Package ‘wkb’

October 12, 2022

Type Package

Title Convert Between Spatial Objects and Well-Known Binary Geometry

Version 0.4-0

Imports sp

Suggests testthat

Author TIBCO Software Inc.

Maintainer Ian Cook <ianmcook@gmail.com>

Description Utility functions to convert between the 'Spatial' classes specified by the package 'sp', and the well-known binary '(WKB)' representation for geometry specified by the 'Open Geospatial Consortium'. Supports 'Spatial' objects of class 'SpatialPoints', 'SpatialPointsDataFrame', 'SpatialLines', 'SpatialLinesDataFrame', 'SpatialPolygons', and 'SpatialPolygonsDataFrame'. Supports 'WKB' geometry types 'Point', 'LineString', 'Polygon', 'MultiPoint', 'MultiLineString', and 'MultiPolygon'. Includes extensions to enable creation of maps with 'TIBCO Spotfire'.

BugReports https://github.com/ianmcook/wkb/issues

NeedsCompilation no

License BSD_3_clause + file LICENSE

Encoding UTF-8

RoxygenNote 6.1.1

Repository CRAN

Date/Publication 2019-12-05 19:00:02 UTC

R topics documented:

hex2raw *Convert String Hex Representation to Raw Vector*

Description

Converts a string hexadecimal representation to a `raw` vector.

Usage

```
hex2raw(hex)
```

Arguments

`hex`	character string or character vector containing a hexadecimal representation.

Details

Non-hexadecimal characters are removed.

Value

A `raw` vector.

The return value is a `list` of `raw` vectors when the argument `hex` contains more than one hexadecimal representation.

See Also

`raw2hex` in package **PKI**, `readWKB`

Examples

```
# create a character string containing a hexadecimal representation
hex <- "0101000000000000000000f03f0000000000000840"

# convert to raw vector
wkb <- hex2raw(hex)


# create a character vector containing a hexadecimal representation
hex <- c("01", "01", "00", "00", "00", "00", "00", "00", "00", "00", "00",
         "f0", "3f", "00", "00", "00", "00", "00", "00", "08", "40")

# convert to raw vector
wkb <- hex2raw(hex)


# create vector of two character strings each containing a hex representation
hex <- c("0101000000000000000000f03f0000000000000840",
         "010100000000000000000000400000000000000040")
```

```
# convert to list of two raw vectors
wkb <- hex2raw(hex)
```

`readWKB` *Convert* WKB *to Spatial Objects*

Description

Converts well-known binary (WKB) geometry representations to `Spatial` objects.

Usage

```
readWKB(wkb, id = NULL, proj4string = CRS(as.character(NA)))
```

Arguments

`wkb`	`list` in which each element is a `raw` vector consisting of a WKB geometry representation.
`id`	character vector of unique identifiers of geometries. The length of `id` must be the same as the length of the `wkb` list.
`proj4string`	projection string of class `CRS`.

Details

Supported WKB geometry types are Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon. All elements in the `list` must have the same WKB geometry type. The WKB geometry representations may use little-endian or big-endian byte order.

The argument `wkb` may also be a `raw` vector consisting of one WKB geometry representation. In that case, the argument `id` must have length one.

Value

An object inheriting class `Spatial`.

The return value may be an object of class `SpatialPoints`, `SpatialLines`, `SpatialPolygons`, or a `list` in which each element is an object of class `SpatialPoints`. The class of the return value depends on the WKB geometry type as shown in the table below.

Type of WKB geometry	**Class of return value**
Point	`SpatialPoints`
LineString	`SpatialLines`
Polygon	`SpatialPolygons`
MultiPoint	`list` of `SpatialPoints`
MultiLineString	`SpatialLines`
MultiPolygon	`SpatialPolygons`

See Also

writeWKB, hex2raw

Examples

```
# create a list of WKB geometry representations of type Point
wkb <- list(
  as.raw(c(0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
           0xf0, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x40)),
  as.raw(c(0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
           0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40))
)

# convert to object of class SpatialPoints
obj <- readWKB(wkb)


# create a list of WKB geometry representations of type MultiPoint
wkb <- list(
  as.raw(c(0x01, 0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x01,
           0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x3f,
           0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x40)),
  as.raw(c(0x01, 0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x01,
           0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40,
           0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40)))

# convert to list of objects of class SpatialPoints
obj <- readWKB(wkb)


# create a list of WKB geometry representations of type MultiLineString
wkb <- list(
  as.raw(c(0x01, 0x05, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x02,
           0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
           0x00, 0x00, 0xf0, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08,
           0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00,
           0x00, 0x00, 0x00, 0x00, 0x00, 0x40)),
  as.raw(c(0x01, 0x05, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x02,
           0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
           0x00, 0x00, 0xf0, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0,
           0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00,
           0x00, 0x00, 0x00, 0x00, 0xf8, 0x3f)))

# convert to object of class SpatialLines
obj <- readWKB(wkb)


# create a list of WKB geometry representations of type Polygon
wkb <- list(
  as.raw(c(0x01, 0x03, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x05, 0x00,
           0x00, 0x00, 0x34, 0x03, 0xf0, 0xac, 0xce, 0x66, 0x5d, 0xc0, 0x8f,
           0x27, 0x95, 0x21, 0xab, 0xa6, 0x44, 0x40, 0xa0, 0x32, 0x81, 0x18,
           0x78, 0x83, 0x5d, 0xc0, 0xc8, 0xd2, 0xa0, 0xee, 0x23, 0x0b, 0x41,
           0x40, 0x80, 0xec, 0x72, 0x54, 0xde, 0xb1, 0x5f, 0xc0, 0xc8, 0xd2,
           0xa0, 0xee, 0x23, 0x0b, 0x41, 0x40, 0xec, 0x1b, 0x04, 0xc0, 0x87,
           0xce, 0x5f, 0xc0, 0x8f, 0x27, 0x95, 0x21, 0xab, 0xa6, 0x44, 0x40,
           0x34, 0x03, 0xf0, 0xac, 0xce, 0x66, 0x5d, 0xc0, 0x8f, 0x27, 0x95,
           0x21, 0xab, 0xa6, 0x44, 0x40)),
  as.raw(c(0x01, 0x03, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x05, 0x00,
           0x00, 0x00, 0x08, 0x36, 0xdc, 0x8b, 0x9f, 0x3d, 0x51, 0xc0, 0x0f,
           0xb3, 0x2a, 0x6a, 0x3f, 0x1c, 0x46, 0x40, 0x47, 0xcb, 0x54, 0xe7,
           0xcb, 0x5e, 0x51, 0xc0, 0x45, 0x81, 0x50, 0x31, 0xfa, 0x80, 0x42,
           0x40, 0xa9, 0xba, 0x74, 0x6d, 0xf5, 0xa1, 0x53, 0xc0, 0x45, 0x81,
           0x50, 0x31, 0xfa, 0x80, 0x42, 0x40, 0xe8, 0x4f, 0xed, 0xc8, 0x21,
           0xc3, 0x53, 0xc0, 0x0f, 0xb3, 0x2a, 0x6a, 0x3f, 0x1c, 0x46, 0x40,
           0x08, 0x36, 0xdc, 0x8b, 0x9f, 0x3d, 0x51, 0xc0, 0x0f, 0xb3, 0x2a,
           0x6a, 0x3f, 0x1c, 0x46, 0x40)))

# convert to object of class SpatialPolygons
obj <- readWKB(wkb)


# specify id and proj4string
obj <- readWKB(
  wkb,
  id = c("San Francisco", "New York"),
  proj4string = sp::CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
)
```

`writeEnvelope` *Envelope of Spatial Objects*

Description

Takes a `Spatial` object and returns a data frame with six columns representing the envelope of each element in the `Spatial` object.

Usage

```
writeEnvelope(obj, centerfun = mean)
```

Arguments

`obj`	object inheriting class `Spatial`.
`centerfun`	function to apply to the x-axis limits and y-axis limits of the bounding box to obtain the x-coordinate and y-coordinate of the center of the bounding box.

Details

`obj` may be an object of class `SpatialPoints`, `SpatialPointsDataFrame`, `SpatialLines`, `SpatialLinesDataFrame`, `SpatialPolygons`, or `SpatialPolygonsDataFrame`, or a `list` in which each element is an object of class `SpatialPoints` or `SpatialPointsDataFrame`.

Value

A data frame with six columns named XMax, XMin, YMax, YMin, XCenter, and YCenter. The first four columns represent the corners of the bounding box of each element in `obj`. The last two columns represent the center of the bounding box of each element in `obj`. The number of rows in the returned data frame is the same as the length of the argument `obj`.

When this function is run in TIBCO Enterprise Runtime for R (TERR), the columns of the returned data frame have the SpotfireColumnMetaData attribute set to enable TIBCO Spotfire to recognize them as containing envelope information.

See Also

Example usage at `writeWKB`

`writeWKB` *Convert Spatial Objects to* WKB

Description

Converts `Spatial` objects to well-known binary (WKB) geometry representations.

Usage

```
writeWKB(obj, endian = "little")
```

Arguments

`obj`	object inheriting class `Spatial`.
`endian`	The byte order (`"big"` or `"little"`) for encoding numeric types. The default is `"little"`.

Details

The argument `obj` may be an object of class `SpatialPoints`, `SpatialPointsDataFrame`, `SpatialLines`, `SpatialLinesDataFrame`, `SpatialPolygons`, or `SpatialPolygonsDataFrame`, or a `list` in which each element is an object of class `SpatialPoints` or `SpatialPointsDataFrame`.

Value

A `list` with class `AsIs`. The length of the returned list is the same as the length of the argument `obj`. Each element of the returned list is a `raw` vector consisting of a WKB geometry representation. The WKB geometry type depends on the class of `obj` as shown in the table below.

Class of `obj`	**Type of WKB geometry**
`SpatialPoints` or `SpatialPointsDataFrame`	Point
`list` of `SpatialPoints` or `SpatialPointsDataFrame`	MultiPoint
`SpatialLines` or `SpatialLinesDataFrame`	MultiLineString
`SpatialPolygons` or `SpatialPolygonsFrame`	Polygon or MultiPolygon

A SpatialPolygons or SpatialPolygonsFrame object is represented as WKB Polygons if each Polygons object within it represents a single polygon; otherwise it is represented as WKB MultiPolygons.

The byte order of numeric types in the returned WKB geometry representations depends on the value of the argument endian. Little-endian byte order is known as NDR encoding, and big-endian byte order is known as XDR encoding.

When this function is run in TIBCO Enterprise Runtime for R (TERR), the return value has the SpotfireColumnMetaData attribute set to enable TIBCO Spotfire to recognize it as a WKB geometry representation.

See Also

writeEnvelope, readWKB

Examples

```
# load package sp
library(sp)

# create an object of class SpatialPoints
x = c(1, 2)
y = c(3, 2)
obj <- SpatialPoints(data.frame(x, y))

# convert to WKB Point
wkb <- writeWKB(obj)


# create a list of objects of class SpatialPoints
x1 = c(1, 2, 3, 4, 5)
y1 = c(3, 2, 5, 1, 4)
x2 <- c(9, 10, 11, 12, 13)
y2 <- c(-1, -2, -3, -4, -5)
Sp1 <- SpatialPoints(data.frame(x1, y1))
Sp2 <- SpatialPoints(data.frame(x2, y2))
obj <- list("a"=Sp1, "b"=Sp2)

# convert to WKB MultiPoint
wkb <- writeWKB(obj)


# create an object of class SpatialLines
l1 <- data.frame(x = c(1, 2, 3), y = c(3, 2, 2))
l1a <- data.frame(x = l1[, 1] + .05, y = l1[, 2] + .05)
l2 <- data.frame(x = c(1, 2, 3), y = c(1, 1.5, 1))
Sl1 <- Line(l1)
Sl1a <- Line(l1a)
Sl2 <- Line(l2)
S1 <- Lines(list(Sl1, Sl1a), ID = "a")
S2 <- Lines(list(Sl2), ID = "b")
obj <- SpatialLines(list(S1, S2))
```

```
# convert to WKB MultiLineString
wkb <- writeWKB(obj)


# create an object of class SpatialPolygons
triangle <- Polygons(
  list(
    Polygon(data.frame(x = c(2, 2.5, 3, 2), y = c(2, 3, 2, 2)))
  ), "triangle")
rectangles <- Polygons(
   list(
     Polygon(data.frame(x = c(0, 0, 1, 1, 0), y = c(0, 1, 1, 0, 0))),
     Polygon(data.frame(x = c(0, 0, 2, 2, 0), y = c(-2, -1, -1, -2, -2)))
   ), "rectangles")
obj <- SpatialPolygons(list(triangle, rectangles))

# convert to WKB MultiPolygon
wkb <- writeWKB(obj)


# use the WKB as a column in a data frame
ds <- data.frame(ID = c("a","b"), Geometry = wkb)

# calculate envelope columns and cbind to the data frame
coords <- writeEnvelope(obj)
ds <- cbind(ds, coords)
```

Index