Package 'sgo'

October 14, 2022

Title Simple Geographical Operations (with OSGB36)

Version 0.9.2

URL https://github.com/clozanoruiz/sgo

BugReports https://github.com/clozanoruiz/sgo/issues

Language en-GB

Description Methods focused in performing the OSGB36/ETRS89 transformation (Great Britain and the Isle of Man only) by using the Ordnance Survey's OSTN15/OSGM15 transformation model. Calculation of distances and areas from sets of points defined in any of the supported Coordinated Systems is also available.

Depends R (>= 3.5.0)

Suggests maps, tinytest

License BSD_2_clause + file LICENSE

Copyright OSTN transformation data included in this module remains Crown Copyright (C) 2016 under the terms of the BSD licence: (c) Copyright and database rights Ordnance Survey Limited 2016, (c) Crown copyright and database rights Land & Property Services 2016 and/or (c) Ordnance Survey Ireland, 2016. All rights reserved.

Encoding UTF-8

ByteCompile yes

RoxygenNote 7.2.1

Collate 'sgo.R' 'sgo_area.R' 'sgo_bng.R' 'sgo_distance.R' 'sgo_laea.R' 'sgo_ngr.R' 'sgo_points.R' 'sgo_set_gcs.R' 'sgo_wgs84.R' 'sgo_transform.R'

NeedsCompilation no

Author Carlos Lozano Ruiz [aut, cre]

Maintainer Carlos Lozano Ruiz <carloslozanoruiz@outlook.com>

Repository CRAN

Date/Publication 2022-09-23 16:20:02 UTC

R topics documented:

`sgo-package` *sgo: Simple Geographical Operations (with OSGB36).*

Description

The sgo package aims to help with spatial or geographic analysis in Open Source R and derivatives. Its main purpose is to perform OSGB36/ETRS89 transformations using the Ordnance Survey's OSTN15 transformation model for Great Britain and the Isle of Man. It also transforms GPS ellipsoid heights to orthometric (mean sea level) heights on the relevant Ordnance Survey mapping datum, using the National Geoid Model OSGM15.

Details

Most of the functions available in this package will become much less accurate if used outside the coverage of OSTN, therefore it is advised to apply these functions on coordinates within Great Britain, the Isle of Man and any areas of sea less than a few miles off shore.

This package assumes that the Coordinate Reference Systems (CRS) ETRS89 and WGS84 are practically the same within the UK, but this shouldn't be a problem for most civilian use of GPS satellites. If a high-precision transformation between WGS84 and ETRS89 is required then it is recommended to use a different package to do the conversion.

Object constructors

- `sgo_points`: 2D/3D point coordinates

Transformation and Conversion functions

Functions to provide coordinate transformations:

- sgo_bng_lonlat: British National Grid (E/N) to Lon/Lat
- sgo_lonlat_bng: Lon/Lat to British National Grid (E/N)
- sgo_bng_ngr: British National Grid (E/N) to National Grid References
- sgo_ngr_bng: National Grid References to British National Grid (E/N)
- sgo_laea_etrs: ETRS89-LAEA Easting/Northing to ETRS89
- sgo_etrs_laea: ETRS89 to ETRS89-LAEA Easting/Northing
- sgo_cart_lonlat: 3D Earth Centred Earth Fixed (ECEF) Cartesian coordinates to polar coordinates
- sgo_lonlat_cart: Polar coordinates to 3D ECEF Cartesian coordinates
- sgo_wgs84_en: WGS84 Lon/Lat to Pseudo-Mercator (E/N)
- sgo_en_wgs84: Pseudo-Mercator (E/N) to WGS84 Lon/Lat
- sgo_transform: Wrapper for all the transformations above
- sgo_coordinates: Extract coordinates from a sgo_points object

Geometric measurements

- sgo_area: Calculate area from an ordered set of points
- sgo_distance: Calculate distance(s) between points

Disclaimer

The OSTN15 transformation model is used in this package, and it is licensed under the BSD Licence (http://opensource.org/licenses/bsd-license.php):

---

sgo_area *Calculate area from an ordered set of points*

Description

Calculates the planar area for a set of points defined in the OS BNG or ETRS89-LAEA. An accurate approximation of the geodetic area is calculated when points are expressed in angular coordinates.

Usage

```
sgo_area(x, interpolate = NULL, ...)
```

Arguments

x	A sgo_points object describing an ordered set of points.
interpolate	Numeric variable. If not NULL, defines the maximum distance in metres between adjacent coordinates. It is only used with angular coordinates.
...	Currently ignored

Details

Calculate areas using the Gauss's area formula (https://en.wikipedia.org/wiki/Shoelace_formula).

When using angular coordinates the function performs an approximation of the geodetic area following the methodology discussed in Berk & Ferlan (2018) where the area on the ellipsoid is determined by using a region-adapted equal-area projection (Albers Equal-Area Conic) with one standard parallel. The standard parallel and the projection origin are tied to the moment centroid of the polygon.

Boundary segments can be divided by interpolating vertices on the projected geodesic to reduce the error introduced by boundary simplification and to provide an even more accurate area computation for angular coordinates. For instance, if `interpolate = 500` then any segment between adjacent coordinates whose length is greater than `interpolate` will be split in parts no greater than `500 m` and new vertices will be added.

The area calculation in this package is best suited for features that would be represented in a large or medium scale (like plots or council boundaries). It will provide much less accurate results for features usually represented at smalle scale (countries, continents, etc.).

Value

Value of the area in squared metres rounded up to the first decimal.

References

Sandi Berk & Miran Ferlan, 2018. *Accurate area determination in the cadaster: case study of Slovenia*. Cartography and Geographic Information Science, 45:1, 1-17. DOI: 10.1080/15230406.2016.1217789

Snyder, J.P. 1987. *Map Projections — A Working Manual*. US Geological Survey Professional Paper, no. 1395. Washington, DC: US Government Printing Office. DOI: 10.3133/pp1395

Examples

```
lon <- c(-6.43698696, -6.43166843, -6.42706831, -6.42102546,
-6.42248238, -6.42639092, -6.42998435, -6.43321409)
lat <- c(58.21740316, 58.21930597, 58.22014035, 58.22034112,
58.21849188, 58.21853606, 58.21824033, 58.21748949)
A <- sgo_area(sgo_points(list(lon, lat), epsg=4326))
```

sgo_bng_lonlat *British National Grid (BNG) Easting/Northing to Geodetic Coordinate System (GCS)*

Description

Converts Ordnance Survey grid reference easting/northing coordinates to GCS longitude/latitude (SW corner of grid square).

Usage

```
sgo_bng_lonlat(x, to = 4258, OSTN = TRUE, OD = FALSE)
```

Arguments

x	A sgo_points object with coordinates defined in the projected coordinate system BNG (EPSGs 27700 or 7405).
to	Numeric. Sets the epsg code of the destination Geodetic Coordinate System. 4258 (ETRS89) by default.
OSTN	Logical variable indicating whether use OSTN15 transformation when TRUE or a less accurate but slightly faster single Helmert transformation when FALSE.
OD	Logical variable. When TRUE, and the output contains a column with heights, then a new column is added to the result indicating the ordnance datum (OD) used on each point. It is ignored when OSTN=FALSE.

Details

The UK Ordnance Survey defined 'OSGB36' as the datum for the UK, based on the 'Airy 1830' ellipsoid. However, in 2014, they deprecated OSGB36 in favour of ETRS89 for longitude/latitude coordinates. Thus, when converting to longitude/latitude the OSGB36 datum should be always converted to ETRS89 (or WGS84).

According to the Transformations and OSGM15 User Guide, p. 8: *"...ETRS89 is a precise version of the better known WGS84 reference system optimised for use in Europe; however, for most purposes it can be considered equivalent to WGS84."* and *"For all navigation, mapping, GIS, and engineering applications within the tectonically stable parts of Europe (including UK and Ireland), the term ETRS89 should be taken as synonymous with WGS84."* This means that ETRS89 and WGS84 datums will be considered equivalent by this routine.

If, for historical reasons, longitude/latitude coordinates must have the old OSGB36 datum, then the parameter to must be set to 4277.

Note: Grid references rounded to whole metres will give latitude/longitude that are accurate to about 5 decimal places: in Great Britain, 1/100000 of a degree of latitude is about 70cm and 1/100000 of a degree of longitude is about 1m. All those coordinates outside the rectangle covered by OSTN15 will be automatically computed using the small Helmert transformation. Such coordinates will be accurate up to about +/-5 metres. Converting from BNG to lon/lat coordinates is slower than the other way around, due to the iterative nature of the algorithm that is built into OSTN15.

Value

An object of class `sgo_points` whose coordinates are defined as Longitude/Latitude.If `OD=TRUE` a column named `height.datum` is added to the resulting object.

References

Ordnance Survey Limited, 2018. *Transformations and OSGM15 user guide*

See Also

`sgo_points`, `sgo_lonlat_bng`, `sgo_coordinates`, `sgo_transform`.

Examples

```
p <- sgo_points(list(651409.903, 313177.270), epsg=27700)
p.89 <- sgo_bng_lonlat(p) #ETRS89 lon/lat
p.36 <- sgo_bng_lonlat(p, to=4277) #OSGB36 lon/lat
```

---

sgo_bng_ngr *BNG Easting/Northing to National Grid References (NGR)*

Description

Converts BNG Easting/Northing coordinates to National Grid References

Usage

```
sgo_bng_ngr(x, digits = 10)
```

Arguments

`x`	A `sgo_points` object with coordinates defined as `epsg=27700` or `epsg=7405`.
`digits`	Numeric. It defines the precision of the resulting grid references.

Details

All resulting grid references will have 10 digits (1m × 1m square) by default. In order to reduce the output precision change the digits parameter accordingly. When `digits=0`, it returns the numeric format of the grid references.

Note that National Grid references are truncated instead of being rounded when converting to less precise references (as the OS system demands). By doing so, the grid reference refers to the lower left corner of the relevant square - to ensure the more precise polygon will remain within the boundaries of the less precise polygon.

Value

A list with at least one column named 'ngr'.

See Also

sgo_points, sgo_ngr_bng.

Examples

```
sgo <- sgo_points(list(x=247455, y=706338, name="Ben Venue"),
coords=c("x", "y"), epsg=27700)
grid10 <- sgo_bng_ngr(sgo)
grid8 <- sgo_bng_ngr(sgo, digits=8)
#and notice the truncating, not rounding, of grid8 regarding grid10.
```

---

sgo_cart_lonlat *Geodetic Coordinate System (GCS) in cartesian coordinates to polar coordinates*

Description

Converts a GCS expressed Earth-centered Earth-fixed (ECEF) cartesian coordinate to Longitude and Latitude and Ellipsoid Height.

Usage

```
sgo_cart_lonlat(x)
```

Arguments

x	A sgo_points object with coordinates expressed in cartesian coordinates

Details

Currently converts from EPSGs 4936 and 4978 to 4937 and 4979

Value

An object of class sgo_points with polar coordinates (Longitude, Latitude and Ellipsoid Height).

See Also

sgo_points, sgo_bng_lonlat.

Examples

```
p <- sgo_points(list(3487823.234, -305433.201, 5313739.634), epsg=4936)
p.xyz <- sgo_cart_lonlat(p) #Cartesian coordinates
```

sgo_coordinates *Extracts coordinates from an* sgo_points *object*

Description

Extract the coordinates of an sgo_points object expressed as a matrix.

Usage

```
sgo_coordinates(x, names.xyz = NULL, as.latlon = FALSE,
ll.format=NULL)
```

Arguments

x	An instance of sgo_points.
names.xyz	Character vector. New names for the columns x, y and possibly z of the object x.
as.latlon	Logical variable. When x is defined in a geodetic coordinate system as lon/lat and this parameter is set to TRUE then it returns the coordinates ordered as lat/lon.
ll.format	Character variable. Applies a format to the returned coordinates when x is defined in a geodetic coordinate system. As of now it only accepts DMS, which will return strings of coordinates formatted as degrees, minutes and seconds (certain accuracy will be lost because seconds are rounded to the second decimal).

Value

A matrix with 2 or 3 named columns.

Examples

```
p <- sgo_points(list(57.47777, -4.22472), epsg=4326)
coords <- sgo_coordinates(p)
```

sgo_distance *Calculate distance(s) between points*

Description

Calculates the distance between OS National Grid Reference points. Points with angular coordinates will use the Harvesine or Vicenty formulae.

Usage

```
sgo_distance(x, y, by.element = FALSE,
  which = ifelse(isTRUE(x$epsg==27700 || x$epsg==7405), "BNG", "Vicenty"),
  grid.true.distance = ifelse(isTRUE(x$epsg==27700 || x$epsg==7405),
  TRUE, FALSE), iterations = 100L)
```

Arguments

x	A sgo_points object describing a set of points in a geodetic coordinate system.
y	A sgo_points object, defaults to x.
by.element	Logical variable. If TRUE, return a vector with distance between the first elements of x and y, the second, etc. If FALSE, return the dense matrix with all pairwise distances.
which	Character vector. For geodetic coordinates one of Harvesine or Vicenty. It defaults to BNG for points in 'OS British National Grid' coordinates.
grid.true.distance	Logical variable. Currently only used for BNG coordinates. If TRUE it returns the true (geodesic) distance.
iterations	Numeric variable. Maximum number of iterations used in the Vicenty method.

Details

This function can use two different methods when working with geodetic coordinates: When which = "Vicenty" the Vincenty's formula is used to calculate the geodesics (distance) on an ellipsoid to an accuracy of up to a millimetre. If such accuracy is not needed, which can also accept the string "Harvesine" which calculates the great-circle distance between two points on a sphere. Harvesines are faster to compute than the Vicenty distances but can result in an error of up to 0.5%.

When working with (BNG) planar coordinates the Local Scale Factor is the scale distortion inherent in the map projection at a point. When grid.true.distance is TRUE the function computes a line scale factor using Simpson's Rule to achieve greater accuracy and approximate the distance to the true geodesic distance. When it is FALSE the Euclidean distance in the plane is calculated.

Note: Considering F as the scale factor, we have that S (True distance) = s (grid distance) / F
For most purposes the scale factor can be taken as constant for distances up to 20km (errors not exceeding 1 or 2 parts er million) and equal to the mid point of the line. For longer lines, this routine computes a scale factor for both ends (F1 and F2) and the mid point (Fm) and then uses the Simpson's Rule:
F = 1/6(F1 + 4Fm + F2)

Value

If by.element is FALSE sgo_distance returns a dense numeric matrix of dimension length(x) by length(y). Otherwise it returns a numeric vector of length x or y, the shorter one being recycled. Distances involving empty geometries are NA. All distances are returned in metres.

References

Thaddeus Vincenty, 1975. *Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations*. Survey Review, 23:176, 88-93, DOI: 10.1179/sre.1975.23.176.88

Examples

```
p1 <- sgo_points(list(-3.9369, 56.1165), epsg=4326)
lon <- c(-4.25181,-3.18827)
lat <- c(55.86424, 55.95325)
pts <- sgo_points(list(longitude=lon, latitude=lat), epsg=4326)
p1.to.pts <- sgo_distance(p1, pts, by.element = TRUE)

## Perimeter of a polygon defined as a series of ordered points:
lon <- c(-6.43698696, -6.43166843, -6.42706831, -6.42102546,
-6.42248238, -6.42639092, -6.42998435, -6.43321409)
lat <- c(58.21740316, 58.21930597, 58.22014035, 58.22034112,
58.21849188, 58.21853606, 58.21824033, 58.21748949)
pol <- sgo_points(list(lon, lat), epsg=4326)
# Create a copy of the polygon with its coordinates shifted one position
# so that we can calculate easily the distance between vertices
coords <- sgo_coordinates(pol)
pol.shift.one <- sgo_points(rbind(coords[-1, ], coords[1, ]), epsg=pol$epsg)
perimeter <- sum(sgo_distance(pol, pol.shift.one, by.element=TRUE))
```

---

sgo_en_wgs84 *Pseudo - Mercator to WGS84 Longitude/Latitude*

Description

Converts Pseudo - Mercator coordinates to WGS84 (EPSG=4326)

Usage

```
sgo_en_wgs84(x, to = 4326)
```

Arguments

x	A sgo_points object describing a set of points in the geodetic coordinate system EPSG=3857.
to	Numeric. Sets the epsg code of the destination Geodetic Coordinate System. 4326 (WGS84) by default. And currently doesn't support any other.

Details

Currently converts ONLY from EPSG 3857 to 4326 (Longitude/Latitude).

Value

An object of class sgo_points whose coordinates are defined as Longitude/Latitude.

References

IOGP Publication 373-7-2 - Geomatics Guidance Note number 7, part 2 (October 2020). https://epsg.org/guidance-notes.html

See Also

sgo_points, sgo_wgs84_en.

Examples

```
p <- sgo_points(list(-11169055.58, 2810000.00), epsg=3857)
res <- sgo_en_wgs84(p)
```

---

sgo_etrs_laea *ETRS89 to ETRS89-LAEA Easting/Northing*

Description

Converts ETRS89 geodetic coordinates to ETRS89-LAEA Easting/Northing (EPSG:3035)

Usage

```
sgo_etrs_laea(x)
```

Arguments

x	A sgo_points object describing a set of points in the geodetic coordinate system EPSG=4258, 4937 or 4936.

Details

ETRS89-LAEA (EPSG:3035) is a CRS for pan-European statistical mapping at all scales or other purposes where true area representation is required.

Value

An object of class sgo_points whose coordinates are defined as Easting/Northing in the EPSG:3035 Projected Coordinate System.

References

IOGP Publication 373-7-2 - Geomatics Guidance Note number 7, part 2 (October 2020). https://epsg.org/guidance-notes.html)

See Also

sgo_points, sgo_area.

Examples

```
p <- sgo_points(list(-3.9369, 56.1165), epsg=4258)
prj <- sgo_etrs_laea(p)
```

`sgo_laea_etrs` *ETRS89-LAEA Easting/Northing to ETRS89 geodetic coordinates*

Description

Converts ETRS89-LAEA Easting/Northing to ETRS89 geodetic coordinates (EPSG:4258)

Usage

```
sgo_laea_etrs(x)
```

Arguments

x	A `sgo_points` object describing a set of points in the projected coordinate system EPSG=3035.

Details

ETRS89-LAEA (EPSG:3035) is a CRS for pan-European statistical mapping at all scales or other purposes where true area representation is required.

Value

An object of class `sgo_points` whose coordinates are defined as Longitude/Latitude in the ETRS89 Coordinate Reference System.

References

IOGP Publication 373-7-2 - Geomatics Guidance Note number 7, part 2 (October 2020). https://epsg.org/guidance-notes.html

See Also

`sgo_points`, `sgo_area`.

Examples

```
prj <- sgo_points(list(3962799.45, 2999718.85), epsg=3035)
p <- sgo_laea_etrs(prj)
```

`sgo_lonlat_bng` *GCS to BNG Easting/Northing*

Description

Converts a geodetic coordinate system to BNG (projected) Easting/Northing coordinates. It also transforms GPS ellipsoid heights to orthometric (mean sea level) heights on the relevant Ordnance Survey mapping datum, using the National Geoid Model OSGM15.

Usage

```
sgo_lonlat_bng(x, to=27700, OSTN=TRUE, OD=FALSE)
```

Arguments

`x`	A `sgo_points` object with coordinates defined in a Geodetic Coordinate System expressed as Longitude and Latitude (e.g. epsg=4258, 4937, 4326, 4979 or 4277).
`to`	Specifies the EPSG code to convert the coordinates to. It can only take the following values: `27700` or `7405`.
`OSTN`	Logical variable indicating whether use OSTN15 transformation when TRUE or a less accurate but slightly faster single Helmert transformation when FALSE.
`OD`	Logical variable. When TRUE, and the output contains a column with heights, then a new column is added to the result indicating the ordnance datum (OD) used on each point. It is ignored when `OSTN=FALSE`.

Details

The UK Ordnance Survey defined 'OSGB36' as the datum for the UK, based on the 'Airy 1830' ellipsoid. However, in 2014, they deprecated OSGB36 in favour of ETRS89 for longitude/latitude coordinates. Thus, `x` should be defined as ETRS89 (or WGS84) most of the times.

Note: When transforming from EPSG=4277 any height included in the input will be simply discarded (see `sgo_points`).

According to the Transformations and OSGM15 User Guide, p. 8: *"...ETRS89 is a precise version of the better known WGS84 reference system optimised for use in Europe; however, for most purposes it can be considered equivalent to WGS84."* and *"For all navigation, mapping, GIS, and engineering applications within the tectonically stable parts of Europe (including UK and Ireland), the term ETRS89 should be taken as synonymous with WGS84."* This means that EPSGs with the ETRS89 datum or WGS84 will be considered equivalent by this routine.

Note: All those coordinates outside the rectangle covered by OSTN15 will be automatically computed using the small Helmert transformation. Such coordinates will be accurate up to about +/-5 metres, therefore the resulting easting and northing coordinates will be rounded to the metre. Since those coordinates are outside of the OSTN15 domain the resulting coordinates will have the resulting height defined as `NA`. Similarly, when using `OSTN=FALSE` on 3D coordinates, the result will have all the heights defined as `NA`. Converting from lon/lat to BNG coordinates is faster than the other way around, due to the iterative nature of the algorithm that is built into OSTN15.

Value

An object of class `sgo_points` whose coordinates are defined as Easting/Northing (epsg=27700 or 7405). They are adjusted to the SW corner of 1m grid square. If `OD=TRUE` a column named `height.datum` is added to the resulting object.

References

Ordnance Survey Limited, 2018. *Transformations and OSGM15 user guide*

See Also

`sgo_points`, `sgo_bng_lonlat`, `sgo_coordinates`, `sgo_transform`.

Examples

```
lon <- c(-4.25181,-3.18827)
lat <- c(55.86424, 55.95325)
pts <- sgo_points(list(longitude=lon, latitude=lat), epsg=4326)
bng.pts <- sgo_lonlat_bng(pts)
```

---

sgo_lonlat_cart *Geodetic Coordinate System (GCS) in polar coordinates to cartesian coordinates*

Description

Converts a GCS expressed in Longitude and Latitude (and Ellipsoid Height) to an Earth-centered Earth-fixed (ECEF) cartesian coordinate system.

Usage

```
sgo_lonlat_cart(x)
```

Arguments

`x`	A `sgo_points` object with coordinates expressed as Longitude and Latitude (and Ellipsoid Height if they are 3D points).

Details

Currently converts from EPSGs `4258` and `4937` to `4936` or from EPSGs `4326`, `4979` to `4978`

Value

An object of class `sgo_points` whose coordinates are defined as a x, y and z cartesian vector.

See Also

`sgo_points`, `sgo_lonlat_bng`.

Examples

```
p <- sgo_points(list(-5.00355049, 56.7968571), epsg=4326)
p.xyz <- sgo_lonlat_cart(p) #Cartesian coordinates
```

---

sgo_ngr_bng *NGR to BNG Easting/Northing*

Description

Converts OS National Grid References to Easting/Northing coordinates

Usage

```
sgo_ngr_bng(x, col = NULL, check.only = FALSE)
```

Arguments

x	A data.frame, list or vector containing strings describing OS National Grid References, with or without whitespace separators. (e.g. 'SU 387 148').
col	Character string with the name of the 'column' containing the vector of NGR values, it is required when x is a list or data.frame with more than one column.
check.only	Logical parameter. If it is set to TRUE then the routine returns a logical vector indicating which references are correct.

Details

All entered standard grid references can range from two-digit references up to 10-digit references (1m × 1m square). If x is a list with 2 or more vector elements, col can be used to inform the function which of the elements contains the NGR strings. The rest of the elements will be appended to the resulting object. See examples.

Value

An object of class sgo_points whose coordinates are defined as Easting/Northing when check.only is kept as FALSE. Otherwise, it returns a logical vector indicating which grid references are correct and which ones are not.

See Also

sgo_points, sgo_bng_ngr.

Examples

```
vec <- c("NN 166 712", "HU38637653")
lst <- list(vec)
v <- sgo_ngr_bng(vec)
l <- sgo_ngr_bng(lst)

# any additional column (here 'attr') will be added to the result
extra <- list(p=c("NN 166712", "HU38637653"),
              attr=c("name1","name2"))
res <- sgo_ngr_bng(extra, col="p")
res

# grid references returned by sgo_bng_ngr are within an
# element (column) named 'ngr'
grid <- sgo_bng_ngr(sgo_points(list(x=247455, y=706338, name="Ben Venue"),
                               coords=c("x","y"),
                               epsg=27700))
bng <- sgo_ngr_bng(grid, col="ngr")

# test
bad <- c("NN 166 712", "AA 3863 7653")
check <- sgo_ngr_bng(bad, check.only=TRUE) #returns a logical vector
```

---

sgo_points *Object containing 2D or 3D point coordinates*

Description

2D or 3D coordinates (and other attributes) of a point or collection of points

Usage

```
sgo_points(x, coords = NULL, epsg = NULL)

## S3 method for class 'sgo_points'
print(x, ..., n = 6L)

## S3 method for class 'sgo_points'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sgo_points'
as.list(x, ...)
```

Arguments

x	A matrix, list or dataframe with at least 2 columns of either easting/northing or longitude/latitude coordinates per row. A column with height values is optional. **Please note** that the order is important when x has only 2 or 3 columns and

coords is not informed: lat/lon or northing/easting (and height) will produce wrong results.

`coords` A vector with the names of the two or three columns containing the X (easting or longitude), Y (northing or latitude) and optionally Z (ellipsoid or orthometric height) coordinates.

`epsg` Specifies the EPSG code of coordinates to store. It can take any of the following values:

- when working with (2D/3D) ETRS89 Datum: `4258`, `4937`, `4936`, `3035`
- when working with (2D/3D) WGS84 Datum:`4326`, `4979`, `4978`
- when working with (2D/3D) OSGB36 Datum:`4277`, `27700`, `7405`
- WGS84/Pseudo-Mercator (Google Maps, OpenStreetMap, etc.): `3857`

`...` Further arguments passed to or from other methods, see print, as.data.frame or as.list .

`n` Maximum number of features to print.

`row.names` NULL or a character vector giving the row names for the data frame. Missing values are not allowed.

`optional` Logical. See as.data.frame

Details

This object stores 2D or 3D point coordinates and any other column-list of attributes related to each point. Note that additional column-lists will be expanded with `NA` values if they contain less elements than coordinates. Currently it only supports the following `epsg`s:

- `4258`: ETRS89, geodetic coordinate system. The columns in `x` must be defined as Longitude and Latitude (`sgo` also accepts a third column for ellipsoid heights). The defined datum for this set of coordinates is ETRS89 (https://epsg.io/4258).
- `4937`: ETRS89, geodetic coordinate system. The columns in `x` must be defined as Longitude, Latitude and Ellipsoid Heights respectively. The defined datum for this set of coordinates is ETRS89 (https://epsg.io/4937).
- `4936`: ETRS89, geodetic coordinate system. The columns in `x` must be defined as cartesian coordinates x, y and z. The defined datum for this set of coordinates is ETRS89 (https://epsg.io/4936).
- `3035`: ETRS-LAEA, projected coordinate system. The columns in `x` must be defined as Easting and Northing. The defined datum for this set of coordinates is ETRS89 (https://epsg.io/3035)
- `4326`: WGS84, geodetic coordinate system. The columns in `x` must be defined as Longitude and Latitude (`sgo` also accepts a third column for ellipsoid heights). The defined datum for this set of coordinates is WGS84 (https://epsg.io/4326).
- `4979`: WGS84, geodetic coordinate system. The columns in `x` must be defined as Longitude, Latitude and Ellipsoid Height respectively. The defined datum for this set of coordinates is WGS84 (https://epsg.io/4979)
- `4978`: WGS84, geodetic coordinate system. The columns in `x` must be defined as cartesian coordinates x, y and z. The defined datum for this set of coordinates is WGS84 (https://epsg.io/4978)

- `4277`: OSGB36, geodetic coordinate system. The 2 columns in `x` must be defined as Longitude and Latitude values respectively. The defined datum for this set of coordinates is OSGB 1936 (https://epsg.io/4277). *Coordinates defined like this should only be used for historical reasons and to convert only to or from BNG coordinates.* **Height values will be discarded when working with this coordinate system.**
- `27700`: British National Grid, projected coordinate system. The columns in `x` must be defined as Easting and Northing (`sgo` also accepts a third column for orthometric heights). The defined datum for this set of coordinates is OSGB 1936 (https://epsg.io/27700).
- `7405`: British National Grid, projected coordinate system. The columns in `x` must be defined as Easting, Northing and ODN Orthometric height respectively (`sgo` accepts heights from other datums like Orkney, Lerwick, Stornoway, Douglas, St.Marys and 'Newlyn offshore'). The defined datum for this set of coordinates is OSGB 1936 (https://epsg.io/7405).
- `3857`: WGS 84 / Pseudo-Mercator, projected coordinate system. The columns in `x` must be defined as Easting and Northing. The defined datum for this set of coordinates is WGS84 (https://epsg.io/3857)

Value

An object of class `sgo_points`. This object is a actually a list with class `sgo_points` and at least 5 elements (or 6 elements if it is 3D):

- `x`: A numeric vector containing easting or longitude coordinates.
- `y`: A numeric vector with northing or latitude coordintes.
- `z`: A numeric vector with height values when the object is 3D.
- `epsg`: A scalar value with the EPSG code of the current Geographic Coordinate System (GCS).
- `datum`: A string describing the geodetic datum that defines the GCS of the object. Currently can take the values "OSGB36", "WGS84" or "ETRS89"
- `dimension`: A string describing whether the object is 2D or 3D. It can take the values "XY" or "XYZ".

See Also

`sgo_coordinates`, `sgo_transform`.

Examples

```
# lists:
p1 <- sgo_points(list(-3.9369, 56.1165), epsg=4326)
lon <- c(-4.25181,-3.18827)
lat <- c(55.86424, 55.95325)
p2 <- sgo_points(list(longitude=lon, latitude=lat), epsg=4326)
#p3 will fill up the list 'desc' with NA's to have the same number of
#elements as coordinates in the list:
p3 <- sgo_points(list(longitude=lon, latitude=lat, desc="c1"),
                 coords=c("longitude", "latitude"), epsg=4326)
# dataframe:
ln <- c(-4.22472, -2.09908)
```

```
lt <- c(57.47777, 57.14965)
n <- c("Inverness", "Aberdeen")
df <- data.frame(n, ln, lt, stringsAsFactors = FALSE)
p4 <- sgo_points(df, coords=c("ln", "lt"), epsg=4326)

# plotting on a map:
if (require(maps)) {
  map('world', regions=('uk'), xlim=c(-9, 0), ylim=c(54.5, 60.9))
  points(x=p1$x, y=p1$y, pch=0, col="green") #Stirling
  points(p4, pch=0, col="red")
  text(p4, labels=p4$n, pos=1, cex=0.9)
}
```

`sgo_set_gcs` *Set GCS of a set of points*

Description

Changes the geodetic coordinate system of a set of points using a single Helmert transformation.

Usage

```
sgo_set_gcs(x, to = NULL)
```

Arguments

`x`	A `sgo_points` object describing a set of points in a geodetic coordinate system.
`to`	Specifies the EPSG code to convert the coordinates to. Currently it can take any of the following values: `4258`, `4937`, `4936`, `4326`, `4979`, `4978` or `4277`.

Details

Changes the geodetic coordinate system of a set of points. Note that the precision of various datums will vary, and the original WGS-84 is not defined to be accurate to better than ±1 metre. Most transformations shouldn't be assumed to be accurate to better than a meter; between OSGB36 and WGS84 somewhat less - the lost of accuracy can be up to ±5m when using single Helmert transformations.

Input points with a projected coordinate system (e.g. 27700, 7405, 3035 or 3857) are not allowed.

Warning This function is mainly for internal use of the program. Since it relies on a single Helmert transformation it is not recommended to call it directly. Use any other of the transformation functions available (sgo-package).

Value

An object of class 'sgo_points'.

See Also

`sgo_points`, `sgo_transform`.

Examples

```
lon <- c(-4.25181,-3.18827)
lat <- c(55.86424, 55.95325)
p <- sgo_points(list(longitude=lon, latitude=lat), epsg=4326)
# warning: a single Helmert transformation is used in the next transformation
p2 <- sgo_set_gcs(p, to=4277)
# if higher precision is required to transform between OSGB36 lon/lat and
# ETRS89/WGS84 lon/lat then use the OSTN15 transformation (will be slower):
# Transform from WGS84 lon/lat coordinates to EPSG:4277 using OSTN15
p2 <- sgo_transform(p, to=4277)
```

---

sgo_transform *Coordinate transformation of a set of points*

Description

Transforms the coordinate system of a set of points to any supported coordinate system.

Usage

```
sgo_transform(x, to = NULL, ...)
```

Arguments

x	A sgo_points object.
to	Specifies the EPSG code to convert the coordinates to. See sgo_points for a list of supported EPSG codes.
...	Additional parameters passed to internal functions. Currently it supports the additional arguments seen in sgo_bng_lonlat and sgo_lonlat_bng.

Details

This function is a wrapper of specific transformation functions (sgo_bng_lonlat, sgo_en_wgs84, sgo_lonlat_bng, sgo_wgs84_en, sgo_laea_etrs, sgo_etrs_laea, sgo_cart_lonlat, sgo_lonlat_cart) that transforms the coordinate system of a set of points to any of the supported coordinate systems.

Please note that this package assumes that the Coordinate Reference Systems (CRS) ETRS89 and WGS84 are the same within the UK, but this shouldn't be a problem for most civilian use of GPS satellites. If a high-precision transformation between WGS84 and ETRS89 is required then it is recommended to use a different package to do the conversion.

According to the Transformations and OSGM15 User Guide, p. 8: *"...ETRS89 is a precise version of the better known WGS84 reference system optimised for use in Europe; however, for most purposes it can be considered equivalent to WGS84."* and *"For all navigation, mapping, GIS, and engineering applications within the tectonically stable parts of Europe (including UK and Ireland), the term ETRS89 should be taken as synonymous with WGS84."*.

Warning: Coordinates defined in the Geodetic Coordinate System EPSG:4277 (with datum OSGB 1936) should only be used to convert to or from BNG coordinates and for historical reasons only.

Value

An object of class 'sgo_points'.

See Also

sgo_points, sgo_coordinates, sgo_set_gcs, sgo_bng_ngr

Examples

```
ln <- c(-4.22472, -2.09908)
lt <- c(57.47777, 57.14965)
n <- c("Inverness", "Aberdeen")
df <- data.frame(n, ln, lt, stringsAsFactors = FALSE)
locations <- sgo_points(df, coords=c("ln", "lt"), epsg=4326)

locations.bng <- sgo_transform(locations, to=27700)
locations.osgb36 <- sgo_transform(locations, to=4277)
locations.ngr <- sgo_bng_ngr(sgo_transform(locations, to=27700))
locations.wgs84EN <- sgo_transform(locations.bng, to=3857)
```

---

sgo_wgs84_en *WGS84 to Easting Northing (Pseudo - Mercator)*

Description

Converts WGS84 coordinates to Easting/Northing (Pseudo-Mercator)

Usage

```
sgo_wgs84_en(x, to = 3857)
```

Arguments

x	A sgo_points object describing a set of points in the geodetic coordinate system EPSG=4326 or 4979.
to	Numeric. Sets the epsg code of the destination Geodetic Coordinate System. 3857 (WGS84) by default. And currently doesn't support any other.

Details

This routine also accepts source data expressed in ETRS89 coordinates (EPSG=4258 or 4937) as it is considered the difference between those two GCS is far less than the accuracy available when working with Pseudo-Mercator coordinates.

The results can be used in maps where Pseudo-Mercator coordinates are needed. Usually, those include Google, Bing, OpenStreetMap and several other webmap applications.

Value

An object of class `sgo_points` whose coordinates are defined as Easting/Northing.

References

IOGP Publication 373-7-2 - Geomatics Guidance Note number 7, part 2 (October 2020). https://epsg.org/guidance-notes.html

See Also

`sgo_points`, `sgo_en_wgs84`.

Examples

```
p <- sgo_points(list(-3.9369, 56.1165), epsg=4326)
res <- sgo_wgs84_en(p)
```

Index