Package 'toolero'

April 24, 2026

Title A Toolkit for Research Workflows

Version 0.2.0

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), withr

Config/testthat/edition 3

Imports cli, fs, glue, janitor, purrr, readr, renv, tibble, usethis, yaml

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 2026-04-24 18:40:02 UTC

Contents

`create_qmd` *Create a new Quarto document from a template*

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(
  path = NULL,
  filename = "analysis.qmd",
  yaml_data = NULL,
  overwrite = FALSE
)
```

Arguments

`path`	A string or `NULL`. Path to the directory where the document will be created. If `NULL`, the user must supply a path explicitly.
`filename`	A string. Name of the generated `.qmd` file. Defaults to `"analysis.qmd"`.
`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`.

Details

`create_qmd()` performs the following steps:

1. Validates that `path` exists.
2. Creates a `data/` folder under `path` and copies `sample.csv` there.
3. Checks for `assets/styles.css` and `assets/header.html` - creates the `assets/` folder if needed and copies both from the package.
4. Copies the template `.qmd` to `path/filename`.

5. If `yaml_data` is provided, reads the YAML file and substitutes values into the document header.

Note: `path` 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 `path`.

Examples

```
# Create with placeholder YAML in a temp directory
create_qmd(path = tempdir())

# Create with a custom filename
create_qmd(path = tempdir(), filename = "report.qmd", overwrite = TRUE)

# Create with pre-populated YAML
yaml_file <- tempfile(fileext = ".yml")
writeLines("author:\n  - name: 'Your Name'", yaml_file)
create_qmd(path = tempdir(), 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]
)
```

---

init_project *Initialize a new R project with a standard folder structure*

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

Arguments

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

Value

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

Examples

```
init_project(file_path = file.path(tempdir(), "project1"),
             use_renv = FALSE, use_git = FALSE)

init_project(file_path = file.path(tempdir(), "project2"),
             uw_branding = TRUE, use_renv = FALSE, use_git = FALSE)

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

---

read_clean_csv *Read and clean a CSV file*

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 (lowercase, 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

`file_path`	A character string with the path to the CSV file.
`verbose`	Logical. If `TRUE`, displays column type messages from `readr::read_csv()`. 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` *Split a data frame by a grouping column and write each group to a CSV file*

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 underscores collapsed, and leading/trailing underscores 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",
```

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

Index