exams (version 1.9-6)

exams2html: Generation of Exams in HTML Format

Description

Automatic generation of exams in HTML format.

Usage

```
exams2html(file, n = 1L, nsamp = NULL, dir = ".", template = "plain",
  name = NULL, quiet = TRUE, edir = NULL, tdir = NULL, sdir = NULL,
  question = "
```

Question

```
", solution = "
```

Solution

```
",
  mathjax = FALSE, resolution = 100, width = 4, height = 4,
  encoding = "", ...)

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

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

Question

```
", solution = "
```

Solution

```
  ",
     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 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

dir
character. The output directory, this has to be set if **`n`** is greater than 1. Current working directory is used by default.

template
character. A specification of a HTML template. The package currently provides **`"plain.html"`**.

name
character. A name prefix for resulting exercises.

quiet
logical. Should output be suppressed when calling `Sweave`?

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`.

sdir
character specifying a directory for storing supplements, by default this is chosen via `tempdir`.

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

solution
character or logical, see to 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 graphics passed to `Sweave`.

`encoding`
character, passed to `Sweave`.

`converter`
character. Workhorse function for transforming LaTeX code to HTML. For details see below.

`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

`...`
arguments passed on to **`make_exercise_transform_html`**.

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`.

Details

`exams2html` generates exams in a very simple HTML format using `xexams`. It proceeds by (1) calling `Sweave` on each exercise, (2) reading the resulting LaTeX code, (3) transforming the LaTeX code to HTML, and (4) embedding the HTML code in a very simple exercise template (which is currently hard-coded). 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 three functions: `ttm` produces HTML with MathML for mathematical formulas, `tth` produces plain HTML that aims to emulate mathematical 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 hard-coded template and writes a single HTML file for each exam.

When generating only a single exam (as in the examples below), **`exams2html`** tries to display this directly in a browser using `browseURL`. This may not work properly in RStudio versions older than 0.97.133. To avoid these problems, one can either upgrade to a more recent version of RStudio or set R's **`browser`** option manually to a suitable value, e.g., to `` `options(browser = NULL)` `` on Windows, `` `options(browser = "/usr/bin/firefox")` `` on Linux, or `` `options(browser = "/Applications/Firefox")` `` on OS X.

See Also

`` `xexams` ``, `` `ttm` ``, `` `tth` ``, `` `tex2image` ``, `` `browseURL` ``

Examples

Run this code

```
## 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)
exams2html("tstat", converter = "tex2image")
}
```

Run the code above in your browser using DataLab

Rdocumentation powered by datacamp

Learn R Programming

Search all packages and functions

exams (version 2.4-1)

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, rds = FALSE,
    resolution = 100, width = 4, height = 4, svg = FALSE, encoding = "UTF-8",
    iname = TRUE, stitle = NULL,
    testid = FALSE, zip = FALSE, num = NULL, mchoice = NULL,
    schoice = mchoice, string = NULL, cloze = NULL,
    points = NULL, rule = NULL, pluginfile = TRUE, forcedownload = FALSE,
    converter = "pandoc-mathjax", envir = NULL, engine = NULL,
    table = "table_shade", css = NULL, ...)

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

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.

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 (along with its sub-directories) in which the files in **`file`** are stored (see also **`xexams`**).
tdir	character specifying a temporary directory, by default this is chosen via `tempfile`. Note that this is cleaned up (i.e., existing files are deleted) and only certain temporary files are preserved.
sdir	character specifying a directory for storing supplements, by default this is chosen via `tempfile`.

verbose	logical. Should information on progress of exam generation be reported?
rds	logical indicating whether the return list should also be saved as an RDS data file.
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, ignored. The encoding is always assumed to be UTF-8.
envir	argument passed to `` `xweave` `` (which passes it to `` `knit` ``).
engine	argument passed to `` `xweave` `` indicating whether `` `"Sweave"` `` (default) or `` `"knitr"` `` should be used for rendering Rnw exercises.
table	character or logical. Control the style for tables in an exercise via a custom class: `` `"table_shade"` `` (equivalent to `` `table = TRUE` ``), `` `"table_rule"` ``, and `` `"table_grid"` `` being provided. See also the details below.
css	character. A character string (or vector) containing the path(s) to CSS style file(s). Alternatively, a string (or vector) with a `` `<style>` `` tag to be included in every exercise. This is used internally for the `` `table` `` styles above, see also the details below.
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 exercise 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., **`num`**) questions. If **`num`** is a function, this will be used to set up the question XML code. If **`num`** is a list, such a function is generated on the fly via **`make_question_moodle`** using the arguments in the list. For example, **`num = list(solution = FALSE)`** can be used to suppress embedding the solution text in the XML.
`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. For more guidance see argument **`num`** and the details below.
`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 exercise 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 Moodle is somewhat restrictive about the number of multiple-choice alternatives when using partial credits (see below for details).
`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.
`forcedownload`	logical. Should all supplementary links be forced to download when clicked (as opposed to opening in the browser)? Only supported if **`pluginfile = TRUE`**. If **`forcedownload = FALSE`** the behavior typically depends on the browser, user settings, and file type.
`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, allowed values are: `"abc"` (default), `"ABCD"`, `"123"` or `"none"`.
`usecase`	logical. Should string questions be case sensitive or not.
`cloze_mchoice_display, cloze_schoice_display`	character. In `cloze` questions, the user may set the visual appearance of choice questions. If `NULL` (default), `"MULTIRESPONSE"` (column of checkboxes) is used for `mchoice` questions and `"MULTICHOICE"` (drop-down menu) for `schoice` questions unless math markup is present in the question list. The latter is not rendered in drop-down menus and hence `"MULTICHOICE_V"` (radio buttons, vertical column) are used. Other options include a horizontal row of either checkboxes (`"MULTIRESPONSE_H"`) or radio buttons (`"MULTICHOICE_H"`), respectively. Shuffled versions of all display types are also available (since Moodle 3.0) by appending an "S", e.g., `"MULTICHOICE_S"` or `"MULTICHOICE_VS"` etc.
`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`.
numwidth, stringwidth	logical, numeric, or character. Should the width of all `num` or `string` sub-items, respectively, in a `cloze` be fixed to the same width? This can be accomplished by adding a wrong solution with a suitable length to all sub-items in Moodle XML. The default (`NULL` or equivalently `FALSE`) is not to do so but let Moodle decide the width of each cell based on the respective correct solution. Alternatively, the arguments can be set to `TRUE` (then the maximum width from the correct solutions is used), an integer (indicating the maximum width) or a character (like `"1111111"`, to be used as the wrong solution). Both arguments can also be set through exextra tags in each of the exercises' meta-information.
converter, ...	arguments passed on to `make_exercise_transform_html`. The default for `converter` is `"pandoc-mathjax"` which assumes that the quiz is imported in a Moodle site with MathJax plugin activated (which is the default setting in Moodle). For using MathML instead of MathJax the `converter` can be set to `NULL` or `"pandoc-mathml"` etc. For details see Zeileis (2019).

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 Markdown or LaTeX text, (3) transforming the text to HTML, and (4) embedding the HTML code in a Moodle XML file 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 Moodle XML file. 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 exercise 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 type (numeric, multiple-choice, etc.) is set up by separate functions that can be specified through the separate arguments **`num`**, **`mchoice`**, **`schoice`**, **`string`**, and **`cloze`** in **`exams2moodle`**. While it is possible to pass a suitable function to these arguments, it is more common to set suitable functions up on the fly using **`make_question_moodle`**. In this case, the arguments **`num`**, **`mchoice`**, **`schoice`**, **`string`** and **`cloze`** can be lists of arguments to pass on to **`make_question_moodle`**. For example, to suppress numbering the multiple-choice answer items with a/b/c/... one has to specify **`mchoice = list(answernumbering = "none")`** (which, by default, also gets passed on to **`schoice`**).

When using partial credits for multiple-choice exercises, only certain numbers of alternatives are supported in Moodle. This is because the Moodle XML format just supports certain percentages which can be added or subtracted from the score of an item. Therefore, it may not be possible to use partial credits for certain combinations of true and false answer alternatives when the overall number of alternatives is greater than 10.

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"`** and **`"fourfold2"`** 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 exercise file. 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 open-ended text questions in Moodle one can use **`string`** questions and then additionally set **`exstringtype`** to **`essay`** and/or **`file`**. See the **`"essayreg"`** question for a worked example. On top of the basic **`exstringtype`** one can make further Moodle-specific customizations via some **`exextra`** options, namely:

`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.

To control the style used for rendering the HTML in Moodle exercises, it is possible to include some custom CSS (cascading style sheets) code via the argument **`css`**. In particular, the **`exams2moodle`** function leverages this for table formatting. It includes its own CSS for this purpose if one of the classes **`"table_shade"`** (rows highlighted with different shades of gray), **`"table_rule"`** (with horizontal lines), or **`"table_grid"`** (with both horizontal and vertical lines) is used.

References

Dougiamas M, et al. (2022). *Moodle, Version 4.0*. https://moodle.org/.

MoodleDocs (2022). *Moodle XML Format*. https://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. tools:::Rd_expr_doi("10.18637/jss.v058.i01").

Zeileis A (2019). *Mathematical Notation in Online R/exams*. https://www.R-exams.org/tutorials/math/

See Also

`xexams`, **`ttm`**, **`tth`**, **`tex2image`**, **`make_exercise_transform_html`**,

Examples

Run this code

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

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

## output directory
dir.create(mydir <- tempfile())

## 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)
```

Run the code above in your browser using DataLab

Rdocumentation | powered by datacamp

Learn R Programming

Search all packages and functions

exams (version 2.4-0)

exams2pdf: Generation of Exams in PDF Format

Description

Automatic generation of exams in PDF format.

Usage

```
exams2pdf(file, n = 1L, nsamp = NULL, dir = ".", template = "plain",
    inputs = NULL, header = list(Date = Sys.Date()), name = NULL,
    control = NULL, encoding = "UTF-8", quiet = TRUE, transform = NULL,
    edir = NULL, tdir = NULL, sdir = NULL, texdir = NULL, texengine = "pdflatex",
    verbose = FALSE, rds = FALSE, points = NULL, seed = NULL,
    attachfile = FALSE, exshuffle = NULL, ...)

  make_exams_write_pdf(template = "plain", inputs = NULL,
    header = list(Date = Sys.Date()), name = NULL, encoding = "UTF-8",
    quiet = TRUE, control = NULL, texdir = NULL, texengine = "pdflatex")
```

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`**.

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, of the same length as `` `template` ``. By default (if `` `name` `` is `` `NULL` ``) the base name of `` `template` `` is used.
`control`	A list of control arguments for the appearance of multiple choice results (see details).
`encoding`	character, ignored. The encoding is always assumed to be UTF-8.
`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 (along with its sub-directories) in which the files in `` `file` `` are stored (see also `` `xexams` ``).
`tdir`	character specifying a temporary directory, by default this is chosen via `` `tempfile` ``. Note that this is cleaned up (i.e., existing files are deleted) and only certain temporary files are preserved.
`sdir`	character specifying a directory for storing supplements, by default this is chosen via `` `tempfile` ``.
`texdir`	character specifying a directory for running `` `texi2dvi` `` in. By default this is chosen via `` `tempfile` `` (and deleted again) but, if specified by the user, the temporary LaTeX files from the last iteration are preserved and not deleted. This is intended especially for debugging purposes.
`texengine`	character. Passed to `` `latexmk` `` if `` `tinytex` `` is available.
`verbose`	logical. Should information on progress of exam generation be reported?
`rds`	logical indicating whether the return list should also be saved as an RDS data file.
`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.
`seed`	integer matrix or logical. Either `` `NULL` `` (default), logical, or a matrix of random seeds for each possible exercise to be set prior to calling `` `driver@sweave` ``. If `` `NULL` `` no random seeds are set. If a matrix, the number of rows must be `` `n` `` and the number of columns must correspond to `` `unlist(file)` ``. If `` `TRUE` `` a suitable matrix of seeds is sampled.

`attachfile`	logical. Should the LaTeX commands `` `url` `` and `` `href` `` be replaced by `` `attachfile` `` commends when used for supplementary files? This enables embedding these supplementary files directly into the PDF when `` `template` `` loads the `` `attachfile` `` LaTeX package.
`exshuffle`	logical or integer. If the `` `exshuffle` `` argument is non-`` `NULL` `` it is used to overrule the `` `exshuffle` `` tag from the `` `file` `` (e.g., `` `exshuffle = FALSE` `` can be used to keep all available answers without permutation).
`...`	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` ``.

The argument `` `control` `` is specified by a named list, currently with elements `` `mchoice.symbol` `` and `` `cloze.collapse` ``. `` `mchoice.symbol` `` has to be a character vector with elements `` `True` `` and `` `False` ``, specifying the symbol used for the questionnaire output in the final PDF file. `` `cloze.collapse` `` specifies the character used for collapsing mchoice/schoice alternatives within a cloze exercise. By default, these are separated by `` `" / "` `` but with `` `cloze.collapse = "\\\\"` `` each alternative would be in a new line. Finally, `` `cloze.collapse = "enumerate"` `` can also be used which employs a nested enumerate environment. In the latter case, the questionnaire uses `` `exclozechoice` `` rather than `` `exmchoice` `` (see `` `exam.tex` `` or `` `solution.tex` `` for an illustration.

References

Gruen B, Zeileis A (2009). Automatic Generation of Exams in R. *Journal of Statistical Software*, **29**(10), 1--14. tools:::Rd_expr_doi("10.18637/jss.v029.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. tools:::Rd_expr_doi("10.18637/jss.v058.i01").

See Also

`xexams`, **`exams`**, `texi2dvi`

Examples

Run this code

```
## load package and enforce par(ask = FALSE)
##
## additionally, for simplicity, enforce using the basic
## tools::texi2dvi() LaTeX interface instead of the more
## flexible/robust tinytex::latexmk()
library("exams")
oopt <- options(device.ask.default = FALSE, exams_tex = "tools")

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

options(exams_tex = oopt$exams_tex)
```

Run the code above in your browser using DataLab

Rdocumentation powered by datacamp

Learn R Programming

Search all packages and functions

exams (version 2.4-1)

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 = "UTF-8", envir = NULL, engine = NULL,
    edir = NULL, tdir = NULL, sdir = NULL, verbose = FALSE,
    points = NULL, exshuffle = NULL, ...)
```

Value

`exams2pandoc` returns a list of exams as generated by **`xexams`**.

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, ignored. The encoding is always assumed to be UTF-8.
`envir`	argument passed to **`xweave`** (which passes it to `knit`).
`engine`	argument passed to **`xweave`** indicating whether **`"Sweave"`** (default) or **`"knitr"`** should be used for rendering Rnw exercises.
`edir`	character specifying the path of the directory (along with its sub-directories) in which the files in **`file`** are stored (see also **`xexams`**).
`tdir`	character specifying a temporary directory, by default this is chosen via `tempfile`. Note that this is cleaned up (i.e., existing files are deleted) and only certain temporary files are preserved.
`sdir`	character specifying a directory for storing supplements, by default this is chosen via `tempfile`.
`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.
`exshuffle`	logical or integer. If the **`exshuffle`** argument is non-**`NULL`** it is used to overrule the **`exshuffle`** tag from the **`file`** (e.g., **`exshuffle = FALSE`** can be used to keep all available answers without permutation).
`...`	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.)

See Also

`xexams`, `pandoc_convert`

Examples

Run this code

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

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

## output directory
dir.create(mydir <- tempfile())

## 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")
```

Run the code above in your browser using DataLab

Rdocumentation | powered by datacamp

Learn R Programming

Search all packages and functions

exams (version 2.4-1)

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, nsamp = NULL, dir = ".", 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 = "UTF-8", startid = 1L,
    points = NULL, showpoints = FALSE, samepage = FALSE, newpage = FALSE,
    twocolumn = FALSE, reglength = 7L, seed = NULL, ...)

  make_nops_template(n, replacement = FALSE, intro = NULL, blank = NULL,
    duplex = TRUE, pages = NULL, file = NULL, nchoice = 5, encoding = "UTF-8",
    samepage = FALSE, newpage = FALSE, twocolumn = FALSE, reglength = 7L)
```

Value

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

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.
`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 either display on the screen (if **`` `n = 1L` ``**) 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. See below for the list of supported languages.
`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 (in a file format supported by pdfLaTeX). If set to **`` `NULL` ``**, the logo is omitted.
`date`	character or **`` `"Date"` ``** object specifying the date of the exam.
`replacement`	logical. Should a replacement exam sheet be included?
`intro`	character. Either a single string with the path to a .tex file or a vector with 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	character vector or list. Either a character vector with LaTeX code to include in the header or a named list with further options to be passed to the LaTeX files.
encoding	character, ignored. The encoding is always assumed to be UTF-8.
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?
newpage	logical. Should each exercise start on a new page? (Technically, a page break is added after each exercise.)
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. In case of **`` `reglength < 7` ``**, internally **`` `reglength = 7` ``** is enforced (and thus necessary in the registration CSV file) but the initial ID digits are fixed to 0 in the exam sheet and corresponding boxes ticked already.
`seed`	integer matrix or logical. Either **`` `NULL` ``** (default), logical, or a matrix of random seeds for each possible exercise to be set prior to calling **`` `driver@sweave` ``**. If **`` `NULL` ``** no random seeds are set. If a matrix, the number of rows must be **`` `n` ``** and the number of columns must correspond to **`` `unlist(file)` ``**. If **`` `TRUE` ``** a suitable matrix of seeds is sampled.
`...`	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.

Tutorial for NOPS workflow: https://www.R-exams.org/tutorials/exams2nops/.

Limitations: (a) Only up to five answer alternatives per question are supported. (b) Currently, only up to 45 questions are supported. If you have more questions, consider splitting the entire exam up into two NOPS exams. (c) Only up to 3 open-ended questions can be included. (d) Each question must have the same number of answer alternatives and the same number of points across random replications. For example, the **`` `n` ``** replications drawn for the first exercise all need, say, five alternatives and two points. Then, the second exercise may have, say, four alternatives and five points and so on. But this may not be mixed within the same exercise number.

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` ``**.

The **`` `language` ``** elements can be specified through a relatively simple text file and the package already ships with: English (**`` `"en"` ``**), Croatian (**`` `"hr"` ``**), Danish (**`` `"da"` ``**), Dutch (**`` `"nl"` ``**), Finnish (**`` `"fi"` ``**), French (**`` `"fr"` ``**), Galician (**`` `"gl"` ``**), German (**`` `"de"` ``**), Hungarian (**`` `"hu"` ``**), Italian (**`` `"it"` ``**), Japanese (**`` `"ja"` ``**), Korean (**`` `"ko"` ``**), Norwegian (Bokmål, **`` `"no"` ``**), Portuguese (**`` `"pt-PT"` ``** or **`` `"pt-BR"` ``**; also **`` `"pt"` ``** is synonymous with **`` `"pt-PT"` ``**), Romanian (**`` `"ro"` ``**), Russian (**`` `"ru"` ``**), Serbian (**`` `"sr"` ``**), Slovak (**`` `"sk"` ``**), Slovenian (**`` `"sl"` ``**), Spanish (**`` `"es"` ``**), Swiss German (**`` `"gsw"` ``**), Turkish (**`` `"tr"` ``**), Vietnamese (**`` `"vi"` ``**). Note that the language names correspond to the ISO 639 codes (https://www.loc.gov/standards/iso639-2/php/code_list.php) or IETF language tags (https://en.wikipedia.org/wiki/IETF_language_tag) if no ISO 639 codes exists (as for Brazilian Portuguese). For more details about the underlying text file in DCF format, see https://www.R-exams.org/tutorials/nops_language/

Examples

Run this code

```
## load package and enforce par(ask = FALSE)
## additionally, for simplicity, enforce using the basic
## tools::texi2dvi() LaTeX interface instead of the more
## flexible/robust tinytex::latexmk()
library("exams")
oopt <- options(device.ask.default = FALSE, exams_tex = "tools")

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

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)
dir.create(mydir <- tempfile())

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

# \donttest{
## 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, dir = mydir,
  institution = "Universit\\\"at Innsbruck",
  name = "uibk", logo = "uibk-logo-bw.png",
  title = "Klausur", language = "de",
  replacement = TRUE, duplex = FALSE)
dir(mydir)
# }

options(exams_tex = oopt$exams_tex)
```

Run the code above in your browser using DataLab