Package: guri (via r-universe)

October 24, 2024

Title ~!gurí_: Unified Format Manager for Research Journals

Version 0.1.0.9002

Description ~gurí_ (Gestor Unificado de formatos para Revistas de Investigación / Unified Format Manager for Research Journals) facilitates the generation of final documents for scientific journals from documents obtained in the 'proofreading' stage. The proposal seeks to solve the difficulties of some academic journals in generating final documents in different formats in a consistent way and without generating duplicated processes. It also takes into account that many scientific journals use docx documents as the basis of their workflows.

License CC BY-NC-SA 4.0 + file LICENSE

URL https://github.com/estedeahora/guri, https://estedeahora.github.io/guri/

BugReports https://github.com/estedeahora/guri/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports cli, dplyr, fs, pandoc, purrr, readxl, rmarkdown, rstudioapi, stringr, tinytex, utils, xml2

Depends R (>= 2.10)

LazyData true

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

RemoteUrl https://github.com/estedeahora/guri

RemoteRef HEAD

RemoteSha b4c3e24b27b646aa041f47c0f784a2d17c719a49

Contents

`CREDIT_to_CSV` *Convert the 'xlsx' file with the credit information to csv format.*

Description

Convert the 'xlsx' file with the credit information to csv format.

Usage

```
CREDIT_to_CSV(art_path, art_id, verbose)
```

Arguments

`art_path`	A string with the path to the article folder.
`art_id`	A string with the article id.
`verbose`	Logical. Specifies whether to display verbose output (default `TRUE`).

Value

Invisible TRUE.

guri_appendix *Make markdown appendix from docx documents*

Description

Make markdown appendix from docx documents

Usage

```
guri_appendix(art_path, art_id)
```

Arguments

art_path	A string with the path to the article folder, where the appendix files are located.
art_id	A string with the article id, used to identify the appendix files. The appendix files should be named as "art_app1.docx", "art_app2.docx", etc.

Value

A character vector containing the names of the converted Markdown files.

guri_article *Generate the output files for individual article.*

Description

Generate the output files for individual article.

Usage

```
guri_article(art_path, art_dir, art_id, verbose = TRUE, clean_files = TRUE)
```

Arguments

art_path	A string with the path to the article folder.
art_dir	A string with the article folder name.
art_id	A string with the article id.
verbose	Logical. Specifies whether to display verbose output (default TRUE).
clean_files	Logical. Should the temporary files be deleted and reordered in folders after the creation of the final files?. Primarily for debugging purposes (default is TRUE).

Value

Invisible TRUE.

guri_clean_files *Clean and reorganize article folder, moving temporary (auxiliary), log and output files.*

Description

Clean and reorganize article folder, moving temporary (auxiliary), log and output files.

Usage

```
guri_clean_files(art_path, art_id, verbose)
```

Arguments

art_path	A string with the path to the article folder.
art_id	A string with the article id.
verbose	Logical. Specifies whether to display verbose output (default TRUE).

Value

Invisible TRUE.

See Also

guri_outputs

guri_config_journal *Create the journal configuration files*

Description

Create the journal configuration files in _config. If csl_name is given, it also places the corresponding csl in the configuration folder.

Usage

```
guri_config_journal(journal_folder = NULL, csl_name = NULL, force = FALSE)
```

Arguments

journal_folder	A string with the path to the journal. If NULL (default), the working folder is used. For journal repositories, this must be provided.
csl_name	A string with the CSL's name (without its extension). See: https://github.com/citation-style-language/styles
force	Logical. Should it be overwritten if a configuration folder already exists (Default: FALSE)?

Value

Invisible TRUE.

`guri_convert` *Converts the corrected manuscript between the formats required by* `~!guri_`.

Description

This function converts a document using `~!guri_`. It takes the path to the article, the article id, the desired output format, and an optional verbose flag. It performs the conversion by calling the rmarkdown::pandoc_convert function.

Usage

```
guri_convert(art_path, art_id, output, verbose = TRUE)
```

Arguments

`art_path`	A string with the path to the article folder.
`art_id`	A string with the article id.
`output`	A string. The desired output format (see).
`verbose`	Logical. Specifies whether to display verbose output (default `TRUE`).

Value

Invisible TRUE

`guri_doi_batch` *Generate a* `doi_batch` *xml file for Crossref deposit*

Description

Generate a `doi_batch` xml file for Crossref deposit

Usage

```
guri_doi_batch(list_art, path_issue)
```

Arguments

`list_art`	A data.frame with the article information as returned by guri_list_articles. It must contain the following columns: `id` (character), with the article id; and `path` (character), with the path to the article.
`path_issue`	A string with the path to the issue folder.

Details

The doi_batch xml file is saved in the doi_register folder, inside the issue folder.

The doi_batch xml file searches each of the article folders for the art[id]_crossref.xml file (inside the '_temp' folder), and uses the information in these files to create a unified doi_batch xml file (each article is saved as a <journal_article> element in the xml file).

In adition to the doi_batch xml file, this function also creates a text file with the information of the articles that will be deposited in Crossref. The text file is saved in the same folder as the doi_batch xml file.

Value

A string with the path to the doi_batch xml file.

---

guri_install *Updates/installs the external dependencies necessary for working ~!guri_.*

Description

Upgrade or install (as necessary) pandoc, as well as the latex distribution (tinytex) and latex packages.

Usage

```
guri_install(pandoc = T, tinytex = T, force = F)
```

Arguments

pandoc	Logical. Should pandoc be installed/updated? (default 'TRUE')
tinytex	Logical. Should tinytex be installed/updated? (default 'TRUE')
force	Logical. Should pandoc and tinytex be forced to reinstall? (default 'FALSE')

Value

Invisible. A list with a logical vector indicating whether tinytex and pandoc are available and two items with the installed versions of Tinitex and Pandoc.

guri_list_articles *Lists the articles in a journal issue*

Description

Lists the articles in a journal issue

Usage

```
guri_list_articles(path_issue)
```

Arguments

path_issue	String with the path to the issue folder to list the articles in the issue.

Value

A tibble with two columns: articles path (art_path) and articles id (art_id).

guri_make_journal *Create the basic file structure for a new journal/journal repository*

Description

Create the basic file structure for a new journal or journal repository (to manage multiple journals).

Usage

```
guri_make_journal(
  journal = NULL,
  repository = FALSE,
  config_options = FALSE,
  example = FALSE,
  force = FALSE
)
```

Arguments

journal	A string with the short name of the journal. This 'short name' can contain only letters, numbers or a low dash (_) and must begin with a letter. Use only if you will work with the journal repository model (for single journal use NULL, default value). The journal name 'example' is not allowed.
repository	Logical. Will it work with the journal repository model (Default: FALSE). If TRUE is chosen, you must set a value for journal (unless example = TRUE) and a separate folder will be created for each journal. If FALSE the root folder will contain the files needed to manage your journal.

config_options	Options to generate the configuration files in '_config'. If FALSE (default), configuration files are not generated in the '_config' folder. If "default" or TRUE, the template and metadata files used by default are copied to the '_config' folder (so that they can be later modified to customise the journal). If it is a *named list* with the parameters from guri_config_journal, they will be passed to this function (e.g. customized = list(csl_name = "chicago-author-date-16th-edition").
example	Logical. Do you want to create the journal provided as an example? (Default = FALSE)
force	Logical. Should the journal be generated even if it already exists in the folder? This will ignore the '.guri' file if present. (Default: FALSE)

Details

Create the journal folder (if repository TRUE). The journal directory includes a configuration folder with the files used to configure the journal output (_config) and a folder with the basic files you will use for the production process (default-files). In addition, the _journal.yaml file will be generated, which you will have to edit manually with the basic journal data.

The folder structure can contain one journal (repository = TRUE) or multiple journals (repository = TRUE). Journal repositories allow to manage multiple journals in a single working environment. The internal structure of the journals within a repository is identical to that of a single journal.

The configuration file features (in _config) can be defined with this function during journal creation or independently using (see guri_config_journal).

If 'example = TRUE' a directory of the journal 'example' (.\example) is created with the necessary file structure to generate the final output files.

Value

Invisible returns the journal folder.

Examples

```
# Create a folder structure for a new journal.

guri_make_journal(journal = "new_journal", repository = TRUE)
fs::dir_tree("new_journal")

unlink("new_journal", recursive = TRUE)
unlink(".guri")

# Create a folder structure for the 'example journal'.

guri_make_journal(example = TRUE, repository = TRUE)
fs::dir_tree("example", type =  "directory")

unlink("example/", recursive = TRUE)
unlink(".guri")
```

guri_outputs *Generate output files for selected articles in an issue*

Description

For selected articles in an issue, generates output files in pdf, xml-jats and html format. In addition, it generates auxiliary and log files (see below for details). If `doi_batch = TRUE` is set, it also generates a single xml file to do the DOI deposit in Crossref for the selected articles.

Usage

```
guri_outputs(
  art_id,
  issue,
  journal = NULL,
  doi_batch = FALSE,
  verbose = TRUE,
  clean_files = TRUE
)
```

Arguments

`art_id`	String or vector of strings with the article id to be processed in the issue. If "all" all articles are processed.
`issue`	String. The issue folder.
`journal`	String (optional, mandatory if repository is `TRUE`). If the journal is not provided, it is assumed that the working directory is the journal repository. See guri_make_journal for details.
`doi_batch`	Logical. If `TRUE` a `doi_batch` file is created in the 'journal/issue/doi_register' folder (default `FALSE`).
`verbose`	Logical. Specifies whether to display verbose output (default `TRUE`).
`clean_files`	Logical. Should the temporary files be deleted and reordered in folders after the creation of the final files?. Primarily for debugging purposes (default is `TRUE`).

Details

The function generates the output files for each (selected) article in the issue folder. If art_id is "all", all articles in the issue folder are processed. The `journal` parameter is mandatory if it is a repository of journals, otherwise it will be `NULL`.

The function generates the following final files for each article:

- `art[id].xml`: a xml-jats file. See: https://jats.nlm.nih.gov/publishing/
- `art[id].html`: a html file with the article content.
- `art[id].pdf`: a pdf file with the article content.

Also, the following auxiliary files are created:

- art[id].md: a markdown file with the article content, used as an intermediate common format for the conversion to other formats.
- art[id].tex: a tex (latex) file used to generate the pdf.
- art[id]_crossref.xml: a xml file with the single article metadata for Crossref deposit. See: https://data.crossref.org/reports/help/schema_doc/5.3.1/index.html
- art[id]_biblio.json: a json file with the article references. Primarily useful for debugging purposes.
- art[id]_AST.native: the 'abstract syntax tree' used for Pandoc conversion.

In addition, if clean_files is TRUE, the function will create following folders in the article directory:

- _output: with the final files generated (xml-jats, html and pdf).
- _temp: with the temporary and auxiliary files generated during the process.
- _log: with the log files generated during the process (only present if verbose is TRUE).

If doi_batch = TRUE, the function makes a single xml file (in 'journal/issue/doi_register') with the metadata of all selected articles to do the DOI deposit in Crossref. An inform file is also created with the information present in the xml file.

Value

Invisible, a logical vector with the success of the process for each article.

---

guri_to_md *Converts the corrected manuscript between formats.*

Description

They convert between the different formats required for the ~!guri_ workflow, applying the appropriate lua filters at each stage and using the appropriate templates and metadata. The ~!guri_to_md function converts between the revised (and formatted) manuscript in docx format to markdown format. The guri_to_html, guri_to_jats and guri_to_pdf functions convert from the markdown file to html, xml (xml-jats) and pdf (tex) formats, respectively. Last, guri_to_AST and guri_biblio are auxiliary functions that generate a file with the Abstract Syntax Tree (mostly for debugging purposes) and a file with the references used by the article, respectively.

Usage

```
guri_to_md(art_path, art_id, verbose = TRUE)

guri_to_html(art_path, art_id, verbose = TRUE)

guri_to_jats(art_path, art_id, verbose = TRUE)

guri_to_pdf(art_path, art_id, verbose = TRUE, pdf = TRUE)

guri_to_crossref(art_path, art_id, verbose = TRUE)
```

```
guri_to_AST(art_path, art_id, verbose = TRUE)

guri_biblio(art_path, art_id, bib_type = "csljson")
```

Arguments

art_path	A string with the path to the article folder.
art_id	A string with the article id.
verbose	Logical. Specifies whether to display verbose output (default TRUE).
pdf	Logical. Should the pdf be generated? (default = TRUE)
bib_type	description

Details

These functions are, mostly, a wrapper of the internal function guri_convert. The functions are exported primarily for debugging and error detection purposes. For day-to-day work it is advisable to use the guri function directly, which coordinates the generation of all these formats and ensures the correct definition of the parameters.

For the generation of pdf files, LuaTex (tinytex::lualatex) is used internally.

Value

Invisible TRUE

Index