# Package ‘SpatialPosition’

May 6, 2015

**Title**  Spatial Position Models  
**Version**  0.9  
**Description**  Computes spatial position models: Stewart potentials, Reilly catchment areas, Huff catchment areas.  
**Depends**  R (>= 3.0.0), raster  
**License**  GPL-3  
**LazyData**  true  
**Imports**  sp  
**Suggests**  rgeos, knitr  
**VignetteBuilder**  knitr  
**NeedsCompilation**  no  
**Author**  Timothee Giraud [cre, aut],  
\quad Hadrien Commenges [aut],  
\quad Joel Boulier [ctb]  
**Maintainer**  Timothee Giraud <timothee.giraud@ums-riate.fr>  
**Repository**  CRAN  
**Date/Publication**  2015-05-06 14:46:24

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>contourStewart</td>
<td>2</td>
</tr>
<tr>
<td>CreateDistMatrix</td>
<td>3</td>
</tr>
<tr>
<td>CreateGrid</td>
<td>4</td>
</tr>
<tr>
<td>huff</td>
<td>5</td>
</tr>
<tr>
<td>plotHuff</td>
<td>6</td>
</tr>
<tr>
<td>plotReilly</td>
<td>7</td>
</tr>
<tr>
<td>plotStewart</td>
<td>8</td>
</tr>
<tr>
<td>rasterHuff</td>
<td>9</td>
</tr>
<tr>
<td>rasterReilly</td>
<td>10</td>
</tr>
<tr>
<td>rasterStewart</td>
<td>11</td>
</tr>
<tr>
<td>reilly</td>
<td>12</td>
</tr>
</tbody>
</table>
contourStewart

Create a SpatialPolygonsDataFrame or a SpatialLinesDataFrame from a Stewart Raster

Description

This function creates a SpatialPolygonsDataFrame or a SpatialLinesDataFrame from the Stewart raster.

Usage

`contourStewart(x, breaks, type = "line")`

Arguments

- `x`: raster; output of the `rasterStewart` function.
- `breaks`: numeric; a vector of break values.
- `type`: character; "poly" or "line". WARNING: the poly option is experimental and needs the rgeos package.

Value

The output of the function is a SpatialPolygonsDataFrame (`type = "poly"`) or a SpatialLinesDataFrame (`type = "line"`).

See Also

`stewart, rasterStewart, plotStewart, contourStewart, CreateGrid, CreateDistMatrix`.

Examples

```r
data(spatData)
# Compute Stewart potentials from known points (spatPts) on a
# grid defined by its resolution
mystewart <- stewart(knownpts = spatPts, varname = "Capacite",
                     typefct = "exponential", span = 1000, beta = 3,
                     resolution = 50, longlat = FALSE,
                     mask = spatMask)
# Create a raster of potentials values
mystewartraster <- rasterStewart(x = mystewart, mask = spatMask)
# Display the raster and get break values
```
CreateDistMatrix

Description

This function creates a distance matrix between two sp objects (SpatialPointsDataFrame or SpatialPolygonsDataFrame).

Usage

CreateDistMatrix(knownpts, unknownpts, longlat = FALSE, bypassctrl = FALSE)

Arguments

knownpts  sp object; rows of the distance matrix.
unknownpts  sp object; columns of the distance matrix.
longlat  logical; euclidean distance (FALSE, default) or Great Circle distance (TRUE).
bypassctrl  logical; bypass the distance matrix size control (see Details).

Details

The function returns a full matrix of distances in the metric of the points if longlat is FALSE, or in kilometers if longlat is TRUE. This is a wrapper for the spdists function.

If the matrix to compute is too large (more than 100,000,000 cells or more than 10,000,000 origins or destinations) the function sends a confirmation message to warn users about the amount of RAM mobilized. Use bypassctrl = TRUE to skip this control.

Value

A distance matrix, row names are knownpts row names, column names are unknownpts row names.
CreateGrid

Create a Regularly Spaced SpatialPointsDataFrame

Description

This function creates a regular grid of SpatialPointsDataFrame from the extent of a given sp object and a given resolution.

Usage

CreateGrid(w, resolution)

Arguments

w
sp object; the spatial extent of this object is used to create the regular SpatialPointsDataFrame.

resolution
numeric; resolution of the grid (in map units).

Value

The output of the function is a SpatialPointsDataFrame of regularly spaced points with the same extent as w.

See Also

CreateDistMatrix.
Examples

```r
# Create a SpatialPointsDataFrame grid of spatMask extent and 200 meters
# resolution
data(spatData)
mygrid <- CreateGrid(w = spatMask, resolution = 200)
plot(mygrid, cex = 0.1, pch = ".")
plot(spatMask, border = "red", lwd = 2, add = TRUE)
```

Description

This function computes the catchment areas as defined by D. Huff (1964).

Usage

```r
huff(knownpts, unknownpts = NULL, matdist = NULL, varname, typefct = "exponential", span, beta, resolution = 2000, longlat = FALSE, mask = NULL)
```

Arguments

- **knownpts**: sp object (SpatialPointsDataFrame or SpatialPolygonsDataFrame); this is the set of known observations to estimate the catchment areas from.
- **unknownpts**: sp object (SpatialPointsDataFrame or SpatialPolygonsDataFrame); this is the set of unknown units for which the function computes the estimates. Not used when resolution is set up. (optional)
- **matdist**: matrix; a distance matrix. Row names match the first column of the knownpts object dataframe. Column names match the first column of the unknownpts object dataframe. (optional)
- **varname**: character; name of the variable in the knownpts dataframe from which values are computed. Quantitative variable with no negative values.
- **typefct**: character; spatial interaction function. Options are "pareto" (default, means power law) or "exponential". If "pareto" the interaction is defined as: \((1 + \alpha \cdot \text{mDistance}) ^ (-\beta)\). If "exponential" the interaction is defined as: \(\exp(-\alpha \cdot \text{mDistance} ^ \beta)\). The alpha parameter is computed from parameters given by the user (beta and span).
- **span**: numeric; distance where the density of probability of the spatial interaction function equals 0.5.
- **beta**: numeric; impedance factor for the spatial interaction function.
- **resolution**: numeric; resolution of the output SpatialPointsDataFrame (in map units).
- **longlat**: logical; euclidean distance (FALSE, default) or Great Circle distance (TRUE). If TRUE inputs are expected in the WGS84 reference system.
- **mask**: sp object; the spatial extent of this object is used to create the regularly spaced SpatialPointsDataFrame output. (optional)
Details

If unknownpts is NULL then resolution must be used.

Value

SpatialPointsDataFrame with the computed catchment areas in a new field named OUTPUT.

References


See Also

huff, rasterHuff, plotHuff, CreateGrid, CreateDistMatrix.

Examples

# Create a SpatialPointsDataFrame grid of spatMask extent and 200 meters
# resolution
data(spatData)
mygrid <- CreateGrid(w = spatMask, resolution = 200)
# Create a distance matrix between known points (spatPts) and mygrid
mymat <- CreateDistMatrix(knownpts = spatPts, unknownpts = mygrid,
                           longlat = FALSE)
# Compute Huff catchment areas from known points (spatPts) on a given
# grid (mygrid) using a given distance matrix (mymat)
myhuff <- huff(knownpts = spatPts, unknownpts = mygrid,
                matdist = mymat, varname = "Capacite",
                typefct = "exponential", span = 1250,
                beta = 3, longlat = FALSE, mask = spatMask)
# Compute Huff catchment areas from known points (spatPts) on a
# grid defined by its resolution
myhuff2 <- huff(knownpts = spatPts, varname = "Capacite",
                 typefct = "exponential", span = 1250, beta = 3,
                 resolution = 200, longlat = FALSE, mask = spatMask)
# The two methods have the same result
identical(myhuff, myhuff2)
# The function output a SpatialPointsDataFrame
class(myhuff)

plotHuff

Plot a Huff Raster

Description

This function plots the raster produced by the rasterHuff function.

Usage

plotHuff(x, add = FALSE)
**plotReilly**

**Description**

This function plots the raster produced by the `rasterReilly` function.

**Usage**

```r
plotReilly(x, add = FALSE, col = rainbow)
```

**Arguments**

- `x` raster; output of the `rasterReilly` function.
- `add` logical; if TRUE the raster is added to the current plot, if FALSE the raster is displayed in a new plot.
- `col` function; color ramp function, such as `colorRampPalette`.

**Details**

Display the raster nicely.

**Examples**

```r
data(spatData)
# Compute Huff catchment areas from known points (spatPts) on a
# grid defined by its resolution
myhuff <- huff(knownpts = spatPts, varname = "Capacite",
               typefct = "exponential", span = 750, beta = 2,
               resolution = 50, longlat = FALSE, mask = spatMask)
# Create a raster of huff values
myhuffraster <- rasterHuff(x = myhuff, mask = spatMask)
# Plot Huff values nicely
plotHuff(x = myhuffraster)
```
plotStewart

Plot a Stewart Raster

Description

This function plots the raster produced by the `rasterStewart` function.

Usage

```r
plotStewart(x, add = FALSE, breaks = NULL, typec = "equal", nclass = 5,
            legend.rnd = 0, col = colorRampPalette(c("#FEA3A3", 
                                                
                "#900000")))
```

Arguments

- **x**: raster; output of the `rasterStewart` function.
- **add**: logical; if TRUE the raster is added to the current plot, if FALSE the raster is displayed in a new plot.
- **breaks**: numeric; vector of break values to map. If used, this parameter overrides `typec` and `nclass` parameters.
- **typec**: character; either "equal" or "quantile", how to discretize the values.
- **nclass**: numeric (integer), number of classes.
- **legend.rnd**: numeric (integer); number of digits used to round the values displayed in the legend.
- **col**: function; color ramp function, such as `colorRampPalette`.

Value

Display the raster nicely and return the list of break values (invisible).
rasterHuff

See Also

stewart, rasterStewart, plotStewart, contourStewart, CreateGrid, CreateDistMatrix.

Examples

data(spatData)
# Compute Stewart potentials from known points (spatPts) on a
# grid defined by its resolution
mystewart <- stewart(knownpts = spatPts, varname = "Capacite",
                     typefct = "exponential", span = 1000, beta = 3,
                     resolution = 50, longlat = FALSE, mask = spatMask)
# Create a raster of potentials values
mystewartraster <- rasterStewart(x = mystewart, mask = spatMask)
# Plot stewart potentials nicely
plotStewart(x = mystewartraster, add = FALSE, nclass = 5)
# Can be used to obtain break values
break.values <- plotStewart(x = mystewartraster, add = FALSE, nclass = 5)
break.values

rasterHuff

Create a Raster from a Huff SpatialPointsDataFrame

Description

This function creates a raster from a regularly spaced Huff SpatialPointsDataFrame (output of the huff function).

Usage

rasterHuff(x, mask = NULL)

Arguments

x	sp object (SpatialPointsDataFrame); output of the huff function.

mask	sp object (SpatialPolygonsDataFrame); this object is used to clip the raster. (optional)

Value

Raster of catchment areas values.

See Also

huff, rasterHuff, plotHuff, CreateGrid, CreateDistMatrix.
rasterReilly

Create a Raster from a Reilly SpatialPointsDataFrame

Description

This function creates a raster from a regularly spaced Reilly SpatialPointsDataFrame (output of the \texttt{reilly} function).

Usage

\begin{verbatim}
rasterReilly(x, mask = NULL)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{x} \hspace{1cm} \texttt{sp} object (\texttt{SpatialPointsDataFrame}); output of the \texttt{reilly} function.
\item \texttt{mask} \hspace{1cm} \texttt{sp} object (\texttt{SpatialPolygonsDataFrame}); this object is used to clip the raster. (optional)
\end{itemize}

Value

Raster of catchment areas values. The raster uses a RAT (\texttt{ratify}) that contains the correspondence between raster values and catchment areas values. Use \texttt{unique(levels(rasterName)[[1]])} to see the correspondence table.

See Also

\texttt{reilly}, \texttt{rasterReilly}, \texttt{plotReilly}, \texttt{CreateGrid}, \texttt{CreateDistMatrix}.

Examples

\begin{verbatim}
data(spatData)
row.names(spatPts) <- spatPts$CodHop
# Compute Huff catchment areas from known points (spatPts) on a
# grid defined by its resolution
myhuff <- huff(knownpts = spatPts, varname = "Capacite",
               typefct = "exponential", span = 750, beta = 2,
               resolution = 50, longlat = FALSE, mask = spatMask)
# Create a raster of huff values
myhuffraster <- rasterHuff(x = myhuff, mask = spatMask)
plot(myhuffraster)
\end{verbatim}

\begin{verbatim}
data(spatData)
rowNnames(spatpts) <- spatpts$CodHop
# Compute Reilly catchment areas from known points (spatPts) on a
# grid defined by its resolution
myreilly <- reilly(knownpts = spatPts, varname = "Capacite",
                   typefct = "exponential", span = 750, beta = 2,
                   resolution = 50, longlat = FALSE, mask = spatMask)
\end{verbatim}
# Create a raster of reilly values
myreillyraster <- rasterReilly(x = myreilly, mask = spatMask)
plot(myreillyraster, col = rainbow(18))
# Correspondance between raster values and reilly areas
head(unique(levels(myreillyraster)[[1]]))

rasterStewart

Create a Raster from a Stewart SpatialPointsDataFrame

Description
This function creates a raster from a regularly spaced Stewart SpatialPointsDataFrame (output of the `stewart` function).

Usage
rasterStewart(x, mask = NULL)

Arguments
- **x**  
  sp object (SpatialPointsDataFrame); output of the `stewart` function.
- **mask**  
  sp object (SpatialPolygonsDataFrame); this object is used to clip the raster. (optional)

Value
Raster of potential values.

See Also
`stewart`, `rasterStewart`, `plotStewart`, `contourStewart`, `CreateGrid`, `CreateDistMatrix`.

Examples
```R
data(spatData)
# Compute Stewart potentials from known points (spatPts) on a
# grid defined by its resolution
mystewart <- stewart(knownpts = spatPts, varname = "Capacite",
  typefct = "exponential", span = 1000, beta = 3,
  resolution = 50, longlat = FALSE, mask = spatMask)
# Create a raster of potentials values
mystewartraster <- rasterStewart(x = mystewart, mask = spatMask)
plot(mystewartraster)
```
Description

This function computes the catchment areas as defined by W.J. Reilly (1931).

Usage

reilly(knownpts, unknownpts = NULL, matdist = NULL, varname,
  typefct = "exponential", span, beta, resolution = 2000, longlat = FALSE,
  mask = NULL)

Arguments

knownpts  sp object (SpatialPointsDataFrame or SpatialPolygonsDataFrame); this is the set of known observations to estimate the catchment areas from.
unknownpts sp object (SpatialPointsDataFrame or SpatialPolygonsDataFrame); this is the set of unknown units for which the function computes the estimates. Not used when resolution is set up. (optional)
matdist    matrix; a distance matrix. Row names match the first column of the knownpts object dataframe. Column names match the first column of the unknownpts object dataframe. (optional)
varname    character; name of the variable in the knownpts dataframe from which values are computed. Quantitative variable with no negative values.
typefct    character; spatial interaction function. Options are "pareto" (default, means power law) or "exponential". If "pareto" the interaction is defined as: (1 + alpha * mDistance) ^ (-beta). If "exponential" the interaction is defined as: exp(- alpha * mDistance ^ beta). The alpha parameter is computed from parameters given by the user (beta and span).
span       numeric; distance where the density of probability of the spatial interaction function equals 0.5.
beta       numeric; impedance factor for the spatial interaction function.
resolution numeric; resolution of the output SpatialPointsDataFrame (in map units).
longlat    logical; euclidean distance (FALSE, default) or Great Circle distance (TRUE). If TRUE inputs are expected in the WGS84 reference system.
mask       sp object; the spatial extent of this object is used to create the regularly spaced SpatialPointsDataFrame output. (optional)

Details

If unknownpts is NULL then resolution must be used.
Value

SpatialPointsDataFrame with the computed catchment areas in a new field named OUTPUT. Values match the row names of knownpts.

References

REILLY, W. J. (1931) The law of retail gravitation, W. J. Reilly, New York.

See Also

reilly, rasterReilly, plotReilly, CreateGrid, CreateDistMatrix.

Examples

# Create a SpatialPointsDataFrame grid of spatMask extent and 200 meters resolution
data(spatData)
mygrid <- CreateGrid(w = spatMask, resolution = 200)
# Create a distance matrix between known points (spatPts) and mygrid
mymat <- CreateDistMatrix(knownpts = spatPts, unknownpts = mygrid,
                           longlat = FALSE)
# Compute Reilly catchment areas from known points (spatPts) on a given grid (mygrid) using a given distance matrix (mymat)
myreilly2 <- reilly(knownpts = spatPts, unknownpts = mygrid,
                     matdist = mymat, varname = "Capacite",
                     typefct = "exponential", span = 1250,
                     beta = 3, longlat = FALSE, mask = spatMask)
row.names(spatPts) <- spatPts$CodHop
# Compute Reilly catchment areas from known points (spatPts) on a grid defined by its resolution
myreilly <- reilly(knownpts = spatPts, varname = "Capacite",
                   typefct = "exponential", span = 1250, beta = 3,
                   resolution = 200, longlat = FALSE, mask = spatMask)
# The function output a SpatialPointsDataFrame
class(myreilly)
# The OUTPUT field values match knownpts row names
head(unique(myreilly$OUTPUT))
spatMask  
*Paris Perimeter*

**Description**
A SpatialPolygonsDataFrame of the Paris perimeter.

spatPts  
*Public Hospitals*

**Description**
A SpatialPointsDataFrame of 18 public hospitals with their capacity (Capacite field = number of beds).

spatUnits  
*Spatial Units of Paris*

**Description**
A SpatialPolygonsDataFrame of the 20 spatial arrondissements of the Paris.

stewart  
*Stewart Potentials*

**Description**
This function computes the potentials as defined by J.Q. Stewart (1942).

**Usage**
```r
stewart(knownpts, unknownpts = NULL, matdist = NULL, varname, 
typefct = "exponential", span, beta, resolution = 2000, longlat = FALSE, 
mask = NULL)
```
Arguments

knownpts     sp object (SpatialPointsDataFrame or SpatialPolygonsDataFrame); this is the set of known observations to estimate the potentials from.

unknownpts   sp object (SpatialPointsDataFrame or SpatialPolygonsDataFrame); this is the set of unknown units for which the function computes the estimates. Not used when resolution is set up. (optional)

matdist      matrix; a distance matrix. Row names match the first column of the knownpts object dataframe. Column names match the first column of the unknownpts object dataframe. (optional)

varname      character; name of the variable in the knownpts dataframe from which potentials are computed. Quantitative variable with no negative values.

typefct      character; spatial interaction function. Options are "pareto" (default, means power law) or "exponential". If "pareto" the interaction is defined as: $(1 + \alpha * mDistance)^(-\beta)$. If "exponential" the interaction is defined as: $\exp(-\alpha * mDistance ^ \beta)$. The alpha parameter is computed from parameters given by the user (beta and span).

span         numeric; distance where the density of probability of the spatial interaction function equals 0.5.

beta         numeric; impedance factor for the spatial interaction function.

resolution   numeric; resolution of the output SpatialPointsDataFrame (in map units).

longlat      logical; euclidean distance (FALSE, default) or Great Circle distance (TRUE). If TRUE inputs are expected in the WGS84 reference system.

mask         sp object; the spatial extent of this object is used to create the regularly spaced SpatialPointsDataFrame output. (optional)

Details

If unknownpts is NULL then resolution must be used.

Value

SpatialPointsDataFrame with the computed potentials in a new field named OUTPUT

References


See Also

stewart, rasterStewart, plotStewart, contourStewart, CreateGrid, CreateDistMatrix.
Examples

# Create a SpatialPointsDataFrame grid of spatMask extent and 200 meters resolution
data(spatData)
mygrid <- CreateGrid(w = spatMask, resolution = 200)

# Create a distance matrix between known points (spatPts) and mygrid
mymat <- CreateDistMatrix(knownpts = spatPts, unknownpts = mygrid,
                           longlat = FALSE)

# Compute Stewart potentials from known points (spatPts) on a given grid (mygrid) using a given distance matrix (mymat)
mystewart <- stewart(knownpts = spatPts, unknownpts = mygrid,
                     matdist = mymat, varname = "Capacite",
                     typeofct = "exponential", span = 1250,
                     beta = 3, longlat = FALSE, mask = spatMask)

# Compute Stewart potentials from known points (spatPts) on a grid defined by its resolution
mystewart2 <- stewart(knownpts = spatPts, varname = "Capacite",
                      typeofct = "exponential", span = 1250, beta = 3,
                      resolution = 200, longlat = FALSE, mask = spatMask)

# The two methods have the same result
identical(mystewart, mystewart2)

# The function output a SpatialPointsDataFrame class
class(mystewart)

# Computed values
summary(mystewart$OUTPUT)
Index

colorRampPalette, 7, 8
contourStewart, 2, 2, 9, 11, 13, 15
CreateDistMatrix, 2, 3, 4, 6–11, 13, 15
CreateGrid, 2, 4, 4, 6–11, 13, 15

huff, 5, 6, 7, 9, 13
plotHuff, 6, 6, 7, 9, 13
plotReilly, 7, 8, 10, 13
plotStewart, 2, 8, 9, 11, 13, 15

rasterHuff, 6, 7, 9, 13
rasterReilly, 7, 8, 10, 10, 13
rasterStewart, 2, 8, 9, 11, 11, 13, 15
ratify, 10
reilly, 8, 10, 12, 13

SpatialPosition, 13
SpatialPosition-package
(SpatialPosition), 13
spatMask, 14
spatPts, 14
spatUnits, 14
spDists, 3
stewart, 2, 9, 11, 13, 14, 15