# Package 'Strategy'

March 7, 2020

**Type** Package

**Title** X

**Version** 1.0

**Date** 2020-03-07

**Author** D. Bonnery

**Maintainer** D. Bonnery <dbonnery@umd.edu>

**Description** Data

**Depends** ggplot2,
leaflet,
spatstat,
sf

**License** GPL (>= 2)

**LazyLoad** yes

**LazyData** true

**RoxygenNote** 7.0.2

## R topics documented:

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 |

### Examples

```
data("breweries91",package="leaflet")
breweries91$goodbear<-sample(as.factor(c("terrific","marvelous","culparterretaping")),nrow(breweries91),replac
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")
```

---

distpointtoseg                 *computes the distance between a point and a segment*

---

## 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))
points(x=p[1],y=p[2] ,col="red",cex=.5)
points(projpointonseg(p,s)[1],projpointonseg(p,s)[2],col="red",cex=.5)
segments(x0 = p[1],y0=p[2],x1=projpointonseg(p,s)[1],y1=projpointonseg(p,s)[2],col="red")
text((projpointonseg(p,s)[1]+p[1])/2,(projpointonseg(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                 *Computes distance between two polygons*

---

## 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

**Examples**

```
polys=lapply(c(0:1),function(x){
cbind(c(x,x,x+.5,x+.5,x),c(0,1,1,0,0))})
par(mfrow=c(1,1))
plot(do.call(rbind,polys),xlab="",yaxt="n")
for(.poly in polys){segments(x0 = .poly[-5,1],y0 = .poly[-5,2],.poly[-1,1],.poly[-1,2])}
distpolytopoly(polys[[1]],polys[[2]])
polys=lapply(c(1:2),function(x){
cbind(c(-x,-x,x,x,-x),c(-x,x,x,-x,-x))})
par(mfrow=c(1,1))
plot(do.call(rbind,polys),xlab="",yaxt="n")
for(.poly in polys){segments(x0 = .poly[-5,1],y0 = .poly[-5,2],.poly[-1,1],.poly[-1,2])}
distpolytopoly(polys[[1]],polys[[2]])
polys=lapply(c(1:2),function(x){
cbind(c(-2,-2,2,2,-2),c(-1,1,1,-1,-1))[,c(x,(1:2)[-x])]})
par(mfrow=c(1,1))
plot(do.call(rbind,polys),xlab="",yaxt="n")
for(.poly in polys){segments(x0 = .poly[-5,1],y0 = .poly[-5,2],.poly[-1,1],.poly[-1,2])}
distpolytopoly(polys[[1]],polys[[2]])
```

---

distsegmenttopoly              *Distance between a segment and a polygon*

---

**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**

```
data(Avo_fields,package="Strategy")
polygon1<-Avo_fields[1,]
A<-polygon1@polygons
B<-A[[1]]@Polygons[[1]]@coords
s<-cbind(runif(2,min = min(B[,1]),max=max(B[,1])),runif(2,min=min(B[,2]),max=max(B[,2])))
plot(B,type='l')
```

```
s1<-s
points(s,type="l",lwd=4,col="green")
x<-vector()
for(i in 1:(nrow(B)-1)){
s2<-B[(i:(i+1)),]
dd<-distsegmenttosegment(s1,s2)
l<-which(c(distpointtoseg(s1[1,],s2),distpointtoseg(s1[2,],s2),distpointtoseg(s2[1,],s1),distpointtoseg(s2[2,
min(as.matrix(dist(rbind(s1,s2),diag=T,upper = T))[3:4,1:2]);dd
sk<-if(l<=2){s2}else{s1}
p=rbind(s1,s2)[l,]
points(x=p[1],y=p[2] ,col="red",cex=2)
points(projpointonseg(p,sk)[1],projpointonseg(p,sk)[2],col="red",cex=2)
segments(x0 = p[1],y0=p[2],x1=projpointonseg(p,sk)[1],y1=projpointonseg(p,sk)[2])
x=c(x,dd)
}
min(x)
distsegmenttopoly(s,B)
distpolytopoly()
```

---

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))
```

```
points(s1,type="l",lwd=4,col="green")
points(s2,type="l",lwd=4,col="blue")
if(dd>0){l<-which(c(distpointtoseg(s1[1,],s2),distpointtoseg(s1[2,],s2),distpointtoseg(s2[1,],s1),distpointtos
s<-if(l<=2){s2}else{s1}
p=rbind(s1,s2)[l,]
points(x=p[1],y=p[2] ,col="red",cex=2)
points(projpointonseg(p,s)[1],projpointonseg(p,s)[2],col="red",cex=2)
segments(x0 = p[1],y0=p[2],x1=projpointonseg(p,s)[1],y1=projpointonseg(p,s)[2],lty=3)}}

par(mfrow=c(3,3))
set.seed(3);replicate(9,zz())
```

---

dist_areas_f                    *Distances between hexagonal bins*

---

### 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)
delta<-0.01
```

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")
```

---

neighbourhoods                       *hexagonal bins*

---

## 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)
plot(neighbourhoods(U,.1))
plot(neighbourhoods(U,.01))
plot(neighbourhoods(U,.001))
```

---

newdist      *compute distances between new infected and exposed*

---

## Description

compute distances between new infected and exposed

## Usage

```
newdist(
  closedistances = NULL,
  U,
  sicks,
  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,

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"]
closedistance=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)x 3 matrix

### Examples

```
polys=lapply(c(0:3,5:7),function(x){
cbind(c(x,x,x+.5,x+.5,x),c(0,1,1,0,0))})
par(mfrow=c(1,1))
plot(do.call(rbind,polys),xlab="",yaxt="n")
for(.poly in polys){segments(x0 = .poly[-5,1],y0 = .poly[-5,2],.poly[-1,1],.poly[-1,2])}
polydistmat(polys)
#data(Avo_fields,package="Strategy")
#polygons<-Avo_fields
#MM<-polydistmat(extractpolygonsaslist(Avo_fields))
#parallel::detectCores()
#save(MM,file=file.path(Mydirectories::googledrive.directory(),"Travail/Recherche/Travaux/Epidemiologie/Strate
```

---

| projpointonseg | *computes the position of a projected point in the basis formed by a segment* |
|---|---|

---

### Description

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

### Usage

```
projpointonseg(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))
points(x=p[1],y=p[2] ,col="red",cex=.5)
points(projpointonseg(p,s)[1],projpointonseg(p,s)[2],col="red",cex=.5)
segments(x0 = p[1],y0=p[2],x1=projpointonseg(p,s)[1],y1=projpointonseg(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))))
```

---

| projpointonseg_a | *computes the position of a projected point in the basis formed by a segment* |
|---|---|

---

### Description

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

### Usage

```
projpointonseg_a(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

```
data(Avo_fields,package="Strategy")
polygon1<-Avo_fields[1,]
A<-polygon1@polygons
B<-A[[1]]@Polygons[[1]]@coords
i=sample(nrow(B)-1,1)
s<-B[i:(i+1),]
p<-c(runif(1,min = 142.162,max=142.165),runif(1,min=-34.171,max=-34.167))
plot(B,type='l')
points(s,type="l",lwd=4,col="red")
points(x=p[1],y=p[2] ,col="red",cex=2)
points(projpointonseg(p,s)[1],projpointonseg(p,s)[2],col="red",cex=2)
segments(x0 = p[1],y0=p[2],x1=projpointonseg(p,s)[1],y1=projpointonseg(p,s)[2])
projpointonseg_a(p,s)
distpointtoseg(p,s)
dist(rbind(p,projpointonseg(p,s)))

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

---

risktobeinfected *Computes the risk to be infected*

---

## Description

Computes the risk to be infected

## Usage

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

## Examples

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

---

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

---

## 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-(prod(1-risktobeinfectedbydistancetooneinfectedunit(.dist,.distriskhalf))*(1-jumprisk)^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(-.dist/(log(2)*.distriskhalf))

### Examples

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

---

segment.intersect　　　　*test if two segments intersect*

---

### 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))
```

---

| test | *runCompare* |
|------|--------------|

---

## Description

Shiny App to

## Usage

```
test()
```

## Examples

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

---

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

---

**Description**

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

**Usage**

```
updatedist(
  closedistances = NULL,
  U,
  sicks,
  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$ra, closedis-
                tances$id

U               a data.frame with the variables hexagon (can be any bin identifier), x, y : coor-
                dinates,

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"]
closedistance=updatedist(NULL,UE,sicks)
do.call(cbind,closedistances)[1:3,]
```

# Index