

Package: PROJ (via r-universe)

October 15, 2026

Title Generic Coordinate System Transformations Using 'PROJ'

Version 0.7.0.9006

Description A wrapper around the generic coordinate transformation software 'PROJ' that transforms coordinates from one coordinate reference system ('CRS') to another. This includes cartographic projections as well as geodetic transformations. The intention is for this package to be used by user-packages such as 'reproj', and that the older 'PROJ.4' and version 5 pathways be provided by the 'proj4' package.

Depends R (>= 3.0.2)

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Imports lifecycle, wk (>= 0.9.2)

LinkingTo wk (>= 0.9.2)

Suggests testthat (>= 3.0.0), spelling, knitr, rmarkdown

URL <https://github.com/hypertidy/PROJ>,
<https://hypertidy.github.io/PROJ/>

BugReports <https://github.com/hypertidy/PROJ/issues>

Language en-US

VignetteBuilder knitr

SystemRequirements PROJ (>= 8.2.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libproj-dev

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

Date/Publication 2026-10-13 00:43:38 UTC

RemoteUrl <https://github.com/hypertidy/PROJ>

RemoteRef HEAD
RemoteSha c9b9392d0154c7f29830c5cf7b86156d3e9878ec

Contents

ok_proj6	2
proj_crs_text	3
proj_data_info	4
proj_factors	4
proj_trans	5
proj_trans_create	7
proj_version	8
xymap	9

Index	10
--------------	-----------

ok_proj6	<i>Is 'PROJ' library >= 6' available</i>
----------	---

Description

[Deprecated]
Test for availability of 'PROJ' system library version 6 or higher.

Usage

ok_proj6()

Details

On unix-alikes, this function is run in .onLoad() to check that version 6 functionality is available. On Windows, the load process sets the data file location with the version 6 API, and that is used as a test instead.

If 'PROJ' library version 6 is not available, the package still compiles and installs but is not functional.

The lack of function can be simulated by setting options(reproj.mock.noproj6 = TRUE), designed for use with the reproj package.

Value

logical, TRUE if the system library 'PROJ >= 6'

Examples

ok_proj6()

proj_crs_text	<i>Generate a projection string.</i>
---------------	--------------------------------------

Description

Input any accepted format of 'PROJ' coordinate reference system specification. Return value is a string in the requested format.

Usage

```
proj_crs_text(source, format = 0L)
```

Arguments

source	input projection specification one of ('PROJ4', 'WKT2', 'EPSG', 'PROJJSON', ... see the library documentation link in Details)
format	integer, 0 for 'WKT', 1 for 'PROJ', 2 for 'PROJJSON'

Details

See the [library documentation](#) for details on input and output formats.

Value

character string in requested format

warning

Note that a PROJ string is not a full specification, in particular this means that a string like "+proj=laea" cannot be converted to full WKT, because it is technically a transformation step not a crs. To get the full WKT form use a string like "+proj=laea +type=crs".

Examples

```
cat(proj_crs_text("EPSG:4326", format = 0L))
proj_crs_text("EPSG:4326", format = 1L)
south55 <- "+proj=utm +zone=55 +south +ellps=GRS80 +units=m +no_defs +type=crs"
proj_crs_text(proj_crs_text(south55), 1L)
```

proj_data_info	<i>Report on the PROJ data directory in use</i>
----------------	---

Description

Returns the path of the proj.db that the PROJ library resolves, its metadata (the PROJ version that built it, database layout version, EPSG version), whether the copy of PROJ data bundled with this package (if any) is active, and the PROJ library version.

Usage

```
proj_data_info()
```

Details

If path is NA the library cannot find a usable proj.db: EPSG codes and other database lookups will fail, though '+proj=' strings still work.

Value

a list with elements 'db' (named character), 'bundled_active' (logical), 'proj_version' (character), and 'message' (character, or NULL): the startup message explaining why a bundled copy was ignored, if that happened

Examples

```
proj_data_info()
```

proj_factors	<i>Compute map projection factors for coordinates</i>
--------------	---

Description

Computes map projection factors (distortion measures) for a set of geographic coordinates and a given projected CRS. Returns a matrix with one row per input coordinate and one column per factor.

Usage

```
proj_factors(lp, crs)
```

Arguments

lp	A two-column numeric matrix of geographic coordinates (longitude, latitude) in decimal degrees.
crs	A PROJ CRS definition string for the target projection (e.g. a PROJ string, WKT, or authority code such as "EPSG:3112"). Using a CRS (rather than a '+proj=' operation string) requires PROJ >= 8.2.

Details

Rows for coordinates where the factors computation fails (for example latitudes outside +/-90) are filled with NaN. Missing input coordinates give NA output.

Value

A numeric matrix with `nrow(lp)` rows and the following columns:

meridional_scale Meridional scale factor (h)
parallel_scale Parallel scale factor (k)
areal_scale Areal scale factor (s)
angular_distortion Angular distortion (omega), in radians
meridian_parallel_angle Meridian-parallel angle (theta prime), in radians
meridian_convergence Meridian convergence (conv), in radians
tissot_semimajor Tissot semimajor axis (a)
tissot_semiminor Tissot semiminor axis (b)
dx_dlam Partial derivative dx/dlambda
dx_dphi Partial derivative dx/dphi
dy_dlam Partial derivative dy/dlambda
dy_dphi Partial derivative dy/dphi

References

[PROJ PJ_FACTORS documentation](#)

Examples

```
# Lambert Azimuthal Equal Area
proj_factors(cbind(147, -42), "+proj=laea +lon_0=147 +lat_0=-42 +type=crs")

# Multiple coordinates
pts <- cbind(c(130, 147, 160), c(-20, -42, -35))
proj_factors(pts, "EPSG:3112")
```

proj_trans

Transform coordinates

Description

Transforms all coordinates in `x` using `wk::wk_handle()` and `proj_trans_create()`.

Usage

```
proj_trans(
  x,
  target_crs,
  source_crs = NULL,
  ...,
  use_z = NA,
  use_m = NA,
  options = character()
)
```

Arguments

<code>x</code>	Input geometry/geography. May take any of the following forms: • A coordinate matrix containing 2, 3 or 4 columns. If named, expects column names "x", "y" and optionally "z" and/or "m". If not named, columns are assumed in xyzm order. Non-coordinate columns are removed. • A data.frame containing coordinates as columns. Expects names "x", "y" and optionally "z" and/or "m". Non-coordinate columns are retained. • A data.frame containing a geometry vector which is readable by <code>wk::wk_handle()</code>, including sfc columns. • A geometry vector which is readable by <code>wk::wk_handle()</code>, including sfc columns.
<code>source_crs, target_crs</code>	Source/Target CRS definition, coerced with <code>wk::wk_crs_proj_definition()</code>
<code>...</code>	Additional parameters forwarded to <code>wk::wk_handle()</code>
<code>use_z, use_m</code>	Used to declare the output type. Use TRUE to ensure the output has that dimension, FALSE to ensure it does not, and NA to leave the dimension unchanged.
<code>options</code>	character vector of "KEY=VALUE" options passed to the PROJ library when choosing the coordinate operation, see Details

Details

Values that are detected out of bounds by library PROJ are allowed, we return Inf in this case, rather than the error "tolerance condition error".

Value

Transformed geometries whose format is dependent on input.

References

see the [PROJ library documentation](#) for details on the underlying functionality

Examples

```
proj_trans(cbind(147, -42), "+proj=laea +type=crs", "EPSG:4326")
proj_trans(cbind(147, -42, -2), "+proj=laea +type=crs", "EPSG:4326")
proj_trans(cbind(147, -42, -2, 1), "+proj=laea +type=crs", "EPSG:4326")
proj_trans(wk::xy(147, -42, crs = "EPSG:4326"), "+proj=laea +type=crs")
proj_trans(wk::wkt("POLYGON ((1 1, 0 1, 0 0, 1 0, 1 1))", crs = "EPSG:4326"), 3112)
```

proj_trans_create	Create a transformation object
-------------------	--------------------------------

Description

Creates a transformation object that transforms coordinates in a wk pipeline.

Usage

```
proj_trans_create(
  source_crs,
  target_crs,
  use_z = NA,
  use_m = NA,
  options = character()
)
```

Arguments

source_crs, target_crs	Source/Target CRS definition, coerced with wk::wk_crs_proj_definition()
use_z, use_m	Used to declare the output type. Use TRUE to ensure the output has that dimension, FALSE to ensure it does not, and NA to leave the dimension unchanged.
options	character vector of "KEY=VALUE" options passed to the PROJ library when choosing the coordinate operation, see Details

Details

options are passed to the PROJ library function `proj_create_crs_to_crs_from_pj()`. The options PROJ accepts are

- `AUTHORITY=name` restrict candidate operations to one authority (e.g. "EPSG"), or "any"
- `ACCURACY=value` minimum desired accuracy in metres
- `ALLOW_BALLPARK=YES/NO` set to NO to disallow ballpark transformations
- `FORCE_OVER=YES/NO` (PROJ >= 9.1.1) set to YES to force the +over flag, so longitudes are not wrapped to -180..180. This is what you want when transforming data that straddles the antimeridian, see the example

- ONLY_BEST=YES/NO (PROJ >= 9.2) set to YES to error rather than fall back to a worse operation when the best one needs a grid that is not available

An option the linked PROJ library predates is dropped with a warning. Any other unrecognized option is an error from PROJ.

Value

A PROJ transformation object

Examples

```
(trans <- proj_trans_create("EPSG:4326", "EPSG:3857"))
wk::wk_transform(wk::xy(1:5, 1:5), trans)

## longitude wrapping at the antimeridian: just past the western edge of
## Web Mercator comes back as +180 by default, -180 with FORCE_OVER
xmin <- -20037508.3427892
inv <- proj_trans_create("EPSG:3857", "EPSG:4326")
wk::wk_transform(wk::xy(xmin - 0.001, 0), inv)
if (proj_version() >= "9.1.1") {
  over <- proj_trans_create("EPSG:3857", "EPSG:4326", options = "FORCE_OVER=YES")
  wk::wk_transform(wk::xy(xmin - 0.001, 0), over)
}

library(wk)
(invtrans <- wk_trans_inverse(trans))

h <- 1852 * 60
## the stretch of Mercator to a square
wk::wk_transform(wk::xy(c(-h * 180, 0, h * 180), c(-h * 180, 0, h * 180)), invtrans)
```

proj_version

Report PROJ library version

Description

This function returns NA if PROJ lib is not available.

Usage

```
proj_version()
```

Value

character string (major.minor.patch)

Examples

```
proj_version()
```

xymap	<i>xymap data for testing</i>
-------	-------------------------------

Description

A copy of the xymap data set from the quadmesh package.

Details

A matrix of longitude/latitude values of the world coastline.

Index

ok_proj6, [2](#)

proj_crs_text, [3](#)

proj_data_info, [4](#)

proj_factors, [4](#)

proj_trans, [5](#)

proj_trans_create, [7](#)

proj_trans_create(), [5](#)

proj_version, [8](#)

wk::wk_crs_proj_definition(), [6](#), [7](#)

wk::wk_handle(), [5](#), [6](#)

xymap, [9](#)