

Package ‘syncdr’

April 14, 2026

Title Facilitate File Handling, Directory Comparison & Synchronization

Version 0.1.2

Description Compare directories flexibly (by date, content, or both) and synchronize files efficiently, with asymmetric and symmetric modes, helper tools, and visualization support for file management.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Imports collapse, data.table, digest, secretbase, cli, DT, fs, joyn
(>= 0.3.0), stats, knitr, utils, rstudioapi

Suggests fst, withr, rmarkdown, mockery, testthat (>= 3.0.0)

Depends R (>= 4.1.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://rossanatat.github.io/syncdr/>,
<https://github.com/RossanaTat/syncdr>

BugReports <https://github.com/RossanaTat/syncdr/issues>

NeedsCompilation no

Author R.Andres Castaneda [aut],
Rossana Tatulli [aut, cre],
Global Poverty and Inequality Data Team World Bank [cph]

Maintainer Rossana Tatulli <rtatulli@worldbank.org>

Repository CRAN

Date/Publication 2026-04-14 00:40:02 UTC

Contents

common_files_asym_sync_to_right	2
compare_directories	4
copy_temp_environment	5
display_dir_tree	6
display_sync_status	6
full_asym_sync_to_right	7
full_symmetric_sync	9
partial_symmetric_sync_common_files	11
partial_update_missing_files_asym_to_right	13
print_syncdr_status	14
save_sync_status	15
search_duplicates	15
toy_dirs	16
update_missing_files_asym_to_right	17

Index	19
--------------	-----------

common_files_asym_sync_to_right

Partial asymmetric synchronization to right (update common files)

Description

Partially synchronize right directory based on left one -i.e., the function will:

- for common_files:
 - if by date only: copy files that are newer in left to right
 - if by date and content: copy files that are newer and different in left to right
 - if by content only: copy files that are different in left to right
- for non common files, nothing changes: i.e.,
 - disregard those files that are only in left
 - keep in right those files that are only in right (i.e., files 'missing in left')

Usage

```
common_files_asym_sync_to_right(
  left_path = NULL,
  right_path = NULL,
  sync_status = NULL,
  by_date = TRUE,
  by_content = FALSE,
  recurse = TRUE,
  force = FALSE,
  backup = FALSE,
  backup_dir = "temp_dir",
```

```

    overwrite = TRUE,
    verbose = getOption("syncdr.verbose")
  )

```

Arguments

left_path	Path to the left/first directory.
right_path	Path to the right/second directory.
sync_status	Object of class "syncdr_status", output of compare_directories().
by_date	logical, TRUE by default
by_content	logical, FALSE by default
recurse	logical, TRUE by default. If recurse is TRUE: when copying a file from source folder to destination folder, the file will be copied into the corresponding (sub)directory. If the sub(directory) where the file is located does not exist in destination folder (or you are not sure), set recurse to FALSE, and the file will be copied at the top level
force	Logical. If FALSE (default), displays a preview of actions and prompts the user for confirmation before proceeding. Synchronization is aborted if the user does not agree. If TRUE, directly performs synchronization without prompting.
backup	Logical. If TRUE, creates a backup of the right directory before synchronization. The backup is stored in the location specified by backup_dir.
backup_dir	Path to the directory where the backup of the original right directory will be stored. If not specified, the backup is stored in temporary directory (tempdir).
overwrite	Logical. If TRUE (default), existing files at the destination are overwritten. If FALSE, existing destination files are preserved and the copy is skipped.
verbose	logical. If TRUE, display directory tree before and after synchronization. Default is FALSE

Value

Invisible TRUE indicating successful synchronization.

Examples

```

# Asymmetric synchronization of common files

e <- toy_dirs()
left <- e$left
right <- e$right
# Synchronize common files by content only
common_files_asym_sync_to_right(
  left_path = left,
  right_path = right,
  by_date = FALSE,
  by_content = TRUE
)

```

compare_directories *Compare Two Directories for Synchronization Status*

Description

This function compares two directories, typically referred to as 'left' and 'right', to determine their synchronization status at the file level. The primary goal is to identify the synchronization status of files present in both directories and those exclusive to either directory.

Usage

```
compare_directories(
  left_path,
  right_path,
  recurse = TRUE,
  by_date = TRUE,
  by_content = FALSE,
  verbose = getOption("syncdr.verbose")
)
```

Arguments

left_path	Path to the left/first directory.
right_path	Path to the right/second directory.
recurse	If TRUE, fully recurses through subdirectories. If a positive integer, specifies the number of levels to recurse.
by_date	Logical. If TRUE (default), compares directories based on the modification date of common files.
by_content	Logical. If TRUE, compares directories based on the hashed content of common files. Default is FALSE
verbose	Logical. If TRUE display additional info on the comparison process. Default is FALSE

Value

A list of class "syncdr_status" containing the following elements:

- Non-common files: Paths and synchronization status of files exclusive to either directory.
- Common files: Paths and synchronization status of files present in both directories.
- Path of the left directory.
- Path of the right directory.
- created_at: A POSIXct timestamp recording when the comparison was performed, used by sync functions to detect stale results.

Sync Status Types

The synchronization status is determined for files present in both directories, as well as for files exclusive to either directory. It can be computed based on modification date only, content only, or both.

For Common Files:

- When comparing by date: 'new', 'old', or 'same'.
- When comparing by date and content: 'new and different', 'new and same', 'old and different', 'old and same', 'same and different', or 'same and same'.
- When comparing by content only: 'different' or 'same'.

For Non-Common Files:

- When comparing by date (or by date and content, or by content only): 'only in left' or 'only in right'.

Examples

```
e <- toy_dirs()
left <- e$left
right <- e$right
compare_directories(left, right)
compare_directories(left, right, by_content = TRUE)
compare_directories(left, right, by_content = TRUE, by_date = FALSE)
```

copy_temp_environment *Create a temporary copy of .syncdrenv to test functions*

Description

This function creates a copy of the original environment, allowing tests to be executed without modifying the original environment.

Usage

```
copy_temp_environment()
```

Value

A list of temporary paths `left` and `right`.

display_dir_tree	<i>Display tree structure of one (or two) directory</i>
------------------	---

Description

Display tree structure of one (or two) directory

Usage

```
display_dir_tree(path_left = NULL, path_right = NULL, recurse = TRUE)
```

Arguments

path_left	path of left directory
path_right	path of right directory
recurse	logical, default to TRUE: show also sub-directories

Value

directories tree

Examples

```
# Create a temporary directory structure
```

```
e <- toy_dirs()
left <- e$left
right <- e$right
```

```
display_dir_tree(
  path_left = left,
  path_right = right
)
```

```
display_dir_tree(path_right = right)
```

display_sync_status	<i>Display status of synchronization/comparison info between two directories in DT table</i>
---------------------	--

Description

Display status of synchronization/comparison info between two directories in DT table

Usage

```
display_sync_status(sync_status_files, left_path, right_path)
```

Arguments

sync_status_files	object of <code>compare_directories()</code> output, either <code>common_files</code> or <code>non_common_files</code>
left_path	A character string specifying the path to left directory.
right_path	A character string specifying the path to right directory.

Value

DT table showing the comparison between the two directories together with their synchronization status

```
full_asym_sync_to_right
```

Full asymmetric synchronization to right directory

Description

This function performs a full asymmetric synchronization of the right directory based on the left directory. It includes the following synchronization steps (see Details below):

Usage

```
full_asym_sync_to_right(
  left_path = NULL,
  right_path = NULL,
  sync_status = NULL,
  by_date = TRUE,
  by_content = FALSE,
  recurse = TRUE,
  force = FALSE,
  delete_in_right = FALSE,
  backup = FALSE,
  backup_dir = "temp_dir",
  overwrite = TRUE,
  verbose = getOption("syncdr.verbose")
)
```

Arguments

left_path	Path to the left/first directory.
right_path	Path to the right/second directory.
sync_status	Object of class "syncdr_status", output of <code>compare_directories()</code> .

<code>by_date</code>	Logical. If TRUE, synchronize based on file modification dates (default is TRUE).
<code>by_content</code>	Logical. If TRUE, synchronize based on file contents (default is FALSE).
<code>recurse</code>	Logical. If TRUE (default), files are copied to corresponding subdirectories in the destination folder. If FALSE, files are copied to the top level of the destination folder without creating subdirectories if they do not exist.
<code>force</code>	Logical. If FALSE (default), displays a preview of actions and prompts the user for confirmation before proceeding. Synchronization is aborted if the user does not agree. If TRUE, directly performs synchronization without prompting.
<code>delete_in_right</code>	Logical. If FALSE (default), files that exist only in the right directory (i.e., absent from the left directory) are preserved during synchronization. If TRUE, those files are deleted from the right directory.
<code>backup</code>	Logical. If TRUE, creates a backup of the right directory before synchronization. The backup is stored in the location specified by <code>backup_dir</code> .
<code>backup_dir</code>	Path to the directory where the backup of the original right directory will be stored. If not specified, the backup is stored in temporary directory (<code>tempdir</code>).
<code>overwrite</code>	Logical. If TRUE (default), existing files at the destination are overwritten. If FALSE, existing destination files are preserved and the copy is skipped.
<code>verbose</code>	logical. If TRUE, display directory tree before and after synchronization. Default is FALSE

Details

- For common files:
 - If comparing by date only (`by_date = TRUE`): Copy files that are newer in the left directory to the right directory.
 - If comparing by date and content (`by_date = TRUE` and `by_content = TRUE`): Copy files that are newer and different in the left directory to the right directory.
 - If comparing by content only (`by_content = TRUE`): Copy files that are different in the left directory to the right directory.
- Copy to the right directory those files that exist only in the left directory.
- Delete from the right directory those files that are exclusive in the right directory (i.e., missing in the left directory)

Value

Invisible TRUE indicating successful synchronization.

Examples

```
e <- toy_dirs(fast = TRUE)
left <- e$left
right <- e$right
full_asym_sync_to_right(
  left_path = left,
```

```
    right_path = right,  
    by_date   = FALSE,  
    by_content = TRUE  
  )  
  sync_status <- compare_directories(left_path = left, right_path = right)  
  full_asym_sync_to_right(sync_status = sync_status)
```

full_symmetric_sync *Full symmetric synchronization*

Description

This function updates directories in the following way:

- For common files:
 - if by date: If the file in one directory is newer than the corresponding file in the other directory, it will be copied over to update the older version. If modification dates are the same, no action is taken
 - if by date and content: If the file in one directory is newer AND different than the corresponding file in the other directory, it will be copied over to update the older version. If modification dates/contents are the same, no action is taken
 - if by content only: this option is not active
- For non common files:
 - if a file exists in one but not in the other it is copied to the other directory

Usage

```
full_symmetric_sync(  
  left_path = NULL,  
  right_path = NULL,  
  sync_status = NULL,  
  by_date = TRUE,  
  by_content = FALSE,  
  recurse = TRUE,  
  force = FALSE,  
  backup = FALSE,  
  backup_dir = "temp_dir",  
  overwrite = TRUE,  
  verbose = getOption("syncdr.verbose")  
)
```

Arguments

left_path Path to the left/first directory.
right_path Path to the right/second directory.

<code>sync_status</code>	Object of class "syncdr_status", output of <code>compare_directories()</code> .
<code>by_date</code>	logical, TRUE by default
<code>by_content</code>	logical, FALSE by default
<code>recurse</code>	logical, TRUE by default. If <code>recurse</code> is TRUE: when copying a file from source folder to destination folder, the file will be copied into the corresponding (sub)directory. If the sub(directory) where the file is located does not exist in destination folder (or you are not sure), set <code>recurse</code> to FALSE, and the file will be copied at the top level
<code>force</code>	Logical. If FALSE (default), displays a preview of actions and prompts the user for confirmation before proceeding. Synchronization is aborted if the user does not agree. If TRUE, directly performs synchronization without prompting.
<code>backup</code>	Logical. If TRUE, creates a backup of both directories before synchronization. The backup is stored in the location specified by <code>backup_dir</code> .
<code>backup_dir</code>	Path to the directory where the backup will be stored. If not specified, the backup is stored in a temporary directory (<code>tempdir</code>).
<code>overwrite</code>	Logical. If TRUE (default), existing files at the destination are overwritten. If FALSE, existing destination files are preserved and the copy is skipped.
<code>verbose</code>	logical. If TRUE, display directory tree before and after synchronization. Default is FALSE

Value

Invisible TRUE indicating successful synchronization.

Examples

```
e <- toy_dirs()
left <- e$left
right <- e$right
# Option 1: provide left and right paths
full_symmetric_sync(
  left_path = left,
  right_path = right,
  by_date = TRUE,
  by_content = TRUE
)

# Option 2: provide a precomputed sync_status object
sync_status <- compare_directories(
  left_path = left,
  right_path = right
)
full_symmetric_sync(sync_status = sync_status)
```

 partial_symmetric_sync_common_files

Partial symmetric synchronization -common files only

Description

This function updates directories in the following way:

- For common files:
 - if by date: If the file in one directory is newer than the corresponding file in the other directory, it will be copied over to update the older version. If modification dates are the same, nothing is done
 - if by date and content: If the file in one directory is newer AND different than the corresponding file in the other directory, it will be copied over to update the older version. If modification dates/contents are the same, nothing is done
 - if by content only: this option is not active
- For non common files: unchanged, i.e.,
 - keep in right those that are only in right
 - keep in left those that are only in left

Usage

```
partial_symmetric_sync_common_files(
  left_path = NULL,
  right_path = NULL,
  sync_status = NULL,
  by_date = TRUE,
  by_content = FALSE,
  recurse = TRUE,
  force = FALSE,
  backup = FALSE,
  backup_dir = "temp_dir",
  overwrite = TRUE,
  verbose = getOption("syncdr.verbose")
)
```

Arguments

left_path	Path to the left/first directory.
right_path	Path to the right/second directory.
sync_status	Object of class "syncdr_status", output of compare_directories().
by_date	logical, TRUE by default
by_content	logical, FALSE by default

recurse	logical, TRUE by default. If recurse is TRUE: when copying a file from source folder to destination folder, the file will be copied into the corresponding (sub)directory. If the sub(directory) where the file is located does not exist in destination folder (or you are not sure), set recurse to FALSE, and the file will be copied at the top level
force	Logical. If FALSE (default), displays a preview of actions and prompts the user for confirmation before proceeding. Synchronization is aborted if the user does not agree. If TRUE, directly performs synchronization without prompting.
backup	Logical. If TRUE, creates a backup of the right directory before synchronization. The backup is stored in the location specified by backup_dir.
backup_dir	Path to the directory where the backup of the original right directory will be stored. If not specified, the backup is stored in temporary directory (tempdir).
overwrite	Logical. If TRUE (default), existing files in the destination are overwritten. If FALSE, existing files are not overwritten.
verbose	logical. If TRUE, display directory tree before and after synchronization. Default is FALSE

Value

Invisible TRUE indicating successful synchronization.

Examples

```
# Create a temporary synchronization environment

e <- toy_dirs()
left <- e$left
right <- e$right

# Partial symmetric synchronization of common files
# Option 1: provide left and right paths
partial_symmetric_sync_common_files(
  left_path = left,
  right_path = right,
  by_date   = TRUE
)

# Option 2: provide a precomputed sync_status object
sync_status <- compare_directories(
  left_path = left,
  right_path = right
)
partial_symmetric_sync_common_files(sync_status = sync_status)
```

partial_update_missing_files_asym_to_right

Partial asymmetric synchronization of non common files

Description

update non common files in right directory based on left one -i.e., the function will:

- for common_files:
 - do nothing, left unchanged
- for non common files,
 - copy those files that are only in left to right
 - keep in right those files that are only in right (i.e., files 'missing in left')

Usage

```
partial_update_missing_files_asym_to_right(
    left_path = NULL,
    right_path = NULL,
    sync_status = NULL,
    recurse = TRUE,
    force = FALSE,
    backup = FALSE,
    backup_dir = "temp_dir",
    overwrite = TRUE,
    verbose = getOption("syncdr.verbose")
)
```

Arguments

left_path	Path to the left/first directory.
right_path	Path to the right/second directory.
sync_status	Object of class "syncdr_status", output of compare_directories().
recurse	logical, TRUE by default. If recurse is TRUE: when copying a file from source folder to destination folder, the file will be copied into the corresponding (sub)directory. If the sub(directory) where the file is located does not exist in destination folder (or you are not sure), set recurse to FALSE, and the file will be copied at the top level
force	Logical. If FALSE (default), displays a preview of actions and prompts the user for confirmation before proceeding. Synchronization is aborted if the user does not agree. If TRUE, directly performs synchronization without prompting.
backup	Logical. If TRUE, creates a backup of the right directory before synchronization. The backup is stored in the location specified by backup_dir.
backup_dir	Path to the directory where the backup of the original right directory will be stored. If not specified, the backup is stored in temporary directory (tempdir).

overwrite	Logical. If TRUE (default), existing files at the destination are overwritten. If FALSE, existing destination files are preserved and the copy is skipped.
verbose	logical. If TRUE, display directory tree before and after synchronization. Default is FALSE

Value

Invisible TRUE indicating successful synchronization.

Examples

```
# Create a temporary synchronization environment

e <- toy_dirs()
left <- e$left
right <- e$right

# Partially update missing files asymmetrically (left → right)
# Option 1: provide left and right paths
partial_update_missing_files_asym_to_right(
  left_path = left,
  right_path = right
)

# Option 2: provide a precomputed sync_status object
sync_status <- compare_directories(
  left_path = left,
  right_path = right
)
partial_update_missing_files_asym_to_right(sync_status = sync_status)
```

`print.syncdr_status` *Print Synchronization Status*

Description

Print Synchronization Status

Usage

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

Arguments

x	object of syncdr_status class created in compare_directories
...	additional arguments

Value

prints syncdr_status object

save_sync_status	<i>Save sync_status file</i>
------------------	------------------------------

Description

Save sync_status file

Usage

```
save_sync_status(dir_path)
```

Arguments

dir_path path to directory

Value

the file is saved in a _syncdr subdirectory within the specified directory

Examples

```
# Set the directory path
e = toy_dirs()
left <- e$left

save_sync_status(dir_path = left)
```

search_duplicates	<i>Search for duplicate files in a directory</i>
-------------------	--

Description

This function searches for duplicate files within a directory based on their content. Duplicate files are identified by having either the same filename and same content or different filenames but same content.

Usage

```
search_duplicates(dir_path, verbose = TRUE)
```

Arguments

dir_path A character string representing the path to the directory to search for duplicates
 verbose Logical. If TRUE, displays a list of duplicate files found (default is TRUE)

Value

A data frame containing information about duplicate files (invisible by default)

Examples

```
# Search for duplicate files in a directory
```

```
e <- toy_dirs()
search_duplicates(dir_path = e$left)
```

toy_dirs	<i>Create toy directories to test syncdr functions</i>
----------	--

Description

create directories in syncdr environment. Directories are deleted when a new R session is started

Usage

```
toy_dirs(verbose = FALSE, fast = FALSE)
```

Arguments

verbose logical: display information. Default is FALSE
 fast logical: if TRUE (default), create a minimal set of files quickly; if FALSE, run full implementation with multiple files and timestamps.

Details

This function is a little slow because it must use `Sys.sleep()` to save files with the same name but different time stamp.

Value

syncdr environment with toy directory paths, i.e., left and right paths

Examples

```
# Create toy directories for testing / examples
```

```
toy_dirs(verbose = TRUE)
```

`update_missing_files_asym_to_right`*Full asymmetric synchronization of non common files*

Description

update non common files in right directory based on left one -i.e., the function will:

- for common_files:
 - do nothing, left unchanged
- for non common files,
 - copy those files that are only in left to right
 - delete in right those files that are only in right (i.e., files 'missing in left')

Usage

```
update_missing_files_asym_to_right(  
    left_path = NULL,  
    right_path = NULL,  
    sync_status = NULL,  
    recurse = TRUE,  
    force = FALSE,  
    backup = FALSE,  
    backup_dir = "temp_dir",  
    copy_to_right = TRUE,  
    delete_in_right = FALSE,  
    exclude_delete = NULL,  
    overwrite = TRUE,  
    verbose = getOption("syncdr.verbose")  
)
```

Arguments

<code>left_path</code>	Path to the left/first directory.
<code>right_path</code>	Path to the right/second directory.
<code>sync_status</code>	Object of class "syncdr_status", output of <code>compare_directories()</code> .
<code>recurse</code>	logical, TRUE by default. If recurse is TRUE: when copying a file from source folder to destination folder, the file will be copied into the corresponding (sub)directory. If the sub(directory) where the file is located does not exist in destination folder (or you are not sure), set recurse to FALSE, and the file will be copied at the top level
<code>force</code>	Logical. If FALSE (default), displays a preview of actions and prompts the user for confirmation before proceeding. Synchronization is aborted if the user does not agree. If TRUE, directly performs synchronization without prompting.

<code>backup</code>	Logical. If TRUE, creates a backup of the right directory before synchronization. The backup is stored in the location specified by <code>backup_dir</code> .
<code>backup_dir</code>	Path to the directory where the backup of the original right directory will be stored. If not specified, the backup is stored in temporary directory (<code>tempdir</code>).
<code>copy_to_right</code>	Logical, default is TRUE. If TRUE, files that exist only in the left directory are copied to the right directory. If FALSE, such files are not copied and remain absent from the right directory.
<code>delete_in_right</code>	Logical, default is FALSE. If TRUE, files that exist only in the right directory (i.e., not present in the left) are deleted. If FALSE, these right-only files are preserved.
<code>exclude_delete</code>	Character vector of basenames or folder name segments to protect from deletion. These are matched against the file basename and each path segment — full paths are not supported. Files matching any element are kept in the right directory even if <code>delete_in_right = TRUE</code> .
<code>overwrite</code>	Logical. If TRUE (default), existing files at the destination are overwritten. If FALSE, existing destination files are preserved and the copy is skipped.
<code>verbose</code>	logical. If TRUE, display directory tree before and after synchronization. Default is FALSE

Value

Invisible TRUE indicating successful synchronization.

Examples

```
# Create a temporary synchronization environment

e <- toy_dirs()
left <- e$left
right <- e$right

# Update missing files asymmetrically (left → right)
# Option 1: provide left and right paths
update_missing_files_asym_to_right(
  left_path = left,
  right_path = right
)

# Option 2: provide a precomputed sync_status object
sync_status <- compare_directories(
  left_path = left,
  right_path = right
)
update_missing_files_asym_to_right(sync_status = sync_status)
```

Index

`common_files_asym_sync_to_right`, [2](#)
`compare_directories`, [4](#), [14](#)
`copy_temp_environment`, [5](#)

`display_dir_tree`, [6](#)
`display_sync_status`, [6](#)

`full_asym_sync_to_right`, [7](#)
`full_symmetric_sync`, [9](#)

`partial_symmetric_sync_common_files`,
[11](#)
`partial_update_missing_files_asym_to_right`,
[13](#)
`print.syncdr_status`, [14](#)

`save_sync_status`, [15](#)
`search_duplicates`, [15](#)
`sys.sleep()`, [16](#)

`toy_dirs`, [16](#)

`update_missing_files_asym_to_right`, [17](#)