Package: learnitgrid (via r-universe)

August 12, 2024

Type Package

Version 0.9.0

Title Manage Rubrics or Assessment Grids for GitHub Repositories

Description Create and manage semi-automatically rubrics to assess GitHub projects (R scripts, R Markdown or Quarto files). Create directed projects where students have to complete documents and submit them to GitHub (classroom) so that they are evaluated using the rubric (or assessment grid).

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 4.2.0)

Imports askpass, cli, cyphr, collapse, cowplot, data.io, diffr, digest, DT, flashClust, fs, ggplot2, gh, glue, here, highr, htmlwidgets, lubridate, openssl, parsermd, purrr, qs, rstudioapi, shiny, shinycssloaders, shinydashboard, stats, stringdist, svMisc, testthat (>= 3.0.0), utils, writexl

Suggests knitr, rmarkdown, spelling

Remotes SciViews/data.io, SciViews/svMisc

License MIT + file LICENSE

URL https://github.com/learnitr/learnitgrid, https://learnitr.github.io/learnitgrid/

BugReports https://github.com/learnitr/learnitgrid/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

VignetteBuilder knitr

Encoding UTF-8

Language en-US

ByteCompile yes

Config/testthat/edition 3

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

RemoteUrl https://github.com/learnitr/learnitgrid

RemoteRef HEAD

RemoteSha 46fa574de14c45242f65d16164b8c247722af9f1

Contents

check_grids *Summarize and check evaluation grids for the learnitgrid Shiny app*

Description

Check all evaluation grids in a correction set and computes the required statistics for the summary page of the learnitgrid shiny app.

Usage

```
check_grids(dir, save.log = TRUE, save.rds = TRUE)
```

Arguments

`dir`	The path to the evaluation grids
`save.log`	Should a log file be saved (yes by default)?
`save.rds`	Should an RDS file with the data be saved (yes by default)?

Value

A data frame with the results of the evaluation grids check for the learnitgrid Shiny app.

`chunk_labels` *Make proper Rmd/Qmd chunk labels from strings for parsermd::parse_rmd()*

Description

Make proper Rmd/Qmd chunk labels from strings for parsermd::parse_rmd()

Usage

```
chunk_labels(x)
```

Arguments

`x`	A character string of chunk labels to convert

Value

A character string of the same length as `x` with "educated" labels.

Examples

```
chunk_labels(c("Summer is hot", "", NA, " ", "Winter is cold  "))
```

`copy_cache` *Copy data cache from a reference folder into an RStudio project*

Description

When GitHub repositories for a Classroom assessment are cloned, the data cache is not present. For large datasets, it may take a long time to get the data from an URL for each repository. By using a single version of the data cache, on can populated the corresponding folder in each repository. The `CC()` function can be used to remove the data from the cache.

Usage

```
copy_cache()

CC(only_if_copied = FALSE, cache_data = cache_data)
```

Arguments

only_if_copied	If TRUE, the cache is only deleted if it was copied
cache_data	The list of copied files, returned by copy_cache()

Value

The list of copied files is returned invisibly.

correct_rmd *Make sure that Rmd/Qmd chunk labels are written without spaces*

Description

Make sure that Rmd/Qmd chunk labels are written without spaces

Usage

```
correct_rmd(rmd)
```

Arguments

rmd	Character string with the content of a Rmd/Qmd file.

Value

The same Rmd/Qmd content, but with "educated" chunk labels.

create_context *Create a context object for a correction set*

Description

Create a list that contains context for a given correction set (must be recalculated when a different correction set/project is selected).

Usage

```
create_context(
  correction,
  base_corr_dir,
  base_templ_dir,
  base_repos_dir,
  repositories,
  assignments,
  github_url,
  branch
)
```

Arguments

`correction`	The correction set
`base_corr_dir`	The base directory for correction sets
`base_templ_dir`	The base directory for correction templates
`base_repos_dir`	The vase directory for GitHub repositories to correct
`repositories`	The repositories
`assignments`	The assignments
`github_url`	The GitHub URL (base part)
`branch`	The GitHub branch concerned by this correction

Value

A list with the context information for the learnitgrid Shiny app.

`df_structure` *Transformation functions for creating reference or result objects for testing progress of the students in their project.*

Description

Transformation functions for creating reference or result objects for testing progress of the students in their project.

Usage

```
df_structure(object, ...)

digest(object, algo = "md5", ...)

object_attr(object, attrib = "class", ...)

object_part(object, part = "x", ...)

object_str(object, part = "x", ...)
```

Arguments

`object`	The object to transform
`...`	Further arguments (not used for now)
`algo`	algorithm to use for digest, `"md5"` by default
`attrib`	The attribute(s) to record from the object
`part`	The part(s) to record from the object (list items)

Value

`df_structure()` returns names, attributes "label" and "units", number of row and columns, classes, if there are missing data and comment in a data frame. `digest()` returns a hash of the object. `object_attr()` returns the attribute(s) of the object. `object_part()` returns the part(s) of an object (list items). `object_str()` returns a str() representation of the object parts.

`dir_path_create` *Make sure a directory exists, or create it*

Description

Make sure a directory exists, or create it

Usage

```
dir_path_create(...)

dir_path_check(...)

file_path_check(...)
```

Arguments

`...`	The successive folders that make the path.

Value

A **path** object is returned invisibly.

Functions

- `dir_path_check()`: Check that a directory exists.
- `file_path_check()`: Check that a file exists.

Examples

```
test_path <- dir_path_create(tempdir(), "dir_path_create_test", "subfolder")
test_path
dir.exists(test_path) # Should be TRUE
dir_path_check(test_path) # Return the path only if it exists
# Remove
unlink(test_path)
unlink(dirname(test_path))
```

`encrypt_solutions` *Encrypt solution files into* `..._solution.xxx.aes` *files*

Description

These functions manage encryption and decryption of solution files. The password must be defined in an option `learnitgrid.key` or an environment variable `LEARNITGRID_KEY` under a digest form.

Usage

```
encrypt_solutions(key = NULL, error = TRUE)

decrypt_solutions(key = NULL, error = TRUE)

set_key()
```

Arguments

`key`	The key to use for encryption/decryption. If not provided, it is asked
`error`	If `TRUE`, an error is generate in case the file encryption or decryption fails.

Value

`NULL` invisibly.

`get_git_stats` *Summarize the git stats for one or more git repositories*

Description

Use data from a git_stats.csv file to get a history of commits done in a repository for the learnitgrid Shiny app.

Usage

```
get_git_stats(
  dir,
  exclude_authors = "github-classroom[bot]",
  type = "all",
  tz = "UTC"
)
```

Arguments

`dir` The path the the git_stats.csv file

`exclude_authors`
The list of authors to exclude from the stats

`type` The type of files to consider ("all", "R", "Rmd", or "Qmd")

`tz` The time zone to use for times

Value

A data frame with git stats data to be used by the learnitgrid Shiny app.

`hook_last_chunk` *Configure Knitr to generate a .Last.chunk object*

Description

Change the Knitr renderer so that the last computed object is automatically saved as `.Last.chunk`, and to record results from a chunk with chunk options `record=`.

Usage

```
hook_last_chunk()
```

Value

The `opts_chunk` or the `knit_hook` set accordingly.

install_grid_example *Install an extended example dataset to try the learnitgrid Shiny application*

Description

Install an extended example dataset to try the learnitgrid Shiny application

Usage

```
install_grid_example(dir = tempdir(), browse = FALSE)
```

Arguments

dir	The directory where to decompress the extended example
browse	Should we browse the example directory after decompression?

Value

The path to the example learnitgrid data is returned invisibly.

See Also

run_grid()

Examples

```
# Install the extended examples in a temporary directory
## Not run:
library(learnitgrid)
ex_dir <- install_grid_example(browse = TRUE)

## End(Not run)
```

is_identical_to_ref *Functions to be used in testthat test of the project*

Description

Functions to be used in testthat test of the project

Usage

```
is_identical_to_ref(name, part = NULL, attr = NULL)

is_equal_to_ref(name, part = NULL, attr = NULL)

is_rendered(quarto, format = "html")

is_rendered_current(quarto, format = "html")

is_data(name, dir = "data", format = "rds", check_df = FALSE)

is_data_df(name, dir = "data", format = "rds", check_df = TRUE)

has_labels_all(name, part = NULL)

has_labels_any(name, part = NULL)

has_units_all(name, part = NULL)

has_units_any(name, part = NULL)

is_display_equation(text, object)

is_display_param_equation(text, object)

is_inline_equation(text, object)

is_inline_param_equation(text, object)
```

Arguments

`name`	The name of the result and reference files
`part`	The part(s) (list item(s)) to co=pare
`attr`	The attribute(s) to compare
`quarto`	The name of the Quarto or R Markdown file
`format`	The format of the rendered file. For `is_rendered()` or `is_rendered_current()` it is `"html'` by default. For the other function, the default is `"rds"`
`dir`	The subdirectory in the package where to look for (`"data"` by default)
`check_df`	Check if the data file contains a data frame
`text`	The text to check
`object`	The object to look for in the `eq__()` function

Value

`TRUE` if the result is identical to the reference, `FALSE` otherwise

`link_to_www` *Create symbolic links under www subdirectory or strip www subdirectory from the path*

Description

Symbolic links in the www subdirectory of the Shiny app are required for the application to properly display html documents or images in html tags.

Usage

```
link_to_www(path, link)

www_relative(path)
```

Arguments

`path`	The (absolute) path containing the documents.
`link`	The name to use for the symbolic link under the www subdirectory of the Shiny app.

Value

`TRUE` if the symbolic link exists for `link_to_www()` or `FALSE` otherwise. The modified path is returned by `www_relative()`

Functions

- `www_relative()`: Strip www/ in front of the relative paths for a Shiny app.

`make_ref` *Create or read reference files for tests*

Description

Reference files allow to check if results are correct. They are not just a copy of the result files. They are reencoded to avoid someone could just copy and paste from the reference to the results directories and cheat.

Usage

```
make_ref(
  name,
  ...,
  dir1 = here::here("tests", "results"),
  dir2 = here::here("tests", "reference"),
  nthreads = parallel::detectCores(logical = FALSE)
)

read_ref(
  name,
  ...,
  dir = here::here("tests", "reference"),
  nthreads = parallel::detectCores(logical = FALSE)
)
```

Arguments

`name`	Name of the result file to transform into a reference file
`...`	Further parameters passed to `qs::qread()`
`dir1`	Directory containing the result file
`dir2`	Directory where to place the reference file
`nthreads`	Number of threads to use for reading or writing the files
`dir`	Directory containing the reference file

Value

The decoded content of the result file for `read_ref()` or the number of bytes written for `make_ref()`.

`make_test` *Run action of a MAKEFILE from R*

Description

The five possible actions are "test", to launch the test on the repository, "clean" to delete several generated files, "original" to configure the repository with original files, "solution" to configure the repository with solution files, and "prepare" to prepare the repository for the final version ready for the assignment in, say Github classroom.

Usage

```
make_test()

make_clean()
```

`make_original()`

`make_solution()`

`make_prepare()`

Value

The result from `make` is returned

`populate_table` *Computes the content of a DT::datatable for a correction set*

Description

Populate a DT table according to selected items with a list of grids (or "all") or according to a selected grid with a list of items (or "all")

Usage

```
populate_table(
  items,
  grids = "all",
  context,
  reorder = (length(items) == 1),
  highlight = FALSE,
  max_lines = 30L,
  on_github = TRUE
)
```

Arguments

`items` The items of the evaluation grid to display in the table, usually either one item, or "all" for everything

`grids` The evaluation grids to display, usually either "all" if only one item, or one grid if "all" items

`context` A context object as computed by `create_context()`.

`reorder` Should the rows in the table be reordered by similarities? This is usually `TRUE` for a single items, or `FALSE` otherwise (and it is computed as such by default).

`highlight` Syntax highlighting for code (slow, thus `FALSE` by default)

`max_lines` The maximum number of content lines that are displayed (truncate very long contents).

`on_github` Should the links point to the GitHub repository or to the local files (default is `TRUE`)?

Value

A data frame with the content to be displayed in a DT::datatable object.

`prepare_files` *Prepare files for original or solution version*

Description

Make sure all files are original or solution versions, and possibly also remove last_saved versions. This is typically used to prepare the repository for an assignment, or to switch from originals to solution to verify the tests.

Usage

```
prepare_files(type = "original", remove_last_saved = FALSE, error = TRUE)
```

Arguments

`type` Either `"original"` or `"solution"`

`remove_last_saved`
Should we also remove the last_saved version (`FALSE` by default)?

`error` If `TRUE`, an error is thrown in case there is no file to prepare

Value

`NULL`

`project_reporter` *Create a testthat reporter suitable to test projects*

Description

Create a testthat reporter suitable to test projects

Usage

```
project_reporter()
```

Value

A testthat reporter

`record_res` *Record, read and write results that capture given characteristics of an object*

Description

Record, read and write results that capture given characteristics of an object

Usage

```
record_res(
  object = ".Last.chunk",
  name = object,
  fun = NULL,
  ...,
  dir = here::here("tests", "results"),
  env = parent.frame()
)

read_res(
  name,
  ...,
  dir = here::here("tests", "results"),
  nthreads = parallel::detectCores(logical = FALSE)
)

write_res(
  object,
  name,
  ...,
  dir = here::here("tests", "results"),
  nthreads = parallel::detectCores(logical = FALSE)
)

RO(
  object = ".Last.chunk",
  name = object,
  fun = NULL,
  ...,
  dir = here::here("tests", "results"),
  env = parent.frame()
)

RN(name, object = ".Last.chunk", fun = NULL, ..., env = parent.frame())

RODFS(
  object = ".Last.chunk",
  name = object,
  fun = df_structure,
  ...,
  env = parent.frame()
)

RNDFS(
  name,
  object = ".Last.chunk",
  fun = df_structure,
  ...,
  env = parent.frame()
)

ROMD5(
  object = ".Last.chunk",
  name = object,
  fun = digest,
  ...,
  env = parent.frame()
)

RNMD5(name, object = ".Last.chunk", fun = digest, ..., env = parent.frame())

ROP(
  object = ".Last.chunk",
  part = "x",
  name = object,
  fun = object_part,
  ...,
  env = parent.frame()
)

RNP(
  name,
  part = "x",
  object = ".Last.chunk",
  fun = object_part,
  ...,
  env = parent.frame()
)

ROA(
  object = ".Last.chunk",
  attrib = "class",
  name = object,
  fun = object_attr,
  ...,
  env = parent.frame()
)

RNA(
  name,
  attrib = "class",
  object = ".Last.chunk",
  fun = object_attr,
  ...,
  env = parent.frame()
)

ROSTR(
  object = ".Last.chunk",
  part = "x",
  name = object,
  fun = object_str,
  ...,
  env = parent.frame()
)

RNSTR(
  name,
  part = "x",
  object = ".Last.chunk",
  fun = object_str,
  ...,
  env = parent.frame()
)
```

Arguments

`object` The object to record results from (`".Last.chunk"` by default, which corresponds to the last R object produced in the current chunk in an R Markdown or Quarto document)

`name` The name of the result file

`fun` The function used to compute results for the object

`...` Additional arguments to pass to `fun=` or to the read/write functions using `qs::qsave()` and `qs::qread()`

`dir` The directory where to save the results

`env` The environment where to look for the object

`nthreads` The number of threads to use for reading and writing

`part` The part(s) (list item(s)) to use

`attrib` The attribute(s) to use

Details

The main function to record the results is `record_res()`. However, there are several shortcuts to record specific characteristics of an object: `RO` is the same as `record_res()`, `RN()` records by default `".Last.chunk"` but focuses on the name of the result file, `RODFS()` records the main structure of a data frame, `RNDFS()` is the same as `RODFS()` but focuses on the name of the result file, `ROMD5()` records the MD5 hash of an object, `RNMD5()` is the same focusing on the name, `ROP()` records one or several parts of an object (items from a list), `RNP()` is the same focusing on the name, `ROA()` records one or several attributes of the objects with `RNA()` focusing on the name, `ROSTR()` records the `utils::str()` summary of the object, and `RNSTR()` is the same focusing on the name of the result file.

Value

The deserialized results for `read_res()` or the number of bytes written invisibly for `write_res()`. `record_res()` invisibly returns `TRUE` or `FALSE` depending on the success of the operation.

run_grid *Run the learnitgrid Shiny application*

Description

Run the learnitgrid Shiny application

Usage

```
run_grid(data_dir = getOption("learnitgrid.data.dir", NULL))
```

Arguments

`data_dir`	The directory containing the 'learnitgrid' data. If missing, the "learnitgrid.data.dir" option is used. If `""`, the small example dataset in this package is used.

Value

Nothing is returned, the function is used for its side-effect of running the learnitgrid Shiny application.

See Also

`install_grid_example()`

Examples

```
## Not run:
library(learnitgrid)
# Run a basic example in the learnitgrid package
run_grid("")
# Please, note that the .Rmd reports that are analyzed in this example
# are NOT included to save space, hence, error messages in corresponding
# columns of the evaluation grids.

# Install an extended example in tempdir() (inspect its content)
install_grid_example(browse = TRUE)
# ... and run the app with it
run_grid()
# The .Rmd reports are included in this example.

## End(Not run)
```

---

select_answer *A simple multiple choice system in a R chunk, compatible with git and R Markdown or Quarto documents*

Description

A simple multiple choice system in a R chunk, compatible with git and R Markdown or Quarto documents. `obfuscate()` and `get_word()` are used to hide correct answers in the testthat tests.

Usage

```
select_answer(x, name = NULL)

obfuscate(x)

get_word(x)
```

Arguments

`x`	A String with the different answers, starting with []. The user has to "check" the correct items by adding ’x’ or ’X’ inside the brackets
`name`	The name of the select_answer. If not provided, the label of the chunk where the `select_answer()` function is placed.

Value

A Markdown paragraph containing only the selected items (and the selected items are also written in a result file)

switch_to_original *Switch to a given version of the document or save such a version*

Description

When there are several versions of a document (R script, R Markdown or Quarto document), switch the main file to one of the saved versions: original version, solution with the correction, or last saved version. The `save_as_original()` and `save_as_solution()` functions are used to create respectively the original and solution versions of the document. These files have same name as the initial file but ending with `_original`, `_solution` or `_last_saved`., and starting with a dot `.`. They are thus "hidden" files. For R scripts, the extension is also changed into `.Rscript` for the original and solution versions.

Usage

```
switch_to_original(file = NULL, error = TRUE)

switch_to_solution(file = NULL, error = TRUE)

switch_to_last_saved(file = NULL, error = TRUE)

save_as_original(file = NULL)

save_as_solution(file = NULL)
```

Arguments

`file`	The file to switch to a version. If not provided, the currently edited file in RStudio is used by default
`error`	In case the version does not exists, do we issue an error or not (no by default)

Value

`NULL` is returned invisibly. The functions are used for their side-effect of switching document versions.

test_dir *Test a project directory, possibly limit the number of uses*

Description

Test a project directory, possibly limit the number of uses

Usage

```
test_dir(path, reporter = project_reporter(), times = 10L, ...)
```

Arguments

`path` The path to test

`reporter` The testthat reporter to use

`times` The maximum number of times the tests can be run by the end-user

`...` Additional arguments to pass to `testthat::test_dir()`

Value

The result of `testthat::test_dir()`

Index