Package ‘eRTG3D’

January 8, 2018

Title Generate Empirically Informed Random Trajectories in 3-D

Version 0.3.1

URL https://github.com/munterfinger/eRTG3D

Description The empirically informed random trajectory generator in three dimensions (eRTG3D) is an algorithm to generate realistic random trajectories in a 3-D space between two given fix points in space. The trajectory generation is based on empirical distribution functions extracted from observed trajectories (training data) and thus reflects the geometrical movement characteristics of the mover.

Depends R (>= 3.4.2)

Imports CircStats (>= 0.2-4), doParallel (>= 1.0.11), ggplot2 (>= 2.2.1), raster (>= 2.6-7), parallel (>= 3.4.2), plyr (>= 1.8.4), plotly (>= 4.7.1), sf (>= 0.5-5)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Merlin Unterfinger [aut, cre],
Kamran Safi [aut],
George Technitis [aut],
Robert Weibel [aut]

Maintainer Merlin Unterfinger <info@munterfinger.ch>

R topics documented:

`dem` *Example digital elevation model (DEM)*

Description

This is data to be included in the package and can be used to test its functionality. The 'dem' data is a rasterLayer and has a resolution of 90 meters. It is the topography of the Swiss midlands. The complete dataset can be downloaded directly from www.cgiar-csi.org.

References

http://www.cgiar-csi.org/data/srtm-90m-digital-elevation-database-v4-1

`dem2track.extent` *Crops the DEM to the extent of the track with a buffer*

Description

Crops the DEM to the extent of the track with a buffer

Usage

```
dem2track.extent(DEM, track, buffer = 100)
```

Arguments

`DEM`	a raster containing a digital elevation model, covering the extent as the track
`track`	data.frame with x,y,z coordinates of the original track
`buffer`	bufferwith, by default set to 100

Value

A the cropped digital elevation model as a raster layer.

Examples

```
dem2track.extent(DEM, track)
```

`filter.dead.ends` *Function to filter out tracks that have found a dead end (=NULL)*

Description

Function to filter out tracks that have found a dead end (=NULL)

Usage

```
filter.dead.ends(cerwList)
```

Arguments

`cerwList`	list of data.frames and NULL entries

Value

A list that is only containing valid tracks.

Examples

```
filter.dead.ends(cerwList)
```

`get.densities.3d` *Extract tldCube and autodifferences functions*

Description

Creates a list consisting of the 3 dimensional probability distribution cube for turning angle, lift angle and step length (turnLiftStepHist) as well as the uni-dimensional distributions of the differences of the turn angles, lift angles and step lengths with a lag of 1 to maintain minimal level of autocorrelation in each of the terms. Additionally also the distribution of the flight height over the ellipsoid (absolute) andthe distribution of flight height over the topography (relative) can be included.

Usage

```
get.densities.3d(turnAngle, liftAngle, stepLength, deltaLift, deltaTurn,
  deltaStep, heightEllipsoid = NULL, heightTopo = NULL, maxBin = 25)
```

Arguments

`turnAngle`	turn angles of the track (t)
`liftAngle`	lift angles of the track (l)
`stepLength`	stepLength of the track (d)
`deltaLift`	auto differences of the turn angles (diff(t))
`deltaTurn`	auto differences of the lift angles (diff(l))
`deltaStep`	auto differences of the step length (diff(d))
`heightEllipsoid`	flight height over the ellipsoid (absolute) or NULL to exclude this distribution
`heightTopo`	flight height over the topography (relative) or NULL to exclude this distribution
`maxBin`	numeric scalar, maximum number of bins per dimension of the tld-cube (turnLiftStepHist)

Value

A list containing the tldCube and the autodifferences functions (and additionally the flight height distribution functions)

Examples

```
get.densities.3d(track, heightDist = TRUE)
```

`get.section.densities.3d`

Extract tldCube and autodifferences functions from track sections

---

Description

Creates a list consisting of the 3 dimensional probability distribution cube for turning angle, lift angle and step length (turnLiftStepHist) as well as the uni-dimensional distributions of the differences of the turning angles, lift angles and step lengths with a lag of 1 to maintain minimal level of autocorrelation in each of the terms.

Usage

```
get.section.densities.3d(trackSections, heightDistEllipsoid = TRUE,
  DEM = NULL, maxBin = 25)
```

Arguments

`trackSections`	list of track sections got by the track.split.3d function
`heightDistEllipsoid`	logical: Should a distribution of the flight height over ellipsoid be extracted and later used in the sim.cond.3d()?
`DEM`	a raster containting a digital elevation model, covering the same extent as the track sections
`maxBin`	numeric scalar, maximum number of bins per dimension of the tld-cube (turnLiftStepHist)

Value

A list containing the tldCube and the autodifferences functions (and additionally the height distribution function)

Examples

```
get.section.densities.3d(trackSections)
```

---

get.track.densities.3d — *Extract tldCube and autodifferences functions from a consistent track*

Description

Get densities creates a list consisting of the 3 dimensional probability distribution cube for turning angle, lift angle and step length (turnLiftStepHist) as well as the uni-dimensional distributions of the differences of the turning angles, lift angles and step lengths with a lag of 1 to maintain minimal level of autocorrelation in each of the terms.

Usage

```
get.track.densities.3d(track, heightDistEllipsoid = TRUE, DEM = NULL,
  maxBin = 25)
```

Arguments

`track`	a data.frame with 3 columns containing the x,y,z coordinates
`heightDistEllipsoid`	logical: Should a distribution of the flight height over ellipsoid be extracted and later used in the sim.cond.3d()?
`DEM`	a raster containting a digital elevation model, covering the same extent as the track
`maxBin`	numeric scalar, maximum number of bins per dimension of the tld-cube (turnLiftStepHist)

Value

A list containing the tldCube and the autodifferences functions (and additionally the height distribution function)

Note

The time between the acquisition of fix points of the track must be constant, otherwise this leads to distorted statistic distributions, which increases the probability of dead ends. In this case please check track.split.3d and get.section.densities.3d

Examples

```
get.track.densities.3d(track, heightDist = TRUE)
```

`is.sf.3d` *Tests if the object is a simple feature collection (class: 'sf, data.frame')*

Description

Tests if the object is a simple feature collection (class: 'sf, data.frame')

Usage

```
is.sf.3d(track)
```

Arguments

`track`	any object to test

Value

A logical: TRUE if is is a simple feature collection (class: 'sf, data.frame') of the sf package, FALSE otherwise.

Examples

```
is.sf.3d(track)
```

`n.sim.cond.3d` *Conditioned Empirical Random Walks (CERW) in 3D*

Description

Creates n conditioned empirical random walks, with a specific starting and ending point, geometrically similar to the initial trajectory by applying sim.cond.3d multiple times.

Usage

```
n.sim.cond.3d(n.sim, n.locs, start = c(0, 0, 0), end = start, a0, g0,
  densities, qProbs, error = FALSE, multicore = FALSE, DEM = NULL,
  BG = NULL)
```

Arguments

`n.sim`	number of CERWs to simulate
`n.locs`	length of the trajectory in locations
`start`	numeric vector of length 3 with the coordinates of the start point
`end`	numeric vector of length 3 with the coordinates of the end point
`a0`	initial incoming heading in radian
`g0`	initial incoming gradient/polar angle in radian
`densities`	list object returned by get.densities.3d() function
`qProbs`	list object returned by qProb.3d() function

`error`	logical: add random noise to the turn angle, lift angle and step length to account for errors measurements?
`multicore`	logical: run computations in parallel (n-1 cores)?
`DEM`	raster layer containing a digital elevation model, covering the area between start and end point
`BG`	a background raster layer that can be used to inform the choice of steps

Value

A list containing the CERWs or NULLs if dead ends have been encountered.

Examples

```
n.sim.cond.3d(n.sim, n.locs, start = c(0,0,0), end=start, a0, g0, densities, qProbs)
```

`niclas` *Example track data.frame*

Description

This is data to be included in the package and can be used to test its functionality. The track consists of x, y and z coordinates and represents the movement of a stork called 'niclas' in the Swiss midlands.

References

https://www.movebank.org

`plot2d` *Plot function to plot the 3d tracks in 2d plane*

Description

Plot function to plot the 3d tracks in 2d plane

Usage

```
plot2d(origTrack, cerwList = NULL, titleText = character(1), DEM = NULL)
```

Arguments

`origTrack`	a data.frame with x,y,z coordinates
`cerwList`	a list containing a data.frame with x,y,z coordinates or a data.frame
`titleText`	string with title of the plot
`DEM`	an object of type 'RasterLayer', needs overlapping extent with the lines

Value

Nothing, plots a 2D ggplot2 object.

Examples

```
plot3d(track)
```

`plot3d` *Plot 3D track(s) with a surface*

Description

Plot 3D track(s) with a surface

Usage

```
plot3d(origTrack, cerwList = NULL, titleText = character(1), DEM = NULL,
  maxHeight = 8000)
```

Arguments

`origTrack`	a data.frame with x,y,z coordinates
`cerwList`	a list containing a data.frame with x,y,z coordinates or a data.frame
`titleText`	string with title of the plot
`DEM`	an object of type 'RasterLayer', needs overlapping extent with the lines
`maxHeight`	Maximum plot height, default 8000m

Value

Plots a 2D ggplot2 object

Examples

```
plot3d(track)
```

`plot3d.densities` *Density plots of turn angle, lift angle and step length*

Description

The function takes either one track or two tracks. The second track can be a list of tracks (eg. the output of n.sim.cons.3d()), Then the densities of turn angle, lift angle and step length of all the simulations is taken. Additionally the autodifferences parameter can be set to true, then the densities of the autodifferences in turn angle, lift angle and step length are visualized.

Usage

```
plot3d.densities(track1, track2 = NULL, autodifferences = FALSE,
  scaleDensities = FALSE)
```

Arguments

`track1`	a data.frame with x,y,z coordinates
`track2`	a list containing a data.frame with x,y,z coordinates or a data.frame
`autodifferences`	logical: Should the densities of the autodifferences in turn angle, lift angle and step length are visualized.
`scaleDensities`	logical: Should densities be scaled between 0 and 1, then sum of the area under the curve is not 1 anymore!

Value

A ggplot2 object.

Examples

```
plot3d.densities(track)
```

`plot3d.multiplot` *Multiple plot function for ggplot objects*

Description

If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE), then plot 1 will go in the upper left, 2 will go in the upper right, and 3 will go all the way across the bottom.

Usage

```
plot3d.multiplot(..., plotlist = NULL, cols = 1, layout = NULL)
```

Arguments

`...`	ggplot objects
`plotlist`	a list of ggplot objects
`cols`	number of columns in layout
`layout`	a matrix specifying the layout. If present, 'cols' is ignored.

Value

Nothing, plots the ggplot2 objects.

Examples

```
plot3d.multiplot(p1, p2, p3)
```

`qProb.3d` *Q probabilities for n steps*

Description

Calculates the Q probability, representing the pull to the target. The number of steps on which the Q prob will be quantified is number of total segments less than one (the last step is defined by the target itself).

Usage

```
qProb.3d(sim, n.locs, multicore = FALSE)
```

Arguments

`sim`	the result of sim.uncond.3d, or a data frame with at least x,y,z-coordinates, the arrival azimuth and the arrival gradient.
`n.locs`	number of total segments to be modelled, the length of the desired conditioned empirical random walk
`multicore`	logical: run computations in parallel (n-1 cores)?

Value

A list containing the Q - tldCubes for every step

Examples

```
qProb.3d(sim, n.locs)
```

`reproduce.track.3d` *Reproduce a track with the eRTG3D*

Description

Simulates n tracks with the geometrical properties of the original track, between the same start and end point.

Usage

```
reproduce.track.3d(track, n.sim = 1, multicore = FALSE, error = TRUE,
  DEM = NULL, BG = NULL, filterDeadEnds = TRUE, plot2d = FALSE,
  plot3d = FALSE, maxBin = 25)
```

Arguments

`track`	data.frame with x,y,z coordinates of the original track
`n.sim`	number of simulations that should be done
`multicore`	logical: run calculations on multiple cores?
`error`	logical: add error term to movement in simulation?
`DEM`	a raster containing a digital elevation model, covering the same extent as the track
`BG`	a raster influencing the probabilities.
`plot2d`	logical: plot tracks on 2d plane?
`plot3d`	logical: plot tracks in 3D?
`maxBin`	numeric scalar, maximum number of bins per dimension of the tld-cube (turnLiftStepHist)
`filterDeadEnds:`	logical: remove tracks (='NULL') that ended in a dead end?

Value

A list or data.frame containing the simulated track(s) (CERW).

Examples

```
reproduce.track.3d(track)
```

`sf2df.3d` *Converts a sf data.frame to a normal dataframe*

Description

Converts a sf data.frame to a normal dataframe

Usage

```
sf2df.3d(track)
```

Arguments

`track`	An object of type 'sf, data.frame'

Value

A data.frame.

Examples

```
sf2df.3d(df)
```

`sim.cond.3d` *Conditioned Empirical Random Walk (CERW) in 3D*

Description

Creates a conditioned empirical random walk, with a specific starting and ending point, geometrically similar to the initial trajectory (extractMethod: raster overlay method can take "simple" or "bilinear")

Usage

```
sim.cond.3d(n.locs, start = c(0, 0, 0), end = start, a0, g0, densities,
  qProbs, error = FALSE, DEM = NULL, BG = NULL)
```

Arguments

`n.locs`	length of the trajectory in locations
`start`	numeric vector of length 3 with the coordinates of the start point
`end`	numeric vector of length 3 with the coordinates of the end point
`a0`	initial incoming heading in radian
`g0`	initial incoming gradient/polar angle in radian
`densities`	list object returned by the get.densities.3d function
`qProbs`	list object returned by the qProb.3d function
`error`	logical: add random noise to the turn angle, lift angle and step length to account for errors measurements?
`DEM`	raster layer containing a digital elevation model, covering the area between start and end point
`BG`	a background raster layer that can be used to inform the choice of steps

Value

A trajectory in the form of data.frame

Examples

```
sim.cond.3d(n.locs, start, end=start, a0, g0, densities, qProbs)
```

`sim.crw.3d` *Simulation of a three dimensional Correlated Random Walk*

Description

Simulation of a three dimensional Correlated Random Walk

Usage

```
sim.crw.3d(nStep, rTurn, rLift, meanStep, start = c(0, 0, 0))
```

Arguments

`nStep`	the number of steps of the simulated trajectory
`rTurn`	the correlation on the turn angle
`rLift`	the correlation of the lift angle
`meanStep`	the mean step length
`start`	a vector of length 3 containing the coordinates of the startpoint of the trajectory

Value

A trajectory in the form of data.frame

Examples

```
sim.crw.3d(nStep, rTurn, rLift, meanStep, start = c(0,0,0))
```

`sim.uncond.3d` *Uncontidioned Empirical Random Walk (UERW) in 3D*

Description

This function creates unconditional walks with prescribed empirical properties (turning angle, lift angle and step length and the auto-differences of them. It can be used for uncon- ditional walks or to seed the conditional walks with comparably long simulations. The conditional walk connecting a given start with a certain end point by a given number of steps needs an attraction term (the Q probability, see qProb.3d) to ensure that the target is approached and hit. In order to calculate the Q probability for each step the distribution of turns and lifts to target and the distribution of distance to target has to be knwown. They can be derived from the empirical data (ideally), or estimated from an unconditional process with the same properties. Creates a unconditioned empirical random walk, with a specific starting point, geometrically similar to the initial trajectory.

Usage

```
sim.uncond.3d(n.locs, start = c(0, 0, 0), a0, g0, densities, error = TRUE)
```

Arguments

`n.locs`	the number of locations for the simulated track
`start`	vector indicating the start point c(x,y,z)
`a0`	initial heading in radian
`g0`	initial gradient/polar angle in radian
`densities`	list object returned by the get.densities.3d function
`error`	logical: add random noise to the turn angle, lift angle and step length to account for errors measurements?

Value

A 3 dimensional trajectory in the form of a data.frame

Note

Simulations connecting start and end points with more steps than 1/10th or more of the number of steps of the empirical data should rather rely on simulated unconditional walks with the same properties than on the empirical data (factor 1500).

Random initial heading

For a random initial heading a0 use: sample(atan2(diff(coordinates(track)[,2]), diff(coordinates(track)[,1])),1)

Examples

```
sim.uncond.3d(n.locs, start=c(0,0,0), a0, g0, densities)
```

`test.eRTG.3d` *Test the functionality of the eRTG3D*

Description

The test simulates a CRW with given parameters and reconstructs it by using the eRTG3D

Usage

```
test.eRTG.3d(multicore = FALSE, returnResult = FALSE, plot2d = FALSE,
  plot3d = FALSE)
```

Arguments

`multicore`	logical: test with multicore?
`returnResult`	logical: return tracks generated?
`plot2d`	logical: plot tracks on 2d plane?
`plot3d`	logical: plot tracks in 3D?

Value

A list containing the original CRW and the simulated track (CERW).

Examples

```
test.eRTG3D.3d()
```

`test.verification.3d` *Internally verification of the simulated track*

Description

Uses two-sample Kolmogorov-Smirnov test to compare the geometric characteristics of the orginal track with the characteristics of the simulated track.

Usage

```
test.verification.3d(track1, track2, alpha = 0.05, plotDensities = FALSE)
```

Arguments

`track1`	data.frame with x,y,z coordinates of the original track
`track2`	data.frame or list of data.frames with x,y,z coordinates of the simulated track
`alpha`	scalar: significance level, default alpha = 0.05
`plotDensities`	logical: plot the densites of turn angle, lift angle and step length of the two tracks?

Value

Test objects of the 6 two-sample Kolmogorov-Smirnov test conducted.

Examples

```
test.verification.3d(track1, track2)
```

`track.properties.3d` *Track properties of a 3D track*

Description

Returns the properties (distances, azimut, polar angle, turn angle & lift angle) of a track in three dimensions.

Usage

```
track.properties.3d(track)
```

Arguments

`track`	data.frame with x,y,z coordinates

Value

The data.frame with track properties

Examples

```
track.properties.3d(track)
```

---

track.split.3d — *This function splits the by outliers in the time lag.*

Description

The length of timeLag must be the the track's length minus 1 and represents the time passed between the fix point acquisition

Usage

```
track.split.3d(track, timeLag)
```

Arguments

`track`	track data.frame with x, y and z coordinates
`timeLag`	a numeric vector with the time passed between the fix point acquisition

Value

A list containing the splitted tracks.

Examples

```
track.split.3d(track, timeLag)
```

---

track2sf.3d — *Converts a track to a sf data.frame*

Description

Converts a track to a sf data.frame

Usage

```
track2sf.3d(track, CRS = NA)
```

Arguments

`track`	eRTG3D track data.frame or a matrix
`CRS`	string containing the proj4 code of the CRS

Value

A track of type 'sf, data.frame'.

Examples

```
track2sf.3d(track, "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
```

`transformCRS.3d` *Transform coordinates reference system of a 3D track*

Description

Attention: Please use this function for CRS transformations, because it is based on the 'st_transform()' from the sf package. Therefore is supports CRS transdormations in 3D. Note: 'spTransform()' from the 'sp' only supports transformations in the 2D plane, which will cause distortions in the third dimension.

Usage

```
transformCRS.3d(track, fromCRS, toCRS)
```

Arguments

`track`	data.frame with x,y,z coordinates
`fromCRS`	string: proj4 of current CRS
`toCRS`	string: proj4 of CRS to be converted in

Value

A data.frame containing x,y,z and variables.

Examples

```
transformCRS.3d(track, fromCRS="+init=epsg:4326", toCRS="+init=epsg:2056")
```

`turnLiftStepHist` *3 dimensional histogram*

Description

Derives a 3 dimensional distribution of a turn angle, lift angle and step length, using the Freedman–Diaconis rule for estimating the number of bins.

Usage

```
turnLiftStepHist(turn, lift, step, printDims = TRUE, rm.zeros = TRUE,
  maxBin = 25)
```

Arguments

`turn`	numeric vector of turn angles
`lift`	numeric vector of lift angles
`step`	numeric vector of step lengths
`printDims`	logical: Should dimensions of tld-Cube be messaged?
`rm.zeros`	logical: should combinations with zero probability be removed?
`maxBin`	numeric scalar, maximum number of bins per dimension of the tld-cube.

Value

A 3 dimensional histogram as data.frame

Examples

```
turnLiftStepHist(turn, lift, step)
```

Index