Some loose thoughts on simplifying transformation parameter derivation

Kristian Evers, NKG Science Week, WG Reference Frames, 2024-03-13

Agency for Data Supply
and Infrastructure

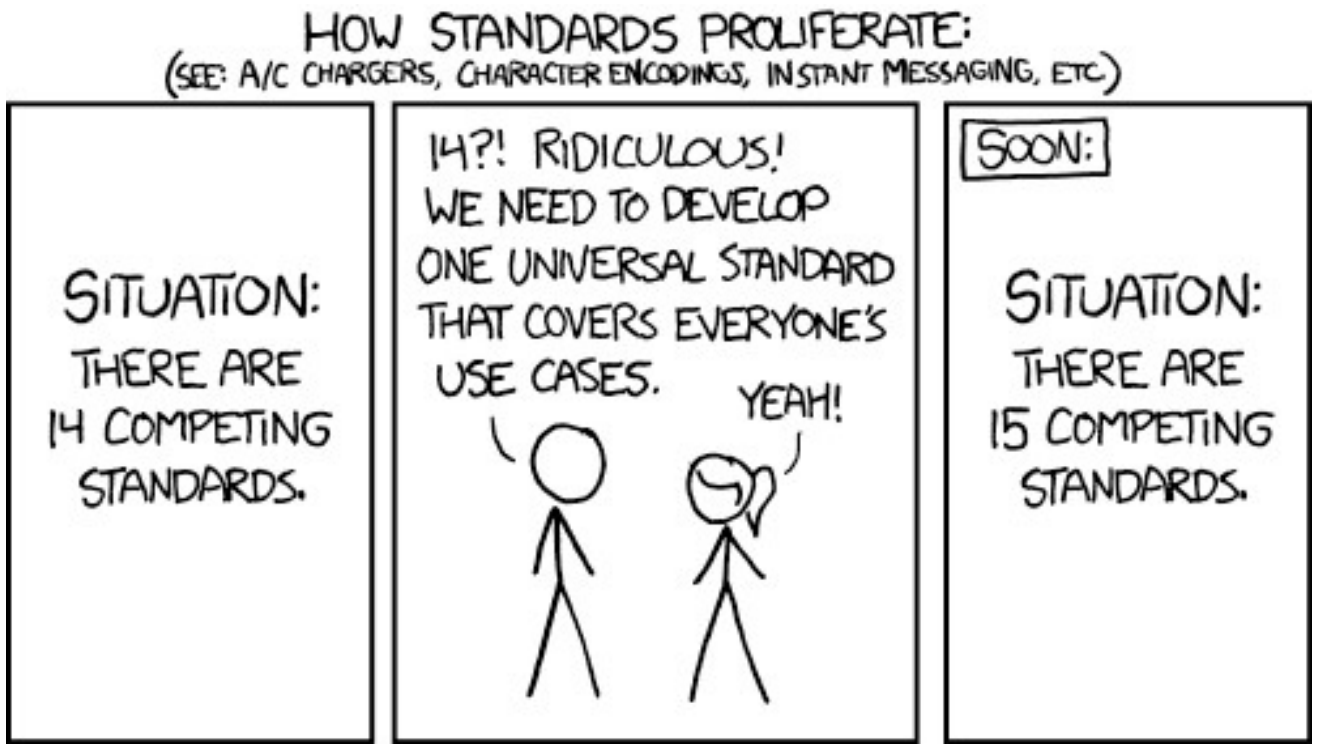
The Problems

- NKG Transformation code not a shared resource
- Research papers are generally not backed by publicly available code
- Many single-purpose implementations of various transformation derivation techniques
- Limited availability of standard software for deriving coordinate transformations

The Platonic Ideal of transformation derivation

A framework that

- handles the boring and tedious task involved in deriving transformation parameters
- lets researchers focus on the science, not software engineering
- offers users with limited programming skill the ability to derive complex coordinate transformations
- consumes data from a range of different data sources
- delivers transformation parameters in well-known formats
- provides detailed statistics and reports
- is modular and easily extendable


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Use case: Derive 7 parameter Helmert

- Data source
  - Geodetic database with coordinates in both IGS14 and ETRS89
- Output
  - Parameters as a PROJ-string

Use case: Adjust geoid to local height system

- Data sources
  - Gravimetric geoid model in Gravsoft format
  - Physical heights and ellipsoid heights at control points in plain text format
- Output
  - Adjusted geoid model in GTG format
  - PROJ-string using the adjusted geoid model

Use case: NKG Transformations

- Data sources
  - Bernese solutions from all NKG countries
    - Coordinates and velocities
  - Defining ETRS89 station coordinates
  - Deformation model in *unknown format*
- Output
  - Transformation parameters in WKT2-format
  - Deformation model in GTG format

Warning: This might be a two-step process

Introducing Transformo

Transformo is a generalized software package for estimating geodetic transformation parameters and models.

The problem Transformo tries to solve can be presented on the idealized form

$$\mathbf{T = M(p, S)}$$

Where **S** and **T** are the source and target coordinates, **M** is a transformation model and **p** is the parameters for the chosen model.

Introducing Transformo continued

The real world is rarely simple so a more realistic form would be

$$T = M(p, S) + R$$

Where **R** is the transformation residuals. The primary function of Transformo is to provide a set of parameters **p** that minimizes the residual **R**.

Introducing Transformo continued

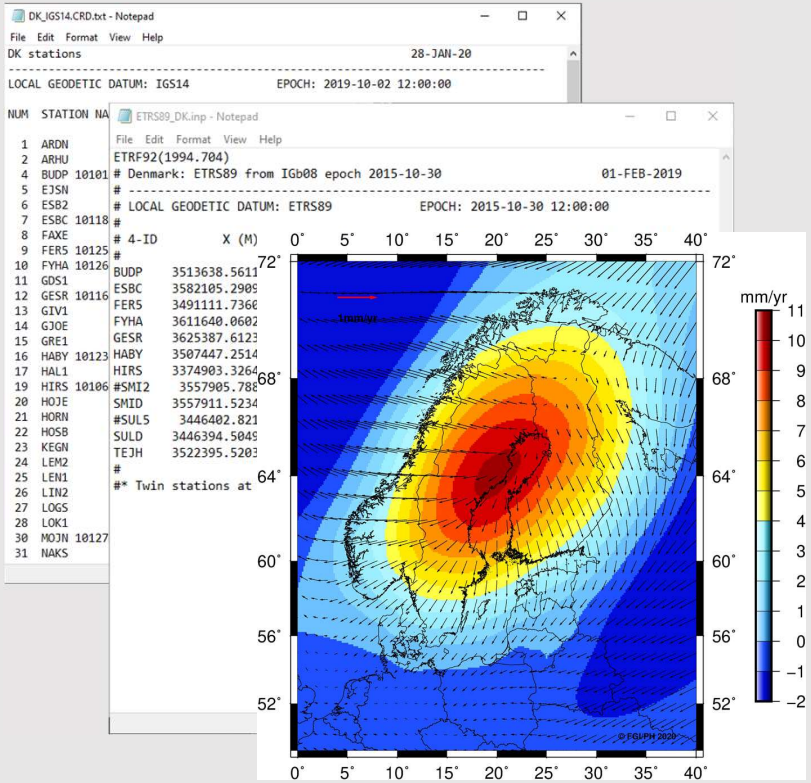
In some cases that might involve a transformation model consisting of several steps:

$$T = M_3(p_3, M_2(p_2, M_1(p_1, S))) + R$$

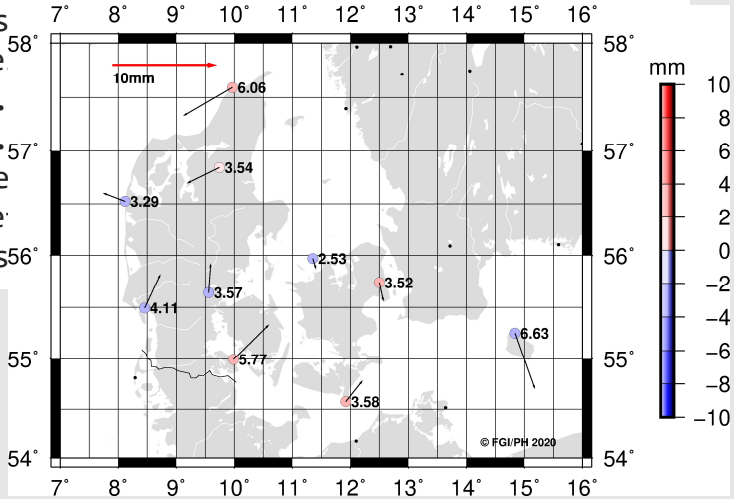
creating a pipeline of connected transformation models and parameters

In other words …

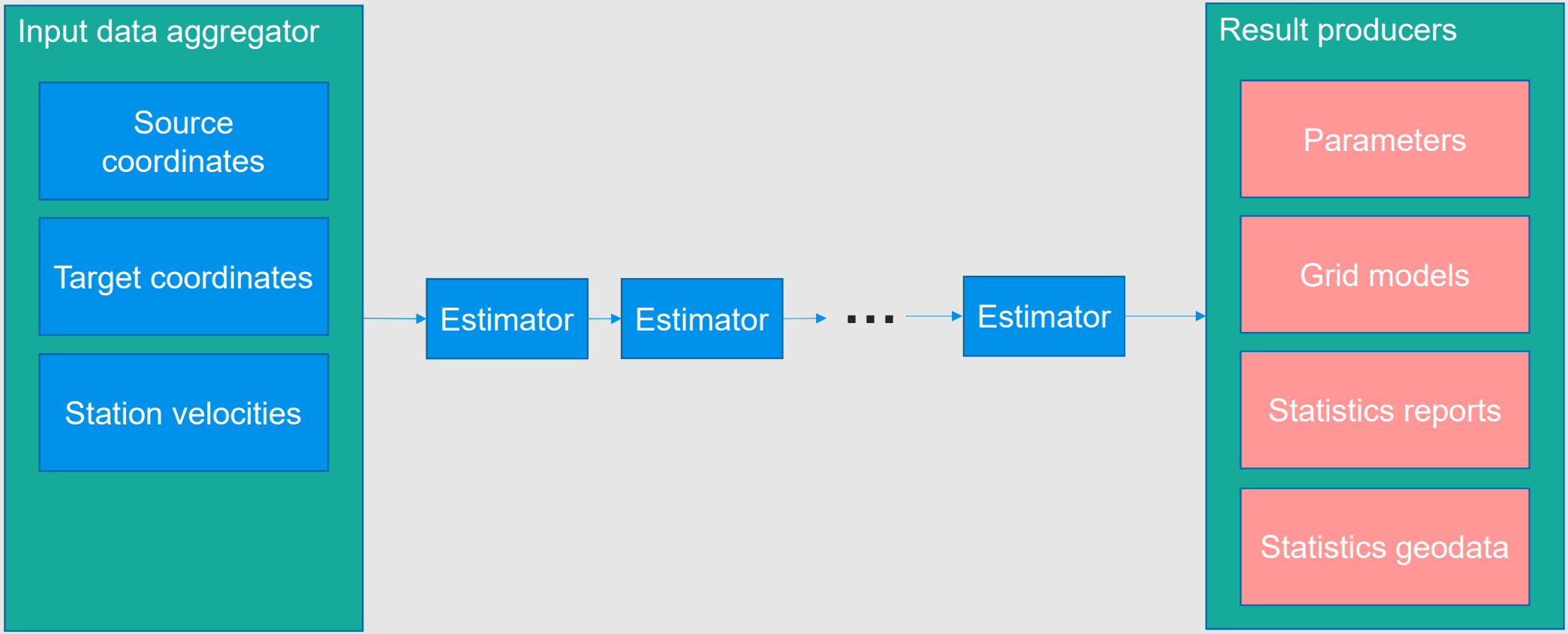
… an application that, given the necessary input data and a transformation model produces parameters, grids, statistics, etc




```
+proj=pipeline
  +step +proj=helmert +x=0 +y=0 +z=0
        +rx=0 +ry=0 +rz=0 +dx=0 +dy=0 +dz=0
        +drx=8.5e-05 +dry=0.000531 +drz=-0.00077
          +s=0 +ds=0 +t_epoch=1989
        +convention=position_vector
  +step +inv +proj=deformation +t_epoch=2000.0
          +grids
  +step +proj=he
          +z=-0.
          +rz=0.
          +conve
  +step +proj=de
          +grids
```

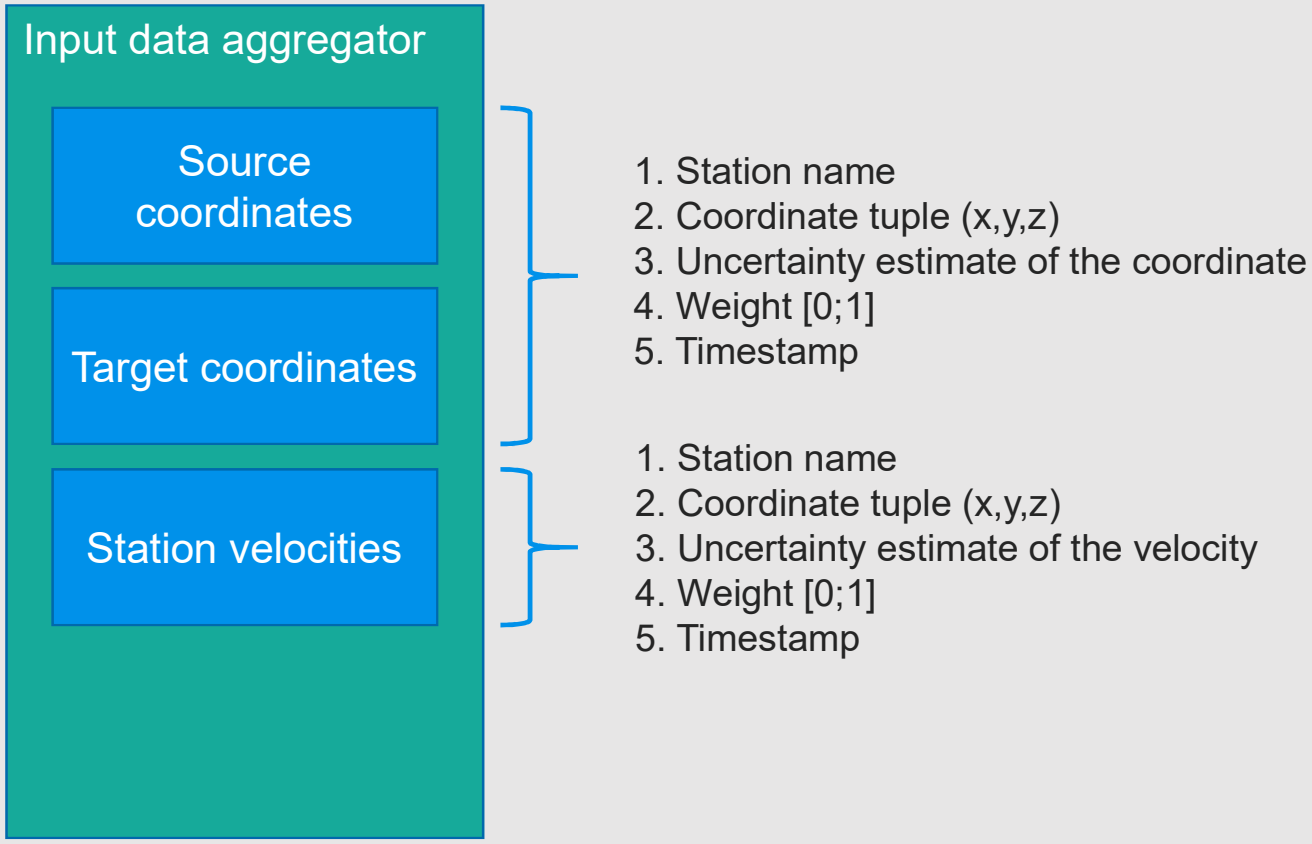


Architecture



Data readers

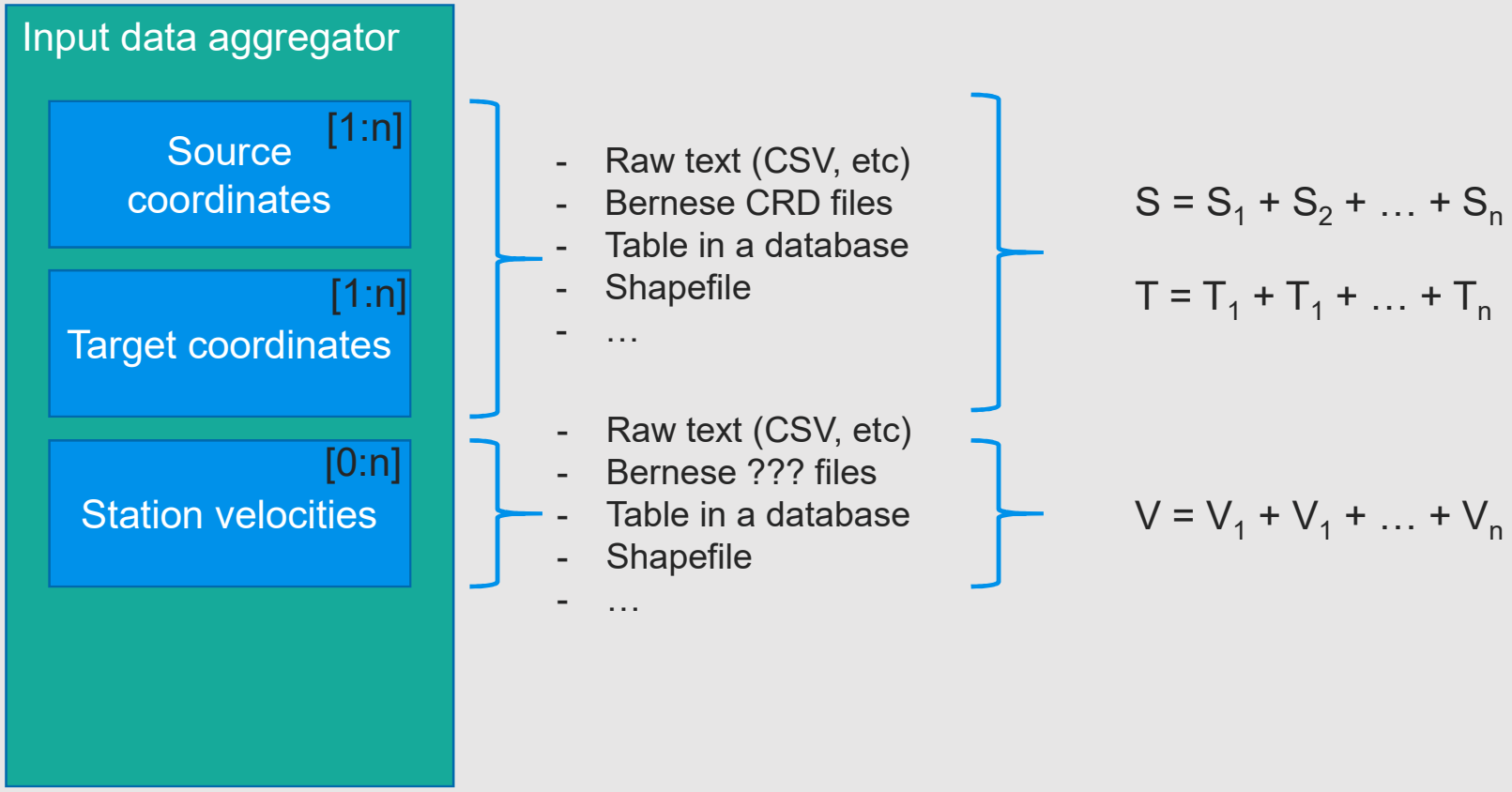
$$T = M_3(p_3, M_2(p_2, M_1(p_1, S))) + R$$



1. Station name
2. Coordinate tuple (x,y,z)
3. Uncertainty estimate of the coordinate
4. Weight [0;1]
5. Timestamp

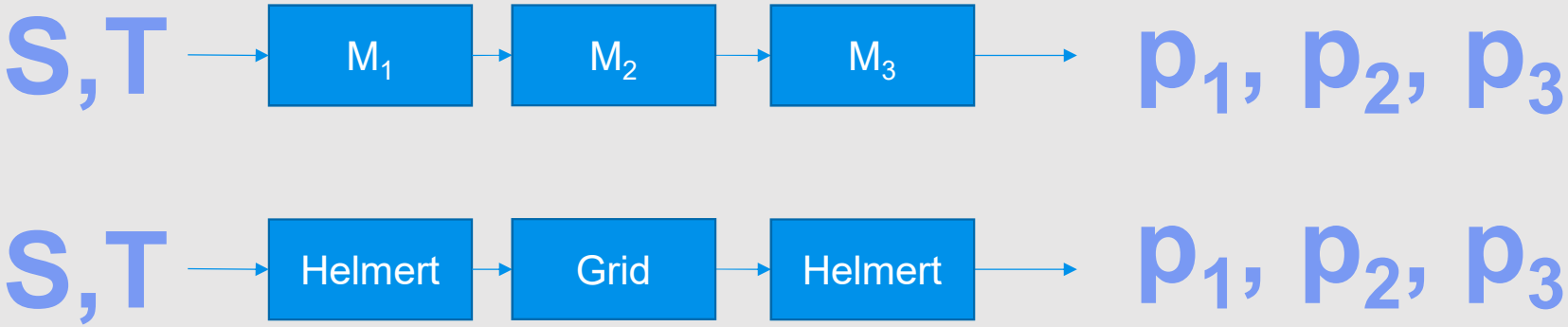
1. Station name
2. Coordinate tuple (x,y,z)
3. Uncertainty estimate of the velocity
4. Weight [0;1]
5. Timestamp

Data readers



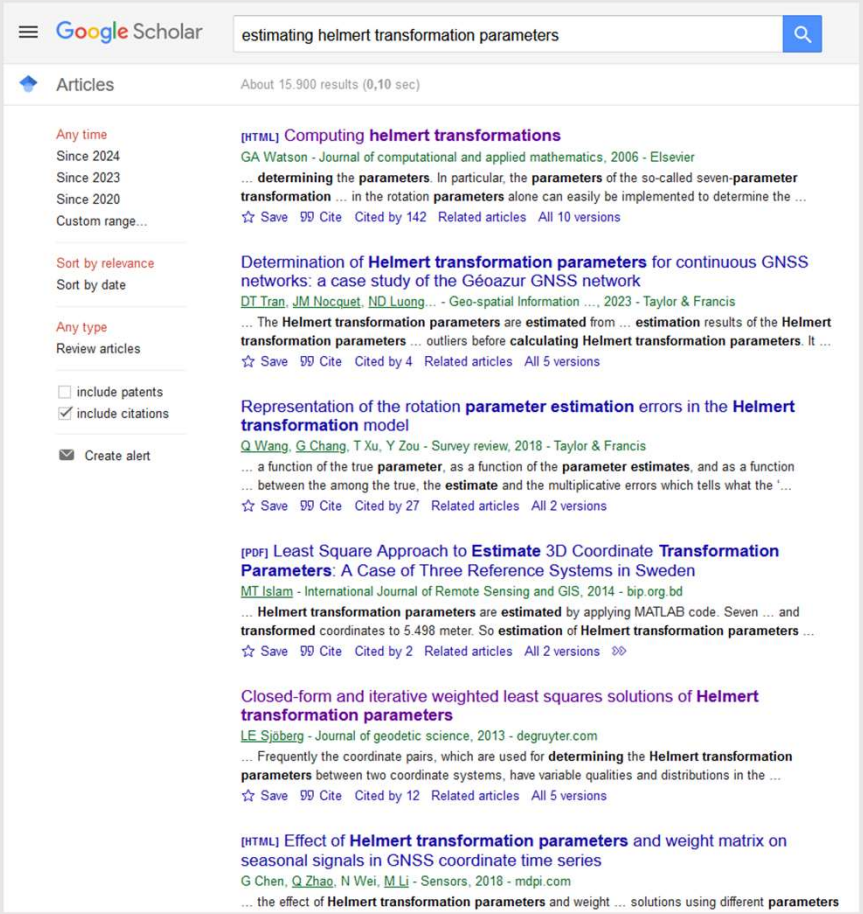
Parameter estimators

$$T = \mathbf{M_3}(p_3, \mathbf{M_2}(p_2, \mathbf{M_1}(p_1, S))) + R$$



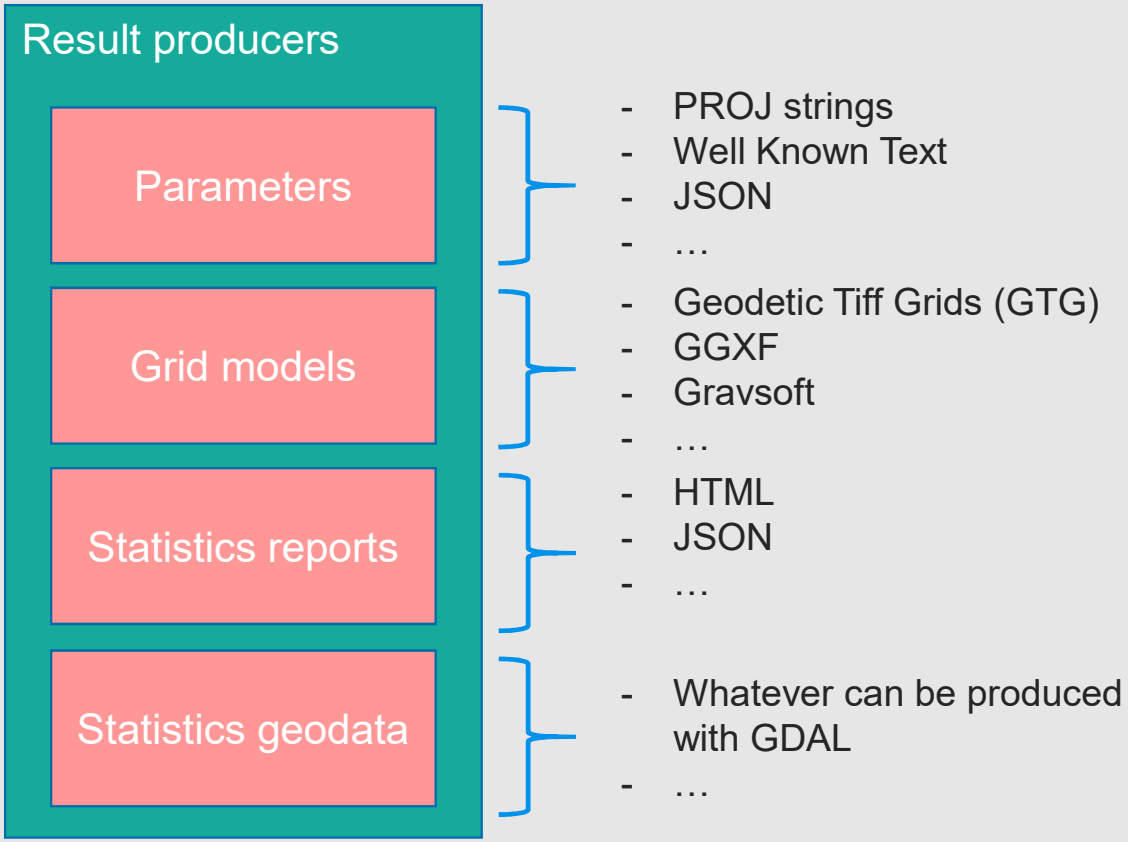
Parameter estimators

- Many estimators could be implemented
  - Helmert in various forms
  - Affine transformation
  - Grid corrections
  - Polynomial transformation
  - TIN transformation
  - Fancy quantum AI adjustor
  - …
- Even competing algorithms for estimating the same type of transformation parameters



Results

$$T = M_3(p_3, M_2(p_2, M_1(p_1, S))) + R$$



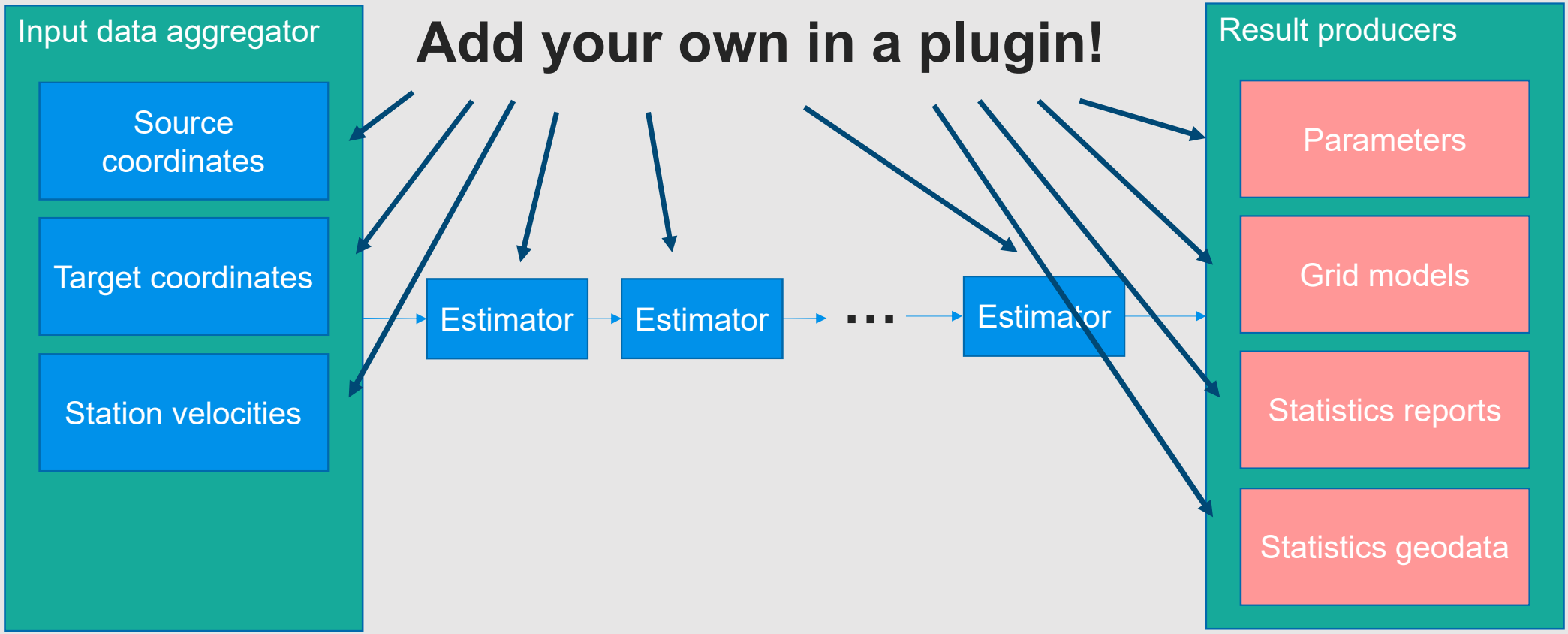
- PROJ strings
- Well Known Text
- JSON
- …

- Geodetic Tiff Grids (GTG)
- GGXF
- Gravsoft
- …

- HTML
- JSON
- …

- Whatever can be produced with GDAL
- …

Extendable architecture



Does it solve The Problems?

Not yes, but it can!

- NKG Transformation code not a shared resource

  The next generation could be build using a framework like this

- Research papers are generally not backed by publicly available code

  Implement just the necessary algorithms, publish as a plugin and submit the code to the Journal of Open Source Software (https://joss.theoj.org/)

- Many single-purpose implementations of various transformation derivation techniques

  With enough readers, estimators and result generators this can be dealt with

- Limited availability of standard software for deriving coordinate transformations

  This would provide an extra option – hopefully better than the rest!


Questions?