Package 'Rtrack'

May 4, 2020

Type Package

Title Spatial Navigation Strategy Analysis

Version 1.0.0

Maintainer Rupert Overall <rtrack@rupertoverall.net>

URL https://rupertoverall.net/Rtrack/,
https://github.com/rupertoverall/Rtrack

BugReports https://github.com/rupertoverall/Rtrack/issues

Description A toolkit for the analysis of paths from spatial tracking experiments (such as the Morris water maze) and calculation of goal-finding strategies.
This package is centered on an approach using machine learning for path classification.

License GPL-3

Encoding UTF-8

LazyData true

Imports crayon, graphics, grDevices, Hmisc, KernSmooth, methods, openxlsx, parallel, pbapply, randomForest, raster, readxl, rgeos, rjson, sp, stats, tools, utils

RoxygenNote 7.1.0

Depends R (>= 2.10)

Suggests knitr, rmarkdown

NeedsCompilation no

Author Rupert Overall [aut, cre] (<https://orcid.org/0000-0002-3882-6073>)

Repository CRAN

Date/Publication 2020-05-04 20:10:02 UTC

R topics documented:

`calculate_metrics` *Calculation of spatial search path metrics.*

Description

Calculates a range of metrics from path coordinates.

Usage

```
calculate_metrics(path, arena)
```

Arguments

`path`	An `rtrack_path` object as returned by `read_path`.
`arena`	An `rtrack_arena` object as returned by `read_arena`.

Details

Metrics are calculated based on normalised coordinate data and are available as the `summary` element of the `rtrack_path` object. Unnormalised values (with the same units as the raw data) are also available as the `unscaled.summary` element. These can be useful for custom plots and are also the values exported by `export_results`. Extended metrics are available as separate elements of the `rtrack_path` object.

Value

An `rtrack_metrics` object containing metrics of the search path. This object is required as input for the `call_strategy` and `plot_path` functions.

See Also

`read_path`, `read_arena`, and also `read_experiment` for processing many tracks at once.

Examples

```
require(Rtrack)
track_file <- system.file("extdata", "Track_1.csv", package = "Rtrack")
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
path <- read_path(track_file, arena, track.format = "ethovision.3.csv")
metrics <- calculate_metrics(path, arena)
```

`call_mwm_strategy_garthe` *Strategy classification using the Garthe classifier.*

Description

Calculates strategies using a method based on Garthe et. al. 2009.

Usage

```
call_mwm_strategy_garthe(metrics, parameters = NULL)
```

Arguments

`metrics`	An `rtrack_metrics` object from `calculate_metrics` or a list of such objects.
`parameters`	A `data.frame` of parameters to adjust output. Currently not implemented.

Value

An `rtrack_strategies` object. The `calls` element contains the called strategy/strategies as well as several additional metrics generated by this method.

See Also

`call_strategy`.

Examples

```
require(Rtrack)
track_file <- system.file("extdata", "Track_1.csv", package = "Rtrack")
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
path <- read_path(track_file, arena, track.format = "ethovision.3.csv")
metrics <- calculate_metrics(path, arena)
strategies <- call_mwm_strategy_garthe(metrics)
# Inspect the strategy call
strategies$calls
```

call_strategy *Calculation of search strategies.*

Description

Calculates Morris water maze strategies from path metrics.

Usage

```
call_strategy(metrics, version = "mouse_rf_6")
```

Arguments

metrics	An rtrack_metrics object from calculate_metrics or a list of such objects.
version	The strategy calling model that should be used. Currently only the default mouse_rf_6 is implemented.

Details

This function implements a classifier based on a trained random forest model.

Value

An rtrack_strategies object. The calls element contains the called strategy/strategies as well as confidence scores for all possible strategies.

See Also

threshold_strategies, plot_strategies.

Examples

```
require(Rtrack)
track_file <- system.file("extdata", "Track_1.csv", package = "Rtrack")
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
path <- read_path(track_file, arena, track.format = "ethovision.3.csv")
metrics <- calculate_metrics(path, arena)
strategies <- call_strategy(metrics)
# Inspect the strategy call
strategies$calls
```

check_experiment *Check experiment data.*

Description

Checks that the experiment description is well-formed and complete.

Usage

```
check_experiment(
  filename,
  format = NA,
  interpolate = FALSE,
  project.dir = NA,
  data.dir = project.dir,
  cluster = NULL,
  author.note = "",
  verbose = FALSE
)
```

Arguments

filename	A spreadsheet file containing a description of the experiment or a JSON file containing an exported experiment archive.
format	An experiment description for reading raw data can be provided as an Excel spreadsheet ('excel'; the default) or as a comma-delimited ('csv') or tab-delimited ("tab", "tsv", "txt" or "text") text file. The value 'json' indicates that the file is an archived experiment in the JSON format (as generated by export_json). Default (NA) is to guess the format from the file extension.
interpolate	Ignored. For compatibility with read_experiment.
project.dir	A directory path specifying where the files needed for processing the experiment are stored. Ignored if format = "json".
data.dir	A directory path specifying where the raw data are stored. This is a folder root and all paths specified in the spreadsheet. Ignored if format = "json".
cluster	Ignored. For compatibility with read_experiment.
author.note	Ignored. For compatibility with read_experiment.
verbose	Ignored. For compatibility with read_experiment.

Details

Information about a full experiment can be assembled into a spreadsheet (currently Excel, CSV and tab-delimited text formats are supported) and used to process large numbers of files in one batch. This function checks the spreadsheet to make sure that it is properly formed and that all the data files referred to are present.

The function can (and ideally should) be run with the same parameters as will be used to call read_experiment, although many of the parameters are not required for the check.

The content of the spreadsheet, the presence and the content of any supporting files are also checked. Checks do not cover validity of the raw data, so it is still possible to have invalid data even if check_experiment returns TRUE (although this suggests an underlying problem with the raw data). Warning and error messages are intended to be useful and help any format issues be quickly resolved.

Value

Invisibly returns TRUE for a successful check or FALSE otherwise.

See Also

read_experiment, export_json.

Examples

```
require(Rtrack)
experiment.description <- system.file("extdata", "Minimal_experiment.xlsx",
  package = "Rtrack")
check_experiment(experiment.description)
```

---

export_json *Export experiment data to a JSON file.*

Description

Creates a representation of the experiment data in the JSON format and optionally writes this to file.

Usage

```
export_json(experiment, tracks = "all", file = NULL)
```

Arguments

experiment	An rtrack_experiment object from read_experiment.
tracks	Which tracks should be exported. Default, 'all' exports the entire experiment object. A subset of tracks can be specified using either numeric indices or a vector of track IDs following usual R standards.
file	The file to which the JSON data will be written. If NULL (the default), nothing will be written.

Details

The exported JSON file contains all the raw data and experiment metadata. The JSON archive contains exactly the same information as if reading from raw data. Calculated metrics are not exported, but can be recreated exactly.

A formal description of the JSON format can be found in the schema file at https://rupertoverall.net/Rtrack/Rtrack_schema_v1.json.

Value

This function invisibly returns the JSON data as a character string.

See Also

read_experiment to import the JSON file back into R.

Examples

```
require(Rtrack)
experiment.description <- system.file("extdata", "Minimal_experiment.xlsx",
  package = "Rtrack")
experiment <- read_experiment(experiment.description, format = "excel",
  project.dir = system.file("extdata", "", package = "Rtrack"))
tempfile <- file.path(tempdir(), "Minimal_experiment.json") # Temporary file
export_json(experiment, file = tempfile)
imported.experiment <- read_experiment(tempfile, format = "json")
# Experiments are identical except for export timestamp/notes
all.equal(experiment, imported.experiment)
identical(experiment$metrics, imported.experiment$metrics)
```

---

export_results *Export experiment results to a data.frame or file.*

Description

Binds experiment data together with analysis results and optionally writes this to file.

Usage

```
export_results(experiment, tracks = "all", file = NULL)
```

Arguments

experiment	An rtrack_experiment object from read_experiment.
tracks	Which tracks should be exported. Default, 'all', exports the entire experiment object. A subset of tracks can be specified using either numeric indices or a vector of track IDs following usual R standards.
file	The file to which the results will be written. If NULL (the default), the data will be returned as a data.frame.

Details

For convenience, the strategy calls are performed within this function so the user does not necessarily have to have run them previously. A non-thresholded set of strategies will be exported. If only a thresholded subset should be exported, then this can be achieved by performing thresholding separately and passing the rownames to this function as the parameter `tracks`.

If `file` is supplied, the file extension will be used to determine which format to save the file in. The formats ".csv", ".csv2" (see `write.table` for details of the formats), ".tsv" (tab-delimited text; can also be written as ".txt" or ".tab") and ".xlsx" (default) are currently supported. If the file extension is not in this list, the data will be written as tab-delimited text with a warning. Note that the Excel '.xlsx' format is supported, but the older '.xls' is not.

Value

A `data.frame` containing the experimental groups and factors (as supplied in the original experiment description) together with the summary metrics and strategy calls). This is returned invisibly if `file` is specified.

Examples

```
require(Rtrack)
experiment.description <- system.file("extdata", "Minimal_experiment.xlsx",
  package = "Rtrack")
experiment <- read_experiment(experiment.description, format = "excel",
  project.dir = system.file("extdata", "", package = "Rtrack"))
# The code below returns a data.frame.
# Use the parameter 'file' to write to a file instead.
export_results(experiment)
```

---

identify_track_format *Check the format of a track file.*

Description

A helper utility to determine the raw data format.

Usage

```
identify_track_format(filename = NULL)
```

Arguments

`filename`	A raw data file containing path coordinates. If this is NULL or missing, then a message is given listing all of the possible format codes.

Details

Raw data from several sources can be read in directly. A number of formats are supported, but it might not be clear which format code corresponds to your data. This function can be run on a typical file to try to guess your file format. If the format is not recognised, please visit the help page at https://rupertoverall.net/Rtrack/help.html where it is also possible to request support for new formats.

Value

The format code as a character string. This code can be used as the `track.format` parameter for `read_path` or in the `_TrackFileFormat` column in the experiment description passed to `read_experiment`.

If the track format cannot be determined, `NA` is returned.

When run without a `filename` parameter, a character vector containing all supported format codes is invisibly returned.

See Also

`read_path`, or `read_experiment` to read the track files.

Examples

```
require(Rtrack)
track_file = system.file("extdata", "Track_1.csv", package = "Rtrack")
identify_track_format(track_file)
```

---

plot_density *Plot a path density map.*

Description

Plots a density map ('heatmap') of the path.

Usage

```
plot_density(
  metrics,
  title = NULL,
  col = (grDevices::colorRampPalette(c("#FCFBFD", "#9E9AC8", "#3F007D")))(256),
  legend = TRUE,
  goal.col = "black",
  goal.lwd = 2,
  resolution = 600,
  margins = c(0, 2, 4, 2),
  ...
)
```

Arguments

metrics	An rtrack_metrics object from calculate_metrics.
title	An optional title for the plot. The default is to use the path name saved in the rtrack_metrics object.
col	Colours for the density map. These can be provided as any vector of colours. The recommended (and default) approach is to use colorRampPalette. The default colouring is a simple white-to-blue scale.
legend	Should a colour scale legend be drawn? Default is TRUE.
goal.col	The colour to plot the goal outlines. Black by default, but it may be useful to change this if a very dark colour scheme is used.
goal.lwd	The width of the lines used to plot the goals. By default this is drawn heavier to make them stand out.
resolution	The resolution of the heatmap in pixels. The default is 600 x 600.
margins	The margins of the plot (see the option mar in par). The defaults should normally not need to be changed.
...	Additional arguments passed to the plot method in the SpatialPolygons-class to modify plot details.

See Also

calculate_metrics, plot_path.

Examples

```
require(Rtrack)
track_file <- system.file("extdata", "Track_1.csv", package = "Rtrack")
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
path <- read_path(track_file, arena, track.format = "ethovision.3.csv")
metrics <- calculate_metrics(path, arena)
plot_density(metrics)
```

---

plot_path *Plot a path.*

Description

Plots the path together with a representation of the arena. These plots are useful for diagnosis of classification problems.

Usage

```
plot_path(
  metrics,
  title = NULL,
  quadrants = FALSE,
  highlight.interpolated = TRUE,
  highlight.initial = TRUE,
  margins = c(0, 2, 4, 2),
  ...
)
```

Arguments

metrics	A metrics object from calculate_metrics.
title	An optional title for the plot. The default is to use the path name saved in the metrics object.
quadrants	Should the quadrants be marked on the plot. Default is FALSE
highlight.interpolated	Should interpolated sections of the path be highlighted (in grey). Default is TRUE.
highlight.initial	Should the initial section of the path be highlighted (in red). Default is TRUE. This is the section of the path equivalent in length to the distance between the start and the goal.
margins	The margins of the plot (see the option mar in par). The defaults should normally not need to be changed.
...	Additional arguments passed to the plot method in the SpatialPolygons-class to modify plot details.

Details

The path is plotted together with the context of the arena. The three concentric zones of the arena (the wall, outer wall and annulus) are drawn in progressively lighter shades of blue. The goal is a filled circle in orange and the old goal is drawn in grey. The direct path to goal is shown as a broken orange line and the 'approach corridor' (in transparent orange) is defined as a triangle fanning out from this line by 20 degrees either side. The path itself is drawn in black with grey sections corresponding to interpolated data (if highlight.interpolated = TRUE, the parameter interpolate = TRUE was set in read_path and there were missing data points in the raw track file). The initial path (the section of the path equivalent in length to the distance between the start and the goal) is drawn in red if highlight.initial = TRUE.

See Also

calculate_metrics, and also read_experiment for processing many tracks at once.

Examples

```
require(Rtrack)
track_file <- system.file("extdata", "Track_1.csv", package = "Rtrack")
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
path <- read_path(track_file, arena, track.format = "ethovision.3.csv")
metrics <- calculate_metrics(path, arena)
plot_path(metrics)
```

---

plot_strategies *Plot water maze strategies.*

Description

Plots the strategy usage for all groups.

Usage

```
plot_strategies(
  strategies,
  experiment,
  factor = NA,
  exclude.probe = FALSE,
  boundaries = NA,
  legend = TRUE,
  screen = FALSE,
  margins = c(5, 4, 4, 8),
  ...
)
```

Arguments

strategies	The strategy calls as returned from call_strategy or similar.
experiment	The experiment object as returned from read_experiment.
factor	The factor by which the data should be grouped.
exclude.probe	Should data from probe trials be excluded (see Details).
boundaries	Where should the boundaries between arena types be drawn (see Details).
legend	Should a legend be drawn. Default is to add a legend to the plot.
screen	Should multiple plots be drawn to one page. Default is FALSE. This can be useful for advanced layout using split.screen.
margins	The margins of the plot (see the option mar in par). The defaults should usually be fine, but they can be overridden if, for example, factor names are very long.
...	Other parameters passed to segments to control the plotted lines.

Details

The strategies returned by read_experiment can be shown in a summary plot. In these plots, the fraction of subjects utilising a particular strategy is shown for each day/trial. If a factor is provided, then one plot will be made for each level of the factor. To view data for mutliple factors, they will need to be collapsed into one composite factor for plotting using this function. If probe trials were used, these can be ignored (not plotted) as the strategy use in the absence of the goal will be somewhat different. For this to work, a column named 'Probe' must be present in the experiment description spreadsheet and must contain the value 'TRUE' for each probe trial.

Boundaries are drawn (as broken vertical lines) between different arena types (for example between acquisition and goal reversal phases of a Morris water maze experiment). By default, these are added between each unique arena definition. If this is not appropriate, then this can be overridden by providing the boundaries parameter with a data.frame with two columns 'day' and 'trial'. Multiple boundaries can be defined by entering the day and trial index into rows of this table. Use boundaries = NULL to suppress boundary lines altogether.

Value

A list of strategy call information.

Examples

```
# This function relies on data too large to include in the package.
# For a worked example, please see the vignette "Rtrack MWM analysis".
```

---

plot_variable *Plot path metrics.*

Description

Plots the metrics that have been calculated from path coordinates.

Usage

```
plot_variable(
  variable,
  experiment,
  factor = NA,
  factor.colours = "auto",
  exclude.probe = FALSE,
  boundaries = NA,
  legend = TRUE,
  margins = c(5, 4, 4, 8),
  ...
)
```

Arguments

variable	The variable/metric that should be plotted. See Details for the ways to specify this.
experiment	The rtrack_experiment object as returned from read_experiment.
factor	The factor by which the data should be grouped. Each factor level will be plotted as a separate series. If not specified, all values are plotted together in one series.
factor.colours	A colour to be used for each factor level. If not specified, colours will be automatically generated. The vector of colours is returned to allow additional plot customisation.
exclude.probe	Should data from probe trials be excluded (see Details).
boundaries	Where should the boundaries between arena types be drawn (see Details).
legend	Should a legend be added. Default is TRUE.
margins	The margins of the plot (see the option mar in par). The defaults should usually be fine, but they can be overridden if, for example, factor names are very long.
...	Other parameters passed to segments to control the plotted lines.

Details

Many of the summary metrics (as returned in the summary component of the calculate_metrics output are useful for analysis in their own right. These can be plotted as mean values over each trial with standard error bars. If a factor is provided, then one data series will be plotted for each level of the factor. To view data for mutliple factors, they will need to be collapsed into one composite factor for plotting using this function. If probe trials were used, then 'latency to goal' and several other variables do not make much sense, so the data for the probe trials can be suppressed. For this to work, a column named 'Probe' must be present in the experiment description spreadsheet and must contain the value 'TRUE' for each probe trial.

The variable parameter can either be specified as the name of one of the summary metrics, the name of one of the columns from the experiment description, or as a numeric vector. In the latter case, the numeric vector must be the same length as the number of tracks in the experiment.

Boundaries are drawn (as broken vertical lines) between different arena types (for example between acquisition and goal reversal phases of a Morris water maze experiment). By default, these are added between each unique arena definition. If this is not appropriate, then this can be overridden by providing the boundaries parameter with a data.frame with two columns 'day' and 'trial'. Multiple boundaries can be defined by entering the day and trial index into rows of this table. Use boundaries = NULL to suppress boundary lines altogether.

Value

A named vector of colours used for each factor level.

Examples

```
# This function relies on data too large to include in the package.
# For a worked example, please see the vignette "Rtrack MWM analysis".
```

read_arena *Read an arena description.*

Description

The user will normally not need to call this function directly. Use read_experiment instead.

Usage

```
read_arena(filename, description = NULL)
```

Arguments

filename	A file specifying the arena.
description	A data.frame containing parameters specifying the arena. If supplied, the filename argument will be ignored. This is intended for internal use only and can be ignored.

Details

Every path must be accompanied by a description of the 'arena'. This description includes arena size, goal coordinates etc. and is unique for every combination of these (i.e. a different arena description file is required for goal reversal trials).

The type parameter specifies the type of arena used. Currently only 'mwm' (for Morris water maze) is supported but other types will be added in the future. For the water maze, this package currently assumes a circular pool and circular platforms (the commonly-used square platforms are approximated by a circle of a diameter equal to the width of the square. This is because the rotational orientation of square patforms is seldom recorded (the behaviour of the package regarding this detail may be changed in future versions).

This function does not need to be explicitly called if read_experiment is being used (in that case, specify the arena file names in the column "_Arena").

Quadrants are defined such that the goal is centred around the north quadrant. Note that this means that the quadrant assignment will change in the case of a goal reversal experiment. This simplifies the experiment set-up considerably without imposing restrictions on more complex (e.g. multiple reversal) study designs.

Value

An rtrack_arena object containing a representation of the arena, which can be passed to read_path.

See Also

read_path, and also read_experiment for processing many tracks at once.

Examples

```
require(Rtrack)
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
```

`read_experiment` *Read experiment data.*

Description

Reads a spreadsheet containing a description of all the files required for an experiment to allow batch execution.

Usage

```
read_experiment(
  filename,
  format = NA,
  interpolate = FALSE,
  project.dir = NA,
  data.dir = project.dir,
  cluster = NULL,
  author.note = "",
  verbose = FALSE
)
```

Arguments

`filename`	A spreadsheet file containing a description of the experiment or a JSON file containing an exported experiment.
`format`	An experiment description for reading raw data can be provided as an Excel spreadsheet ('excel') or as a comma-delimited ('csv') or tab-delimited ('tab', 'tsv', 'txt' or 'text') text file. The value 'json' indicates that the file is an archived experiment in the JSON format (as generated by `export_json`). Default (`NA`) is to guess the format from the file extension.
`interpolate`	This is passed to the `read_path` function and specifies whether missing data points will be interpolated when reading raw swim path data. Default is `FALSE`.
`project.dir`	A directory path specifying where the files needed for processing the experiment are stored. Default (`NA`) means the project files are in the same directory as the experiment description (specified by `filename`). Ignored if `format = "json"`.
`data.dir`	A directory path specifying where the raw data are stored. All paths specified in the experiment description spreadsheet are interpreted as being relative to the `data.dir` directory. Default is the same directory as `project.dir`. Ignored if `format = "json"`.

cluster	A cluster object as generated by makeCluster or similar.
author.note	Optional text describing the experiment. This might be useful if the data is to be published or otherwise shared. Appropriate information might be author names and a link to a publication or website.
verbose	Should feedback be printed to the console. This is only useful for debugging and takes a little longer to run. Default is FALSE.

Details

Information about a full experiment can be assembled into a spreadsheet (currently Excel and CSV formats are supported) and used to process large numbers of files in one batch. The project directory (project.dir) is where the arena description files are found. This will typically be the same place as the experiment description file (and is set to be this by default). This does not need to be the same as the current working directory. An optional data directory (data.dir) can also be specified separately allowing the storage-intensive raw data to be kept in a different location (for example on a remote server). Together, these options allow for flexibility in managing your raw data storage. Individual tracks are associated with their raw data file, experimental group metadata, an arena and any other parameters that the strategy-calling methods require. Required columns are "_TrackID", "_TargetID", "_Day", "_Trial", "_Arena" "_TrackFile" and "_TrackFileFormat" (note the leading underscore "_"). Any additional columns (without a leading underscore) will be interpreted as user-defined factors or other metadata and will be passed on to the final analysis objects and thus be available for statistical analysis.

For details on how interpolation is performed (if interpolate is set to TRUE), see the documentation for read_path.

For larger experiments, a computing cluster (using the parallel package) can be specified, which will be passed to analysis functions and allow the track analysis to be performed in parallel.

Value

An rtrack_experiment object containing a complete description of the experiment.

See Also

read_path, read_arena, identify_track_format to identify the format of your raw track files, and check_experiment.

Examples

```
require(Rtrack)
experiment.description = system.file("extdata", "Minimal_experiment.xlsx",
  package = "Rtrack")
experiment = read_experiment(experiment.description)
```

read_path *Read path coordinates from raw data formats.*

Description

The user will normally not need to call this directly. Use `read_experiment` instead.

Usage

```
read_path(
  filename,
  arena,
  id = NULL,
  track.format = "none",
  track.index = NULL,
  interpolate = FALSE,
  time.bounds = c(NA, NA)
)
```

Arguments

`filename`	A raw data file containing path coordinates. See details for supported formats.
`arena`	The `arena` object associated with this track. This is required to calibrate the track coordinates to the coordinate space of the arena.
`id`	An optional name for the experiment. Default is to generate this from the filename provided.
`track.format`	The format of the raw file.
`track.index`	Only for formats where multiple tracks are stored in one file (ignored otherwise). This parameter indicates which section of the file corresponds to the track to be read. The exact usage depends on the format being read.
`interpolate`	Should missing data points be interpolated. Default is `FALSE`. Interpolation is performed at evenly-spaced intervals after removing outliers.
`time.bounds`	A vector of length 2 specifying the bounds on the measurement times (see Details).

Details

Raw data from several sources can be read in directly. The formats currently supported are 'ethovision.xt.excel' (for swim paths exported from the latest Ethovision software), 'ethovision.3.csv' (for data exported from the older Ethovision version 3) and 'raw.csv'. The 'raw.csv' format is a simple comma-delimited text file containing three columns 'Time', 'X' and 'Y'. The timestamp values in 'Time' should be in seconds from the start of the trial recording and coordinates should be in real-world units (e.g. cm, in).

If `interpolate` is set to `TRUE`, then the raw data will be cleaned to remove outlier points and ensure that time points are evenly spaced. The following procedures are used: 1. any points with missing

coordinate data are removed; 2. any points lying outside the arena are removed: 3. any points with excessive inter-point distances (outliers) are removed by first removing points that are more than 1 SD from the mean distance, then recalculating the mean and SD and repeating this step - this is typically sufficient to remove noise from video tracking (such as reflections from a water maze pool); 4. new time intervals are calculated from the first non-missing data point to the last using the sampling rate of the raw data; 5. interpolation of x and y values is performed at the new time points using the 'constant' interpolation method from `approx`.

The raw path recordings can be truncated if necessary by specifying the `time.bounds` parameter. This is a vector of length 2 containing the earliest and latest time points that should be retained for analysis (any points outside these bounds will be discarded). A value of `NA` indicates that the path should not be truncated at that end (default is `c(NA,NA)` meaning that the path will extend to the start and end of the recorded values). The units used must match the time units in the track files. This option should not normally need to be set, but may be useful if data acquisition begins before, or ends after, the actual experimental trial (e.g. if recording was running during entry and exit from the arena).

Value

An `rtrack_path` object containing the extracted swim path. This is a list comprised of the components `raw.t` (timestamp), `raw.x` (x coordinates), `raw.y` (y coordinates),`t`, `x` and `y` (normalised, cleaned and possibly interpolated coordinates).

See Also

`read_arena`, `identify_track_format` to identify the format code for your raw data, and also `read_experiment` for processing many tracks at once.

Examples

```
require(Rtrack)
track_file <- system.file("extdata", "Track_1.csv", package = "Rtrack")
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
path <- read_path(track_file, arena, track.format = "ethovision.3.csv")
```

`Rtrack` *Analysis of spatial tracking data and strategy classification.*

Description

A toolkit for the analysis of paths from spatial tracking experiments (such as the Morris water maze) and calculation of goal-finding strategies. This package is centered on an approach using machine learning for path classification.

Functions provided

calculate_metrics

call_mwm_strategy_garthe

call_strategy

check_experiment

export_json

export_results

identify_track_format

plot_density

plot_path

plot_strategies

plot_variable

read_arena

read_experiment

read_path

threshold_strategies

---

threshold_strategies *Subset an rtrack_strategies object.*

Description

Subsets strategy calls based on a threshold.

Usage

```
threshold_strategies(strategies, threshold = NULL)
```

Arguments

strategies	An rtrack_strategies object as generated by call_strategy.
threshold	A numeric value between 0 and 1 or a logical value depending on the strategy-calling method used (see Details).

Details

For strategy-calling algorithms yielding a confidence score (such as call_strategy), a value between 0 and 1 will return a new rtrack_strategies object only including calls with a confidence score above the given threshold. For strategy-calling algorithms that include a '0' or 'unknown' strategy, (such as call_mwm_strategy_garthe), TRUE will remove these unknown strategies while FALSE will return an unchanged rtrack_strategies object.

Value

An `rtrack_strategies` object including only above-threshold calls. In addition, the component `thresholded` is set to `TRUE` if thresholding was performed.

Examples

```
require(Rtrack)
track_file <- system.file("extdata", "Track_1.csv", package = "Rtrack")
arena_description <- system.file("extdata", "Arena_SW.txt", package = "Rtrack")
arena <- read_arena(arena_description)
path <- read_path(track_file, arena, track.format = "ethovision.3.csv")
metrics <- calculate_metrics(path, arena)
strategies <- call_strategy(metrics)
# Inspect the strategy call (minimal experiment only has one track)
strategies$calls
# Thresholding at 0.4 will retain the track (confidence = 0.58)
strategies = threshold_strategies(strategies, threshold = 0.4)
strategies$calls
# Thresholding at 0.6 will discard the track, still returning an (empty) rtrack_strategies object
strategies = threshold_strategies(strategies, threshold = 0.6)
strategies$calls
```

Index