Package: mrgsim.ds (via r-universe)

May 8, 2026

Type Package

Title 'Apache' 'Arrow' Dataset-Backed Simulation Outputs for 'mrgsolve'

Version 0.0.1.9000

Maintainer Kyle T Baron <kylebtwin@imap.cc>

Description Provides tools for creating and managing file streams in support of large simulation or other outputs.

License GPL (>=2)

Depends R (>= 4.1), mrgsolve

Imports arrow, rlang, dplyr, scales, glue, methods, fs, tibble, lobstr, digest

Suggests testthat, knitr, rmarkdown, withr, duckdb, dbplyr, DBI, lattice

Encoding UTF-8

URL https://github.com/p-emex/mrgsim.ds, https://p-emex.github.io/mrgsim.ds/, https://p-emex.r-universe.dev/mrgsim.ds

BugReports https://github.com/p-emex/mrgsim.ds/issues

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Config/pak/sysreqs cmake make libuv1-dev libssl-dev

Repository https://p-emex.r-universe.dev

Date/Publication 2026-05-08 19:43:08 UTC

RemoteUrl https://github.com/p-emex/mrgsim.ds

RemoteRef HEAD

RemoteSha 233639467be5b97f1a3cb7924a24d0c8efdf8560

Contents

`as_arrow_ds` *Coerce an mrgsimsds object to an arrow data set*

Description

Extracts the underlying arrow::Dataset from an mrgsimsds object, allowing you to work directly with the Arrow API or pass the dataset to other Arrow-aware tools. For a list, only `mrgsimsds` elements are retained and a single dataset spanning all their files is returned.

Usage

```
as_arrow_ds(x, ...)

## S3 method for class 'mrgsimsds'
as_arrow_ds(x, ...)
```

Arguments

x	an mrgsimsds object or a list of mrgsimsds objects.
...	not used.

Value

An 'Apache' 'Arrow' arrow::Dataset object.

Examples

```
mod <- house_ds(end = 5)

out <- mrgsim_ds(mod, events = ev(amt = 100))

as_arrow_ds(out)
```

---

as_arrow_table.mrgsimsds

Coerce an mrgsimsds object to an arrow table

Description

Coerce an mrgsimsds object to an arrow table

Usage

```
## S3 method for class 'mrgsimsds'
as_arrow_table(x, ..., schema = NULL)
```

Arguments

x	an mrgsimsds object.
...	passed to `arrow::as_arrow_table()`.
schema	passed to `arrow::as_arrow_table()`.

Value

An 'Apache' 'Arrow' arrow::Table of simulated data.

Examples

```
mod <- house_ds(end = 5)

out <- mrgsim_ds(mod, events = ev(amt = 100))

arrow::as_arrow_table(out)
```

as_duckdb_ds *Coerce an mrgsimsds object to a DuckDB table*

Description

The conversion is handled by `as_arrow_ds()`.

Usage

```
as_duckdb_ds(x, ...)
```

Arguments

x	an mrgsimsds object or a list of mrgsimsds objects.
...	passed to `as_arrow_ds()`.

Value

A `tbl` of the simulated data in DuckDB; see `arrow::to_duckdb()`.

See Also

`as_arrow_ds()`

Examples

```
mod <- house_ds(end = 5)

out <- mrgsim_ds(mod, events = ev(amt = 100))

if(requireNamespace("duckdb")) {
  as_duckdb_ds(out)
}
```

as_mrgsim_ds *Coerce an mrgsims object to 'Apache' 'Arrow'-backed mrgsimsds object*

Description

Converts the output of `mrgsolve::mrgsim()` to an `mrgsimsds` object by writing the simulation data to a parquet file in `tempdir()`. Files in `tempdir()` are auto-deleted on garbage collection by default. Use `move_ds()` or `save_ds()` to relocate files outside `tempdir()`, which automatically disables gc, or call `gc_ds()` to control gc directly.

Usage

```
as_mrgsim_ds(x, verbose = FALSE, gc = TRUE)
```

Arguments

x	an mrgsims object.
verbose	if TRUE, print progress information to the console.
gc	initial gc setting; if TRUE, a finalizer function will attempt to remove files once the object is out of scope. This value is not locked: move_ds() and save_ds() will automatically adjust gc based on whether the files remain under tempdir(). To lock the gc setting and prevent automatic adjustment, call gc_ds() after creation.

Value

An object with class mrgsimsds.

See Also

mrgsim_ds().

Examples

```
mod <- house_ds()

data <- ev_expand(amt = 100, ID = 1:10)

out <- mrgsolve::mrgsim(mod, data)

obj <- as_mrgsim_ds(out)

obj
```

---

as_tibble.mrgsimsds *Coerce an mrgsimsds object to a tbl*

Description

Coerce an mrgsimsds object to a tbl

Usage

```
## S3 method for class 'mrgsimsds'
as_tibble(x, ...)

## S3 method for class 'mrgsimsds'
collect(x, ...)

## S3 method for class 'mrgsimsds'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	an mrgsimsds object.
...	passed to dplyr::as_tibble() or dplyr::collect().
row.names	passed to base::as.data.frame().
optional	passed to base::as.data.frame().

Value

A tbl containing simulated data.

Examples

```
mod <- house_ds(end = 5)

out <- mrgsim_ds(mod, events = ev(amt = 100))

as.data.frame(out)

tibble::as_tibble(out)

dplyr::collect(out)
```

---

copy_ds *Copy an mrgsimsds object*

Description

Creates a new mrgsimsds object pointing to the same parquet files as x. By default the new object takes ownership of those files, which means the original object loses ownership and its files will not be deleted when it is garbage collected.

Usage

```
copy_ds(x, own = TRUE)
```

Arguments

x	an mrgsimsds object to copy.
own	logical; if TRUE the new object takes ownership of the files; if FALSE ownership is left unchanged.

Value

A new mrgsimsds object with the same files and fields as x, a fresh memory address, and pid set to the current process.

Examples

```
mod <- house_ds()

out <- mrgsim_ds(mod)

out2 <- copy_ds(out)

check_ownership(out)

check_ownership(out2)
```

---

current_location *Get the current location of mrgsimsds object files*

Description

Get the current location of mrgsimsds object files

Usage

```
current_location(x)
```

Arguments

x	an mrgsimsds object.

files_ds *Get names of backing files*

Description

Get names of backing files

Usage

```
files_ds(x)
```

Arguments

x	an mrgsimsds object.

Value

A character vector of absolute paths to the parquet files backing `x`.

See Also

`move_ds()`, `save_ds()`

gc_ds *Set garbage collection behavior for mrgsimsds objects*

Description

Controls whether the underlying parquet files are automatically deleted when the object is garbage collected (`value`) and whether a message is issued when that deletion occurs (`notify`). Set `value = FALSE` to protect files from cleanup; set back to `TRUE` to re-enable automatic deletion. The `notify` flag is intended for debugging only; the `mrgsim.ds.show.gc` option provides the same behavior package-wide.

Calling `gc_ds()` with `value` locks the gc setting: once a value is explicitly set, the package will never automatically change it when files are moved or written. A warning is issued if gc is locked to `TRUE` but files are moved outside of `tempdir()`, since those files would then be auto-deleted on garbage collection.

Usage

```
gc_ds(x, value = NULL, notify = NULL, ...)

## S3 method for class 'mrgsimsds'
gc_ds(x, value = NULL, notify = NULL, ...)

## S3 method for class 'list'
gc_ds(x, value = NULL, notify = NULL, ...)
```

Arguments

x	an mrgsimsds object or a list of objects.
value	logical; if TRUE the underlying files will be deleted on garbage collection. Passing any value also locks the gc setting so that subsequent file operations (see move_ds()) do not automatically adjust it.
notify	logical; if TRUE a message will be issued when files are deleted on garbage collection. For debugging only; see also the mrgsim.ds.show.gc option.
...	not used.

Value

When x is an mrgsimsds object, it is returned invisibly with gc and/or gc_notify updated.

When x is a list, it is returned invisibly with gc_ds() applied to every mrgsimsds element; non-mrgsimsds elements are left unchanged.

Examples

```
mod <- modlib_ds("popex", outvars = "IPRED")

data <- ev_expand(amt = 100, ID = 1:5)

out <- mrgsim_ds(mod, data)

out <- gc_ds(out, value = FALSE)

out <- gc_ds(out, value = TRUE)

out <- lapply(1:3, function(rep) {
  out <- mrgsim_ds(mod, data)
  out
})

out <- gc_ds(out, value = FALSE)
```

---

is_mrgsimsds *Check if object inherits mrgsimsds*

Description

Check if object inherits mrgsimsds

Usage

```
is_mrgsimsds(x)
```

Arguments

x	object to check.

Value

TRUE if x inherits from mrgsimsds; FALSE otherwise.

Examples

```
mod <- house_ds()

out <- mrgsim_ds(mod, events = ev(amt = 100))

is_mrgsimsds(out)

is_mrgsimsds(list())
```

---

list_temp *Manage simulated outputs in the per-session temporary directory*

Description

Functions for inspecting and cleaning up package-managed parquet files in tempdir(). list_temp() shows what is present; purge_temp() resets the simulation file system.

purge_temp() deletes all package-managed files unconditionally and clears the ownership maps, resetting the system to a clean state. It is intended for use in testing teardown or session cleanup, not routine usage.

Usage

```
list_temp(quietly = FALSE)

purge_temp(quietly = FALSE)
```

Arguments

quietly	if TRUE, suppresses console output (the file listing for list_temp() and the deletion summary for purge_temp()).

Value

list_temp() returns a character vector of file paths invisibly, and prints a summary to the console unless quietly = TRUE.

purge_temp() returns NULL invisibly.

Examples

```
mod <- house_ds()

out <- lapply(1:10, \(x) mrgsim_ds(mod))

list_temp()

purge_temp()

list_temp()
```

`move_ds` *Move, rename, combine files in mrgsimsds objects*

Description

Use `move_ds()` to change the enclosing directory. `rename_ds()` keeps the files in place, but changes the file names. `combine_ds()` brings simulated data from multiple backing file into a single file.

Automatic gc adjustment:

Only `move_ds()` automatically updates the gc flag based on where the files end up: files that remain under `tempdir()` keep `gc = TRUE`; files moved outside `tempdir()` get `gc = FALSE`, protecting them from automatic deletion. Neither `rename_ds()` nor `combine_ds()` changes the gc flag because neither changes the file location.

This automatic adjustment is skipped if the gc setting has been locked by a prior call to `gc_ds()`. A warning is issued if gc is locked to `TRUE` but files land outside `tempdir()`.

The object (`x`) is required to own the underlying files in order to move, rename, or combine them.

All three functions modify `x` in place and file ownership stays with `x`.

Usage

```
move_ds(x, path, quietly = FALSE)

rename_ds(x, id)

combine_ds(x)
```

Arguments

`x`	an mrgsimsds object.
`path`	the new directory location for backing files.
`quietly`	if `FALSE`, a message is printed about the potentially new location of the backing files on move.
`id`	a short name used to create data set files for the simulated output.

Value

All three functions return `x` invisibly. The updated file list is accessible via `x$files`.

See Also

`save_ds()`, `files_ds()`, `gc_ds()`

Examples

```
mod <- house_ds()

out <- lapply(1:3, \(x) { mrgsim_ds(mod, events = ev(amt = 100)) })

out <- reduce_ds(out)

out <- rename_ds(out, "new-name")

out$files

out <- combine_ds(out)

out$files
```

`mread_ds` *Load an mrgsolve model for Arrow-backed simulation*

Description

Thin wrappers around mrgsolve model-loading functions (`mread()`, `mcode()`, `modlib()`, `house()`, `mread_cache()`) that additionally call `save_process_info()` to stamp the model with the current process ID. This stamp is required by `mrgsim_ds()` to correctly associate simulation outputs with the process that created them.

Usage

```
mread_ds(...)

mcode_ds(...)

modlib_ds(...)

house_ds(...)

mread_cache_ds(...)
```

Arguments

`...`	passed to the corresponding mrgsolve function.

Value

A model object with process information saved, suitable for use with `mrgsim_ds()`.

See Also

`save_process_info()`.

Examples

```
mod <- house_ds()

mod
```

`mrgsim.ds` *mrgsim.ds: 'Apache' 'Arrow' Dataset-Backed Simulation Outputs for 'mrgsolve'*

Description

`mrgsim.ds` provides an Apache Arrow-backed simulation output object for mrgsolve, greatly reducing the memory footprint of large simulations and providing a high-performance pipeline for summarizing huge simulation outputs. The arrow-based simulation output objects in R claim ownership of their files on disk. Those files are automatically removed when the owning object goes out of scope and becomes subject to the R garbage collector. While "anonymous", parquet-formatted files hold the data in `tempdir()` as you are working in R, functions are provided to move this data to more permanent locations for later use.

Package-wide options

- `mrgsim.ds.show.gc`: print messages to the console when object files are removed prior to object cleanup.

Function listing

- Load models
  - `mread_ds()`
  - `mcode_ds()`
  - `mread_cache_ds()`
  - `modlib_ds()`
  - `house_ds()`
- Generate Apache Arrow dataset-backed outputs
  - `mrgsim_ds()`
  - `as_mrgsim_ds()`
- S3 Methods

  - head.mrgsimsds()
  - tail.mrgsimsds()
  - dim.mrgsimsds()
  - names.mrgsimsds()
- Move, rename, or combine files
  - move_ds()
  - rename_ds()
  - combine_ds()
- Save and restore
  - save_ds()
  - read_ds()
  - write_parquet_ds()
  - write_dataset_ds()
- Ownership
  - ownership()
  - check_ownership()
  - list_ownership()
  - take_ownership()
  - disown()
  - copy_ds()
- Work with lists of outputs
  - reduce_ds()
  - refresh_ds()
  - prune_ds()
- Manage tempdir
  - list_temp()
  - purge_temp()
- Enter dplyr / arrow pipelines with
  - dplyr::mutate()
  - dplyr::select()
  - dplyr::filter()
  - dplyr::summarise()
  - dplyr::summarize()
  - dplyr::rename()
  - dplyr::arrange()
  - dplyr::group_by()
  - dplyr::distinct()
  - dplyr::relocate()
  - dplyr::count()
  - dplyr::pull()

- Coerce to R objects
  - as.data.frame()
  - dplyr::as_tibble()
  - dplyr::collect()
  - as_arrow_ds()
  - arrow::as_arrow_table()
  - as_duckdb_ds()

Author(s)

Maintainer: Kyle T Baron <kylebtwin@imap.cc> (ORCID) [copyright holder]

See Also

Useful links:

- https://github.com/p-emex/mrgsim.ds
- https://p-emex.github.io/mrgsim.ds/
- https://p-emex.r-universe.dev/mrgsim.ds
- Report bugs at https://github.com/p-emex/mrgsim.ds/issues

Examples

```
mod <- house_ds(end = 32)

data <- evd_expand(amt = seq(100, 300, 10))

out <- mrgsim_ds(mod, data)

out

head(out)

tail(out)

plot(out, nid = 10)

list_temp()

ownership()

## Not run:

rename_ds(out, "reg-100-300")

list_temp()

move_ds(out, "data/sim/regimens")

## End(Not run)
```

mrgsim_ds *Simulate from a model object, returning an 'Apache' 'Arrow'-backed output object*

Description

Runs mrgsolve::mrgsim() and writes simulation output to a parquet file in tempdir(), returning an mrgsimsds object. Files in tempdir() are auto-deleted on garbage collection by default. Use move_ds() or save_ds() to relocate files outside tempdir(), which automatically disables gc, or call gc_ds() to control gc directly. Note that full argument names must be used for all arguments.

Usage

```
mrgsim_ds(x, ..., tags = list(), verbose = FALSE, gc = TRUE)
```

Arguments

x	a model object loaded through mread_ds(), mcode_ds(), modlib_ds(), mread_cache_ds(), or house_ds().
...	passed to mrgsolve::mrgsim().
tags	a named list of atomic data to tag (or mutate) the simulated output.
verbose	if TRUE, print progress information to the console.
gc	initial gc setting; if TRUE, a finalizer function will attempt to remove files once the object is out of scope. This value is not locked: move_ds() and save_ds() will automatically adjust gc based on whether the files remain under tempdir(). To lock the gc setting and prevent automatic adjustment, call gc_ds() after creation.

Value

An object with class mrgsimsds.

See Also

as_mrgsim_ds(), mrgsimsds-methods.

Examples

```
mod <- house_ds()

data <- ev_expand(amt = 100, ID = 1:10)

out <- mrgsim_ds(mod, data, end = 72, delta = 0.1)

out <- mrgsim_ds(mod, data, tags = list(rep = 1))

head(out)
```

`mrgsimsds-methods` *Interact with mrgsimsds objects*

Description

Basic S3 methods for inspecting and plotting mrgsimsds objects: `dim()`, `head()`, `tail()`, `names()`, and `plot()`.

Usage

```
## S3 method for class 'mrgsimsds'
dim(x)

## S3 method for class 'mrgsimsds'
head(x, n = 6L, ...)

## S3 method for class 'mrgsimsds'
tail(x, n = 6L, ...)

## S3 method for class 'mrgsimsds'
names(x)

## S3 method for class 'mrgsimsds'
plot(
  x,
  y = NULL,
  ...,
  nid = 16,
  batch_size = 20000,
  logy = FALSE,
  .dots = list()
)
```

Arguments

`x`	an mrgsimsds object, output from `mrgsim_ds()` or `as_mrgsim_ds()`.
`n`	number of rows to return.
`...`	arguments to be passed to or from other methods.
`y`	a formula for plotting simulated data; if not provided, all columns will be plotted.
`nid`	number of subjects to plot.
`batch_size`	size of batch when reading data for plot method.
`logy`	if `TRUE`, plot data with log y-axis.
`.dots`	a list of items to pass to `mrgsolve::plot_sims()`.

Details

`head()` and `tail()` only look at the first and last `file` in the data set, respectively, when simulations are stored across multiple files. It is possible this won't correspond to the first and last chunks rows of data you will see when collecting the data via `dplyr::collect()`.

Value

- `dim()`: integer vector of length 2 (rows, cols).
- `head()`, `tail()`: a tibble of the first or last n rows.
- `names()`: character vector of column names.
- `plot()`: a plot object, returned invisibly.

Examples

```
mod <- house_ds(end = 24)

mod <- omat(mod, diag(0.04, 4))

data <- ev_expand(amt = c(100, 300), ID = 1:20)

set.seed(10203)

out <- mrgsim_ds(mod, data = data)

dim(out)
head(out)
tail(out)
nrow(out)
ncol(out)
plot(out, ~ CP + RESP, nid = 10)
```

`mrgsimsds-verbs` *dplyr verbs for mrgsimsds objects*

Description

Standard dplyr verbs dispatched on an mrgsimsds object. Each verb extracts the underlying Arrow Dataset and forwards all arguments to the corresponding dplyr generic, returning a lazy Arrow query that can be materialized with `dplyr::collect()`.

Usage

```
## S3 method for class 'mrgsimsds'
group_by(.data, ..., .add = FALSE, .drop = TRUE)

## S3 method for class 'mrgsimsds'
```

```
select(.data, ...)

## S3 method for class 'mrgsimsds'
mutate(.data, ...)

## S3 method for class 'mrgsimsds'
filter(.data, ..., .preserve = FALSE)

## S3 method for class 'mrgsimsds'
arrange(.data, ..., .by_group = FALSE)

## S3 method for class 'mrgsimsds'
rename(.data, ...)

## S3 method for class 'mrgsimsds'
summarise(.data, ..., .groups = NULL)

## S3 method for class 'mrgsimsds'
distinct(.data, ..., .keep_all = FALSE)

## S3 method for class 'mrgsimsds'
relocate(.data, ..., .before = NULL, .after = NULL)

## S3 method for class 'mrgsimsds'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'mrgsimsds'
pull(.data, var = -1, name = NULL, as_vector = TRUE, ...)
```

Arguments

.data, x	an mrgsimsds object.
...	passed to the corresponding dplyr generic.
.add, .drop	passed to dplyr::group_by().
.preserve	passed to dplyr::filter().
.by_group	passed to dplyr::arrange().
.groups	passed to dplyr::summarise().
.keep_all	passed to dplyr::distinct().
.before, .after	passed to dplyr::relocate().
wt, sort	passed to dplyr::count().
name	passed to dplyr::pull().
var	passed to dplyr::pull().
as_vector	passed to dplyr::pull().

Value

A lazy Arrow query object. Use dplyr::collect() to materialize the result into a tibble. pull() is an exception — it collects immediately and returns a vector.

Examples

```
library(dplyr)

mod <- house_ds(end = 24)

data <- evd_expand(amt = c(100, 300), ID = 1:10)

out <- mrgsim_ds(mod, data)

out |> filter(TIME > 0) |> select(ID, TIME, CP) |> collect()

out |> group_by(ID) |> summarise(auc = sum(CP)) |> collect()

out |> mutate(WEEK = TIME / 168) |> collect()
```

---

ownership *Ownership of simulation files*

Description

Functions to check ownership or disown simulation output files on disk.

One situation where you need to take over ownership is when you are simulating in parallel, and the simulation happens in another R process. mrgsim.ds ownership is established when the simulation returns and the mrgsimsds object is created. When this happens in another R process (e.g., on a worker node), there is no way to transfer that information back to the parent process. In that case, a call to take_ownership() once the results are returned to the parent process would be appropriate. Typically, these results are returned as a list and a call to reduce_ds() will create a single object pointing to and owning multiple files. Therefore, it should be rare to call take_ownership() directly; if doing so, please make sure you understand what is going on.

Usage

```
ownership()

list_ownership(full.names = FALSE)

check_ownership(x)

disown(x)

take_ownership(x)
```

Arguments

full.names	if TRUE, include the directory path when listing file ownership.
x	an mrgsimsds object.

Value

- check_ownership: TRUE if x owns the underlying files; FALSE otherwise.
- list_ownership: a data.frame of ownership information.
- ownership: nothing; used for side effects.
- disown: x is returned invisibly; it is not modified.
- take_ownership: x is returned invisibly after its hash and the package-level ownership maps are updated in place.

See Also

reduce_ds(), copy_ds().

Examples

```
mod <- house_ds()

out <- mrgsim_ds(mod, id = 1)

check_ownership(out)

ownership()

list_ownership()

e1 <- ev(amt = 100)
e2 <- ev(amt = 200)

out <- list(mrgsim_ds(mod, e1), mrgsim_ds(mod, e2))

sims <- reduce_ds(out)

ownership()

check_ownership(sims)

check_ownership(out[[1]])

check_ownership(out[[2]])
```

print.mrgsimsds *Print an mrgsimsds object*

Description

Print an mrgsimsds object

Usage

```
## S3 method for class 'mrgsimsds'
print(x, n = 8, ...)
```

Arguments

x	an mrgsimsds object.
n	number of rows to show from the cached head data.
...	not used.

Value

x invisibly.

Examples

```
mod <- house_ds(end = 24)

out <- mrgsim_ds(mod, events = ev(amt = 100))

print(out)
```

prune_ds *Prune a list of mrgsimsds objects*

Description

Filters a mixed list down to only the elements that are mrgsimsds objects, dropping anything else (e.g. NULL, data frames, character vectors). When passed a single mrgsimsds object it is returned invisibly unchanged.

Usage

```
prune_ds(x, ..., inform = TRUE)

## S3 method for class 'mrgsimsds'
prune_ds(x, ...)

## S3 method for class 'list'
prune_ds(x, ..., inform = TRUE)
```

Arguments

x	a list of R objects or a single mrgsimsds object.
...	not used.
inform	(list method only) issue a message when objects in some list slots are dropped.

Value

When x is a list, it will be returned with only the mrgsimsds objects retained. If no mrgsimsds objects are found, an empty list is returned with a warning.

When x is an mrgsimsds object, it will be invisibly returned.

Examples

```
mod <- house_ds(end = 24)

out <- mrgsim_ds(mod, events = ev(amt = 100))

sims <- list(out, letters)

prune_ds(sims)
```

---

reduce_ds *Reduce a list of mrgsimsds objects into a single object*

Description

Combines a list of mrgsimsds objects — typically the replicate outputs from a parallel simulation — into one object backed by all of their parquet files. Ownership of every file is transferred to the new object.

Usage

```
reduce_ds(x, ...)

## S3 method for class 'mrgsimsds'
reduce_ds(x, ...)

## S3 method for class 'list'
reduce_ds(x, ...)
```

Arguments

x	a list of mrgsimsds objects or a single mrgsimsds object.
...	not used.

Details

gc behavior:
The returned object always gets a fresh, unlocked gc state: `gc_locked` is set to `FALSE` and gc is determined by file location via the same rule used at creation time — `TRUE` if files are under `tempdir()`, `FALSE` otherwise. Any gc lock set on the input objects is not carried over. To lock the gc setting on the result, call `gc_ds()` after reducing.

Value

When `x` is a list, a new mrgsimsds object is returned that owns all underlying parquet files; the input objects are disowned.

When `x` is an mrgsimsds object, it is validated, refreshed, and returned invisibly with its `pid` updated to the current process.

Examples

```
mod <- modlib_ds("popex", outvars = "IPRED")

data <- ev_expand(amt = 100, ID = 1:10)

out <- lapply(1:3, function(rep) {
  out <- mrgsim_ds(mod, data)
  out
})

length(out)

sims <- reduce_ds(out)

sims

check_ownership(sims)

lapply(out, check_ownership)
```

refresh_ds *Refresh 'Arrow' dataset pointers*

Description

Arrow dataset pointers become invalid when an object is created in a worker process and returned to the head node (e.g. after a parallel simulation). `refresh_ds()` rebuilds the pointer by re-opening the parquet files via `arrow::open_dataset()` and updates `pid` and `dim` in place. Because refreshing is itself the fix for an invalid pointer, it checks that files exist but does not call `safe_ds()` first.

Usage

```
refresh_ds(x, ...)

## S3 method for class 'mrgsimsds'
refresh_ds(x, ...)

## S3 method for class 'list'
refresh_ds(x, ...)
```

Arguments

`x`	an mrgsimsds object or a list of objects.
`...`	for future use.

Value

When `x` is an mrgsimsds object, it is returned invisibly with its Arrow pointer, `pid`, and `dim` refreshed in place.

When `x` is a list, it is returned invisibly with `refresh_ds()` applied to every mrgsimsds element; non-mrgsimsds elements are left unchanged.

Examples

```
mod <- house_ds()

data <- ev_expand(amt = 100, ID = 1:100)

out <- lapply(1:3, function(rep) {
  mrgsim_ds(mod, data)
})

refresh_ds(out)
```

save_ds *Save and restore an mrgsimsds object*

Description

`save_ds()` serializes an mrgsimsds object to an `.rds` file, moving the backing parquet files to the same directory as `file`. Parquet filenames are stored as bare basenames inside the `.rds`, so the `.rds` file and its parquet files must stay in the same directory to be portable. **Do not restore the file with** `readRDS()`; use `read_ds()` instead.

`read_ds()` deserializes a file written by `save_ds()`, rebuilds the Arrow Dataset pointer, and transfers full ownership of the backing files to the returned object.

Usage

```
save_ds(x, file, quietly = FALSE)

read_ds(file)
```

Arguments

`x`	an mrgsimsds object.
`file`	for `save_ds()`, the path to the output `.rds` file; the directory component determines where backing parquet files are moved. For `read_ds()`, the path to an `.rds` file written by `save_ds()`.
`quietly`	if `FALSE`, a message is printed about the potentially new location of the backing files on move.

Value

`save_ds()` returns the path to the written `.rds` file, invisibly.

`read_ds()` returns the restored mrgsimsds object invisibly. gc is disabled (`gc = FALSE`) on the returned object and the caller holds ownership of the backing files.

See Also

`move_ds()`, `gc_ds()`

Examples

```
mod <- house_ds()

out <- mrgsim_ds(mod, events = ev(amt = 100))

file <- save_ds(out, file.path(tempdir(), "out.rds"))

out2 <- read_ds(file)
```

save_process_info *Save information about the R process that loaded a model*

Description

Stamps the model object with the current process ID and `tempdir()` path so that `mrgsim_ds()` knows where to write output files. This is called automatically by `mread_ds()`, `house_ds()`, and the other model-loading wrappers. Call it directly only when you load a model through the base mrgsolve functions (e.g. `mrgsolve::mread()`) and still want to use `mrgsim_ds()`.

Usage

```
save_process_info(x)
```

Arguments

`x`	a model object.

Value

An updated model object suitable for using with `mrgsim_ds()`.

Examples

```
mod <- mrgsolve::house()

mod <- save_process_info(mod)
```

write_parquet_ds *Write simulations to a parquet file or partitioned dataset*

Description

Use these functions to escape the `mrgsim.ds` universe. `write_parquet_ds()` writes all simulated data to a single `.parquet` file. `write_dataset_ds()` writes to a directory, optionally partitioned, via `arrow::write_dataset()`; the caller takes responsibility for the resulting files.

Usage

```
write_parquet_ds(x, sink, ...)

write_dataset_ds(x, path, ...)
```

Arguments

x	an mrgsimsds object.
sink	passed to arrow::write_parquet().
...	passed to the underlying arrow function.
path	passed to arrow::write_dataset().

Value

write_parquet_ds() returns x invisibly.

write_dataset_ds() returns path invisibly.

See Also

save_ds() to persist an object while staying within the mrgsim.ds universe.

Index