Package 'shadow'

January 19, 2017

Type Package

Title R Package for Geometric Shade Calculations

Version 0.2.0

Description
Functions for calculating (1) shade height at a single point, (2) shaded proportion of a building facade; (3) a polygonal layer of shade footprints on the ground; and (4) Sky View Factor value at a single point. Typical inputs include a polygonal layer of buildings outline along with the height of each building, sun azimuth and sun elevation. The package also provides functions for related preliminary calculations: converting polygons to line segments, finding segment azimuth, shifting segments by azimuth and distance, and constructing the footprint of a line of sight between an observer and the sun.

License MIT + file LICENSE

LazyData TRUE

Imports rgeos (>= 0.3), raster (>= 2.4-15), methods

Depends R (>= 3.2.3), sp (>= 1.1.1)

RoxygenNote 5.0.1

Suggests knitr, rmarkdown, bookdown, maptools (>= 0.8), testthat, plyr (>= 1.8.4), reshape2(>= 1.4.2)

VignetteBuilder knitr

NeedsCompilation no

Author Michael Dorman [aut, cre],
Evyatar Erell [ctb],
Itai Kloog [ctb],
Adi Vulkan [ctb]

Maintainer Michael Dorman <dorman@post.bgu.ac.il>

Repository CRAN

Date/Publication 2017-01-19 18:31:34

R topics documented:

`build` *Polygonal layer of four buildings.*

Description

A `SpatialPolygonsDataFrame` representing the outlines of four buildings. The attribute `BLDG_HT` contains building height, in meters.

Usage

```
build
```

Format

A `SpatialPolygonsDataFrame` with 4 features and 2 attributes:

build_id Building ID

BLDG_HT Building height, in meters

Details

A dataset containing the prices and other attributes of almost 54,000 diamonds.

classifyAz *Classify azimuth of line segments*

Description

Classify azimuth of line segments

Usage

```
classifyAz(sl)
```

Arguments

sl	A SpatialLines* object

Value

A list with two elements:

- az the segment azimuth values
- q the corresponding quarters

Examples

```
data(build)
build_seg = toSeg(build[1, ])
az = classifyAz(build_seg)
plot(build_seg, col = rainbow(4)[az$q])
raster::text(rgeos::gCentroid(build_seg, byid = TRUE), round(az$az))
```

deg2rad *Degrees to radians*

Description

Degrees to radians

Usage

```
deg2rad(deg)
```

Arguments

deg	Angle in degrees

Value

numeric Angle in radians

Examples

```
deg2rad(360) == 2*pi
```

---

rad2deg *Radians to degrees*

Description

Radians to degrees

Usage

```
rad2deg(rad)
```

Arguments

rad	Angle in radians

Value

numeric Angle in degrees

Examples

```
rad2deg(2*pi) == 360
```

---

ray *Line between two points*

Description

The function connects two points into a line segment.

Usage

```
ray(from, to)
```

Arguments

from	A SpatialPoints* object specifying origin.
to	A SpatialPoints* object specifying destination.

Value

A SpatialLines object.

Examples

```
ctr = rgeos::gCentroid(build)
angles = seq(0, 359, 20)
sun = mapply(
  shadow:::.sunLocation,
  sun_az = angles,
  MoreArgs = list(
    location = ctr,
    sun_elev = 10)
)
rays = mapply(ray, MoreArgs = list(from = ctr), to = sun)
rays$makeUniqueIDs = TRUE
rays = do.call(rbind, rays)
plot(rays)
sun = do.call(rbind, sun)
text(sun, as.character(angles))
```

---

shadeFootprint *Shade footprint on the ground*

Description

Creates a polygonal layer of shade footprints on the ground, given sun position and extruded obstacles (usually a buildings layer). The calculation method was inspired by Morel Weisthal's MSc thesis at Ben-Gurion University.

Usage

```
shadeFootprint(build, height_field, solar_pos, b = 0.01)
```

Arguments

build	A SpatialPolygonsDataFrame object specifying the buildings outline.
height_field	The name of the column with building height in buildings
solar_pos	A matrix with the solar azimuth (in degrees from North), and elevation
b	Buffer size for shade footprints of individual segments of a given polygon; used to eliminate minor internal holes in the resulting shade polygon.

Value

A SpatialPolygonsDataFrame object representing shade footprint plus buildings outline.

References

Weisthal, M. (2014). Assessment of potential energy savings in Israel through climate-aware residential building design (Doctoral dissertation, Ben-Gurion University of the Negev). http://aranne5.bgu.ac.il/others/WeisthalMorel.pdf

Examples

```
data(build)
location = rgeos::gCentroid(build)
time = as.POSIXct("2004-12-24 13:30:00", tz = "Asia/Jerusalem")
solar_pos = maptools::solarpos(
  matrix(c(34.7767978098526, 31.9665936050395), ncol = 2),
  time
  )
footprint = shadeFootprint(build, "BLDG_HT", solar_pos)
plot(footprint, col = adjustcolor("lightgrey", alpha.f = 0.5))
plot(build, add = TRUE, col = "darkgrey")
```

`shadeHeight` *Shade height calculation considering sun position and buildings outlines.*

Description

This function calculates shade height at a given point (`location`), taking into account:

- Buildings outline, given by a polygonal layer including a height attribute
- Sun position, given by azimuth and elevation angles

Usage

```
shadeHeight(location, build, height_field, solar_pos, b = 0.1,
  messages = TRUE)
```

Arguments

`location`	A `SpatialPoints*` object specifying the location for which to calculate shade height
`build`	A `SpatialPolygonsDataFrame` object specifying the buildings outline
`height_field`	The name of the column with building height in `build`
`solar_pos`	A matrix with two columns: solar azimuth (in degrees from North), and elevation
`b`	Buffer size when joining intersection points with building outlines, to determine intersection height
`messages`	Whether a message regarding distance units of the CRS should be displayed

Value

Shade height, in meters; `NA` if there is no shade, `Inf` if there is complete shade (i.e. sun below horizon)

Note

For a correct geometric calculation, make sure that:

- The layers `location` and `build` are projected
- The values in `height_field` of `build` are given in the same distance units as the CRS (e.g. meters when using UTM)

Examples

```
# Single location
location = rgeos::gCentroid(build)
time = as.POSIXct("2004-12-24 13:30:00", tz = "Asia/Jerusalem")
solar_pos = maptools::solarpos(
  matrix(c(34.7767978098526, 31.9665936050395), ncol = 2),
  time
  )
plot(build, main = time)
plot(location, add = TRUE)
sun = shadow:::.sunLocation(location = location, sun_az = solar_pos[1,1], sun_elev = solar_pos[1,2])
sun_ray = ray(from = location, to = sun)
build_outline = as(build, "SpatialLinesDataFrame")
inter = rgeos::gIntersection(build_outline, sun_ray)
plot(sun_ray, add = TRUE, col = "yellow")
plot(inter, add = TRUE, col = "red")
shadeHeight(location, build, "BLDG_HT", solar_pos)

## Not run:

# Grid
ext = as(raster::extent(build), "SpatialPolygons")
r = raster::raster(ext, res = 3)
proj4string(r) = proj4string(build)
grid = raster::rasterToPoints(r, spatial = TRUE)
grid = sp::SpatialPointsDataFrame(grid, data.frame(grid_id = 1:length(grid)))
height_field = "BLDG_HT"
for(i in 1:length(grid)) {
  grid$shade_height[i] =
    shadeHeight(grid[i, ], build, height_field, solar_pos, messages = FALSE)
}
shade = as(grid, "SpatialPixelsDataFrame")
shade = raster::raster(shade, layer = "shade_height")
plot(shade, col = grey(seq(0.9, 0.2, -0.01)), main = time)
raster::contour(shade, add = TRUE)
plot(build, add = TRUE, border = "red")


## End(Not run)
```

shadeImageWall *Create an image of wall shading*

Description

The function divides a given wall to a grid of rectangular 'cells', then finds whether each cell is shaded (at least partially) or not, by repeatedly calling shadeHeight on regular sample points along the wall facade.

Usage

```
shadeImageWall(seg, seg_height_field, build, build_height_field, solar_pos,
  sample_dist = 1, shift_dist = 0.01, messages = TRUE)
```

Arguments

seg	A SpatialLinesDataFrame object representing the wall footprint
seg_height_field	The name of the column with wall height in seg
build	A SpatialPolygonsDataFrame object specifying the buildings outline.
build_height_field	The name of the column with building height in build
solar_pos	A matrix with two columns: solar azimuth (in degrees from North), and elevation
sample_dist	Horizontal sampling distance of seg for creating the shade image
shift_dist	The distance for shifting the examined locations away from wall to avoid self-shading. Default is 1 cm.
messages	Whether a message regarding distance units of the CRS should be displayed.

Value

A data.frame representing the shade image of seg, with the following columns:

- solar_pos_row The corresponding sun position (i.e. the respective row in the solar_pos matrix
- width The horizontal distance along wall facade, from left to right, with 0 representing the left 'side' of the wall (when the viewer stands in front of it)
- height_upper, height_ctr, height_lower

Examples

```
data(build)
time = as.POSIXct("2004-12-24 12:30:00", tz = "Asia/Jerusalem")
solar_pos = maptools::solarpos(
  matrix(c(34.7767978098526, 31.9665936050395), ncol = 2),
  time
)
seg = shadow::toSeg(build[2, ])[5, ]

# Show wall position on a map
plot(build)
plot(seg, add = TRUE, col = "red", lwd = 3)

# Calculate wall 'image'
img = shadow::shadeImageWall(
  seg = seg,
  seg_height_field = "BLDG_HT",
  build = build,
  build_height_field = "BLDG_HT",
  solar_pos = solar_pos,
  sample_dist = 1,
  shift_dist = 0.01
)

# Plot wall image
z = reshape2::acast(img, width ~ height_ctr, value.var = "shade")
image(
  x = sort(unique(img$width)),
  y = sort(unique(img$height_ctr)),
  z = z,
  asp = 1, axes = FALSE, frame.plot = FALSE,
  xlab = "Ground distance (m)", ylab = "Height (m)",
  col = c("yellow", "grey")
)
rect(
  xleft = min(img$width) - max(diff(sort(img$width)))/2,
  ybottom = min(img$height_ctr) - max(diff(sort(img$width)))/2,
  xright = max(img$width) + max(diff(sort(img$width)))/2,
  ytop = max(img$height_ctr) + max(diff(sort(img$width)))/2
)
axis(side = 1, labels = TRUE)
axis(side = 2, labels = TRUE)
```

---

shadePropWall *Shaded proportion of building walls*

Description

Calculates the shaded proportion of a segment (usually representing a building wall), given the segment vertical height as well as the outlines and heights of obstacles (usually buildings).

Usage

```
shadePropWall(seg, seg_height_field, build, build_height_field, solar_pos,
  sample_dist = 1, shift_dist = 0.01, messages = TRUE)
```

Arguments

seg	A SpatialLinesDataFrame object where each feature is a single segment representing a wall.
seg_height_field	The name of the column with wall height in seg
build	A SpatialPolygonsDataFrame object specifying the buildings outline
build_height_field	The name of the column with building height in build
solar_pos	A matrix with the solar azimuth (in degrees from North), and elevation
sample_dist	Distance between sampling points along wall
shift_dist	The distance for shifting the examined locations away from wall to avoid self-shading. Default is 1 cm.
messages	Whether a message regarding distance units of the CRS should be displayed.

Value

Proportion of shaded area of the given segment.

Note

For a correct geometric calculation, make sure that:

- The layers location and build are projected
- The values in height_field of build are given in the same distance units as the CRS (e.g. meters when using UTM)

Examples

```
data(build)
time = as.POSIXct("2004-12-24 13:30:00", tz = "Asia/Jerusalem")
solar_pos = maptools::solarpos(
  matrix(c(34.7767978098526, 31.9665936050395), ncol = 2),
  time
)
seg = shadow::toSeg(build[2, ])[5:10, ]

# Calculate shade proportion for walls 5-10 of 2nd building
props = shadePropWall(
  seg = seg,
  seg_height_field = "BLDG_HT",
  build = build,
  build_height_field = "BLDG_HT",
  solar_pos = solar_pos,
```

```
  sample_dist = 1,
  shift_dist = 0.01
)

# Plot
plot(build)
plot(seg, add = TRUE, col = "red", lwd = 2)
raster::text(rgeos::gCentroid(seg, byid = TRUE), round(props, 2), cex = 0.75)
```

---

shadow *shadow: R Package for Geometric Shade Calculations*

Description

Functions for calculating:

- Shade height at a single point
- Shaded proportion of a building facade
- A polygonal layer of shade footprints on the ground
- Sky View Factor (SVF) value at a single point

Typical inputs include a polygonal layer of buildings outline along with the height of each building, sun azimuth and sun elevation. The package also provides functions for related preliminary calculations: converting polygons to line segments, finding segment azimuth, shifting segments by azimuth and distance, and constructing the footprint of a line of sight between an observer and the sun.

---

shiftAz *Shift features by azimuth and distance.*

Description

Shift features by azimuth and distance.

Usage

```
shiftAz(object, az, dist)
```

Arguments

object	The object to be shifted.
az	Shift azimuth, in decimal degrees.
dist	Shift distance, in object projection units.

Value

The shifted `object`.

Examples

```
data(build)
s = c(270, 90, 180, 0)
build_shifted = shiftAz(build, az = s, dist = 2.5)
plot(build)
plot(build_shifted, add = TRUE, border = "red")
raster::text(rgeos::gCentroid(build, byid = TRUE), s)
```

`SVF` *Sky View Factor (SVF) calculation*

Description

Calculates the Sky View Factor (SVF) at a given location, given extruded obstacles (usually a buildings layer).

Usage

```
SVF(location, build, height_field, res = 5, b = 0.01)
```

Arguments

`location`	A `SpatialPoints*` object specifying the location for which to calculate shade height
`build`	A `SpatialPolygonsDataFrame` object specifying the buildings outline.
`height_field`	The name of the column with building height in `build`
`res`	Circular sampling resolution, in decimal degrees. Default is 5 degrees, i.e. 0, 5, 10... 355.
`b`	Buffer size when joining intersection points with building outlines, to determine intersection height.

Value

A numeric value between 0 (sky completely obstructed) and 1 (sky completely visible).

Examples

```
data(build)
location0 = rgeos::gCentroid(build)
svf = SVF(location0, build, "BLDG_HT")
location1 = raster::shift(location0, 0, -15)
svf1 =  SVF(location1, build, "BLDG_HT")
location2 = raster::shift(location0, -10, 20)
svf2 =  SVF(location2, build, "BLDG_HT")

plot(build)
plot(location0, add = TRUE)
plot(location1, add = TRUE)
plot(location2, add = TRUE)
raster::text(location0, round(svf, 2), pos = 3)
raster::text(location1, round(svf1, 2), pos = 4)
raster::text(location2, round(svf2, 2), pos = 3)
```

---

toGMT *Local time to GMT*

Description

The function transforms a POSIXct object in any given time zone to GMT.

Usage

```
toGMT(time)
```

Arguments

time	Time, a POSIXct object.

Value

A a POSIXct object, in GMT.

Examples

```
time = as.POSIXct("1999-01-01 12:00:00", tz = "Asia/Jerusalem")
toGMT(time)
```

toSeg *Convert polygons or lines to segments*

Description

Split lines or polygons to separate segments.

Usage

```
toSeg(x)
```

Arguments

x	A SpatialLines* or a SpatialPolygons* object.

Value

A SpatialLines object where each segment is represented by a separate feature.

References

This function uses a modified version of code from the following 'r-sig-geo' post by Roger Bivand: https://stat.ethz.ch/pipermail/r-sig-geo/2013-April/017998.html

Examples

```
data(build)
seg = toSeg(build[1, ])
plot(seg, col = sample(rainbow(length(seg))))
raster::text(rgeos::gCentroid(seg, byid = TRUE), 1:length(seg))
```

Index