Package ‘googlePolylines’

November 23, 2018

Type Package
Title Encoding Coordinates into 'Google' Polylines
Version 0.7.2
Date 2018-11-23
Description Encodes simple feature ('sf') objects and coordinates, and decodes polylines using the 'Google' polyline encoding algorithm (<https://developers.google.com/maps/documentation/utilities/polylinealgorithm>).
License GPL-3
SystemRequirements C++11
Depends R (>= 3.3.0)
Imports Rcpp (>= 0.12.13)
LinkingTo Rcpp, BH
RoxygenNote 6.1.0
Suggests covr, knitr, rmarkdown, sf, testthat
VignetteBuilder knitr
NeedsCompilation yes
Author David Cooley [aut, cre],
   Paulo Barcelos [ctb] (Author of c++ decode_polyline),
   Chris Muir [ctb]
Maintainer David Cooley <dcooley@symbolix.com.au>
Repository CRAN
Date/Publication 2018-11-23 05:30:03 UTC

R topics documented:

sfencode-package .................................................. 2
decode ................................................................. 3
code ................................................................. 3
codeCoordinates .................................................... 5
geometryRow .......................................................... 6
sfencode-package

polyline_wkt ................................................................. 7
sfAttributes ................................................................. 8
wkt_polyline ................................................................. 9

Index 10

sfencode-package A short title line describing what the package does

Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.
Maintainer: Your Name <your@email.com>

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

## Not run:
## Optional simple examples of the most important functions
## These can be in \dontrun{} and \donttest{} blocks.

## End(Not run)
decode  

**Decode Polyline**

**Description**
Decodes encoded polylines into a list of data.frames.

**Usage**
decode(polylines)

**Arguments**
polylines vector of encoded polyline strings

**Examples**
```r
polylines <- c(
  "ohlbdnbhnsuq@am\{tAw\'qsAeyhGvkz\'@fge\}A",
  "ggmnDtm\}wmlgc\}DesuQvvrLofo\}DrqGtzzV"
)

decode(polylines)
```

---

encode  

**Encode**

**Description**
Encodes coordinates into an encoded polyline.

**Usage**
encode(obj, ...)

```r
## S3 method for class 'sf'
encode(obj, strip = FALSE, ...)

## S3 method for class 'data.frame'
encode(obj, lon = NULL, lat = NULL, 
byrow = FALSE, ...)
```
encode

Arguments

- obj: either an sf object or data.frame
- ...: other parameters passed to methods
- strip: logical indicating if sf attributes should be stripped. Useful if you want to reduce the size even further, but you will lose the spatial attributes associated with the sf object
- lon: vector of longitudes
- lat: vector of latitudes
- byrow: logical indicating if the encoding should be done for each row

Details

The function assumes Google Web Mercator projection (WSG 84 / EPSG:3857 / EPSG:900913) for inputs and outputs.

Will work with
- sf and sfc objects directly
- data.frames - It will attempt to find lat & lon coordinates, or you can explicitly define them using the lat and lon arguments

Value

sfencoded object

Note

When an sfencoded object is column-subset using `\[ \` and the encoded column is retained, the attributes of the column will remain. This is different behaviour to standard subsetting of data.frames, where all attributes are dropped by default. See examples.

When encoding an sf object, only the XY dimensions will be used, the Z or M (3D and/or Measure) dimensions are dropped.

See Also

encodeCoordinates

Examples

```
## data.frame
df <- data.frame(polygonId = c(1,1,1),
                 lineId = c(1,1,1),
                 lon = c(-80.190, -66.118, -64.757, -80.190),
                 lat = c(26.774, 18.466, 32.321, 26.774))

## on a data.frame, it will attempt to find the lon & lat columns
encode(df)
```
## encodeCoordinates

**Encode coordinates**

### Description

Encodes a vector of lon & lat coordinates

### Usage

`encodeCoordinates(lon, lat)`

### Arguments

- `lon`  
  vector of longitudes
- `lat`  
  vector of latitudes
See Also

encode

Examples

## Not run:

## Grouping by polygons and lines

```r
df <- data.frame(polygonId = c(1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,1,1,1),
        lineId = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
        lon = c(-80.190, -66.118, -64.757, -80.190, -70.579, -67.514, -66.668, -70.579,
                -70, -49, -51, -70),
        lat = c(26.774, 18.466, 32.321, 26.774, 28.745, 29.570, 27.339, 28.745,
                22, 23, 22, 22))

## using dplyr groups

library(dplyr)

df %>%
  group_by(polygonId, lineId) %>%
  summarise(polyline = encodeCoordinates(lon, lat))

## using data.table

library(data.table)

setDT(df)

df[, encodeCoordinates(lon = lon, lat = lat), by = .(polygonId, lineId)]

## End(Not run)
```

---

**geometryRow**

**Geometry Row**

**Description**

Extracts specific geometry rows of an sfencoded object

**Usage**

```r
geometryRow(x, geometry = c("POINT", "LINESTRING", "POLYGON"),
            multi = TRUE)
```

**Arguments**

- **x**: sfencoded object
- **geometry**: the specific geometry to extract
- **multi**: logical indicating if MULTI geometry objects are included
Value

the row indexes for the requested geometry

Examples

```r
# Not run:

df <- data.frame(myId = c(1,1,1,1,1,1,1,1,2,2,2,2,2),
                 lineId = c(1,1,1,1,2,2,2,2,1,1,1,2),
                 lon = c(-80.190, -66.118, -64.757, -80.190, -70.579, -67.514, -66.668, -70.579, -70, -49, -51, -70),

p1 <- as.matrix(df[1:4, c("lon", "lat")])
p2 <- as.matrix(df[5:8, c("lon", "lat")])
p3 <- as.matrix(df[9:12, c("lon", "lat")])

point <- sf::st_sfc(sf::st_point(x = c(df[1,"lon"], df[1,"lat"])))
multipoint <- sf::st_sfc(sf::st_multipoint(x = as.matrix(df[1:2, c("lon", "lat")])))
polygon <- sf::st_sfc(sf::st_polygon(x = list(p1, p2)))
linestring <- sf::st_sfc(sf::st_linestring(p3))
multilinestring <- sf::st_sfc(sf::st_multilinestring(list(p1, p2)))
multipolygon <- sf::st_sfc(sf::st_multipolygon(x = list(list(p1, p2), list(p3))))

sf <- rbind(
  st_sf(geo = polygon),
  st_sf(geo = multilinestring),
  st_sf(geo = linestring),
  st_sf(geo = point)
)

encode(sf)

enc <- encode(sf)
geometryRow(enc, "POINT")
geometryRow(enc, "LINESTRING")
geometryRow(enc, "POLYGON")

# End(Not run)
```

---

**Description**

Converts encoded polylines into well-known text.
Usage

polyline_wkt(obj)

Arguments

obj sfencoded object or encoded_column of encoded polylines

Details

'Polylines' refers to lat/lon coordinates encoded into strings using Google’s polyline encoding algorithm.
The function assumes Google Web Mercator projection (WSG 84 / EPSG:3857 / EPSG:900913) for inputs and outputs.

Value

well-known text representation of the encoded polylines

Note

This will not work if you have specified strip = TRUE for encode()

Examples

## Not run:

library(sf)
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"))

## encode to polylines
enc <- encode(nc)

## convert encoded lines to well-known text
wkt <- polyline_wkt(enc)

## End(Not run)

---

sfAttributes sf Attributes

Description

Retrieves the sf attributes stored on the sfencoded object

Usage

sfAttributes(x)
**wkt_polyline**

**Arguments**

- `x`  
  sfencoded object

**Value**

- list of sf attributes

---

**Description**

Converts well-known text into encoded polylines.

**Usage**

```r
wkt_polyline(obj)
```

**Arguments**

- `obj`  
  sfencoded object or wkt_column of well-known text

**Details**

'Polylines' refers to lat/lon coordinates encoded into strings using Google’s polyline encoding algorithm.

**Value**

encoded polyline representation of geometries

**Examples**

```r
## Not run:

library(sf)
c <- sf::st_read(system.file("shape.nc.shp", package="sf"))

## encode to polylines
c <- encode(c)

## convert encoded lines to well-known text
c <- polyline_wkt(c)

## convert well-known text back to polylines
c <- wkt_polyline(wkt)

## End(Not run)
```
Index

*Topic package
  sfencode-package, 2

decode, 3
encode, 3, 6
encodeCoordinates, 4, 5
geometryRow, 6
polyline_wkt, 7
sfAttributes, 8
sfencode (sfencode-package), 2
sfencode-package, 2
wkt_polyline, 9