Package 'worcs'

January 10, 2025

Type Package

Title Workflow for Open Reproducible Code in Science

Version 0.1.17

Description Create reproducible and transparent research projects in 'R'. This package is based on the Workflow for Open Reproducible Code in Science (WORCS), a step-by-step procedure based on best practices for Open Science. It includes an 'RStudio' project template, several convenience functions, and all dependencies required to make your project reproducible and transparent. WORCS is explained in the tutorial paper by Van Lissa, Brandmaier, Brinkman, Lamprecht, Struiksma, & Vreede (2021). <doi:10.3233/DS-210031>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/cjvanlissa/worcs

RoxygenNote 7.3.2

Imports methods, rmarkdown, renv, prereg (>= 0.6.0), gert (>= 2.0.1), ranger, yaml, digest, rticles (>= 0.27), tinytex, credentials, usethis (>= 3.1.0), gh, xfun, cli, rlang

Suggests papaja (>= 0.1.1), targets, lavaan, tidySEM, tarchetypes, remotes, knitr, missRanger, curl, fs, withr, desc, purrr, testthat (>= 2.1.0)

SystemRequirements pandoc (>= 1.14) - https://pandoc.org

VignetteBuilder knitr

NeedsCompilation no

Author Caspar J. Van Lissa [aut, cre] (<https://orcid.org/0000-0002-0808-5024>), Aaron Peikert [aut] (<https://orcid.org/0000-0001-7813-818X>), Andreas M. Brandmaier [aut] (<https://orcid.org/0000-0001-8765-6982>)

Maintainer Caspar J. Van Lissa <c.j.vanlissa@tilburguniversity.edu>

Repository CRAN

Date/Publication 2025-01-10 15:50:02 UTC

Contents

add_endpoint *Add endpoint to WORCS project*

Description

Add a specific endpoint to the WORCS project file. Endpoints are files that are expected to be exactly reproducible (e.g., a manuscript, figure, table, et cetera). Reproducibility is checked by ensuring the endpoint's checksum is unchanged.

Usage

```
add_endpoint(filename = NULL, worcs_directory = ".", verbose = TRUE, ...)
```

Arguments

filename	Character, indicating the file to be tracked as endpoint. Default: NULL.
worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory. Default: '.'
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
...	Additional arguments.

Value

No return value. This function is called for its side effects.

See Also

snapshot_endpoints check_endpoints

Examples

```
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "add_endpoint")
dir.create(test_dir)
setwd(test_dir)
file.create(".worcs")
writeLines("test", "test.txt")
add_endpoint("test.txt")
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

add_license_file *Add License File to Project*

Description

This function wraps `usethis`' `licenses` functions, which are designed for R-packages. This function makes them applicable to other use cases (e.g., WORCS projects, FAIR theory).

Usage

```
add_license_file(path = ".", license = "ccby", ...)
```

Arguments

`path`	Character, indicating the directory in which to create the license file. Default: '.'.
`license`	Character, indicating which license function to call. The `usethis` functions all have the form `use_{licensename}_license()`. The `license` argument consists only of the `{licensename}`, e.g. `ccby`.
`...`	Additional arguments passed to `usethis` function.

Value

No return value. This function is called for its side effects.

Examples

```
tmpdr <- file.path(tempdir(), "license")
dir.create(tmpdr)
add_license_file(path = tmpdr,
                 license = "proprietary",
                 copyright_holder = "test")
unlink(tmpdr, recursive = TRUE)
```

add_manuscript *Add Rmarkdown manuscript*

Description

Adds an Rmarkdown manuscript to a 'worcs' project.

Usage

```
add_manuscript(
  worcs_directory = ".",
  manuscript = "APA6",
  remote_repo = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

worcs_directory	Character, indicating the directory in which to create the manuscript files. Default: '.', which points to the current working directory.
manuscript	Character, indicating what template to use for the 'R Markdown' manuscript. Default: 'APA6'. Available choices include: "APA6", "github_document", "None" and the templates from the rticles package. See Details.
remote_repo	Character, 'https' link to the remote repository for this project. This link should have the form https://[...].git. This link will be inserted in the draft manuscript.
verbose	Logical. Whether or not to print messages to the console during project creation. Default: TRUE
...	Additional arguments passed to and from functions.

Details

Available choices include the following manuscript templates:

'APA6' An APA6 style template from the papaja package

'github_document' A github_document from the rmarkdown package

'acm_article' acm style template from the rtices package

'acs_article' acs style template from the rtices package

'aea_article' aea style template from the rtices package

'agu_article' agu style template from the rtices package

'ajs_article' ajs style template from the rtices package

'amq_article' amq style template from the rtices package

'ams_article' ams style template from the rtices package

'arxiv_article' arxiv style template from the rtices package

'asa_article' asa style template from the rtices package

'bioinformatics_article' bioinformatics style template from the rtices package

'biometrics_article' biometrics style template from the rtices package

'copernicus_article' copernicus style template from the rtices package

'ctex_article' ctex style template from the rtices package

'elsevier_article' elsevier style template from the rtices package
'frontiers_article' frontiers style template from the rtices package
'glossa_article' glossa style template from the rtices package
'ieee_article' ieee style template from the rtices package
'ims_article' ims style template from the rtices package
'informs_article' informs style template from the rtices package
'iop_article' iop style template from the rtices package
'isba_article' isba style template from the rtices package
'jasa_article' jasa style template from the rtices package
'jedm_article' jedm style template from the rtices package
'joss_article' joss style template from the rtices package
'jss_article' jss style template from the rtices package
'lipics_article' lipics style template from the rtices package
'mdpi_article' mdpi style template from the rtices package
'mnras_article' mnras style template from the rtices package
'oup_article' oup style template from the rtices package
'peerj_article' peerj style template from the rtices package
'pihph_article' pihph style template from the rtices package
'plos_article' plos style template from the rtices package
'pnas_article' pnas style template from the rtices package
'rjournal_article' rjournal style template from the rtices package
'rsos_article' rsos style template from the rtices package
'rss_article' rss style template from the rtices package
'sage_article' sage style template from the rtices package
'sim_article' sim style template from the rtices package
'springer_article' springer style template from the rtices package
'tf_article' tf style template from the rtices package
'trb_article' trb style template from the rtices package
'wellcomeor_article' wellcomeor style template from the rtices package

Value

No return value. This function is called for its side effects.

Examples

```
the_test <- "worcs_manuscript"
old_wd <- getwd()
dir.create(file.path(tempdir(), the_test))
file.create(file.path(tempdir(), the_test, ".worcs"))
add_manuscript(file.path(tempdir(), the_test),
               manuscript = "None")
setwd(old_wd)
unlink(file.path(tempdir(), the_test))
```

add_preregistration *Add Rmarkdown preregistration*

Description

Adds an Rmarkdown preregistration template to a 'worcs' project.

Usage

```
add_preregistration(
  worcs_directory = ".",
  preregistration = "cos_prereg",
  verbose = TRUE,
  ...
)
```

Arguments

worcs_directory	Character, indicating the directory in which to create the manuscript files. Default: '.', which points to the current working directory.
preregistration	Character, indicating what template to use for the preregistration. Default: "cos_prereg"; use "None" to omit a preregistration. See Details for other available choices.
verbose	Logical. Whether or not to print messages to the console during project creation. Default: TRUE
...	Additional arguments passed to and from functions.

Details

Available choices include the templates from the prereg package, and several unique templates included with worcs:

'PSS' Preregistration and Sharing Software (Krypotos, Klugkist, Mertens, & Engelhard, 2019)

'Secondary' Preregistration for secondary analyses (Mertens & Krypotos, 2019)

'aspredicted_prereg' aspredicted template from the prereg package

'brandt_prereg' brandt template from the prereg package

'cos_prereg' cos template from the prereg package

'fmri_prereg' fmri template from the prereg package

'prp_quant_prereg' prp_quant template from the prereg package

'psyquant_prereg' psyquant template from the prereg package

'rr_prereg' rr template from the prereg package

'vantveer_prereg' vantveer template from the prereg package

Value

No return value. This function is called for its side effects.

Examples

```
the_test <- "worcs_prereg"
old_wd <- getwd()
dir.create(file.path(tempdir(), the_test))
file.create(file.path(tempdir(), the_test, ".worcs"))
add_preregistration(file.path(tempdir(), the_test),
                    preregistration = "cos_prereg")
setwd(old_wd)
unlink(file.path(tempdir(), the_test))
```

---

add_recipe *Add Recipe to Generate Endpoints*

Description

Add a recipe to a WORCS project file to generate its endpoints.

Usage

```
add_recipe(
  worcs_directory = ".",
  recipe = "rmarkdown::render('manuscript/manuscript.Rmd')",
  terminal = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory. Default: '.'
recipe	Character string, indicating the function call to evaluate in order to reproduce the endpoints of the WORCS project.
terminal	Logical, indicating whether or not to evaluate the `recipe` in the terminal (`TRUE`) or in R (`FALSE`). Defaults to `FALSE`
verbose	Logical. Whether or not to print status messages to the console. Default: `TRUE`
...	Additional arguments.

Value

No return value. This function is called for its side effects.

See Also

add_endpoint snapshot_endpoints check_endpoints

Examples

```
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "add_recipe")
dir.create(test_dir)
setwd(test_dir)
file.create(".worcs")
writeLines("test", "test.txt")
add_recipe()
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

---

add_synthetic *Add synthetic data to WORCS project*

Description

This function adds a user-specified synthetic data resource for public use to a WORCS project with closed data.

Usage

```
add_synthetic(
  data,
  synthetic_name = paste0("synthetic_", original_name),
  original_name,
  worcs_directory = ".",
  verbose = TRUE,
  ...
)
```

Arguments

data	A data.frame containing the synthetic data.
synthetic_name	Character, naming the file synthetic data should be written to. By default, prepends "synthetic_" to the original_name.
original_name	Character, naming an existing data resource in the WORCS project with which to associate the synthetic data object.
worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory.
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
...	Additional arguments passed to and from functions.

Value

Returns NULL invisibly. This function is called for its side effects.

See Also

open_data closed_data save_data

Examples

```
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "add_synthetic")
dir.create(test_dir)
setwd(test_dir)
worcs:::write_worcsfile(".worcs")
# Prepare data
df <- iris[1:3, ]
# Run closed_data without synthetic
closed_data(df, codebook = NULL, synthetic = FALSE)
# Manually add synthetic
add_synthetic(df, original_name = "df.csv")
# Remove original from file and environment
file.remove("df.csv")
rm(df)
# See that load_data() now loads the synthetic file
load_data()
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

---

add_targets *Add targets to WORCS Project*

Description

Add a computational pipeline to a worcs project using the targets and tarchetypes packages (which must be installed). See those packages for extensive documentation.

Usage

```
add_targets(worcs_directory = ".", verbose = TRUE, ...)
```

Arguments

worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory. Default: '.'
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
...	Arguments passed to targets::use_targets().

Value

No return value. This function is called for its side effects.

Examples

```
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "targets")
dir.create(test_dir)
setwd(test_dir)
file.create(".worcs")
add_targets()
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

---

check_endpoints *Check endpoints in WORCS project*

Description

Check that the checksums of all endpoints in a WORCS project match their snapshots.

Usage

```
check_endpoints(worcs_directory = ".", verbose = TRUE, ...)
```

Arguments

worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory. Default: '.'
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
...	Additional arguments.

Value

Returns a logical value (TRUE/FALSE) invisibly.

See Also

add_endpoint snapshot_endpoints

Examples

```
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "check_endpoint")
dir.create(test_dir)
setwd(test_dir)
file.create(".worcs")
writeLines("test", "test.txt")
add_endpoint("test.txt")
check_endpoints()
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

---

check_worcs *Evaluate project with respect to WORCS checklist*

Description

Evaluates whether a project meets the criteria of the WORCS checklist (see `worcs_checklist`).

Usage

```
check_worcs(path = ".", verbose = TRUE)
```

Arguments

`path`	Character. Path to a WORCS project folder (a project with a `.worcs` file). Default: '.' (path to current directory).
`verbose`	Logical. Whether or not to show status messages while evaluating the checklist. Default: `TRUE`.

Value

A `data.frame` with a description of the criteria, and a column with evaluations (`$pass`). For criteria that must be evaluated manually, `$pass` will be `FALSE`.

Examples

```
example_dir <- file.path(tempdir(), "badge")
dir.create(example_dir)
write("a", file.path(example_dir, ".worcs"))
check_worcs(path = example_dir)
```

`check_worcs_installation` *Check worcs dependencies*

Description

This function checks that all worcs dependencies are correctly installed, and suggests how to remedy any missing dependencies.

Usage

```
check_worcs_installation(what = "all")

check_dependencies(package = "worcs")

check_git()

check_github(pat = TRUE, ssh = FALSE)

check_ssh()

check_tinytext()

check_rmarkdown()

check_renv()
```

Arguments

`what`	Character vector indicating which dependencies to check. Default: `"all"`. All checks defined in the Usage section can be called, e.g. `check_git` can be called using the argument `what = "git"`.
`package`	Atomic character vector, indicating for which package to check the dependencies.
`pat`	Logical, whether to run tests for the existence and functioning of a GitHub Personal Access Token (PAT). This is the preferred method of authentication, so defaults to TRUE.
`ssh`	Logical, whether to run tests for the existence and functioning of an SSH key. This method of authentication is not recommended, so defaults to FALSE.

Value

Logical, indicating whether all checks passed or not.

Examples

```
check_worcs_installation("none")
```

cite_all *Comprehensive citation Knit function for 'RStudio'*

Description

This is a wrapper for render. First, this function parses the citations in the document, converting citations marked with double at sign, e.g.: @@reference2020, into normal citations, e.g.: @reference2020. Then, it renders the file.

Usage

```
cite_all(...)
```

Arguments

...	All arguments are passed to render.

Value

Returns NULL invisibly. This function is called for its side effect of rendering an 'R Markdown' file.

Examples

```
# NOTE: Do not use this function interactively, as in the example below.
# Only specify it as custom knit function in an 'R Markdown' file, like so:
# knit: worcs::cite_all

if (rmarkdown::pandoc_available("2.0")){
  file_name <- file.path(tempdir(), "citeall.Rmd")
  loc <- rmarkdown::draft(file_name,
                          template = "github_document",
                          package = "rmarkdown",
                          create_dir = FALSE,
                          edit = FALSE)
  write(c("", "Optional reference: @reference2020"),
        file = file_name, append = TRUE)
  cite_all(file_name)
}
```

cite_essential *Essential citations Knit function for 'RStudio'*

Description

This is a wrapper for render. First, this function parses the citations in the document, removing citations marked with double at sign, e.g.: @@reference2020. Then, it renders the file.

Usage

```
cite_essential(...)
```

Arguments

...	All arguments are passed to render.

Value

Returns NULL invisibly. This function is called for its side effect of rendering an 'R Markdown' file.

Examples

```
# NOTE: Do not use this function interactively, as in the example below.
# Only specify it as custom knit function in an R Markdown file, like so:
# knit: worcs::cite_all

if (rmarkdown::pandoc_available("2.0")){
  file_name <- tempfile("citeessential", fileext = ".Rmd")
  rmarkdown::draft(file_name,
                   template = "github_document",
                   package = "rmarkdown",
                   create_dir = FALSE,
                   edit = FALSE)
  write(c("", "Optional reference: @reference2020"),
        file = file_name, append = TRUE)
  cite_essential(file_name)
}
```

---

closed_data *Use closed data in WORCS project*

Description

This function saves a data.frame as a .csv file (using write.csv), stores a checksum in '.worcs', appends the .gitignore file to exclude filename, and saves a synthetic copy of data for public use. To generate these synthetic data, the function synthetic is used.

Usage

```
closed_data(
  data,
  filename = paste0(deparse(substitute(data)), ".csv"),
  codebook = paste0("codebook_", deparse(substitute(data)), ".Rmd"),
  value_labels = paste0("value_labels_", deparse(substitute(data)), ".yml"),
  worcs_directory = ".",
  synthetic = TRUE,
  save_expression = write.csv(x = data, file = filename, row.names = FALSE),
```

```
  load_expression = read.csv(file = filename, stringsAsFactors = TRUE),
  ...
)
```

Arguments

data	A data.frame to save.
filename	Character, naming the file data should be written to. By default, constructs a filename from the name of the object passed to data.
codebook	Character, naming the file the codebook should be written to. An 'R Markdown' codebook will be created and rendered to github_document ('markdown' for 'GitHub'). By default, constructs a filename from the name of the object passed to data, adding the word 'codebook'. Set this argument to NULL to avoid creating a codebook.
value_labels	Character, naming the file the value labels of factors and ordinal variables should be written to. By default, constructs a filename from the name of the object passed to data, adding the word 'value_labels'. Set this argument to NULL to avoid creating a file with value labels.
worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory.
synthetic	Logical, indicating whether or not to create a synthetic dataset using the synthetic function. Additional arguments for the call to synthetic can be passed through ....
save_expression	An R-expression used to save the data. Defaults to write.csv(x = data, file = filename, row.names = FALSE), which writes a comma-separated, spreadsheet-style file. The arguments data and filename are passed from open_data() to the expression defined in save_expression.
load_expression	An R-expression used to load the data from the file created by save_expression. Defaults to read.csv(file = filename, stringsAsFactors = TRUE). This expression is stored in the project's .worcs file, and invoked by load_data().
...	Additional arguments passed to and from functions.

Value

Returns NULL invisibly. This function is called for its side effects.

See Also

open_data closed_data save_data

Examples

```
old_wd <- getwd()
test_dir <- file.path(tempdir(), "data")
dir.create(test_dir)
```

```
setwd(test_dir)
worcs:::write_worcsfile(".worcs")
df <- iris[1:3, ]
closed_data(df, codebook = NULL)
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

`data_label` *Label factor variables using metadata*

Description

For each column of `x`, this function checks whether value labels exist in `value_labels`. If so, integer values are replaced with these value labels.

Usage

```
data_label(
  x,
  variables = names(x),
  value_labels = read_yaml(paste0("value_labels_", substitute(x), ".yml"))
)
```

Arguments

`x`	A `data.frame`.
`variables`	Column names of `x` to replace, Default: `names(x)`
`value_labels`	A list with value labels, typically read from metadata generated by `open_data` or `closed_data`. Default: `read_yaml(paste0("value_labels_", substitute(x), ".yml"))`

Value

A `data.frame`.

Examples

```
## Not run:
if(interactive()){
 labs <- list(x = list(class = "factor", `1` = "a", `2` = "b"))
 df <- data.frame(x = 1:2)
 data_label(df, value_labels = labs)
 }

## End(Not run)
```

data_unlabel *Drop value labels*

Description

Coerces `factor` and `ordered` variables to class `integer`.

Usage

```
data_unlabel(x, variables = names(x)[sapply(x, inherits, what = "factor")])
```

Arguments

`x`	A `data.frame`.
`variables`	Column names of `x` to coerce to integer.

Value

A `data.frame`.

Examples

```
## Not run:
if(interactive()){
 df <- data.frame(x = factor(c("a", "b")))
 data_unlabel(df)
 }

## End(Not run)
```

descriptives *Describe a dataset*

Description

Provide descriptive statistics for a dataset.

Usage

```
descriptives(x, ...)
```

Arguments

`x`	An object for which a method exists.
`...`	Additional arguments.

Value

A `data.frame` with descriptive statistics for `x`.

Examples

```
descriptives(iris)
```

---

export_project *Export project to .zip file*

Description

Export project to .zip file

Usage

```
export_project(zipfile = NULL, worcs_directory = ".", open_data = TRUE)
```

Arguments

`zipfile`	Character. Path to a `.zip` file that is to be created. The default argument `NULL` creates a `.zip` file in the directory one level above the 'worcs' project directory. By default, all files tracked by 'Git' are included in the `.zip` file, excluding 'data.csv' if `open_data = FALSE`.
`worcs_directory`	Character. Path to the WORCS project directory to export. Defaults to `"."`, which refers to the current working directory.
`open_data`	Logical. Whether or not to include the original data, 'data.csv', if this file exists. If `open_data = FALSE` and an open data file does exist, then it is excluded from the `.zip` file. If it does not yet exist, a synthetic data set is generated and added to the `.zip` file.

Value

Logical, indicating the success of the operation. This function is called for its side effect of creating a `.zip` file.

Examples

```
export_project(worcs_directory = tempdir())
```

github_action_check_endpoints *Set up GitHub Actions to Check Endpoints*

Description

Sets up a GitHub Action to perform continuous integration (CI) for a WORCS project. CI automatically evaluates `check_endpoints()` or `reproduce(check_endpoints = TRUE)`. at each push or pull request.

Usage

```
github_action_check_endpoints(worcs_directory = ".")

github_action_reproduce(worcs_directory = ".")
```

Arguments

`worcs_directory`
: Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory. Default: '.'

Value

No return value. This function is called for its side effects.

See Also

`use_github_action` `add_endpoint` `check_endpoints`

git_ignore *Modify .gitignore file*

Description

Arguments passed through `...` are added to the .gitignore file. Elements already present in the file are modified. When `ignore = TRUE`, the arguments are added to the .gitignore file, which will cause 'Git' to not track them.

When `ignore = FALSE`, the arguments are prepended with `!`, This works as a "double negation", and will cause 'Git' to track the files.

Usage

```
git_ignore(..., ignore = TRUE, repo = ".")
```

Arguments

...	Any number of character arguments, representing files to be added to the .gitignore file.
ignore	Logical. Whether or not 'Git' should ignore these files.
repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.

Value

No return value. This function is called for its side effects.

Examples

```
dir.create(".git")
git_ignore("ignorethis.file")
unlink(".git", recursive = TRUE)
file.remove(".gitignore")
```

---

git_remote_connect *Connect to Existing 'GitHub' Repository*

Description

Given that a 'GitHub' user is configured, with the appropriate permissions, this function connects to an existing repository.

Usage

```
git_remote_connect(repo, remote_repo)
```

Arguments

repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.
remote_repo	Character, indicating the name of a repository on your account.

Value

Invisibly returns a list with the following elements:

- repo_url: Character, URL of the connected repository
- repo_exists: Logical
- prior_commits: Logical

See Also

gh_whoami git_fetch, git_remote

Examples

```
## Not run:
git_remote_connect()

## End(Not run)
```

`git_remote_create` *Create a New 'GitHub' Repository*

Description

Given that a 'GitHub' user is configured, with the appropriate permissions, this function creates a new repository on your account.

Given that a 'GitHub' user is configured, with the appropriate permissions, this function pushes the current branch (if safe), then publishes a 'GitHub' Release of the repository indicated by `repo` to that user's account.

Usage

```
git_remote_create(name, private = TRUE)

git_release_publish(repo = ".", tag_name = NULL, release_name = NULL)
```

Arguments

`name`	Name of the repository to be created.
`private`	Whether or not the repository should be private, defaults to `FALSE`.
`repo`	The path to the 'Git' repository.
`tag_name`	Optional character string to specify the tag name. By default, this is set to `NULL` and `git_release_publish()` uses version numbers starting with `0.1.0` for both the `tag_name` and `release_name` arguments. Override this behavior, for example, to increment the major version number by specifying `0.2.0`.
`release_name`	Optional character string to specify the tag name. By default, this is set to `NULL` and `git_release_publish()` uses version numbers starting with `0.1.0` for both the `tag_name` and `release_name` arguments. Override this behavior, for example, to increment the major version number by specifying `0.2.0`.

Value

Invisibly returns a logical value, indicating whether the function was successful or not.

No return value. This function is called for its side effects.

Examples

```
git_remote_create()
## Not run:
git_release_publish()

## End(Not run)
```

`git_update` *Add, commit, and push changes.*

Description

This function is a wrapper for `git_add`, `git_commit`, and `git_push`. It adds all locally changed files to the staging area of the local 'Git' repository, then commits these changes (with an optional) `message`, and then pushes them to a remote repository. This is used for making a "cloud backup" of local changes. Do not use this function when working with privacy sensitive data, or any other file that should not be pushed to a remote repository. The `git_add` argument `force` is disabled by default, to avoid accidentally committing and pushing a file that is listed in `.gitignore`.

Usage

```
git_update(
  message = paste0("update ", Sys.time()),
  files = ".",
  repo = ".",
  author,
  committer,
  remote,
  refspec,
  password,
  ssh_key,
  mirror,
  force,
  verbose = TRUE
)
```

Arguments

`message`	a commit message
`files`	vector of paths relative to the git root directory. Use "." to stage all changed files.
`repo`	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.
`author`	A git_signature value, default is git_signature_default.
`committer`	A git_signature value, default is same as author
`remote`	name of a remote listed in git_remote_list()

refspec	string with mapping between remote and local refs
password	a string or a callback function to get passwords for authentication or password protected ssh keys. Defaults to askpass which checks getOption('askpass').
ssh_key	path or object containing your ssh private key. By default we look for keys in ssh-agent and credentials::ssh_key_info.
mirror	use the –mirror flag
force	use the –force flag
verbose	display some progress info while downloading

Value

No return value. This function is called for its side effects.

Examples

```
git_update()
```

---

git_user *Set global 'Git' credentials*

Description

This function is a wrapper for git_config_global_set. It sets two name/value pairs at once: name = "user.name" is set to the value of the name argument, and name = "user.email" is set to the value of the email argument.

Usage

```
git_user(name, email, overwrite = !has_git_user(), verbose = TRUE)
```

Arguments

name	Character. The user name you want to use with 'Git'.
email	Character. The email address you want to use with 'Git'.
overwrite	Logical. Whether or not to overwrite existing 'Git' credentials. Use this to prevent code from accidentally overwriting existing 'Git' credentials. The default value uses has_git_user to set overwrite to FALSE if user credentials already exist, and to TRUE if no user credentials exist.
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE

Value

No return value. This function is called for its side effects.

Examples

```
do.call(git_user, worcs:::get_user())
```

has_git_user *Check whether global 'Git' credentials exist*

Description

Check whether the values `user.name` and `user.email` exist exist for the current repository. Uses `git_signature_default`.

Usage

```
has_git_user(repo = ".")
```

Arguments

`repo`	The path to the git repository.

Value

Logical, indicating whether 'Git' global configuration settings could be retrieved, and contained the values `user.name` and `user.email`.

Examples

```
testdir <- file.path(tempdir(), "test_git_user")
dir.create(testdir)
gert::git_init(testdir)
has_git_user(testdir)
unlink(testdir, recursive = TRUE)
```

load_data *Load WORCS project data*

Description

Scans the WORCS project file for data that have been saved using `open_data` or `closed_data`, and loads these data into the global (working) environment. The function will load the original data if available on the current system. If only a synthetic dataset is available, this function loads the synthetic data. The name of the object containing the data is derived from the file name by removing the file extension, and, when applicable, the prefix `"synthetic_"`. Thus, both `"data.csv"` and `"synthetic_data.csv"` will be loaded into an object called `data`.

Usage

```
load_data(
  worcs_directory = ".",
  to_envir = TRUE,
  envir = parent.frame(1),
  verbose = TRUE,
  use_metadata = TRUE
)
```

Arguments

worcs_directory	Character, indicating the WORCS project directory from which to load data. The default value `"."` points to the current directory.
to_envir	Logical, indicating whether to load objects directly into the environment, or return a `list` containing the objects. The environment is designated by argument `envir`. Loading objects directly into the global environment is user-friendly, but has the risk of overwriting an existing object with the same name, as explained in `load`. The function `load_data` gives a warning when this happens.
envir	The environment where the data should be loaded. The default value `parent.frame(1)` refers to the global environment in an interactive session.
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
use_metadata	Logical. Whether or not to use the codebook and value labels and attempt to coerce the class and values of variables to those recorded therein. Default: TRUE

Value

Returns a list invisibly. If `to_envir = TRUE`, this list contains the loaded data files. If `to_envir = FALSE`, the list is empty, and the loaded data files are attached directly to the global environment.

Examples

```
test_dir <- file.path(tempdir(), "loaddata")
old_wd <- getwd()
dir.create(test_dir)
setwd(test_dir)
worcs:::write_worcsfile(".worcs")
df <- iris[1:5, ]
suppressWarnings(closed_data(df, codebook = NULL))
load_data()
data
rm("data")
file.remove("data.csv")
load_data()
data
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

load_entrypoint *Load project entry points*

Description

Loads the designated project entry point into the default editor, using `file.edit`.

Usage

```
load_entrypoint(worcs_directory = ".", verbose = TRUE, ...)
```

Arguments

worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value `"."` points to the current directory.
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
...	Additional arguments passed to `file.edit`.

Value

No return value. This function is called for its side effects.

Examples

```
## Not run:
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "entrypoint")
dir.create(test_dir)
setwd(test_dir)
# Prepare worcs file and dummy entry point
worcs:::write_worcsfile(".worcs", entry_point = "test.txt")
writeLines("Hello world", con = file("test.txt", "w"))
# Demonstrate load_entrypoint()
load_entrypoint()
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)

## End(Not run)
```

make_codebook *Create codebook for a dataset*

Description

Creates a codebook for a dataset in 'R Markdown' format, and renders it to 'markdown' for 'GitHub'. A codebook contains metadata and documentation for a data file. We urge users to customize the automatically generated 'R Markdown' document and re-knit it, for example, to add a paragraph with details on the data collection procedures. The variable descriptives are stored in a `.csv` file, which can be edited in 'R' or a spreadsheet program. Columns can be appended, and we encourage users to complete at least the following two columns in this file:

category Describe the type of variable in this column. For example: "morality".

description Provide a plain-text description of the variable. For example, the full text of a questionnaire item: "People should be willing to do anything to help a member of their family".

Re-knitting the 'R Markdown' file (using `render`) will transfer these changes to the 'markdown' file for 'GitHub'.

Usage

```
make_codebook(
  data,
  filename = "codebook.Rmd",
  render_file = TRUE,
  csv_file = gsub("rmd$", "csv", filename, ignore.case = TRUE),
  verbose = TRUE
)
```

Arguments

`data`	A data.frame for which to create a codebook.
`filename`	Character. File name to write the codebook `rmarkdown` file to.
`render_file`	Logical. Whether or not to render the document.
`csv_file`	Character. File name to write the codebook `rmarkdown` file to. By default, uses the filename stem of the `filename` argument. Set to `NULL` to write the codebook only to the 'R Markdown' file, and not to `.csv`.
`verbose`	Logical. Whether or not to print status messages to the console. Default: TRUE

Value

`Logical`, indicating whether or not the operation was successful. This function is mostly called for its side effect of rendering an 'R Markdown' codebook.

Examples

```
if(rmarkdown::pandoc_available("2.0")){
  library(rmarkdown)
  library(knitr)
  filename <- tempfile("codebook", fileext = ".Rmd")
  make_codebook(iris, filename = filename, csv_file = NULL)
  unlink(c(
    ".worcs",
    filename,
    gsub("\\.Rmd", "\\.md", filename),
    gsub("\\.Rmd", "\\.html", filename),
    gsub("\\.Rmd", "_files", filename)
  ), recursive = TRUE)
}
```

---

notify_synthetic *Notify the user when synthetic data are being used*

Description

This function prints a notification message when some or all of the data used in a project are synthetic (see `closed_data` and `synthetic`). See details for important information.

Usage

```
notify_synthetic(..., msg = NULL)
```

Arguments

`...`	Objects of class `worcs_data`. The function will check if these are original or synthetic data.
`msg`	Expression containing the message to print in case not all `worcs_data` are original. This message may refer to `is_synth`, a logical vector indicating which `worcs_data` objects are synthetic.

Details

The preferred way to use this function is to provide specific data objects in the function call, using the `...` argument. If no such objects are provided, `notify_synthetic` will scan the parent environment for objects of class `worcs_data`.

This function is emphatically designed to be included in an 'R Markdown' file, to dynamically generate a notification message when a third party 'Knits' such a document without having access to all original data.

Value

No return value. This function is called for its side effect of printing a notification message.

See Also

closed_data synthetic add_synthetic

Examples

```
df <- iris
class(df) <- c("worcs_data", class(df))
attr(df, "type") <- "synthetic"
notify_synthetic(df, msg = "synthetic")
```

`open_data` *Use open data in WORCS project*

Description

This function saves a data.frame as a `.csv` file (using `write.csv`), stores a checksum in '.worcs', and amends the `.gitignore` file to exclude `filename`.

Usage

```
open_data(
  data,
  filename = paste0(deparse(substitute(data)), ".csv"),
  codebook = paste0("codebook_", deparse(substitute(data)), ".Rmd"),
  value_labels = paste0("value_labels_", deparse(substitute(data)), ".yml"),
  worcs_directory = ".",
  save_expression = write.csv(x = data, file = filename, row.names = FALSE),
  load_expression = read.csv(file = filename, stringsAsFactors = TRUE),
  ...
)
```

Arguments

`data`	A data.frame to save.
`filename`	Character, naming the file data should be written to. By default, constructs a filename from the name of the object passed to `data`.
`codebook`	Character, naming the file the codebook should be written to. An 'R Markdown' codebook will be created and rendered to `github_document` ('markdown' for 'GitHub'). By default, constructs a filename from the name of the object passed to `data`, adding the word 'codebook'. Set this argument to `NULL` to avoid creating a codebook.
`value_labels`	Character, naming the file the value labels of factors and ordinal variables should be written to. By default, constructs a filename from the name of the object passed to `data`, adding the word 'value_labels'. Set this argument to `NULL` to avoid creating a file with value labels.

`worcs_directory`
: Character, indicating the WORCS project directory to which to save data. The default value `"."` points to the current directory.

`save_expression`
: An R-expression used to save the `data`. Defaults to `write.csv(x = data, file = filename, row.names = FALSE)`, which writes a comma-separated, spreadsheet-style file. The arguments `data` and `filename` are passed from `open_data()` to the expression defined in `save_expression`.

`load_expression`
: An R-expression used to load the `data` from the file created by `save_expression`. Defaults to `read.csv(file = filename, stringsAsFactors = TRUE)`. This expression is stored in the project's `.worcs` file, and invoked by `load_data()`.

`...`
: Additional arguments passed to and from functions.

Value

Returns `NULL` invisibly. This function is called for its side effects.

See Also

open_data closed_data save_data

Examples

```
test_dir <- file.path(tempdir(), "data")
old_wd <- getwd()
dir.create(test_dir)
setwd(test_dir)
worcs:::write_worcsfile(".worcs")
df <- iris[1:5, ]
open_data(df, codebook = NULL)
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

---

reproduce *Reproduce WORCS Project*

Description

Evaluate the recipe contained in a WORCS project to derive its endpoints.

Usage

```
reproduce(worcs_directory = ".", verbose = TRUE, check_endpoints = TRUE, ...)
```

Arguments

worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory. Default: '.'
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
check_endpoints	Logical. Whether or not to call check_endpoints() after reproducing the recipe. Default: TRUE
...	Additional arguments.

Value

No return value. This function is called for its side effects.

See Also

add_endpoint snapshot_endpoints check_endpoints

Examples

```
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "reproduce")
dir.create(test_dir)
setwd(test_dir)
file.create(".worcs")
worcs:::add_recipe(recipe = 'writeLines("test", "test.txt")')
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

---

skew_kurtosis *Calculate skew and kurtosis*

Description

Calculate skew and kurtosis, standard errors for both, and the estimates divided by two times the standard error. If this latter quantity exceeds an absolute value of 1, the skew/kurtosis is significant. With very large sample sizes, significant skew/kurtosis is common.

Usage

```
skew_kurtosis(x, verbose = FALSE, se = FALSE, ...)
```

Arguments

x	An object for which a method exists.
verbose	Logical. Whether or not to print messages to the console, Default: FALSE
se	Whether or not to return the standard errors, Default: FALSE
...	Additional arguments to pass to and from functions.

Value

A matrix of skew and kurtosis statistics for x.

Examples

```
skew_kurtosis(datasets::anscombe)
```

---

snapshot_endpoints *Snapshot endpoints in WORCS project*

Description

Update the checksums of all endpoints in a WORCS project.

Usage

```
snapshot_endpoints(worcs_directory = ".", verbose = TRUE, ...)
```

Arguments

worcs_directory	Character, indicating the WORCS project directory to which to save data. The default value "." points to the current directory. Default: '.'
verbose	Logical. Whether or not to print status messages to the console. Default: TRUE
...	Additional arguments.

Value

No return value. This function is called for its side effects.

See Also

add_endpoint check_endpoints

Examples

```
# Create directory to run the example
old_wd <- getwd()
test_dir <- file.path(tempdir(), "update_endpoint")
dir.create(test_dir)
setwd(test_dir)
file.create(".worcs")
writeLines("test", "test.txt")
add_endpoint("test.txt")
writeLines("second test", "test.txt")
snapshot_endpoints()
# Cleaning example directory
setwd(old_wd)
unlink(test_dir, recursive = TRUE)
```

---

synthetic *Generate synthetic data*

Description

Generates a synthetic version of a `data.frame`, with similar characteristics to the original. See Details for the algorithm used.

Usage

```
synthetic(
  data,
  model_expression = ranger(x = x, y = y),
  predict_expression = predict(model, data = xsynth)$predictions,
  missingness_expression = NULL,
  verbose = TRUE
)
```

Arguments

`data`	A data.frame of which to make a synthetic version.
`model_expression`	An R-expression to estimate a model. Defaults to `ranger(x = x, y = y)`, which uses the fast implementation of random forests in `ranger`. The expression is evaluated in an environment containing objects `x` and `y`, where `x` is a `data.frame` with the predictor variables, and `y` is a `vector` of outcome values (see Details).
`predict_expression`	An R-expression to generate predicted values based on the model estimated by `model_expression`. Defaults to `predict(model, data = xsynth)$predictions`. This expression must return a vector of predicted values. The expression is evaluated in an environment containing objects `model` and `xsynth`, where `model` is the model estimated by `model_expression`, and `xsynth` is the `data.frame` of synthetic data used to predict the next column (see Details).

missingness_expression
: Optional. An R-expression to impute missing values. Defaults to NULL, which means listwise deletion is used. The expression is evaluated in an environment containing the object data, as specified in the call to synthetic. It must return a data.frame with the same dimensions and column names as the original data. For example, use missingness_expression = missRanger::missRanger(data = data) for a fast implementation of the excellent 'missForest' single imputation technique.

verbose
: Logical, Default: TRUE. Whether to show a progress bar while running the algorithm and provide informative messages.

Details

Based on the work by Nowok, Raab, and Dibben (2016), this function uses a simple algorithm to generate a synthetic dataset with similar characteristics to the original. The algorithm is as follows:

1. Let x be the original data.frame, with columns 1:j
2. Let xsynth be a synthetic data.frame, with columns 1:j
3. Column 1 of xsynth is a bootstrapped version of column 1 of x
4. Using model_expression, a predictive model is built for column c, for c along 2:j, with c predicted from columns 1:(c-1) of the original data.
5. Using predict_expression, columns 1:(c-1) of the synthetic data are used to predict synthetic values for column c.

Variables are thus imputed in order of occurrence in the data.frame. To impute in a different order, reorder the data.

Note that, for data synthesis to work properly, it is essential that the class of variables is defined correctly. The default algorithm ranger supports numeric, integer, and factor types. Other types of variables should be converted to one of these types, or users can use a custom model_expression and predict_expressio when calling synthetic.

Note that for data synthesis to work properly, it is essential that the class of variables is defined correctly. The default algorithm ranger supports numeric, integer, factor, and logical data. Other types of variables should be converted to one of these types.

Users can provide use a custom model_expression and predict_expression to use a different algorithm when calling synthetic.

As demonstrated in the example, users could call lm as a model_expression to use linear regression, which preserves linear marginal relationships but can give rise to values out of range of the original data. Or users could call sample as a predict_expression to bootstrap each variable, a very quick solution that maintains univariate distributions but loses all marginal relationships. These examples are not exhaustive, and users can even create custom functions.

Value

A data.frame with synthetic data, based on data.

References

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. Journal of Statistical Software, 74(11), 1-26. doi:10.18637/jss.v074.i11.

Examples

```
## Not run:
# Example using the iris dataset and default ranger algorithm
iris_syn <- synthetic(iris)

# Example using lm as prediction algorithm (only works for numeric variables)
# note that, within the model_expression, a new data.frame is created because
# lm() requires a separate data argument:
dat <- iris[, 1:4]
synthetic(dat,
          model_expression = lm(.outcome ~ .,
                                data = data.frame(.outcome = y,
                                xsynth)),
          predict_expression = predict(model, newdata = xsynth))

## End(Not run)
# Example using bootstrapping:
synthetic(iris,
          model_expression = NULL,
          predict_expression = sample(y, size = length(y), replace = TRUE))
## Not run:
# Example with missing data, no imputation
iris_missings <- iris
for(i in 1:10){
  iris_missings[sample.int(nrow(iris_missings), 1, replace = TRUE),
                sample.int(ncol(iris_missings), 1, replace = TRUE)] <- NA
}
iris_miss_syn <- synthetic(iris_missings)

# Example with missing data, imputation by median/mode substitution
# First, define a simple function for median/mode substitution:
imp_fun <- function(x){
  if(is.data.frame(x)){
    return(data.frame(sapply(x, imp_fun)))
  } else {
    out <- x
    if(inherits(x, "numeric")){
      out[is.na(out)] <- median(x[!is.na(out)])
    } else {
      out[is.na(out)] <- names(sort(table(out), decreasing = TRUE))[1]
    }
    out
  }
}

# Then, call synthetic() with this function as missingness_expression:
iris_miss_syn <- synthetic(iris_missings,
```

```
                          missingness_expression = imp_fun(data))

## End(Not run)
```

---

worcs_badge *Add WORCS badge to README.md*

Description

Evaluates whether a project meets the criteria of the WORCS checklist (see worcs_checklist), and adds a badge to the project's README.md.

Usage

```
worcs_badge(
  path = ".",
  update_readme = "README.md",
  update_csv = "checklist.csv"
)
```

Arguments

path	Character. This can either be the path to a WORCS project folder (a project with a .worcs file), or the path to a checklist.csv file. The latter is useful if you want to evaluate a manually updated checklist file. Default: '.' (path to current directory).
update_readme	Character. Path to the README.md file to add the badge to. Default: 'README.md'. Set to NULL to avoid updating the README.md file.
update_csv	Character. Path to the README.md file to add the badge to. Default: 'checklist.csv'. Set to NULL to avoid updating the checklist.csv file.

Value

No return value. This function is called for its side effects.

Examples

```
example_dir <- file.path(tempdir(), "badge")
dir.create(example_dir)
write("a", file.path(example_dir, ".worcs"))
worcs_badge(path = example_dir,
update_readme = NULL)
```

worcs_checklist *WORCS checklist*

Description

This checklist can be used to see whether a project adheres to the principles of open reproducible code in science, as set out in the WORCS paper.

Usage

```
data(worcs_checklist)
```

Format

A data frame with 15 rows and 5 variables.

Details

category	`factor`	Category of the checklist element.
name	`factor`	Name of the checklist element.
description	`factor`	What are the requirements to claim that this checklist element is met?
importance	`factor`	Whether the checklist element is essential to obtain a green 'open science' badge, or optional.
check	`logical`	Whether the criterion is checked automatically by `worcs_badge`.

References

Van Lissa, C. J., Brandmaier, A. M., Brinkman, L., Lamprecht, A., Peikert, A., , Struiksma, M. E., & Vreede, B. (2021) doi:10.3233/DS210031.

worcs_project *Create new WORCS project*

Description

Creates a new 'worcs' project. This function is invoked by the 'RStudio' project template manager, but can also be called directly to create a WORCS project through syntax or the console.

Usage

```
worcs_project(
  path = "worcs_project",
  manuscript = "APA6",
  preregistration = "cos_prereg",
  add_license = "CC_BY_4.0",
  use_renv = TRUE,
  use_targets = FALSE,
  remote_repo = "https",
  verbose = TRUE,
  ...
)
```

Arguments

path	Character, indicating the directory in which to create the 'worcs' project. Default: 'worcs_project'.
manuscript	Character, indicating what template to use for the 'R Markdown' manuscript. Default: 'APA6'. Available choices include `APA6` from the `papaja` package, a `github_document`, and templates included in the `rticles` package. For more information, see `add_manuscript`.
preregistration	Character, indicating what template to use for the preregistration. Default: 'cos_prereg'. Available choices include: `"PSS"`, `"Secondary"`, `"None"`, and all templates from the `prereg` package. For more information, see `add_preregistration`.
add_license	Character, indicating what license to include. Default: 'ccby'. Available options include: `c("cc0", "ccby", "gpl", "gpl3", "agpl", "agpl3", "apache", "apl2","lgpl", "mit", "proprietary", "None"`. For more information, see `use_cc0_license`.
use_renv	Logical, indicating whether or not to use 'renv' to make the project reproducible. Default: TRUE. See `init`.
use_targets	Logical, indicating whether or not to use 'targets' to create a Make-like pipeline. Default: FALSE See `targets-package`.
remote_repo	Character, URL of, or name for, the remote repository for this project. If a URL of an existing repository is specified, it should have the form `https://github.com[username][repo].g` (preferred) or `git@[...].git` (if using SSH). Alternatively, a name for a new repository can be provided. If a 'GitHub' user is authenticated on your device, this repository will be created on your account. Finally, a commit will be made containing the 'README.md' file, and will be pushed to the remote repository. Default: 'https', which results in no repository being created.
verbose	Logical. Whether or not to print messages to the console during project creation. Default: TRUE
...	Additional arguments passed to and from functions.

Value

No return value. This function is called for its side effects.

Examples

```
the_test <- "worcs_template"
old_wd <- getwd()
dir.create(file.path(tempdir(), the_test))
do.call(git_user, worcs:::get_user())
worcs_project(file.path(tempdir(), the_test, "worcs_project"),
              manuscript = "github_document",
              preregistration = "None",
              add_license = "None",
              use_renv = FALSE,
              remote_repo = "https")
setwd(old_wd)
unlink(file.path(tempdir(), the_test))
```

Index