

Package: projr (via r-universe)

October 15, 2026

Title Comprehensive Interface to the PROJ Library

Version 0.0.1

Description Direct access to the PROJ coordinate transformation library ($\geq 9.6.0$), including coordinate transformation, CRS inspection, decomposition, identification, operation discovery, projection factors, and database queries. Uses cpp11 for bindings to the PROJ C API.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements PROJ ($\geq 9.6.0$)

LinkingTo cpp11

Suggests testthat ($\geq 3.0.0$)

Config/testthat/edition 3

URL <https://github.com/hypertidy/projr>

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

Config/pak/sysreqs libproj-dev

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

Date/Publication 2026-02-25 11:45:56 UTC

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

RemoteRef HEAD

RemoteSha a5465d01810310d5dada416ff290474c8e534af7

Contents

proj_as_proj_string	2
proj_as_projjson	3
proj_as_wkt	3

proj_crs 4

proj_crs_get_geodetic_crs 4

proj_db_path 5

proj_factors 5

proj_get_area_of_use 6

proj_get_authorities 6

proj_get_codes 7

proj_get_ellipsoid 7

proj_get_id 8

proj_get_name 8

proj_get_type 9

proj_identify 9

proj_is_crs 10

proj_trans 10

proj_trans_bounds 11

proj_trans_xy 11

proj_trans_xyz 12

proj_trans_xyzt 13

proj_transform 13

proj_version 14

Index 15

proj_as_proj_string	<i>Export CRS as PROJ string</i>
---------------------	----------------------------------

Description

Export CRS as PROJ string

Usage

proj_as_proj_string(x)

Arguments

x a projr_crs object

Value

character string

proj_as_projjson	<i>Export CRS as PROJJSON</i>
------------------	-------------------------------

Description

Export CRS as PROJJSON

Usage

```
proj_as_projjson(x)
```

Arguments

x	a projr_crs object
---	--------------------

Value

character string (JSON)

proj_as_wkt	<i>Export CRS as WKT</i>
-------------	--------------------------

Description

Export CRS as WKT

Usage

```
proj_as_wkt(  
  x,  
  type = c("WKT2_2019", "WKT2_2019_SIMPLIFIED", "WKT2_2015", "WKT2_2015_SIMPLIFIED",  
           "WKT1_GDAL", "WKT1_ESRI")  
)
```

Arguments

x	a projr_crs object
type	WKT variant

Value

character string

Examples

```
proj_as_wkt(proj_crs("EPSG:4326"))  
proj_as_wkt(proj_crs("EPSG:4326"), type = "WKT1_GDAL")
```

proj_crs	Create a CRS object
----------	---------------------

Description

Create from any supported representation: EPSG code, PROJ string, WKT, PROJJSON, or database name (e.g. "WGS 84").

Usage

```
proj_crs(x)
```

Arguments

x	CRS definition string
---	-----------------------

Value

An external pointer to a PROJ PJ object, with class "projr_crs"

Examples

```
proj_crs("EPSG:4326")
proj_crs("EPSG:32755")
proj_crs("+proj=laea +lon_0=147 +lat_0=-42 +datum=WGS84 +type=crs")
```

proj_crs_get_geodetic_crs	Get the geodetic CRS underlying a projected or compound CRS
---------------------------	---

Description

Get the geodetic CRS underlying a projected or compound CRS

Usage

```
proj_crs_get_geodetic_crs(x)
```

Arguments

x	a projr_crs object
---	--------------------

Value

a projr_crs object

Examples

```
proj_crs_get_geodetic_crs(proj_crs("EPSG:32755"))
```

proj_db_path	<i>Path to the PROJ database</i>
--------------	----------------------------------

Description

Path to the PROJ database

Usage

```
proj_db_path()
```

Value

character string

proj_factors	<i>Cartographic projection distortion factors</i>
--------------	---

Description

Compute scale, angular distortion, Tissot indicatrix parameters and partial derivatives at given geographic coordinates for a projected CRS.

Usage

```
proj_factors(lon, lat, crs)
```

Arguments

lon, lat	numeric vectors of geographic coordinates (degrees)
crs	projected CRS definition string

Value

data.frame with distortion metrics

Examples

```
proj_factors(seq(140, 155, by = 5), rep(-42, 4),  
             "+proj=utm +zone=55 +datum=WGS84")
```

`proj_get_area_of_use` *Get the area of use of a CRS*

Description

Get the area of use of a CRS

Usage

```
proj_get_area_of_use(x)
```

Arguments

`x` a `projr_crs` object

Value

list with west, south, east, north, name

Examples

```
proj_get_area_of_use(proj_crs("EPSG:32755"))
```

`proj_get_authorities` *List database authorities*

Description

List database authorities

Usage

```
proj_get_authorities()
```

Value

character vector

proj_get_codes	<i>List codes for an authority and type</i>
----------------	---

Description

List codes for an authority and type

Usage

```
proj_get_codes(auth, type = "CRS")
```

Arguments

auth	authority name (e.g. "EPSG")
type	object type (e.g. "ProjectedCRS", "GeographicCRS", "Ellipsoid")

Value

character vector

Examples

```
head(proj_get_codes("EPSG", "Ellipsoid"))
```

proj_get_ellipsoid	<i>Get ellipsoid parameters</i>
--------------------	---------------------------------

Description

Get ellipsoid parameters

Usage

```
proj_get_ellipsoid(x)
```

Arguments

x	a projr_crs object
---	--------------------

Value

list with name, semi_major, semi_minor, inv_flattening, is_semi_minor_computed

Examples

```
proj_get_ellipsoid(proj_crs("EPSG:4326"))  
proj_get_ellipsoid(proj_crs("EPSG:4283"))
```

proj_get_id	<i>Get the authority and code of a PROJ object</i>
-------------	--

Description

Get the authority and code of a PROJ object

Usage

```
proj_get_id(x, index = 0L)
```

Arguments

x	a projr_crs object
index	zero-based identifier index (usually 0)

Value

list with authority and code

proj_get_name	<i>Get the name of a PROJ object</i>
---------------	--------------------------------------

Description

Get the name of a PROJ object

Usage

```
proj_get_name(x)
```

Arguments

x	a projr_crs object
---	--------------------

Value

character string

proj_get_type	<i>Get the type of a PROJ object</i>
---------------	--------------------------------------

Description

Get the type of a PROJ object

Usage

```
proj_get_type(x)
```

Arguments

x	a projr_crs object
---	--------------------

Value

character string

proj_identify	<i>Identify a CRS from its definition</i>
---------------	---

Description

Given a CRS (especially one parsed from WKT), find matching entries in the PROJ database with confidence scores.

Usage

```
proj_identify(x, auth_name = NULL)
```

Arguments

x	a projr_crs object
auth_name	optional authority to restrict search (e.g. "EPSG")

Value

data.frame with authority, code, confidence

Examples

```
crs <- proj_crs("+proj=longlat +datum=WGS84 +type=crs")
proj_identify(crs)
```

proj_is_crs	<i>Test if a PROJ object is a CRS</i>
-------------	---------------------------------------

Description

Test if a PROJ object is a CRS

Usage

proj_is_crs(x)

Arguments

x a projr_crs object

Value

logical

proj_trans	<i>Transform coordinates</i>
------------	------------------------------

Description

Transforms coordinates between CRS. Supply just x/y for 2D, add z for 3D, add t for 4D. Output list shape mirrors input — you get back only the components you put in.

Usage

proj_trans(x, y, z = NULL, t = NULL, source, target)

Arguments

x, y numeric vectors (required)
z numeric vector or NULL
t numeric vector or NULL
source, target CRS definition strings

Value

Named list of numeric vectors (x, y, and optionally z, t)

Examples

```
# 2D
proj_trans(147:148, c(-42, -45), source = "OGC:CRS84", target = "EPSG:32755")

# 3D - just add z
proj_trans(147, -42, z = 100, source = "OGC:CRS84", target = "EPSG:4978")
```

proj_trans_bounds	<i>Transform a bounding box</i>
-------------------	---------------------------------

Description

Handles densification, antimeridian crossings, and polar regions correctly.

Usage

```
proj_trans_bounds(xmin, ymin, xmax, ymax, source, target, densify_pts = 21L)
```

Arguments

xmin, ymin, xmax, ymax	extent in source CRS
source, target	CRS definition strings
densify_pts	number of points per edge for densification

Value

Named numeric vector: xmin, ymin, xmax, ymax

Examples

```
proj_trans_bounds(140, -50, 155, -35, "OGC:CRS84", "EPSG:32755")
```

proj_trans_xy	<i>Transform xy coordinates</i>
---------------	---------------------------------

Description

Transform xy coordinates

Usage

```
proj_trans_xy(x, y, source, target)
```

Arguments

- x, y numeric vectors of coordinates (degrees for geographic CRS)
- source, target CRS definition strings (EPSG codes, WKT, PROJ strings)

Value

Named list with x and y numeric vectors

Examples

```
proj_trans_xy(147:148, c(-42, -45), "OGC:CRS84", "EPSG:32755")
```

proj_trans_xyz	<i>Transform xyz coordinates</i>
----------------	----------------------------------

Description

Transform xyz coordinates

Usage

```
proj_trans_xyz(x, y, z, source, target)
```

Arguments

- x, y numeric vectors of coordinates (degrees for geographic CRS)
- z numeric vector of elevation/vertical values
- source, target CRS definition strings (EPSG codes, WKT, PROJ strings)

Value

Named list with x, y, and z numeric vectors

Examples

```
proj_trans_xyz(147, -42, 100, "OGC:CRS84", "EPSG:4978")
```

proj_trans_xyzt	<i>Transform xyzt coordinates</i>
-----------------	-----------------------------------

Description

Transform xyzt coordinates

Usage

```
proj_trans_xyzt(x, y, z, t, source, target)
```

Arguments

x, y	numeric vectors of coordinates (degrees for geographic CRS)
z	numeric vector of elevation/vertical values
t	numeric vector of time values (decimal year for dynamic CRS)
source, target	CRS definition strings (EPSG codes, WKT, PROJ strings)

Value

Named list with x, y, z, and t numeric vectors

proj_transform	<i>Transform coordinates with explicit direction</i>
----------------	--

Description

Transform coordinates with explicit direction

Usage

```
proj_transform(
  x,
  y,
  z = NULL,
  t = NULL,
  source,
  target,
  direction = c("forward", "inverse", "identity")
)
```

Arguments

- x, y numeric vectors (required)
- z numeric vector or NULL
- t numeric vector or NULL
- source, target CRS definition strings
- direction One of "forward" (source to target), "inverse" (target to source), or "identity".

Value

Named list of numeric vectors

Examples

```
# Forward
pt <- proj_transform(147, -42, source = "OGC:CRS84", target = "EPSG:32755")
# Inverse - swap conceptually without swapping source/target
proj_transform(pt$x, pt$y, source = "OGC:CRS84", target = "EPSG:32755",
               direction = "inverse")
```

proj_version	<i>PROJ version info</i>
--------------	--------------------------

Description

PROJ version info

Usage

```
proj_version()
```

Value

list with version, searchpath, paths

Index

[proj_as_proj_string, 2](#)
[proj_as_projjson, 3](#)
[proj_as_wkt, 3](#)
[proj_crs, 4](#)
[proj_crs_get_geodetic_crs, 4](#)
[proj_db_path, 5](#)
[proj_factors, 5](#)
[proj_get_area_of_use, 6](#)
[proj_get_authorities, 6](#)
[proj_get_codes, 7](#)
[proj_get_ellipsoid, 7](#)
[proj_get_id, 8](#)
[proj_get_name, 8](#)
[proj_get_type, 9](#)
[proj_identify, 9](#)
[proj_is_crs, 10](#)
[proj_trans, 10](#)
[proj_trans_bounds, 11](#)
[proj_trans_xy, 11](#)
[proj_trans_xyz, 12](#)
[proj_trans_xyzt, 13](#)
[proj_transform, 13](#)
[proj_version, 14](#)