CSAR — X-Mapping Model for ^{210}Pb Dating

Python Implementation — Release 1
Created by J.M. Abril-Hernández (2025)

Departamento de Física Aplicada I, E.T.S.I. Agronómica, University of Seville (Spain)
ORCID: https://orcid.org/0000-0003-2540-5576 email: jmabril@us.es

---

1. Bibliographic support and related software

The reader can find a review on the ^{210}Pb-based dating models for recent sediments in Abril-Hernández, 2025; https://doi.org/10.1016/j.jenvrad.2025.107749 (Open access).

The CSAR model was first presented in the paper:

J.M. Abril. ^{210}Pb-based dating of recent sediments with the χ-mapping version of the Constant Sediment Accumulation Rate (CSAR) model. J. Environ. Radioact. 268–269 (2023), Article 107247, 10.1016/j.jenvrad.2023.107247.

CSAR belongs to the family of χ-mapping models that follows the method first used by the TERESA model for the ^{210}Pb-based dating of recent sediments. A Python implementation of an upgraded version of this model is available at:

Abril, Jose M., A Python Implementation of the upgraded χ-Mapping TERESA Model for 210Pb Dating of Recent Sediments: Software and Case Studies. Available at SSRN: https://ssrn.com/abstract=5897121 or http://dx.doi.org/10.2139/ssrn.5897121

The software packages are available in Github (initial release and release 2)

https://github.com/jmabril1964/TERESApy

https://github.com/jmabril1964/TERESApy-Release2

They are also available in Zenodo (https://doi.org/10.5281/zenodo.17289007 and https://doi.org/10.5281/zenodo.17954891)

It is recommended to get first familiar with TERESA software package; then the use of this software for CSAR will be quite straightforward, since it follows the same logic than release 2 for TERESA.

📁 Required Files in the Working Folder

- **Core_C1.txt**
  Empirical data with the ^{210}Pb profile.
- **/aleat_S1>**
  Folder containing the library of random samples. It can be generated once using:
- **Random_generator_S1.py**
  Script for generating the library of random samples.

- **Configuration.json**
  Input file containing all parameters required to run the TERESA codes.
- **CSAR_map.py**
  Generates the 4D map for X_df.
- **CSAR_cronos.py**
  Defines the confidence region, generates clouds of solutions for plotting, and produces the final result files.

By default, the output files
`Map3D.txt`, `Absolute_minimum.txt`, `Cloud.txt`, `Plot.txt`, and `Solution.txt`
are created in the same folder.
Paths can be customized in the `.json` file and in the code.

For graphical outputs it is recommended using Gnuplot, although the files with the numerical outputs can be used with MATLAB, Python, or other software for graphics.

⚙ Preparing the input data

Core_C1.txt is a 3-column text file containing the basic empirical data of the core consisting of: i) the mass depth of the slice, referred to its bottom, in units of g cm^{-2}; ii) the $^{210}Pb_{exc}$ mass activity concentration (in units of Bq kg^{-1}, and typically found as the total ^{210}Pb minus the ^{226}Ra mass activity concentrations) and iii) the associated uncertainty (also in Bq kg^{-1}).

Note that CSAR does not need the complete recovery of the $^{210}Pb_{exc}$ inventory, so the last measured slice may contain $^{10}Pb_{exc}$ values well above zero.

CSAR needs a continuous record, so if some slices were not measured, you need estimate the missing values, typically through interpolations. This pre-treatment of the data is not included in the software, and the user can do it with Excel or by other means. The starting point for CSAR is this text file, with a 3-coloumn format separated by space or tabulation. This simple format has the advantage of being directly read and plotted by graphical software such as Gnuplot.

Note that in ^{210}Pb-dating models we use two mass depth scales, one referring to the midpoint of each sediment sliced (e.g., used for plotting $^{210}Pb_{exc}$ versus mass depth profiles and applying the CF-CS model) and another referred to the bottom of the slice (e.g., used to estimate the ages with the CRS model). The software CSAR interprets the mass depth scale referred to the bottom of the slice, as above commented, and from it, the code generates the second mass depth scale referred to the midpoint of the slice, for using the appropriate one in each step of the calculations.

Random_generator_S1.py must be run for the first time. It generates a library with the canonical representative samples of normal-typified distributions (mean = 0, sigma = 1) of size ranging from 5 up to 99 (or higher, if needed), labelled as z_0. It is copied in lists z_1 and then randomly sorted. The result is stored in a 1-column text files named "aleat_S1_N.txt"

The output files will be created in the same folder where the code is actually placed. It is suggested to create then a sub-folder named "aleat_S1" and store all the aleat_S1_N files into it.

Alternatively, the folder can be created first, placing and running **Random_generator_S1.py** into it.

This will be your library from where the other codes will read the necessary information. Note that your library is "unique" since, although all the users will share the same z_0, the random rearrangement can be different from one user to another. Using a library ensure the repeatability of computations and separating the effect of random sorting from the variability in the distributions of A_0.

You can use the folder /aleat_S1> for all your application cases without the need of running Random_generator_S1 again, or you can create different libraries if you are particularly interested in testing the effect in chronologies of the random sorting in z_1.

The label _S1 is used here to distinguish the library for CSAR (consisting in 1-column text files) from that used for TERESA (consisting in 2-column text files).

⚙ Preparing the `configuration.json` File

1. Empirical Data

Specify the name and path of the file containing the empirical ^{210}Pb profile.

2. Defining the 3D Parametric Domain

You must define a 3D-domain that presumably contains the best interpretation of the empirical data. This domain is discretized into a regular mesh; each grid point defines a solver.

Central values of the parameters:

- `A0_central`
- `w_central`
- `sA_central`

Sampling intervals are defined using the factors:

- `f_A`, `f_w`, `f_sA`

Example for A0: $[(1 - f\_A)\cdot A0\_central\ ,\ (1 + f\_A)\cdot A0\_central]$
(similar for the other parameters)

All eight parameters are floats.

3. Mesh Resolution

Each interval is discretized into **NR** parts, so the total number of solvers is: NR^3

`NR` is an integer, usually even.

4. Computational Cost

CPU time increases with the number of empirical data points (**N**) and with **NR**. It can range from few minutes up to more than one hour.

Running **CSAR_cronos** is almost instantaneous.

5. Distribution Options

By default, CSAR uses **normal distributions** for initial activity concentrations. You may instead use **lognormal distributions**, in which case the central values and relative deviations must refer to the underlying log-space.

Controlled by:

- `"OP_LN_A0"`

`false` → normal distribution
`true` → lognormal distribution

6. Optional Time Mark

Parameters:

- `kr` — index of the slice with known age (Python indexing: 0-based)
- `Tmr` — known age (yr) of the bottom of the slice
- `sgt` — uncertainty (yr)
- `peso` — weight of the temporal constraint
  - 0 → no time mark
  - 1 → temporal constraint acts as an additional spatial constraint
  - floats allowed (e.g., 2.0)

Even if `peso = 0`, the code still reads `kr`, `Tmr`, and `sgt`. You can state any value (e.g., 1,1,1).

Avoid `sgt = 0`.

7. Confidence Region Method

Two options:

1. **Default method** — simultaneous optimization of all three parameters
2. **Interesting parameters method** — fixes A0 and sA at their absolute minimum and optimizes w. This option is useful for TERESA, but less interesting for CSAR.

Controlled by:

- `"interesting_parameters"` (true/false)

✓ Example `configuration.json`

```
{
  "Core_data": "Core_C1.txt",
  "A0_central": 559.2,
  "w_central": 0.0385,
  "sA_central": 0.20,
  "OP_LN_A0": false,
  "f_A": 0.15,
  "f_w": 0.10,
  "f_sA": 0.25,
  "NR":50,
  "kr":9,
  "Tmr":43.0,
  "sgt":1.0,
  "peso":0,
  "interesting_parameters": false
}
```

▶ Running `CSAR_map.py`

This script reads:

- empirical ^{210}Pb_exc data
- random samples from `/aleat_S1>`
- `configuration.json`

Outputs

- **Map3D.txt**
  Four input values for each solver and the corresponding X_df.
- **Absolute_min.txt**
  Absolute minimum across all solvers, including all parameter values used.

Internal Process

The code defines a sorting function:
Given `(A0_solver, w_solver, sA_solver)`, it generates N pairs `(A0i, wi)` and finds the ordering that minimizes the distance to the empirical profile, including decay and optional time-mark constraints.

▶ Running `CSAR_cronos.py`

This script reads:

- `configuration.json`
- random samples from the library
- `Absolute_min.txt`

- `Map3D.txt`

It computes the **1-sigma interval** and writes all solutions within it to:

- **Cloud.txt**

Then the `profile` function generates full profiles (including chronology), stored in:

- **Plot.txt** (ready for Gnuplot)

Confidence Region Options

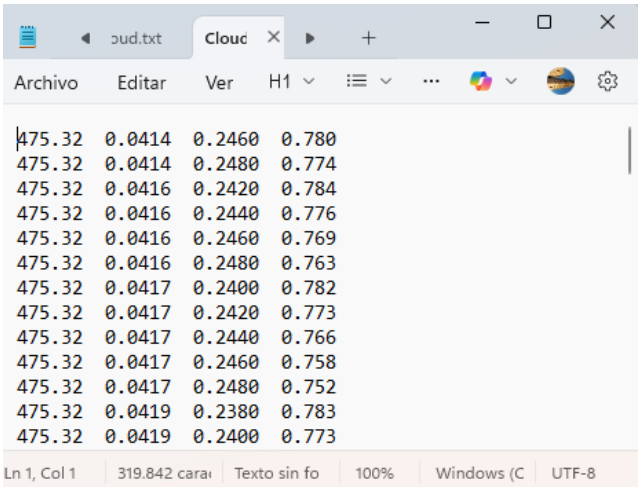
Declared in the configuration `.json` file:

- `interesting_parameters = false` → coefficient 3.53 (default)
- `interesting_parameters = true` → coefficient 1.0

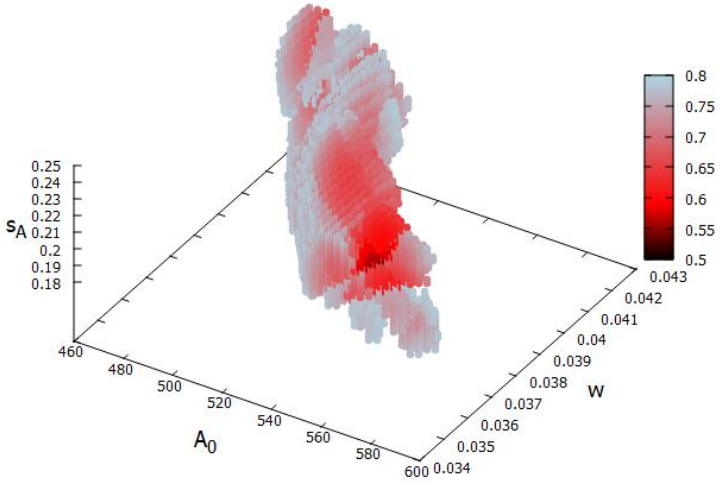
The sampling step is dynamically adjusted to keep the number of sampled solutions below **6000**.

OUPUT FILES AND PLOTS

Cloud.txt file



This file is a sub-set of the Map3D.txt file containing those solvers within the confidence region in χ^2. Each file display the three parameters defining a *solver* $(\bar{A}_0.\bar{w}.s_A)$, being the fourth column its χ_df value. It can be plotted to bring a graphical view of the 'topology of the attractor'.



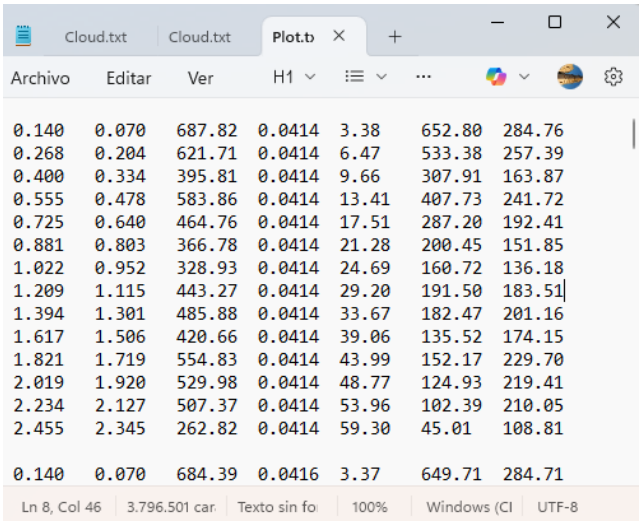
This plot has been generated with gnuplot by using this sprit (lines starting with # can be omitted)

```
# Axis labels
set xlabel "A_{0}" font ",14"
set ylabel "w" font ",14"
set zlabel "s_{A}"font ",14"

set palette defined (0 "black", 0.3 "red", 1 "light-blue")
set cbrange [*:*]
set colorbox

# Plot data: columns 1,2,3 as coordinates; column 4 controls color
splot "Cloud.txt" using 1:2:3:4 with points pt 7 ps 1 palette notitle
```

Plot.txt file



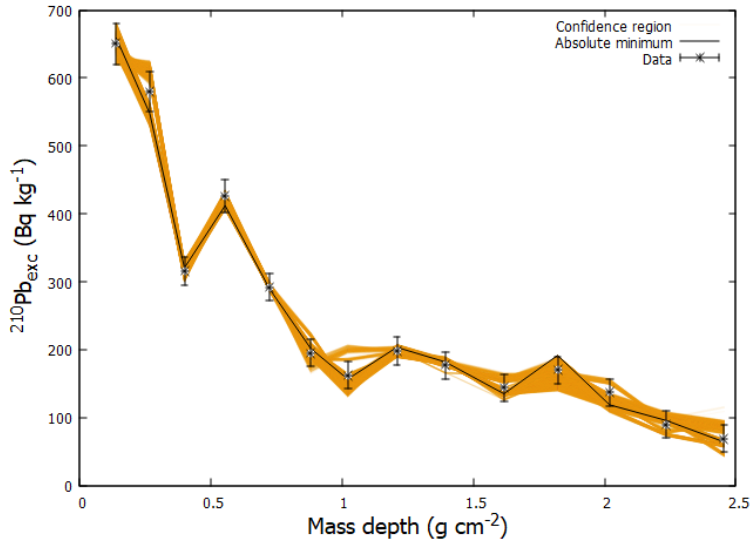
It is a 7-column file with an empty line separating each *solver*:

m_i, mi_m, Sol_A, Sol_w, Sol_T, Sol_Ath, Sol_flux

Column descriptions:
- m_i: Mass depth at the bottom of the sediment slice.
- mi_m: Mass depth at the midpoint of the slice.
- Sol_A: Initial activity concentration at mi_m (in Bq/kg).
- Sol_w: Mean sedimentation rate in the slice (at mi_m), in g/(cm²·yr).
- Sol_T: Age refers to the bottom of the slice (in years).
- Sol_Ath: Theoretical $^{210}Pb_{ex}c$ profile at mi_m, comparable to the empirical profile (in Bq/kg).
- Sol_flux: Mean $^{210}Pb_{ex}c$ flux captured in the slice, in Bq/(m²·yr).

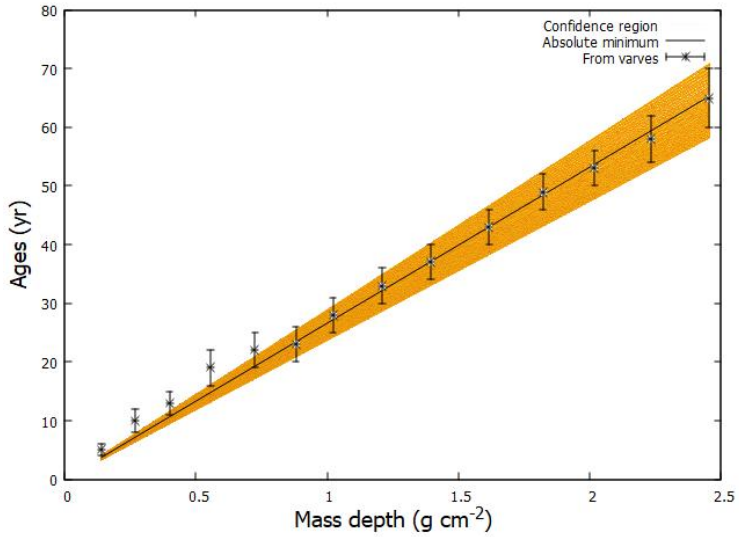
The file contains all the solvers within the confidence region of χ^2.

The '**Solution.txt**' file has the same structure, but contains a single solver (the absolute minimum).



This figure has been generated with the following sript in Gnuplot:

```
set xlabel "Mass depth (g cm^{-2})" font ",14"
set ylabel "^{210}Pb_{exc} (Bq kg^{-1})" font ",14"
plot 'Plot.txt' us 1:6 w l lc 4 lw 0.05 tit "Confidence region", 'Solution.txt' us 1:6 w l lc -1 tit
"Absolute minimum", 'Core_C1.txt' us 1:2:3 w err lc 8 tit "Data"
```



This figure has been generated with the following sript in Gnuplot:

```
set xlabel "Mass depth (g cm^{-2})" font ",14"
set ylabel "Ages (yr)" font ",14"
plot 'Plot.txt' us 1:5 w l lc 4 lw 0.05 tit "Confidence region", 'Solution.txt' us 1:5 w l lc -1 tit
"Absolute minimum", 'Ages_C1.txt' us 1:2:3 w err lc 8 tit "From varves"
```