

Package ‘Strategy’

October 20, 2020

Type Package

Title X

Version 1.0

Date 2020-10-20

Author D. Bonnery

Maintainer D. Bonnery <daniel.bonnery@gmail.com>

Description Data

Depends ggplot2,
leaflet,
spatstat,
sf

License GPL (>= 2)

LazyLoad yes

LazyData true

RoxygenNote 7.0.2

R topics documented:

addpiechartclustermarkers	2
closestpointonpolygon	3
closestpointonseg	4
closestpointsontwopolygons	4
closestpointsontwopolygons_n	5
closestpointsontwosegments	6
closestpointsontwosegments_n	6
distpointtopoly	7
distpointtoseg	8
distpolytopoly	9
distpolytopoly2	10
distsegmenttopoly	10
distsegmenttosegment	11
dist_areas_f	12

extractpolygonsaslist	12
Generate_Discrete_Time_Epidemic	13
Generate_U	13
generate_U2	14
Interactive2	15
interactive_1	15
neighbourhoods	16
newdist	16
polydistmat	17
polysmalldistmat	18
projpointonseg_a	19
ranges.gap	20
rangesbygradients_f	20
rangesoverlap	21
risktobeinfected	22
risktobeinfectedbydistancetoallinfectedunit	22
risktobeinfectedbydistancetooneinfectedunit	23
segment.intersect	24
shinyapp1	24
subsets	25
triangleorientation	25
updatedist	26
Index	28

addpiechartclustermarkers

Change leaflet cluster markers to pie charts

Description

Change leaflet cluster markers to pie charts

Usage

```
addpiechartclustermarkers(map, .data, .colors, group, ...)
```

Arguments

map	the map to add awesome pie chart cluster markers to
.data	data for the cluster markers
.colors	a vector of colors of at least the same size that nlevels(.data[[group]])
group	the name of a factor variable of .data
...	further arguments to be passed to addAwesomeMarkers

Examples

```
data("breweries91",package="leaflet")
breweries91$goodbear<-sample(as.factor(c("terrific","marvelous","culparterretaping")),nrow(breweries91),replace=TRUE)
library(leaflet)
library(dplyr)
leaflet(breweries91) %>%
  addTiles() %>%
  addAwesomeMarkers()
map<-leaflet(breweries91) %>%addTiles()
addpiechartclustermarkers(map,.data=breweries91,.colors=c("red","green","blue"),group="goodbear")
leaflet(breweries91) %>%
  addTiles() %>%
  addpiechartclustermarkers(.data=breweries91,.colors=c("red","green","blue"),group="goodbear")
```

`closestpointonpolygon` *computes the coordinates of the closest point on the border of a polygon to a point in the plane*

Description

computes the coordinates of the closest point on the border of a polygon to a point in the plane

Usage

```
closestpointonpolygon(p, .poly)
```

Arguments

<code>p</code>	a numeric vector of length 2
<code>.poly</code>	a nx2 numeric matrix, representing a polygon. Each row of the matrix are the coordinates of a vertice of the polygon.

Value

the coordinates of the closest point on the polygon

Examples

```
zz<-function(){
  .poly=matrix(sample(0:4,6,rep=T),3,2)[c(1:3,1),]
  p<-sample(0:4,2,rep=T)
  dd<-distpointtopoly(p,.poly)
  plot(rbind(p,.poly),cex=.5,main=paste0("Distance: ", signif(dd,3)),
       asp=1,xlim=range(cbind(p,.poly)),ylim=range(cbind(p,.poly)),xaxt='n',yaxt='n',xlab='',ylab='')
  points(.poly,type="l",lwd=2)
  cc<-rbind(p,closestpointonpolygon(p,.poly))
  points(cc,col="red",cex=2)
  points(cc,type="l",lty=3,col="red")
}
```

```
}

par(oma=c(0,0,0,0),mfrow=c(2,2))
set.seed(3);replicate(4,zz())
```

closestpointonseg	<i>computes the coordinates of the closest point on a segment to a point in the plane</i>
-------------------	---

Description

computes the coordinates of the closest point on a segment to a point in the plane

Usage

```
closestpointonseg(p, s)
```

Arguments

p	a numeric vector of length 2
s	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

Examples

```
zz<-function(p){
  s=matrix(c(0,1,0,0),2,2)
  plot(s,type="l",xlim=c(-.5,1.5),
    xlab="",ylab="",
    xaxt='n',yaxt='n')
  points(x=p[1],y=p[2] ,col="red",cex=.5)
  points(closestpointonseg(p,s)[1],closestpointonseg(p,s)[2],col="red",cex=.5)
  segments(x0 = p[1],y0=p[2],x1=closestpointonseg(p,s)[1],y1=closestpointonseg(p,s)[2],col="red")
  par(mfrow=c(3,3))
  set.seed(1);replicate(9,zz(c(runif(1,-.5,1.5),runif(1,-1,1))))
```

closestpointsonwopolygons	<i>computes the coordinates of the closest point on the border of a polygon to a point in the plane</i>
---------------------------	---

Description

computes the coordinates of the closest point on the border of a polygon to a point in the plane

Usage

```
closestpointstwopolygons(poly1, poly2)
```

Arguments

poly1 a nx2 numeric matrix, representing a polygon. Each row of the matrix are the coordinates of a vertex of the polygon.

poly2 a nx2 numeric matrix, representing a polygon. Each row of the matrix are the coordinates of a vertex of the polygon.

Value

the coordinates of the closest point on the polygon

Examples

```
zz<-function(){
  poly1=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
  poly2=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
  s<-rbind(poly1,poly2)
  dd<-distpolytopoly(poly1,poly2)
  plot(s,cex=.5,main=paste0("Distance: ", signif(dd,3)),asp=1,xlim=range(s),ylim=range(s),xaxt='n',yaxt='n',xlab=
  points(poly1,type="l",lwd=2)
  points(poly2,type="l",lwd=2)
  cc<-closestpointstwopolygons(poly1,poly2)
  points(cc,col="red",cex=2)
  points(cc,type="l",col="red",lty=3)
}

par(oma=c(0,0,0,0),mfrow=c(2,2))
set.seed(2);replicate(4,zz())
```

closestpointstwopolygons_n

Compute minimum distance between two polygons

Description

Compute minimum distance between two polygons

Usage

```
closestpointstwopolygons_n(poly1, poly2)
```

Arguments

poly1 a nx2 numeric matrix, representing a polygon. Each row of the matrix are the coordinates of a vertex of the polygon.

poly2 a nx2 numeric matrix, representing a polygon. Each row of the matrix are the coordinates of a vertex of the polygon.

Examples

```

zz<-function(){
poly1=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
poly2=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
s<-rbind(poly1,poly2)
dd<-distpolytopoly(poly1,poly2)
plot(s,cex=.5,main=paste0("Distance: ", signif(dd,3)),asp=1,xlim=range(s),ylim=range(s),xaxt='n',yaxt='n',xlab=
points(poly1,type="l",lwd=2)
points(poly2,type="l",lwd=2)
for(cc in closestpointsonwopolygons_n(poly1,poly2)){
points(cc,col="red",cex=2)
points(cc,type="l",col="red",lty=3)}
}

par(oma=c(0,0,0,0),mfrow=c(2,2))
set.seed(2);replicate(4,zz())

```

closestpointsontwosegments

computes the coordinates of the closest point on a segment to a point in the plane

Description

computes the coordinates of the closest point on a segment to a point in the plane

Usage

```
closestpointsontwosegments(s1, s2)
```

Arguments

s1	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.
s2	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

closestpointsontwosegments_n

returns a list of one or more pair of points, one on each of two segments, with minimal distance

Description

returns a list of one or more pair of points, one on each of two segments, with minimal distance

Usage

```
closestpointstwowsegments_n(s1, s2)
```

Arguments

s1 a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

s2 a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

Value

a list of 2x2 numeric matrices, representing a segment.

Examples

```
zz<-function(){
  s1=matrix(sample(0:4,4,rep=T),2,2)
  s2=matrix(sample(0:4,4,rep=T),2,2)
  s<-rbind(s1,s2)
  dd<-distsegmenttosegment(s1,s2)
  plot(s,cex=.5,main=paste0("Distance: ", signif(dd,3)),asp=1,xlim=range(s),ylim=range(s),xaxt='n',yaxt='n',xlab=
  points(s1,type="l",lwd=2)
  points(s2,type="l",lwd=2)
  cc<-closestpointstwowsegments_n(s1,s2)
  for(ccc in cc){points(ccc ,col="red",cex=2)
  points(ccc,type="l",col="red",lty=3)}
}

par(oma=c(0,0,0,0),mfrow=c(3,3))
set.seed(3);replicate(9,zz())
closestpointstwowsegments_n(matrix(c(0,3,0,0),2,2),matrix(c(1,2,0,0),2,2))
```

distpointtopoly	<i>computes the distance between a point and a polygon</i>
-----------------	--

Description

computes the distance between a point and a polygon

Usage

```
distpointtopoly(p, .poly)
```

Arguments

p a numeric vector of length 2

.poly a n x2 numeric matrix, representing a polygon. Each row of the matrix are the coordinates of a verticeof the polygon.

Examples

```

zz<-function(){
p<-sample(0:6,2,rep=T)
.poly<-matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
plot(rbind(.poly,p),
xlab="",ylab="",
cex=.2,main=paste0("Distance: ",signif(distpointtopoly(p,.poly),3)))
points(.poly,type='l')
points(x=p[1],y=p[2] ,col="red",cex=1)
points(closestpointonpolygon(p,.poly)[1],closestpointonpolygon(p,.poly)[2],col="red",cex=1)
points(rbind(p,closestpointonpolygon(p,.poly)),col="red",lty=3,type='l')
par(mfrow=c(3,3),oma=c(0,0,1,0),mar=c(2,2.1,1,0.1))
set.seed(1);replicate(9,zz())

```

distpointtoseg	<i>computes the distance between a point and a segment</i>
----------------	--

Description

computes the distance between a point and a segment

Usage

```
distpointtoseg(p, s)
```

Arguments

p	a numeric vector of length 2
s	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

Examples

```

zz<-function(p){
s=matrix(c(0,3,0,0),2,2)
plot(s,type="l",xlim=c(-2,5),ylim=c(-2,2),
xlab="",ylab="",
xaxt='n',yaxt='n')
points(x=p[1],y=p[2] ,col="red",cex=.5)
points(closestpointonseg(p,s)[1],closestpointonseg(p,s)[2],col="red",cex=.5)
segments(x0 = p[1],y0=p[2],x1=closestpointonseg(p,s)[1],y1=closestpointonseg(p,s)[2],col="red")
text((closestpointonseg(p,s)[1]+p[1])/2,(closestpointonseg(p,s)[2]+p[2])/2,round(distpointtoseg(p,s),2))
par(mfrow=c(3,3))
set.seed(1);replicate(9,zz(c(sample(-2:3,1),sample(-2:2,1))))

```

distpolytopoly	<i>Computes distance between two polygons</i>
----------------	---

Description

Computes distance between two polygons

Usage

distpolytopoly(poly1, poly2)

Arguments

- poly1 a polygon (a n x 2 numerical matrix)
- poly2 a polygon (a n x 2 numerical matrix)

Value

a positive number, the distance between the two polygons

See Also

distsegmenttopoly

Examples

```
zz<-function(){
poly1=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
poly2=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
s<-rbind(poly1,poly2)
dd<-distpolytopoly(poly1,poly2)
plot(s,cex=.5,main=paste0("Distance: ", signif(dd,3)),asp=1,xlim=range(s),ylim=range(s),xaxt='n',yaxt='n',xlab=
points(poly1,type="l",lwd=2)
points(poly2,type="l",lwd=2)
for(cc in closestpointstwopolygons_n(poly1,poly2)){
points(cc,type="l",col="red",lty=3)}}

par(mfrow=c(2,2),oma=c(0,0,1,0),mar=c(0.1,0.1,1,0.1))
set.seed(2);replicate(4,zz())
```

distpolytopoly2	<i>Computes distance between two polygons, only works for polygons that do not intersect</i>
-----------------	--

Description

Computes distance between two polygons, only works for polygons that do not intersect

Usage

```
distpolytopoly2(poly1, poly2)
```

Arguments

poly1	a polygon (a n x 2 numerical matrix)
poly2	a polygon (a n x 2 numerical matrix)

Value

a positive number, the distance between the two polygons

distsegmenttopoly	<i>Distance between a segment and a polygon</i>
-------------------	---

Description

Distance between a segment and a polygon

Usage

```
distsegmenttopoly(s, .poly)
```

Arguments

s	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.
.poly	a polygon (a nx2 matrix, each line is a point)

Examples

```
zz<-function(){
  .poly<-matrix(sample(0:6,6,T),3,2)[c(1:3,1),]
  s<-matrix(sample(0:6,6,T),2,2)
  plot(rbind(.poly,s),xlab="",yaxt="n")
  points(.poly,type="l");points(s,type="l")
  points(closestpointsonwopolygons(s,.poly),lty=3,col="red")}
```

distsegmenttosegment *distance segment to segment*

Description

distance segment to segment

Usage

```
distsegmenttosegment(s1, s2)
```

Arguments

s1 a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

s2 a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

Value

a number

Examples

```
zz<-function(){
s1=matrix(sample(0:4,4,rep=T),2,2)
s2=matrix(sample(0:4,4,rep=T),2,2)
s<-rbind(s1,s2)
dd<-distsegmenttosegment(s1,s2)
plot(s,cex=.5,main=paste0("Distance: ", signif(dd,3)),asp=1,xlim=range(s),ylim=range(s),xaxt='n',yaxt='n',xlab=
points(s1,type="l",lwd=2)
points(s2,type="l",lwd=2)
cc<-closestpointstotwosegments(s1,s2)
points(cc,col="red",cex=2)
points(cc,type="l",col="red",lty=3)
}

par(mfrow=c(3,3),oma=c(0,0,0,0),mar=c(1.1,1.2,1,1.1))
set.seed(3);replicate(9,zz())
```

dist_areas_f	<i>Distances between hexagonal bins</i>
--------------	---

Description

Distances between hexagonal bins

Usage

```
dist_areas_f(
  U,
  delta = (range(U$x)[2] - range(U$x)[1])/100,
  h = neighbourhoods(U, delta)
)
```

Arguments

U : a dataframe containing the numerical variables x and y and preferable hexagon
 delta : needed if hexagon is not a variable of U: bins will be recomputed

Value

a named matrix

Examples

```
data(U)
dist_areas_f(U)[1:3,1:3]
dist_areas_f(U,0.03)[1:3,1:3]
```

extractpolygonsaslist	<i>converts a shapefile to list of polygons (nx2 matrices)</i>
-----------------------	--

Description

converts a shapefile to list of polygons (nx2 matrices)

Usage

```
extractpolygonsaslist(shp)
```

Generate_Discrete_Time_Epidemic
Generate epidemic

Description

Generate epidemic

Usage

```
Generate_Discrete_Time_Epidemic(
  U,
  TT,
  .distriskhalf = 5 * 10(-4),
  jumprisk = 10-6,
  delta = 0.05
)
```

Arguments

U	a data.frame
TT	an integer
.distriskhalf	a positive number(default 5*10 ⁽⁻⁴⁾)
jumprisk	=10 ⁻⁶ a positive number
delta	=0.05 a positive number

Examples

```
.distriskhalf=5*10(-4);jumprisk=10-6;delta=0.05; TT=10
UE<-Generate_Discrete_Time_Epidemic(U,3)
```

Generate_U *Generate spatial data that matches population counts*

Description

Generate spatial data that matches population counts

Usage

```
Generate_U(SpatialData, .id = NULL, .spatialobject, type = "random")
```

Arguments

SpatialData	: an object of class that includes
type	: argument to be passed to sp::spsample

Examples

```

data(parish110217popest,package="dataONS")
data("mtcty150217population",package="dataONS")
shapeData2<-dataONS::dataParishes_December_2011_Boundaries_EW_BFC()
yy<-unique(get(data(Output_Area_to_Parish_to_Local_Authority_District_December_2011_Lookup_in_England_and_Wale
names(yy)<-tolower(names(yy))
shapeData<-sp::merge(shapeData2,yy,by="par11cd",duplicateGeoms = TRUE)
parish110217popest2<-parish110217popest[
  is.element(parish110217popest$PAR11CD,
    shapeData$par11cd)&
    parish110217popest$year=="mid_2006",
    c("PAR11CD","Population")]
  names(parish110217popest2)<-tolower(names(parish110217popest2))
shapeData=sp::merge(shapeData,parish110217popest2,by="par11cd",duplicateGeoms = TRUE)
shapeData$population[is.na(shapeData$population)]<-mean(shapeData$population,na.rm=TRUE)
shapeData<-subset(shapeData,is.element(lad11nm ,c("Allerdale", "Barrow-in-Furness", "Carlisle", "C

U<-Generate_U(shapeData,.id="par11cd",.spatialobject="st_areasha",type="random")
popbins<-quantile(shapeData$population,(seq_len(11)-1)/10)
poppal <- colorBin(heat.colors(5), bins=popbins, na.color = "#aaff56",reverse = T)
library(leaflet)

leaflet(U) %>%
  addPolygons(data=shapeData,
    stroke=TRUE,
    weight=1,
    color="black",
    fillOpacity=5,
    fillColor=~poppal(shapeData$population)) %>%
  addTiles() %>%
  addLegend(title = "Population count", pal=poppal,
    values=shapeData$population,
    opacity=1,
    na.label = "Not Available")

```

generate_U2

*Generate a population of avocado trees***Description**

Generate a population of avocado trees

Usage

```
generate_U2()
```

Interactive2	<i>runCompare</i>
--------------	-------------------

Description

Shiny App to

Usage

Interactive2()

Examples

```
package1<-NULL  
package2<-NULL  
runCompare()
```

interactive_1	<i>runCompare</i>
---------------	-------------------

Description

Shiny App to

Usage

interactive_1()

Examples

```
package1<-NULL  
package2<-NULL  
runCompare()
```

neighbourhoods	<i>hexagonal bins</i>
----------------	-----------------------

Description

hexagonal bins

Usage

```
neighbourhoods(  
  U,  
  delta = (range(U$x, na.rm = TRUE)[2] - range(U$x, na.rm = TRUE)[1])/100  
)
```

Arguments

U : a dataframe containing the numerical variables x and y
delta: controls the bin diameter

Value

a hexbin object hexagonal bins

Examples

```
# plot the hex bins of cumbria  
data(U,package="Strategy")  
library("hexbin")  
plot(neighbourhoods(U,.1))  
plot(neighbourhoods(U,.01))  
plot(neighbourhoods(U,.001))
```

newdist	<i>compute distances between new infected and exposed</i>
---------	---

Description

compute distances between new infected and exposed

Usage

```
newdist(
  closedistances = NULL,
  U,
  sick,
  new.sicks = NULL,
  delta = 0.005,
  dist_areas = dist_areas_f(U, delta)
)
```

Arguments

closedistances	NULL, or a named list with 2 named elements: closedistances\$ra, closedistances\$id
U	a data.frame with the variables hexagon (can be any bin identifier), x, y : coordinates,
sicks	a vector of integers
new.sicks	a vector of integers
delta	a positive number : a threshold
dist_areas:	a function between

Value

NULL, or a named list with 2 named elements: closedistances\$ra, closedistances\$id

Examples

```
data(UE,package="Strategy")
delta<-.005
sicks<-(1:nrow(UE))[UE$I001=="sick"]
closedistances=newdist(NULL,UE,sicks)
do.call(cbind,closedistances)[1:3,]
```

polydistmat

Compute distance matrix for a list of polygons

Description

Compute distance matrix for a list of polygons

Usage

```
polydistmat(list.poly)
```

Arguments

list.poly	a list of nx2 numeric matrices
-----------	--------------------------------

Value

a $(n*(n-1)/2) \times 3$ matrix

Examples

```
zz<-function(){
  list.poly=plyr::alply(cbind(rep(0:8,9),rep(0:8,each=9))[sample(81,4),],1,function(x){
    cbind(x[1]+c(0,0,.5,.5,0),x[2]+c(0,.5,.5,0,0))})
  gradients=cbind(c(0,1),c(1,0),c(1,1),c(1,-1))
  par(mfrow=c(1,1),oma=c(0,0,0,0),mar=c(0.1,0.1,.1,0.1))
  plot(do.call(rbind,list.poly),xlab="",yaxt="n",ylab="",cex=.1)
  for(i in 1:length(list.poly)){.poly=list.poly[[i]]
    points(.poly,type="l")
    text(mean(.poly[,1]),mean(.poly[,2]),as.roman(i))
  }
  X=polydistmat(list.poly)
  X<-cbind(X,floor(rank(X[,3])))
  colorlink=topo.colors(2*max(X[,4]))[X[,4]]
  for(i in 1:nrow(X)){
    cc<-closestpointstwopolygons(list.poly[[X[i,1]]],list.poly[[X[i,2]]])
    points(cc,col=colorlink[i],type="l",lty=3)
    text(mean(cc[,1]),mean(cc[,2]),signif(X[i,3],3))
  }
  colnames(X)<-c("polygon 1","polygon 2", "distance","col")
  X[,1:3]}
  set.seed(1);zz()
```

polysmalldistmat

Compute distance matrix for a list of polygons

Description

Compute distance matrix for a list of polygons

Usage

```
polysmalldistmat(
  list.poly,
  delta,
  gradients = apply(cbind(c(0, 1), c(1, 0), c(1, 1), c(1, -1)), 2, function(x) {
    x/(sqrt(sum(x^2))) })
)
```

Arguments

`list.poly` a list of nx2 numeric matrices
`delta` a positive number
`gradients` a 2x n matrix each column representing a vector.

Value

a $(n*(n-1)/2) \times 3$ matrix

Examples

```
zz<-function(delta){
  list.poly=plyr::alply(cbind(rep(0:8,9),rep(0:8,each=9))[sample(81,20),],1,function(x){
    cbind(x[1]+c(0,0,.5,.5,0),x[2]+c(0,.5,.5,0,0)))
    gradients=cbind(c(0,1),c(1,0),c(1,1),c(1,-1))
    par(mfrow=c(1,1),oma=c(0,0,1,0),mar=c(0.1,0.1,1,0.1))
    plot(do.call(rbind,list.poly),xlab="",yaxt="n",ylab="",cex=.1,main=paste0("Match polygons distant less than ",de
    for(.poly in list.poly){points(.poly,type="l")}
    X=polysmallldistmat(list.poly,delta)
    for(i in 1:nrow(X)){
      points(closestpointson twopolygons(list.poly[[X[i,1]]],list.poly[[X[i,2]]]),col="red",type="l",lty=3)
    }
    set.seed(1);zz(.5)
    set.seed(1);zz(1)
    set.seed(1);zz(2)
```

projpointonseg_a	<i>computes the position of a projected point in the basis formed by a segment</i>
------------------	--

Description

computes the position of a projected point in the basis formed by a segment

Usage

```
projpointonseg_a(p, s, method = "euclidean")
```

Arguments

p	a numeric vector of length 2
s	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of a extreme point of the segment.

Examples

```
zz<-function(p){
  s=matrix(c(0,1,0,0),2,2)
  plot(s,type="l",xlim=c(-.5,1.5),
  xlab="",ylab="",
  yaxt='n',yaxt='n')
  points(x=p[1],y=p[2],col="red",cex=2)
  points(closestpointonseg(p,s)[1],closestpointonseg(p,s)[2],col="red",cex=2)
  segments(x0 = p[1],y0=p[2],x1=closestpointonseg(p,s)[1],y1=closestpointonseg(p,s)[2],col="red")
  text(projpointonseg_a(p,s),-.8,paste0("a=",projpointonseg_a(p,s)))}
```

```
par(oma=c(0,0,0,0),mfrow=c(1,4))
zz(c(-.5,1))
zz(0:1)
zz(c(.5,1))
zz(c(1.5,1))
```

<code>ranges.gap</code>	<i>Interval length between two ranges (0 if they overlap)</i>
-------------------------	---

Description

Interval length between two ranges (0 if they overlap)

Usage

```
ranges.gap(r1, r2)
```

Arguments

<code>r1</code>	a range a length 2 numerical vector
<code>r2</code>	a range a length 2 numerical vector

Examples

```
par(mfrow=c(1,4),oma=c(0,0,0,0))
set.seed(10);replicate(4,(function(){
  r1<-sample(1:8,2);r2<-sample(1:5,2)
  plot(cbind(c(r1,r2),0),yaxt='n',xlab='',ylab='',
  main=paste0("Gap: ",ranges.gap(r1,r2)))
  points(cbind(r1,0),type="l",lwd=3)
  points(cbind(r2,0),type="l",col="red")
}))()
```

<code>rangesbygradients_f</code>	<i>Compute range of a polygon along certain gradients</i>
----------------------------------	---

Description

Compute range of a polygon along certain gradients

Usage

```
rangesbygradients_f(
  .poly,
  gradients = -apply(cbind(c(0, 1), c(1, 0), c(1, 1), c(1, -1)), 2, function(x) {
    x/(sqrt(sum(x^2))) })
)
```

Arguments

`gradients` a 2x n matrix each column representing a vector.

`list.poly` a list of nx2 numeric matrices

`delta` a positive number

Examples

```
a=c(1-1/sqrt(2),1/sqrt(2));.poly=cbind(1+c(0,0,a,1,1,a[2:1]),1+c(a,1,1,a[2:1],0,0))[c(1:8,1),]
plot(.poly,type='l',xlab='',ylab='')
rangesbygradients_f(.poly,gradients=cbind(c(0,1),c(1,0),c(1,-1)))
```

<code>rangesoverlap</code>	<i>tells if two ranges overlap</i>
----------------------------	------------------------------------

Description

tells if two ranges overlap

Usage

```
rangesoverlap(r1, r2)
```

Arguments

`r1` a range a length 2 numerical vector

`r2` a range a length 2 numerical vector

Examples

```
par(mfrow=c(1,4),oma=c(0,0,0,0))
set.seed(8);replicate(4,(function(){
r1<-sample(1:8,2);r2<-sample(1:5,2)
plot(cbind(c(r1,r2),0),yaxt='n',xlab='',ylab='',xaxt='n',
main=paste0(if(rangesoverlap(r1,r2)){"0"}else{"Does not o"},"verlap"))
points(cbind(r1,0),type="l",lwd=3)
points(cbind(r2,0),type="l",col="red")
}))()
```

risktobeinfected	<i>Computes the risk to be infected</i>
------------------	---

Description

Computes the risk to be infected

Usage

```
risktobeinfected(
  U,
  closedistances = NULL,
  sick,
  new.sicks = NULL,
  .distriskhalf = 5 * 10^(-4),
  jumprisk = 10^-6,
  delta = 0.01,
  previouslyexposed = c(),
  previousrisk = NULL
)
```

Examples

```
y=rep("Sane",nrow(U));y[sample(length(y),10)]<-"sick"
jumprisk=10^-6
.distriskhalf=10^-6
```

risktobeinfectedbydistancetoallinfectedunit	<i>Risk to be infected by many neighbours a neighbour at a certain distance</i>
---	---

Description

Risk to be infected by many neighbours a neighbour at a certain distance

Usage

```
risktobeinfectedbydistancetoallinfectedunit(
  .dist,
  nI,
  .distriskhalf = 5 * 10^(-4),
  jumprisk = 10^-6
)
```

Arguments

.dist : a vector (distances)
 .disthalfrisk : distance for which the risk is one half
 nI: total number of infected
 jumprisk: probability to be infected by one person, no matter how far he/she is

Value

$1 - (\text{prod}(1 - \text{risktobeinfectedbydistancetooneinfectedunit}(\text{.dist}, \text{distriskhalf})) * (1 - \text{jumprisk})^{\text{nI}})$

Examples

```
#Risk to be ingfected 2 m from the victim when the 50%risk distance is 1 m:
risktobeinfectedbydistancetooneinfectedunit(2,1)
```

risktobeinfectedbydistancetooneinfectedunit

Risk to be infected by a neighbour at a distance x

Description

Risk to be infected by a neighbour at a distance x

Usage

```
risktobeinfectedbydistancetooneinfectedunit(.dist, .distriskhalf = 5 * 10^(-4))
```

Arguments

.dist : a distance
 .disthalfrisk : distance for which the risk is one half

Value

$\exp(-\text{.dist}/(\log(2) * \text{distriskhalf}))$

Examples

```
#Risk to be ingfected 2 m from the victim when the 50%risk distance is 1 m:
risktobeinfectedbydistancetooneinfectedunit(2,1)
```

segment.intersect	<i>test if two segments intersect</i>
-------------------	---------------------------------------

Description

test if two segments intersect

Usage

```
segment.intersect(s1, s2)
```

Arguments

s1	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of an extreme point of the segment.
s2	a 2x2 numeric matrix, representing a segment. Each row of the matrix are the coordinates of an extreme point of the segment.

Examples

```
zz<-function(s1=matrix(sample(0:3,4,rep=T),2,2),s2=matrix(sample(0:3,4,rep=T),2,2)){
  si<-segment.intersect(s1,s2)
  s<-rbind(s1,s2)
  plot(s,cex=.5,main=if(si){"Intersect"}else{"Disjoint"},xlab="",ylab="",xaxt='n', yaxt='n',xlim=range(s)+c(-1,1),
  points(s1,type="l")
  points(s2,type="l")
  text(s[,1],s[,2],toupper(letters[1:4]),cex=1,col="red")
}
par(mfrow=c(3,4),mar=c(5,3,2,2))
set.seed(12);replicate(4,zz())
zz(matrix(0,2,2),matrix(0,2,2))
zz(matrix(0,2,2),matrix(1,2,2))
zz(matrix(c(1,1,1,1),2,2),matrix(c(2,3,2,3),2,2))
zz(matrix(c(1,1,0,0),2,2),matrix(c(0,1,0,0),2,2))
zz(matrix(c(1,3,1,3),2,2),matrix(c(2,4,2,4),2,2))
zz(matrix(c(0,1,0,1),2,2),matrix(c(2,3,2,3),2,2))
zz(matrix(c(0,4,0,4),2,2),matrix(c(1,3,1,3),2,2))
zz(matrix(c(0,1,0,1),2,2),matrix(c(0,3,0,3),2,2))
```

shinyapp1	<i>Shiny app 1</i>
-----------	--------------------

Description

launches shiny app 1

Usage

```
shinyapp1()
```

subsets

All ordered n-sized subsets of 1...N n

Description

All ordered n-sized subsets of 1...N n

Usage

```
subsets(
  n,
  N,
  subsets_1 = if (n >= 2) { subsets(n - 1, N) } else { NULL }
)
```

Examples

```
subsets(1,10)
subsets(2,10)
subsets(3,10)
subsets(8,10)
subsets(9,10)
subsets(10,10)
```

triangleorientation

computes the orientation of a triangle

Description

computes the orientation of a triangle

Usage

```
triangleorientation(s)
```

Arguments

s a 3x2 numeric matrix, representing a triangle. Each row of the matrix are the coordinates of an extreme point of the triangle.

Examples

```

zz<-function(p){
s=matrix(sample(0:4,6,rep=T),3,2)
plot(s[c(1:3,1),],type="l",main=if(triangleorientation(s)==1){"+"}else{if(triangleorientation(s)==-1){"-"}else{
text(s[,1],s[,2],toupper(letters[1:3]),cex=1,col="red")
}
}
par(mfrow=c(2,2),mar=c(5,3,2,2))
set.seed(7);replicate(4,zz(c(sample(-2:3,1),sample(-2:2,1))))

```

updatedist	<i>update the list of the already nown distances between subjects with distances between new infected and exposed</i>
------------	---

Description

update the list of the already nown distances between subjects with distances between new infected and exposed

Usage

```

updatedist(
  closeddistances = NULL,
  U,
  sick,
  new.sicks = NULL,
  delta = 0.005,
  dist_areas = dist_areas_f(U, delta)
)

```

Arguments

closeddistances	NULL, or a named list with 2 named elements: closeddistances\$ra, closeddistances\$id
U	a data.frame with the variables hexagon (can be any bin identifier), x, y : coordinates,
sicks	a vector of integers indicating the row numbers in U for sick
new.sicks	a vector of integers indicating the row numbers in U for new sick
delta	a positive number : a threshold
dist_areas:	a function between

Value

NULL, or a named list with 2 named elements: closeddistances\$ra, closeddistances\$id

Examples

```
data(UE,package="Strategy")
delta<-.005
sicks<-(1:nrow(UE))[UE$I001=="sick"]
closedistances=updatedist(NULL,UE,sicks)
do.call(cbind,closedistances)[1:3,]
```

Index

`addpiechartclustermarkers`, [2](#)

`closestpointonpolygon`, [3](#)
`closestpointonseg`, [4](#)
`closestpointstwopolygons`, [4](#)
`closestpointstwopolygons_n`, [5](#)
`closestpointstwosegments`, [6](#)
`closestpointstwosegments_n`, [6](#)

`dist_areas_f`, [12](#)
`distpointtopoly`, [7](#)
`distpointtoseg`, [8](#)
`distpolytopoly`, [9](#)
`distpolytopoly2`, [10](#)
`distsegmenttopoly`, [10](#)
`distsegmenttosegment`, [11](#)

`extractpolygonsaslist`, [12](#)

`Generate_Discrete_Time_Epidemic`, [13](#)
`Generate_U`, [13](#)
`generate_U2`, [14](#)

`Interactive2`, [15](#)
`interactive_1`, [15](#)

`neighbourhoods`, [16](#)
`newdist`, [16](#)

`polydistmat`, [17](#)
`polysmalldistmat`, [18](#)
`projpointonseg_a`, [19](#)

`ranges.gap`, [20](#)
`rangesbygradients_f`, [20](#)
`rangesoverlap`, [21](#)
`risktobeinfected`, [22](#)
`risktobeinfectedbydistancetoallinfectedunit`,
[22](#)
`risktobeinfectedbydistancetooneinfectedunit`,
[23](#)

`segment.intersect`, [24](#)
`shinyapp1`, [24](#)
`subsets`, [25](#)

`triangleorientation`, [25](#)

`updatedist`, [26](#)