Package 'flexpolyline'

May 8, 2026

Type Package

Title Flexible Polyline Encoding

Version 0.3.0

Description Binding to the C++ implementation of the flexible polyline encoding by HERE <https://github.com/heremaps/flexible-polyline>. The flexible polyline encoding is a lossy compressed representation of a list of coordinate pairs or coordinate triples. The encoding is achieved by:
(1) Reducing the decimal digits of each value;
(2) encoding only the offset from the previous point;
(3) using variable length for each coordinate delta; and
(4) using 64 URL-safe characters to display the result.

License GPL-3

URL https://munterfi.github.io/flexpolyline/, https://github.com/munterfi/flexpolyline/

BugReports https://github.com/munterfi/flexpolyline/issues/

LinkingTo Rcpp

Imports Rcpp, sf (>= 0.9-3)

Suggests testthat (>= 2.3.2), stringr (>= 1.4.0), knitr (>= 1.28), rmarkdown (>= 2.1), covr (>= 3.5.0)

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation yes

Author Merlin Unterfinger [aut, cre] (ORCID: <https://orcid.org/0000-0003-2020-2366>), HERE Europe B.V. [aut, cph] (Flexible polyline encoding C++ implementation)

Maintainer Merlin Unterfinger <info@munterfinger.ch>

Repository CRAN

Date/Publication 2023-02-12 22:40:02 UTC

Contents

`decode` *Decode a flexible polyline encoded string*

Description

This function calls `hf::polyline_decode` and `hf::get_third_dimension` of the C++ implementation of the flexible polyline encoding by HERE. Depending on the dimensions of the encoded line, a two or three dimensional line is decoded.

Usage

```
decode(encoded)
```

Arguments

`encoded`	character, encoded flexible polyline string.

Value

A matrix containing the coordinates of the decoded line.

Examples

```
# 2d line
decode("BFoz5xJ67i1B1B7PzIhaxL7Y")

# 3d line
decode("BlBoz5xJ67i1BU1B7PUzIhaUxL7YU")
```

decode_sf *Wrapper function for decoding to simple features*

Description

A wrapper function for decode that converts the input polylines, encoded in the flexible polyline enoding, to simple feature geometries of the sf package.

Usage

```
decode_sf(encoded, crs = sf::NA_crs_)
```

Arguments

encoded	character, encoded flexible polyline string.
crs	integer or character, coordinate reference system to assign to the sf object (default = sf::NA_crs_).

Value

An sf object, containing the geometries of the decoded lines (Geometry type: "LINESTRING").

Note

The function returns an sf object, therefore the input set of encoded polylines must be of consistent dimension (e.g "XY", "XYM" or "XYZ") to meet the requirements of the constructor of sf objects. For mixed dimensions use the decode function directly.

Examples

```
decode_sf("B1Voz5xJ67i1Bgkh9B")
decode_sf("BFoz5xJ67i1B1B7PlU9yB")
decode_sf("BlXoz5xJ67i1Bgkh9B1B7Pgkh9BzIhagkh9BqK-pB_ni6D")
```

encode *Encode a line in the flexible polyline encoding format*

Description

This function calls hf::polyline_encode of the C++ implementation of the flexible polyline encoding by HERE. Depending on the dimensions of the input coordinates, a two or three dimensional line is encoded.

Usage

```
encode(line, precision = 5L, third_dim = 3L, third_dim_precision = 5L)
```

Arguments

`line`	matrix, coordinates of the line in 2d or 3d (column order: LNG, LAT, DIM3).
`precision`	integer, precision to use in encoding (between 0 and 15, `default=5`).
`third_dim`	integer, type of the third dimension (0: ABSENT, 1: LEVEL, 2: ALTITUDE, 3: ELEVATION, 4, 6: CUSTOM1, 7: CUSTOM2, `default=3`).
`third_dim_precision`	integer, precision to use in encoding for the third dimension (between 1 and 15, `default=5`).

Value

The line as string in the flexible polyline encoding format.

Examples

```
# 2D
line2d <- matrix(
  c(8.69821, 50.10228,
    8.69567, 50.10201,
    8.69150, 50.10063,
    8.68752, 50.09878),
  ncol = 2, byrow = TRUE
)
encode(line2d)

# 3D
line3d <- matrix(
  c(8.69821, 50.10228, 10,
    8.69567, 50.10201, 20,
    8.69150, 50.10063, 30,
    8.68752, 50.09878, 40),
  ncol = 3, byrow = TRUE
)
encode(line3d)
```

`encode_sf` *Wrapper function for encoding simple features*

Description

A wrapper function for encode that converts simple feature geometries of the sf package to flexible polyline encoded strings.

Usage

```
encode_sf(
  geom,
  precision = 5,
  third_dim = NULL,
  third_dim_precision = precision
)
```

Arguments

geom	simple feature, sf, sfc or sfg object with geometry type "POINT", "LINESTRING" or "POLYGON".
precision	integer, precision to use in encoding (between 0 and 15, default=5).
third_dim	integer, type of the third dimension (0: ABSENT, 1: LEVEL, 2: ALTITUDE, 3: ELEVATION, 4, 6: CUSTOM1, 7: CUSTOM2, default=NULL).
third_dim_precision	integer, precision to use in encoding for the third dimension (between 1 and 15, default=precision).

Value

The line as string in the flexible polyline encoding format.

Examples

```
# 3D point
point3d <- sf::st_point(
  matrix(c(8.69821, 50.10228, 10), ncol = 3, byrow = TRUE),
  dim = "XYZ"
)
encode_sf(point3d)

# 2D linestring
line2d <- sf::st_linestring(
  matrix(c(
    8.69821, 50.10228,
    8.69567, 50.10201,
    8.68752, 50.09878
  ), ncol = 2, byrow = TRUE)
)
encode_sf(line2d)

# 3D polygon
poly3d <- sf::st_polygon(list(
  matrix(c(
    8.69821, 50.10228, 10,
    8.69567, 50.10201, 20,
    8.69150, 50.10063, 30,
    8.69821, 50.10228, 10
  ), ncol = 3, byrow = TRUE)
```

```
), dim = "XYM")
encode_sf(poly3d)
```

`get_third_dimension` *Get third dimension of a flexible polyline encoded string*

Description

This function calls `hf::get_third_dimension` of the C++ implementation of the flexible polyline encoding by HERE and return the type of the third dimension.

Usage

```
get_third_dimension(encoded)
```

Arguments

`encoded`	character, encoded flexible polyline string.

Value

A string describing the third dimension.

Examples

```
# 2d line
get_third_dimension("BFoz5xJ67i1B1B7PzIhaxL7Y")

# 3d line
get_third_dimension("BlBoz5xJ67i1BU1B7PUzIhaUxL7YU")
```

`set_third_dimension` *Set third dimension of a flexible polyline encoded string*

Description

This function decodes the flexible polyline encoded line, changes the third dimension and encodes the line again.

Usage

```
set_third_dimension(
  encoded,
  third_dim_name,
  precision = 5L,
  third_dim_precision = 5L
)
```

Arguments

encoded	character, encoded flexible polyline string.
third_dim_name	character, name of the third dimension to set (ABSENT, LEVEL, ALTITUDE, ELEVATION, CUSTOM1, CUSTOM2).
precision	integer, precision to use in encoding (between 0 and 15, default=5).
third_dim_precision	integer, precision to use in encoding for the third dimension (between 1 and 15, default=5).

Value

The line with the new third dimension as string in the flexible polyline encoding format.

Note

The precision is not read from the header of the encoded line. Therefore it must be provided as a parameter for re-encoding.

Examples

```
# 2d line (nothing happens...)
set_third_dimension("BFoz5xJ67i1B1B7PzIhaxL7Y", "ELEVATION")

# 3d line
set_third_dimension("BlBoz5xJ67i1BU1B7PUzIhaUxL7YU", "ELEVATION")
```

Index