
ForamGeochem Documentation

Release 0.0.1

Oscar Branson

Feb 03, 2020

1	Installation	3
2	Using ForamGeochem	5
2.1	Converters	5
2.2	Transfer Functions	6
2.3	Uncertainty Propagation	6
3	Mg/Ca	9
3.1	Exponential	9
3.2	Holland et al (sub.)	9
4	Parameters	11
5	Mg/Ca	13
5.1	Converters	13
5.2	Transfer Functions	13
6	Inner Workings	15
6.1	<i>proxy</i> class	15
6.2	Helpers	15

In these pages you'll find general information on how to use the `foramgeochem` module, and currently implemented proxies.

CHAPTER 1

Installation

foramgeochem is available as a Python package, and can be installed from a terminal window using `pip`:

```
pip install foramgeochem
```

This requires a working Python installation. If you don't have one, are unsure if you do, try installing the [Anaconda Python Distribution](#), which should include everything you need to get started.

Using ForamGeochem

Each proxy in *foramgeochem* is included as a sub-module, which contains all functions related to that proxy. For example, all functions relating to the Mg/Ca palaeothermometer are within the `foramgeochem.mgca` module. To import the Mg/Ca submodule:

```
from foramgeochem import mgca
```

Implemented Proxies:

- *Mg/Ca*

2.1 Converters

Each proxy contains one or more ‘converters’. These contain everything you need to use a particular form of the proxy, including the ability to load parameter presets, and automatic uncertainty propagation. This will be sufficient for most cases.

For example, the `mgca` module contains two converters: `exponential` and `Holland`, which contain two different forms of the proxy equation. The former relates Mg/Ca to temperature via the ‘classic’ exponential equation, whereas the latter uses the new formulation of Holland et al (sub.) that includes for multiple aspects of seawater chemistry. These converters are available directly from the `mgca` submodule, after you’ve imported it (above), and can be used like this:

```
my_record = mgca.exponential(mgca_f=[1, 2, 3, 4], parameters='Multispecies_Anand')
```

This creates a ‘converter’ object called *my_record* (you can call this whatever you want), which stores your data and loads parameters from an internal database. To find out what state your converter object is in, simply type *my_record* to see a printout of its current state:

```
my_record

> Exponential Mg/Ca-Temperature Relationship
> -----
```

(continues on next page)

(continued from previous page)

```

> mgca_f = A * exp(temperature * B)
>
> Parameter Set: Multispecies_Anand
>   Exponential function re-fit to multispecies data of Anand et al (1996)
>   by Holland et al (sub.).
>
> Parameter Values:
>   A: 0.445+/-0.030
>   B: 0.0845+/-0.0027
>
> Variables:
>   Accepted: {'temperature', 'mgca_f'}
>   Provided: {'mgca_f'}

```

Here, you can see which Mg/Ca-temperature relationship and parameter set you are using, as well as which variables the converter accepts and what you’ve provided. The reported parameter uncertainties here are 1SD errors, which will be propagated through all calculations.

To calculate temperature from your Mg/Ca data:

```

record.calc_temp()

> array([9.58058964571389+/-0.48410211276655907,
>        17.779096397932225+/-0.2363683581369949,
>        22.574915409889144+/-0.12495085593950557,
>        25.977603150150557+/-0.12556444392484026], dtype=object)

```

2.2 Transfer Functions

At the ‘lower level’, each proxy contains a set of ‘Transfer Functions’ used to convert between all variables. These can be accessed within the *tfr* module inside each proxy. For Mg/Ca, these are in `foramgeochem.mgca.tfr`.

2.3 Uncertainty Propagation

Parameter uncertainties are propagated automatically in all calculations using the `uncertainties` module. This module uses [linear error propagation theory](#) to estimate the uncertainty in calculated variables, and automatically handles uncertainty correlations. Its main limitation is that it assumes your uncertainties are normally distributed. This approximation will be sufficient in most cases, but for cases where you *know* your uncertainties are non-normal, we recommend Monte-Carlo uncertainty sampling.

The uncertainties module can also be used to incorporate uncertainties on input variables (e.g. measurement uncertainty):

```

from uncertainties import unumpy as unp

meas_mgca_f = unp.uarray(nominal_values=[1, 2, 3, 4], std_devs=[.08, .12, .11, .09])

my_record = mgca.exponential(mgca_f=meas_mgca_f, parameters='Multispecies_Anand')

calced_temperature = my_record.calc_temp()

calced_temperature # contains packaged means and uncertainties

```

(continues on next page)

(continued from previous page)

```
> array([9.58058964571389+/-1.0628813063828466,  
>        17.779096397932225+/-0.7480046939404693,  
>        22.574915409889144+/-0.4513323339150851,  
>        25.977603150150557+/-0.2942633993831899], dtype=object)  
  
# to get means alone:  
unp.nominal_values(calced_temperature)  
  
> array([ 9.58058965, 17.7790964 , 22.57491541, 25.97760315])  
  
# and standard deviations:  
unp.std_devs(calced_temperature)  
  
> array([1.06288131, 0.74800469, 0.45133233, 0.2942634 ])
```


Functions for converting between foraminiferal Mg/Ca and environmental parameters.

3.1 Exponential

The ‘classic’ exponential function, where Mg/Ca depends only on temperature, following:

$$Mg/Ca_{foram} = A e^{Temp B}$$

3.1.1 Functions

Converter:

- `foramgeochem.mgca.exponential`

Transfer functions:

- `exp_mgca_2_temp()`
- `exp_temp_2_mgca()`

3.2 Holland et al (sub.)

The multi-factor formulation of Holland et al (sub.), which accounts for the influence of Temperature, Mg/Ca_{SW}, [Ca]_{SW} and a species-specific carbon-system parameter (*carb*, i.e. DIC_{SW}, [CO₃]_{SW} or pH).

$$Mg/Ca_{foram} = Mg/Ca_{SW}^A DIC_{SW}^B e^{(C [Ca]_{SW} + DT + E)}$$

3.2.1 Functions

Converter:

- `foramgeochem.mgca.holland_2020`

Transfer functions:

- `holland2018_calc_mgca()`
- `holland2018_calc_temp()`
- `holland2018_calc_mgca_sw()`
- `holland2018_calc_Ca_sw()`
- `holland2018_calc_carb_sw()`

CHAPTER 4

Parameters

5.1 Converters

5.2 Transfer Functions

CHAPTER 6

Inner Workings

6.1 *proxy* class

6.2 Helpers