Package: triangulation (via r-universe)

September 18, 2024

Type Package

Title Determine Position of Observer

Version 0.5.0

Date 2016-10-22

Description Measuring angles between points in a landscape is much easier than measuring distances. When the location of three points is known the position of the observer can be determined based solely on the angles between these points as seen by the observer. This task (known as triangulation) however requires onerous calculations - these calculations are automated by this package.

License LGPL

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation no

Author Mathias Milfeldt [aut, cre]

Maintainer Mathias Milfeldt <mathias@milfeldt.dk>

Repository CRAN

Date/Publication 2016-10-29 13:47:55

Contents

`determine_angles` *Determine angles as seen by observer*

Description

Determine the angles (between three known points) as seen by an observer with a known position.

Usage

```
determine_angles(A, B, C, observer_position = c(0, 0), output_plot = TRUE,
  lines_in_plot = TRUE, angles_in_plot = TRUE, decimals_in_plot = 2)
```

Arguments

`A`	A point defined by a vector containing an x- and an y-coordinate
`B`	A point defined by a vector containing an x- and an y-coordinate
`C`	A point defined by a vector containing an x- and an y-coordinate
`observer_position`	A vector containing an x- and an y-coordinate
`output_plot`	Boolean variable indicating whether a plot should be created
`lines_in_plot`	Boolean variable indicating whether lines should be drawn in the plot
`angles_in_plot`	Boolean variable indicating whether the angles should be printet in the plot
`decimals_in_plot`	Integer indicating the number of decimals used

Value

The angles as seen by the observer expressed in radians.

Examples

```
determine_angles(A = c(0, 0), B = c(10, 0), C = c(5, 5), observer_position=c(4,1))

determine_angles(A = c(0, 0), B = c(10, 0), C = c(5, 5), observer_position=c(4,40),
angles_in_plot = FALSE)
```

determine_position *Determine position of observer*

Description

Determine the position of an observer based on angles between three known points as seen by the observer. At least two angles must be provided - preferably `observer_angle_AB` and `observer_angle_AC` (since this combination allows for solutions outside the triangle formed by the points `A`, `B` and `C`)

Usage

```
determine_position(A, B, C, observer_angle_AB, observer_angle_AC,
  observer_angle_BC = NA, output_plot = TRUE, lines_in_plot = TRUE,
  coordinates_in_plot = TRUE, decimals_in_plot = 2)
```

Arguments

`A`	A point defined by a vector containing an x- and an y-coordinate
`B`	A point defined by a vector containing an x- and an y-coordinate
`C`	A point defined by a vector containing an x- and an y-coordinate
`observer_angle_AB`	An angle (numeric) expressed in radians (or alternatively the symbol `NA`)
`observer_angle_AC`	An angle (numeric) expressed in radians (or alternatively the symbol `NA`)
`observer_angle_BC`	An angle (numeric) expressed in radians (or alternatively the symbol `NA`)
`output_plot`	Boolean variable indicating whether a plot should be created
`lines_in_plot`	Boolean variable indicating whether lines should be drawn in the plot
`coordinates_in_plot`	Boolean variable indicating whether the coordinates should be printet in the plot
`decimals_in_plot`	Integer indicating the number of decimals used

Value

Coordinates indicating the observers position. Note that several solutions might exist.

Examples

```
determine_position(A = c(0, 0), B = c(10, 0), C = c(5, 5 * 3^0.5), observer_angle_AB = pi * 2/3,
observer_angle_AC = pi * 1/2)

determine_position(A = c(0, 0), B = c(10, 0), C = c(5, 5), observer_angle_AB = pi * 5/6,
observer_angle_AC = pi * 1/2, observer_angle_BC = NA, lines_in_plot = FALSE)

determine_position(A = c(0, 0), B = c(10, 0), C = c(5, 5), observer_angle_AB = pi * 5/6,
observer_angle_AC = pi * 1/2, observer_angle_BC = pi * 2/3, lines_in_plot = FALSE)
```

determine_region *Determine confidence region for position*

Description

This function is similar to determine_position()except for the fact that it is assumed that the angles are subject to measurement error. Hence a confidence region (error 'ellipse') is returned instead of an exact position.

Usage

```
determine_region(A, B, C, observer_angle_AB, observer_angle_AC,
  angle_error = pi/24, number_of_points = 200, output_plot = TRUE,
  lines_in_plot = FALSE, coordinates_in_plot = FALSE,
  decimals_in_plot = 2)
```

Arguments

A	A point defined by a vector containing an x- and an y-coordinate
B	A point defined by a vector containing an x- and an y-coordinate
C	A point defined by a vector containing an x- and an y-coordinate
observer_angle_AB	An angle (numeric) expressed in radians
observer_angle_AC	An angle (numeric) expressed in radians
angle_error	A numeric indicating the measurement error in radians
number_of_points	A numeric indicating the number of error points tested
output_plot	Boolean variable indicating whether a plot should be created
lines_in_plot	Boolean variable indicating whether lines should be drawn in the plot
coordinates_in_plot	Boolean variable indicating whether the coordinates should be printet in the plot
decimals_in_plot	Integer indicating the number of decimals used

Value

Coordinates indicating the outer border of the confidence region. Note that several different regions may exist.

Examples

```
determine_region(A = c(0, 0), B = c(10, 0), C = c(5, 5 * 3^0.5), observer_angle_AB = pi * 2/3,
observer_angle_AC = pi * 1/2)

determine_region(A = c(0, 0), B = c(10, 0), C = c(5, 5), observer_angle_AB = pi * 5/6,
observer_angle_AC = pi * 1/2, lines_in_plot = FALSE)
```

Index