

Package ‘lwgeom’

January 21, 2018

Version 0.1-3

Title Bindings to Selected 'liblwgeom' Functions for Simple Features

Description

Access to selected functions found in 'liblwgeom' <<https://github.com/postgis/postgis/tree/svn-trunk/liblwgeom>>, the light-weight geometry library used by 'PostGIS' <<http://postgis.net/>>.

Depends R (>= 3.3.0)

Imports Rcpp, units, sf (>= 0.6-0)

Suggests covr, sp, geosphere, testthat

LinkingTo Rcpp, sf (>= 0.6-0)

SystemRequirements GEOS (>= 3.3.0), PROJ.4 (>= 4.8.0)

License GPL-2

Copyright file COPYRIGHTS

URL <https://github.com/r-spatial/lwgeom/>

BugReports <https://github.com/r-spatial/lwgeom/issues/>

Collate init.R RcppExports.R geohash.R split.R subdivide.R valid.R
transform.R bounding_circle.R geodetic.R bearing.R
snap_to_grid.R twkb.R

RoxygenNote 6.0.1

NeedsCompilation yes

Author Edzer Pebesma [aut, cre] (0000-0001-8049-7069),
Colin Rundel [ctb],
liblwgeom developers [cph]

Maintainer Edzer Pebesma <edzer.pebesma@uni-muenster.de>

Repository CRAN

Date/Publication 2018-01-21 16:46:47 UTC

R topics documented:

bounding_circle	2
lwgeom_extSoftVersion	3
lw_geodetic	3
st_as_sfc.TWKB	5
st_geod_azimuth	5
st_geohash	6
st_snap_to_grid	7
st_split	7
st_subdivide	8
st_transform_proj	9
valid	10
Index	11

bounding_circle	<i>Generate the minimum bounding circle</i>
-----------------	---

Description

Generate the minimum bounding circle

Usage

```
st_minimum_bounding_circle(x, nQuadSegs = 30)
```

Arguments

x	object of class sfg, sfg or sf
nQuadSegs	number of segments per quadrant (passed to st_buffer)

Details

st_minimum_bounding_circle uses the lwgeom_calculate_mbc method also used by the PostGIS command ST_MinimumBoundingCircle.

Value

Object of the same class as x

Examples

```
library(sf)

x = st_multipoint(matrix(c(0,1,0,1),2,2))
y = st_multipoint(matrix(c(0,0,1,0,1,1),3,2))

mbcx = st_minimum_bounding_circle(x)
mbcy = st_minimum_bounding_circle(y)

if (.Platform$OS.type != "windows") {
  plot(mbcx, axes=TRUE); plot(x, add=TRUE)
  plot(mbcy, axes=TRUE); plot(y, add=TRUE)
}

nc = st_read(system.file("gpkg/nc.gpkg", package="sf"))
state = st_union(st_geometry(nc))

if (.Platform$OS.type != "windows") {
  plot(st_minimum_bounding_circle(state), asp=1)
  plot(state, add=TRUE)
}
```

lwgeom_extSoftVersion *Provide the external dependencies versions of the libraries linked to sf*

Description

Provide the external dependencies versions of the libraries linked to sf

Usage

```
lwgeom_extSoftVersion()
```

lw_geodetic *liblwgeom geodetic functions*

Description

liblwgeom geodetic functions for length, area, segmentizing, covers

Usage

```

st_geod_area(x)

st_geod_length(x)

st_geod_segmentize(x, max_seg_length)

st_geod_covers(x, y, sparse = TRUE)

st_geod_covered_by(x, y, sparse = TRUE)

st_geod_distance(x, y, tolerance = 0, sparse = FALSE)

```

Arguments

x	object of class sf, sfc or sfg
max_seg_length	segment length in degree, radians, or as a length unit (e.g., m)
y	object of class sf, sfc or sfg
sparse	logical; if TRUE, return a sparse matrix (object of class sgbp), otherwise, return a dense logical matrix.
tolerance	double or length units value: if positive, the first distance less than tolerance is returned, rather than the true distance

Details

st_area will give an error message when the area spans the equator and lwgeom is linked to a proj.4 version older than 4.9.0 (see [lwgeom_extSoftVersion](#))

Note

this function should is used by [st_distance](#), do not use it directly

Examples

```

library(sf)
nc = st_read(system.file("gpkg/nc.gpkg", package="sf"))
st_geod_area(nc[1:3,])
# st_area(nc[1:3,])
l = st_sfc(st_linestring(rbind(c(7,52), c(8,53))), crs = 4326)
st_geod_length(l)
library(units)
pol = st_polygon(list(rbind(c(0,0), c(0,60), c(60,60), c(0,0))))
x = st_sfc(pol, crs = 4326)
seg = st_geod_segmentize(x[1], set_units(10, km))
plot(seg, graticule = TRUE, axes = TRUE)
pole = st_polygon(list(rbind(c(0,80), c(120,80), c(240,80), c(0,80))))
pt = st_point(c(0,90))
x = st_sfc(pole, pt, crs = 4326)
st_geod_covers(x[c(1,1,1)], x[c(2,2,2,2)])

```

```
pole = st_polygon(list(rbind(c(0,80), c(120,80), c(240,80), c(0,80))))
pt = st_point(c(30,70))
x = st_sfc(pole, pt, crs = 4326)
st_geod_distance(x, x)
```

st_as_sfc.TWKB	<i>create sfc object from tiny well-known binary (twkb)</i>
----------------	---

Description

create sfc object from tiny well-known binary (twkb)

Usage

```
## S3 method for class 'TWKB'
st_as_sfc(x, ...)
```

Arguments

x	list with raw vectors, of class TWKB
...	ignored

See Also

<https://github.com/TWKB/Specification/blob/master/twkb.md>

Examples

```
l = structure(list(as.raw(c(0x02, 0x00, 0x02, 0x02, 0x02, 0x08, 0x08))), class = "TWKB")
library(sf) # load generic
st_as_sfc(l)
```

st_geod_azimuth	<i>compute azimuth between sequence of points</i>
-----------------	---

Description

compute azimuth between sequence of points

Usage

```
st_geod_azimuth(x)
```

Arguments

x	object of class sf, sfc or sfg
---	--------------------------------

Examples

```
library(sf)
p = st_sfc(st_point(c(7,52)), st_point(c(8,53)), crs = 4326)
st_geod_azimuth(p)
```

st_geohash	<i>compute geohash from (average) coordinates (requires lwgeom)</i>
------------	---

Description

compute geohash from (average) coordinates (requires lwgeom)

Usage

```
st_geohash(x, precision = 0)
```

Arguments

x	object of class sf, sfc or sfg
precision	integer; precision (length) of geohash returned. From the liblwgeom source: “where the precision is non-positive, a precision based on the bounds of the feature. Big features have loose precision. Small features have tight precision.”

Details

see <http://geohash.org/> or <https://en.wikipedia.org/wiki/Geohash>.

Value

character vector with geohashes

Examples

```
library(sf)
lwgeom::st_geohash(st_sfc(st_point(c(1.5,3.5)), st_point(c(0,90))), 2)
lwgeom::st_geohash(st_sfc(st_point(c(1.5,3.5)), st_point(c(0,90))), 10)
```

st_snap_to_grid	<i>Snap geometries to a grid</i>
-----------------	----------------------------------

Description

Snap geometries to a grid

Usage

```
st_snap_to_grid(x, size, origin)
```

Arguments

x	object with geometries to be snapped
size	numeric or (length) units object; grid cell size in x-, y- (and possibly z- and m-) directions
origin	numeric; origin of the grid

Value

object of the same class as x

Examples

```
# obtain data
library(sf)
x = st_read(system.file("gpkg/nc.gpkg", package="sf"), quiet = TRUE)[1, ] %>%
  st_geometry %>%
  st_transform(3395)

# snap to a grid of 5000 m
y = st_snap_to_grid(x, 5000)

# plot data for visual comparison
par(mfrow = c(1, 2))
plot(x, main = "original data")
plot(y, main = "snapped to 5000 m")
```

st_split	<i>Return a collection of geometries resulting by splitting a geometry</i>
----------	--

Description

Return a collection of geometries resulting by splitting a geometry

Usage

```
st_split(x, y)
```

Arguments

x object with geometries to be splitted

y object split with (blade); if y contains more than one feature geometry, the geometries are [st_combined](#)

Value

object of the same class as x

Examples

```
library(sf)
l = st_as_sfc('MULTILINESTRING((10 10, 190 190), (15 15, 30 30, 100 90))')
pt = st_sfc(st_point(c(30,30)))
lwgeom::st_split(l, pt)
```

<code>st_subdivide</code>	<i>Return a collection of geometries resulting by subdividing a geometry</i>
---------------------------	--

Description

Return a collection of geometries resulting by subdividing a geometry

Usage

```
st_subdivide(x, max_vertices)
```

Arguments

x object with geometries to be subdivided

max_vertices integer; maximum size of the subgeometries (at least 8)

Value

object of the same class as x

Examples

```
library(sf)
demo(nc, ask = FALSE, echo = FALSE)
x = st_subdivide(nc, 10)
plot(x[1])
```

st_transform_proj	<i>Transform or convert coordinates of simple features directly with Proj.4</i>
-------------------	---

Description

Transform or convert coordinates of simple features directly with Proj.4

Usage

```
st_transform_proj(x, crs, ...)

## S3 method for class 'sfc'
st_transform_proj(x, crs, ...)

## S3 method for class 'sf'
st_transform_proj(x, crs, ...)

## S3 method for class 'sfg'
st_transform_proj(x, crs, ...)
```

Arguments

x	object of class sf, sfc or sfg
crs	object or class crs, or input to st_crs (proj4string, or EPSG code)
...	ignored

Details

Transforms coordinates of object to new projection, using Proj.4 and not the GDAL API.

The `st_transform_proj` method for `sfg` objects assumes that the CRS of the object is available as an attribute of that name.

Examples

```
library(sf)
p1 = st_point(c(7,52))
p2 = st_point(c(-30,20))
sfc = st_sfc(p1, p2, crs = 4326)
sfc
st_transform_proj(sfc, "+proj=wintri")
library(sf)
nc = st_read(system.file("shape/nc.shp", package="sf"))
st_transform_proj(nc[1,], "+proj=wintri +over")
st_transform_proj(structure(p1, proj4string = "+init=epsg:4326"), "+init=epsg:3857")
```

valid	<i>Make an invalid geometry valid</i>
-------	---------------------------------------

Description

Make an invalid geometry valid

Usage

```
st_make_valid(x)
```

Arguments

x object of class sfg, sfg or sf

Details

st_make_valid uses the lwgeom_makevalid method also used by the PostGIS command ST_makevalid.

Value

Object of the same class as x

Examples

```
library(sf)
x = st_sfc(st_polygon(list(rbind(c(0,0),c(0.5,0),c(0.5,0.5),c(0.5,0),c(1,0),c(1,1),c(0,1),c(0,0)))))
suppressWarnings(st_is_valid(x))
y = lwgeom::st_make_valid(x)
st_is_valid(y)
y %>% st_cast()
```

Index

`bounding_circle`, 2

`lw_geodetic`, 3

`lwgeom_extSoftVersion`, 3, 4

`st_as_sfc.TWKB`, 5

`st_combine`, 8

`st_crs`, 9

`st_distance`, 4

`st_geod_area (lw_geodetic)`, 3

`st_geod_azimuth`, 5

`st_geod_covered_by (lw_geodetic)`, 3

`st_geod_covers (lw_geodetic)`, 3

`st_geod_distance (lw_geodetic)`, 3

`st_geod_length (lw_geodetic)`, 3

`st_geod_segmentize (lw_geodetic)`, 3

`st_geohash`, 6

`st_make_valid (valid)`, 10

`st_minimum_bounding_circle`
 (`bounding_circle`), 2

`st_snap_to_grid`, 7

`st_split`, 7

`st_subdivide`, 8

`st_transform_proj`, 9

`valid`, 10