

Package ‘toolero’

April 27, 2026

Title A Toolkit for Research Workflows

Version 0.3.0

Description Provides utility functions to help researchers implement best practices for their coding projects. Includes tools for reading and cleaning data files, initializing R projects with a standard folder structure, creating Quarto documents from a reproducible template, detecting the execution context across interactive, Quarto, and script-based workflows, and splitting data frames into group-level output files.

License MIT + file LICENSE

Depends R (>= 4.2.0)

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0),

Config/testthat/edition 3

Imports cli, fs, glue, janitor, purrr, readr, renv, tibble, usethis, yaml, rlang, rvest, xml2, quarto, withr

URL <https://github.com/erwinlares/toolero>,

<https://erwinlares.github.io/toolero/>

BugReports <https://github.com/erwinlares/toolero/issues>

VignetteBuilder knitr

NeedsCompilation no

Author Erwin Lares [aut, cre] (ORCID: <<https://orcid.org/0000-0002-3284-828X>>)

Maintainer Erwin Lares <erwin.lares@wisc.edu>

Repository CRAN

Date/Publication 2026-04-27 18:40:02 UTC

Contents

create_qmd	2
detect_execution_context	3
generate_kb_xml	4
init_project	6
read_clean_csv	7
write_by_group	8
Index	10

create_qmd	<i>Create a new Quarto document from a template</i>
------------	---

Description

Creates a new Quarto document in the specified directory, along with a sample dataset and UW-Madison branded assets. Optionally pre-populates the YAML header with user-supplied metadata.

Usage

```
create_qmd(
  filename = NULL,
  path = ".",
  yaml_data = NULL,
  overwrite = FALSE,
  use_purl = TRUE
)
```

Arguments

filename	A string or NULL. Name of the generated .qmd file. Must be supplied explicitly, e.g. "analysis.qmd".
path	A string. Path to the directory where the document will be created. Defaults to "." (the current working directory).
yaml_data	A string or NULL. Path to a YAML file containing metadata to pre-populate the document header. If NULL (the default), the template is copied as-is with placeholder prompts intact.
overwrite	A logical. Whether to overwrite existing files. Defaults to FALSE.
use_purl	Logical. If TRUE (the default), creates a _quarto.yml file with a post-render hook and a purl.R script inside R/ that extracts R code from the rendered document into a .R file. The target document is resolved dynamically by scanning the project root for .qmd files, so the same purl.R works regardless of the document name.

Details

`create_qmd()` performs the following steps:

1. Validates that path exists.
2. Validates that filename is supplied.
3. Creates a `data/` folder under path and copies `sample.csv` there.
4. Checks for `assets/styles.css` and `assets/header.html` - creates the `assets/` folder if needed and copies both from the package.
5. Copies the template `.qmd` to `path/filename`.
6. If `yaml_data` is provided, reads the YAML file and substitutes values into the document header.
7. If `use_purl = TRUE`, writes a `_quarto.yml` with a post-render hook pointing to `R/purl.R`, and copies `purl.R` from the package templates into `path/R/purl.R`.

Note: `filename` has no default value and must always be supplied explicitly. Use `tempdir()` for temporary output during testing or exploration.

Value

Invisibly returns path.

Examples

```
# Create a document in a temp directory
create_qmd(path = tempdir(), filename = "analysis.qmd")

# Create with a custom filename, without the purl hook
create_qmd(path = tempdir(), filename = "report.qmd",
           overwrite = TRUE, use_purl = FALSE)

# Create with pre-populated YAML
yaml_file <- tempfile(fileext = ".yaml")
writeLines("author:\n - name: 'Your Name'", yaml_file)
create_qmd(path = tempdir(), filename = "analysis.qmd",
           yaml_data = yaml_file, overwrite = TRUE)
```

detect_execution_context

Detect the current execution context

Description

Identifies which of three execution environments the code is currently running in: an interactive R session, a `quarto` render call, or a plain Rscript invocation. This is useful for writing code that behaves correctly across all three contexts, such as resolving input file paths in a portable way.

Usage

```
detect_execution_context(interactive_fn = interactive)
```

Arguments

`interactive_fn` A function. Used to detect whether the session is interactive. Defaults to `base::interactive`. Override in tests to simulate different execution environments.

Details

Detection follows a priority order:

1. If `interactive()` is TRUE, returns "interactive".
2. If the environment variable `QUARTO_DOCUMENT_PATH` is set and non-empty, returns "quarto".
3. Otherwise, returns "rscript".

Value

A character string, one of "interactive", "quarto", or "rscript".

Examples

```
context <- detect_execution_context()

input_file <- switch(context,
  interactive = "data/sample.csv",
  quarto      = params$input_file,
  rscript     = commandArgs(trailingOnly = TRUE)[1]
)
```

generate_kb_xml

Generate a KB-importable XML file from a Quarto document

Description

Takes a Quarto document and produces an XML file that is directly importable into a UW-Madison Knowledge Base (KB) article. The function re-renders the `.qmd` with `embed-resources: true` so all visual assets are self-contained, extracts the HTML body, and wraps it in the KB XML structure along with metadata drawn from the document's YAML header.

Usage

```
generate_kb_xml(html_path, qmd_path = NULL, output_dir = NULL)
```

Arguments

html_path	A string. Path to the rendered HTML file. Used to infer the output filename and, if qmd_path is NULL, the location of the source .qmd.
qmd_path	A string or NULL. Path to the source .qmd file. If NULL (the default), inferred by replacing the .html extension of html_path with .qmd.
output_dir	A string or NULL. Directory where the .xml file will be written. If NULL (the default), written to the same directory as html_path.

Details

generate_kb_xml() performs the following steps:

1. Validates that html_path exists.
2. Infers qmd_path from html_path if not supplied, then validates it.
3. Extracts title, description, and categories from the .qmd YAML header and maps them to kb_title, kb_summary, and kb_keywords.
4. Re-renders the .qmd in an isolated temporary directory with embed-resources: true so all CSS, images, and JS are self-contained. The data/ and assets/ folders are copied alongside the .qmd to ensure the render succeeds.
5. Extracts the <body> from the embedded HTML.
6. Escapes HTML entities in the body for XML compatibility, as required by the UW-Madison KB import format.
7. Builds the XML structure with kb_title, kb_keywords, kb_summary, and kb_body nodes.
8. Writes the .xml file to output_dir.

Temporary files are managed via withr::local_tempdir() and are automatically cleaned up when the function exits, even on error.

When importing the resulting XML into the KB, check the *Decode HTML entity in body content* option.

Value

Invisibly returns the path to the written .xml file.

Examples

```
# Infer qmd_path automatically, write XML alongside the HTML
# generate_kb_xml(html_path = "docs/analysis.html")

# Supply qmd_path explicitly and write to a specific output directory
# generate_kb_xml(
#   html_path = "docs/analysis.html",
#   qmd_path  = "analysis.qmd",
#   output_dir = "exports"
# )
```

init_project	<i>Initialize a new R project with a standard folder structure</i>
--------------	--

Description

`init_project()` creates a new R project at the given path with an opinionated folder structure suited for research workflows. It optionally initializes `renv` for package management and `git` for version control.

Usage

```
init_project(  
  path,  
  use_renv = TRUE,  
  use_git = TRUE,  
  extra_folders = NULL,  
  open = FALSE,  
  uw_branding = FALSE  
)
```

Arguments

<code>path</code>	A character string with the path and name of the new project (e.g., <code>"~/Documents/my-project"</code>).
<code>use_renv</code>	Logical. If <code>TRUE</code> , initializes <code>renv</code> in the new project. Defaults to <code>TRUE</code> .
<code>use_git</code>	Logical. If <code>TRUE</code> , initializes a git repository in the new project. Defaults to <code>TRUE</code> .
<code>extra_folders</code>	A character vector of additional folder names to create inside the project. Defaults to <code>NULL</code> .
<code>open</code>	Logical. If <code>TRUE</code> , opens the new project in RStudio after creation. Defaults to <code>TRUE</code> .
<code>uw_branding</code>	Logical. If <code>TRUE</code> , creates an <code>assets/</code> folder and populates it with UW-Madison RCI branding files (<code>styles.css</code> , <code>header.html</code> , <code>rci-banner.png</code>). Defaults to <code>FALSE</code> .

Value

Called for its side effects. Does not return a value.

Examples

```
## Not run:  
init_project(path = file.path(tempdir(), "project1"),  
             use_renv = FALSE, use_git = FALSE)  
  
init_project(path = file.path(tempdir(), "project2"),  
             uw_branding = TRUE, use_renv = FALSE, use_git = FALSE)
```

```
init_project(path = file.path(tempdir(), "project3"),
             extra_folders = c("notebooks"),
             use_renv = FALSE, use_git = FALSE)

## End(Not run)
```

read_clean_csv	<i>Read and clean a CSV file</i>
----------------	----------------------------------

Description

`read_clean_csv()` reads a CSV file and cleans the column names in one step. It leverages `readr::read_csv()` for reading and `janitor::clean_names()` for making column names tidyverse-friendly (lower-case, no spaces, no special characters). By default, column type messages are suppressed. Set `verbose = TRUE` to display them.

Usage

```
read_clean_csv(file_path, verbose = FALSE)
```

Arguments

<code>file_path</code>	A character string with the path to the CSV file.
<code>verbose</code>	Logical. If TRUE, displays column type messages from <code>readr::read_csv()</code> . Defaults to FALSE.

Value

A tibble with clean column names.

Examples

```
# Read and clean a CSV file silently
sample_path <- system.file("templates", "sample.csv", package = "toolero")
data <- read_clean_csv(sample_path)

# Show column type messages
data <- read_clean_csv(sample_path, verbose = TRUE)
```

write_by_group	<i>Split a data frame by a grouping column and write each group to a CSV file</i>
----------------	---

Description

Splits a data frame by a single grouping column and writes each group to a separate CSV file. Optionally writes a manifest file listing the output files, their group values, and row counts.

Usage

```
write_by_group(data, group_col, output_dir = NULL, manifest = FALSE)
```

Arguments

data	A data frame or tibble to split and save.
group_col	A string. The name of the column to group by.
output_dir	A string or NULL. Path to the directory where output files will be written. Created if it does not exist. If NULL, the user must supply a path explicitly.
manifest	A logical. Whether to write a manifest.csv file to output_dir listing the output files, group values, and row counts. Defaults to FALSE.

Details

Output filenames are derived from the group values of group_col. Values are sanitized before use as filenames: converted to lowercase, spaces and special characters replaced with -, consecutive dashes collapsed, and leading/trailing dashes stripped.

If manifest = TRUE, a manifest.csv is written to output_dir containing three columns: group_value, n_rows, and file_path.

Note: output_dir has no default value. Always supply an explicit path to avoid writing files to unexpected locations. Use tempdir() for temporary output during testing or exploration.

Value

Invisibly returns output_dir.

Examples

```
# Split a small data frame by group and write to a temp directory
data <- data.frame(
  species = c("Adelie", "Adelie", "Gentoo"),
  mass    = c(3750, 3800, 5000)
)
write_by_group(data, group_col = "species", output_dir = tempdir())

# Same but also write a manifest
write_by_group(data, group_col = "species",
```

write_by_group

9

```
output_dir = tempdir(), manifest = TRUE)
```

Index

`create_qmd`, 2

`detect_execution_context`, 3

`generate_kb_xml`, 4

`init_project`, 6

`read_clean_csv`, 7

`write_by_group`, 8