Package 'Strategy'

March 6, 2020

Type Package

Title X

Version 1.0

Date 2020-03-06

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() %>%
addTiles() %>%
addpiechartclustermarkers(breweries91,.colors=c("red","green","blue"),group="goodbear")
```

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
h<-neighbourhoods(U,delta)
U$hexagon<-paste0(h@cID)
hD<-dist_areas_f(U,h)

sss1=sample(nrow(U),1000)
sss2=sample(nrow(U),1000)
x=sapply(1:1000,function(i){dist(U[c(sss1[i],sss2[i]),c("x","y")])})
y<-sapply(1:1000,function(i){hD[U$hexagon[sss1[i]],U$hexagon[sss2[i]]]})
plot(x,y,pch=".")
```

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))
```

`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)
```

test *runCompare*

Description

Shiny App to

Usage

```
test()
```

Examples

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

Index