PathInterpolatR

September 24, 2015

Title Methods for path interpolation

Version 0.1

Description Methods for performing path interpolation on movement data (e.g., GPS tracking data) are provided. Included are functions for performing interpolations using linear, Bezier curves, Catmull-Rom splines, time geographic constrained random walks, and kinematic interpolation.

Depends R (>= 3.1.2)

Imports sp, rgeos

License GPL-3

LazyData true

URL WorldWideWeb

R topics documented:

`PathInterpolatR-package`

PathInterpolatR: Methods for Path Interpolation

Description

The package `PathInterpolatR` provides tools for performing path interpolation on movement data (such as GPS tracking data). These tools were developed in support of the paper Long (2015) and extend the work of other authors (see references within).

Details

Currently the code has been developed for the simple scenario of interpolation between two points (where points on either side may or may not be known). Eventually, I will extend the code to facilitate straightforward implementation of interpolation with common movement data structures from other R packages (e.g., `adehabitat` and `MOVE`).

Author(s)

Jed Long

References

Long, JA (2015) Kinematic interpolation of movement data. *International Journal of Geographical Information Science*. DOI: 10.1080/13658816.2015.1081909.

`bezier` *Interpolate using Bezier curve*

Description

Perform path interpolation using a Bezier curve interpolation method.

Usage

```
bezier(xyt, t.slice)
```

Arguments

`xyt`	a 4x3 dataframe containing the coordinates and times for the four control points. Each row of the dataframe should be arranged as x, y, t.
`t.slice`	a single time (POSIX or numeric), or list of times, to be interpolated for. The times must lie between those of the middle two control points (i.e., the times associated with rows 2 & 3 in the `xyt` dataframe).

Details

Bezier curves can be useful for interpolating movement when the object exhibits curvi-linear movement shapes (Long 2015). For example, the movements of marine mammals often exhibit this property (Tremblay *et al.* 2006) . The Bezier method, as implemented here, requires four control points (2-D coordinates) as input, along with their corresponding times; these four points are passed to the function as a dataframe with four rows, and three columns corresponding to x, y, & t (the names of the columns do not matter). The function interpolates the movement trajectory for the times indicated by parameter `t.slice`. The `t.slice` times must lie between the time of the 2nd and 3rd control point. Times can be passed in as either POSIX-class objects or numeric, but not a mixture of both.

Value

The function returns a dataframe (with `nrow = length{t.slice}`) corresponding to the interpolated locations.

References

Long, JA (2015) Kinematic interpolation of movement data. *International Journal of Geographical Information Science*. DOI: 10.1080/13658816.2015.1081909.

Tremblay, YC *et al.* (2006) Interpolation of animal tracking data in a fluid environment. *Journal of Experimental Biology*. 209(1): 128-140.

Examples

```
data(contrived)
xyt <- contrived
###times for interpolation
t.slice <- c(1.5,2,2.5,3,3.5,4,4.5,5,5.5)
a <- bezier(xyt,t.slice)
plot(xyt[,1],xyt[,2],pch=20)
points(a[,1],a[,2])
```

`contrived` *Contrived example dataset*

Description

A dataset pertaining to the contrived example from Long (2015) used to demonstrate each method for path interpolation.

Format

A dataframe (`contrived`) delineating the four anchor points for path interpolation from the contrived example in Long (2015).

Details

Contrived example dataset for contrasting different path interpolation methods (open circles) with kinematic interpolation (grey crosses). In the contrived example, the object begins at the point z(0) = (0,-3) with a velocity of 0 m/s and then moves to the origin z(1) = (0, 0) with a velocity of 3 m/s to the North v(1) = (0,3). The object reaches position z(6) = (10,10) after 5 s, it now has a velocity of 3 m/s to the East v(6) = (3,0) and it continues on to location z(7) = (13,10). The time difference between the second and thrid point is 5 s and the object's location is estimated every 0.5 s in between.

References

Long, JA (2015) Kinematic interpolation of movement data. *International Journal of Geographical Information Science*. DOI: 10.1080/13658816.2015.1081909.

`kinematic` *Interpolate using kinematic interpolation*

Description

Perform kinematic path interpolation on a movement dataset. Kinematic path interpolation was introduced in the paper Long (2015). Kinematic interpolation is appropriate for fast moving objects, recorded with relatively high resolution tracking data.

Usage

```
kinematic(xytvv, t.slice)
```

Arguments

`xytvv`	a 2x5 dataframe containing the coordinates, times, and initial and final velocities (as 2D vectors) of the two points to be interpolated between, often termed the anchor points. Each row of the dataframe should be arranged as x, y, t, vx, vy.
`t.slice`	a single time (POSIX or numeric), or list of times, to be interpolated for. The times must lie between those of the points in `xytv`.

Details

Kinematic interpolation requires the user to input the coordinates of the anchor points between which the interpolation is occurring, as well as initial and final velocities associate with the anchor points. In practice, these velocities may be explicitly known, or estimated from the tracking data.

Value

The function returns a dataframe (with `nrow = length{t.slice}`) corresponding to the interpolated locations.

References

Long, JA (2015) Kinematic interpolation of movement data. *International Journal of Geographical Information Science*. DOI: 10.1080/13658816.2015.1081909.

Examples

```
data(contrived)
xyt <- contrived
###times for interpolation
t.slice <- c(1.5,2,2.5,3,3.5,4,4.5,5,5.5)
### add velocities for kinematic analysis
xytvv <- cbind(xyt[2:3,],rbind(c(0,3),c(3,0)))
a <- kinematic(xytvv,t.slice)
plot(xyt[,1],xyt[,2],pch=20)
points(a[,1],a[,2])
```

linear *Interpolate using linear interpolation*

Description

Perform linear path interpolation on a movement dataset.

Usage

```
linear(xyt, t.slice)
```

Arguments

xyt	a 2x3 dataframe containing the coordinates and times of the two points to be interpolated between, often termed the anchor points. Each row of the dataframe should be arranged as x, y, t.
t.slice	a single time (POSIX or numeric), or list of times, to be interpolated for. The times must lie between those of the points in xyt.

Details

Linear interpolation is the simplest, and most commonly employed methods for path interpolation.

Value

The function returns a dataframe (with nrow = length{t.slice}) corresponding to the interpolated locations.

Examples

```
data(contrived)
xyt <- contrived
###times for interpolation
t.slice <- c(1.5,2,2.5,3,3.5,4,4.5,5,5.5)
a <- linear(xyt[2:3,],t.slice)
plot(xyt[,1],xyt[,2],pch=20)
points(a[,1],a[,2])
```

splineCR *Interpolate using Catmull-Rom spline*

Description

Perform path interpolation using the Catumull-Rom spline interpolation method.

Usage

```
splineCR(xyt, t.slice)
```

Arguments

xyt	a 4x3 dataframe containing the coordinates and times for the four control points. Each row of the dataframe should be arranged as x, y, t.
t.slice	a single time (POSIX or numeric), or list of times, to be interpolated for. The times must lie between those of the middle two control points (i.e., the times associated with rows 2 & 3 in the xyt dataframe).

Details

Catmull-Rom splines can be useful for interpolating movement when the object exhibits curvi-linear movement shapes (Long 2015). For example, hurricanes often exhibit this property. The Catmull-Rom method requires four control points (2-D coordinates) as input, along with their corresponding times (Barry & Goldman 1988); these four points are passed to the function as a dataframe with four rows, and three columns corresponding to x, y, & t (the names of the columns do not matter). The function interpolates the movement trajectory for the times indicated by parameter t.slice. The t.slice times must lie between the time of the 2nd and 3rd control point. Times can be passed in as either POSIX-class objects or numeric, but not a mixture of both.

Value

The function returns a dataframe (with nrow = length{t.slice}) corresponding to the interpolated locations.

References

Long, JA (2015) Kinematic interpolation of movement data. *International Journal of Geographical Information Science*. DOI: 10.1080/13658816.2015.1081909.

Barry, PJ, Goldman, RN (1988) A recursive evaluation algorithm for a class of Catmull-Rom splines. *ACM SIGGRAPH Computer Graphics*. 22(4): 199-204.

Examples

```
data(contrived)
xyt <- contrived
###times for interpolation
t.slice <- c(1.5,2,2.5,3,3.5,4,4.5,5,5.5)
a <- splineCR(xyt,t.slice)
plot(xyt[,1],xyt[,2],pch=20)
points(a[,1],a[,2])
```

---

tgrw *Interpolate using time geographic constrained random walk*

Description

Perform path interpolation using the constrained random walk method outlined in the paper by Technitis et al. (2015), as implemented in the paper by Long (2015).

Usage

```
tgrw(xyt, t.slice, vmax = NA)
```

Arguments

xyt	a 2x3 dataframe containing the coordinates and times of the two points to be interpolated between, often termed the anchor points of the space-prism. Each row of the dataframe should be arranged as x, y, t.
t.slice	a single time (POSIX or numeric), or list of times, to be interpolated for. The times must lie between those of the points in xyt.
vmax	parameter controlling the bounds of the constrained random walk. Default value is 1.5 x d/t where d is the distance between the anchor points and t the time difference.

Details

Many moving objects exhibit movement properties that can be modelled via random walks. Thus, in many cases it is of interest to use random walks as a model for path interpolation. The time geographic constraned random walk is a special case of the random walk, wherby the interpolation is constrained by the space-time prism. The size of the space-time prism is controlled by the parameter vmax, which sets how far the interpolation is allowed to 'wander'. For more details, please see Technitis et al. (2015).

Value

The function returns a dataframe (with `nrow = length{t.slice}`) corresponding to the interpolated locations.

References

Long, JA (2015) Kinematic interpolation of movement data. *International Journal of Geographical Information Science*. DOI: 10.1080/13658816.2015.1081909.

Technitis, G. *et al.* (2015) From A to B, randomly: A point-to-point random trajectory generator for animal movement. *International Journal of Geographical Information Science*. 29(6): 912-934.

Examples

```
data(contrived)
xyt <- contrived
###times for interpolation
t.slice <- c(1.5,2,2.5,3,3.5,4,4.5,5,5.5)
a <- tgrw(xyt[2:3,],t.slice,vmax=6)
plot(xyt[,1],xyt[,2],pch=20)
points(a[,1],a[,2])
```

Index