Package: movedesign (via r-universe)

October 16, 2026

Title Study Design Toolbox for Movement Ecology Studies

Version 0.3.3

Maintainer Inês Silva <i.simoes-silva@hzdr.de>

Description Toolbox and 'shiny' application to help researchers design movement ecology studies, focusing on two key objectives: estimating home range areas, and estimating fine-scale movement behavior, specifically speed and distance traveled. It provides interactive simulations and methodological guidance to support study planning and decision-making. The application is described in Silva et al. (2023) <doi:10.1111/2041-210X.14153>.

License GPL (>= 3)

URL https://ecoisilva.github.io/movedesign/, https://ecoisilva.r-universe.dev/movedesign

BugReports https://github.com/ecoisilva/movedesign/issues

Depends R (>= 4.1.0)

Imports bayestestR, bsplus, config (>= 0.3.1), crayon, ctmm (>= 0.6.1), data.table, dplyr, fontawesome, gdtools, gfonts, ggiraph, ggplot2, ggtext, golem (>= 0.3.2), grDevices, lubridate, parallel, parsedate, patchwork, quarto, reactable, rintrojs, rlang, scales, shiny (>= 1.7.1), shinyalert, shinybusy, shinydashboard, shinydashboardPlus, shinyFeedback, shinyjs, shinyWidgets, stats, stringr, terra, tidyr, tools, utils, viridis

Suggests knitr, rmarkdown, shinytest2

VignetteBuilder knitr, quarto

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE, roclets = c(``rd", ``collate", ``namespace"))

Config/roxygen2/version 8.0.0

Config/pak/sysreqs
libabsl-dev libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libgmp3-dev make libgsl0-dev libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libmpfr-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://ecoisilva.r-universe.dev

Date/Publication 2026-06-18 13:52:45 UTC

RemoteUrl https://github.com/ecoisilva/movedesign

RemoteRef HEAD

RemoteSha 2c42fdc73a3e7caeacd3cb4e828fb4b0729daaa9

Contents

fitting_models . . . 3
fixrates . . . 4
md_check . . . 4
md_compare . . . 6
md_compare_preview . . . 8
md_configure . . . 10
md_merge . . . 11
md_optimize . . . 13
md_plot . . . 15
md_plot_preview . . . 17
md_plot_replicates . . . 18
md_prepare . . . 20
md_replicate . . . 23
md_run . . . 25
md_simulate . . . 27
md_stack . . . 30
movmods . . . 31
plot.movedesign_optimized . . . 32
print.movedesign_check . . . 32
print.movedesign_input . . . 33
print.movedesign_optimized . . . 33
print.movedesign_output . . . 34
print.movedesign_processed . . . 35
print.movedesign_report . . . 36
read_md . . . 36
run_app . . . 37
summary.movedesign_check . . . 38
summary.movedesign_input . . . 38
summary.movedesign_optimized . . . 39
summary.movedesign_output . . . 40
summary.movedesign_processed . . . 41
summary.movedesign_report . . . 42

Index **43**

`fitting_models` *Fit continuous-time movement models*

Description

This function fits continuous-time movement models to simulated location data using the `ctmm` package. It estimates movement parameters for each simulated trajectory, allowing for parallel execution. It currently supports both home range and speed estimation workflows.

Usage

```
fitting_models(
  obj,
  set_target = c("hr", "ctsd"),
  parallel = FALSE,
  trace = FALSE,
  ncores = parallel::detectCores(),
  ...
)
```

Arguments

`obj`	A list of simulated movement datasets, each a `telemetry` object compatible with `ctmm` R package.
`set_target`	A character vector specifying the research targets. Current options: **"hr"** Home range estimation. **"ctsd"** Speed & distance estimation.
`parallel`	Logical. If `TRUE`, enables parallel processing.
`trace`	Logical. If `TRUE` (default), prints progress and timing messages to the console.
`ncores`	Integer. Number of CPU cores to use for parallel processing. Defaults to all available cores minus one.
`...`	Additional arguments used internally.

Details

The function generates initial parameter estimates for each dataset using `ctmm::ctmm.guess()`. If the data includes simulated location error, it adds an error model accordingly. Models are fitted using `ctmm::ctmm.select()`, which performs model selection to find the best-fit movement process. Finally, all fitted models are recentered to `(0, 0)` for downstream consistency.

Value

A list of fitted movement models, all recentered to the origin.

Note

This function is intended for internal use and may assume inputs follow specific structure and constraints not referenced explicitly.

See Also

`ctmm::ctmm.guess()`, `ctmm::ctmm.select()`

---

fixrates *Fix rates of animal tracking devices.*

Description

A dataset listing typical GPS fix rates for animal tracking devices. Useful for selecting typical sampling schedules in wildlife tracking projects.

Usage

```
fixrates
```

Format

A data.frame with 40 rows and 7 variables:

dti_notes Human-readable fix schedule, e.g., "1 fix every month". Helps interpret sampling intervals in practical terms.

dti Sampling interval in seconds, i.e., time between consecutive location fixes.

frq Sampling frequency in seconds, i.e., how often a fix occurs (inverse of dti).

frq_hrs Sampling frequency in hours, offering a more intuitive unit for comparison.

highlighted Logical. TRUE if the fix rate is commonly used in animal tracking studies. Useful for identifying standard settings. ...

---

md_check *Assess output convergence in simulation outputs*

Description

Evaluates whether the cumulative mean of a tracked error metric in simulation outputs has stabilized, indicating convergence. This function helps determine if repeated simulations or resampling have produced stable estimates, which is critical for reliable inference in animal movement projects.

Use this function after running `md_run()` or `md_replicate()` to check the reliability of outputs before further interpretation or reporting.

Usage

```
md_check(
  obj,
  m = NULL,
  tol = 0.05,
  n_converge = 9,
  plot = TRUE,
  pal = c("#007d80", "#A12C3B")
)
```

Arguments

obj	A movedesign or related object returned by md_run() or md_replicate().
m	Numeric (optional). If provided, restricts the convergence check to results for a specific population sample size (m). Defaults to NULL, which checks up to the maximum population sample size.
tol	Numeric. The tolerance threshold for absolute change in the cumulative mean to declare convergence. Defaults to 0.05.
n_converge	Integer. Number of consecutive steps within tolerance required to confirm convergence.
plot	Logical. If TRUE (default), generates a plot of stepwise changes in the cumulative mean, highlighting when convergence is achieved.
pal	Character vector of color(s) of the plot, such as c("#007d80", "#A12C3B")) (default).

Details

The cumulative mean of error is calculated, and the absolute changes over the last n_converge steps are inspected. If all are below the specified tolerance, convergence is declared.

If plot = TRUE, a plot is shown of absolute stepwise change in the cumulative mean, with a shaded region indicating the convergence threshold, aiding visual assessment.

Value

An object of class "movedesign_check" with the following elements:

has_converged Logical scalar indicating whether convergence was achieved.

recent_deltas Numeric vector of absolute changes in cumulative mean over the last n_converge steps.

max_delta Maximum absolute change among the last steps.

tolerance Numeric, the input tolerance tol.

n_converge Integer, the input n_converge.

variable Character. Name of the variable checked.

recent_cummean Numeric vector. The last cumulative means checked.

See Also

md_run(), md_replicate()

Examples

```
if(interactive()) {

  output <- md_replicate(input, n_replicates = 20)
  md_check(output, tol = 0.05, n_converge = 10)

}
```

`md_compare` *Compare estimation error across study design workflows*

Description

The final step in the `movedesign` workflow. Takes replicated outputs from two or more study designs, ranks them by estimation performance, identifies the best-performing design, and produces a density plot of relative error across replicates.

Use this function after running `md_replicate()` for each design and confirming convergence with `md_check()`. For a quick visual comparison before full replication, use `md_compare_preview()` instead.

Usage

```
md_compare(
  x,
  stat = c("mean", "median"),
  ci = 0.8,
  method = "HDI",
  pal = c("#007d80", "#A12C3B"),
  m = NULL,
  show_text = TRUE
)
```

Arguments

`x`	A list of at least two `movedesign_output` objects, each returned by `md_replicate()`. All designs must share the same `set_target`. Designs typically differ in sampling parameters such as `dur`, `dti`, or `n_individuals`.
`stat`	Character. Summary statistic used to represent the centre of the error distribution in the plot and ranking. Must be `"mean"` (default) or `"median"`.
`ci`	Numeric. Coverage probability of the credible interval computed over the distribution of relative error across replicates. Must be strictly between 0 and 1. Defaults to `0.80` (80% CI).

method	Character. Method used to compute the credible interval, passed to bayestestR::ci(). Defaults to "HDI" (Highest Density Interval), which is generally preferred for asymmetric or skewed error distributions. See ?bayestestR::ci for all available options.
pal	A character vector of colors for the density curves, CI shading, and centre line. Defaults to c("#007d80", "#A12C3B").
m	(Optional) Numeric. If provided, restricts all results and ranking to a specific *population* sample size. Defaults to NULL, which uses the maximum sample size defined by n_individuals.
show_text	Logical. Whether to display annotation text in the plot with the mean (or median) relative errors. Defaults to TRUE.

Details

Designs are ranked separately for each target metric (home range, speed) and group (if present). The ranking criterion combines absolute relative error and distance of the credible interval from zero: designs with lower error and tighter intervals closer to zero rank higher. A design is identified as the overall winner only if it ranks first across all groups for a given target.

The function prints a density plot showing the full distribution of relative error across replicates for each design. The centre statistic (stat) and credible interval (ci) are overlaid. When groups are present, density curves for each group are shown side by side using the colors in pal.

Recommended workflow:

This function is designed to be called at the end of the movedesign workflow:

1. Build each design with md_prepare().
2. Run md_replicate() for each design.
3. Confirm convergence with md_check().
4. Compare and rank designs with md_compare().

Value

An object of class movedesign_report. A density plot of relative error is printed as a side effect; to reproduce it later, call md_plot() on any of the input designs.

This object contains:

ranking Data frame with one row per design per target (and per group if grouping is used). Columns include the centre error statistic (error), credible interval bounds (error_lci, error_uci), CI width, distance from zero error, and rank. Lower rank indicates better performance.

winners Data frame identifying designs that rank first across all groups for each target. A design is a winner when it has the lowest absolute error and the credible interval closest to zero across every group. Returns empty if no single design dominates.

See Also

md_replicate(), md_check() for convergence diagnostics, and refer to bayestestR::ci() for details on credible interval computation and interpretation.

Other workflow_steps: md_prepare(), md_replicate(), md_run(), md_simulate()

Examples

```
if (interactive()) {

  data(buffalo)
  inputA <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 1, unit = "month"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = TRUE,
    grouped = TRUE,
    set_target = "hr",
    which_meta = "mean")

  inputB <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 10, unit = "days"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = TRUE,
    grouped = TRUE,
    set_target = "hr",
    which_meta = "mean")

  outputA <- md_replicate(inputA, n_replicates = 20)
  outputB <- md_replicate(inputB, n_replicates = 20)

  # Plot with 80% credible intervals:
  md_compare(list(outputA, outputB), ci = 0.80, method = "HDI")

}
```

---

md_compare_preview *Compare estimation error across designs (single replicate)*

Description

Plots estimation error of the chosen targets for two or more study designs, each represented by a single `md_run()` output. Use this function to quickly compare how design choices affect estimation performance before committing to a full replication with `md_replicate()`.

Because each design is represented by a single stochastic run, results are preliminary. For robust, publication-ready comparisons, run `md_replicate()` for each design and compare with `md_compare()`.

Usage

```
md_compare_preview(
  ...,
  n_resamples = NULL,
  error_threshold = 0.05,
  pal = c("#007d80", "#A12C3B")
)
```

Arguments

`...`	One or more `movedesign_processed` objects, each returned by `md_run()`, or a single list containing such objects. Each element represents one study design to compare. Designs typically differ in sampling parameters such as `dur`, `dti`, or `n_individuals`, but any valid inputs can be compared.
`n_resamples`	A single positive integer. The number of random combinations of individuals generated at each population sample size per design. Each combination produces one population-level estimate. Set to `NULL` to plot raw estimates without resampling.
`error_threshold`	Numeric. Relative error threshold shown as a horizontal reference line in the plot (e.g. `0.05` for 5%).
`pal`	A character vector of two colors, used for estimates within and outside the error threshold respectively. Defaults to `c("#007d80", "#A12C3B")`.

Details

If `n_resamples` is not `NULL`, the function draws `n_resamples` random combinations of individuals at each population sample size and computes a population-level estimate for each. This step

Each design is represented by a single stochastic run. Apparent differences between designs may reflect random variation rather than genuine performance differences. Use `md_replicate()` to generate robust, replicated results for each design, and `md_compare()` to compare multiple designs.

Value

A `ggplot` object. Displays relative error as a function of population sample size, with one panel (or two if two target) per design. Point estimates, confidence intervals, and a horizontal reference line at `error_threshold`.

See Also

`md_run()` to generate each input object. `md_plot_preview()` for a single-design equivalent. `md_replicate()` for robust multi-replicate outputs per design. `md_check()` to assess convergence across replicates.

Examples

```
if (interactive()) {

  data(buffalo)
```

```
  inputA <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 1, unit = "month"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = FALSE,
    grouped = TRUE,
    set_target = "hr",
    which_meta = "mean")

  inputB <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 10, unit = "days"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = TRUE,
    grouped = TRUE,
    set_target = "hr",
    which_meta = "mean")

  outputA <- md_run(inputA)
  outputB <- md_run(inputB)
  md_compare_preview(list(outputA,
                          outputB), error_threshold = 0.05)
}
```

`md_configure` *Interactively configure movement design setup*

Description

Guides the user to assign each argument required for a movement design workflow, including species label and key simulation settings. Users may choose to set a specific population sample size (number of animals tagged/to be tagged) or optimize the population sample size considering a specific analytical target.

Usage

```
md_configure(data, models = NULL, parallel = FALSE)
```

Arguments

`data`	A named list of simulated movement datasets, each a `telemetry` object compatible with `ctmm` `R` package. Each object must contain valid metadata and timestamped locations.

models	(Optional) Named list of fitted ctmm models (from ctmm::ctmm.fit() or ctmm::ctmm.select()). If not supplied, models are fitted automatically.
parallel	Logical. If TRUE, enables parallel processing.

Details

The argument data is **required** and must be supplied directly (as a list of telemetry objects, obtained from ctmm::as.telemetry()). The argument models is optional, and if omitted, models will be fitted automatically.

Value

An object of class movedesign_input (and movedesign). This is a structured S3 list containing all validated inputs, model fits, and derived parameters for the study design workflow.

See Also

md_prepare()

Examples

```
if(interactive()) {
  data(buffalo)
  md_params <- md_configure(data = buffalo)
}
```

---

md_merge *Merge multiple simulation outputs*

Description

Pools two or more md_run() outputs into a single movedesign_processed object by concatenating all simulated individuals, fitted models, and seeds. The merged object behaves exactly as if all individuals had been simulated in one call: if each input contains 5 individuals, the merged output contains 10.

The distinction from md_stack() is important. md_merge() treats all inputs as parts of one larger dataset; replicate identity is lost and individual counts accumulate. md_stack() instead assigns a replicate ID to each md_run() output and aggregates population-level inference across them, keeping each run as a separate replicate.

Call md_merge() directly only when you have run md_run() separately and need to pool the raw outputs before downstream analyses.

Usage

```
md_merge(x, ...)
```

Arguments

x	Either a list of movedesign_processed objects, or the first of multiple objects passed individually. All objects must share the same set_target and sampling parameters.
...	Reserved for internal use.

Details

Before merging, all inputs are checked for consistent metadata (e.g. dur, dti, set_target). Movement timescale parameters (tau_p, tau_v) are compared after rounding to one decimal place, to tolerate minor numerical differences arising from separate model fitting runs. If any field mismatches are found, the function stops with an informative message listing the affected fields.

Value

A single movedesign_output object that contains all merged simulation outputs and inherits metadata from the first input object.

See Also

md_prepare, md_run

Examples

```
if (interactive()) {

  data(buffalo)
  input <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 1, unit = "month"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = FALSE,
    set_target = "hr",
    which_meta = "mean")

  output1 <- md_run(input)
  output2 <- md_run(input)

  # Both of the following are equivalent:

  md_merge(output1, output2)
  md_merge(list(output1, output2))

}
```

`md_optimize` *Optimize sampling parameters and population sample size*

Description

Repeatedly simulates movement datasets across a range of candidate population sample sizes to identify the minimal sample size and associated sampling parameters (i.e., duration, sampling interval) needed to achieve estimates for key movement and space-use metrics (e.g., home range area, speed) within the specified relative error threshold.

The function quantifies estimation error for each metric and sample size, evaluating which population sample size reliably meet target thresholds, and reports final recommendations.

Usage

```
md_optimize(
  obj,
  n_replicates = 10,
  error_threshold = 0.05,
  plot = FALSE,
  verbose = TRUE,
  parallel = FALSE,
  ncores = parallel::detectCores(),
  trace = TRUE,
  ...
)
```

Arguments

`obj`	A movement design input object (see `md_prepare()` or `md_configure()`).
`n_replicates`	Integer. Number of simulation replicates at each candidate sample size.
`error_threshold`	Numeric. Upper limit of the relative error in estimation (e.g., `0.05` for 5%) deemed acceptable by the user. The function will attempt to find sampling parameters and sample sizes that keep errors below this threshold.
`plot`	Logical. If TRUE, displays a diagnostic plot of the final results.
`verbose`	Logical. If `TRUE` (default), prints a summary of the convergence check to the console.
`parallel`	Logical; if `TRUE`, enables parallel processing. Default is `FALSE`.
`ncores`	Integer; number of CPU cores to use for parallel processing. Defaults to all available cores detected by `parallel::detectCores()`.
`trace`	Logical; if `TRUE` (default), prints progress and timing messages to the console.
`...`	Reserved for internal use.

Details

The function iteratively runs movement design simulations for increasing population sample sizes (`m`), evaluating error for each replicate and metric via meta-analyses. Convergence is checked using the error threshold and stability of cumulative mean error. The function stops when a sample size meets all criteria or the maximum population sample size is reached. Results can be visualized using if `plot = TRUE`.

Value

A list of class `movedesign_report` containing:

- `summary`: Data frame of summary statistics for each replicate, sample size, and metric.
- `error_threshold`: Numeric. The error threshold used.
- `sampling_duration`: Character string. Final sampling duration.
- `sampling_interval`: Character string. Final sampling interval.
- `sample_size_achieved`: Logical. Indicates if convergence was achieved and the threshold met.
- `init_m`: Integer. Maximum sample size evaluated.
- `minimum_m`: Integer. Minimum sample size achieving the threshold.

Note

Some biologgers inherently involve a trade-off between fix frequency and battery life. Shorter intervals between location fixes offer higher temporal resolution but reduce deployment duration due to increased power consumption. In contrast, longer deployments require less frequent sampling to conserve battery.

This trade-off makes it challenging to estimate multiple metrics with differing data needs: high-resolution data (shorter intervals) improve speed estimation, while extended deployments (longer durations) are vital for accurate home range area estimates. A sampling design that minimizes error for one metric may increase error for another.

Researchers facing these constraints should consider prioritizing a single research target (e.g., either home range area *or* speed), or use stratified designs to balance data needs across individuals.

See Also

`md_prepare()`, `md_configure()`

Examples

```
if(interactive()) {
  obj <- md_configure(data = buffalo,
                      models = models)

  out <- md_optimize(tmp,
                     error_threshold = 0.05,
                     plot = TRUE)
}
```

`md_plot` *Visualize study design outputs*

Description

Produces a publication-ready density plot showing the distribution of relative error estimates from study design simulations. The plot highlights the mean and a shaded credible interval (CI) region, following the computation of credible intervals as implemented in `bayestestR::ci()`. If groups are present, density curves for each group are overlaid for comparison, using customizable colors.

This function is typically used after running `md_replicate()`, providing a visual diagnostic of simulation results.

Usage

```
md_plot(
  x,
  stat = c("mean", "median"),
  ci = 0.8,
  method = "HDI",
  pal = c("#007d80", "#A12C3B"),
  m = NULL,
  show_text = TRUE,
  ...
)
```

Arguments

`x`	A `movedesign_output` object, as returned by `md_replicate()`. The object must contain a `summary` data frame with, at a minimum, the following columns: **error** Relative error values for each replicate. **error_lci** Lower credible interval bound for error. **error_uci** Upper credible interval bound for error. **group** (Optional) Group label for comparing densities.
`stat`	Character string specifying which summary statistic to display. Must be `"mean"` or `"median"`. Defaults to `"mean"`.
`ci`	Numeric scalar between 0 and 1. The probability of the credible interval (CI) to be estimated. Default to `0.80` (80%).
`method`	Character. Credible interval estimation method (passed to `bayestestR::ci()`; default: `"HDI"`). See `?bayestestR::ci()` for more details.
`pal`	Character vector of color(s) for the density, CI shading, and mean line. If a single group, supply one color (default: `"#007d80"`). If groups are present, supply two colors (default: `c("#007d80", "#A12C3B")`).
`m`	Numeric (Optional). If provided, restricts the results to a specific population sample size (`m`). Defaults to `NULL`, which checks up to the maximum population sample size.

show_text	Logical, whether to display text annotations in the plots. Default is TRUE.
...	Reserved for internal use.

Details

This plot helps users assess the reliability of simulation outputs by visualizing the distribution of relative errors. When multiple groups are simulated, the plot enables direct visual comparison of performance across groups. If credible intervals cannot be calculated, a warning is issued and only the density curves are displayed.

It is strongly recommended to use md_check() to assess whether the distributions shown here have stabilized. Checking for convergence ensures that the summary statistics and uncertainty estimates depicted in the plot are reliable and not unduly influenced by too few replicates or ongoing variability. Running md_check() helps you determine if additional simulation replicates are needed to achieve stable inference in your design evaluation.

Value

A ggplot object showing:

- Density curve(s) of the relative error distribution.
- Shaded region for the central credible interval.
- Vertical dashed lines at mean(s).
- Overlaid densities if multiple groups are present.
- Percent-formatted x-axis for interpretation.

This object can be further customized with additional ggplot2 layers if needed.

See Also

md_replicate(), md_check() for convergence diagnostics, and refer to bayestestR::ci() for details on credible interval computation and interpretation.

Examples

```
if (interactive()) {
  input <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 1, unit = "month"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = TRUE,
    grouped = TRUE,
    set_target = "hr",
    which_meta = "mean"
  )

  output <- md_replicate(input, n_replicates = 20)
```

```
  # Plot with 80% credible intervals:
  md_plot(output, ci = 0.80, method = "HDI")
}
```

`md_plot_preview` *Preview plot for movedesign workflow outputs (single replicate)*

Description

Generates a quick visualization of relative error for home range or movement speed estimation from a single replicate of a movedesign workflow. The plot can display either the estimates from that replicate for a random combination of individuals, or, when resampling is enabled, summaries derived from repeated draws of individuals at each population sample size (based on the specified number of resamples).

This functions shows preliminary outputs for a single stochastic run from `md_run()` (a `movedesign_processed` object) and should not be used to evaluate study design by itself. Instead, users should run `md_replicate()` and check for convergence with `md_check()`.

Usage

```
md_plot_preview(
  obj,
  n_resamples = NULL,
  error_threshold = 0.05,
  pal = c("#007d80", "#A12C3B"),
  ...
)
```

Arguments

`obj`	An object of class `movedesign_processed`, as returned by `md_run()`.
`n_resamples`	A single positive integer. The number of random combinations of individuals generated at each population sample size. Each combination produces one population-level estimate. Set to `NULL` to plot raw estimates without resampling.
`error_threshold`	Numeric. Relative error threshold shown as a reference line in the plot (e.g. `0.05` for 5%).
`pal`	A character vector of two colors, used for estimates within and outside the error threshold respectively. Defaults to `c("#007d80", "#A12C3B")`.
`...`	Reserved for internal use.

Details

This plot summarizes a single replicate, so it is subject to stochastic variation. The plot shown here may look very different with another run of the same design. Use `md_replicate()` to aggregate results across many independent runs, and `md_check()` to confirm that estimates have stabilised before drawing conclusions.

Value

A `ggplot` object. Displays relative error as a function of population sample size, with point estimates, confidence intervals, and a horizontal reference line at `error_threshold`.

See Also

`md_run()` to generate the input object. `md_replicate()` for robust outputs based on multiple replicates. `md_check()` to assess convergence across replicates.

Examples

```
if (interactive()) {

  data(buffalo)
  input <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 1, unit = "month"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = FALSE,
    grouped = TRUE,
    set_target = "hr",
    which_meta = "mean")

  output <- md_run(input)
  md_plot_preview(output, error_threshold = 0.05)
}
```

`md_plot_replicates` *Plot estimation error, replicates and confidence intervals*

Description

This function produces two complementary visualizations of replicate performance across a range of population sample sizes m, up to the maximum number of individuals requested. It summarizes and plots relative errors regarding the estimation of a set target (*e.g.*, home range or speed estimation), and classified by whether the error falls within a user-defined acceptable threshold.

"Estimation error (for a single random replicate)" Error for one randomly-selected replicate, rendered as point estimates with confidence interval bars. Reflects the outcome a researcher would observe from a *single* empirical study.

"Mean estimation error across all replicates" Mean error collapsed across **all** replicates (with confidence intervals as a narrow bar, and prediction intervals as a wide shaded band), with per-replicate jitter in the background. Conveys the full distribution of outcomes with the set sampling parameters and population sample size.

Both panels share a common color palette (within/outside the acceptable error threshold) and, when two groups are present, a common shape scale. By default, the function plots only the second item listed above.

Usage

```
md_plot_replicates(
  obj,
  ci = 0.95,
  view = "summary_only",
  pal = c("#007d80", "#A12C3B"),
  ...
)
```

Arguments

`obj`	A movement design output object (returned by either `md_replicate()` or `md_optimize()`).
`ci`	Confidence level for the intervals. Applied to both the narrow confidence bars and wide prediction bands. Must be between `0` and `1`. Default: `0.95` (95%).
`view`	Layout selector. Indicate whether to return the complete two-panel layout (`"both"`) or only the aggregated summary plot with all replicates (`"summary_only"`).
`pal`	Character vector of two valis hex color code for within and for outside the threshold (default: `c("#007d80", "#A12C3B")`).
`...`	Reserved for internal use.

Value

A `patchwork` / `ggplot` object containing:

- Top plot: A `ggplot` object displaying the results from a single randomly selected replicate, showing individual error estimates and their confidence intervals.
- Bottom plot: A `ggplot` object summarizing mean relative error across all replicates, with aggregated estimates in the foreground and individual replicates shown in lighter tones in the background.

See Also

`md_plot()` for the density plot for the maximum m.
`md_replicate()` to produce a `movedesign_output`.
`md_optimize()` to produce a `movedesign_optimized`.

Examples

```
if(interactive()) {

  obj <- md_replicate(...)

  # Default: both panels
  md_plot_replicates(obj)

  # Summary panel, custom palette, 80% CI:
  md_plot_replicates(
    obj,
    ci = 0.90,
    view = "summary_only",
    pal = c("#1e2a38", "#9e3419"))

}
```

`md_prepare` *Prepare movement study design inputs*

Description

Prepares and validates all inputs needed to evaluate the study design of animal movement projects using parameters derived from empirical tracking data. The function checks data integrity, fits or verifies movement models, extracts key parameters, and consolidates all settings into a structured object for reproducible and streamlined downstream analyses.

If you do not have empirical data, use `md_simulate()` instead, which builds inputs from user-specified parameters.

Usage

```
md_prepare(
  species = NULL,
  data,
  models = NULL,
  n_individuals = NULL,
  dur = NULL,
  dti = NULL,
  set_target = c("hr", "ctsd"),
  which_meta = "mean",
  add_individual_variation = FALSE,
  groups = NULL,
  parallel = FALSE,
  .seed = NULL
)
```

Arguments

species	Character. A label for the focal species (scientific or common name). Used for display and bookkeeping only; does not affect results.
data	A named list of telemetry objects, created with `ctmm::as.telemetry()`, to be used as the empirical basis for the simulations. Each telemetry object must contain valid metadata and timestamped locations.
models	(Optional) Named list of fitted movement models, one per individual, created with `ctmm::ctmm.select()`. Names must match those in `data`. If not supplied, models are fitted automatically.
n_individuals	A single positive integer. The target number of animals in the study design (equivalent to number of tags to be deployed in the field). This defines the *population* sample size used in downstream analyses, and does not need to match the number of individuals in `data`.
dur	Study duration. A list with elements `value` (numeric) and `unit` (character). **Example:** `list(value = 2, unit = "months")` Valid units: `"second"`, `"minute"`, `"hour"`, `"day"`, `"month"`, `"year"`.
dti	Sampling interval between consecutive GPS fixes. A list with elements `value` (numeric) and `unit` (character). Same valid units as `dur`. **Example:** `list(value = 1, unit = "day")`
set_target	Character vector specifying the target metrics to be evaluated in the study design workflow. Choose one or both: • `"hr"` - home range area • `"ctsd"` - continuous-time speed and distance Defaults to `c("hr", "ctsd")`.
which_meta	Character. Specifies the analytical target for population-level inference. Choose one: • `"mean"` (default) - estimates the average across all individuals. • `"ratio"` - compares the mean between groups `"A"` and `"B"`. Requires `groups` to be specified. • `NULL` - single-individual inference. Requires `data` to be a single telemetry object rather than a list.
add_individual_variation	Logical. If `TRUE`, simulates variation by drawing movement parameters from the population distribution. This produces more realistic between-individual variability. Defaults to `FALSE`.

groups	(Optional) A named list assigning individuals to two groups, required when `which_meta = "ratio"`. Each element is a character vector of individual names (matching `data`).
	Example:
	`list(`
	`  A = c("Animal_01", "Animal_02"),`
	`  B = c("Animal_03", "Animal_04")`
	`)`

parallel	Logical. Whether to use parallel processing during model fitting. Defaults to FALSE.
.seed	(Optional) Integer. Random seed for reproducibility. If NULL (default), a seed is chosen automatically and stored in the returned object so results can be reproduced later. Only needed to be specifiedwhen reproducing Shiny app analyses in the R console.

Details

This function is designed to streamline and standardize the preparation of input data and study design parameters for simulation-based movement ecology analyses. It performs the following key steps:

- Validates that data is a non-empty list of telemetry objects.
- Fits movement models to each individual if not supplied.
- Checks supplied movement models for validity.
- Extracts parameters (tau_p, tau_v, sigma) from fitted models for use in downstream simulations.
- Consolidates all settings, parameters, and model objects into a single structured object.

By default (add_individual_variation = FALSE), all simulated animals share the same movement parameters, estimated from the population mean. Setting add_individual_variation = TRUE instead randomly draws parameters from the population distribution for each individual, which better reflects natural variability but increases uncertainty in downstream estimates.

Value

An object of class movedesign_input, accepted by all downstream functions such as md_run() and md_replicate(). Contains the validated inputs, fitted models, extracted movement parameters, and all metadata needed for study design evaluation.

See Also

md_simulate() to build inputs from directly specified parameters rather than empirical data, md_run(), md_replicate().

Other workflow_steps: md_compare(), md_replicate(), md_run(), md_simulate()

Examples

```
if(interactive()) {

  data(buffalo)
  input <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 1, unit = "month"),
    dti = list(value = 1, unit = "day"),
```

```
    set_target = "hr",
    which_meta = "mean")

 summary(input)
}
```

`md_replicate` *Replicate study design workflow and aggregate outputs*

Description

Runs the full `movedesign` workflow multiple times and aggregates outputs across independent replicates. Use this function after `md_run()` to quantify how stochasticity and design choices affect estimation performance, and to produce the robust, replicated results needed for a reliable design evaluation. Use `md_check()` afterwards to assess whether enough replicates have been run for stable inference.

Can also extend a previous run of `md_replicate()`: passing an existing `movedesign_output` object appends new replicates to the existing outputs rather than starting over.

Usage

```
md_replicate(
  obj,
  n_replicates,
  verbose = TRUE,
  trace = TRUE,
  parallel = FALSE,
  error_threshold = 0.05,
  ncores = parallel::detectCores(),
  ...
)
```

Arguments

`obj`	An object of class `movedesign_input`, as returned by `md_prepare()` or `md_simulate()`, or a `movedesign_output` object from a previous call of this function. Passing a `movedesign_output` appends `n_replicates` to the existing results.
`n_replicates`	A single positive integer. The number of independent replicates to run. Must be at least `5`. Start with a modest number (*e.g.*, `20`), then use `md_check()` to assess convergence. If convergence has not been reached, pass the output back to this function to append more replicates.
`verbose`	Logical. If `TRUE` (default), evaluates population-level inference at every *population* sample size up to `n_individuals`, saving results at each step. This shows how estimation performance changes as sample size grows. If `FALSE`, inference is run only once at the maximum sample size defined by `n_individuals` in `md_prepare()`.

`trace`	Logical. If `TRUE` (default), prints progress and timing messages to the console for each replicate. Set to `FALSE` for silent execution.
`parallel`	Logical. If `TRUE`, runs replicates in parallel. Defaults to `FALSE`. Not supported on Windows, where execution falls back to sequential automatically.
`error_threshold`	Numeric. The acceptable error threshold used when summarising estimation performance across replicates (e.g. `0.05` for 5%).
`ncores`	Integer. Number of CPU cores to use when `parallel = TRUE`. Defaults to all available cores via `parallel::detectCores()`. Ignored when `parallel = FALSE` or on Windows.
`...`	Reserved for internal use.

Details

Each replicate calls `md_run()` with a unique random seed, ensuring results are statistically independent. If the function is interrupted, it returns all results completed up to that point rather than discarding them. This makes it safe to stop a long run early and still retrieve partial results.

Parallel processing:

Setting `parallel = TRUE` can substantially reduce runtime for large replication runs.

Appending replicates:

Passing a `movedesign_output` object as `obj` adds new replicates to the existing results. This is useful when an initial run needs more replicates for stable inference without discarding completed work.

Assessing convergence:

There is no universal rule for how many replicates are sufficient. After an initial run, use `md_check()` to evaluate whether the cumulative mean of the tracked error metric has stabilised across replicates. If convergence has not been reached, pass the returned `movedesign_output` object back to `md_replicate()` to append more replicates without discarding completed work. Repeat until `md_check()` confirms convergence.

Value

An object of class `movedesign_output`, accepted by `md_check()`, `md_plot()`, and `md_plot_replicates()`.

See Also

`md_prepare()` and `md_simulate()` to build the input object. `md_run()` for a single exploratory run before committing to full replication. `md_check()` to assess whether cumulative estimation error has stabilised across replicates (the recommended criterion for deciding when enough replicates have been run). `md_plot()` and `md_plot_replicates()` to visualize outputs.

Other workflow_steps: `md_compare()`, `md_prepare()`, `md_run()`, `md_simulate()`

Examples

```
if (interactive()) {

  data(buffalo)
  input <- md_prepare(
    data = buffalo,
    models = models,
    species = "buffalo",
    n_individuals = 5,
    dur = list(value = 1, unit = "month"),
    dti = list(value = 1, unit = "day"),
    add_individual_variation = TRUE,
    grouped = FALSE,
    set_target = "hr",
    which_meta = "mean")

  output <- md_replicate(input, n_replicates = 5)
  md_check(output)

  # Append more replicates to an existing result:
  output <- md_replicate(output, n_replicates = 10)
}
```

---

md_run *Run study design workflow*

Description

The main workhorse of the movedesign workflow. Runs one full round of simulation and analyses to evaluate whether the design meets its estimation targets. #' Call this function once your design has been built using md_prepare() for empirical data, or md_simulate() for user-specified parameters.

Because a single run is subject to stochastic variation, treat outputs from md_run() as exploratory, and use md_replicate() for more robust inferences (as it aggregates results across multiple replicates).

Usage

```
md_run(design, trace = TRUE, .seeds = NULL)
```

Arguments

design	An object of class movedesign_input, as returned by md_prepare() or md_simulate().
trace	Logical. If TRUE (default), prints progress messages and elapsed time for each step. Set to FALSE for silent execution.

`.seeds` (Optional) List of integer seeds, one per individual, used to reproduce a previous run exactly. Seeds from a prior run are stored in the `$seedList` slot of the object returned by this function. Leave as `NULL` (default) for a fresh run with automatically generated seeds.

Details

Progress messages are printed by default. Every individual simulation is assigned a unique random seed, stored in `$seedList` of the returned object. Passing that list to `.seeds` in a subsequent call reproduces the run exactly. This is particularly useful when replicating a result first produced in the Shiny app.

Typical workflow:

- Prepare a study design with `md_prepare()`.
- Run all simulations and analyses with `md_run()`.
- Summarize or plot outputs from the returned object.

Value

An object of class `movedesign_processed`, accepted by downstream functions such as `md_plot_preview()`, or `md_compare_preview()`.

See Also

`md_prepare()` and `md_simulate()` to build the input object. `md_replicate()` to run the workflow multiple times and aggregate results, which is recommended over a single `md_run()` call for any final design evaluation. `md_plot_preview()` and `md_compare_preview()` to inspect or compare these preliminary outputs.

Other workflow_steps: `md_compare()`, `md_prepare()`, `md_replicate()`, `md_simulate()`

Examples

```
if(interactive()) {

input <- md_prepare(
  data = buffalo,
  models = models,
  species = "buffalo",
  n_individuals = 5,
  dur = list(value = 1, unit = "month"),
  dti = list(value = 1, unit = "day"),
  add_individual_variation = FALSE,
  set_target = "hr",
  which_meta = "mean")

output <- md_run(input)
}
```

`md_simulate` *Simulate movement data from species parameters*

Description

Simulates continuous-time movement trajectories based on user-specified movement parameters. The function is designed to support study design workflows by generating synthetic tracking data under specified the movement, sampling, and analytical assumptions.

Use this function when you do not have empirical data available. For workflows grounded in real data, use `md_prepare()` instead, which extracts movement parameters directly from fitted models.

Usage

```
md_simulate(
  n_individuals = NULL,
  tau_p,
  tau_v,
  sigma,
  dur = NULL,
  dti = NULL,
  set_target = c("hr", "ctsd"),
  which_meta = "mean",
  grouped = FALSE,
  parallel = FALSE,
  seed = NULL
)
```

Arguments

`n_individuals` Integer. Number of tracked individuals (tags) to simulate. Defines the target population sample size. When `grouped = TRUE`, this number is currently split evenly between the two groups, so it must be even.

`tau_p` Position autocorrelation timescale, corresponding to the average *home range crossing time*.

Provide a list with two elements:

- `value` - a numeric value (e.g. `6`)
- `unit` - a character string: one of `"second"`, `"minute"`, `"hour"`, `"day"`, `"month"`, or `"year"`

Example: `list(value = 6, unit = "hours")`

When `grouped = TRUE`, provide a named list with two entries, one per group:

```
list(
  A = list(value = 6,  unit = "hours"),
  B = list(value = 12, unit = "hours")
)
```

`tau_v` Velocity autocorrelation timescale, corresponding to directional persistence (how long does an animal maintains a consistent direction and speed before changing course).

Same format as `tau_p`.

`sigma` Location variance parameter. Captures the overall spatial extent of movement.

Same format as `tau_p`.

`dur` Sampling duration. A list with elements `value` (numeric) and `unit` (character).

Example: `list(value = 3, unit = "months")`

`dti` Sampling interval between relocations. A list with elements `value` (numeric) and `unit` (character).

Example: `list(value = 2, unit = "hours")`

`set_target` Character vector specifying the target metrics to be evaluated in the study design workflow. Choose one or both:

- `"hr"` - home range area
- `"ctsd"` - continuous-time speed and distance

Defaults to `c("hr", "ctsd")`.

`which_meta` Character specifying the population-level analytical target. Choose one:

- `"mean"` (default) - estimates the average value across all individuals.
- `"ratio"` - compares the mean between groups A and B. Requires `grouped = TRUE`.

`grouped` Logical. Set to `TRUE` to simulate two distinct groups (e.g. males and females) with different movement parameters. When `TRUE`, all three parameter arguments (`tau_p`, `tau_v`, `sigma`) must be named lists for both groups, and `n_individuals` must be even. Defaults to `FALSE`.

`parallel` Logical. Whether to use parallel processing during model fitting. Defaults to `FALSE`.

`seed` Optional integer. Random seed for reproducibility. If `NULL` (default), a seed is chosen automatically and stored in the returned object so results can be reproduced later.

Details

Each simulated trajectory represents a single continuously-tracked animal, generated from a continuous-time movement model with the parameters you supply. The time vector is constructed from `dur` and `dti`, and then one trajectory per individual is drawn from that model.

Simulated data are immediately passed through the full `movedesign` workflow (model fitting, aggregation, and estimation of the target metrics) so the returned object is ready for study design evaluation without further steps.

When `grouped = TRUE`, simulations are generated independently for each using their own movement parameters but currently share the same sampling parameters (`dur` and `dti`). Group structure only affects downstream inference when `which_meta = "ratio"`.

Value

An object of class `movedesign_input`. This is the standard input object for the `movedesign` workflow and can be passed directly to downstream functions such as `md_run()` or `md_replicate()`. It contains the simulated trajectories, fitted movement models, and all metadata needed for study design evaluation.

Note

Results are only as informative as the parameters you provide. Where possible, derive parameters from real tracking data using `md_prepare()`. Simulations based on arbitrary or weakly justified parametersvalues may be useful for exploration purposes, but should be interpreted with caution in any design or inference context.

See Also

`md_prepare()` to derive parameters from real tracking data.

Other workflow_steps: `md_compare()`, `md_prepare()`, `md_replicate()`, `md_run()`

Examples

```
if(interactive()) {

# Single group:
# (simulate 10 individuals over 3 months with fixes every 2 hours)

input <- md_simulate(
  n_individuals = 4,
  tau_p = list(value = 6, unit = "hours"),
  tau_v = list(value = 30, unit = "minutes"),
  sigma = list(value = 1, unit = "km^2"),
  dur = list(value = 1, unit = "month"),
  dti = list(value = 2, unit = "hours"))

# Two groups with different parameters:

input_grouped <- md_simulate(
  n_individuals = 10,
  tau_p  = list(
    A = list(value = 6,  unit = "hours"),
    B = list(value = 12, unit = "hours")
  ),
  tau_v  = list(
    A = list(value = 0.5, unit = "hours"),
    B = list(value = 1,   unit = "hours")
  ),
  sigma  = list(
    A = list(value = 1, unit = "km^2"),
    B = list(value = 2, unit = "km^2")
  ),
  dur = list(value = 3, unit = "months"),
  dti = list(value = 2, unit = "hours"),
```

```
  grouped = TRUE,
  which_meta = "ratio")

}
```

`md_stack` *Stack simulation outputs as replicates*

Description

Assigns a replicate ID to each `md_run()` output, re-runs population-level resampling for each, and aggregates inference results into a unified output. Calling `md_stack()` on a list of `n` `md_run()` outputs produces the same result as calling `md_replicate()` with `n_replicates = n`.

Use this function when the `md_run()` calls have already been made; for example, when runs were executed in parallel outside the standard workflow, or recovered after an interruption.

The distinction from `md_merge()` is important. `md_merge()` pools all inputs into one larger dataset (*e.g.*, if each run has 5 individuals, the output has `10`, and the design corresponds to a single replicate. `md_stack()` assigns each run a separate replicate ID: number of individuals does not accumulate, and population-level inference is aggregated across replicates.

Usage

```
md_stack(obj, error_threshold = 0.05, ...)
```

Arguments

`obj`	A list of `movedesign_processed` objects, each returned by `md_run()`. All objects must share the same `set_target`, `dur`, and `dti`.
`error_threshold`	Numeric. The acceptable error threshold used when summarising estimation performance across replicates (e.g. `0.05` for 5%).
`...`	Reserved for internal use.

Value

A list of class `movedesign_output`.

movmods *Table of movement processes.*

Description

Lists all continuous-time movement process models in **ctmm**. Each row is a different movement model applicable for animal movement.

Usage

```
movmods
```

Format

A `data.frame` with 5 rows and 6 variables:

name Full descriptive name of the model (e.g., "Ind. Ident. Distr. (IID)"). Used throughout `ctmm`. See reference for more details on each model and their properties.

name_short Abbreviated name, used where space is limited.

tau_p Logical. TRUE if the model includes the position autocorrelation timescale (i.e., home range crossing time).

tau_v Logical. TRUE if the model includes the velocity autocorrelation timescale (i.e., directional persistence).

hrange Logical; TRUE if the model supports range residency, meaning the animal is likely to remain within a bounded area or "home range" instead of expanding indefinitely.

pars Character string summarizing which autocorrelation parameters (e.g., tau_p, tau_v) the model estimates. Shown in HTML for documentation. ...

References

- Calabrese et al. (2016). `ctmm`: an `R` package for analyzing animal relocation data as a continuous-time stochastic process. Methods in Ecology and Evolution, 7(9), 1124-1132 doi:10.1111/2041-210X.12559.
- Silva et al. (2022). Autocorrelation-informed home range estimation: A review and practical guide. Methods in Ecology and Evolution, 13(3), 534-544 <10.1111/2041-210X.13786>.

plot.movedesign_optimized *Plot movedesign report outputs*

Description

S3 method for plotting a `movedesign_optimized` object. Returns the precomputed ggplot stored in the object.

Usage

```
## S3 method for class 'movedesign_optimized'
plot(x, ...)
```

Arguments

`x`	A `movedesign_optimized` object returned by `md_optimized()` or similar.
`...`	Unused

print.movedesign_check *Print method for* `movedesign_check` *objects*

Description

Print method for `movedesign_check` objects

Usage

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

Arguments

`x`	An object of class `movedesign_check`
`...`	Unused

print.movedesign_input

Print method for movedesign_input

Description

Print method for movedesign_input

Usage

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

Arguments

x	An object of class movedesign_input
...	Additional arguments

print.movedesign_optimized

Print method for movedesign_optimized *objects*

Description

Print a structured summary of a movedesign_optimized object produced by md_optimize(). This includes study design details, replication settings, estimation performance per target metric, and an optional convergence assessment.

Usage

```
## S3 method for class 'movedesign_optimized'
print(
  x,
  verbose = TRUE,
  m = NULL,
  ci = 0.95,
  tol = 0.05,
  n_converge = 9,
  plot = TRUE,
  pal = c("#007d80", "#A12C3B"),
  ...
)
```

Arguments

x	An object of class movedesign_optimized.
verbose	Logical. If TRUE, run md_check() and print full convergence diagnostics. Also displays a convergence plot if plot = TRUE. Defaults to FALSE.
m	Numeric (optional). Restricts results to a specific population sample size. Defaults to NULL, which uses the maximum sample size.
ci	Numeric. Confidence level for intervals (applied to narrow confidence bars and wide prediction bands). Must be between 0 and 1. Default is 0.95.
tol	Numeric. Tolerance threshold for absolute change in cumulative mean to declare convergence. Default is 0.05.
n_converge	Integer. Number of consecutive steps within tolerance required to confirm convergence. Default is 9.
plot	Logical. If TRUE, generates a convergence plot. Default is TRUE.
pal	Character vector of colors for the convergence plot, e.g. c("#007d80", "#A12C3B"). Default is c("#007d80", "#A12C3B").
...	Additional arguments (currently unused).

---

print.movedesign_output

Print method for movedesign_output

Description

Print method for movedesign_output

Usage

```
## S3 method for class 'movedesign_output'
print(
  x,
  verbose = FALSE,
  m = NULL,
  ci = 0.95,
  tol = 0.05,
  n_converge = 9,
  plot = TRUE,
  pal = c("#007d80", "#A12C3B"),
  ...
)
```

Arguments

x	An object of class movedesign_output.
verbose	Logical. If TRUE, convergence diagnostics are evaluated using md_check() and included in the output.
m	Optional integer specifying the sample size used in convergence checks.
ci	Numeric value giving the confidence level used when summarizing estimator error. Default is 0.95.
tol	Numeric tolerance used when assessing convergence. Default is 0.05.
n_converge	Integer giving the number of convergence steps to evaluate.
plot	Logical indicating whether convergence diagnostics should produce plots when verbose = TRUE. Default is TRUE.
pal	Character vector specifying colors used in convergence plots.
...	Additional arguments

---

print.movedesign_processed

Print method for movedesign_processed

Description

Print method for movedesign_processed

Usage

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

Arguments

x	An object of class movedesign_processed
...	Additional arguments

print.movedesign_report *Print method for* movedesign_report *objects*

Description

Print method for movedesign_report objects

Usage

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

Arguments

x	An object of class movedesign_report
...	Unused

read_md *Read a* movedesign *session file*

Description

Reads a .rds file exported from the **movedesign** Shiny application, validates its structure, and reconstitutes each component to its corresponding movedesign class.

The returned list always contains:

design_input A movedesign_input object: sampling design parameters as specified by the user.

design_processed A movedesign_processed object: results from a single simulation replicate.

(Optional) If multiple replicates have been run:

design_output A movedesign_output object: aggregated results across all simulation replicates.

Usage

```
read_md(filepath)
```

Arguments

filepath	Path to an .rds session file.

Value

A named list of three **movedesign** objects.

Examples

```
## Not run:
md_objects <- read_md("path/to/my_file.rds")

md_objects$design_input      # movedesign_input object
md_objects$design_processed  # movedesign_processed object
md_objects$design_output     # movedesign_output (if available)

## End(Not run)
```

`run_app` *Run movedesign R Shiny application*

Description

Run movedesign R Shiny application

Usage

```
run_app(
  onStart = NULL,
  options = list(),
  enableBookmarking = NULL,
  uiPattern = "/",
  ...
)
```

Arguments

`onStart`	A function called before the app runs. Only relevant for programmatic usage.
`options`	A named list passed to `shiny::runApp`.
`enableBookmarking`	One of `url`, `server`, or `disable`. The default, `NULL`, respects any previous call to `enableBookmarking`.
`uiPattern`	A regular expression used to match request paths. The request path must match the expression in full to be handled by the UI.
`...`	arguments to pass to golem_opts. See `?golem::get_golem_options` for more details.

Value

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

summary.movedesign_check *Summary method for* movedesign_check *objects*

Description

Summary method for movedesign_check objects

Usage

```
## S3 method for class 'movedesign_check'
summary(object, verbose = FALSE, ...)
```

Arguments

object	An object of class movedesign_check
verbose	Add interpretation text
...	Unused

summary.movedesign_input *Summary method for* movedesign_input

Description

Summary method for movedesign_input

Usage

```
## S3 method for class 'movedesign_input'
summary(object, ...)
```

Arguments

object	An object of class movedesign_input
...	Additional arguments

summary.movedesign_optimized *Summarise a study design optimization*

Description

Print a structured summary of a movedesign_optimized object produced by md_optimize(). The summary reports the study design, replication settings, estimation performance for each target metric, and a convergence assessment.

This method runs automatically when calling summary(output) on a movedesign_optimized object.

Usage

```
## S3 method for class 'movedesign_optimized'
summary(
  object,
  verbose = FALSE,
  error_threshold = NULL,
  m = NULL,
  ci = 0.95,
  tol = 0.05,
  n_converge = 9,
  plot = TRUE,
  pal = c("#007d80", "#A12C3B"),
  ...
)
```

Arguments

object	A movedesign_optimized object returned by md_replicate() or md_stack().
verbose	Logical. If TRUE, run md_check() and print the full convergence diagnostics. This can also display a convergence plot when plot = TRUE. If FALSE (default), only the convergence status is printed.
error_threshold	Numeric. Upper limit of the relative error in estimation (e.g., 0.05 for 5%) deemed acceptable by the user.
m	Numeric (Optional). If provided, restricts the results to a specific population sample size (m). Defaults to NULL, which checks up to the maximum population sample size.
ci	Confidence level for the intervals. Applied to both the narrow confidence bars and wide prediction bands. Must be between 0 and 1. Default: 0.95 (95%).
tol	Numeric. The tolerance threshold for absolute change in the cumulative mean to declare convergence. Defaults to 0.05.

n_converge	Integer. Number of consecutive steps within tolerance required to confirm convergence.
plot	Logical. If TRUE (default), generates a plot of stepwise changes in the cumulative mean, highlighting when convergence is achieved.
pal	Character vector of color(s) of the plot, such as c("#007d80", "#A12C3B")) (default).
...	Additional arguments

---

summary.movedesign_output

Summarise a study design output

Description

Print a structured summary of a movedesign_output object produced by md_replicate() or md_stack(). The summary reports the study design, replication settings, estimation performance for each target metric, and a convergence assessment.

This method runs automatically when calling summary(output) on a movedesign_output object.

Usage

```
## S3 method for class 'movedesign_output'
summary(
  object,
  verbose = FALSE,
  m = NULL,
  ci = 0.95,
  tol = 0.05,
  n_converge = 9,
  plot = TRUE,
  pal = c("#007d80", "#A12C3B"),
  ...
)
```

Arguments

object	A movedesign_output object returned by md_replicate() or md_stack().
verbose	Logical. If TRUE, run md_check() and print the full convergence diagnostics. This can also display a convergence plot when plot = TRUE. If FALSE (default), only the convergence status is printed.
m	Numeric (Optional). If provided, restricts the results to a specific population sample size (m). Defaults to NULL, which checks up to the maximum population sample size.
ci	Confidence level for the intervals. Applied to both the narrow confidence bars and wide prediction bands. Must be between 0 and 1. Default: 0.95 (95%).

tol	Numeric. The tolerance threshold for absolute change in the cumulative mean to declare convergence. Defaults to 0.05.
n_converge	Integer. Number of consecutive steps within tolerance required to confirm convergence.
plot	Logical. If TRUE (default), generates a plot of stepwise changes in the cumulative mean, highlighting when convergence is achieved.
pal	Character vector of color(s) of the plot, such as c("#007d80", "#A12C3B")) (default).
...	Additional arguments

See Also

md_replicate(), md_stack() to generate results. md_check() to inspect convergence directly. md_compare() to compare designs after convergence.

Examples

```
if(interactive()) {

data(buffalo)

input <- md_prepare(
  species = "African buffalo",
  data = buffalo,
  n_individuals = 5,
  dur = list(value = 1, unit = "month"),
  dti = list(value = 1, unit = "day"),
  set_target = "hr",
  which_meta = "mean")

output <- md_replicate(input, n_replicates = 20)

# Print standard summary:
summary(output)

# Run full convergence diagnostics:
summary(output, verbose = TRUE, tol = 0.05)

}
```

---

summary.movedesign_processed

Summary method for movedesign_processed

Description

Summary method for movedesign_processed

Usage

```
## S3 method for class 'movedesign_processed'
summary(object, ...)
```

Arguments

object	An object of class movedesign_processed
...	Additional arguments

---

summary.movedesign_report *Summary method for* movedesign_report *objects*

Description

Summary method for movedesign_report objects

Usage

```
## S3 method for class 'movedesign_report'
summary(object, ...)
```

Arguments

object	An object of class movedesign_report
...	Unused

Index

∗ **datasets**
    fixrates, 4
    movmods, 31
∗ **workflow_steps**
    md_compare, 6
    md_prepare, 20
    md_replicate, 23
    md_run, 25
    md_simulate, 27

bayestestR::ci(), *7*

ctmm::as.telemetry(), *21*

fitting_models, 3
fixrates, 4

md_check, 4
md_check(), *6, 7*, *9*, *16–18*, *23, 24*, *34, 35*, *39–41*
md_compare, 6
md_compare(), *7–9*, *22*, *24*, *26*, *29*, *41*
md_compare_preview, 8
md_compare_preview(), *6*, *26*
md_configure, 10
md_configure(), *13, 14*
md_merge, 11
md_merge(), *30*
md_optimize, 13
md_optimize(), *19*, *33*, *39*
md_plot, 15
md_plot(), *7*, *19*, *24*
md_plot_preview, 17
md_plot_preview(), *9*, *26*
md_plot_replicates, 18
md_plot_replicates(), *24*
md_prepare, *12*, 20
md_prepare(), *7*, *11*, *13, 14*, *23–27*, *29*
md_replicate, 23
md_replicate(), *5–9*, *15–19*, *22–26*, *29, 30*, *39–41*
md_run, *12*, 25
md_run(), *5–9*, *11*, *17, 18*, *22–26*, *29, 30*
md_simulate, 27
md_simulate(), *7*, *20*, *22–26*
md_stack, 30
md_stack(), *11*, *39–41*
movmods, 31

parallel::detectCores(), *24*
plot.movedesign_optimized, 32
print.movedesign_check, 32
print.movedesign_input, 33
print.movedesign_optimized, 33
print.movedesign_output, 34
print.movedesign_processed, 35
print.movedesign_report, 36

read_md, 36
run_app, 37

summary.movedesign_check, 38
summary.movedesign_input, 38
summary.movedesign_optimized, 39
summary.movedesign_output, 40
summary.movedesign_processed, 41
summary.movedesign_report, 42