

Package: reportifyr (via r-universe)

September 27, 2026

Title Reproducible Reporting Made Simple with R
Version 0.4.0
Description Create reproducible reports with Microsoft Word and R.
License GPL (>= 3)
Encoding UTF-8
Roxygen list(markdown = TRUE)
RoxygenNote 7.3.3
Depends R (>= 4.1.0)
Imports assertthat, cli, digest, dplyr, flextable, fs, pyro, gert, ggplot2, here, jsonlite, log4r, officer, processx, purrr, rlang, this.path, tictoc, xml2, yaml
Suggests knitr, mockery, rmarkdown, testthat (>= 3.0.0), withr
Config/testthat/edition 3
LazyData true
URL <https://a2-ai.github.io/reportifyr/>
Repository <https://a2-ai.r-universe.dev>
Date/Publication 2026-05-28 16:06:02 UTC
RemoteUrl <https://github.com/a2-ai/reportifyr>
RemoteRef HEAD
RemoteSha 6761ebe2339d5db13886647cff04857a2050d198

Contents

add_footnotes	2
add_plots	4
add_plots_alt_text	6
add_tables	6
add_tables_alt_text	7
build_report	8
finalize_document	9

find_project_root	11
fit_flextable_to_page	11
format.rpfy_context	12
format_flextable	12
get_log_file	13
get_meta_abbrevs	13
get_meta_type	14
get_venv_uv_paths	15
ggsave_with_metadata	15
initialize_python	17
initialize_report_project	17
make_doc_dirs	18
preview_metadata	19
preview_metadata_files	19
print.rpfy_context	20
remove_tables_figures_footnotes	20
reportifyr	21
rpfy_context	23
save_rds_with_metadata	24
script_source	26
shiny_source	27
sync_report_project	27
toggle_logger	28
update_object_footnotes	29
validate_alt_text_magic_strings	30
validate_config	30
validate_docx	31
validate_object	31
validate_rpfy_context	32
validate_script_source	33
validate_shiny_source	33
write_csv_with_metadata	34
write_object_metadata	35

Index	37
--------------	-----------

add_footnotes	<i>Inserts Footnotes in appropriate places in a Microsoft Word file</i>
---------------	---

Description

Reads in a .docx file and returns a new version with footnotes placed at appropriate places in the document.

Usage

```
add_footnotes(
  docx_in,
  docx_out,
  figures_path,
  tables_path,
  standard_footnotes_yaml = NULL,
  config_yaml = NULL,
  include_object_path = FALSE,
  footnotes_fail_on_missing_metadata = TRUE,
  debug = FALSE
)
```

Arguments

docx_in	The file path to the input .docx file.
docx_out	The file path to the output .docx file to save to.
figures_path	The file path to the figures and associated metadata directory.
tables_path	The file path to the tables and associated metadata directory.
standard_footnotes_yaml	The file path to the standard_footnotes.yaml. Default is NULL. If NULL, a default standard_footnotes.yaml bundled with the reportifyr package is used.
config_yaml	The file path to the config.yaml. Default is NULL, a default config.yaml bundled with the reportifyr package is used.
include_object_path	A boolean indicating whether to include the file path of the figure or table in the footnotes. Default is FALSE.
footnotes_fail_on_missing_metadata	A boolean indicating whether to stop execution if the metadata .json file for a figure or table is missing. Default is TRUE.
debug	Debug.

Examples

```
## Not run:

# -----
# Load all dependencies
# -----
docx_in <- here::here("report", "shell", "template.docx")
doc_dirs <- make_doc_dirs(docx_in = docx_in)
figures_path <- here::here("OUTPUTS", "figures")
tables_path <- here::here("OUTPUTS", "tables")
standard_footnotes_yaml <- here::here("report", "standard_footnotes.yaml")

# -----
```

```

# Step 1.
# `add_tables()` will format and insert tables into the `.docx` file.
# -----
add_tables(
  docx_in = doc_dirs$doc_in,
  docx_out = doc_dirs$doc_tables,
  tables_path = tables_path
)

# -----
# Step 2.
# Next we insert the plots using the `add_plots()` function.
# -----
add_plots(
  docx_in = doc_dirs$doc_tables,
  docx_out = doc_dirs$doc_tabs_figs,
  figures_path = figures_path
)

# -----
# Step 3.
# Now we can add the footnotes with the `add_footnotes` function.
# -----
add_footnotes(
  docx_in = doc_dirs$doc_tabs_figs,
  docx_out = doc_dirs$doc_draft,
  figures_path = figures_path,
  tables_path = tables_path,
  standard_footnotes_yaml = standard_footnotes_yaml,
  include_object_path = FALSE,
  footnotes_fail_on_missing_metadata = TRUE
)

## End(Not run)

```

add_plots

Inserts Figures in appropriate places in a Microsoft Word file

Description

Reads in a .docx file and returns a new version with figures placed at appropriate places in the document.

Usage

```

add_plots(
  docx_in,
  docx_out,
  figures_path,
  config_yaml = NULL,

```

```

    fig_width = NULL,
    fig_height = NULL,
    debug = FALSE
  )

```

Arguments

docx_in	The file path to the input .docx file.
docx_out	The file path to the output .docx file to save to.
figures_path	The file path to the figures directory.
config_yaml	The file path to the config.yaml. Default is NULL, a default config.yaml bundled with the reportifyr package is used.
fig_width	A global controller. The figure width in inches. Default is NULL. If NULL, the width is determined by the figure's pixel dimensions.
fig_height	A global controller. The figure height in inches. Default is NULL. If NULL, the height is determined by the figure's pixel dimensions.
debug	Debug.

Examples

```

## Not run:

# -----
# Load all dependencies
# -----
docx_in <- here::here("report", "shell", "template.docx")
doc_dirs <- make_doc_dirs(docx_in = docx_in)
figures_path <- here::here("OUTPUTS", "figures")
tables_path <- here::here("OUTPUTS", "tables")
standard_footnotes_yaml <- here::here("report", "standard_footnotes.yaml")

# -----
# Step 1.
# `add_tables()` will format and insert tables into the `.docx` file.
# -----
add_tables(
  docx_in = doc_dirs$doc_in,
  docx_out = doc_dirs$doc_tables,
  tables_path = tables_path
)

# -----
# Step 2.
# Next we insert the plots using the `add_plots()` function.
# -----
add_plots(
  docx_in = doc_dirs$doc_tables,
  docx_out = doc_dirs$doc_tabs_figs,
  figures_path = figures_path
)

```

```
## End(Not run)
```

add_plots_alt_text	<i>Inserts alt text for figures within a Microsoft Word file.</i>
--------------------	---

Description

Inserts alt text for figures within a Microsoft Word file.

Usage

```
add_plots_alt_text(docx_in, docx_out, figures_path = NULL, debug = FALSE)
```

Arguments

docx_in	The file path to the input .docx file.
docx_out	The file path to the output .docx file to save to.
figures_path	The file path to the figures directory. Used to embed artifact hashes in alt text. Default is NULL.
debug	Debug.

Examples

```
## Not run:  
add_plots_alt_text("doc-figs.docx", "doc-draft.docx")  
  
## End(Not run)
```

add_tables	<i>Inserts Tables in appropriate places in a Microsoft Word file</i>
------------	--

Description

Reads in a .docx file and returns a new version with tables placed at appropriate places in the document.

Usage

```
add_tables(docx_in, docx_out, tables_path, config_yaml = NULL, debug = FALSE)
```

Arguments

docx_in	The file path to the input .docx file.
docx_out	The file path to the output .docx file to save to.
tables_path	The file path to the tables and associated metadata directory.
config_yaml	The file path to the config.yaml. Default is NULL, a default config.yaml bundled with the reportifyr package is used.
debug	Debug.

Examples

```
## Not run:

# -----
# Load all dependencies
# -----
docx_in <- here::here("report", "shell", "template.docx")
doc_dirs <- make_doc_dirs(docx_in = docx_in)
figures_path <- here::here("OUTPUTS", "figures")
tables_path <- here::here("OUTPUTS", "tables")
standard_footnotes_yaml <- here::here("report", "standard_footnotes.yaml")

# -----
# Step 1.
# `add_tables()` will format and insert tables into the `.docx` file.
# -----
add_tables(
  docx_in = doc_dirs$doc_in,
  docx_out = doc_dirs$doc_tables,
  tables_path = tables_path
)

## End(Not run)
```

add_tables_alt_text *Inserts alt text for tables within a Microsoft Word file.*

Description

Inserts alt text for tables within a Microsoft Word file.

Usage

```
add_tables_alt_text(docx_in, docx_out, tables_path = NULL, debug = FALSE)
```

Arguments

docx_in	The file path to the input .docx file.
docx_out	The file path to the output .docx file to save to.
tables_path	The file path to the tables directory. Used to embed artifact hashes in alt text. Default is NULL.
debug	Debug.

Examples

```
## Not run:
add_tables_alt_text("document-tabs.docx", "document-tabs_at.docx")

## End(Not run)
```

build_report	<i>Updates a Microsoft Word file to include formatted plots, tables, and footnotes</i>
--------------	--

Description

Reads in a .docx file and returns a new version with plots, tables, and footnotes replaced.

Usage

```
build_report(
  docx_in,
  docx_out = NULL,
  figures_path,
  tables_path,
  standard_footnotes_yaml = NULL,
  config_yaml = NULL,
  add_footnotes = TRUE,
  include_object_path = FALSE,
  footnotes_fail_on_missing_metadata = TRUE
)
```

Arguments

docx_in	The file path to the input .docx file.
docx_out	The file path to the output .docx file to save to. Default is NULL.
figures_path	The file path to the figures and associated metadata directory.
tables_path	The file path to the tables and associated metadata directory.
standard_footnotes_yaml	The file path to the standard_footnotes.yaml. Default is NULL. If NULL, a default standard_footnotes.yaml bundled with the reportifyr package is used.

`config_yaml` The file path to the `config.yaml`. Default is `NULL`, a default `config.yaml` bundled with the `reportifyr` package is used.

`add_footnotes` A boolean indicating whether to insert footnotes into the `docx_in` or not. Default is `TRUE`.

`include_object_path` A boolean indicating whether to include the file path of the figure or table in the footnotes. Default is `FALSE`.

`footnotes_fail_on_missing_metadata` A boolean indicating whether to stop execution if the metadata `.json` file for a figure or table is missing. Default is `TRUE`.

Examples

```
## Not run:

# -----
# Load all dependencies
# -----
docx_in <- here::here("report", "shell", "template.docx")
doc_dirs <- make_doc_dirs(docx_in = docx_in)
figures_path <- here::here("OUTPUTS", "figures")
tables_path <- here::here("OUTPUTS", "tables")
standard_footnotes_yaml <- here::here("report", "standard_footnotes.yaml")

# -----
# Step 1.
# Run the `build_report()` wrapper function to replace figures, tables, and
# footnotes in a `.docx` file.
# -----
build_report(
  docx_in = doc_dirs$doc_in,
  docx_out = doc_dirs$doc_draft,
  figures_path = figures_path,
  tables_path = tables_path,
  standard_footnotes_yaml = standard_footnotes_yaml
)

## End(Not run)
```

<code>finalize_document</code>	<i>Finalizes the Microsoft Word file by removing magic strings and bookmarks</i>
--------------------------------	--

Description

Reads in a `.docx` file and returns a finalized version with magic strings and bookmarks removed.

Usage

```
finalize_document(docx_in, docx_out = NULL, config_yaml = NULL)
```

Arguments

docx_in	The file path to the input .docx file.
docx_out	The file path to the output .docx file to save to. Default is NULL. If NULL, docx_out is assigned doc_dirs\$doc_final using make_doc_dirs(docx_in = docx_in).
config_yaml	The file path to the config.yaml. Default is NULL, a default config.yaml bundled with the reportifyr package is used.

Examples

```
## Not run:

# -----
# Load all dependencies
# -----
docx_in <- here::here("report", "shell", "template.docx")
doc_dirs <- make_doc_dirs(docx_in = docx_in)
figures_path <- here::here("OUTPUTS", "figures")
tables_path <- here::here("OUTPUTS", "tables")
standard_footnotes_yaml <- here::here("report", "standard_footnotes.yaml")

# -----
# Step 1.
# Run the `build_report()` wrapper function to replace figures, tables, and
# footnotes in a `.docx` file.
# -----
build_report(
  docx_in = doc_dirs$doc_in,
  docx_out = doc_dirs$doc_draft,
  figures_path = figures_path,
  tables_path = tables_path,
  standard_footnotes_yaml = standard_footnotes_yaml
)

# -----
# Step 2.
# If you are ready to finalize the `.docx` file, run the `finalize_document()`
# function. This will remove the ties between reportifyr and the document, so
# please be mindful!
# -----
finalize_document(
  docx_in = doc_dirs$doc_draft,
  docx_out = doc_dirs$doc_final
)

## End(Not run)
```

find_project_root	<i>Find the reportifyr project root directory</i>
-------------------	---

Description

Searches upward from start_path for a reportifyr init file (e.g., .report_init.json). Returns the directory containing the init file, or NULL if none is found.

Usage

```
find_project_root(start_path = getwd())
```

Arguments

start_path Path to start searching from. Defaults to the current working directory.

Value

Absolute path to the project root directory, or NULL if no init file is found.

Examples

```
## Not run:
find_project_root()

## End(Not run)
```

fit_flextable_to_page	<i>Autofits a flextable object, then fits the object to the page width</i>
-----------------------	--

Description

Autofits a flextable object, then fits the object to the page width

Usage

```
fit_flextable_to_page(ft, page_width = 6)
```

Arguments

ft A flextable object
 page_width The width of page in inches

Value

A flextable object fit to page

Examples

```
## Not run:
# Load library for examples:
library(flextable)

# Create a flextable object and fit to default page width:
ft <- flextable(iris)
fit_flextable_to_page(ft)

# Create a flextable object and specify page width to fit to:
ft <- flextable(iris)
fit_flextable_to_page(ft, page_width = 6.5)

## End(Not run)
```

format.rpfy_context	<i>Format a one-line summary of a reportifyr context</i>
---------------------	--

Description

Format a one-line summary of a reportifyr context

Usage

```
## S3 method for class 'rpfy_context'
format(x, ...)
```

Arguments

x	An rpfy_context object.
...	Unused.

Value

Character scalar suitable for logging.

format_flextable	<i>Formats data frames to a flextable specification</i>
------------------	---

Description

Formats data frames to a flextable specification

Usage

```
format_flextable(data_in, table1_format = FALSE)
```

Arguments

`data_in` The input data to be formatted. Must be either a data frame or a flextable object.

`table1_format` A boolean indicating whether to apply table1-style formatting. Default is FALSE.

Value

A formatted flextable

Examples

```
## Not run:
dt <- head(iris, 10)
format_flextable(
  data_in = dt
)

## End(Not run)
```

get_log_file	<i>Get the current log file path</i>
--------------	--------------------------------------

Description

Get the current log file path

Usage

```
get_log_file()
```

Value

path to the log file, or NULL if not set

get_meta_abbrevs	<i>Get meta abbreviations from standard_footnotes.yaml</i>
------------------	--

Description

Get meta abbreviations from standard_footnotes.yaml

Usage

```
get_meta_abbrevs(path_to_footnotes_yaml)
```

Arguments

path_to_footnotes_yaml

The file path to the standard_footnotes.yaml file.

Value

A list of meta_abbrevs to be called while performing an analysis

Examples

```
## Not run:
standard_footnotes_yaml <- here::here("report", "standard_footnotes.yaml")
meta_abbrevs <- get_meta_abbrevs(path_to_footnotes_yaml = standard_footnotes_yaml)

## End(Not run)
```

get_meta_type

Get meta types from standard_footnotes.yaml

Description

Get meta types from standard_footnotes.yaml

Usage

```
get_meta_type(path_to_footnotes_yaml)
```

Arguments

path_to_footnotes_yaml

The file path to the standard_footnotes.yaml file.

Value

A list of meta_type to be called while performing an analysis

Examples

```
## Not run:
standard_footnotes_yaml <- here::here("report", "standard_footnotes.yaml")
meta_type <- get_meta_type(path_to_footnotes_yaml = standard_footnotes_yaml)

## End(Not run)
```

get_venv_uv_paths	<i>get_venv_uv_paths (deprecated)</i>
-------------------	---------------------------------------

Description

Deprecated; forwards to `pyro::get_venv_uv_paths()`. Prefer `pyro::get_venv_uv_paths()` in new code.

Usage

```
get_venv_uv_paths()
```

Value

list of paths to uv and venv directory for calling py scripts

Examples

```
## Not run:
get_venv_uv_paths()

## End(Not run)
```

ggsave_with_metadata	<i>Wrapper around the ggplot2 ggsave function. Saves a ggplot (or other grid object) and captures analysis relevant metadata in a .json file</i>
----------------------	--

Description

Extension to the `ggsave()` function that allows capturing object metadata as a separate .json file.

Usage

```
ggsave_with_metadata(
  filename,
  plot = ggplot2::last_plot(),
  meta_type = "NA",
  meta_equations = NULL,
  meta_notes = NULL,
  meta_abbrevs = NULL,
  config_yaml = NULL,
  context = NULL,
  ...
)
```

Arguments

filename	The filename for the plot to save to.
plot	The plot object to save. Default is the last displayed plot (<code>ggplot2::last_plot()</code>).
meta_type	A string to specify the type of object. Default is "NA".
meta_equations	A string or vector of strings representing equations to include in the metadata. Default is NULL.
meta_notes	A string or vector of strings representing notes to include in the metadata. Default is NULL.
meta_abbrevs	A string or vector of strings representing abbreviations to include in the metadata. Default is NULL.
config_yaml	The file path to the <code>config.yaml</code> . Default is NULL. If provided, <code>add_path_overlay</code> in the config controls whether a path is stamped onto the saved PNG: "source" stamps the source script path, "object" stamps the saved image's path (relative to the project root), and "none" (default) disables the overlay.
context	An <code>rpfy_context</code> built via <code>rpfy_context()</code> . When NULL (default) an ephemeral context is constructed with all defaults. Passing an existing context avoids re-computing session-level fields on every artifact.
...	Additional arguments passed to the <code>ggplot2::ggsave()</code> function.

Examples

```
## Not run:

# -----
# Construct and save a simple ggplot
# -----
g <- ggplot2::ggplot(
  data = Theoph,
  ggplot2::aes(x = Time, y = conc, group = Subject)
) +
  ggplot2::geom_point() +
  ggplot2::geom_line() +
  ggplot2::theme_bw()

# Save a png using the wrapper function
figures_path <- here::here("OUTPUTS", "figures")
plot_file_name <- "01-12345-pk-timecourse1.png"
ggsave_with_metadata(filename = file.path(figures_path, plot_file_name))

## End(Not run)
```

initialize_python	<i>Initializes python virtual environment (deprecated)</i>
-------------------	--

Description

This function is deprecated; it forwards to `pyro::initialize_python()`. The Python environment (uv, Python interpreter, pinned dependencies) is owned by the pyro package; call `pyro::initialize_python()` directly in new code.

Usage

```
initialize_python(continue = NULL)
```

Arguments

continue	Optional argument to bypass asking user for confirmation to install python deps.
----------	--

Value

invisibly the metadata_file path

Examples

```
## Not run:  
initialize_python()  
  
## End(Not run)
```

initialize_report_project	<i>Create report directories within a project</i>
---------------------------	---

Description

Create report directories within a project

Usage

```
initialize_report_project(  
  project_dir,  
  report_dir_name = NULL,  
  outputs_dir_name = NULL  
)
```

Arguments

`project_dir` The file path to the main project directory where the directory structure will be created. The directory must already exist; otherwise, an error will be thrown.

`report_dir_name` The directory name for where reports will be saved. Default is NULL. If NULL, report will be used.

`outputs_dir_name` The directory name for where artifacts will be saved. Default is NULL. If NULL, OUTPUTS will be used.

Examples

```
## Not run:
initialize_report_project(project_dir = tempdir())

## End(Not run)
```

make_doc_dirs	<i>Helper function that defines document output paths</i>
---------------	---

Description

Helper function that defines document output paths

Usage

```
make_doc_dirs(docx_in)
```

Arguments

`docx_in` The file path to the input .docx file.

Value

A list of document paths

Examples

```
## Not run:

# -----
# Load all dependencies
# -----
docx_in <- here::here("report", "shell", "template.docx")
doc_dirs <- make_doc_dirs(docx_in = docx_in)

## End(Not run)
```

preview_metadata	<i>Previews a single metadata file for an object</i>
------------------	--

Description

Previews a single metadata file for an object

Usage

```
preview_metadata(file_name)
```

Arguments

file_name The file path of the file whose metadata you want to preview.

Value

A single row data frame consisting of metadata type and footnotes for the object supplied

Examples

```
## Not run:
figures_path <- here::here("OUTPUTS", "figures")
plot_file_name <- "myplot.png"
preview_metadata(file.path(figures_path, plot_file_name))

## End(Not run)
```

preview_metadata_files	<i>Preview all metadata .json files in a directory</i>
------------------------	--

Description

Preview all metadata .json files in a directory

Usage

```
preview_metadata_files(file_dir)
```

Arguments

file_dir The file path to a directory containing metadata .json files.

Value

A data frame of metadata footnotes and meta type

Examples

```
## Not run:
figures_path <- here::here("OUTPUTS", "figures")
preview_metadata_file(figures_path)

## End(Not run)
```

```
print.rpfy_context      Print a reportifyr context
```

Description

Print a reportifyr context

Usage

```
## S3 method for class 'rpfy_context'
print(x, ...)
```

Arguments

x	An rpfy_context object.
...	Unused.

Value

The context invisibly.

```
remove_tables_figures_footnotes
      Removes Tables, Figures, and Footnotes from a Word file
```

Description

Reads in a .docx file and returns a new version with tables, figures, and footnotes removed from the document.

Usage

```
remove_tables_figures_footnotes(
  docx_in,
  docx_out,
  config_yaml = NULL,
  figures_path = NULL,
  tables_path = NULL
)
```

Arguments

<code>docx_in</code>	The file path to the input .docx file.
<code>docx_out</code>	The file path to the output .docx file to save to.
<code>config_yaml</code>	The file path to the config.yaml. Default is NULL, a default config.yaml bundled with the reportifyr package is used.
<code>figures_path</code>	The file path to the figures directory. Used for hash-based skip of unchanged artifacts. Default is NULL.
<code>tables_path</code>	The file path to the tables directory. Used for hash-based skip of unchanged artifacts. Default is NULL.

Examples

```
## Not run:

# -----
# Load all dependencies
# -----
docx_in <- here::here("report", "shell", "template.docx")
doc_dirs <- make_doc_dirs(docx_in = docx_in)

# -----
# Removing tables, figures, and footnotes
# -----
remove_tables_figures_footnotes(
  docx_in = doc_dirs$doc_in,
  docx_out = doc_dirs$doc_clean
)

## End(Not run)
```

reportifyr

*reportifyr: An R package to aid in the drafting of reports.***Description**

This package aims to ease table, figure, and footnote insertion and formatting into reports.

reportifyr setup functions

- [initialize_report_project](#): Creates report directory with shell, draft, scripts, final sub-directories and adds standard_footnotes.yaml to /report directory. Initializes python virtual environment through a subcall to [initialize_python](#) creates OUTPUTS/figures, OUTPUTS/tables, OUTPUTS/listings directories
- [initialize_python](#): Creates virtual environment in options("venv_dir") if set or project root otherwise. Also installs python-docx and pyyaml packages.

Analysis output saving functions

- [ggsave_with_metadata](#): Wrapper for saving ggplot that also creates metadata for plot object
- [save_rds_with_metadata](#): Wrapper for saveRDS that also creates metadata for tabular object and saves as rtf
- [write_csv_with_metadata](#): Wrapper for write.csv that also creates metadata for tabular object and saves as rtf
- [format_flextable](#): Formats a tabular data object as a flextable with simple formatting.

metadata interaction functions

- [write_object_metadata](#): Creates a metadata.json file for the input object path. Called within all analysis output saving functions.
- [update_object_footnotes](#): Used to update footnotes fields within object metadata json files.
- [preview_metadata_files](#): Generates a data frame of all object metadata within input directory and displays object, meta type, equations notes, abbreviations.
- [preview_metadata](#): Generates the metadata data frame of the singular input file.
- [get_meta_type](#): Generates meta_type object to allow user to see available meta_types in standard_footnotes.yaml within report directory
- [get_meta_abbrevs](#): Generates meta_abbrev object to allow user to see available abbreviations in standard_footnotes.yaml within report directory.

Document interaction functions

- [add_tables](#): Adds tables into the word document
- [add_footnotes](#): Adds footnotes for figures and tables
- [add_plots](#): Adds plots into the word document
- [remove_tables_figures_footnotes](#): Removes all tables, figures, and footnotes associated with a magic string {rpfy}:object.ext
- [remove_magic_strings](#): Removes all magic strings from a document cutting its tie to reportifyr but producing a final document.
- [remove_bookmarks](#): Removes all bookmarks from document

Report building function

- [build_report](#): Wrapper function to remove old tables, figures, and footnotes and adds new ones. Calls [remove_tables_figures_footnotes](#), [make_doc_dirs](#), [add_plots](#), [add_tables](#), [add_footnotes](#)
- [finalize_document](#): Wrapper function to remove magic strings and bookmarks from document.

Utility functions

- [make_doc_dirs](#): FILL IN
- [fit_flextable_to_page](#): FILL IN
- [validate_object](#): FILL IN
- [validate_config](#): FILL IN

Author(s)

Maintainer: Jacob Dumbleton <jacob@a2-ai.com>

Authors:

- Matthew Smith <matthews@a2-ai.com>
- Theodoros Papathanasiou <theo.x.papathanasiou@gsk.com>
- Tarjinder Sahota <tarjinder.z.sahota@gsk.com>
- Devin Pastoor <devin@a2-ai.com>
- Agnes Kim <agnes@a2-ai.com>
- Anne Zheng <anne@a2-ai.com>
- Jenna Johnson <jenna@a2-ai.com>
- Fernando Carreno <fernando.o.carreno@gsk.com>

Other contributors:

- Kashyap Patel <kashyap.x.patel@gsk.com> [contributor]
- Maxwell Chirehwa <maxwell.t.chirehwa@gsk.com> [contributor]
- Wesley Cummings <wes@a2-ai.com> [contributor]
- Elizabeth LeBeau <elizabeth@a2-ai.com> [contributor]

See Also

Useful links:

- <https://a2-ai.github.io/reportifyr/>

rpfy_context

Build a reportifyr metadata context

Description

Captures session-level metadata once so that it can be reused across many *_with_metadata() wrapper calls without recomputing on every artifact. Construct the context at the top of a script or Shiny app and pass it via the context argument to [ggsave_with_metadata\(\)](#), [save_rds_with_metadata\(\)](#), [write_csv_with_metadata\(\)](#), or [write_object_metadata\(\)](#).

Usage

```
rpfy_context(origin = NULL, addl_metadata = NULL)
```

Arguments

- | | |
|---------------|---|
| origin | Controls what is recorded in source_meta in the artifact JSON. One of: <ul style="list-style-type: none"> • NULL (default) — auto-detect via this.path + git lookup, producing a script_source (or an empty list if detection fails). • An rpfy_source_meta object — the output of shiny_source() or script_source(). |
| addl_metadata | Optional named list of caller-supplied provenance tags, written under the addl_meta key in the JSON. Values must be scalar character, numeric, or logical. Purely informational; never rendered into the report footer. |

Value

An object of class rpfy_context — a list with fields system_meta, source_meta, addl_metadata, author, project_root.

Examples

```
## Not run:
ctx <- rpfy_context(
  origin = shiny_source("myapp"),
  addl_metadata = list(analyst = "jake", study = "PK-001")
)

print(ctx)
validate_rpfy_context(ctx)

ggsave_with_metadata("plot.png", context = ctx, meta_type = "efficacy")

## End(Not run)
```

save_rds_with_metadata

Wrapper around the saveRDS function. Saves an object as .RDS and .RTF and captures analysis relevant metadata in a .json file

Description

Extension to the saveRDS() function that allows capturing object metadata as a separate .json file.

Usage

```
save_rds_with_metadata(
  object,
  file = "",
  config_yaml = NULL,
  meta_type = "NA",
  meta_equations = NULL,
  meta_notes = NULL,
```

```

    meta_abbrevs = NULL,
    table1_format = FALSE,
    context = NULL,
    ...
)

```

Arguments

<code>object</code>	The R object to serialize.
<code>file</code>	The connection or name of the file where the R object is saved.
<code>config_yaml</code>	The file path to the <code>config.yaml</code> . Default is <code>NULL</code> . If <code>NULL</code> , a default value of <code>TRUE</code> for <code>save_rtf()</code> is used.
<code>meta_type</code>	A string to specify the type of object. Default is <code>"NA"</code> .
<code>meta_equations</code>	A string or vector of strings representing equations to include in the metadata. Default is <code>NULL</code> .
<code>meta_notes</code>	A string or vector of strings representing notes to include in the metadata. Default is <code>NULL</code> .
<code>meta_abbrevs</code>	A string or vector of strings representing abbreviations to include in the metadata. Default is <code>NULL</code> .
<code>table1_format</code>	A boolean indicating whether to apply table1-style formatting. Default is <code>FALSE</code> .
<code>context</code>	An <code>rpfy_context</code> built via <code>rpfy_context()</code> . When <code>NULL</code> (default) an ephemeral context is constructed with all defaults. Passing an existing context avoids re-computing session-level fields on every artifact.
<code>...</code>	Additional arguments passed to the <code>saveRDS()</code> function.

Examples

```

## Not run:

# -----
# Save a simple table
# -----
tables_path <- here::here("OUTPUTS", "tables")
outfile_name <- "01-12345-pk-theoph.RDS"

save_rds_with_metadata(
  object = Theoph,
  file = file.path(tables_path, outfile_name)
)

## End(Not run)

```

script_source

Build a structured source value for an artifact produced from a script

Description

Returns a classed list suitable for the origin argument of `rpfy_context()`. Typically produced internally by `rpfy_context()` via auto-detection; exposed for callers that want to stamp a known script identity. The type discriminator is not stored on the list; it is emitted at serialization time so the class is the single source of truth.

Usage

```
script_source(path, creation_author, latest_author, creation_time, latest_time)
```

Arguments

path	Character scalar. Path to the source script, relative to the project root.
creation_author	Character scalar. Git author of the first commit touching the script.
latest_author	Character scalar. Git author of the most recent commit touching the script.
creation_time	Character scalar. Timestamp of the first commit.
latest_time	Character scalar. Timestamp of the most recent commit.

Value

A named list with `creation_author`, `latest_author`, `path`, `creation_time`, and `latest_time` elements, classed as `c("script_source", "rpfy_source_meta")`.

Examples

```
## Not run:
src <- script_source(
  path          = "scripts/01-pk.R",
  creation_author = "jake",
  latest_author  = "jake",
  creation_time  = "2026-01-01 00:00:00",
  latest_time    = "2026-04-20 12:00:00"
)
ctx <- rpfy_context(origin = src)

## End(Not run)
```

shiny_source	<i>Build a structured source value for an artifact produced from a Shiny app</i>
--------------	--

Description

Returns a classed list suitable for the `origin` argument of `rpfy_context()`. The type discriminator is not stored on the list; it is emitted at serialization time so the class is the single source of truth.

Usage

```
shiny_source(app_name, app_version = NULL)
```

Arguments

- `app_name` Character scalar. Name of the installed R package that is the Shiny app.
- `app_version` Optional character scalar version. When `NULL` (default) the version is resolved via `utils::packageVersion()`, which requires the app to be installed as a package.

Value

A named list with `app_name` and `app_version` elements, classed as `c("shiny_source", "rpfy_source_meta")`.

Examples

```
## Not run:
ctx <- rpfy_context(origin = shiny_source("myapp"))
ggsave_with_metadata(
  filename = "OUTPUTS/figures/plot.png",
  context = ctx
)

## End(Not run)
```

sync_report_project	<i>Synchronizes report project with the bundled pyro Python environment and reconciles the report config.yaml against the .<report>_init.json snapshot. Always runs <code>pyro::initialize_python(groups = "reportifyr")</code>; prompts the user only when the project <code>.venv</code> is missing, otherwise runs non-interactively (a no-op when <code>uv</code> reports no work needed).</i>
---------------------	--

Description

Synchronizes report project with the bundled pyro Python environment and reconciles the report config.yaml against the .<report>_init.json snapshot. Always runs pyro::initialize_python(groups = "reportifyr"); prompts the user only when the project .venv is missing, otherwise runs non-interactively (a no-op when uv reports no work needed).

Usage

```
sync_report_project(project_dir, report_dir_name = NULL)
```

Arguments

- project_dir The file path to the main project directory where the directory structure will be created. The directory must already exist; otherwise, an error will be thrown.
- report_dir_name The directory name for where reports will be saved. Default is NULL. If NULL, report will be used.

Examples

```
## Not run:
sync_report_project(here::here())

## End(Not run)
```

toggle_logger	<i>Updates the logging level for console output. Default is set to WARN. File logging is always at DEBUG level when log_file is provided. Set options(rpfy.no_log = TRUE) to suppress file logging for the session (console output is unaffected).</i>
---------------	--

Description

Updates the logging level for console output. Default is set to WARN. File logging is always at DEBUG level when log_file is provided. Set options(rpfy.no_log = TRUE) to suppress file logging for the session (console output is unaffected).

Usage

```
toggle_logger(quiet = FALSE, log_file = get_log_file(), lazy_file = FALSE)
```

Arguments

- quiet suppresses messaging about log level.
- log_file path to log file. Defaults to the current session log file set on package load. Use get_log_file() to retrieve the current path. Pass a custom path to write to a different file.
- lazy_file if TRUE, delay log file creation until first log write.

Examples

```
## Not run:
Sys.setenv("RPFY_VERBOSE" = "DEBUG")
toggle_logger()

## End(Not run)
```

update_object_footnotes

Updates an object's footnote metadata - equations, notes, or abbreviations

Description

Updates an object's footnote metadata - equations, notes, or abbreviations

Usage

```
update_object_footnotes(
  file_path,
  overwrite = FALSE,
  meta_equations = NULL,
  meta_notes = NULL,
  meta_abbrevs = NULL
)
```

Arguments

file_path	The file path to the object or its metadata file.
overwrite	A boolean indicating whether to overwrite existing metadata entries. Default is FALSE (appends to existing entries).
meta_equations	A string or vector of strings representing equations to include or overwrite in the metadata.
meta_notes	A string or vector of strings representing notes to include or overwrite in the metadata.
meta_abbrevs	A string or vector of strings representing abbreviations to include or overwrite in the metadata.

Examples

```
## Not run:
update_object_footnotes("example_metadata.json", equations = c("K10 = CL/VC", "K12 = Q/VC"))

## End(Not run)
```

`validate_alt_text_magic_strings`*Validate alt text of figures/tables against their magic strings in a Microsoft Word file*

Description

Validate alt text of figures/tables against their magic strings in a Microsoft Word file

Usage

```
validate_alt_text_magic_strings(docx_in, debug = FALSE)
```

Arguments

<code>docx_in</code>	The file path to the input .docx file.
<code>debug</code>	Debug.

Examples

```
## Not run:  
validate_alt_text_magic_strings("template.docx")  
  
## End(Not run)
```

`validate_config`*Validate config.yaml file*

Description

Validate config.yaml file

Usage

```
validate_config(path_to_config_yaml)
```

Arguments

<code>path_to_config_yaml</code>	The file path to the config.yaml.
----------------------------------	-----------------------------------

Value

bool TRUE if config is valid, FALSE otherwise.

Examples

```
## Not run:
validate_config(here::here("report/config.yaml"))

## End(Not run)
```

validate_docx	<i>Validates input Microsoft Word file to ensure proper functionality with reportifyr</i>
---------------	---

Description

Validates input Microsoft Word file to ensure proper functionality with reportifyr

Usage

```
validate_docx(docx_in, config_yaml)
```

Arguments

docx_in	The file path to the input .docx file.
config_yaml	The file path to the config.yaml.

Examples

```
## Not run:
validate_docx(
  here::here("report/shell/template.docx"),
  here::here("report/config.yaml")
)

## End(Not run)
```

validate_object	<i>Validates a file's hash against a stored hash in the associated _metadata.json file</i>
-----------------	--

Description

Validates a file's hash against a stored hash in the associated _metadata.json file

Usage

```
validate_object(file)
```

Arguments

file The connection or name of the file where the R object is saved.

Value

A boolean declaring if the hashes are equal or not

Examples

```
## Not run:
tables_path <- here::here("OUTPUTS", "tables")
outfile_name <- "01-12345-pk-theoph.csv"

validate_object(file = file.path(tables_path, out_name))

## End(Not run)
```

validate_rpfy_context *Validate a reportifyr context*

Description

Checks that an rpfy_context has the expected fields and that each field holds a valid value. Errors on any problem.

Usage

```
validate_rpfy_context(context)
```

Arguments

context An rpfy_context object.

Value

The context invisibly.

`validate_script_source`*Validate a script_source object*

Description

Errors if `x` is not a `script_source` or if any of `path`, `creation_author`, `latest_author`, `creation_time`, `latest_time` are not non-empty character scalars.

Usage

```
validate_script_source(x)
```

Arguments

`x` A `script_source` object.

Value

`x` invisibly.

`validate_shiny_source` *Validate a shiny_source object*

Description

Errors if `x` is not a `shiny_source` or if `app_name` / `app_version` are not non-empty character scalars.

Usage

```
validate_shiny_source(x)
```

Arguments

`x` A `shiny_source` object.

Value

`x` invisibly.

```
write_csv_with_metadata
```

Wrapper around the write.csv function. Saves data as .RDS and .RTF and captures analysis relevant metadata in a .json file

Description

Extension to the `write.csv()` function that allows capturing object metadata as a separate .json file.

Usage

```
write_csv_with_metadata(
  object,
  file,
  config_yaml = NULL,
  meta_type = "NA",
  meta_equations = NULL,
  meta_notes = NULL,
  meta_abbrevs = NULL,
  table1_format = FALSE,
  context = NULL,
  ...
)
```

Arguments

<code>object</code>	The R object to serialize.
<code>file</code>	The connection or name of the file where the R object is saved.
<code>config_yaml</code>	The file path to the config.yaml. Default is NULL. If NULL, a default value of TRUE for <code>save_rtf()</code> is used.
<code>meta_type</code>	A string to specify the type of object. Default is "NA".
<code>meta_equations</code>	A string or vector of strings representing equations to include in the metadata. Default is NULL.
<code>meta_notes</code>	A string or vector of strings representing notes to include in the metadata. Default is NULL.
<code>meta_abbrevs</code>	A string or vector of strings representing abbreviations to include in the metadata. Default is NULL.
<code>table1_format</code>	A boolean indicating whether to apply table1-style formatting. Default is FALSE.
<code>context</code>	An <code>rpfy_context</code> built via <code>rpfy_context()</code> . When NULL (default) an ephemeral context is constructed with all defaults. Passing an existing context avoids re-computing session-level fields on every artifact.
<code>...</code>	Additional arguments passed to the <code>utils::write.csv()</code> function.

Examples

```
## Not run:

# -----
# Save a simple table
# -----
tables_path <- here::here("OUTPUTS", "tables")
outfile_name <- "01-12345-pk-theoph.csv"

write_csv_with_metadata(
  object = Theoph,
  file = file.path(tables_path, outfile_name),
  row_names = FALSE
)

## End(Not run)
```

`write_object_metadata` *Writes an object's metadata .json file*

Description

Writes an object's metadata .json file

Usage

```
write_object_metadata(
  object_file,
  meta_type = NULL,
  meta_equations = NULL,
  meta_notes = NULL,
  meta_abbrevs = NULL,
  table1_format = FALSE,
  context = NULL
)
```

Arguments

<code>object_file</code>	The file path of the object to write metadata for.
<code>meta_type</code>	A string to specify the type of object. Default is "NA".
<code>meta_equations</code>	A string or vector of strings representing equations to include in the metadata. Default is NULL.
<code>meta_notes</code>	A string or vector of strings representing notes to include in the metadata. Default is NULL.
<code>meta_abbrevs</code>	A string or vector of strings representing abbreviations to include in the metadata. Default is NULL.

table1_format	A boolean indicating whether table1 formatting is used for add_tables(). Default is FALSE.
context	An rpfy_context built via rpfy_context() . When NULL (default) an ephemeral context is constructed with all defaults. Passing an existing context avoids re-computing session-level fields on every artifact.

Examples

```
## Not run:
figures_path <- here::here("OUTPUTS", "figures")
plot_file_name <- "01-12345-pk-timecourse1.png"

write_object_metadata(object_file = file.path(figures_path, plot_file_name))

## End(Not run)
```

Index

`add_footnotes`, [2](#), [22](#)
`add_plots`, [4](#), [22](#)
`add_plots_alt_text`, [6](#)
`add_tables`, [6](#), [22](#)
`add_tables_alt_text`, [7](#)

`build_report`, [8](#), [22](#)

`finalize_document`, [9](#), [22](#)
`find_project_root`, [11](#)
`fit_flextable_to_page`, [11](#), [22](#)
`format.rpfy_context`, [12](#)
`format_flextable`, [12](#), [22](#)

`get_log_file`, [13](#)
`get_meta_abbrevs`, [13](#), [22](#)
`get_meta_type`, [14](#), [22](#)
`get_venv_uv_paths`, [15](#)
`ggsave_with_metadata`, [15](#), [22](#)
`ggsave_with_metadata()`, [23](#)

`initialize_python`, [17](#), [21](#)
`initialize_report_project`, [17](#), [21](#)

`make_doc_dirs`, [18](#), [22](#)

`preview_metadata`, [19](#), [22](#)
`preview_metadata_files`, [19](#), [22](#)
`print.rpfy_context`, [20](#)

`remove_bookmarks`, [22](#)
`remove_magic_strings`, [22](#)
`remove_tables_figures_footnotes`, [20](#), [22](#)
`reportifyr`, [21](#)
`reportifyr-package (reportifyr)`, [21](#)
`rpfy_context`, [23](#)
`rpfy_context()`, [16](#), [25–27](#), [34](#), [36](#)

`save_rds_with_metadata`, [22](#), [24](#)
`save_rds_with_metadata()`, [23](#)
`script_source`, [26](#)

`script_source()`, [24](#)
`shiny_source`, [27](#)
`shiny_source()`, [24](#)
`sync_report_project`, [27](#)

`toggle_logger`, [28](#)

`update_object_footnotes`, [22](#), [29](#)

`validate_alt_text_magic_strings`, [30](#)
`validate_config`, [22](#), [30](#)
`validate_docx`, [31](#)
`validate_object`, [22](#), [31](#)
`validate_rpfy_context`, [32](#)
`validate_script_source`, [33](#)
`validate_shiny_source`, [33](#)

`write_csv_with_metadata`, [22](#), [34](#)
`write_csv_with_metadata()`, [23](#)
`write_object_metadata`, [22](#), [35](#)
`write_object_metadata()`, [23](#)