An introduction to PROJ

EUREF Tutorial: Transformations using PROJ
May 21, 2019, Tallinn

Kristian Evers - kreve@sdfe.dk - @kbevers

http://bit.do/EUREFTutorial

Agenda

Follow along at
http://bit.do/EUREFTutorial

- PROJ?
- A quick primer on geodetic standardization and coordinate transformation
- The PROJ transformation language
- The PROJ applications
- The PROJ community

PROJ?

What is PROJ?

PROJ is

- a generic coordinate transformation tool
- a shared database of parameters and definitions of geodetic objects
- a real world implementation of the ISO19111 standard on “Referencing by coordinates”
- a code library that can be re-used in third party applications
- a “language” for describing geodetic transformations

Or simply put, PROJ is the *Swiss army knife of geodesy*

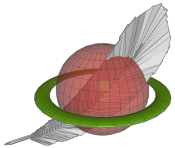
Why use PROJ?

- It is free (as in beer *and* free speech)
- It is open source (MIT license)
- It is available almost everywhere
- It is used in a a very large subset of geospatial software
- It implements almost all of the coordinate reference systems and transformations registered in the EPSG database
- It is probably the most complete coordinate transformation software in the world

How to use PROJ?

- Through 3rd party software like QGIS or GDAL
- Build it into your own software
- Use the command line tools that comes as part of the PROJ package
  - **`proj`**
  - **`cs2cs`**
  - **`cct`**
  - **`geod`**
  - **`projinfo`**
  - **`gie`**

You probably already have used PROJ

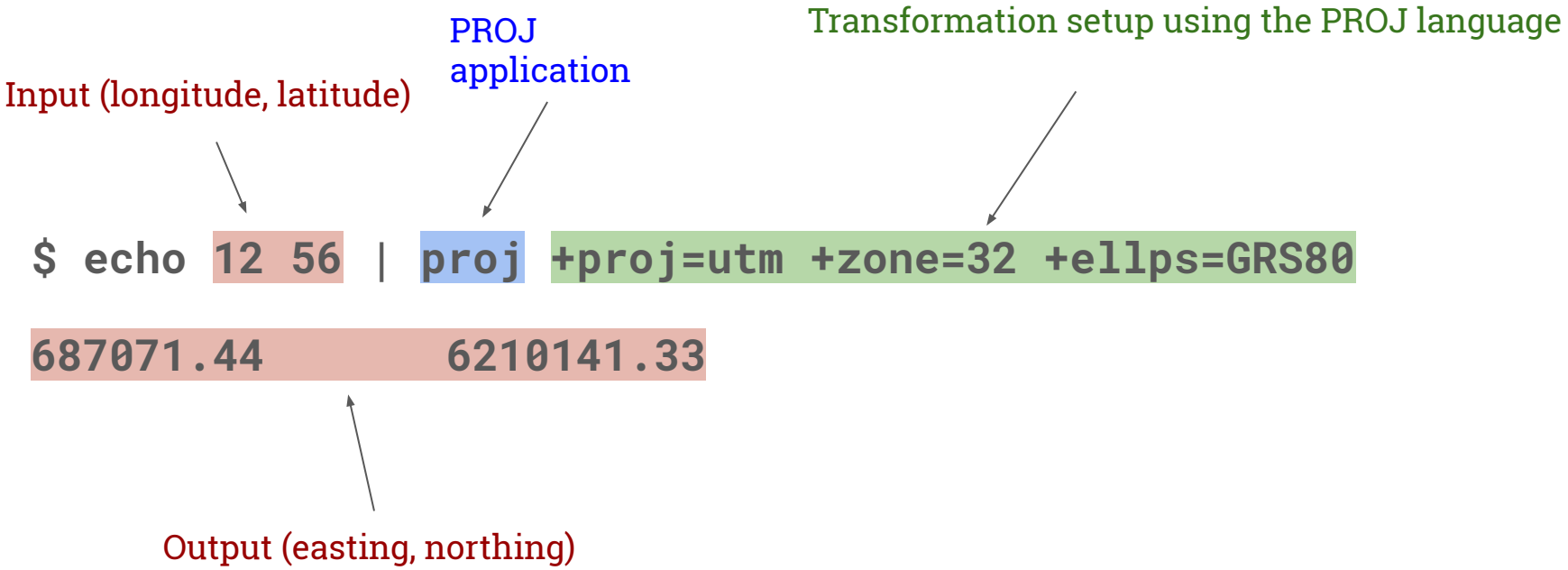


A simple example

```
$ echo 12 56 | proj +proj=utm +zone=32 +ellps=GRS80

687071.44       6210141.33
```

A simple example

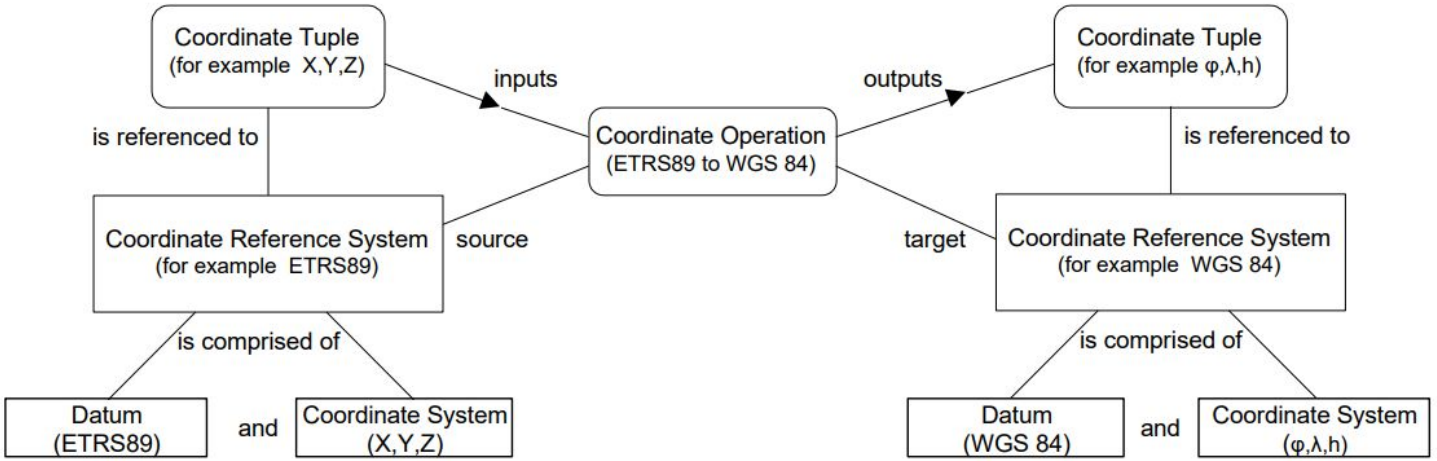


A quick primer on geodetic standardization and coordinate transformation

4 THEORETICAL CONCEPTS

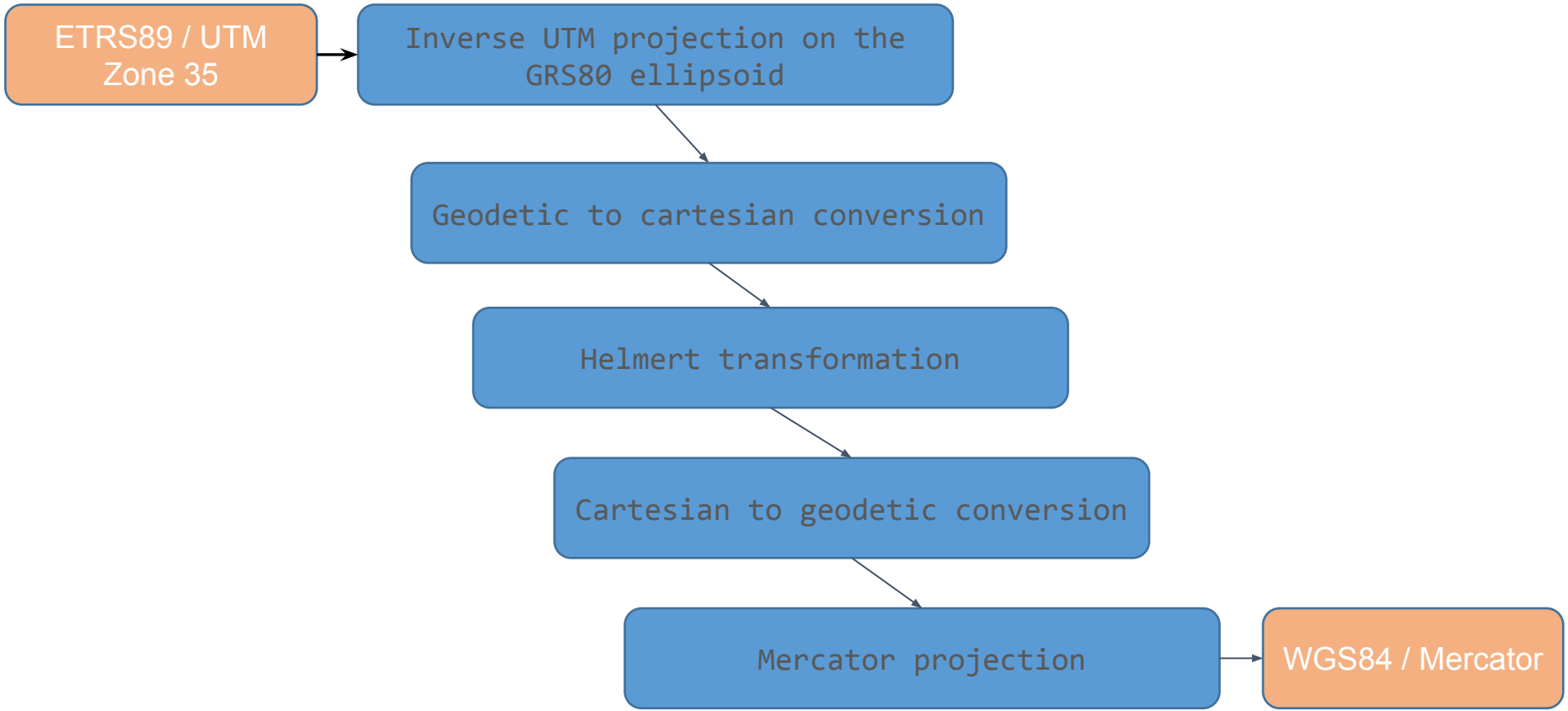
4.1 Coordinates

The high level abstract model for spatial referencing by coordinates is shown in the diagram below:



A coordinate is one of a sequence of values describing a position. The sequence is sometimes called a coordinate tuple. Coordinates are referenced to a *coordinate reference system* (CRS). A coordinate reference system is a *coordinate system* (CS) – an abstract mathematical concept without any relationship to a physical object – that is referenced through a *datum* to the Earth or some other object such as a vessel. A *coordinate operation* may be used to change coordinate values which are referenced to one CRS to being referenced to a second CRS.

ETRS89/UTM Zone 35 -> WGS84/Mercator



Standardization

Geospatial standards are developed internationally in a loose collaboration between the (mostly industry based) OGC - The Open Geospatial Consortium and the (mostly governmental/academic) ISO Technical Committee 211.

OGC works mostly on the directly industry related aspects (formats, protocols...), while ISO/TC211 is more focused on the foundational aspects.

Many standards specifications are cross-published under both labels, which is useful for the occasional user, since the OGC editions are cost free and openly available, while the ISO counterparts are sold by the national member organisations of the ISO.

The geodetically most relevant volumes of the ISO19100 series

6709 - Representation of latitude, longitude and altitude
19111 - Spatial referencing by coordinates
19116 - Positioning services
19127 - Geodetic codes and parameters
19156 – Observations & Measurements
19161 - Geodetic references – Part 1: ITRS
19162 - WKT representation of coordinate reference systems

Also of importance for international geospatial standardization: IOGP, the international organization of oil and gas producers, maintaining the EPSG registry, the de facto standard for representing coordinate reference systems and related transformations.

ISO/TC 211

OGC®
Making location count.

IOGP
International Association of Oil & Gas Producers

The PROJ transformation language

Basics

In PROJ, a conversion or transformation is defined by a “**proj-string**”.

proj-strings can be said to make up a “**geodetic language**” for describing transformations.

The language is comprised of a string of **parameter/value pairs** that control the desired coordinate operation, e.g:

```
+proj=operation +key_1=value_1 +key_2=value_2 … +key_n=value_n
```

Keys can exist without a value, in which case they can be seen as flags turning on some property of the operation.

The **+proj** parameter is the mandatory operation parameter.

Basics

Every ***+proj*** is an operation that takes a coordinate tuple as input and outputs a coordinate tuple.

In general a coordinate tuples consist of 4 numbers.

Usually the ordering of the coordinate tuple is (x, y, z, t), although this can be changed.

Operations only change the coordinate components that it affects:

```
(12, 55, 34.7, 2017.75) -> proj=merc lon_0=12 -> (0, 7326837.72, 34.7, 2017.75)
```

Projections

aea
aeqd
airy
aitoff
alsk
apian
august
bacon
bipc
boggs
bonne
calcofi
cass
cc
cea
chamb
collg
comill
crast
denoy
eck1
eck2
eck3
eck4
eck5
eck6
eqc
eqdc
euler
etmerc
fahey
fouc
fouc_s
gall
geoc
geos
gins8
gn_sinu
gnom
goode
gs48
gs50
hammer
hatano
healpix
rhealpix
igh
imw_p
isea
kav5
kav7
krovak
labrd
laea
lagrng
larr
lask
lcc
lcca
leac
lee_os
loxim
lsat
mbt_s
mbt_fps
mbtfpp
mbtfpq
mbtfps
merc
mil_os
mill
misrsom
moll
murd1
murd2
murd3
natearth
natearth2
nell
nell_h
nicol
nsper
nzmg
ob_tran
ocea
oea
omerc
ortel
ortho
pconic
patterson
poly
putp1
putp2
putp3
putp3p
putp4p
putp5
putp5p
putp6
putp6p
qua_aut
qsc
robin
rouss
rpoly
sch
sinu
somerc
stere
sterea
gstmerc
tcc
tcea
times
tissot
tmerc
tpeqd
tpers
ups
urm5
urmfps
utm
vandg
vandg2
vandg3
vandg4
vitk1
wag1
wag2
wag3
wag4
wag5
wag6
Wag7
webmerc
weren
wink1
wink2
wintri

Projections

All projections take geodetic coordinates as input and returns projected coordinates in units of meters.

Examples of projections described with the PROJ syntax:

```
+proj=merc

+proj=utm +zone=32

+proj=lcc +lat_1=20 +lat_2=40

+proj=lcc +lat_1=40 +lat_0=40 +k_0=0.9988085293 +x_0=600000 +y_0=600000
          +a=6378298.3 +b=6356657.142669561 +pm=madrid
```

Generic parameters

False easting:
    +x_0

False northing:
    +y_0

Latitude of origin:
    +lat_0

Longitude of origin:
    +lon_0

Scaling factor:
    +k_0

Prime meridian:
    +pm

Earth size:
    +R, +a

Earth shape:
    +ellps, +rf, +f, +es, +e, +b

Ellipsoids

A complete ellipsoid definition is comprised of a size (primary) and a shape (secondary) parameter.

Size parameters supported are:

- +R, defining the radius of a spherical planet
- +a, defining the semimajor axis of an ellipsoidal planet

Shape parameters supported are:

- +rf, the reverse flattening of the ellipsoid
- +f, the flattening of the ellipsoid
- +es, the eccentricity squared
- +e, the eccentricity
- +b, the semiminor axis

Ellipsoids

The ellipsoid can also be set simply by using the parameter **+ellps**.

A selection of valid options for **+ellps** are:

```
    GRS80 a=6378137.0      rf=298.257222101 GRS 1980(IUGG, 1980)
     airy a=6377563.396    b=6356256.910    Airy 1830
   bessel a=6377397.155    rf=299.1528128   Bessel 1841
   clrk66 a=6378206.4      b=6356583.8      Clarke 1866
     intl a=6378388.0      rf=297.          International 1909 (Hayford)
    WGS60 a=6378165.0      rf=298.3         WGS 60
    WGS66 a=6378145.0      rf=298.25        WGS 66
    WGS72 a=6378135.0      rf=298.26        WGS 72
    WGS84 a=6378137.0      rf=298.257223563 WGS 84
   sphere a=6370997.0      b=6370997.0      Normal Sphere (r=6370997)
```

Run **proj -le** for the full list

Ellipsoids

Spherical earth with radius 7000km:

```
+proj=merc +R=7000000
```

Using the GRS80 ellipsoid:

```
+proj=merc +ellps=GRS80
```

Expressing ellipsoid by semi-major axis and reverse flattening (1/f):

```
+proj=merc +a=6378137.0 +rf=298.25
```

Transformations

affine
deformation
geogoffset
helmert
molodensky
molobadekas
horner
hgridshift
vgridshift

Helmert

7 parameters shift:

```
+proj=helmert +x=0.67678    +y=0.65495   +z=-0.52827
              +rx=-0.022742 +ry=0.012667 +rz=0.022704 +s=-0.01070
```

14 parameter shift, ITRF2008 -> ITRF88:

```
+proj=helmert +x=0.0127     +y=0.0065     +z=-0.0209  +s=0.00195
              +dx=-0.0029   +dy=-0.0002   +dz=-0.0006 +ds=0.00001
              +rx=-0.00039  +ry=0.00080   +rz=-0.00114
              +drx=-0.00011 +dry=-0.00019 +drz=0.00007
              +epoch=1988.0 +convention=position_vector
```

Gridshifting

Datum shift by means of grids.

Horizontal gridshift using several grids:

```
+proj=hgridshift +grids=@conus,@alaska,@ntv2_0.gsb,@ntv_can.dat
```

Vertical gridshift using optional grid:

```
+proj=vgridshift +grids=@dvr90.gtx,egm96_16.gtx
```

Pipelines

With transformation pipelines it is possible to combine everything we've seen so far into one transformation.

When constructing a transformation pipeline several coordinate operation are chained together in a series of steps that in combination make out the desired transformation.

Usually pipelines combine datum shifts and projections.

Individual steps in a pipeline can be reversed by using the parameter ***+inv***.

Pipelines

Recipe of a pipeline:

```
+proj=pipeline
+step operation_1 [+inv]
+step operation_2 [+inv]
+step operation_3 [+inv]
       ...
+step operation_n [+inv]
```

Where **operation_x** is a proj-string defining a specific operation

Pipelines

Datum shift with a Helmert transform. In- and output in geodetic coordinates:

```
+proj=pipeline
+step +proj=cart +ellps=intl
+step +proj=helmert +t_epoch=2000 +convention=position_vector
      +x=0.0019 +y=0.0017 +z=0.0105
      +rx=0 +ry=0 +rz=0 +s=-0.00134
      +dx=-0.0001 +dy=-0.0001 +dz=0.0018
      +drx=0 +dry=0 +drz=0 +ds=-8e-05
 +step +inv +proj=cart +ellps=GRS80
```

Pipelines

Transformation from Danish System34 to UTM Zone 32 / ETRS89. In- and output in projected coordinates:

```
+proj=pipeline
+step +init=./s45b.pol:s45b_tc32
+step +inv +proj=utm +ellps=intl +zone=32
+step +proj=cart +ellps=intl
+step +proj=helmert
      +x=-81.0703  +y=-89.3603  +z=-115.7526
     +rx=-0.48488 +ry=-0.02436 +rz=-0.41321 s=-0.540645
+step +inv +proj=cart +ellps=GRS80
+step +proj=utm +ellps=GRS80 +zone=33
```

Pipelines principles

1. Pipelines must consist of at least one step

+proj=pipeline

Will fail to create.

Pipelines principles

2. Pipelines without a forward path can’t be constructed

+proj=pipeline +step +inv +proj=urm5

Will fail to create since the **urm5** projection doesn’t have an inversion projection.

1. Parameters added before the first **+step** are global and will be applied to all steps
2. Units of operations must match between steps

Pipelines principles

3. Parameters added before the first **+step** are global and will be applied to all steps

```
+proj=pipeline +ellps=GRS80
+step +proj=merc +inv
+step +proj=lcc
```

is equivalent to

```
+proj=pipeline
+step +proj=merc +ellps=GRS80 +inv
+step +proj=lcc  +ellps=GRS80
```

Pipelines principles

4. Units of operations must match between steps

```
+proj=pipeline
+step +proj=merc
+step +proj=robin
```

The above will fail because **merc** outputs projected coordinates and **robin** expects longitude and latitude as input.

PROJ cares about ***projected coordinates***, ***angular coordinates*** and ***geocentric coordinates*** - not differences between km and m, that's your responsibility!

Conversions

Geodetic to cartesian (geocentric) conversion:

```
+proj=cart +ellps=GRS80
```

Axisswap. Change the order and direction of axes:

```
+proj=axisswap +order=1,2,3,-4
```

Unitconvert. Convert between distance and time units:

```
+proj=unitconvert +t_in=gps_week +t_out=decimalyear
                  +xy_in=m +xy_out=dm
                  +z_in=cm +z_out=mm
```

The PROJ applications

proj

proj converts geodetic coordinates into projected coordinates.

```
$ echo 12 56 | proj +proj=tmerc +lat_0=55 +ellps=intl
746490.28 176509.00
```

Input is in the form (longitude, latitude) and output is on the form (easting, northing).

proj *only* works with projections, i.e. no transformations.

See more on https://proj4.org/apps/proj.html

cs2cs

cs2cs transforms between a source and destination CRS:

```
$ echo 56 12 123 | cs2cs EPSG:4256 EPSG:25832
687071.44    6210141.33 123.00

$ echo 56 12 123 |cs2cs ETRS89 "ETRS89 / UTM Zone 32N"
687071.44    6210141.33 123.00
```

cs2cs will try to find the best transformation for you depending on you input coordinates - be careful, you may not always get what you want!

The format of the input and output is determined by the source and destination CRS.

See more on https://proj4.org/apps/cs2cs.html

projinfo

projinfo returns info on transformations and coordinate reference systems:

$ projinfo ETRS89
See next slide

$ projinfo “ETRS89 / UTM Zone 32N”
See next slide

Queries the geodetic database that PROJ is build on

See more on https://proj4.org/apps/projinfo.html

```
$ projinfo ETRS89

WKT2_2018 string:
GEOGCRS["ETRS89",
    DATUM["European Terrestrial Reference System 1989",
        ELLIPSOID["GRS 1980",6378137,298.257222101,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    USAGE[
        SCOPE["unknown"],
        AREA["Europe - ETRS89"],
        BBOX[32.88,-16.1,84.17,40.18]],
    ID["EPSG",4258]]
```

```
$ projinfo "ETRS89 / UTM Zone 32N"

WKT2_2018 string:
PROJCRS["ETRS89 / UTM zone 32N",
    BASEGEOGCRS["ETRS89",
        DATUM["European Terrestrial Reference System 1989",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]]],
    CONVERSION["UTM zone 32N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",9,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["unknown"],
        AREA["Europe - 6°E to 12°E and ETRS89 by country"],
        BBOX[38.76,6,83.92,12]],
    ID["EPSG",25832]]
```

cct

cct is the tool for the super user - it gives you complete control over your transformation. As well as the ability to shoot yourself in the foot.

```
$ echo 3496723 743251 5264442 2019 | cct +proj=helmert
+x=100 +dy=20 +drz=0.4 +t_epoch=2000
+convention=position_vector
 3496795.6143    743759.8397  5264442.0000     2019.0000
```

Note that **cct** *always* expects a 4D coordinate.

See more on https://proj4.org/apps/cct.html

geod

geod performs (inverse) geodesic computations

```
$ echo 56 12 59 24 |geod -I +ellps=GRS80 +units=km
60d6'20.275"  -109d45'34.25"   791.775

$ echo 59 24 -109d45\'34.25\" 791.775 | geod +ellps=GRS80
+units=km
56d0'0.005"N 12d0'0.015"E 60d6'20.287"
```

See more on https://proj4.org/apps/geod.html

gie

gie is The Geospatial Integrity Investigation Environment for PROJ

```
$ gie test/gie/GDA.gie
----------------------------------------------------------------
Reading file 'test/gie/GDA.gie'
----------------------------------------------------------------
total:  3 tests succeeded,  0 tests skipped,  0 tests failed.
----------------------------------------------------------------
```

See more on https://proj4.org/apps/gie.html

```
<gie>
...
------------------------------------------------------------------------
ITRF2014@2018 to GDA2020  -  Test point ALIC (Alice Springs)
------------------------------------------------------------------------
Just the Helmert transformation, to verify that we are within 100 um
------------------------------------------------------------------------
operation proj = helmert    exact convention=coordinate_frame

         x = 0   rx = 0   dx = 0   drx = 0.00150379
         y = 0   ry = 0   dy = 0   dry = 0.00118346
         z = 0   rz = 0   dz = 0   drz = 0.00120716
         s = 0   ds = 0   t_epoch = 2020.0
------------------------------------------------------------------------
tolerance  40 um
accept     -4052052.6588 4212835.9938 -2545104.6946 2018.0 #ITRF2014@2018
expect     -4052052.7373 4212835.9835 -2545104.5867        #GDA2020
------------------------------------------------------------------------

</gie>
```

The PROJ community

PROJ is something we make together

Everyone can - and arguably should - contribute

- Help PROJ-users that are less experienced than yourself
- Write bug reports
- Request new features
- Write documentation for your favorite map projection
- Add tests
- Fix bugs
- Implement new features

Here's how: https://proj4.org/community/contributing.html

… but who really did it?

```
$ git shortlog --summary --numbered
   903  Kristian Evers
   897  Frank Warmerdam
   811  Even Rouault
   201  Thomas Knudsen
   140  Howard Butler
    74  Charles Karney
    56  Mike Toews
    44  Kurt Schwehr
    40  Elliott Sales de Andrade
    35  Bas Couwenberg
    27  Martin Desruisseaux
    22  Mateusz Loskot
    21  Didier Richard
    14  Micah Cochran
    12  julien2512
    11  Kai Pastor
    11  Aaron Puchert
    10  Bojan Savric
     8  Chris Mayo
     8  Andrey Kiselev
     6  Javier Goizueta
     5  Ivan Veselov
     6  Javier Goizueta
     5  Ivan Veselov
     5  Jeff Whitaker
     5  Philippe Rivière
     5  mbull
     4  Andrea Antonello
     4  Mateusz Łoskot
     4  Alan D. Snow
     3  Nyall Dawson
     3  Adam Wulkiewicz
     3  Ben Boeckel
     3  drhaynes
     2  Karoline Skaar
     2  Bojan Šavrič
     2  Etienne Jacques
     2  Piyush Agram
     2  Marco Bernasocchi
     2  Ture Pålsson
     2  Vedran Stojnović
     2  Yann Chemin
     2  Mike Adair
     2  Andrew Bell
```

PROJ

PROJ is a generic coordinate transformation software that transforms geospatial coordinates from one coordinate reference system (CRS) to another. This includes cartographic projections as well as geodetic transformations.

PROJ includes command line applications for easy conversion of coordinates from text files or directly from user input. In addition to the command line utilities PROJ also exposes an application programming interface, or API in short. The API lets developers use the functionality of PROJ in their own software without having to implement similar functionality themselves.

PROJ started purely as a cartography application letting users convert geodetic coordinates into projected coordinates using a number of different cartographic projections. Over the years, as the need has become apparent, support for datum shifts has slowly worked its way into PROJ as well. Today PROJ supports more than a hundred different map projections and can transform coordinates between datums using all but the most obscure geodetic techniques.

You can download the source code for PROJ on the download section and find links to prepackaged executables in the installation section.

In addition to this website the PROJ documentation is also available in PDF form.

Next

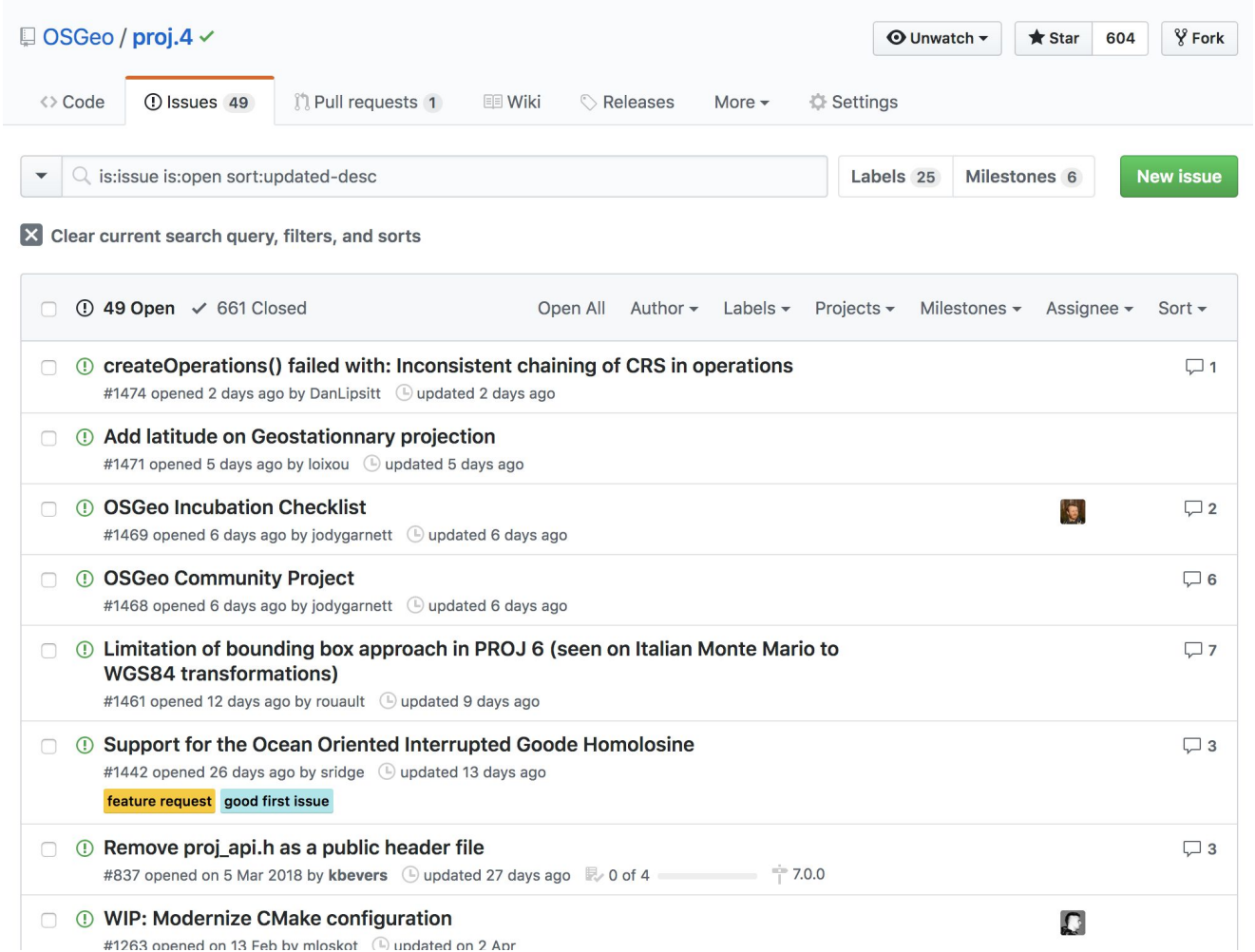
https://proj4.org/

May 2019 Archives by thread

- **Messages sorted by:** [subject] [author] [date]
- **More info on this list...**

Starting: *Wed May 1 04:26:42 PDT 2019*
Ending: *Mon May 20 00:45:27 PDT 2019*
Messages: 92

- [PROJ] question about EPSG:3003 *a.furieri at lqt.it*
  - [PROJ] question about EPSG:3003 *Even Rouault*
- [PROJ] PROJ >=5.0.0 on Travis CI *Michael Sumner*
  - [PROJ] PROJ >=5.0.0 on Travis CI *Sebastiaan Couwenberg*
    - [PROJ] PROJ >=5.0.0 on Travis CI *Kurt Schwehr*
      - [PROJ] PROJ >=5.0.0 on Travis CI *Michael Sumner*
  - [PROJ] PROJ >=5.0.0 on Travis CI *Even Rouault*
    - [PROJ] PROJ >=5.0.0 on Travis CI *Michael Sumner*
- [PROJ] Getting "proj" and "datum" for a CRS *Nyall Dawson*
  - [PROJ] Getting "proj" and "datum" for a CRS *Kristian Evers*
    - [PROJ] Getting "proj" and "datum" for a CRS *Even Rouault*
      - [PROJ] Getting "proj" and "datum" for a CRS *Nyall Dawson*
      - [PROJ] Getting "proj" and "datum" for a CRS *Even Rouault*
      - [PROJ] Getting "proj" and "datum" for a CRS *Kristian Evers*
      - [PROJ] Getting "proj" and "datum" for a CRS *Martin Desruisseaux*
  - [PROJ] Getting "proj" and "datum" for a CRS *Even Rouault*
- [PROJ] Transforming between ITRF realizations *Trey Stafford*
  - [PROJ] Transforming between ITRF realizations *Alan Snow*
    - [PROJ] Transforming between ITRF realizations *Even Rouault*
      - [PROJ] Transforming between ITRF realizations *Trey Stafford*
      - [PROJ] Transforming between ITRF realizations *Alan Snow*
      - [PROJ] Transforming between ITRF realizations *Markus Metz*
- [PROJ] PROJ 6.1.0RC1 *Kristian Evers*
  - [PROJ] PROJ 6.1.0RC1 *Bas Couwenberg*
    - [PROJ] PROJ 6.1.0RC1 *Kristian Evers*
      - [PROJ] PROJ 6.1.0RC1 *Bas Couwenberg*
      - [PROJ] PROJ 6.1.0RC1 *Even Rouault*
      - [PROJ] PROJ 6.1.0RC1 *Bas Couwenberg*
      - [PROJ] PROJ 6.1.0RC1 *Even Rouault*
      - [PROJ] PROJ 6.1.0RC1 *Bas Couwenberg*
      - [PROJ] PROJ 6.1.0RC1 *Even Rouault*
      - [PROJ] PROJ 6.1.0RC1 *Bas Couwenberg*



https://github.com/OSGeo/proj.4/

Q & A