Package ‘mousetRajectory’

September 8, 2023

Type Package

Title Mouse Trajectory Analyses for Behavioural Scientists

Version 0.2.1

Description Helping psychologists and other behavioural scientists to analyze mouse movement (and other 2-D trajectory) data. Bundles together several functions that compute spatial measures (e.g., maximum absolute deviation, area under the curve, sample entropy) or provide a shorthand for procedures that are frequently used (e.g., time normalization, linear interpolation, extracting initiation and movement times). For more information on these dependent measures, see Wirth et al. (2020) <doi:10.3758/s13428-020-01409-0>.

License GPL (>= 3)

URL https://github.com/mc-schaaf/mousetRajectory,
https://mc-schaaf.github.io/mousetRajectory/

BugReports https://github.com/mc-schaaf/mousetRajectory/issues

Depends R (>= 3.5.0)

Imports lifecycle, methods, signal (>= 0.7), utils

Suggests dplyr, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), V8

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.2.3

NeedsCompilation no

Author Roland Pfister [aut, cre, cph] (<https://orcid.org/0000-0002-4429-1052>),
Solveig Tonn [aut] (<https://orcid.org/0000-0001-5254-8391>),
Moritz Schaaf [aut] (<https://orcid.org/0000-0002-9959-2928>),
Robert Wirth [aut] (<https://orcid.org/0000-0001-8446-1880>)

Maintainer Roland Pfister <mail@roland-pfister.net>

Repository CRAN

Date/Publication 2023-09-08 06:50:02 UTC

R topics documented:

`auc` *Area Under the Curve*

Description

Computes the (signed) Area Under the Curve (AUC) of a path, defined by vectors of x and y coordinates, as compared to an ideal line passing through the start and end points.

Usage

```
auc(x_vector, y_vector, x_start, y_start, x_end, y_end, geometric = FALSE)
```

Arguments

`x_vector`	x-coordinates of the executed path.
`y_vector`	y-coordinates of the executed path.
`x_start`	x-coordinate of the start point of the ideal line. Defaults to the first value in `x_vector`.
`y_start`	y-coordinate of the start point of the ideal line. Defaults to the first value in `y_vector`.
`x_end`	x-coordinate of the end point of the ideal line. Defaults to the last value in `x_vector`.
`y_end`	y-coordinate of the end point of the ideal line. Defaults to the last value in `y_vector`.

geometric — Whether the sign of areas that stem from a movement in the reverse direction of the ideal line should be reversed. Defaults to `FALSE`, indicating an time-based instead of geometric interpretation. Only impacts the AUC if the trajectory is not monotonically increasing relative to the ideal line.

Details

The ideal line is a line, not a line segment, i.e., it has infinite length. The supplied vectors are assumed to be ordered by time. Counterclockwise deviations from the ideal line are considered positive, clockwise deviations as negative for the computation of the AUC. Thus, negative AUCs are possible.

Value

AUC as single number (-Inf to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
x_vals <- c(0, 0, 0, 1, 2)
y_vals <- c(0, 1, 2, 2, 2)
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
auc(x_vals, y_vals) # counterclockwise deviation: positive

x_vals <- c(0, 1, 2, 2, 2)
y_vals <- c(0, 0, 0, 1, 2)
auc(x_vals, y_vals) # clockwise deviation: negative
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
x_vals <- -x_vals
auc(x_vals, y_vals) # now it is counterclockwise again

x_vals <- c(0, 0, 1, 2, 2)
y_vals <- c(0, 1, 1, 1, 2)
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
auc(x_vals, y_vals) # might create small rounding errors; this should be 0
all.equal(0, auc(x_vals, y_vals)) # indeed interpreted by R as basically 0

x_vals <- c(0, 1, 2, 1)
y_vals <- c(0, 1, 1, 0)
plot(x_vals, y_vals, type = "l")
lines(c(0, 1), c(0, 0), lty = "dashed", lwd = 2) # ideal
auc(x_vals, y_vals)
auc(x_vals, y_vals, geometric = TRUE) # note the difference
```

curvature *Curvature*

Description

Computes the curvature of a path, defined by vectors of x and y coordinates, as compared to an ideal path, as defined by the start and end points of the path.

Usage

```
curvature(x_vector, y_vector)
```

Arguments

x_vector	x-coordinates of the executed path.
y_vector	y-coordinates of the executed path.

Details

The supplied vectors are assumed to be ordered by time.

Value

Single number indicating the curvature (1 to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
x_vals <- c(0, 0, 0, 1, 2)
y_vals <- c(0, 1, 2, 2, 2)
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
curvature(x_vals, y_vals)

x_vals <- c(0, 1, 2, 2, 2)
y_vals <- c(0, 0, 0, 1, 2)
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
curvature(x_vals, y_vals)

x_vals <- c(0, 0, 1, 2, 2)
y_vals <- c(0, 1, 1, 1, 2)
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
```

```
curvature(x_vals, y_vals)
```

`direction_changes` *xFlips*

Description

Checks how often a number sequence changes from decreasing monotonically to increasing monotonically (or vice versa).

Usage

```
direction_changes(numeric_vector)
```

Arguments

`numeric_vector` Numbers, ordered by their time of appearance.

Details

The supplied vectors are assumed to be ordered by time. Values do not have to be strictly monotonically in-/decreasing. I.e., `c(0, 1, 1, 2)` would return 0, as $x_n >= x_n - 1$ is satisfied for $2 <= n <= length(c(0, 1, 1, 2))$.

Value

Single number indicating how often `numeric_vector` changes direction (0 to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
direction_changes(c(0, 1, 1, 2))
direction_changes(c(0, 1, 1, 0))
direction_changes(c(0, 1, 0, 1))
```

`index_max_acceleration` *Time point of maximum acceleration*

Description

Computes the index of the peak acceleration of a trajectory, defined by vectors of x and y coordinates, and assumed to be equidistant in time.

Usage

```
index_max_acceleration(x_vector, y_vector, absolute = FALSE)
```

Arguments

`x_vector`	x-coordinates of the executed path.
`y_vector`	y-coordinates of the executed path.
`absolute`	Should negative accelerations (i.e., deceleration) be included? Defaults to `FALSE`.

Details

The supplied vectors are assumed to be ordered by time with equal time differences.

Value

Single number indicating the index of peak acceleration (1 to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
x_vals <- c(0, 1, 2, 3, 6, 10, 12, 14, 15)
y_vals <- c(0, 0, 0, 0, 0, 0, 0, 0, 0)
index_max_acceleration(x_vals, y_vals)
# acceleration maximal between x_vals[4] and x_vals[5]
```

index_max_velocity *Time point of maximum velocity*

Description

Computes the index of the peak velocity of a trajectory, defined by vectors of x and y coordinates, and assumed to be equidistant in time.

Usage

```
index_max_velocity(x_vector, y_vector)
```

Arguments

x_vector	x-coordinates of the executed path.
y_vector	y-coordinates of the executed path.

Details

The supplied vectors are assumed to be ordered by time with equal time differences.

Value

Single number indicating the index of peak velocity (1 to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
x_vals <- c(0, 1, 2, 3, 6, 10, 12, 14, 15)
y_vals <- c(0, 0, 0, 0, 0, 0, 0, 0, 0)
index_max_velocity(x_vals, y_vals)
# velocity maximal between x_vals[5] and x_vals[6]

numbers <- seq(-(3 / 4) * pi, (3 / 4) * pi, by = 0.001)
y_vector <- sin(numbers)
plot(numbers, y_vector)
index_max_velocity(rep(0, length(numbers)), y_vector)
abline(v = numbers[index_max_velocity(rep(0, length(numbers)), y_vector)])
which.max(cos(numbers)) # first derivative of sin, max at 0 degrees
```

interp2 — *Interpolation aka Time-Normalization*

Description

Convenient wrapper to `signal::interp1()` for linear interpolation. Assumes that you want interpolated values of `xy_old` at `n_xy_new` equidistant data points.

Usage

```
interp2(time_old, xy_old, n_xy_new = 101)
```

Arguments

`time_old`	Timestamps of the `xy_old` coordinates.
`xy_old`	To-be normalized x or y coordinates.
`n_xy_new`	Number of equidistant timepoints that should be generated. Defaults to 101.

Value

Vector of length `n_xy_new` with interpolated x or y values.

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
plot(interp2(0:10, (0:10)^2))
```

is_monotonic — *Test if vector is monotonically in-/decreasing*

Description

Checks if a `numeric_vector` is monotonically in-/decreasing. In particular, it always a good idea to check the time stamps of trajectory data and verify that the logging worked properly.

Usage

```
is_monotonic(numeric_vector, decreasing = FALSE, strict = TRUE, warn = TRUE)
```

Arguments

`numeric_vector`	Number sequence to-be checked.
`decreasing`	Should the `numeric_vector` be increasing or decreasing? Defaults to `FALSE`.
`strict`	Must the values in-/decrease *strictly*? Defaults to `TRUE`, indicating that a strict, not a weak definition of monotony is applied.
`warn`	Will a warning be issued if the `numeric_vector` is not monotonic? Defaults to `TRUE`.

Details

All objects of length 0 or 1 are monotonic. Data with missing values will not pass the check.

Value

A length-one logical, indicating whether the vector is monotonic.

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
is_monotonic(c(1, 2, 3, 4), warn = FALSE)
is_monotonic(c(1, 2, 2, 3), warn = FALSE)
is_monotonic(c(1, 2, 2, 3), strict = FALSE, warn = FALSE)
is_monotonic(c(4, 0, -1, -1, -5),
  decreasing = TRUE,
  strict = FALSE, warn = FALSE
)
```

`is_monotonic_along_ideal`

Test if vector is monotonically increasing along the ideal trajectory

---

Description

[Experimental] Checks if a trajectory, defined by vectors of x and y coordinates, is monotonically increasing relative to an ideal line passing through the start and end points.

Usage

```
is_monotonic_along_ideal(
  x_vector,
  y_vector,
  x_start,
  y_start,
  x_end,
  y_end,
  strict = TRUE,
  warn = TRUE
)
```

Arguments

`x_vector`	x-coordinates of the executed path.
`y_vector`	y-coordinates of the executed path.
`x_start`	x-coordinate of the start point of the ideal line. Defaults to the first value in `x_vector`.
`y_start`	y-coordinate of the start point of the ideal line. Defaults to the first value in `y_vector`.
`x_end`	x-coordinate of the end point of the ideal line. Defaults to the last value in `x_vector`.
`y_end`	y-coordinate of the end point of the ideal line. Defaults to the last value in `y_vector`.
`strict`	Must the values increase *strictly*? Defaults to `FALSE`, indicating that a weak, not a strict definition of monotony is applied.
`warn`	Will a warning be issued if the trajectory is not monotonic (relative to the ideal line)? Defaults to `TRUE`.

Details

Computes the orthogonal projection of the trajectory points onto the ideal line and checks whether the distances of this projection to the start point are monotonic. All objects of length 0 or 1 are monotonic. Data with missing values will not pass the check.

Value

A length-one logical, indicating whether the trajectory is monotonic.

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
# common use-case: exclude movements that miss the target area and to go back
# movement 1:
x_vals1 <- c(0, 0.95, 1)
y_vals1 <- c(0, 1.3, 1)
# movement 2:
x_vals2 <- y_vals1
y_vals2 <- x_vals1
# movement 3:
x_vals3 <- c(0, -0.1, 0.5, 1)
y_vals3 <- c(0, 0.5, 0, 1)

# note that the first two movements are symmetric to the ideal line:
plot(x_vals1, y_vals1, type = "l", xlim = c(-0.1, 1.3), ylim = c(-0.1, 1.3))
lines(x_vals2, y_vals2, type = "l")
lines(x_vals3, y_vals3, type = "l")
lines(c(0, 1), c(0, 1), lty = "dashed", lwd = 2) # ideal
is_monotonic_along_ideal(x_vals1, y_vals1, warn = FALSE)
is_monotonic_along_ideal(x_vals2, y_vals2, warn = FALSE)
is_monotonic_along_ideal(x_vals3, y_vals3, warn = FALSE)
# Note that the third movement is regarded as monotonic although both
# x and y coordinates are not.
# In contrast, excluding movements based on monotony of the y-coordinate
# would exclude the first and third movement:
is_monotonic(y_vals1, warn = FALSE)
is_monotonic(y_vals2, warn = FALSE)
is_monotonic(y_vals3, warn = FALSE)

# Also works if movements go into negative direction:
# movement 1:
x_vals1 <- c(0, -0.95, -1)
y_vals1 <- c(0, 1.3, 1)
# movement 3:
x_vals3 <- c(0, 0.1, -0.5, -1)
y_vals3 <- c(0, 0.5, 0, 1)

plot(x_vals1, y_vals1, type = "l", xlim = c(-1.3, 0.1), ylim = c(-0.1, 1.3))
lines(x_vals3, y_vals3, type = "l")
lines(-c(0, 1), c(0, 1), lty = "dashed", lwd = 2) # ideal
is_monotonic_along_ideal(x_vals1, y_vals1, warn = FALSE)
is_monotonic_along_ideal(x_vals3, y_vals3, warn = FALSE)
```

---

max_ad *(signed) Maximum Absolute Deviation*

Description

Computes the (signed) Maximum Absolute Deviation (MAD) of a path, defined by vectors of x and y coordinates, as compared to an ideal line passing through the start and end points.

Usage

```
max_ad(x_vector, y_vector, x_start, y_start, x_end, y_end)
```

Arguments

x_vector	x-coordinates of the executed path.
y_vector	y-coordinates of the executed path.
x_start	x-coordinate of the start point of the ideal line. Defaults to the first value in x_vector.
y_start	y-coordinate of the start point of the ideal line. Defaults to the first value in y_vector.
x_end	x-coordinate of the end point of the ideal line. Defaults to the last value in x_vector.
y_end	y-coordinate of the end point of the ideal line. Defaults to the last value in y_vector.

Details

The ideal line is a line, not a line segment, i.e., it has infinite length. The supplied vectors are assumed to be ordered by time. Counterclockwise deviations from the ideal line are considered positive, clockwise deviations as negative for the computation of the MAD. Thus, negative MADs are possible. If more than one value is considered maximal, the first maximal value is returned.

Value

(signed) MAD as single number (-Inf to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
x_vals <- c(0, 0, 0, 1, 2)
y_vals <- c(0, 1, 2, 2, 2)
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
max_ad(x_vals, y_vals) # counterclockwise deviation: positive

x_vals <- c(0, 1, 2, 2, 2)
y_vals <- c(0, 0, 0, 1, 2)
plot(x_vals, y_vals, type = "l")
lines(c(0, 2), c(0, 2), lty = "dashed", lwd = 2) # ideal
max_ad(x_vals, y_vals) # clockwise deviation: negative
x_vals <- -x_vals
max_ad(x_vals, y_vals) # now it is counterclockwise again
```

```
x_vals <- c(0, 0, 1, 2, 3, 6, 3)
y_vals <- c(0, 2, 2, 2, 2, 1, 0)
plot(x_vals, y_vals, type = "l")
lines(c(0, 3), c(0, 0), lty = "dashed", lwd = 2) # ideal
max_ad(x_vals, y_vals) # the ideal trajectory has infinite length

x_vals <- c(0, 1, 2, 3)
y_vals <- c(0, 1, -1, 0)
plot(x_vals, y_vals, type = "l")
lines(x_vals, -y_vals, col = "red")
lines(c(0, 3), c(0, 0), lty = "dashed", lwd = 2) # ideal
max_ad(x_vals, y_vals)
max_ad(x_vals, -y_vals) # the "first" maximal value is returned
```

`point_crosses` *Number of times a point is crossed*

Description

Checks how often a number (`relevant_point`) is being crossed by an number sequence (`numeric_vector`).

Usage

```
point_crosses(numeric_vector, relevant_point = 0)
```

Arguments

`numeric_vector` Numbers, ordered by their time of appearance.

`relevant_point` Number which has to be crossed.

Details

The supplied vectors are assumed to be ordered by time.

Value

Number of times that `numeric_vector` crosses the `relevant_point` (0 to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
x_vals <- c(-1, 1, -1, 1, -1, 1)
point_crosses(x_vals, 0)
point_crosses(x_vals, 1)
point_crosses(x_vals, -1)
```

---

sampen *Sample entropy*

Description

Computes the sample entropy (sampen), as given by Richman & Moorman (2000), doi:10.1152/ajpheart.2000.278.6.H2039.

Usage

```
sampen(
  timeseries_array,
  dimensions = 2,
  tolerance = 0.2,
  standardize = TRUE,
  use_diff = FALSE
)
```

Arguments

`timeseries_array`	Array of numbers over which the sampen is to be computed.
`dimensions`	Number of embedding dimensions for which to compute the sampen. Sometimes also called "template length".
`tolerance`	Tolerance for the comparisons of two number sequences.
`standardize`	Whether to standardize the timeseries_array.
`use_diff`	Whether to use the differences between adjacent points.

Details

As suggested by Richman & Moorman (2000), doi:10.1152/ajpheart.2000.278.6.H2039, the last possible vector of length `dimensions` is not considered because it has no corresponding vector of length `dimensions + 1`, ensuring a sampen estimation with a low bias introduced by the length of the `timeseries_array`. The function was deliberately implemented in R with C-style code. While this makes the function rather slow for large `timeseries_array`s, it enables maximal transparency. For an overview over faster sampen functions in R that, however, are distributed in binary or need source compilation, see Chen et al. (2019), doi:10.1093/biomethods/bpz016.

Value

Single number indicating the sampen for the given parameters (0 to +Inf).

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
x_vals <- rep(c(0, 0, 0, 0, 0, 1), 20)
sampen(x_vals, dimensions = 1, tolerance = 1 / 2, standardize = FALSE)
sampen(x_vals, dimensions = 3, tolerance = 1 / 2, standardize = FALSE)
sampen(x_vals,
  dimensions = 3, tolerance = 1 / 2, standardize = FALSE,
  use_diff = TRUE
)
sampen(x_vals, dimensions = 3, tolerance = 1, standardize = FALSE)
```

---

starting_angle *Starting angle*

Description

Computes the angle (in degrees) between a line, defined by two points with coordinates (`x0`, `y0`) and (`x1`, `y1`), and the specified axis.

Usage

```
starting_angle(x0, x1, y0, y1, swap_x_y = TRUE)
```

Arguments

`x0`	x-value of the first point.
`x1`	x-value of the second point.
`y0`	y-value of the first point.
`y1`	y-value of the second point.
`swap_x_y`	Whether to compute the angle relative to the x or y axis. Defaults to `TRUE`, indicating that the angle is relative to the y axis.

Details

If the angle is computed relative to the x axis, counterclockwise changes are counted as positive. If the angle is computed relative to the y axis, clockwise changes are counted as positive.

Value

Angle in degrees with $-180 <= angle <= 180$.

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
# Note that not the mathematical definition of angle is used by default:
starting_angle(0, 1, 0, 0)
starting_angle(0, 1, 0, 0, swap_x_y = FALSE)
# angles are clockwise and relative to the y-axis.

# Note that return values are in the range [-180, 180], not [0, 360]:
starting_angle(0, -1, 0, -1)
starting_angle(0, 1, 0, -1, swap_x_y = FALSE)
```

`time_circle_entered` *Completion Time*

Description

[Experimental] Checks when the specified circle was first entered by a trajectory.

Usage

```
time_circle_entered(
  x_vector,
  y_vector,
  t_vector,
  x_mid = 0,
  y_mid = 0,
  radius = 1,
  include_radius = TRUE,
  warn = TRUE
)
```

Arguments

`x_vector`	x-coordinates of the executed path.
`y_vector`	y-coordinates of the executed path.
`t_vector`	Timestamps of the executed trajectory.
`x_mid`	x-coordinate of the center of the circle.

`y_mid`	y-coordinate of the center of the circle.
`radius`	radius of the center of the circle.
`include_radius`	Whether points lying exactly on the radius should be included in the circle. Defaults to `TRUE`.
`warn`	whether a warning should be thrown if the first entry of t_vector is returned. Defaults to `TRUE`.

Value

Value of `t_vector` at the first time at which the trajectory is in the circle.

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
time_circle_entered(0:10, rep(0, 11), 0:10,
  x_mid = 10, y_mid = 0, radius = 1
)
time_circle_entered(0:10, rep(0, 11), 0:10,
  x_mid = 10, y_mid = 0, radius = 1,
  include_radius = FALSE
)
```

`time_circle_left` *Initiation Time*

Description

[Experimental] Checks when the specified circle was first left by a trajectory.

Usage

```
time_circle_left(
  x_vector,
  y_vector,
  t_vector,
  x_mid = 0,
  y_mid = 0,
  radius = 1,
  include_radius = TRUE,
  warn = TRUE
)
```

Arguments

x_vector	x-coordinates of the executed path.
y_vector	y-coordinates of the executed path.
t_vector	Timestamps of the executed trajectory.
x_mid	x-coordinate of the center of the circle.
y_mid	y-coordinate of the center of the circle.
radius	radius of the center of the circle.
include_radius	Whether points lying exactly on the radius should be included in the circle. Defaults to TRUE.
warn	whether a warning should be thrown if the first entry of t_vector is returned. Defaults to TRUE.

Value

Value of t_vector at the first time at which the trajectory is out of the circle.

References

Wirth, R., Foerster, A., Kunde, W., & Pfister, R. (2020). Design choices: Empirical recommendations for designing two-dimensional finger tracking experiments. Behavior Research Methods, 52, 2394 - 2416. doi:10.3758/s13428020014090

Examples

```
time_circle_left(0:10, rep(0, 11), 0:10)
time_circle_left(0:10, rep(0, 11), 0:10, include_radius = FALSE)
```

Index