

CaveCalcV2.0 - Manual

Any queries do not hesitate to reach out at
samuelhollowood.bnc.ox.ac.uk

Contents

1	Installation	2
1.1	Install Python 3	2
1.2	Installing CaveCalcV2.0 module	3
1.3	Testing Installation	4
1.4	Accessing the Graphical User Interface (GUI)	5
2	Model Inputs	5
2.1	Graphical User Interface (GUI)	5
2.2	Python API	7
2.3	Simulating aragonite precipitation	8
3	Model Outputs	9
3.1	File outputs	9
3.2	Plotting Model Outputs	10
3.2.1	Plotting via the GUI	10
3.2.2	Plotting via Python API	11
4	The Carbonate Data Analyser (CDA)	12
4.1	User's Measured Data	13
4.2	Model inputs	13
4.2.1	GUI	13
4.2.2	Python API	15
4.3	CDA Outputs and Plotting	16

5	Advanced manual	18
5.1	Changing D(X) and $\alpha_{44/40}$ within the database files	18
5.2	Non-environmental model inputs	19
6	Tables	20

1 Installation

Step-by-step code for installation can be found in the CaveCalcV2.0 GitHub Repository (<https://github.com/Samhollowood/CaveCalcV2.0/tree/main>). Here, I outline a more thorough installation guide. CaveCalcV2.0 is a Python module that can be installed on Windows, Mac OS X, and Linux systems. Installation of CaveCalcV2.0 requires users to execute some simple commands using the terminal (Mac/Linux) or Command Prompt (Windows). To install CaveCalcV2.0:

1. Install Python 3
2. Install CaveCalcV2.0 module

Notice there is now no need to install IPhreeqc COM for Windows users. This configuration of the IPhreeqc COM server is now done automatically when running the setup.py file (see section 1.2).

For users that are new to using the Terminal window (Mac/Linux) or Command Prompt (Windows), a limited number of useful commands and file locations are provided in the original manual (<https://github.com/Rob-Owen/cavecalc/blob/master/manual.pdf>) in table 1.

1.1 Install Python 3

Python is the programming language used by CaveCalc. It must be installed as a prerequisite. Python 2.7 (the pre-installed default on most systems) is not supported. CaveCalcV2.0 has been tested on Python 3.10.9, and is likely compatible with any version above 3. Python can be downloaded from <https://www.python.org/downloads>.

It is recommended to use Python with Anaconda (<https://www.anaconda.com/download>), a scientific Python distribution, as it simplifies the installation process for dependencies like NumPy, SciPy, and Matplotlib. Anaconda also provides an integrated development environment (IDE) like Spyder, which makes it easier to run and debug models. For users who prefer not to install Anaconda, Python can still be used directly, as the dependencies are installed as part of the setup file anyway.

To check the Python version, run the following command in the Terminal (Mac/Linux) or Command Prompt (Windows):

```
python -V
```

On some Mac/Linux systems, the command `python` may refer to Python 2.x. Use `python3` to ensure Python 3.x is invoked.

If you encounter any issues:

- Ensure Python is installed and added to your PATH
- Confirm that you are using Python 3.x (not Python 2.x).

1.2 Installing CaveCalcV2.0 module

To download the source code for CaveCalcV2.0, you have two options:

1. Using Git

(requires Git to be installed on your system). If Git is not installed, you can download it from <https://git-scm.com>. Open a terminal (Mac/Linux) or command prompt (Windows) and run:

```
git clone https://github.com/Samhollowood/CaveCalcV2.0.git
```

2. Downloading as a ZIP File

Go to the CaveCalcV2.0 GitHub repository and click the green 'Code' button, then select 'Download ZIP'. Extract the ZIP file to your desired directory.

Once you have downloaded the source code, you need to navigate to the directory containing the CaveCalcV2.0 folder. If you used Git or extracted the ZIP file, you can navigate to the directory using the following command:

```
cd CaveCalcV2.0
```

This goes to the working directory to the CaveCalcV2.0 folder, where you can proceed with the installation steps. Note that if you downloaded via the ZIP file, then you may need to add several folders to get to the downloaded CaveCalcV2.0. For example:

```
cd /Users/downloads/CaveCalcV2.0
```

If the ZIP file was extracted into the downloads folder.

If you are using Anaconda, it is recommended to activate your environment before running the installation script. To activate the base environment, run:

```
conda activate base
```

This ensures that CaveCalcV2.0 and its dependencies are installed in your Anaconda base environment. If you prefer to install it into a local anaconda/python environment, you can create one using:

```
conda create -n cavecalc_env
conda activate cavecalc_env
```

or if you do not have Anaconda:

```
python3 -m venv cavecalc_env
source cavecalc_env/bin/activate
```

Then, to install the CaveCalcV2.0 module, run the following command to install CaveCalcV2.0 and its dependencies:

```
python setup.py install
```

This command runs the CaveCalc installation script, which installs the necessary dependencies (e.g., NumPy, SciPy, Matplotlib) and makes CaveCalcV2.0 available as an executable package.

1.3 Testing Installation

After completing the installation, it is recommended to verify that CaveCalcV2.0 has been installed successfully. Navigate to the examples directory within the CaveCalcV2.0 folder:

```
cd CaveCalcV2.0/examples
```

Then, run the following command to execute a sample script:

```
python example1.py
```

If the script runs without errors and produces the expected output, the installation is complete. Any errors/debugging, please contact samuel.hollowood@bnc.ox.ac.uk.

1.4 Accessing the Graphical User Interface (GUI)

To launch the CaveCalcV2.0 graphical user interface (GUI), navigate to the scripts directory within the CaveCalcV2.0 folder:

```
cd CaveCalcV2.0/scripts
```

Then, run the following command:

```
python cc_input_gui.py
```

This will open the CaveCalcV2.0 GUI, where you can interact with the software's features through a user-friendly interface. Here, users can define inputs to run CaveCalcV2.0 models, along with the new Carbonate Data Analyser (CDA) mode.

2 Model Inputs

Once installation is complete, users can begin running forward models. Due to the large number of model inputs, it may be challenging for new users to determine which parameters are most relevant. To address this, updates have been made to improve awareness of each input's impact on speleothem chemistry (see Table 2). There are a few notes here that may be of use to the user.

Note: Atmospheric end-member inputs are only applicable if atmospheric exchange is set to a value greater than 0. In the GUI, this parameter can be found under the *Mix Gas* drop-down heading as the input *Second Gas Fraction (0–1)*. Via the Python API, the model name is `atmo.exchange`.

Note:: If you decide to include trace elements or calcium isotopes, please define them in either the soil (mmol/kgw) or bedrock (mmol/mol); otherwise, default values are 0.

2.1 Graphical User Interface (GUI)

1. General:

- Open the GUI (see section 1.4).
- The GUI should look something like Figure 1.
- Define inputs. Inputs are now organized within drop-down headings. Clicking on a heading reveals all inputs associated with that category.
- Hover over an input to see its impact on speleothem chemistry.

2. Vary Inputs:

- Click the asterisks (*) next to certain input can generate linearly spaced arrays of input values, allowing you to run multiple models simultaneously (Figure ??).

3. Define Output Directory:

- In the bottom left of the GUI (Figure 1), you are able to chose an output directory
- If no output directory is specified, one will be created automatically as `cavecalc_output` in your working directory.

4. Run Model:

- Click the Run CaveCalc Only! button to initiate simulations. Model progress is displayed in the console.

5. Outputs:

- Three output files are generated within the output directory:
 - `settings.pkl` and `results.pkl` (standard CaveCalc outputs).
 - `settings_and_results.csv` (a new, readable format for settings and results).
 - All model inputs keys in which the settings stores can be found in Table 2 output keys in which these files store can be found in Table 3.

6. Access Output GUI for Post Processing:

- You can access output via the Load Output GUI button in the GUI, or by running the following command:

```
python cc_output_gui.py
```

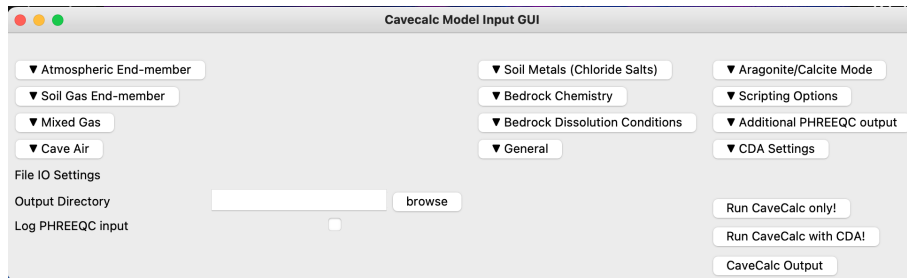


Figure 1: CaveCalcV2.0 GUI. Inputs are grouped under drop-down headings for ease of navigation.

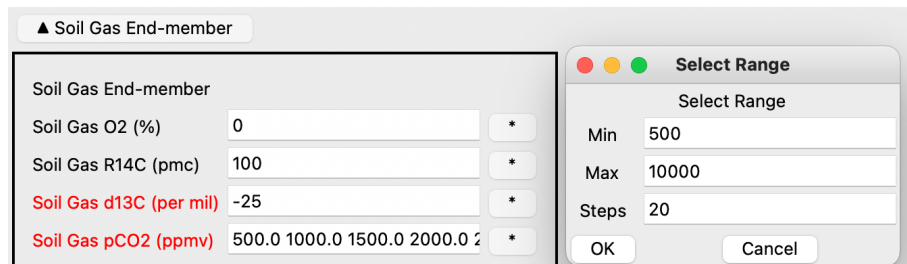


Figure 2: Example of varying an input in steps. This functionality facilitates the generation of multiple models.

2.2 Python API

A new folder, `CaveCalcV2.0/API_Model`, contains Python scripts (`.py` files) that allow users to run models through an IDE environment such as Spyder or Jupyter Notebooks. Among these scripts, `run_models.py` enables users to execute forward models by defining inputs in a settings dictionary `s = {}`. For further guidance on the model inputs, refer to Table 2, which provides information on all model names, realistic input values, and their impact on speleothem chemistry.

Any input can be added, modified, or removed based on the requirements of the study. If an input is removed, the model will use the default value. A list of the default values can be found in `./CaveCalcV2.0/cavecalc/data/defaults.py`.

The general framework of the `run_models.py` script is as follows:

```

1 # Import modules
2 from cavecalc.forward_models import ForwardModels
3

```

```

4  # STEP 1: Define the non-default settings and run models
5  s = {# Define input values (see Table 2).
6  }
7
8  # STEP 2: Run forward models
9  model = ForwardModels(settings=s)
10 model.run_models()
11 model.save()

```

In the real `run_models.py` script, the settings dictionary (`s = {}`) is already populated and all available model names are explicitly defined as their default values. This is apart from the `soil_pCO2`, which is defined as multiple values in a list, showing the user that some inputs can be defined as multiple values. Advanced users may want to create their own scripts for more flexibility, using `run_models.py` as a template.

2.3 Simulating aragonite precipitation

In the GUI, users can define the precipitate mineralogy through the drop-down heading Aragonite/Calcite Mode (Fig. 1). This option allows the user to write ‘Calcite’ or ‘Aragonite’ as the precipitating mineral.

Note: There is no need to explicitly define the database for this functionality.

Alternatively, users can simulate mineralogy directly via the API by including `model_name = ‘precipitate_mineralogy’` in the settings dictionary. The input can be set to either ‘Calcite’ or ‘Aragonite’ depending on the desired simulation:

```

1  # Import modules
2  from cavecalc.forward_models import ForwardModels
3
4  # STEP 1: Define the non-default settings and run models
5  s = {# Define input values (see Table 2).
6  'precipitate_mineralogy': 'Aragonite',
7  }
8
9  # STEP 2: Run forward models
10 model = ForwardModels(settings=s)
11 model.run_models()
12 model.save()

```

The output keys will vary slightly based on the selected precipitate mineralogy (see Table 3)

For instance:

- If calcite is simulated, the output key for a trace element X will be: `X/Ca(mol/mol)_Calcite`.
- If aragonite is simulated, the output key for the same trace element will instead be: `X/Ca(mol/mol)_Aragonite`.
- `d18O_Aragonite` and `d13_Aragonite` and `d44Ca_Aragonite` will be outputs for simulation of aragonite

3 Model Outputs

3.1 File outputs

The new `settings_and_results.csv` file offers a more user-friendly format compared to the original `.pkl` files in CaveCalc. This file combines the data from `settings.pkl` and `results.pkl` into a single, comprehensive table, allowing users to easily view input settings alongside corresponding model outputs, including each step of speleothem chemistry per model (i.e. multiple outputs for a single input setting value).

Advanced users who prefer working with pickle files can still access `settings.pkl` and `results.pkl` directly for custom analyses.

There are also options to convert outputs pickle files into individual `.csv` (one `.csv` per model output) and `.mat` files, for post-processing.

Note : These `.csv` files are different from `settings_and_results.csv`. For these, each output has its own `.csv`, and only stores output (not input settings).

Conversion to these files can be done in the Output GUI, after loading the data from the output directory. Here is an example of converting to a `.csv` and `.mat` file via scripting below:

```

1  # Import modules
2  import cavecalc.analyse as cca
3
4  # Initialize the Evaluate class
5  e = cca.Evaluate()
6  out_dir = 'path/to/output' # Define the output directory
7  e.load_data(out_dir) # Load model data from the output directory
8
9  # Save in different formats
10 e.save_csvs(out_dir)
11 e.save_mat(out_dir)

```

3.2 Plotting Model Outputs

CaveCalc offers two types of plots for visualizing model results:

1. **Plot_models**: Displays model results, with a line representing each model.
2. **Plot_points**: Generates scatter points at specific indices for selected outputs.

3.2.1 Plotting via the GUI

Plots can be generated via the Output GUI (Figure 3). On the Output GUI, load the data by clicking Load Data and then browse to the output directory in which the `settings.pkl` and `results.pkl` are stored. The models loaded should then increase to the number of models ran.

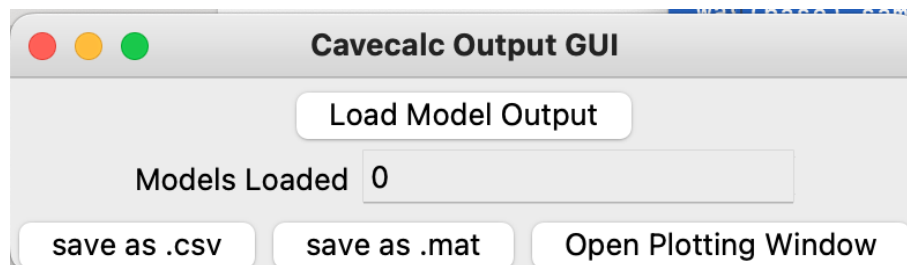


Figure 3: CaveCalcV2.0 Output GUI.

Then clicking Open Plotting Window, will open the plotting functionality. The plotting window should look like Figure ??.

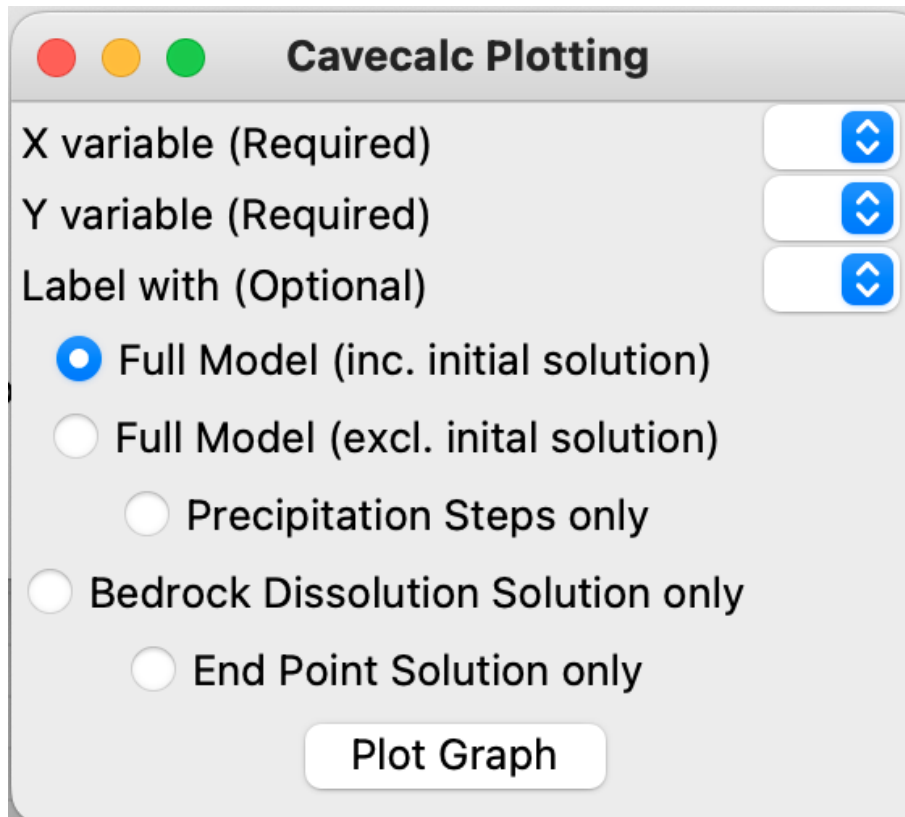


Figure 4: CaveCalcV2.0 Plotting Window

Users can then click the drop-down to plot model outputs as X and Y variables.

3.2.2 Plotting via Python API

The following example demonstrates how to plot model outputs using Python scripting :

```

1  # Import modules
2  import cavecalc.analyse as cca
3
4  # Initialize the Evaluate class
5  e = cca.Evaluate()
6  out_dir = 'path/to/output' # Define the output directory
7  e.load_data(out_dir) # Load model data from the output directory
8
9  # Example 1: Plot model results (e.g., degassing slope against calcite d13C,
```

```

10 # with each line representing a different soil d13C value)
11 plot = e.plot_models(x_key='f_ca', y_key='d13C_Calcite',
    ↪ label_with='soil_d13C')
12
13 # Example 2: Plot scatter points (e.g., equilibrium value of f_ca vs calcite
    ↪ d13C)
14 plot_2 = e.plot_points(x_key='f_ca', y_key='d13C_Calcite',
15                        plot_index=-1, label_with='soil_d13C')

```

The `x_key` and `y_key` parameters define the output variables plotted on the x-axis and y-axis, respectively. The `label_with` parameter determines which input variable is being used to distinguish the different models in the plot.

- `x_key`: Specifies the variable for the x-axis. - Example: `x_key='f_ca'` sets the x-axis to the fraction of calcium removed due to degassing. - Changing this value (e.g., `x_key='pH'`) will plot pH instead of `f_ca` on the x-axis.

- `y_key`: Specifies the variable for the y-axis. - Example: `y_key='d13C_Calcite'` plots calcite d13C on the y-axis. - Changing this to `y_key='Mg/Ca(mol/mol)_Calcite'` would plot the Calcite Mg/Ca instead.

- `label_with`: Determines how different models are labelled in the plot. - Example: `label_with='soil_d13C'` means each line in `plot_models` corresponds to a different model input soil_d13C value. - Changing this to `label_with='temperature'` would distinguish models by the model input temperature instead.

A comprehensive guide to model output keys is provided in Table 3. Users can modify these parameters to visualize different model relationships and trends.

The `plot_index` parameter determines which stage of the speleothem chemistry process is visualized:

- `plot_index=0`: Solution after soil water equilibration.
- `plot_index=1`: Solution after bedrock dissolution.
- `plot_index=-1`: Final equilibrium values for solid and solution phases.

This feature allows users to analyze both intermediate and final model outputs.

4 The Carbonate Data Analyser (CDA)

The only difference between the CDA and standard CaveCalcV2.0 model runs is that users must import their measured data, and optionally define tolerance intervals for each proxy inside the measured data. The operating procedure can be found in the manuscript ().

4.1 User's Measured Data

To use the CDA, users need to import their measured speleothem data in a compatible format. An example .csv file is provided in the supplementary information of the manuscript () or available directly in the CaveCalcV2.0 GitHub Repository. The file should contain the following columns:

- Age (e.g., the age of the sample in yr or kyr BP)
- Measured Data (e.g., measured d18O, d13C, or other isotopic/elemental values)

Isotopic values should be defined as ‰, then a consistent standard. I recommend using d13C (‰, VPDB) and d18O (‰, VPDB) as this is the most popular reference standard. Elemental values should be stored as mmol/mol.

The .csv file can be used as a template to organize your own data in the required format. Columns can be added or deleted depending on the number of proxies you wish to incorporate.

Note:

- Providing a high-resolution dataset will increase the processing time of the CDA.
- High-resolution data may distort the automated plotted output, but will still provide tabular outputs for independent post-processing.

4.2 Model inputs

Here, I outline how to run the CDA with CaveCalcV2.0, either via the GUI (see section 4.2.1) or Python API (see section 4.2.2).

****NOTE: If you decide to include trace-elements or calcium isotopes, please define them in either the soil (mmol/kgw), or bedrock (mmol/mol), otherwise default values are 0****

4.2.1 GUI

It may be difficult to narrow down the input values you wish to investigate for the CDA. As above, if the user is including trace elements and calcium isotopes, they must be defined in the soil or bedrock. As general guidance, environmental model inputs highlighted in red in the GUI denote their high influence on speleothem chemistry and are the inputs within the default plots that are generated. These inputs would be a good place to start. With time and more experience, users may wish to test the CDA with different inputs (e.g. including inputs to test pyrite oxidation).

Required inputs for the CDA are found under the ‘CDA Settings’ field (Fig 5). Here, the user can browse the file path to the measured speleothem data. Optionally, they may also define tolerance intervals of the proxies within the measured data.

Note: Remember the tolerance intervals constitute what is a match or not ($\| \text{measured proxy} - \text{cavecalc proxy} \| < \text{tolerance interval} = \text{match}$)

▲ CDA Settings

CDA Settings

Users filepath to input data	<input type="text"/>	<input type="button" value="browse"/>
d13C(‰) Tolerance	<input type="text" value="1"/>	<input type="button" value="∗"/>
d18O(‰) Tolerance	<input type="text" value="1"/>	<input type="button" value="∗"/>
DCP (%) Tolerance	<input type="text" value="3"/>	<input type="button" value="∗"/>
d44Ca(‰) Tolerance	<input type="text" value="1"/>	<input type="button" value="∗"/>
Mg/Ca(mmol/mol) Tolerance	<input type="text" value="0.5"/>	<input type="button" value="∗"/>
Sr/Ca(mmol/mol) Tolerance	<input type="text" value="0.3"/>	<input type="button" value="∗"/>
Ba/Ca(mmol/mol) Tolerance	<input type="text" value="0.3"/>	<input type="button" value="∗"/>
U/Ca(mmol/mol) Tolerance	<input type="text" value="0.3"/>	<input type="button" value="∗"/>

Plot CDA results vs measured data

CDA results path:

Figure 5: CaveCalcV2.0 GUI

To initialize a CDA run:

1. Define model inputs

2. Under the **CDA Settings** field, browse to the file path to your measured speleothem data.
3. Optionally, define tolerance intervals for proxies within the measured data.
4. Click the **Run CaveCalc with CDA!** button.

If successful, the terminal (Mac/Linux) or command prompt (Windows) will display:

```
CDA is initialized
```

The output will be stored in a folder named `CDA Results`, created within the output directory. When the first match with the measured data is found, the following will be displayed:

```
Created new file path/to/out_dir/CDA Results/Matches.
csv and saved results.
```

After the model runs are complete:

- If there is a match, the CDA will generate plots (see Section 4.3).
- Tabular outputs will also be stored in the `CDA Results` in which the data for the plots (and other inputs/outputs) are stored

4.2.2 Python API

The CaveCalcV2.0 GitHub directory includes an `API_model` folder containing a `run_CDA.py` script for running the CDA. The main difference between a standard CaveCalc run and a CDA run is the need to define the path to your measured data (`user_filepath`) and optionally define tolerance intervals for proxies (e.g., `tolerance_d13C`, `tolerance_d18O`, `tolerance_d44Ca`). If no tolerance intervals are specified, default values are applied.

Below is an example framework for running the CDA:

```
1 # Import modules
2 from cavecalc.forward_models import ForwardModels
3 import cavecalc.analyse as cca
4
5 # STEP 1: Define the non-default settings and run models
6 settings = {
7     # Environmental model inputs
8     'soil_d13C': [-11, -18, -25],
```

```

9      'soil_pCO2': [260, 600, 1000, 2000, 5000, 8000],
10     'gas_volume': [0, 100, 200, 300, 400, 500],
11     'bedrock_d13C': 0,
12     'temperature': [5, 10, 15],
13     'cave_pCO2': [260, 800, 4500],
14     'precipitate_mineralogy': 'Calcite',
15
16     # Path to measured data
17     'user_filepath': 'path/to/data.csv', # Define for CDA
18
19     # Optional tolerance intervals
20     'tolerance_d13C': 0.5,
21     'tolerance_d18O': 0.5,
22     'tolerance_d44Ca': 0.5,
23     'tolerance_MgCa': 0.3,
24     'tolerance_SrCa': 0.3,
25     'tolerance_BaCa': 0.3,
26     'tolerance_UCa': 0.3,
27 }
28
29 # STEP 2: Run CDA models
30 CDA_model = ForwardModels(settings=settings)
31 CDA_model.run_models()
32 CDA_model.save()

```

In this example, the settings include parameters such as `soil_d13C`, `soil_pCO2`, which are plotted in the default selection of figures for the CDA (see section 4.3). These are also the inputs which are highlighted in red within the GUI. Users can modify these or add other parameters as needed. A full list of model inputs and their realistic ranges is provided in Table 2.

Note: Remove tolerance intervals for proxies not included in your measured data.

4.3 CDA Outputs and Plotting

The CDA creates a folder within the model output directory named `CDA Results`. This folder contains the following files:

Matches.csv A CSV file that stores the inputs, outputs, and residuals of all matches with the measured data.

All_outputs.csv A CSV file that stores all inputs, outputs, and residuals from the CDA model runs.

Tolerance.csv A CSV file that stores the tolerance intervals used for the measured proxy data in the CDA.

Input_ranges.csv A CSV file that stores the range of model inputs used in the CDA model runs.

fCa (the fraction of calcium remaining in solution) and $d13C_{initial_solution}$ are selected model outputs which are not isotopic and elemental values of the speleothem. We are open to community feedback to provide any more outputs that would be stored within the `Matches.csv` and `All_outputs.csv` files. The definition of all model outputs is found in Table 3).

Plots are generated automatically after completing all model runs with the CDA. These plots include all variables highlighted in red in the GUI (Figure 5). For additional details on the default plots and their interpretation, refer to the manuscript ().

Note: Some input values that the user investigated may not be part of the default plots but are still archived in the tabular files `Matches.csv` and `All_outputs.csv`.

If needed, users can manually generate plots later using either the GUI or the Python API.

To plot CDA results using the GUI:

1. Open the **CDA Settings** field
2. Under plot CDA vs measured data, navigate to the CDA Results folder by browsing to the archived output directory.
3. Click Plot

The code for plotting CDA results can also be executed using the Python API. Below is an example script:

```
1  # Import modules
2  import cavecalc.analyse as cca
3
4  # Initialize the Evaluate class
5  e = cca.Evaluate()
6
7  # Define path to measured data and archived output directory
8  user_filepath = 'path/to/data.csv' # Path to the measured data used in the CDA
   ↳ run
9  out_dir = 'path/to/out_dir'        # Path to the archived output directory
   ↳ containing CDA results
10
```

```

11 # Plot CDA results
12 plot = e.plot_CDA(user_filepath, out_dir)

```

This script requires the `path/to/data.csv`, which contains the measured data used in the CDA run, and the `path/to/out_dir`, which contains the path to the CDA results.

5 Advanced manual

5.1 Changing D(X) and $\alpha_{44/40}$ within the database files

Users may want to use their own D(X) and $\alpha_{44/40}$ values which are more applicable to their cave system or study (e.g. values derived from drip sites within caves). Altering D(X) and $\alpha_{44/40}$ can be done in the database files, which are ‘calcite.dat’ (for calcite mineralogy) and ‘aragonite.dat’ (for aragonite mineralogy).

Navigate to these files (`cavecalc/data/.`), and open them with a text editor (or any app with text editor capabilities). The databases are large, and the relevant fractionation factors and partition coefficients can be found via a search within the document:

```

Log_alpha_MgCa_Calcite/H2O(1) or Log_alpha_MgCa_Aragonite/H2O(1)
Log_alpha_SrCa_Calcite/H2O(1) or Log_alpha_SrCa_Aragonite/H2O(1)
Log_alpha_BaCa_Calcite/H2O(1) or Log_alpha_BaCa_Aragonite/H2O(1)
Log_alpha_SrCa_Calcite/H2O(1) or Log_alpha_UCa_Aragonite/H2O(1)
Log_alpha_44Ca_Calcite/Ca(aq) or Log_alpha_44Ca_Aragonite/Ca(aq)

```

Under these variables, you will find the reference and the temperature-dependent (in kelvin) relationship with each trace element (existing as $1000 \cdot \ln(X/Ca)$):

$$-\ln_{1000}(\alpha) = C + xT(K) \quad (1)$$

Where C is the constant in the relationship (the first number after the $-\ln_{1000}(\alpha)$), and x is the slope (the second number after the $-\ln_{1000}(\alpha)$). Ca fractionation factors have a single value, because there is not yet an established temperature-dependent relationship. Users may define their own temperature dependence, or replace an original D(X) with a single value. For example, a D(X) of 3 would be $1000 \cdot \ln(D(X)) = 1098.612$ within the database file, which would look like:

```
Log_alpha_XCaCalcite/H2O(1) or Log_alpha_XCa_Aragonite/H2O(1)
#D(X) = 3 in 1000*ln(D(X)) form
-ln_alpha1000    1098.612
```

Note that the PHREEQC databases only deal with temperature dependence and there is currently no way to incorporate a growth rate dependence among other factors.

5.2 Non-environmental model inputs

A few input options exist to handle file IO, manipulate model output, and help with debugging. These are summarised in Table 1.

‘Log PHREEQC input’ controls whether to log commands passed from CaveCalc to IPHreeqc. If set to TRUE, CaveCalc will write a log file of PHREEQC input for each model calculation performed. The log files generated are valid PHREEQC input and may be run through PHREEQC to provide further information about a model run. This is particularly useful for debugging failed model runs, and may also help users better understand the inner workings of CavecalcV2.0. ‘PHREEQC Log Filename’ determines the file naming convention for any log files created; is replaced with an arbitrary number when more than one model is run.

‘PHREEQC Database Filename’ gives the name of the PHREEQC database to be used. By default this is calcite.dat (the database distributed with cavecalc). If another suitable database is available, its name may be given here. If the file is located outside the cavecalc/data directory, the full path should be given. ‘Output Directory’ specifies the location where model output files and any log files will be saved; if left blank, the current directory is used.

‘Totals’, ‘Molalities’ and ‘Isotopes’ input parameters are used to customise the chemical data returned by Cavecalc. These inputs are passed to the SELECTED_OUTPUT PHREEQC input block (see PHREEQC manual).

Paramater	Model	Default Value
Log PHREEQC Input	phreeqc_log_file	FALSE
Output Directory	out_dir	''
PHREEQC Database Filename	database	calcite.dat
PHREEQC Log Filename	phreeqc_log_file_name	log-{}.phr
Totals	totals	
Molalities	molalities	HCO3- CO3-2
Isotopes	isotopes	R(44Ca) R(18O) ...*

Table 1:

6 Tables

Model Name	Realistic Range of Values	Impact on speleothem chemistry
Amtospheric Gas end-member		
atm_O2	0.21 (%)	If the atmosphere and soil is exchanged, this value impacts gaseous oxygen in the soil, with implications for pyrite oxidation.
atm_d18O	-25 to 5 (‰, VSMOW)	The oxygen isotopic composition of rainwater. Impacts speleothem d18O (‰, VPDB)
atm_pCO2	150 to 480 ppmv	The atmospheric CO2 concentration. If the atmosphere and soil is exchanged, this value impacts gaseous CO2 in the soil.
atm_d13C	-9 to -6 (‰, VPDB)	The carbon isotopic composition of the atmosphere. If the atmosphere and soil is exchanged, this value impacts the carbon isotopic composition in the soil.

atm_14C	100 (pmc)	The radiocarbon activity in the atmosphere. If the atmosphere and soil is exchanged, this impact speleothem DCP.
Soil Gas end-member		
soil_O2	0-0.21 (%)	The O2 percentage in the soil, with implications for pyrite oxidation
soil_R14C	100 (pmc)	The radiocarbon activity in the soil. Impacts speleothem DCP.
soil_d13C	-30 to 6 (‰)	The carbon isotopic composition within the soil. Impacts speleothem d13C.
soil_pCO2	260 to 10,000 (ppmv)	The CO2 in the concentration in the soil. Impacts amount of bedrock dissolution. Also impacts the degassing slope and the amount of prior carbonate precipitation. Impacts speleothem d13C, d44Ca, DCP and X/Ca
Atmospheric and soil gas mixing		
atmo_exchange	0 to 1	Sets the atmospheric contribution (see atmospheric gas end-members) to the soil

Model Name	Realistic Range of Values	Impact on speleothem chemistry
Soil metals		
soil_Ba	Site-specific (mol/kgw)	Sets the terrigenous input of Barium
soil_Ca	Site-specific (mol/kgw)	Sets the terrigenous input of Calcium
soil_Mg	Site-specific (mol/kgw)	Sets the terrigenous input of Magnesium
soil_Sr	Site-specific (mol/kgw)	Sets the terrigenous input of Strontium
soil_U	Site-specific (mol/kgw)	Sets the terrigenous input of Uranium
soil_d44Ca	~0	Sets the calcium isotopic composition of terrigenous inputs
Bedrock Chemistry		
bedrock_BaCa	Site-specific (mmol/mol)	Sets the Ba/Ca of the bedrock
bedrock_MgCa	Site-specific (mmol/mol)	Sets the Mg/Ca of the bedrock
bedrock_SrCa	Site-specific (mmol/mol)	Sets the Sr/Ca of the bedrock
bedrock_UCa	Site-specific (mmol/mol)	Sets the U/Ca of the bedrock
bedrock_d44Ca	Site-specific (‰)	Sets the calcium isotopic composition of the bedrock
bedrock_d13C	-3 to 4 (‰, VPDB)	Sets the carbon isotopic composition of the bedrock
bedrock_d18O	-5 to 3 (‰, VPDB)	Sets the oxygen isotopic composition of the bedrock
bedrock_mineral	‘Dolomite’, ‘Calcite’ or ‘Aragonite’	Sets the mineralogy of the bedrock
Bedrock Conditions		
bedrock (moles)	10 (default) (moles)	Moles of bedrock. Defaults to excess moles
bedrock_pyrite	0 to (moles)	Moles of pyrite in bedrock. Initialises pyrite oxidation in the presence of oxygen.

gas_volume	0 to 500 (L/kg)	Controls the amount of gas present during dissolution. Impacts speleothem d13C and DCP
reprecip	False or True	Allows for carbonate re-precipitation. Impacts speleothem d13C and DCP.
Aragonite/Calcite Mode		
precipitate_mineralogy	'Calcite' or 'Aragonite'	Sets the speleothem mineralogy
General		
temperature	0 to 35 (Celsius)	Sets in-cave temperature. Impacts the kinetics, thermodynamics, partitioning coefficients and fractionation factors.
kinetics_mode	‘ , ’	Sets how the model is ran. See original publication.
Scripting Options		
co2_decrement	0.5	Sets the fraction of CO2 removed at each degassing step
calcite_sat_limit	1	Sets the index at which CaCO3 precipitation occurs

Table 2:

Model Outputs (Key)	Definition
Ba(mol/kgw)	The Barium concentration within the solution
Ba/Ca(mol/mol)	The ratio of Barium to Calcium within the solution
Ba/Ca(mol/mol).Calcite / Ba/Ca(mol/mol).Aragonite	Speleothem Ba/Ca value
C(mol/kgw)	The Carbon concentration within the solution
Ca(mol/kgw)	The Calcium concentration within the solution
CO2(mol/kgw)	The CO2 concentration within the solution
CO3-2(mol/kgw)	The carbonate ion concentration within the solution
d13C	The Dissolved Inorganic Carbon (DIC) d13C
d13C.Calcite / d13C.Aragonite	Speleothem d13C
d13C.CO2(aq)	The d13C of CO2 in aqueous form within the solution
d13C.CO3-2	The d13C of the carbonate ion within the solution
d13C.HCO3- d18O	The d13C of the bicarbonate ion within the solution
d18O	The d18O of the H2O reservoir (VSMOW)
d18O.Calcite / d18O.Aragonite	Speleothem d18O (VSMOW)
d18O.CO2(aq)	The d18O of CO2 in aqueous form within the solution (VSMOW)
d18O.CO3-2	The d18O of the carbonate ion within the solution (VSMOW)
d18O.HCO3- d18O.PDB	The d18O of the bicarbonate ion within the solution (VSMOW)
d44Ca	Speleothem d18O (VPDB)
d44Ca.Calcite / d44Ca.Aragonite	The d44Ca within the solution
DCP	Speleothem d44Ca
f_c	Speleothem Dead Carbon Percentage
f_ca	Fraction of carbon remaining in solution
HCO3-(mol/kgw)	Fraction of calcium remaining in the solution
Mg(mol/kgw)	Concentration of the bicarbonate ion within the solution
Mg/Ca(mol/mol)	The concentration of Magnesium within the solution
Mg/Ca(mol/mol).Calcite / Mg/Ca(mol/mol).Aragonite	The ratio of Magnesium to Calcium within the solution
O(mol/kgw)	Speleothem Mg/Ca
pH	Concentration of oxygen within the solution
R14C	pH of the solution
si.Calcite	Speleothem R14C
Sr(mol/kgw)	Saturation index with respect to calcite
Sr/Ca(mol/mol)	Concentration of Strontium within the solution
Sr/Ca(mol/mol).Calcite / Sr/Ca(mol/mol).Aragonite	The ratio of Strontium to Calcium within the solution
step_desc	Speleothem Sr/Ca
U(mol/kgw)	The step description (e.g soil-water equilibration, bedrock dissolution or degassing?)
U/Ca(mol/mol)	Concentration of Uranium within the solution
U/Ca(mol/mol).Calcite / U/Ca(mol/mol).Aragonite	The ratio of Uranium to Calcium within the solution
	Speleothem U/Ca

Table 3: