Package: lwgeom (via r-universe)

October 26, 2024

Version 0.2-14

Title Bindings to Selected 'liblwgeom' Functions for Simple Features

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

Depends R (>= 3.3.0)

Imports Rcpp, units, sf (>= 1.0-15)

Suggests covr, sp, geosphere, testthat

LinkingTo Rcpp, sf (>= 0.6-0)

SystemRequirements GEOS (>= 3.5.0), PROJ (>= 4.8.0), sqlite3

License GPL-2

Copyright file COPYRIGHTS

Encoding UTF-8

URL https://r-spatial.github.io/lwgeom/, 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 bearing.R snap_to_grid.R startpoint.R twkb.R perimeter.R clockwise.R geod.R wkt.R wrap_x.R

RoxygenNote 7.3.1

NeedsCompilation yes

Author Edzer Pebesma [aut, cre] (<https://orcid.org/0000-0001-8049-7069>), Colin Rundel [ctb], Andy Teucher [ctb], liblwgeom developers [cph]

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

Repository CRAN

Date/Publication 2024-02-21 21:30:02 UTC

Contents

`bounding_circle` *Generate the minimum bounding circle*

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)
}
```

---

geod *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)

longitude coordinates returned are rescaled to [-180,180)

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)
```

---

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()
```

lwgeom_make_valid *Make an invalid geometry valid*

Description

Make an invalid geometry valid

Usage

```
lwgeom_make_valid(x)
```

Arguments

x	object of class sfc

perimeter *compute perimeter from polygons or other geometries*

Description

compute perimeter from polygons or other geometries

Usage

```
st_perimeter_lwgeom(x)

st_perimeter(x)

st_perimeter_2d(x)
```

Arguments

x	object of class sf, sfc or sfg

Value

numerical vector with perimeter for each feature (geometry), with unit of measure when possible

st_astext *Return Well-known Text representation of simple feature geometry*

Description

Return Well-known Text representation of simple feature geometry or coordinate reference system

Usage

```
st_astext(x, digits = options("digits"), ..., EWKT = FALSE)

st_asewkt(x, digits = options("digits"))
```

Arguments

x	object of class sfg, sfc, or sf
digits	integer; number of decimal digits to print
...	ignored
EWKT	logical; use PostGIS Enhanced WKT (includes srid)

Details

The returned WKT representation of simple feature geometry conforms to the simple features access specification and extensions (if EWKT = TRUE), known as EWKT, supported by PostGIS and other simple features implementations for addition of SRID to a WKT string.

st_asewkt() returns the Well-Known Text (WKT) representation of the geometry with SRID meta data.

Examples

```
library(sf)
pt <- st_sfc(st_point(c(1.0002,2.3030303)), crs = 4326)
st_astext(pt, 3)
st_asewkt(pt, 3)
```

st_as_sfc.TWKB *create sfc object from tiny well-known binary (twkb)*

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_force_polygon_cw *Force a POLYGON or MULTIPOLYGON to be clockwise*

Description

Check if a POLYGON or MULTIPOLYGON is clockwise, and if not make it so. According to the 'Right-hand-rule', outer rings should be clockwise, and inner holes should be counter-clockwise

Usage

```
st_force_polygon_cw(x)
```

Arguments

x	object with polygon geometries

Value

object of the same class as x

Examples

```
library(sf)
polys <- st_sf(cw = c(FALSE, TRUE),
               st_as_sfc(c('POLYGON ((0 0, 1 0, 1 1, 0 0))',
                           'POLYGON ((1 1, 2 2, 2 1, 1 1))')))

st_force_polygon_cw(polys)
st_force_polygon_cw(st_geometry(polys))
st_force_polygon_cw(st_geometry(polys)[[1]])
```

st_geod_azimuth *compute azimuth between sequence of points*

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 *compute geohash from (average) coordinates*

Description

compute geohash from (average) coordinates

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_is_polygon_cw *Check if a POLYGON or MULTIPOLYGON is clockwise*

Description

Check if a POLYGON or MULTIPOLYGON is clockwise. According to the 'Right-hand-rule', outer rings should be clockwise, and inner holes should be counter-clockwise

Usage

```
st_is_polygon_cw(x)
```

Arguments

x	object with polygon geometries

Value

logical with length the same number of features in 'x'

Examples

```
library(sf)
polys <- st_sf(cw = c(FALSE, TRUE),
               st_as_sfc(c('POLYGON ((0 0, 1 0, 1 1, 0 0))',
                           'POLYGON ((1 1, 2 2, 2 1, 1 1))')))

st_is_polygon_cw(polys)
st_is_polygon_cw(st_geometry(polys))
st_is_polygon_cw(st_geometry(polys)[[1]])
```

---

st_linesubstring *get substring from linestring*

Description

get substring from linestring

Usage

```
st_linesubstring(x, from, to, tolerance, ...)
```

Arguments

x	object of class sfc, sf or sfg
from	relative distance from origin (in [0,1])
to	relative distance from origin (in [0,1])
tolerance	tolerance parameter, when to snap to line node node
...	ignored

Value

object of class sfc

Examples

```
library(sf)
lines = st_sfc(st_linestring(rbind(c(0,0), c(1,2), c(2,0))), crs = 4326)
spl = st_linesubstring(lines, 0.2, 0.8) # should warn
plot(st_geometry(lines), col = 'red', lwd = 3)
plot(spl, col = 'black', lwd = 3, add = TRUE)
st_linesubstring(lines, 0.49999, 0.8) # three points
st_linesubstring(lines, 0.49999, 0.8, 0.001) # two points: snap start to second node
```

---

st_snap_to_grid *Snap geometries to a grid*

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
err = try(y <- st_snap_to_grid(x, 5000))

# plot data for visual comparison
if (!inherits(err, "try-error")) {
 opar = par(mfrow = c(1, 2))
 plot(x, main = "orginal data")
 plot(y, main = "snapped to 5000 m")
 par(opar)
}
```

---

st_split *Return a collection of geometries resulting by splitting a geometry*

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_combine 'd

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)))
st_split(l, pt)
```

st_startpoint *Return the start and end points from lines*

Description

Return the start and end points from lines

Usage

```
st_startpoint(x)

st_endpoint(x)
```

Arguments

x	line of class sf, sfc or sfg

Details

see https://postgis.net/docs/ST_StartPoint.html and https://postgis.net/docs/ST_EndPoint.html.

Value

sf object representing start and end points

Examples

```
library(sf)
m = matrix(c(0, 1, 2, 0, 1, 4), ncol = 2)
l = st_sfc(st_linestring(m))
lwgeom::st_startpoint(l)
lwgeom::st_endpoint(l)
l2 = st_sfc(st_linestring(m), st_linestring(m[3:1, ]))
lwgeom::st_startpoint(l2)
lwgeom::st_endpoint(l2)
```

st_subdivide *Return a collection of geometries resulting by subdividing a geometry*

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 *Transform or convert coordinates of simple features directly with Proj.4 (bypassing GDAL)*

Description

Transform or convert coordinates of simple features directly with Proj.4 (bypassing GDAL)

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	character; target CRS, or, in case of a length 2 character vector, source and target CRS
...	ignored

Details

Transforms coordinates of object to new projection, using PROJ directly rather than the GDAL API used by st_transform.

If `crs` is a single CRS, it forms the target CRS, and in that case the source CRS is obtained as `st_crs(x)`. Since this presumes that the source CRS is accepted by GDAL (which is not always the case), a second option is to specify the source and target CRS as two proj4strings in argument `crs`. In the latter case, `st_crs(x)` is ignored and may well be `NA`.

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")
```

---

st_wrap_x *Splits input geometries by a vertical line and moves components falling on one side of that line by a fixed amount*

Description

Splits input geometries by a vertical line and moves components falling on one side of that line by a fixed amount

Usage

```
st_wrap_x(x, wrap, move)
```

Arguments

`x`	object with geometries to be split
`wrap`	x value of split line
`move`	amount by which geometries falling to the left of the line should be translated to the right

Value

object of the same class as `x`

Examples

```
library(sf)
demo(nc, ask = FALSE, echo = FALSE)
x = st_wrap_x(nc, -78, 10)
plot(x[1])
```

Index