Package 'indexr'

February 17, 2025

Title A Thoughtful Saver of Results

Version 0.2.2

Date 2025-02-14

Description Helps with the thoughtful saving, reading, and management of result
files (using 'rds' files). The core functions take a list of parameters that are
used to generate a unique hash to save results under. Then, the same
parameter list can be used to read those results back in. This
is helpful to avoid clunky file naming when running a large number of
simulations. Additionally, helper functions are available for compiling a flat
file of parameters of saved results, monitoring result usage, and cleaning up
unwanted or unused results. For more information, visit the 'indexr' homepage
<https://lharris421.github.io/indexr/>.

BugReports https://github.com/lharris421/indexr/issues

License GPL-3

URL https://lharris421.github.io/indexr/,
https://github.com/lharris421/indexr

Encoding UTF-8

RoxygenNote 7.3.2

Imports stringr, readr, dplyr, digest, glue, methods

Suggests testthat (>= 3.0.0)

NeedsCompilation no

Author Logan Harris [aut, cre] (<https://orcid.org/0000-0002-1211-7623>)

Maintainer Logan Harris <logan-harris@uiowa.edu>

Repository CRAN

Date/Publication 2025-02-17 12:00:05 UTC

Contents

check_hash_existence   *Check for the Existence of Results Under a Set of Parameters*

Description

This function checks for the existence of results saved under specified parameter list in RDS files (saved with indexr) within a given folder.

Usage

```
check_hash_existence(
  folder,
  parameters_list,
  halt = FALSE,
  hash_includes_timestamp = FALSE,
  ignore_na = TRUE,
  alphabetical_order = TRUE,
  algo = "xxhash64",
  ignore_script_name = FALSE
)
```

Arguments

folder	A string specifying the directory containing the RDS files.
parameters_list	
	A list of parameters for which a corresponding hash named file is checked.
halt	Logical; if TRUE, the function stops execution if an existing file is found. This may be useful as a check before running a simulation.
hash_includes_timestamp	
	Logical; if TRUE, timestamps are included in the hash generation process.
ignore_na	Logical; if TRUE, NA values are ignored during hash generation.
alphabetical_order	
	Logical; if TRUE, parameters are sorted alphabetically before hash generation.
algo	Character string specifying the hashing algorithm to use. Default is "xxhash64". See ?digest
ignore_script_name	
	Logical. If TRUE, the script name is ignored during hash generation.

Value

A logical of whether or not a file exists, unless `halt` = TRUE and a file is found, then an error is thrown.

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

## Save an object
parameters_list <- list(example = "check_hash_existence")
save_objects(folder = tmp_dir, results = 1, parameters_list = parameters_list)

## Check that an object under specified parameters is saved
check_hash_existence(folder = tmp_dir, parameters_list)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

cleanup_from_hash_table

Remove Files Based on Hash Table

Description

Allows the user to leverage the `generate_hash` function to generate a table that is subsequently used to remove indicated results.

Usage

```
cleanup_from_hash_table(
  folder,
  hash_table,
  mode = c("manual", "all"),
  column = NULL,
  request_confirmation = TRUE
)
```

Arguments

`folder`	A string specifying the directory containing the RDS files.
`hash_table`	A `data.frame` from `create_hash_table`.
`mode`	A character string. When mode = "manual" (default) the function expects that the user will add a column to a hash table that indicated which files to delete. When mode = "all", any results in the hash table will be removed.

column   A character string indicating the logical column in `hash_table` specifying which
                           files to delete.

request_confirmation
                           Logical, if TRUE will request user input before proceeding to delete files.

Details

There are a few ways to use this. When `mode = "manual"` (default) the function expects that the
user will add a column to a hash table that indicated which files to delete. When `mode = "all"`,
any results in the hash table will be removed. This is generally only used when a `filter_list` is
passed to `create_hash_table`.

Value

Nothing, this function is called for its side effects.

See Also

[create_hash_table()](#)

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

## Save example objects
parameters_list1 <- list(example = "tagging1")
parameters_list2 <- list(example = "tagging2")
save_objects(folder = tmp_dir, results = 1, parameters_list = parameters_list1)
save_objects(folder = tmp_dir, results = 2, parameters_list = parameters_list2)

## See the files saved
list.files(tmp_dir)

## Create hash table (flat file of result parameters)
hash_table <- create_hash_table(folder = tmp_dir)

## Delete "all" files based on hash table, without confirmation
cleanup_from_hash_table(
  folder = tmp_dir, hash_table = hash_table, mode = "all", request_confirmation = FALSE
)

## See the files have been deleted
list.files(tmp_dir)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

compress_incremental *Combine Results Saved by* save_objects *with* incremental=TRUE

Description

This function is only intended to be used after save_objects with incremental=TRUE. In this case, save_objects with save results under temporary hashes in a folder with the hash corresponding the the parameters. compress_incremental then combines the results and saves them under the corresponding hash and deletes the old directory with the temporary results.

Usage

```
compress_incremental(
  folder,
  parameters_list,
  hash_includes_timestamp = FALSE,
  ignore_na = TRUE,
  alphabetical_order = TRUE,
  algo = "xxhash64",
  ignore_script_name = FALSE,
  remove_folder = TRUE
)
```

Arguments

folder   Character string specifying the path to the directory where the temporary folder
                  was saved (should be the same as supplied to save_objects.

parameters_list
                  The named list of arguments used with save_objects.

hash_includes_timestamp
                  Logical. If TRUE, the timestamp is included in the hash generation.

ignore_na   Logical. If TRUE, NA values in parameters_list are ignored during hash gen-
                  eration.

alphabetical_order
                  Logical. If TRUE, the names in parameters_list are sorted alphabetically be-
                  fore hash generation.

algo   Character string specifying the hashing algorithm to use. Default is "xxhash64".
                  See ?digest

ignore_script_name
                  Logical. If TRUE, the script name is ignored during hash generation.

remove_folder   Logical. If TRUE, the folder and the temporary results files will be discarded
                  after the combined results are saved.

Details

If the individual results can be put into a data.frame they will be, otherwise they will be stored as
a list.

Value

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

See Also

save_objects()

Examples

```
## Save results incrementally
params <- list(a = "1", b = "2")

tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)
for (i in 1:10) {
  save_objects(tmp_dir, data.frame(idx = i, val = rnorm(1)), params, incremental = TRUE)
}

## See contents of tmp directory for incremental file
list.files(file.path(tmp_dir, generate_hash(params)))

## Compress results into a single file
compress_incremental(tmp_dir, params)
list.files(tmp_dir)

## Read in compressed file and view results
read_objects(tmp_dir, params)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

create_hash_table *Create a Table of the Parameters for Saved Results from RDS Files*

Description

Reads in all the parameter files for a give folder, flattens nested lists, and then combines the parameters into a data frame. Each row in the resulting data frame represents the arguments used for one RDS file, identified by its hash. Optionally, the function can filter the data frame based on specified criteria and save it to a file.

Usage

```
create_hash_table(folder, save_path = NULL, filter_list = NULL)
```

Arguments

`folder`	A string specifying the directory containing the RDS files.
`save_path`	An optional string specifying the path to save the resulting hash table as a CSV file. If NULL, the hash table is not saved.
`filter_list`	An optional list of filters to apply to the hash table. Each element of the list should be named according to a column in the hash table and contain the value to filter for in that column.

Details

Saving the hash table can be helpful for the manipulation of parameters (see ?`update_hash_table`) or for removal of unwanted results (see ?`cleanup_from_hash_table`).

Value

A data frame where each row corresponds to an `parameters_list` from an RDS file, with an additional column for the hash of each set of arguments.

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

## Save objects
obj1 <- rnorm(1000)
obj2 <- data.frame(
  x = runif(100),
  y = "something",
  z = rep(c(TRUE, FALSE), 50)
)
obj3 <- list(obj1, obj2)

params1 <- list(
  distribution = "normal",
  other_params = list(param1 = TRUE, param2 = 1, param3 = NA)
)
params2 <- list(
  distribution = "uniform",
  other_params = list(param1 = FALSE, param2 = 2, param3 = "1", param4 = 4)
)
params3 <- list(
  distribution = "composite",
  other_params = list(param1 = TRUE, param2 = 3, param3 = 1)
)

save_objects(tmp_dir, obj1, params1)
save_objects(tmp_dir, obj2, params2)
save_objects(tmp_dir, obj3, params3)

## Create hash table (and save it)
```

```
create_hash_table(tmp_dir, save_path = file.path(tmp_dir, "hash_table.csv"))

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

generate_hash   *Generate a Consistent Hash for an Argument List*

Description

This function generates a hash value for a given list of arguments. It is designed to produce a consistent hash by optionally removing NA values, ordering arguments alphabetically, handling timestamp inclusion, etc.

Usage

```
generate_hash(
  parameters_list,
  hash_includes_timestamp = FALSE,
  ignore_na = TRUE,
  alphabetical_order = TRUE,
  algo = "xxhash64",
  ignore_script_name = FALSE
)
```

Arguments

parameters_list

> A named list of arguments for which the hash will be generated. Each element in the list should correspond to a parameter.

hash_includes_timestamp

> Logical; if FALSE, any timestamp included in parameters_list will be removed before hash generation. If TRUE, the timestamp will be included in the hash calculation.

ignore_na   Logical; if TRUE, any NA values in parameters_list will be removed before hash generation.

alphabetical_order

> Logical; if TRUE, the arguments in parameters_list will be sorted alphabetically by their names before hash generation.

algo   The hash algorithm to use (See ?digest)

ignore_script_name

> Logical. If TRUE, the script name is ignored during hash generation.

Value

A character string representing the hash value of the provided argument list.

Examples

```
args <- list(param1 = "value1", param2 = 100, param3 = NA)
generate_hash(args)
```

---

read_objects	*Read Objects Based on Parameter List*

Description

Reads R objects from specified folders based on a generated hash of the provided parameters_list.

Usage

```
read_objects(
  folders,
  parameters_list,
  hash_includes_timestamp = FALSE,
  ignore_script_name = FALSE,
  ignore_na = TRUE,
  alphabetical_order = TRUE,
  algo = "xxhash64",
  print_hash = FALSE,
  tagging_file_name = "indexr_tagging.txt",
  silent = FALSE
)
```

Arguments

folders	Character vector specifying the paths to directories containing the saved objects. The function will check each folder in order to find the file.
parameters_list	
	A named list of arguments used to generate a unique hash for the file.
hash_includes_timestamp	
	Logical. If TRUE, the timestamp is included in the hash generation.
ignore_script_name	
	Logical. If TRUE, the script name is ignored during hash generation.
ignore_na	Logical. If TRUE, NA values in parameters_list are ignored during hash generation.
alphabetical_order	
	Logical. If TRUE, the names in parameters_list are sorted alphabetically before hash generation.
algo	Character string specifying the hashing algorithm to use. Default is "xxhash64".
print_hash	Logical. If TRUE, prints the generated hash. This is helpful for debugging.
tagging_file_name	
	Character string of a txt file that is being used for tagging results. See ?start_tagging.

silent   Logical. If TRUE, no check is done that pairs of results files (parameters and as-
                  sociated results) is done. This check is not necessary, but done by default to keep
                  the user aware of a scenario that usually results from manual file manipulation.

Details

This function attempts to read an R object from files located in one of the specified folders. The
file name is based on the hash of the provided arguments. If the object is successfully read and a
tagging files exists and is specified, the function appends the hash and the current timestamp to the
tagging file in the folder where the file was found.

Value

The data stored in the file retrieved, typically the results. Returns NULL if the file is not found in any
of the specified folders.

See Also

save_objects()

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

## Example using parameter list to run simulation and save results
parameters_list <- list(
  iterations = 1000,
  x_dist = "rnorm",
  x_dist_options = list(n = 10, mean = 1, sd = 2),
  error_dist = "rnorm",
  error_dist_options = list(n = 10, mean = 0, sd = 1),
  beta0 = 1,
  beta1 = 1
)

betas <- numeric(parameters_list$iterations)
for (i in 1:parameters_list$iterations) {
  x <- do.call(parameters_list$x_dist, parameters_list$x_dist_options)
  err <- do.call(parameters_list$error_dist, parameters_list$error_dist_options)
  y <- parameters_list$beta0 + parameters_list$beta1*x + err
  betas[i] <- coef(lm(y ~ x))["x"]
}

save_objects(folder = tmp_dir, results = betas, parameters_list = parameters_list)

## Read back in (consider clearing environment before running)
## Re-setup
tmp_dir <- file.path(tempdir(), "example")

parameters_list <- list(
```

```
    iterations = 1000,
    x_dist = "rnorm",
    x_dist_options = list(n = 10, mean = 1, sd = 2),
    error_dist = "rnorm",
    error_dist_options = list(n = 10, mean = 0, sd = 1),
    beta0 = 1,
    beta1 = 1
)

betas <- read_objects(folder = tmp_dir, parameters_list = parameters_list)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

rehash   *Rehash RDS Files in a Directory*

Description

This function processes all RDS files in a specified directory, generating new hashes for each file's
args_list and renaming the files accordingly. It's useful when changing the hash generation algo-
rithm or parameters (if the parameters are manually changed for some reason).

Usage

```
rehash(
  folder,
  hash_includes_timestamp = FALSE,
  ignore_na = TRUE,
  alphabetical_order = TRUE,
  algo = "xxhash64"
)
```

Arguments

folder   A string specifying the directory containing the RDS files to be rehashed.

hash_includes_timestamp

                Logical; if TRUE, includes timestamps in the hash generation.

ignore_na   Logical; if TRUE, NA values are ignored during hash generation.

alphabetical_order

                Logical; if TRUE, parameters are sorted alphabetically before hash generation.

algo   The (potentially new) hash algorithm to use (see ?digest)

Value

The function does not return a value but renames the RDS files in the specified directory based on
new hashes.

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

# Save example objects
obj1 <- rnorm(1000)
obj2 <- data.frame(
  x = runif(100),
  y = "something",
  z = rep(c(TRUE, FALSE), 50)
)
obj3 <- list(obj1, obj2)

params1 <- list(
  distribution = "normal",
  other_params = list(param1 = TRUE, param2 = 1, param3 = NA)
)
params2 <- list(
  distribution = "uniform",
  other_params = list(param1 = FALSE, param2 = 2, param3 = "1", param4 = 4)
)
params3 <- list(
  distribution = "composite",
  other_params = list(param1 = TRUE, param2 = 3, param3 = 1)
)

save_objects(tmp_dir, obj1, params1)
save_objects(tmp_dir, obj2, params2)
save_objects(tmp_dir, obj3, params3)

## See current file names
list.files(tmp_dir)

## Rehash with new algo
rehash(tmp_dir, algo = "xxhash32")

## Observe new file names
list.files(tmp_dir)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

save_objects | *Save Simulation Results with Names as Hashes from the Parameters that Generated Them*

Description

Saves RDS files to a specified folder with a name that is a hash generated from a list of parameters used for the simulation. There are a number of options that control the behavior, however, the

default functionality likely covers 99% of use cases.

Usage

```
save_objects(
  folder,
  results,
  parameters_list = NULL,
  ignore_na = TRUE,
  alphabetical_order = TRUE,
  overwrite = FALSE,
  include_timestamp = TRUE,
  hash_includes_timestamp = FALSE,
  algo = "xxhash64",
  get_script_name = TRUE,
  ignore_script_name = FALSE,
  incremental = FALSE,
  silent = FALSE
)
```

Arguments

folder	Character string specifying the path to the directory where the objects will be saved.
results	The R object or list of objects to be saved.
parameters_list	
	A named list of arguments used to generate a unique hash for the file.
ignore_na	Logical. If TRUE, NA values in parameters_list are ignored during hash generation.
alphabetical_order	
	Logical. If TRUE, the names in parameters_list are sorted alphabetically before hash generation.
overwrite	Logical. If TRUE, existing files with the same hash will be overwritten. If FALSE and a conflict occurs, the results will be saved under a temporary hash.
include_timestamp	
	Logical. If TRUE, a timestamp is added to parameters_list.
hash_includes_timestamp	
	Logical. If TRUE, the timestamp is included in the hash generation.
algo	Character string specifying the hashing algorithm to use. Default is "xxhash64". See ?digest
get_script_name	
	Logical. If TRUE, attempts to get the script name and add it to parameters_list. Only works if script is run from command line, in an interactive session, this will always be NULL.
ignore_script_name	
	Logical. If TRUE, the script name is ignored during hash generation.

| incremental | Logical. If TRUE, results are saved in a subfolder named after the hash and can be combined with compress_incremental. Note, if TRUE, no checks will be done for results that already exist, the user should check this in their script with check_hash_existence. |
| silent | Logical. If TRUE, no check is done that pairs of results files (parameters and associated results) is done. This check is not necessary, but done by default to keep the user aware of a scenario that usually results from manual file manipulation. |

Details

This function saves R objects to disk with a file name based on a generated hash of the provided arguments. It supports incremental saving, where multiple results can be saved under the same hash in a subdirectory and later collected. This can be helpful for a simulation that runs and saves results in parallel for the SAME set of simulation parameters.

Value

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

See Also

[read_objects()](read_objects)

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

## Example using parameter list to run simulation and save results
parameters_list <- list(
  iterations = 1000,
  x_dist = "rnorm",
  x_dist_options = list(n = 10, mean = 1, sd = 2),
  error_dist = "rnorm",
  error_dist_options = list(n = 10, mean = 0, sd = 1),
  beta0 = 1,
  beta1 = 1
)

betas <- numeric(parameters_list$iterations)
for (i in 1:parameters_list$iterations) {
  x <- do.call(parameters_list$x_dist, parameters_list$x_dist_options)
  err <- do.call(parameters_list$error_dist, parameters_list$error_dist_options)
  y <- parameters_list$beta0 + parameters_list$beta1*x + err
  betas[i] <- coef(lm(y ~ x))["x"]
}

save_objects(folder = tmp_dir, results = betas, parameters_list = parameters_list)

## Read back in (consider clearing environment before running)
```

```
## Re-setup
tmp_dir <- file.path(tempdir(), "example")

parameters_list <- list(
  iterations = 1000,
  x_dist = "rnorm",
  x_dist_options = list(n = 10, mean = 1, sd = 2),
  error_dist = "rnorm",
  error_dist_options = list(n = 10, mean = 0, sd = 1),
  beta0 = 1,
  beta1 = 1
)

betas <- read_objects(folder = tmp_dir, parameters_list = parameters_list)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

start_tagging	*Monitor result file usage and cleanup unused files*

Description

Tagging is mainly helpful for removing unused results. start_tagging() initializes the tagging process by creating a txt file in the results directory which will keep a record of which results are being read by read_objects(). cleanup() removes any .rds files in the specified folder that are not listed in the tagging file. close_tagging() deletes the tagging file, ending the tagging session.

Usage

```
start_tagging(folder, tagging_file_name = "indexr_tagging.txt")

cleanup(
  folder,
  tagging_file_name = "indexr_tagging.txt",
  cutoff_date = NULL,
  request_confirmation = TRUE
)

close_tagging(folder, tagging_file_name = "indexr_tagging.txt")
```

Arguments

folder   A character string specifying the path to the directory where the result files are saved and where the tagging file will be created.

tagging_file_name
                A character string for a txt file the tagging information is to be saved under.

cutoff_date   A character string in "%Y-%m-%d %H:%M:%S" format used to specify that
               any tagged files before the date should also be removed.

request_confirmation

               Logical, if TRUE will request user input before proceeding to delete files.

Value

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

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

## Save example objects
parameters_list1 <- list(example = "tagging1")
parameters_list2 <- list(example = "tagging2")
save_objects(folder = tmp_dir, results = 1, parameters_list = parameters_list1)
save_objects(folder = tmp_dir, results = 2, parameters_list = parameters_list2)

## See the files have been saved
list.files(tmp_dir)

## Start tagging
start_tagging(tmp_dir)

## Read back in one the first file, this causes this file to be tagged
res1 <- read_objects(folder = tmp_dir, parameters_list = parameters_list1)

## Remove untagged file without confirmation (that for parameters_list2)
cleanup(tmp_dir, request_confirmation = FALSE)

## See that one file was removed
list.files(tmp_dir)

## Close tagging (just removes tagging file)
close_tagging(tmp_dir)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

---

update_from_hash_table

Update File Names Based on New Parameters in Adjusted Hash Table

---

Description

This function updates names of existing results by re-hashing each set of parameters with potentially updated values based on adjustments made to a hash table (see ?create_hash_table) by user. It loads RDS files based on their existing hashes, compares to the corresponding entry in a hash table, generates new hashes where needed, and saves the files with the new hashes. The old files are deleted if their hashes differ from the new ones.

Usage

```
update_from_hash_table(
  hash_table,
  rds_folder,
  hash_includes_timestamp = FALSE,
  ignore_na = TRUE,
  alphabetical_order = TRUE,
  algo = "xxhash64"
)
```

Arguments

hash_table	A file path to a modified hash table generated by create_hash_table.
rds_folder	A string specifying the directory containing the RDS files associated with the hash table.
hash_includes_timestamp	
	Logical; if TRUE, timestamps are included in the hash generation.
ignore_na	Logical; if TRUE, NA values are ignored during hash generation.
alphabetical_order	
	Logical; if TRUE, parameters are sorted alphabetically before hash generation.
algo	Character string specifying the hashing algorithm to use. Default is "xxhash64". See ?digest

Value

The function does not return a value but saves updated RDS files and deletes old files as needed.

See Also

[create_hash_table()](create_hash_table)

Examples

```
## Setup
tmp_dir <- file.path(tempdir(), "example")
dir.create(tmp_dir)

## Save objects
obj1 <- rnorm(1000)
obj2 <- data.frame(
```

```
  x = runif(100),
  y = "something",
  z = rep(c(TRUE, FALSE), 50)
)
obj3 <- list(obj1, obj2)

params1 <- list(
  distribution = "normal",
  other_params = list(param1 = TRUE, param2 = 1, param3 = NA)
)
params2 <- list(
  distribution = "uniform",
  other_params = list(param1 = FALSE, param2 = 2, param3 = "1", param4 = 4)
)
params3 <- list(
  distribution = "composite",
  other_params = list(param1 = TRUE, param2 = 3, param3 = 1)
)

save_objects(tmp_dir, obj1, params1)
save_objects(tmp_dir, obj2, params2)
save_objects(tmp_dir, obj3, params3)

## Create hash table
create_hash_table(tmp_dir, save_path = file.path(tmp_dir, "hash_table.csv"))

## Read in hash table, make a change, and save
hash_table <- read.csv(file.path(tmp_dir, "hash_table.csv"))
hash_table$distribution <- "something different"
write.csv(hash_table, file.path(tmp_dir, "hash_table.csv"))

## See file names before change
list.files(tmp_dir)

update_from_hash_table(
  hash_table = file.path(tmp_dir, "hash_table.csv"),
  rds_folder = tmp_dir
)

## See difference to before running update_hash_table()
list.files(tmp_dir)

## Cleanup
unlink(tmp_dir, recursive = TRUE)
```

Index