Package: reproj (via r-universe)

August 10, 2024

Type Package

Title Coordinate System Transformations for Generic Map Data

Version 0.7.0

Description Transform coordinates from a specified source to a specified target map projection. This uses the 'PROJ' library directly, by wrapping the 'PROJ' package which leverages 'libproj', 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

Depends R (>= 3.2.5)

Imports proj4, crsmeta (>= 0.3.0), PROJ (>= 0.4.5)

Suggests testthat, covr

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

SystemRequirements PROJ (>= 4.4.6)

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

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

Repository https://hypertidy.r-universe.dev

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

RemoteRef HEAD

RemoteSha 18d497cb4ac52659483cabd7a8370c193d630785

Contents

reproj.sc 2
reproj_extent 4

Index **6**

`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)

reproj_xy(x, target, ..., source = NULL)

reproj_xyz(x, target, ..., source = NULL)
```

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

We currently use the proj4 package.

The `reproj()` and related functions drive `proj4::ptransform()` and sort 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, 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 function `reproj()` always returns a 3-column matrix *unless* `four = TRUE`, and PROJ package is available then a 4-column matrix is returned.

Functions `reproj_xy()` and `reproj_xyz()` are helpers for `reproj()` and always return 2- or 3-column matrix respectively.

Note that any integer input for `source` or `target` will be formatted to a character string like "EPSG:<integer_code>" as a simple convenience. Note that there are other authorities besides EPSG, so the pattern "AUTH:code" is a general one and you should really be explicit.

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

numeric matrix of the transformed coordinates, either 2, 3, or 4 columns depending on the shape of the input, or the argument 'four' in `reproj()`. Use `reproj_xy()` or `reproj_xyz()` for those specific 2- and 3-column cases.

Dependencies

- The PROJ package is a stub atm and is not used.

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 = "OGC:CRS84"`.

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. The libproj package will be used if it makes it to CRAN.

Examples

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

---

reproj_extent *Reproject extent*

Description

A four figure extent (xmin, xmax, ymin, ymax) is used to approximate the boundary of its reprojected version by interpolating new vertices along each edge.

Usage

```
reproj_extent(extent, target, limit = NULL, ..., source = NULL)
```

Arguments

`extent`	a four element vector of extent `c(xmin, xmax, ymin, ymax)`
`target`	target specification (PROJ.4 string or epsg code)
`limit`	if used, a one or two element numeric vector to give the maximum radius to the edge of the extent from the middle
`...`	arguments passed to `proj4::ptransform()`
`source`	source specification (PROJ.4 string or epsg code)

Details

This is a simple version of what GDAL's 'SuggestedWarpOutput' does, and similar functions like the raster package 'projectExtent()'.

Internal functions unpack the various stages, and might be exposed in future. These stages are

1. interpolate around the boundary with correct ordering (can be used as a polygon or line)
2. reproject the interpolated boundary
3. summarize the interpolated boundary to the new extent

Value

four value extent `c(xmin, xmax, ymin, ymax)`

Examples

```
reproj_extent(c(0, 10, 0, 20), "+proj=laea", source = "+proj=longlat")
```

Index

proj4::ptransform(), *2–4*

reproj *(*reproj.sc*)*, 2
reproj(), *3*
reproj-package, *4*
reproj.sc, 2
reproj_extent, 4
reproj_xy *(*reproj.sc*)*, 2
reproj_xy(), *3*
reproj_xyz *(*reproj.sc*)*, 2
reproj_xyz(), *3*