Package 'reproj'

April 15, 2020

Type Package

Title Coordinate System Transformations for Generic Map Data

Version 0.4.2

Description Transform coordinates from a specified source to a specified target map projection. This uses the 'PROJ' library directly, by wrapping the 'PROJ' package (if functional), otherwise the 'proj4' package. The 'reproj()' function is generic, methods may be added to remove the need for an explicit source definition. If 'proj4' is in use 'reproj()' handles the requirement for conversion of angular units where necessary. This is for use primarily to transform generic data formats and direct leverage of the underlying 'PROJ' library. (There are transformations that aren't possible with 'PROJ' and that are provided by the 'GDAL' library, a limitation which users of this package should be aware of.) The 'PROJ' library is available at <https://proj.org/>.

License GPL-3

LazyData TRUE

Depends R (>= 3.2.5)

Imports proj4, PROJ (>= 0.1.6), crsmeta (>= 0.3.0)

Suggests testthat, covr

RoxygenNote 7.1.0

Encoding UTF-8

SystemRequirements PROJ (>= 4.4.6)

BugReports https://github.com/hypertidy/reproj/issues/

URL https://github.com/hypertidy/reproj/

NeedsCompilation no

Author Michael D. Sumner [aut, cre] (<https://orcid.org/0000-0002-2471-7511>)

Maintainer Michael D. Sumner <mdsumner@gmail.com>

Repository CRAN

Date/Publication 2020-04-15 14:50:06 UTC

R topics documented:

`reproj-package` *Reproject data from source to target coordinate system.*

Description

reproj provides helpers for easily reprojecting generic data, by depending on a reprojection engine (proj4 for now).

Details

The function `reproj` is designed to take an input data set `x` and then a `target` coordinate system specification. The `source` argument is not positional (must be named) and must be provided. Currently the coordinate system may be a 'PROJ string' or 'EPSG code' either as a number or text.

Methods are provided for data frame and matrix, add S3 methods for you classes in your own package. For classed objects, or objects with a known method for finding the 'source' coordinate system your method can provide that logic.

See reproj for global options to control assumptions about data that is input in longitude latitude form.

There is an option set at start up `reproj.mock.noproj6` which is designed for testing the support in the PROJ package. Even if this package is functional this option can be set to true so that reproj falls-back to use the proj4 package instead.

`reproj.sc` *Reproject coordinates.*

Description

Reproject coordinates from a matrix or data frame by explicitly specifying the 'source' and 'target' projections.

Usage

```
## S3 method for class 'sc'
reproj(x, target = NULL, ..., source = NULL)

## S3 method for class 'mesh3d'
reproj(x, target, ..., source = NULL)
```

```
## S3 method for class 'quadmesh'
reproj(x, target, ..., source = NULL)

## S3 method for class 'triangmesh'
reproj(x, target, ..., source = NULL)

reproj(x, target, ..., source = NULL, four = FALSE)

## S3 method for class 'matrix'
reproj(x, target, ..., source = NULL, four = FALSE)

## S3 method for class 'data.frame'
reproj(x, target, ..., source = NULL, four = FALSE)
```

Arguments

x	coordinates
target	target specification (PROJ.4 string or epsg code)
...	arguments passed to proj4::ptransform()
source	source specification (PROJ.4 string or epsg code)
four	if TRUE, and PROJ version 6 is available return four columns xyzt (not just three xyz)

Details

If the modern version of the library 'proj' is available, reproj() uses the PROJ package, otherwise it falls pack to the proj4 package.

If using proj4, reproj drives the function proj4::ptransform() and sorts out the requirements for it so that we can simply give coordinates in data frame or matrix form, with a source projection and a target projection.

If using PROJ, reproj can pass in a wider variety of source and target strings, not just "proj4string" and we are completely subject to the new rules and behaviours of the PROJ library. We always assume "visualization order", i.e. longitude then latitude, easting then northing (as X, Y).

The basic function reproj() takes input in generic form (matrix or data frame) and returns a 3-column matrix (or 4-column if four = TRUE), by transforming from map projection specified by the source argument to that specified by the target argument. Only column order is respected, column names are ignored.

This model of working also allows adding methods for specific data formats that already carry a suitable source projection string. Currently we support types from the silicate and quadmesh and rgl packages, and only the target string need be specified.

This model has obvious flexibility, for packages to import the generic and call it with the correct source (from the data format) and the target from user, or process controlled mechanism.

The source argument must be named, and if it is not present a light check is made that the source data could be "longitude/latitude" and transformation to target is applied (this can be controlled by setting options).

The functionreproj() always returns a 3-column matrix *unless* four = TRUE, and PROJ::ok_proj6() is TRUE and then a 4-column matrix is returned.

Note that any integer input for source or target will be formatted to a character string like "EPSG:<integer_code>" ('PROJ' package in use) or "+init=epsg:<integer_code>" ('proj4' package in use), depending on the outcome of PROJ::ok_proj6(). This test function can be configured to alternatively use 'proj4' for expert use.

Until recently the proj4 package was the only one available for generic data that will transform between arbitrary coordinate systems specified by *source* and *target* coordinate systems and with control over 'xy' versus 'xyz' input and output. This package adds some further features by wrapping the need to convert longitude/latitude data to or from radians.

Other R packages for transforming coordinates are geared toward data that's in a particular format. It's true that only GDAL provides the full gamut of available geographic map projections, but this leaves a huge variety of workflows and applications that don't need that level of functionality.

Value

matrix

Dependencies

- The PROJ package is used preferentially if is functional, using the underlying 'PROJ-lib' at version 6 or higher. * The proj4 package is used if PROJ is not functional. The proj4 package works perfectly well with the PROJ-lib at versions 4, 5, 6, or 7 and if this is preferred reproj can be set to ignore the PROJ R package (see reproj-package).

Global options

Assuming longitude/latitude input:

The behaviour is controlled by user-settable options which on start up are reproj.assume.longlat = TRUE and reproj.default.longlat = "+proj=longlat +datum=WGS84 +no_defs".

If the option reproj.assume.longlat is set to FALSE then the source argument must be named explicitly, i.e. reproj(xy,t_srs,source = s_srs), this is to help catch mistakes being made. The target is the second argument in reproj though it is the third argument in proj4::ptransform. This function also converts to radians on input or output as required.

If the option reproj.assume.longlat is set to TRUE and the input data appear to be sensible longitude/latitude values, then the value of reproj.default.longlat is used as the assumed source projection.

Controlling use or PROJ or proj4:

See reproj-package for another option set reproj.mock.noproj6 for package testing for expert use.

Warning

There are a number of limitations to the PROJ library please use at your own risk. The sf package provides a better supported facility to modern code and especially for datum transformations.

Examples

```
reproj(cbind(147, -42), target = "+proj=laea +datum=WGS84",
                         source = "+proj=longlat +datum=WGS84")
```

Index