

Package: washr (via r-universe)

September 17, 2026

Title Publication Toolkit for Water, Sanitation and Hygiene (WASH)
Data

Version 1.1.1

Date 2026-09-09

Description A toolkit to set up an R data package in a consistent structure. Automates tasks like tidy data export, data dictionary documentation, README and website creation, and citation management.

License GPL (>= 3)

URL <https://openwashdata.github.io/washr/>

BugReports <https://github.com/openwashdata/washr/issues>

Imports cffr (>= 1.0.1), desc (>= 1.4.3), jsonlite, pkgdown (>= 2.0.7), rlang (>= 1.1.3), tools, usethis (>= 2.2.3), utils, yaml, lifecycle

Suggests brand.yml, devtools (>= 2.4.5), knitr, remotes, rmarkdown, testthat (>= 3.2.1), withr (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev git make libharfbuzz-dev libgit2-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libnode-dev libx11-dev

Repository <https://openwashdata.r-universe.dev>

Date/Publication 2026-09-17 13:28:33 UTC

RemoteUrl <https://github.com/openwashdata/washr>

RemoteRef HEAD

RemoteSha d20d2c7e4b329064e4faf0de9e68758b1538aa8a

Contents

setup_ci	2
setup_dictionary	3
setup_rawdata	3
setup_readme	4
setup_roxygen	5
setup_website	6
update_citation	7
update_description	8
update_metadata	9
use_brand	10
Index	12

setup_ci	<i>Set up the R CMD check workflow for the data package</i>
----------	---

Description

setup_ci() writes the GitHub Actions workflow that runs R CMD check on every push and pull request to main and dev, on macOS, Windows and three versions of R on Linux. The openwashdata review standard requires this workflow with the dev trigger, so a package scaffolded with washr passes that part of the review floor by construction.

The workflow lands in .github/workflows/R-CMD-check.yaml and .github is added to .Rbuildignore. When README.Rmd exists with badge markers, the matching R CMD check badge is added between them (the README template from setup_readme() carries it already). An existing workflow file is kept as it is, so the function is safe to run again.

Usage

```
setup_ci()
```

Value

The path of the workflow file, invisibly.

See Also

Next: [setup_rawdata\(\)](#). The badge appears in the README written by [setup_readme\(\)](#).

Other setup functions: [setup_dictionary\(\)](#), [setup_rawdata\(\)](#), [setup_roxygen\(\)](#)

Examples

```
## Not run:
setup_ci()

## End(Not run)
```

setup_dictionary	Create a dictionary file for tidy data sets
------------------	---

Description

setup_dictionary() generates a dictionary CSV file in the data/ directory. The dictionary file contains information on the tidy data sets such as directory, file names, variable names, variable types, and descriptions. If tidy data exists, the dictionary is populated with relevant information; otherwise, it creates an empty dictionary CSV file.

Usage

```
setup_dictionary()
```

Value

NULL. Error if raw data is not found or not in a package directory.

See Also

Before: [setup_rawdata\(\)](#). Next: [setup_roxygen\(\)](#) once the descriptions in data-raw/dictionary.csv are written.

Other setup functions: [setup_ci\(\)](#), [setup_rawdata\(\)](#), [setup_roxygen\(\)](#)

Examples

```
## Not run:
setup_rawdata()
# Go to data_processing.R, clean the raw data and export tidy data
setup_dictionary()

## End(Not run)
```

setup_rawdata	Create the data-raw directory with a data-processing.R template
---------------	---

Description

setup_rawdata() creates a directory for raw data and an example script named data_processing.R for importing, processing and exporting the tidy data. The template assumes that the dataset name is the same as the data package name.

Usage

```
setup_rawdata()
```

Value

NULL. This function will create a directory "data-raw" under the package directory.

See Also

Before: `setup_ci()`. Next: `setup_dictionary()` once the processing script has exported the tidy data with `usethis::use_data()`.

Other setup functions: `setup_ci()`, `setup_dictionary()`, `setup_roxygen()`

Examples

```
## Not run:
  setup_rawdata()

## End(Not run)
```

 setup_readme

Generate the README RMarkdown file

Description

`setup_readme()` uses the openwashdata README template to generate README files based on datasets retrieved from the `data/` directory. It helps in creating consistent and informative README documentation for your data packages.

The template documents the first data object in `data/` (alphabetically); add a section per further object by hand. It stops when `data/` holds no data object, because every data section needs one.

Usage

```
setup_readme(force = FALSE, has_example = FALSE)
```

Arguments

<code>force</code>	Logical. If FALSE (the default), the function stops when a README.Rmd already exists. Set to TRUE to overwrite the existing file.
<code>has_example</code>	Logical. Should the README include an Example section with a commented ggplot2 scaffold for a first plot of the data? Defaults to FALSE. Pairs with the <code>has_example</code> argument of <code>setup_website()</code> , which adds the matching article to the site.

Value

NULL. This function creates a README.Rmd under the package directory.

See Also

Before: [update_description\(\)](#). Next: [setup_website\(\)](#), which builds the site from README.md.

Other publishing functions: [setup_website\(\)](#), [use_brand\(\)](#)

Examples

```
## Not run:
# Generate the README file after setting up the dictionary
setup_dictionary()
# Complete and save the dictionary CSV file with variable descriptions
setup_readme()
# With an Example section to fill with a first plot
setup_readme(has_example = TRUE)

## End(Not run)
```

setup_roxygen	<i>Set up roxygen documentation for all tidy data sets using the dictionary</i>
---------------	---

Description

Creates or updates Roxygen documentation for all tidy data sets found in the dictionary file.

When first run, this function creates the Roxygen documentation with placeholders for the title and description field. The dictionary should include columns for directory, file name, variable name, variable type, and description. This function generates Roxygen comments with this information, facilitating consistent and thorough documentation for your data sets.

When re-run this function, this function updates only the variable description entries in the Roxygen documentation files within R/ directory. The title and description fields remain unchanged.

Usage

```
setup_roxygen()
```

Value

NULL. This function creates documentation files inside "R/". Error if tidy data cannot be found.

See Also

Before: [setup_dictionary\(\)](#). Next: [update_description\(\)](#).

Other setup functions: [setup_ci\(\)](#), [setup_dictionary\(\)](#), [setup_rawdata\(\)](#)

Examples

```
## Not run:
setup_dictionary()
# Once the dictionary is created, go to data-raw/dictionary.csv and complete the column description.
setup_roxygen()

## End(Not run)
```

setup_website

Set up a pkgdown website for the data package

Description

setup_website() writes the openwashdata pkgdown configuration and builds the site from the package documentation and README.md. The configuration comes from the washr template: the Pages URL as the site URL, the openwashdata analytics header, the funding sidebar, the authors footer, and a reference index with one entry per data object in data/.

The function is safe to re-run. An existing _pkgdown.yml is kept as it is and only the site is rebuilt, so hand edits and the brand wiring from use_brand() survive. The example article is created once.

By default the built docs/ folder is tracked in git, which is how openwashdata packages publish their site from main. When the package deploys through the pkgdown GitHub Actions workflow instead (.github/workflows/pkgdown.yml exists), docs/ stays ignored.

Usage

```
setup_website(has_example = FALSE, track_docs = NULL)
```

Arguments

has_example	Logical. Should the pkgdown website include a vignette page for writing an example? Defaults to FALSE.
track_docs	Logical. Remove docs from .gitignore so the built site is committed? Defaults to TRUE unless a pkgdown workflow exists under .github/workflows/.

Value

NULL. Error if no README file is found.

See Also

Before: setup_readme(). Next: use_brand() for the brand, and update_citation() once the release has a DOI.

Other publishing functions: setup_readme(), use_brand()

Examples

```
## Not run:
# Set up the pkgdown website including a vignette page
setup_website(has_example = TRUE)

## End(Not run)
```

update_citation	<i>Update the citation file for the dataset.</i>
-----------------	--

Description

Create a citation *.cff file for the dataset from a given DOI (Digital Object Identifier). When a DOI is supplied, it adds the DOI badge to the README RMarkdown file and re-builds the README.md and pkgdown website if they exist. Before a release exists, call it without arguments to generate the citation files without a DOI or badge.

Usage

```
update_citation(doi = NULL, build = TRUE, type = c("dataset", "software"))
```

Arguments

doi	DOI (Digital Object Identifier), e.g., 10.5281/zenodo.11185699.
build	Logical. Rebuild README.md and the pkgdown site after the citation files change? Defaults to TRUE. Set to FALSE to regenerate the citation files alone, e.g., in scripts and tests. Defaults to NULL for the pre-release call, in which case no DOI is recorded and no badge is added.
type	The CFF type of the work: "dataset" (the default, a data package) or "software". Before 1.1.1 the file always said software, the cffr default. Zenodo's GitHub integration ignores this field; the resource type of a deposit comes from a .zenodo.json (#56).

Value

NULL. A citation .cff file is written under the root directory.

See Also

Before: [setup_website\(\)](#). Run again with the DOI after the Zenodo release; [update_metadata\(\)](#) then picks the DOI up.

Other metadata functions: [update_description\(\)](#), [update_metadata\(\)](#)

Examples

```
## Not run:
  update_citation(doi = "10.5281/zenodo.11185699")
  # Regenerate the citation files without rebuilding README.md and the site
  update_citation(build = FALSE)

## End(Not run)
```

update_description	<i>Update the DESCRIPTION file to conform with openwashdata standards</i>
--------------------	---

Description

This function updates the DESCRIPTION file of an R package to comply with openwashdata standards. It ensures that fields such as License, Language, Date, URL, and others are correctly specified. Existing URL and Config/Needs/website entries are preserved and merged with the openwashdata defaults. A CC BY 4.0 license is only set when the package does not have a license yet; an existing license is left untouched.

Usage

```
update_description(
  file = ".",
  github_user = "https://github.com/openwashdata/"
)
```

Arguments

file	Character. The file path to the DESCRIPTION file of the R package. Defaults to the current working directory.
github_user	Character. The URL path to the GitHub user or organization that hosts the current package. Defaults to "https://github.com/openwashdata".

Value

NULL. Update fields directly in DESCRIPTION file.

See Also

Before: [setup_roxygen\(\)](#). Next: [update_metadata\(\)](#) for the schema.org metadata, then [setup_readme\(\)](#).
Other metadata functions: [update_citation\(\)](#), [update_metadata\(\)](#)

Examples

```
## Not run:
# Update DESCRIPTION file in the current package
update_description()

# Update DESCRIPTION file in a specific package
update_description(file = "path/to/your/package/DESCRIPTION")

# Update DESCRIPTION file with a specific GitHub user
update_description(github_user = "https://github.com/yourusername")

## End(Not run)
```

update_metadata	<i>Generate and embed the schema.org metadata of the data package</i>
-----------------	---

Description

[Experimental]

update_metadata() derives a schema.org Dataset description from the canonical sources of the package and writes it as a JSON-LD block into the head of every pkgdown page, where dataset search engines read it. Nothing is hand edited: to change a value, change its source and run the function again. Running it twice produces no change. It ends by listing the fields it could not fill and where to fill them.

The sources are:

Field	Source
name, description, version, datePublished, license	Title, Description, Version, Date, License in DESCRIPTION
url	the pkgdown site (a github.io entry in URL), else the repository
keywords	X-schema.org-keywords in DESCRIPTION, comma separated
spatialCoverage, temporalCoverage	X-schema.org-spatialCoverage and X-schema.org-temporalCovera
creator, maintainer, funder, publisher	Authors@R roles aut/cre, cre, fnd, cph; ORCID from the comment field
identifier, sameAs	the DOI in CITATION.cff, written by update_citation()
variableMeasured	data-raw/dictionary.csv
distribution	every file in inst/extdata that belongs to a dataset, one entry per file

The JSON-LD lands in pkgdown/templates/in-header.html, which pkgdown picks up on the next site build. The file also keeps the in_header includes from _pkgdown.yml working. It is not shipped in the package tarball.

Usage

```
update_metadata(quiet = FALSE)
```

Arguments

quiet Logical. Suppress the messages and the report of blank fields. Defaults to FALSE.

Value

The Dataset description as a list, invisibly. The "blank" attribute names the fields that could not be filled and says where to fill them.

See Also

Before: [update_description\(\)](#), which writes the DESCRIPTION fields this reads. Next: [setup_readme\(\)](#). The DOI comes from [update_citation\(\)](#).

Other metadata functions: [update_citation\(\)](#), [update_description\(\)](#)

Examples

```
## Not run:
update_metadata()

## End(Not run)
```

use_brand

Install or refresh the openwashdata brand in the active package

Description

use_brand() copies the openwashdata brand definition (_brand.yml) and the logo files it references from the central [openwashdata/brand](#) repository into the package root. Re-running the function refreshes an existing copy and reports which files changed, so consuming packages stay in sync with the central definition.

Brand values are never edited locally: change them in openwashdata/brand first, then refresh consumers with use_brand().

Usage

```
use_brand(ref = "main", pkgdown = TRUE, source = NULL)
```

Arguments

ref Character. Git reference (branch or tag) of openwashdata/brand to copy from. Defaults to "main".

pkgdown Logical. Should _pkgdown.yml be wired to use the brand via bslib? Defaults to TRUE.

source Character. Advanced: an alternative source for the brand files, either a local directory or a URL prefix. When NULL (the default), the raw GitHub content of openwashdata/brand at ref is used. Mainly useful for tests and offline work.

Details

With `pkgdown = TRUE` (the default), an existing `_pkgdown.yml` is pointed at the brand through `bslib` (`template.bslib.brand`), so the next `pkgdown::build_site()` renders the site with the brand fonts and colors. The wiring rewrites `_pkgdown.yml` through the `yaml` package, which does not preserve comments in that file. When no `_pkgdown.yml` exists, the wiring is skipped with a hint to run `setup_website()` first. Building the wired site requires the `brand.yml` package (`bslib` asks for it at build time); it is listed in Suggests and installed on demand.

Value

Invisibly, a character vector of the files written or updated (empty when everything was already current).

See Also

Before: `setup_website()`, which writes the `_pkgdown.yml` this wires. For branded PDF and Word reports that read the installed `_brand.yml`, the `openwashdata` Quarto extension `quarto-owd`.

Other publishing functions: `setup_readme()`, `setup_website()`

Examples

```
## Not run:
# Install the brand and wire the pkgdown site
use_brand()

# Refresh later, without touching _pkgdown.yml
use_brand(pkgdown = FALSE)

## End(Not run)
```

Index

* metadata functions

- update_citation, [7](#)
- update_description, [8](#)
- update_metadata, [9](#)

* publishing functions

- setup_readme, [4](#)
- setup_website, [6](#)
- use_brand, [10](#)

* setup functions

- setup_ci, [2](#)
- setup_dictionary, [3](#)
- setup_rawdata, [3](#)
- setup_roxygen, [5](#)

pkgdown::build_site(), [11](#)

setup_ci, [2](#)
setup_ci(), [3–5](#)
setup_dictionary, [3](#)
setup_dictionary(), [2, 4, 5](#)
setup_rawdata, [3](#)
setup_rawdata(), [2, 3, 5](#)
setup_readme, [4](#)
setup_readme(), [2, 6, 8, 10, 11](#)
setup_roxygen, [5](#)
setup_roxygen(), [2–4, 8](#)
setup_website, [6](#)
setup_website(), [4, 5, 7, 11](#)

update_citation, [7](#)
update_citation(), [6, 8–10](#)
update_description, [8](#)
update_description(), [5, 7, 10](#)
update_metadata, [9](#)
update_metadata(), [7, 8](#)
use_brand, [10](#)
use_brand(), [5, 6](#)