoice2string(x, single = FALSE)
string2mchoice(x, single = FALSE)
mchoice2text(x, markup = c("latex", "markdown"))
answerlist(..., sep = ". ", markup = c("latex", "markdown"))
```

Arguments

x	an object, see below for examples.
single	logical. Should the function check whether exactly a single answer is true?
...	character vectors to be included in answer lists.
sep	character for separation between vectors, see below for examples.
markup	character indicating which markup (LaTeX vs. Markdown) should be generated.

Details

Three convenience functions for facilitating work with multiple choice solutions of exams. All have almost trivial definitions, see also examples below.

See Also

exams

Examples

```
## multiple choice answer
mc <- c(TRUE, FALSE, TRUE, FALSE, FALSE)

## switching to string representation
mchoice2string(mc)

## reverse string encoding
string2mchoice("10100")

## switching to text
mchoice2text(mc)

## generating answerlist based on questions,
## solutions and explanations
qu <- c("Zurich is the capital of Switzerland.",
        "Italian is an official language in Switzerland.",
        "Switzerland is part of the European Union.")
sol <- c(FALSE, TRUE, FALSE)
ex <- c("The capital of Switzerland is Bern.",
        "The four official languages are: German, French, Italian, Romansh.",
"Switzerland is part of the Schengen Area but not the European Union.")
answerlist(qu)
answerlist(ifelse(sol, "True", "False"), ex)
```

nops_eval *Evaluate NOPS Exams*

Description

Evaluate NOPS exams produced with exams2nops, and scanned by nops_scan.

Usage

```
nops_eval(register = dir(pattern = "\\.csv$"), solutions = dir(pattern = "\\.rds$"),
  scans = dir(pattern = "^nops_scan_[[:digit:]]*\\.zip$"),
  points = NULL, eval = exams_eval(partial = TRUE, negative = FALSE, rule = "false2"),
  mark = c(0.5, 0.6, 0.75, 0.85), dir = ".", results = "nops_eval",
  html = NULL, col = hcl(c(0, 0, 60, 120), c(70, 0, 70, 70), 90),
  encoding = "UTF-8", language = "en", interactive = TRUE,
  string_scans = dir(pattern = "^nops_string_scan_[[:digit:]]*\\.zip$"),
  string_points = seq(0, 1, 0.25))
```

Arguments

register	character. File name of a CSV file (semicolon-separated) of the registered students. Must contain columns "registration" (registration number), "name" (student name), "id" (some user name or other string unique for each student). The file name should not contain spaces, umlaut or other special characters (e.g., something like "exam-2015-07-01.csv" is recommended).
solutions	character. File name of the RDS exam file produced by exams2nops.
scans	character. File name of the ZIP file with scanning results (containing Daten.txt and PNG files) as produced by nops_scan.
points	numeric. Vector of points per exercise. By default read from solutions.
eval	list specification of evaluation policy as computed by exams_eval.
mark	logical or numeric. If mark = FALSE, no marks are computed. Otherwise mark needs to be a numeric vector with threshold values to compute marks. The thresholds can either be relative (all lower than 1) or absolute. In case results exactly matching a threshold, the better mark is used.
dir	character. File path to the output directory (the default being the current working directory).
results	character. Prefix for output files.
html	character. File name for individual HTML report files, by default the same as register with suffix .html.
col	character. Hex color codes used for exercises with negative, neutral, positive, full solution.
encoding	character. Encoding of register, e.g., "latin1" or "UTF-8" (default).

language	character. Path to a DCF file with a language specification. Currently, the package ships: English ("en"), Dutch ("nl"), French ("fr"), German ("de"), Italian ("it"), Romanian ("ro"), Portuguese ("pt"), Spanish ("es").
interactive	logical. Should possible errors in the Daten.txt file by corrected interactively? Requires the **png** package for full interactivity.
string_scans	character. Optional file name of the ZIP file with scanning results of string exercise sheets (if any) containing Daten2.txt and PNG files as produced by nops_scan.
string_points	numeric. Vector of length 5 with points assigned to string results.

Details

nops_eval is a companion function for exams2nops and nops_scan. It evaluates the scanned exams by computing the sums of the points achived and (if desired) maps them to marks. Furthermore a HTML report for each individual student is generated (e.g., for upload into a learning management system).

Value

A data.frame with the detailed exam results is returned invisibly. It is also written to a CSV file in the current directory, along with a ZIP file containing the HTML reports.

See Also

exams2nops, nops_scan

Examples

```
## --- Preliminaries ---

## load package and enforce par(ask = FALSE)
library("exams")
options(device.ask.default = FALSE)

## set up a temporary working directory in which all files are managed
odir <- getwd()
dir.create(mydir <- tempfile())
setwd(mydir)


## --- Step 1 ---
## exam generation

## define an exam (= list of exercises)
myexam <- list(
  "tstat2.Rnw",
  "ttest.Rnw",
  "relfreq.Rnw",
  "anova.Rnw",
  c("boxplots.Rnw", "scatterplot.Rnw"),
```

```
  "cholesky.Rnw"
)

## create multiple exams on the disk with different numbers of points
## per exercise (see ?exams2nops for more examples)
set.seed(403)
ex1 <- exams2nops(myexam, n = 2, dir = ".", date = "2015-07-29",
  points = c(1, 1, 1, 2, 2, 3), showpoints = TRUE)
dir()

## assume the PDF exams were already printed (and possibly backed up
## in a different directory) so that they are not needed anymore
file.remove(dir(pattern = "pdf$"))


## --- Step 2 ---
## scan results

## assume two participants filled out the printed exam sheets
## and the corresponding scans are in two PNG files,
img <- dir(system.file("nops", package = "exams"), pattern = "nops_scan",
  full.names = TRUE)

## copy the PNG files to the working directory
file.copy(img, to = ".")

## read the scanned images (all locally available .png files) and collect
## results in a ZIP archive (see ?nops_scan for more details)
nops_scan()
dir()

## the ZIP archive contains copies of the PNG images so that these are
## can be deleted here (possibly after backup in a different directory)
file.remove(dir(pattern = "png$"))


## -- Step 3 ---
## evaluate results

## three files are required: (a) an RDS file with the exam meta-information
## (see Step 1), (b) a ZIP file with the scanned sheets (see Step 2), (c) a
## CSV file with the student infomation (registration number, name, and some
## for of ID/username)

## here we create the CSV file on the fly but in practice this will typically
## be processed from some registration service or learning management system etc
write.table(data.frame(
  registration = c("1501090", "9901071"),
  name = c("Jane Doe", "Ambi Dexter"),
  id = c("jane_doe", "ambi_dexter")
), file = "Exam-2015-07-29.csv", sep = ";", quote = FALSE, row.names = FALSE)
dir()
## now the exam can be evaluated creating an output data frame (also stored
```

```
## as CSV file) and individual HTML reports (stored in a ZIP file),

## as there is only exactly on CSV/RDS/ZIP file in the current directory,
## these are found automatically - furthermore an evaluation scheme without
## partial points and differing points per exercise are used
ev1 <- nops_eval(eval = exams_eval(partial = FALSE, negative = FALSE))
dir()

## inspect evaluated data
ev1

## inspect corresponding HTML reports
if(interactive()) {
unzip("nops_eval.zip")
browseURL(file.path(mydir, "jane_doe",    "Exam-2015-07-29.html"))
browseURL(file.path(mydir, "ambi_dexter", "Exam-2015-07-29.html"))
}


## --- Options ---
if(interactive()) {
## below three typically needed options are discussed:
## (a) using a different evaluation strategy (here with partial credits),
## (b) using a different language (here de/German),
## (c) an error of the participant when filling in the registration number.

## as for (a): partial credits should only be used for multiple-choice questions
## where at least one alternative is correct and at least one is false
## [note that in this example this is not the case for the first question
## (single-choice) and the third question for Jane Doe (no alternative correct)]

## as for (c): for Ambi Dexter such an error was included in the PNG example
## image, the actual number is "9911071" but the crosses indicate "9901071"

## clean up previous evaluation
file.remove(c("nops_eval.csv", "nops_eval.zip"))

## write correct registration information
write.table(data.frame(
  registration = c("1501090", "9911071"),
  name = c("Jane Doe", "Ambi Dexter"),
  id = c("jane_doe", "ambi_dexter")
), file = "Exam-2015-07-29.csv", sep = ";", quote = FALSE, row.names = FALSE)

## call nops_eval() with modified options, where the error in the registration
## number of Ambi Dexter will trigger an interactive prompt
ev2 <- nops_eval(eval = exams_eval(partial = TRUE, rule = "false2"),
  language = "de")

## inspect evaluated data
ev2
cbind(ev1$points, ev2$points)
```

```
## inspect corresponding HTML reports
unzip("nops_eval.zip")
browseURL(file.path(mydir, "jane_doe",    "Exam-2015-07-29.html"))
browseURL(file.path(mydir, "ambi_dexter", "Exam-2015-07-29.html"))
}

## switch back to original working directory
setwd(odir)
```

---

nops_scan *Read Scanned NOPS Exams*

Description

Read scanned NOPS exams produced with exams2nops.

Usage

```
nops_scan(
  images = dir(pattern = "\\.PNG$|\\.png$|\\.PDF|\\.pdf$"),
  file = NULL, dir = ".",
  verbose = TRUE, rotate = FALSE, cores = NULL, n = NULL,
  density = 300,
  size = 0.029, threshold = c(0.04, 0.42), minrot = 0.002,
  string = FALSE)
```

Arguments

images	character. Names of the PDF/PNG images containing the scanned exams. By default all PDF/PNG images in the current working directory are used.
file	character or logical. Optional file name for the output ZIP archive containing the PNG images and the scan results. If file = FALSE no ZIP archive is created. By default a suitable name using the current time/date is used.
dir	character. Directory in which the ZIP file should be created. By default the current working directory.
verbose	logical. Should progress information be displayed?
rotate	logical. Should the input PDF/PNG images be rotated by 180 degrees first?
cores	numeric. If set to an integer mclapply is called internally using the desired number of cores to read the scanned exams in parallel.
n	numeric. The number of answer fields to read (in multiples of 5), i.e., 5, 10, ..., 45. By default taken from the type field.
density	numeric. Resolution used in the conversion of PDF images to PNG. This requires ImageMagick's convert to be available on the system.
size	numeric. Size of the boxes containing the check marks relative to the image height. This can be tweaked somewhat but should typically be between 0.23 and 0.31.

threshold	numeric. Vector of thresholds for the gray levels in the check mark boxes. If the average gray level is between the gray levels, the box is checked. If it is above the second threshold, some heuristic is employed for judging whether the box contains a cross or not.
minrot	numeric. Minimum angle for rotating images, i.e., images with a lower angle are considered to be ok.
string	logical. Are the files to be scanned manually marked string exercises (rather than single/multiple choice exercises)?

Details

nops_scan is a companion function for exams2nops. Exams generated with exams2nops can be printed and the filled out answer page can be scanned. Then, nops_scan can be employed to read the information in the scanned PDF/PNG images. The results are one text line per image containing the information in a very simple space-separated format.

If images only contains PNG files, then the R function readPNG is sufficient for reading the images into R. If images contains PDF files, these need to be converted to PNG first which requires PDFTk, GhostScript, and ImageMagick's convert to be available on the system. On Linux(-esque) systems this is typically easy to install by pdftk and imagemagick. The download links for Windows are: http://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/pdftk_free-2.02-win-setup.exe, http://www.imagemagick.org/script/binary-releases.php#windows, http://www.ghostscript.com/download/gsdnld.html.

Practical recommendations:

The scanned images produced by scanners or copying machines typically become smaller in size if the images are read in just black/white (or grayscale). This may sometimes even improve the reliability of reading the images afterwards.

The printed exams are often stapled in the top left corner which has to be unhinged somehow by the exam participants. Although this may damage the exam sheet, this is usually no problem for scanning it. However, the copying machine's sheet feeder may work better if the sheets are turned upside down (so that the damaged corner is not fed first into the machine). This often improves the scanning results considerably and can be accomodated by setting rotate = TRUE in nops_scan.

Value

A character vector with one element per scanned file (returned invisily if written to an output ZIP archive). The output contains the following space-separated information: file name, sheet ID (11 digits), scrambling (2 digits), type of sheet (3 digits, number of questions rounded up to steps of 5), 0/1 indicator whether the replacement sheet was used, registration number (7 digits), 45 multiple choice answers of length 5 (all 00000 if unused).

See Also

exams2nops, nops_eval

Examples

```
## scanned example images stored in exams package
img <- dir(system.file("nops", package = "exams"), pattern = "nops_scan",
```

```
  full.names = TRUE)

## read content
res <- nops_scan(img, file = FALSE)
writeLines(res)
```

---

num_to_schoice *Generate Single-Choice Question List from Numeric Solution*

Description

A function for generating a single-choice question list for one correct numeric solution along with four wrong solutions.

Usage

```
num_to_schoice(correct, wrong = NULL, range = c(0.5, 1.5) * correct,
  delta = 1, digits = 2, method = c("runif", "delta"), sign = FALSE,
  verbose = getOption("num_to_choice_warnings"))
```

Arguments

correct	numeric vector of length 1 with correct solution.
wrong	numeric vector (optional) with wrong solutions.
range	numeric vector of length 2 with range of random wrong solutions.
delta	numeric. Minimal distance between solutions.
digits	integer. Digits that should be displayed.
method	character specifying method for generating random results.
sign	logical. Should the sign be changed randomly?
verbose	logical. Should warnings be issued if no suitable set of wrong solutions can be found?

Details

The function num_to_schoice (or equivalently num2schoice) can be used for generating a single-choice question list for a numeric correct solution. The question list always comprises five elements, one of which is the correct solution. The wrong solutions can be provided or are generated randomly. If wrong is provided only up to 2 elements of it are used in order to assure some random solutions.

Two methods can be used to generate the wrong solutions: Either simply runif or otherwise a full equi-distant grid for the range with step size delta is set up from which a discrete uniform sample is drawn. The former is preferred if the range is large enough while the latter performs better if the range is small (as compared to delta.

Exercise templates using num_to_schoice should be thoroughly tested in order to avoid problems with too small ranges or almost identical correct and wrong answers! This can potentially cause problems, infinite loops, etc.

Value

num_to_schoice/num2schoice returns either NULL (if no suitable question list can be found) or a list with the following components:

solutions	a logical vector of length 5 indicating the correct solution,
questions	a character vector of length 5 with question list.

See Also

matrix_to_schoice

Examples

```
set.seed(1)
## just a correct solution
num_to_schoice(123.45)

## or equivalently
set.seed(1)
num2schoice(123.45)

## just a correct integer solution
num_to_schoice(123, digits = 0)

## a correct solution with a wider range
num_to_schoice(123.45, range = c(0, 200))

## here, the defaults can't work...
## num_to_schoice(0.1234)

## alternatives could be
num_to_schoice(0.1234, range = c(0, 1), delta = 0.03, method = "delta")
num_to_schoice(0.1234, range = c(-5, 5), delta = 0.05)
num_to_schoice(0.1234, wrong = c(0.2749, 1.9723), delta = 0.05)
num_to_schoice(0.1234, wrong = c(0.2749, 1.9723), range = c(-5, 5), delta = 0.05)
```

---

read_exercise *Reading LaTeX/Markdown Exercise Files*

Description

Reading an exercise in either LaTeX format (i.e., after Sweave was run) or Markdown format (i.e., after knit was run).

Usage

```
read_exercise(file)
read_metainfo(file)
```

Arguments

file	character. Name of the LaTeX (.tex) or Markdown (.md) file that should be read into R.

Details

read_exercise extracts the LaTeX/Markdown code from the question and solution environments/sections of the exercise file, extracting the corresponding answerlists separately (if any). Paths to supplementary files (such as graphics or data files) are stored and the metainformation is extracted (by calling read_metainfo which also includes sanity checks).

The supported metainformation commands are described in detail in vignette("exams2", package = "exams"), see Table 2. Essentially the extype command in is mapped to the type element of the returned list etc. (see the Value section below), using the right storage mode for each command (numeric, character, logical). Additionally, there is an exextra command which allows to set up arbitrary additional metainformation elements.

Value

read_exercise returns a list with elements

question	a character vector with LaTeX/Markdown code from the question environment (excluding the answerlist environment, if any).
questionlist	a character vector with LaTeX/Markdown code from the answerlist environment within the question environment (if any).
solution	a character vector with LaTeX/Markdown code from the solution environment (excluding the answerlist environment, if any).
solutionlist	a character vector with LaTeX/Markdown code from the answerlist environment within the solution environment (if any).
metainfo	a list of metainformation options, see below.
supplements	a character vector with paths to supplementary files such as graphics or data files (if any).

read_metainfo returns a list with the following elements. Most elements may also be NULL (or empty) if the underlying information is not specified in the file. If file specifies extra information, there may also be additional list elements.

file	character with file name (without extension) of the exercise template.
markup	character indicating whether "latex" or "markdown" markup is used in the exercise.
type	character indicating exercise type: num, mchoice, schoice, string, or cloze.
name	character with short name/description (to be used for printing within R).
title	character with a pretty longer title.
section	character with sections for groups of exercises (using slashes for subsections like a URL).
version	character with version of exercise.

solution	correct solution. The type/value depends on the type of the exercise: num leads to a numeric vector (of length 1 or 2), mchoice/schoice lead to logical vector, string to a character vector (of length 1), and cloze leads to a list of solutions (depending on clozetype).
tolerance	numeric tolerance limits (of length 1 or 2) for numeric solutions.
clozetype	character indicating the types of the elements of a cloze exercise.
points	numeric with (default) points for correct solution.
time	numeric with (default) time (in seconds) for solution.
shuffle	logical indicating whether mchoice/schoice answers should be shuffled (in Moodle or other e-learning systems).
single	logical indicating whether radio buttons should be used in Moodle.
length	numeric with length of solution.
string	character with a collapsed string of the solution (and tolerance) for backward compatibility with exams.
maxchars	character with the maximum number of characters allowed in QTI text answers (exercise type: string).
abstention	character with the label to be used for an abstention button in schoice/mchoice answers (currently only supported by Moodle).

See Also

xexams

Examples

```
## xexams() uses read_exercise() by default to read in
## each individual exercise, e.g., here for only a single
## exam with only a single exercise the result is:
set.seed(1090)
xexams("tstat.Rnw")[[1]][[1]]

## the corresponding Markdown version has:
set.seed(1090)
xexams("tstat.Rmd")[[1]][[1]]
```

---

stresstest_exercise *Stress Testing Exercises*

Description

In order to check the correct behavior of an exercise it is compiled several times. In each iteration the objects created by the exercise are collected and its values can be inspected.

Usage

```
## Stresstest function.
stresstest_exercise(file, n = 100, verbose = TRUE, seeds = NULL,
  stop_on_error = length(as.character(unlist(file))) < 2,
  ...)

## Plotting stresstest results.
## S3 method for class 'stress'
plot(x, type = c("overview", "solution", "ordering", "runtime"),
  threshold = NULL, variables = NULL,
  spar = TRUE, ask = TRUE, ...)
```

Arguments

file	character. A specification of an exercise file. If multiple files should be tested, argument file can also be a vector, matrix or list of files. The latter case sets argument plot = FALSE.
n	integer. The number of replications.
verbose	logical. Should the seeds used for compiling the exercise be prompted on the console.
seeds	The seeds that should be used when compiling the exercise. The default is seeds = 1:n.
stop_on_error	logical. Should the function stop on error or return the the seed, the file name and the error message. Useful when testing a number of exercises.
x	An object returned from function stresstest_exercise.
type	character. type == "overview" plots the basic overview, i.e, the runtimes, numeric solutions, position of correct solution(s), number of correct solutions, solution ordering frequencies, if available. If type == "solution", the numeric solutions are plotted against all input parameters stored in the objects element of x. type == "ordering" draws spineplots of ordering positions vs. input parameters stored in objects. type == "runtime" plots the compiling runtimes vs. objects.
threshold	numeric. Can be used to set a threshold, e.g., for numeric solutions a factor is created, factor(solution <= threshold), that is used on the y-axis of a spineplot.
variables	character. The variables that should be used from the objects for plotting.
spar	logical. Should graphical parameters be set or not.
ask	logical. For multiple plots, should the user be asked to hit the return key to see the next plot.
...	Arguments passed to xexams.

Details

In order to check the correct behavior of an exercise function stresstest_exercise runs xexams n times using different seeds. If an error occurs when compiling, the error can be reproduced by

setting the seed that is prompted at the console and create the exercise again, e.g., with exams2html. This way errors can be detected systematically.

All objects with length 1, which are created by the exercise, are collected in a data frame. These objects are assumed to be input parameters that control the output of the exercise. This can be used to detect certain input values that, e.g., lead to very long run times, or drive the number of correct answers in multiple choice exercises, etc.

For single and multiple choice type question the position(s) of the correct solution(s) is returned. For single choice questions that are created from a numeric version, e.g., using function num_to_schoice the answers are again converted to numeric and the ordering of the correct solution is reported. The ordering is sometimes heavily driven by some input parameters, e.g., the correct solution is always the largest or the smallest. For none "numeric" choice questions, the ordering is based on the lexicographical order of the answerlist.

Value

Function stresstest_exercise returns an object of class "stress" (a named list) with the following elements:

seeds	The seeds that where used.
runtime	Compiling times for each iteration.
objects	A data frame of length 1 objects that are created by the exercise.
solution	The numeric solution, availability is depending on the type of exercise.
position	A matrix indicating the position of correct solutions.
ordering	The ordering of the correct solution, only available for choice exercises.
ntrue	The number of correct answers in multiple choice type questions.

See Also

xexams, num_to_schoice

Examples

```
## Not run: ## Stress testing.
t1 <- stresstest_exercise("tstat.Rnw", n = 100)
t2 <- stresstest_exercise("tstat2.Rnw", n = 100)

## Plotting.
plot(t1, type = "overview")
plot(t1, type = "solution")
plot(t1, type = "solution", threshold = 30)
plot(t2, type = "ordering")
plot(t2, type = "runtime")

## For custom inspection, object can be
## transformed to a data.frame.
head(as.data.frame(t2))

## Multiple testing.
```

```
files <- list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  "scatterplot",
  "relfreq"
)
t3 <- stresstest_exercise(files, n = 100)
plot(t3)

## End(Not run)
```

---

tex2image *Transforming LaTeX Code Using ImageMagick or pdf2svg*

Description

Transformation of LaTeX code into an image by compiling to PDF and then transforming to PNG (by default) via ImageMagick's `convert` command or to SVG via `pdf2svg`.

Usage

```
tex2image(tex, format = "png", width = 6, pt = 12, density = 350,
  dir = NULL, tdir = NULL, idir = NULL,
  width.border = 0L, col.border = "white", resize = 650, shave = 2,
  packages = c("amsmath", "amssymb", "amsfonts"),
  header, header2 = NULL, Sweave = TRUE, show = TRUE,
  name = "tex2image")
```

Arguments

`tex`	character vector or list of character vectors. Each character vector is either the name of a LaTeX file or a vector containing LaTeX code directly.
`format`	character. Suffix for the type of graphic to convert to.
`width`	numeric. Width of the text in inch.
`pt`	numeric. Pointsize of the text.
`density`	numeric. Resolution density of the image.
`dir`	character specifying the output directory.
`tdir`	character specifying a temporary directory, by default this is chosen via `tempdir`.
`idir`	character specifying the path additional LaTeX inputs required.
`width.border`	numeric. Width of the framebox border.
`col.border`	character. Color of framebox border.
`resize`	numeric. Number of pixels for resizing the image.
`shave`	numeric. Number of pixels to shave the framebox from the edges.

packages	character. Names of LaTeX packages to be included.
header	character. LaTeX code to be included in the header of the LaTeX file before the beginning of the document. By default the parindent is set to 0 and sans serif fonts (phv) are used for both text and math.
header2	character. LaTeX code to be included in the header of the LaTeX file after the beginning of the document.
Sweave	logical. Should the LaTeX package Sweave.sty be included in the header?
show	logical. Show the resulting image(s) using browseURL.
name	character. Base name of the image file.

Details

tex2image converts LaTeX code to image files, e.g., for inclusion in web pages. It proceeds in the following steps: (1) LaTeX code is embedded into a suitable .tex file. (2) This is compiled to PDF using texi2dvi. (3) The PDF is converted to an image file. By default, conversion is to PNG using ImageMagick's convert function or alternatively to SVG via pdfcrop followed by pdf2svg.

To enable suitable clipping and cropping, each vector of LaTeX code has to fit onto a single page (A4). It is automatically embedded into a framebox that is removed in the conversion to image.

If tex is a list of LaTeX chunks, then these are compiled to separate pages of a single PDF in a single LaTeX run. Each page is subsequently converted to a separate image.

The respective image manipulation tools, i.e., either ImageMagick's convert or pdfcrop/pdf2svg, are assumed to be installed and available in the search path.

Value

Character vector with path(s) to image(s) generated from the LaTeX code.

See Also

texi2dvi

Examples

```
## Not run:
## some simple LaTeX
tex <- c("This is \\textbf{bold} and this \\textit{italic}.",
  "Points on the unit circle: $x^2 + y^2 = 1$.")

## default settings: PNG with sans serif fonts and width=6
tex2image(tex)

## SVG output
tex2image(tex, format = "svg")

## reduce width
tex2image(tex, width = 2)

## switch off header (-> LaTeX uses its standard serif fonts)
```

```
tex2image(tex, header = NULL)

## End(Not run)
```

---

xexams *Extensible Generation of Exams*

Description

Extensible automatic generation of exams including multiple choice questions and arithmetic problems.

Usage

```
xexams(file, n = 1L, nsamp = NULL,
  driver = list(sweave = NULL, read = NULL, transform = NULL, write = NULL),
  dir = ".", edir = NULL, tdir = NULL, sdir = NULL, verbose = FALSE,
  points = NULL)
exams_metainfo(x, ...)
```

Arguments

file	character. A specification of a (list of) exercise files, for details see below.
n	integer. The number of copies to be taken from file.
nsamp	integer. The number(s) of exercise files sampled from each list element of file. Sampling without replacement is used if possible. (Only if some element of nsamp is larger than the length of the corresponding element in file, sampling with replacement is used.)
driver	list with elements sweave (weaver function or list of arguments for the default xweave), read (function for reading exercise files, defaulting to read_exercise), transform (function to transform each exercise, by default no transformations are done), write (function to write exams to output files, by default nothing is written). For more details, see below.
dir	character. The output directory passed on to driver$write.
edir	character specifying the path of the directory in which the files in file are stored (see also below).
tdir	character specifying a temporary directory, by default this is chosen via tempdir. Note that this is cleaned up and potentially temporary files are deleted.
sdir	character specifying a directory for storing supplements, by default this is chosen via tempdir.
verbose	logical. Should information on progress of exam generation be reported?
points	integer. How many points should be assigned to each exercise? Note that this argument overules any exercise points that are provided within the expoints tags of the exercise files (if any). The vector of points supplied should either have length 1 or the number of exercises in the exam.
x	a list as returned by xexams.
...	currently not used.

Details

xexams is meant to provide an extensible framework for generating exams based on exercises in R/LaTeX format (via Sweave) or R/Markdown format (via knit) and rendering them into various output formats such as PDF, HTML, or XML (e.g., for Moodle or IMS QTI). xexams is typically not called by the user directly but is used as a common infrastructure for functions such as exams2pdf, exams2html, exams2moodle, exams2qti12, or exams2lops.

xexams generates exams from lists (or vectors) of Rnw/Rmd source files by: (1) running driver$sweave on each exercise (by default xweave is used, calling Sweave or knit), (2) running driver$read on the resulting LaTeX/Markdown file which by default uses read_exercise to read question/solution texts plus metainformation and stores the result in a list, (3) running driver$transform on this list for possible transformations (e.g., from LaTeX to HTML), (4) running driver$write on the list of exercises within each exam.

Each exercise in an exam is essentially a standalone source file that xexams knows (almost) nothing about, it just calls driver$sweave in each iteration and assumes that driver$read can read the resulting LaTeX or Markdown file into a list.

The specification in file should be either of form "foo.Rnw" (or equivalently just "foo") or "foo.Rmd", where the file should either be in the local directory, the edir directory or in the exercises directory of the package. file can either be a simple vector or a list of vectors. In the latter case, exercises are chosen randomly within each list element. For example, the specification file = list(c("a", "b"), "xyz") will result in an exam with two exercises: the first exercise is chosen randomly between "a" and "b" while "xyz" is always included as the second exercise.

Value

A list of exams (of length n), each of which is a list of exercises (whose length depends on the length of file and nsamp), each of which is a list (whose length/contents depends on driver$read).

When using the default reader, the resulting list can be simplified using exams_metainfo, returning the same (classed) structure as the older exams interface.

References

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. http://www.jstatsoft.org/v58/i01/.

See Also

xweave, exams, exams2pdf, exams2html, exams2moodle, exams2qti12, exams2lops

Examples

```
## define an exam with five exercises
myexam <- list(
  "boxplots",
  c("tstat", "ttest", "confint"),
  c("regression", "anova"),
  "scatterplot",
```

```
  "relfreq"
)

## run exams with default drivers (i.e., no transformations or writer)
x <- xexams(myexam, n = 2)
## x is a list of 2 exams,
## each of which contains 5 exercises,
## each of which contains LaTeX code for question(list) and solution(list),
## plus metainformation and potential supplements

## The first exercise in each exam is "boxplots", a multiple choice question.
## Its general question text is
x[[1]][[1]]$question
## with a list of multiple choice questions given as
x[[1]][[1]]$questionlist
## the corresponding graphic is in supplementary file
x[[1]][[1]]$supplements

## The metainformation is a list read for the \ex*{} items
x[[1]][[1]]$metainfo

## The metainformation can also be extracted/printed as
## in the old exams() (rather than xexams()) interface
exams_metainfo(x)
```

---

xweave *Wrapper Function for Weaving Either Rnw or Rmd Exercises*

Description

Simple wrapper function that either calls Sweave for Rnw exercises or knit for Rmd exercises.

Usage

```
xweave(file, quiet = TRUE, encoding = NULL, engine = NULL, envir = new.env(),
  pdf = TRUE, png = FALSE, svg = FALSE, height = 6, width = 6, resolution = 100, ...)
```

Arguments

file, quiet, encoding	arguments passed to Sweave or knit, respectively.
engine	character indicating whether "Sweave" (default) or "knitr" should be used for rendering Rnw exercises.
envir	argument passed to knit.
pdf, png, svg, height, width, resolution, ...	arguments passed to Sweave or opts_chunk, respectively. In the latter case: pdf/png/svg are mapped to dev; height/width are mapped to fig.height/fig.width; and resolution is mapped to dpi.

Details

Depending on whether `file` has an `.Rnw` or `.Rmd` suffix, either `Sweave` or `knit` is called for weaving the file by default. Rnw exercises can optionally also be weaved by `knit` by setting `engine = "knitr"`.

If `png = TRUE` or `svg = TRUE` when calling `Sweave`, then the resulting includegraphics statements are supplemented with the `.png` or `.svg` suffix of the corresponding graphics. For `svg` a simple graphics device hook `.xweave_svg_grdevice` is provided on-the-fly for plug-in into `Sweave`.

See Also

`Sweave`, `knit`

Index