STPtrajectories

March 5, 2017

R topics documented:

alibi_query 1
axes_STP_plot 2
calculate_PPA 3
getMinimalSpeed 5
getVmaxtrack 5
RTG 6
STPtrajectories 8
STP_plot 9
STP_Track 10

Index **13**

`alibi_query` *alibi_query*

Description

CONTAINS ERRORS. PROBLEMS WITH CASE 3!!!!! This function tests wether there was a possible meeting between two individuals or other moving objects. If the individuals of two trajecories could have met is tested by applying the alibi query to segments that overlap in time. The alibi query is a Boolean query that checks whether two moving individuals, that are given by two samples of space-time points and speed limitations, could have met each other. The query tests if two space-time prisms intersect. Kuijpers et al. (2011) provide the analytical solution for the alibi query that is used by this function

Usage

```
alibi_query(STP_track1, STP_track2)
```

Arguments

`STP_track1`	STP_track1
`STP_track2`	STP_track2

Value

True or False for the alibi query

Author(s)

Mark ten Vregelaar

References

- Kuijpers, B., Grimson, R., & Othman, W. (2011). An analytic solution to the alibi query in the space-time prisms model for moving object data. International Journal of Geographical Information Science, 25(2), 293-322.

Examples

```
library(spacetime)
library(sp)

## create 2 STP_tracks
# time
t1 <- strptime("01/01/2017 00:00:00", "%m/%d/%Y %H:%M:%S")
t2 <- t1+5*60*60 # 5 hours after t1
time1<-seq(t1,t2,30*60)
time2<-time1+0.25*60*60
# spatial coordinates
x1=c(seq(0,25,5),seq(27.5,37.5,2.5))
y1=sample(-2:2, 11,replace = TRUE)
x2=c(seq(0,25,5),seq(27.5,37.5,2.5))
y2=sample(-2:2, 11,replace = TRUE)

n = length(x1)
crs_NL = CRS("+init=epsg:28992")

# create class STIDF
stidf1 = STIDF(SpatialPoints(cbind(x1,y1),crs_NL), time1, data.frame(co2 = rnorm(n),O2=rnorm(n)))
stidf2 = STIDF(SpatialPoints(cbind(x2,y2),crs_NL), time2, data.frame(co2 = rnorm(n),O2=rnorm(n)))

# Track-class {trajectories}
my_track1<-Track(stidf1)
my_track2<-Track(stidf2)
# set maximum speed
v1<-getVmaxtrack(my_track1)+0.00015
v2<-getVmaxtrack(my_track2)+0.00030
# STP_track class
STP_track1<-STP_Track(my_track1,v1)
STP_track2<-STP_Track(my_track2,v2)

## the alibi query
alibi_query(STP_track1,STP_track2)
```

---

axes_STP_plot *axes_STP_plot*

Description

This function adds a bbox with axis to a STP_plot

Usage

```
axes_STP_plot(minmaxT, z_factor, n_ticks_xy = 3, n_ticks_z = 5)
```

Arguments

`minmaxT`	a vector of length 2 with two "POSIXct" or"POSIXt" values
`z_factor`	the z facfor used in the plot
`n_ticks_xy`	number of ticks used for the x and y axes
`n_ticks_z`	number of ticks used for the z axes

Author(s)

Mark ten Vregelaar

`calculate_PPA` *calculate_PPA*

Description

Function for calculating the Potetial Path Area(PPA) of a STP_track. This function can calculate the PPA for the entire trajectory, a specfic moment in time or a time range.

Usage

```
calculate_PPA(STP_track, time = NULL, points = NULL, x_density = 250,
  time_interval = 1, quadsegs = 12)
```

Arguments

`STP_track`	The STP_track for which the PPA needs to be calculated
`time`	Time("POSIXct" or"POSIXt") for which the PPA needs to be calculated. Use time = c(time1,time2) to calculate PPA for a time range. Default is NULL: calculate PPA for entire STP_track
`points`	The points used for the PPA calculation given as a vector of integers. Default is NULL: calculate PPA for entire STP_track
`x_density`	Paramter used for calculating the PPA of entire STPs. The amount of x coordinates for which the corresponding y coordinate(s) will be calculated. Only relevant if the PPA for at least 1 complete STP needs to be calculated
`time_interval`	The time interval in minutes used for calculating the PPA. Only used for calculating the PPA for a specfic moment in time and if only a part of the PPA of a STP needs to be calculated. Default is every minute
`quadsegs`	Passed to buffer. Number of line segments to use to approximate a quarter circle. Only used where paramter time_interval is relavant

Value

The Potential Path Area as SpatialPolygons

Author(s)

Mark ten Vregelaar

Examples

```
library(spacetime)
library(sp)
#--------------------------create a STP_Track--------------------------
# set time
t1 <- strptime("01/01/2017 00:00:00", "%m/%d/%Y %H:%M:%S")
t2 <- t1+5*60*60
time<-seq(t1,t2,30*60)
# set coordinates
x=c(seq(0,25,5),seq(27.5,37.5,2.5))
y=sample(-2:2, 11,replace = TRUE)

n = length(x)
crs_NL = CRS("+init=epsg:28992")

# create class STIDF
stidf1 = STIDF(SpatialPoints(cbind(x,y),crs_NL), time, data.frame(co2 = rnorm(n),O2=rnorm(n)))

# Track-class {trajectories}
my_track1<-Track(stidf1)

# set maximum speed
v1<-getVmaxtrack(my_track1)+0.00015

# STP_track class
STP_track1<-STP_Track(my_track1,v1)
#------------------------------example 1------------------------------
## PPA entire track
#calculate PPA
PPA<-calculate_PPA(STP_track1)

# plot results
plot(STP_track1,type='b')
plot(PPA,add=TRUE)
#------------------------------example 2------------------------------
## PPA only using every second point
# calculate PPA
PPA<-calculate_PPA(STP_track1,points = seq(1,11,2))

# plot results
plot(STP_track1,type='b')
plot(PPA,add=TRUE)
#------------------------------example 3------------------------------
## PPA of a specfic moment in time
# calculate PPA
time <- strptime("01/01/2017 01:15:00", "%m/%d/%Y %H:%M:%S")
PPA<-calculate_PPA(STP_track1,time = time)

# plot results
plot(STP_track1,type='b')
plot(PPA,add=TRUE)
#------------------------------example 4------------------------------
```

```
## PPA for a time range
# calculate PPA
timerange1 <- c(t1,strptime("01/01/2017 02:15:00", "%m/%d/%Y %H:%M:%S"))
PPA<-calculate_PPA(STP_track1,time = timerange1)

# plot results
plot(STP_track1,type='b')
plot(PPA,add=TRUE)
```

`getMinimalSpeed` *getMinimalSpeed*

Description

This functions calculates the minimal speed required to reach every point. The speed is based on linear movement between two points.

Usage

```
getMinimalSpeed(track)
```

Arguments

`track`	the trajectory as STP_Track or Track

Value

Vector of minimal speeds in unit spatial projection unit/s The speed required to reach the next point in the available time

Author(s)

Mark ten Vregelaar

`getVmaxtrack` *getVmaxtrack*

Description

This functions calculates the maximum speed found in a trajectory. The maximum speed is based on linear movement between measured points.

Usage

```
getVmaxtrack(track)
```

Arguments

`track`	the trajectory as STP_Track or Track

Value

max speed of the moving object

Author(s)

Mark ten Vregelaar

`RTG` *RTG*

Description

This random trajectory generator (RTG) works as described in Technitis et al.(2015). It creates a trajectory based on a the space-time prism concept, it randomly adds a user defined number of points between the points of a trajectory. The new point is randomly placed in the PPA of the corresponding point in time. The added points are evenly divided over time and are always within the space-time prism.

Usage

```
RTG(STP_track, n_points = 1, max_time_interval = NULL, quadsegs = 12,
  iter = 4)
```

Arguments

`STP_track`	the STP_Track to which the randomly generated space-time points are added
`n_points`	number of points will be added between the two points. If no value is provided new point(s) will be added between all consecutive space-time points
`max_time_interval`	The max_time_interval determines between which space-time points the random points are added. If the time difference between two points is bigger than max_time_interval,
`quadsegs`	Passed to buffer. Number of line segments to use to approximate a quarter circle. Only used where paramter time_interval is relavant
`iter`	number of times to try to place sample points in the PPA before giving up and returning NULLter (default = 4) - this may occur when trying to hit a small and awkwardly shaped polygon in a large bounding box with a small number of points.

Value

a STP_Track with the newly added random space-time points. Slot data has NAs for the new points. Vmax values for new connections are equal to the vmax values of the original connections.

Author(s)

Mark ten Vregelaar

References

- Technitis, G., Othman, W., Safi, K., & Weibel, R. (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. http://www.tandfonline.com/doi/abs/10.1080/13658816.2014.999682

Examples

```
library(spacetime)
library(sp)
#------------------------------example 1------------------------------
## Create a random trajecory based on a begin and end point
## Create trajectory with only two points
# Time
t1 <- as.POSIXct(strptime("01/01/2017 00:00:00", "%m/%d/%Y %H:%M:%S"))
t2 <- t1+0.5*60*60 # 2 hours after t1
time<-c(t1,t2)
# Spatial coordinates
x=c(5,10);y=c(10,20)
n = length(x)
crs_NL = CRS("+init=epsg:28992")

# create class STIDF
stidf1 = STIDF(SpatialPoints(cbind(x,y),crs_NL), time, data.frame(co2 = rnorm(n),O2=rnorm(n)))

# Track-class {trajectories}
track1<-Track(stidf1)

# Set maximum speed
v1<-getVmaxtrack(track1)+0.001
# STP_track class
STP1<-STP_Track(track1,v1)
plot(STP1,type='p',col='red',pch=16,cex=2)

# Create a random trajectory between the two points
random_STP_track<-RTG(STP1,n_points = 10)
plot(random_STP_track,type='b',add=TRUE)

#------------------------------example 2------------------------------
## Add points to a trajectory with multple points
## Create a STP_track
np <-6 # Number of points orignal track
t1 <- as.POSIXct(strptime("01/01/2017 00:00:00", "%m/%d/%Y %H:%M:%S"))
random1<-cumsum(sample((0.5*60):(2.8*60*60),np))
time<-t1+random1

x=random1/2
y=seq(1,100,length.out = np)

n = length(x)
crs_NL = CRS("+init=epsg:28992")

# Create class STIDF
stidf2 = STIDF(SpatialPoints(cbind(x,y),crs_NL), time, data.frame(co2 = rnorm(n),O2=rnorm(n)))

# Track-class {trajectories}
```

```
track2<-Track(stidf2)

# Set maximum speed
v1<-getVmaxtrack(track2)+0.1

# STP_track class
STP_track2<-STP_Track(track2,v1)
# STP_track2 is track with different time intervals between the space-time points.
# The distance between two points increases with the time interval
plot(STP_track2,type='p',col='red',pch=16,cex=2)

## Fill blank spot of trajecotries in two steps
# Add 2 random points in between two sapce-time points that more than 90 minutes apart
filled_track1 <-RTG(STP_track2,n_points = 2,max_time_interval = 120)
plot(filled_track1,type='p',pch=16,add=TRUE,col='blue')

# Add 1 random point in between two sapce-time points that more than 45 minutes apart
filled_track2 <-RTG(filled_track1,n_points = 1,max_time_interval = 60)
plot(filled_track2,type='b',add=TRUE,cex=0.7)
```

STPtrajectories *STPtrajectories.*

Description

Package for handling Space-Time Prism(STP) trajectories. It contains functions to calculate Potential Path Areas(PPAs), create random trajectories and to test for possible encounters by applying the alibi query. It also provides functions to visulize the STPs treajectories in 3D.

Background

A trajectory consists of successive points in space and time. The location of the individual between two successive points is unknown, but based on a maximum speed a space-time prism can be calculated. A space-time prism(STP) is defined as the collection of space-time locations the individual could reach, given a speed limitation. The STP_Track class can be used to handle space-time prism trajectories.

The alibi_query uses this concept to test if two individuals could have met each other. The package provides related functions to help users in the analysis of their trajectories and take into account the uncertainty about the location of an individual. These include methods to visualise space-time prisms(STP_plot), calculate the potential path area(calculate_PPA) and a random trajectory generator(RTG).

help

need help:

- manual: `https://github.com/markvregel/STPtrajectories/blob/master/STPtrajectories.pdf`
- vignette: `http://htmlpreview.github.io/?https://raw.githubusercontent.com/markvregel/STPtrajectories/master/vignettes/STP_Tracks.html`
- check functions help and examples

Author(s)

Mark ten Vregelaar

See Also

- github: https://github.com/markvregel/STPtrajectories
- trajecotries package https://cran.rstudio.com/web/packages/trajectories/index.html

---

STP_plot *STP_plot*

Description

This function visualizes STPs in 3D

Usage

```
STP_plot(STP_track, time_interval, zfactor = 1, col = "red", st = NULL)
```

Arguments

`STP_track`	A STP_Track
`time_interval`	the time interval in minutes.Determines the amount of PPAs that are plotted
`zfactor`	realtive size of z axis compared to x and y axis
`st`	start time as "POSIXct" or"POSIXt". For plotting multiple STP_tracks use 1 starttime
`color`	of STP(s)

Author(s)

Mark ten Vregelaar

Examples

```
library(spacetime)
library(sp)
## create 2 STP_tracks
# time
t1 <- strptime("01/01/2017 00:00:00", "%m/%d/%Y %H:%M:%S")
t2 <- t1+5*60*60 # 5 hours after t1
time1<-seq(t1,t2,30*60)
time2<-time1+0.25*60*60
# spatial coordinates
x1=c(seq(0,25,5),seq(27.5,37.5,2.5))
y1=sample(-2:2, 11,replace = TRUE)
x2=c(seq(0,25,5),seq(27.5,37.5,2.5))
y2=sample(-2:2, 11,replace = TRUE)

n = length(x1)
crs_NL = CRS("+init=epsg:28992")
```

```
# create class STIDF
stidf1 = STIDF(SpatialPoints(cbind(x1,y1),crs_NL), time1, data.frame(co2 = rnorm(n),O2=rnorm(n)))
stidf2 = STIDF(SpatialPoints(cbind(x2,y2),crs_NL), time2, data.frame(co2 = rnorm(n),O2=rnorm(n)))

# Track-class {trajectories}
my_track1<-Track(stidf1)
my_track2<-Track(stidf2)
# set maximum speed
v1<-getVmaxtrack(my_track1)+0.00015
v2<-getVmaxtrack(my_track2)+0.00030
# STP_track class
STP_track1<-STP_Track(my_track1,v1)
STP_track2<-STP_Track(my_track2,v2)


## 3D STP plot of STP_tracks
z_fac<-0.2 # relative size of z scale/aspect ratio to spatial scale
# plot STPS first STP_track
STP_plot(STP_track1,time_interval = 1,z_fac)
# plot STPS second STP_track
STP_plot(STP_track2,time_interval = 1,z_fac,'blue',st = STP_track1@endTime[1])
# provide st for correct starting location first STP

# calculate first and last moment in time
min_max_Time<-c(STP_track1@endTime[1],STP_track2@endTime[length(STP_track2)])
# add axes
axes_STP_plot(min_max_Time,z_factor = z_fac)

# add title and change background colour
library(rgl)
title3d(main = '2 randomly generated STP tracks')
bg3d('lightblue')
```

`STP_Track` *STP_Track class*

Description

A class to represent Space-Time Prism(STP) trajecories. These are trajectories with a maximum speed for each segment. The maximum speed is added to the connections slot of class Track. The STP_Track can also combined in classes Tracks and TracksColletion of the trajecories package

Usage

```
STP_Track(track, vmax)
```

Arguments

`Track`	object of class Track

Slots of class "Track"

`sp`: spatial locations of the track points, with length n

`time`: time stamps of the track points

`endTime`**:** end time stamps of the track points

`data`**:** `data.frame` with n rows, containing attributes of the track points

`connections`**:** `data.frame`, with n-1 rows, containing attributes between the track points such as distance and speed

See Also

rajecotries package :https://cran.rstudio.com/web/packages/trajectories/index.html

Examples

```
library(spacetime)
library(sp)
#--------------------------create a STP_Track--------------------------
#------------------------------example 1------------------------------
## create trajectory data
t1 <- as.POSIXct(strptime("01/01/2017 12:00:00", "%m/%d/%Y %H:%M:%S"))
t2 <- as.POSIXct(strptime("01/7/2017 12:00:00", "%m/%d/%Y %H:%M:%S"))
time <- seq(t1,t2,2*60*60)
n <- length(time)
x = cumsum(runif(n) * 8000)
y = smooth(cumsum(runif(n,-0.7,1) * 16000))

crs_NL = CRS("+init=epsg:28992")

points <- SpatialPoints(cbind(x,y),crs_NL)

temp <-18 + cumsum(runif(n,-0.3,0.25))
altitude <- 200 + cumsum(runif(n,-0.75,1)*50)

data <- data.frame(temperature = temp, elevation = altitude)
## create a STP_track
# create class STIDF
stidf1 = STIDF(points, time, data)

# Track-class {trajectories}
my_track1<-Track(stidf1)

# set maximum speed
v1<-10/3.6# speed 10 km/h = 2.777778 m/s

# STP_track class
STP_track1<-STP_Track(my_track1,v1)
# plot
plot(STP_track1,type='p',pch=19,cex=0.8)
# calculate PPA and add to plot
PPA<-calculate_PPA(STP_track1)
plot(PPA,add=TRUE)

#------------------------------example 2------------------------------
## vmax depends on elevation
# assuming that max speed is lower as result of the thinner air
vmax<- getVmaxtrack(STP_track1) + (max(STP_track1@data$elevation[1:n-1])-STP_track1@data$elevation[1:n-1])*0
STP_track1@connections$vmax<-vmax
# calculate PPA
PPA<-calculate_PPA(STP_track1)
```

```
# create tracksCollection and plot
tracks = Tracks(list(tr1 = STP_track1))
tracksCollection = TracksCollection(list(tr = tracks))
stplot(tracksCollection, attr = "elevation", lwd = 3, scales = list(draw =TRUE),
    sp.layout=PPA,xlim = PPA@bbox[1,],ylim = PPA@bbox[2,],main= "Track with PPA\n vmax depends on altitude",
      sub='colour is altitude in meters',xlab='x',ylab='y')
#------------------------------example 3------------------------------
## vmax depends on the distance to get to next point.
# Thus on the distance that needs to be covered in the avialable time
# Assuming that if two points are closer together the max speed is lower
STP_track1@connections$vmax<-getMinimalSpeed(STP_track1)*1.5
# calculate PPA
PPA<-calculate_PPA(STP_track1)
# create tracksCollection and plot
plot(PPA,add=TRUE)
tracks = Tracks(list(tr1 = STP_track1))
tracksCollection = TracksCollection(list(tr = tracks))
stplot(tracksCollection, attr = "vmax", lwd = 3, scales = list(draw = TRUE),
      sp.layout=PPA,xlim = PPA@bbox[1,],ylim = PPA@bbox[2,],
    main= "Track with PPA\n vmax depends on the distance and time budget between consecutive points",
      sub='colour is vmax in m/s',xlab='x',ylab='y',cex.main = 0.75)


#--------------------------subset a STP_Track--------------------------
# make sure vmax is high enough
STP_track1@connections$vmax<-getVmaxtrack(STP_track1)+1
#------------------------------example 1------------------------------
## subset based on space-time points
# get first 10 points
STP_track1_10<-STP_track1[1:10,'']
# only keep every second point
STP_track1_depleted<-STP_track1[seq(1,n,2),'']
plot(STP_track1,type='b')
plot(STP_track1_depleted,type='b')
#------------------------------example 2------------------------------
## subset based on time
STP_track1_a<-STP_track1[1:n,'2017-01-01 12:00:00 CET::2017-01-02 16:00:00 CET']
# only keep every second point within the time interval
STP_track1_b<-STP_track1[seq(1,n,2),'2017-01-01 12:00:00 CET::2017-01-02 16:00:00 CET']
# all points in the night(between 22:00 and 08:00)
STP_track1_c<-STP_track1[1:n,"T22:00/T08:00"] # see package xts for more handy subsetting tricks
```

Index

alibi_query, 1, *8*
axes_STP_plot, 2

calculate_PPA, 3, *8*

getMinimalSpeed, 5
getVmaxtrack, 5

RTG, 6, *8*

STP_plot, *8*, 9
STP_Track, *5, 6*, *8*, 10
STPtrajectories, 8
STPtrajectories-package
*(*STPtrajectories*)*, 8

Track, *5*, *10*
Tracks, *10*
TracksColletion, *10*