Package 's2'

April 21, 2018

Type Package

Title Google's S2 Library for Geometry on the Sphere

Version 0.4-0

Description R bindings for Google's s2 library for geometric calculations on the sphere.

License Apache License (== 2.0)

Depends R (>= 3.0.0)

Imports methods, Rcpp (>= 0.12.5)

Suggests globe

URL https://github.com/spatstat/s2, https://code.google.com/archive/p/s2-geometry-library/

SystemRequirements OpenSSL >= 1.0.0, C++11

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 6.0.1.9000

Author Ege Rubak [aut, cre],
Jeroen Ooms [aut] (configure and cleanup script),
Edzer Pebesma [aut] (Interface to sf),
Google, Inc. [cph] (Original s2 source code)

Maintainer Ege Rubak <rubak@math.aau.dk>

Repository CRAN

Date/Publication 2018-04-21 21:17:58 UTC

R topics documented:

`s2-package` *s2: Google's S2 Library for Geometry on the Sphere*

Description

R bindings for Google's s2 library for geometric calculations on the sphere.

Details

This package aims at providing bindings for the C++ library s2 for geometry on the sphere. The C++ library was originally developed by Google under the Apache license. At the moment very few things are exposed and the main concern is to make the C++ code compile with the R toolchain on the three major platforms and have the package accepted on CRAN. Once this proof of concept has been achieved more of the underlying features will be made available to the useR.

Author(s)

Maintainer: Ege Rubak <rubak@math.aau.dk>

Authors:

- Jeroen Ooms <jeroen.ooms@stat.ucla.edu> (configure and cleanup script)
- Edzer Pebesma <edzer.pebesma@uni-muenster.de> (Interface to sf)

Other contributors:

- Google, Inc. (Original s2 source code) [copyright holder]

See Also

Useful links:

- https://github.com/spatstat/s2
- https://code.google.com/archive/p/s2-geometry-library/

`S2CapFromAxisHeight` *Construct a S2Cap using axis and height*

Description

Constructs a S2Cap from axis and height

Usage

```
S2CapFromAxisHeight(axis, height)
```

Arguments

`axis`	A numeric vector with three entries represententing the direction axis.
`height`	Single numeric representing the height of the cap.

`S2Cap_area` *Area of s2cap*

Description

Area of a cap

Usage

```
S2Cap_area(cap)
```

Arguments

`cap`	Named list containing axis and height of cap.

S2Cap_Contains *Test of Containment in S2Cap*

Description

Test whether the given object(s) on the sphere are contained in a cap. At the moment only point containment can be tested.

Usage

```
S2Cap_Contains(cap, x)
```

Arguments

cap	Named list containing axis and height of cap.
x	Object(s) to test for containment in 'cap'. At the moment only points are handled and they must be specified as a three-column matrix.

Examples

```
cap <- list(axis = c(0,1,0), height = 0.1)
S2Cap_Contains(cap, matrix(c(0,1,0),ncol=3))
S2Cap_Contains(cap, matrix(c(1,0,0),ncol=3))
```

S2Cap_GetRectBound *Bounding latitude and longitude rectangle for spherical cap*

Description

Bounding latitude and longitude rectangle for spherical cap

Usage

```
S2Cap_GetRectBound(x)
```

Arguments

x	cap

S2Cell *Make a list of S2Cells*

Description

Make a list of S2Cells

Usage

```
S2Cell(x)
```

Arguments

x	Input to create cells from. Currently only an object of class S2CellId is supported.

S2CellIdFromPoint *Make a Vector of S2CellIds From Points on the Sphere*

Description

Create a vector of S2CellIds corresponding to the cells at the given level containing the given points. The default level (30) corresponds to leaf cells (finest level).

Usage

```
S2CellIdFromPoint(x, level = 30L)
```

Arguments

x	Three-column matrix reprensenting the points.
level	Integer between 0 and 30 (incl).

Value

An object of class ‘S2CellId‘.

S2CellId_ToPoint *Convert S2CellId to a S2Point*

Description

Convert S2CellId to a S2Point

Usage

```
S2CellId_ToPoint(x)
```

Arguments

x	Object of class S2CellId.

Value

Three-column matrix reprensenting the points..

S2CellId_ToString *Make a vector of S2CellId strings*

Description

Make a vector of S2CellId strings

Usage

```
S2CellId_ToString(x)
```

Arguments

x	A charecter vector with S2CellIds (in token form).

Value

A character vector with S2CellId strings.

S2Covering *Approximate a Region on the Sphere by a Covering of S2Cells*

Description

Approximate a region on the sphere by a (possibly interior) covering of S2Cells.

Usage

```
S2Covering(x, max_cells, min_level, max_level, interior = FALSE)
```

Arguments

x	Region to cover. Currently it must be a polygon or cap.
max_cells	Positive integer. Maximal number of cells to use in the covering.
min_level	Integer between 0 and 30 specifying the lowest cell level to use. Must be less than or equal to ‘max_level‘.
max_level	Integer between 0 and 30 specifying the highest cell level to use. Must be greater than or equal to ‘min_level‘.
interior	Logical to get an interior covering.

Value

A list containing an entry ‘ids‘ with the ids of the S2Cells used to cover the region and possibly other entries for internal usage.

S2LatLngRect *Create a rectangle of latitude and longitude on the sphere*

Description

Create a rectangle of latitude and longitude on the sphere

Usage

```
S2LatLngRect(lo, hi)
```

Arguments

lo	Latitude and longitude (in that order) in degrees of lower left corner.
hi	Latitude and longitude (in that order) in degrees of upper right corner.

S2LatLngRect_area *Area of s2latlngrect*

Description

Area of a S2LatLngRect.

Usage

```
S2LatLngRect_area(x)
```

Arguments

x	Named list containing axis and height of cap.

S2Point_interpolate *Interpolation of points on unit sphere*

Description

Given a sequence of points on the unit sphere add interpolating points so consecutive points are within distance ‘eps‘ of each other.

Usage

```
S2Point_interpolate(x, eps)
```

Arguments

x	Matrix with three columns representing the points.
eps	Strictly positive real number. Values greater than or equal to pi correspond to no interpolation.

S2Polygon *Construct a S2Polygon*

Description

This function builds an 'S2Polygon' using the C++ polygon builder. See the header file inst/include/s2/s2polygonbuilder.h for details on how to use the arguments.

Usage

```
S2Polygon(x, validate = TRUE, xor_edges = TRUE, vertex_merge_radius = 0,
  edge_splice_fraction = 0.866, undirected_edges = FALSE)
```

Arguments

x	Input to construct the polygon from. At the moment the only valid input is a list of loops.
validate	Logical to validate the S2Polygon. Default is 'TRUE'.
xor_edges	Logical to indicate that edges should be 'xor'ed to avoind multiple loops with common edges. Default is 'TRUE'.
vertex_merge_radius	Numeric indicating that vertices within this distance should be merged. Defaults to zero (i.e. no merging).
edge_splice_fraction	Determines when edges are spliced. See C++ header as indicated in the description of this function. Default is 0.866.
undirected_edges	Logical to indicate that input edges should be considered undirected. Default is 'FALSE'. multiple loops with common edges. Default is 'TRUE'.

S2Polygons_area *compute areas for a list of s2polygons*

Description

this function is equivalent to st_area, in that it returns a numeric vector with polygon areas

Usage

```
S2Polygons_area(x)
```

Arguments

x	List of list of loops represented by three-column matrices.

Details

the area is on the unit sphere, in [0, 4 * pi]

`S2Polygons_centroid` *compute centroids for a list of s2polygons*

Description

this function is equivalent to st_centroid, in that it returns a numeric vector with polygon centroids

Usage

```
S2Polygons_centroid(x)
```

Arguments

`x`	List of list of loops represented by three-column matrices.

`S2Polygons_intersect` *for two sets of s2polygons, which ones intersect?*

Description

this function is equivalent to st_intersects, in that it returns a sparse matrix with indexes for pairs of intersecting polygons

Usage

```
S2Polygons_intersect(x, y)
```

Arguments

`x`	List of list of loops represented by three-column matrices.
`y`	List of list of loops represented by three-column matrices.

S2Polygons_intersection *intersection of sets of s2polygons*

Description

this function generalizes S2Polygon_intersection, by allowing two lists of polygons, each list element equal to a polygon of that function.

Usage

```
S2Polygons_intersection(x, y)
```

Arguments

x	List of list of loops represented by three-column matrices.
y	List of list of loops represented by three-column matrices.

S2Polygon_Contains *Test of Containment in S2Polygon*

Description

Test whether the given object(s) on the sphere are contained in a polygon. At the moment only point containment can be tested.

Usage

```
S2Polygon_Contains(poly, x, approx = TRUE)
```

Arguments

poly	Named list containing an entry called ‘loops‘ containing a list of polygon loops represented by three-column matrices.
x	Object(s) to test for containment in ‘poly‘. At the moment only points are handled and they must be specified as a three-column matrix.
approx	Logical to use approximate testing of point in polygon (allows points very slightly outside the polygon). Useful for allowing points directly on the border.

S2Polygon_union *Union and intersection of two s2polygons*

Description

Given two lists of three column matrices representing S2Polygons calculate the union or intersection. Assumes the polygons have already been validated and put into the correct format by [‘S2Polygon‘]. Note the input is only the loops/rings of two polygons not the entire objects with areas and hole indicators.

Usage

```
S2Polygon_union(x, y)
```

Arguments

x	List of loops represented by three-column matrices.
y	List of loops represented by three-column matrices.

S2Polyline_dist *Distance from points to line on sphere*

Description

Distance from points to line on sphere.

Usage

```
S2Polyline_dist(line, x)
```

Arguments

line	Line represented by a sequence of vertices given as a single three-column matrices with one line per vertex.
x	Points represented by three-column matrices.

Index