Package 'RapidPolygonLookup'

May 7, 2026

Type Package

Title POLYGON LOOKUP USING KD TREES

Version 0.1.1

Date 2019-01-13

Depends R(>= 2.10.0), sp, RANN, PBSmapping, RgoogleMaps

Author Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

Maintainer Markus Loecher <markus.loecher@gmail.com>

Description Facilitates efficient polygon search using kd trees.
Coordinate level spatial data can be aggregated to higher geographical identities like census blocks, ZIP codes or police district boundaries. This process requires mapping each point in the given data set to a particular identity of the desired geographical hierarchy. Unless efficient data structures are used, this can be a daunting task. The operation point.in.polygon() from the package sp is computationally expensive. Here, we exploit kd-trees as efficient nearest neighbor search algorithm to dramatically reduce the effective number of polygons being searched.

License GPL

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2019-01-14 10:30:11 UTC

Contents

RapidPolygonLookup-package 2
AddRanges 3
california.tract10 3
CropSpatialPolygonsDataFrame 4
DiagnoseFailure 5
FindPolygonInRanges 6
RapidPolygonLookup 7

SearchForPolygon 8
sf.crime.2012 9
sf.polys 10

Index **11**

`RapidPolygonLookup-package`
Polygon lookup using kd trees

Description

This package facilitates efficient polygon search using kd trees. Coordinate level spatial data can be aggregated to higher geographical identities like census blocks, ZIP codes or police district boundaries. This process requires mapping each point in the given data set to a particular identity of the desired geographical hierarchy. Unless efficient data structures are used, this can be a daunting task. The operation point.in.polygon() from the package sp is computationally expensive. Here, we exploit kd-trees as efficient nearest neighbor search algorithm to dramatically reduce the effective number of polygons being searched.

Details

Package:	RapidPolygonLookup
Type:	Package
Title:	Polygon lookup using kd trees
Version:	0.1
Date:	2013-11-18
Depends:	R(>= 2.10.0), sp, RANN, PBSmapping, RgoogleMaps
Author:	"Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>, Madhav Kum
Maintainer:	"Markus Loecher" <markus.loecher@gmail.com>
License:	GPL
LazyLoad:	yes

Author(s)

Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

AddRanges *Add xlim and ylim for each polygon*

Description

This function computes the bounding box for each polygon and adds this information to the list. The bounding boxes can be used in various applications. Our main motivation is for the massive PointsInPolygon search to exclude those polygons as candidates whose bounding box does not contain the current point.

Usage

```
AddRanges(poly.list)
```

Arguments

poly.list	polygon list with three elements: data, polys, and poly.centers

Value

Returns augmented polygon list with additional element – "ranges"

Author(s)

Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

Examples

```
data(sf.polys, envir = environment())
sf.polys <- AddRanges(sf.polys)
str(sf.polys$ranges)
```

california.tract10 *Census Tract spatial polygons for the state of California*

Description

Object of class SpatialPolygonsDataFrame containing spatial polygons of Census tracts in California. The object has been originally created from the 2010 US Census tiger/line boundary files (http://www.census.gov/geo/www/tiger/) for Census Tracts. The polygons have been manually cropped to the area in and around San Francisco.

Usage

```
data(california.tract10)
```

Format

An object of class SpatialPolygonsDataFrame from the sp package

`data` data frame containing information for 457 variables (excluding ids) available from the summary file 1

`polygons` polygons of Census Tracts

`plotOrder` plotting order of polygons

`bbox` bounding box of spatial polygons

`proj4string` projection of polygons. All polygons are projected in CRS(" +proj=longlat +ellps=GRS80 +datum=NAD83 +no_defs +towgs84=0,0,0")

Details

For details on the summary variables present in the data frame please refer

http://www.census.gov/prod/cen2000/doc/sf1.pdf

Source

http://cran.r-project.org/web/packages/UScensus2010/index.html

References

Zack W. Almquist (2010). US Census Spatial and Demographic Data in R: The UScensus2000 Suite of Packages. Journal of Statistical Software, 37(6), 1-31. URL http://www.jstatsoft.org/v37/i06/ http://www.census.gov/prod/cen2000/doc/sf1.pdf

Examples

```
data(california.tract10, envir = environment())
plot(california.tract10)
```

CropSpatialPolygonsDataFrame

Crop polygons to bounding box and adds polygon centers

Description

This function serves three purposes: (i) changes the (complicated) data structure of a spatial polygon (from the sp package) to a format which is aligned with the (simpler) PBSmapping polygon format. (ii) clips/crops the polygons to a pre specified bounding box (iii) computes and adds the polygon centers for each polygon

Usage

```
CropSpatialPolygonsDataFrame(x, bb = NULL, verbose = 0)
```

Arguments

x	object of class SpatialPolygonsDataFrame
bb	bounding box to crop the polygons
verbose	level of verbosity

Value

New list with separate entries for data, polys, and poly centers

Author(s)

Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

Examples

```
# San Francisco:
data(california.tract10, envir = environment())
sf.polys <- CropSpatialPolygonsDataFrame(x= california.tract10,
                                         bb= data.frame(X=c(-122.5132, -122.37),
                                                        Y= c(37.70760, 37.81849)))
```

DiagnoseFailure *Visualize points that could not be mapped using RapidPolygonLookup()*

Description

This functions plots the points that could not be mapped using RapidPolygonLookup() The points are overlayed on the polygons to contextualize their geographical location and understand the reason behind their exclusion.

Usage

```
DiagnoseFailure(XY.polys, poly.list = NULL)
```

Arguments

XY.polys	output from function RapidPolygonLookup()
poly.list	polygon list with 3 or 4 elements: data, polys, poly.centers, and possibly ranges. Needs to be supplied if RapidPolygonLookup() was run with keep.data= FALSE

Author(s)

Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

Examples

```
data(sf.crime.2012, envir = environment())
data(sf.polys, envir = environment())
cat(nrow(sf.crime.2012), "rows in SF crime \n")

XY.kdtree <- RapidPolygonLookup(sf.crime.2012[,c("X","Y")], poly.list= sf.polys,
                                k= 10, N= 1000,
                                poly.id= "fips", poly.id.colname= "census.block",
                                keep.data= TRUE, verbose= TRUE)
DiagnoseFailure(XY.kdtree)
```

`FindPolygonInRanges` *Use range-search to map points to polygon.*

Description

This function searches the lat-long ranges of polygons to come up with a shorter list of candidates on which point.in.polygon() from the sp package can be applied.

Usage

```
FindPolygonInRanges(poly.list, XY, poly.id = "fips", poly.id.colname = "census.block",
    verbose = 0)
```

Arguments

`poly.list`	polygon list with 3 or 4 elements: data, polys, poly.centers, and possibly ranges
`XY`	data frame containing X-Y columns
`poly.id`	column name in 'poly.list$data' containing the polygon identifier
`poly.id.colname`	desired column name in the output data frame containing the polygon identifier
`verbose`	level of verbosity

Author(s)

Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

Examples

```
data(sf.crime.2012, envir = environment())
data(sf.polys, envir = environment())

sf.polys <- AddRanges(sf.polys)
XY <- FindPolygonInRanges(sf.polys, sf.crime.2012[1:1000,], verbose=0)

which(is.na(XY["census.block"]))
table(XY$rank)
```

RapidPolygonLookup *Efficient spatial polygon search using kd-trees.*

Description

Given spatial partitions such as census blocks, ZIP codes or police district boundaries, we are frequently faced with the need to spatially aggregate data. Unless efficient data structures are used, this can be a daunting task. The operation point.in.polygon() from the package sp is computationally expensive. Here, we exploit kd-trees as efficient nearest neighbor search algorithm to dramatically reduce the effective number of polygons being searched. Points that are left unmapped are put through a linear search to find the associated polygon.

Usage

```
RapidPolygonLookup(XY, polygons, poly.list = NULL, k = 10, N = nrow(XY),
    poly.id = "fips", poly.id.colname = "census.block", keep.data = TRUE,
    verbose = 0)
```

Arguments

XY	data frame containing X-Y or (lon-lat, long-lat, longitude-latitude) columns
polygons	polygons to crop and add poly centres
poly.list	polygon list with three elements: data, polys, and poly.centers as output from function CropSpatialPolygonsDataFrame()
k	maximum number of near neighbours to compute. The default value is set to 10
N	number of rows of XY to search
poly.id	column name in 'poly.list$data' containing the polygon identifier
poly.id.colname	desired column name in the output data frame containing the polygon identifier
keep.data	retain polygon list and centers for future referece
verbose	level of verbosity

Value

The original points augmented with polygon ID are returned along with the poly centers and other call information

Author(s)

Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

Examples

```
data(sf.crime.2012, envir = environment())
data(sf.polys, envir = environment())
cat(nrow(sf.crime.2012), "rows in SF crime \n")

XY.kdtree <- RapidPolygonLookup(sf.crime.2012[,c("X","Y")], poly.list= sf.polys,
                                k= 10, N= 1000,
                                poly.id= "fips", poly.id.colname= "census.block",
                                keep.data= TRUE, verbose= TRUE)

XY.kdtree.DF <- XY.kdtree$XY
table(XY.kdtree.DF$rank, useNA= "always")
hist(XY.kdtree.DF$rank, xlab = "rank of neighbor")
```

`SearchForPolygon` *Use kd-trees to search the nearest neighbour polygons for a given set of points*

Description

This function uses the nn2() function from the RANN package to come up with a shorter list of candidates on which point.in.polygon() from the sp package can be applied.

Usage

```
SearchForPolygon(poly.list, XY, k, poly.id, poly.id.colname,
    verbose = 0)
```

Arguments

`poly.list`	polygon list with 3-4 elements: poly.centers, data, polys and possibly ranges
`XY`	data frame containing X-Y columns to assign polygons to
`k`	maximum number of nearest neighbours to compute. The default value is set to 10.
`poly.id`	column name in 'poly.list$data' containing the polygon identifier
`poly.id.colname`	desired column name in the output data frame containing the polygon identifier
`verbose`	level of verbosity

Value

Returns data frame with identified polygon and nearest neighbour rank

Author(s)

Markus Loecher <markus.loecher@gmail.com> and Madhav Kumar <madhavkumar2005@gmail.com>

Examples

```
data(sf.crime.2012, envir = environment())
data(sf.polys, envir = environment())
XY.polys <- SearchForPolygon(poly.list= sf.polys, XY= sf.crime.2012[1:1000,], k= 10,
                             poly.id= "fips", poly.id.colname= "census.block",
                             verbose= TRUE)
```

---

sf.crime.2012 *Sample data with lat/long information*

Description

2012 crime incident data from the city of San Francisco

Usage

```
data(sf.crime.2012)
```

Format

A data frame with 20,000 randomly selected observations with the following variables and their types:

`Date` character

`X` numeric

`Y` numeric

`violent` Factor

Details

There are no more details required

Source

https://data.sfgov.org/Public-Safety/SFPD-Reported-Incidents-2003-to-Present/dyj4-n68b

Examples

```
data(sf.crime.2012, envir = environment())
```

sf.polys *Spatial polygons of San Francisco*

Description

Cropped spatial polygons from California Census tracts bounded between San Francisco limits

Usage

```
data(sf.polys)
```

Format

A list object with the following elements:

`data` data frame retained from California tracts object of class SpatialPolygonsDataFrame

`polys` PolySet object from PBSmapping containing the spatial polygons

`poly.centers` PolyData object from PBSmapping containing the polygon centroids

Details

This object is created from a function of CropSpatialPolygonsDataFrame() from the RapidPolygonLookup package

Source

http://cran.r-project.org/web/packages/UScensus2010/index.html

References

Zack W. Almquist (2010). US Census Spatial and Demographic Data in R: The UScensus2000 Suite of Packages. Journal of Statistical Software, 37(6), 1-31. URL http://www.jstatsoft.org/v37/i06/

Examples

```
data(sf.polys, envir = environment())
plotPolys(sf.polys$polys)
```

Index

∗ **datasets**
    california.tract10, 3
    sf.crime.2012, 9
    sf.polys, 10
∗ **package**
    RapidPolygonLookup-package, 2

AddRanges, 3

california.tract10, 3
CropSpatialPolygonsDataFrame, 4

DiagnoseFailure, 5

FindPolygonInRanges, 6

RapidPolygonLookup, 7
RapidPolygonLookup-package, 2

SearchForPolygon, 8
sf.crime.2012, 9
sf.polys, 10