interferences Documentation

Release 0.0.3

Morgan Williams

May 13, 2022

CONTENTS

interferences is a set of tools for analysing inorganic mass spectra and interference patterns.

- On this site you can browse the API, or look through some of the usage examples.
- There's also a quick installation guide, a list of recent changes and some notes on where the project is heading in the near future.
- If you're interested in contributing to the project, there are many potential avenues, whether you're experienced with Python or not.

CHAPTER

ONE

INSTALLATION

`bash pip install git+git://github.com/morganjwilliams/interferences.
git@develop#egg=interferences `

1.1 Upgrading interferences

`bash pip install --upgrade git+git://github.com/morganjwilliams/interferences.
git@develop#egg=interferences `

1.2 Optional Dependencies

Optional dependencies (*dev*, *docs*) can be specified during *pip* installation.

`bash pip install git+git://github.com/morganjwilliams/interferences.
git@develop#egg=interferences[docs] `

CHAPTER

TWO

EXAMPLES

This example gallery includes a variety of examples for using `interferences` which you can copy, download and alter, or run on Binder.

2.1 Spectra : stemplot

Once you have a table, one way to visualise how the ion peaks are distributed is to use `stemplot()`, which you can also access from your dataframe using the `mz` accessor:

```
import matplotlib.pyplot as plt
from interferences import build_table
from pyrolite.geochem.ind import REE
```

Let's build a table to play with first, focusing on potential interferences for thulium (Tm), which has only one stable isotope (^{169}Tm), and is susceptible to interferences, especially for quadrupole ICP-MS:

```
window = ("Tm[169]", 0.1)
df = build_table(REE() + ["O", "N", "H"], window=window, max_atoms=2)
```

Out:

```
  0%|          | 0/113 [00:00<?, ?it/s]
                      :   0%|          | 0/113 [00:00<?, ?it/s]
Gd @ 14 rows          :   0%|          | 0/113 [00:00<?, ?it/s]
Tb @ 2 rows           :   0%|          | 0/113 [00:00<?, ?it/s]
Dy @ 14 rows          :   0%|          | 0/113 [00:00<?, ?it/s]
Ho @ 2 rows           :   0%|          | 0/113 [00:00<?, ?it/s]
Er @ 12 rows          :   0%|          | 0/113 [00:00<?, ?it/s]
Tm @ 2 rows           :   0%|          | 0/113 [00:00<?, ?it/s]
Yb @ 14 rows          :   0%|          | 0/113 [00:00<?, ?it/s]
Lu @ 4 rows           :   0%|          | 0/113 [00:00<?, ?it/s]
La-O @ 12 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
La-N @ 8 rows         :   0%|          | 0/113 [00:00<?, ?it/s]
Ce-O @ 24 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
Ce-N @ 16 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
Pr-O @ 6 rows         :   0%|          | 0/113 [00:00<?, ?it/s]
Pr-N @ 4 rows         :   0%|          | 0/113 [00:00<?, ?it/s]
Nd-O @ 42 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
Nd-N @ 28 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
Sm-O @ 42 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
```

(continues on next page)

(continued from previous page)

```
Sm-N @ 28 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
Eu-O @ 12 rows        :   0%|          | 0/113 [00:00<?, ?it/s]
Eu-O @ 12 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Eu-H @ 8 rows         :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Eu-N @ 8 rows         :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Gd-O @ 42 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Gd-H @ 28 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Gd-N @ 28 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Gd-Sm @ 98 rows       :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Gd-Eu @ 28 rows       :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Gd-Gd @ 56 rows       :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Tb-O @ 6 rows         :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Tb-H @ 4 rows         :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Tb-N @ 4 rows         :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Tb-Sm @ 14 rows       :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Tb-Eu @ 4 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Tb-Gd @ 14 rows       :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Tb-Tb @ 2 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Dy-O @ 42 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Dy-H @ 28 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Dy-N @ 28 rows        :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Dy-Nd @ 98 rows       :  17%|#6        | 19/113 [00:00<00:00, 183.34it/s]
Dy-Nd @ 98 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Dy-Sm @ 98 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Dy-Eu @ 28 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Dy-Gd @ 98 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Dy-Tb @ 14 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Dy-Dy @ 56 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-O @ 6 rows         :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-H @ 4 rows         :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-N @ 4 rows         :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Ce @ 8 rows        :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Pr @ 2 rows        :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Nd @ 14 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Sm @ 14 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Eu @ 4 rows        :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Gd @ 14 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Tb @ 2 rows        :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Dy @ 14 rows       :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Ho-Ho @ 2 rows        :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Er-O @ 36 rows        :  34%|###3      | 38/113 [00:00<00:00, 175.51it/s]
Er-O @ 36 rows        :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-H @ 24 rows        :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-N @ 24 rows        :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-La @ 24 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Ce @ 48 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Pr @ 12 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Nd @ 84 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Sm @ 84 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Eu @ 24 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Gd @ 84 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Tb @ 12 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
```

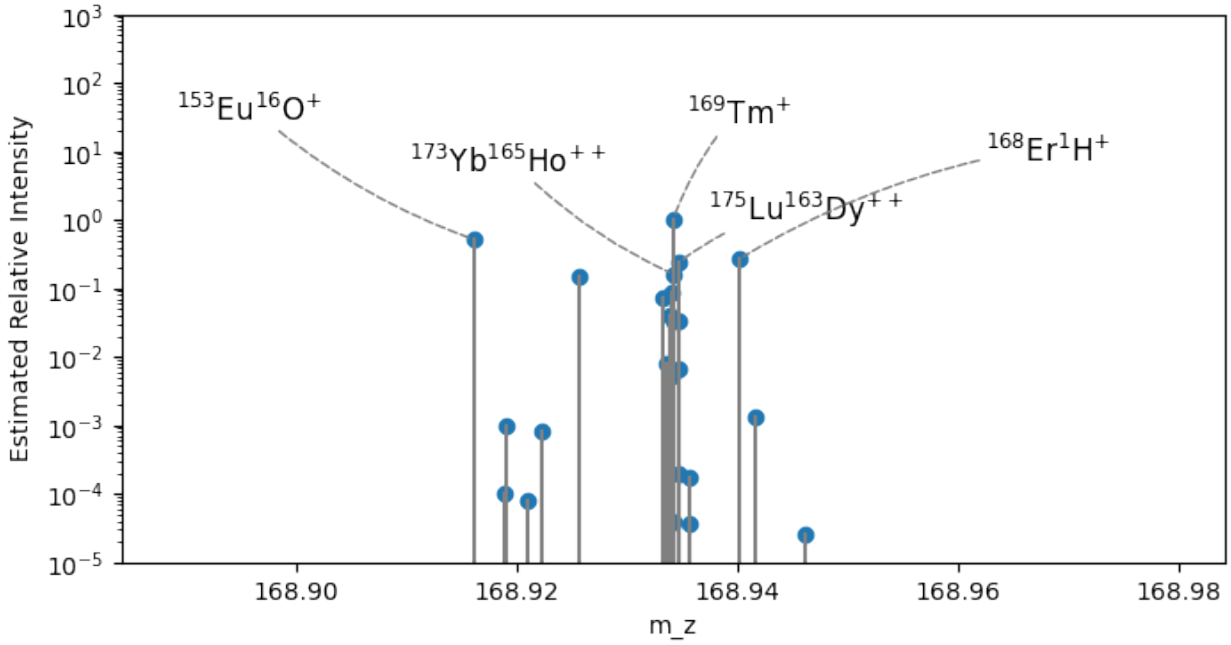
(continues on next page)

(continued from previous page)

```
Er-Dy @ 84 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Ho @ 12 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Er-Er @ 42 rows       :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Tm-O @ 6 rows         :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Tm-H @ 4 rows         :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Tm-N @ 4 rows         :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Tm-La @ 4 rows        :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Tm-Ce @ 8 rows        :  50%|####9     | 56/113 [00:00<00:00, 175.60it/s]
Tm-Ce @ 8 rows        :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Pr @ 2 rows        :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Nd @ 14 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Sm @ 14 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Eu @ 4 rows        :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Gd @ 14 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Tb @ 2 rows        :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Dy @ 14 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Ho @ 2 rows        :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Er @ 12 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Tm-Tm @ 2 rows        :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-H @ 28 rows        :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-La @ 28 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-Ce @ 56 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-Pr @ 14 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-Nd @ 98 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-Sm @ 98 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-Eu @ 28 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-Gd @ 98 rows       :  65%|######5   | 74/113 [00:00<00:00, 172.37it/s]
Yb-Gd @ 98 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Yb-Tb @ 14 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Yb-Dy @ 98 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Yb-Ho @ 14 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Yb-Er @ 84 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Yb-Tm @ 14 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Yb-Yb @ 56 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-H @ 8 rows         :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-La @ 8 rows        :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Ce @ 16 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Pr @ 4 rows        :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Nd @ 28 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Sm @ 28 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Eu @ 8 rows        :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Gd @ 28 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Tb @ 4 rows        :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Dy @ 28 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Ho @ 4 rows        :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Er @ 24 rows       :  81%|########1 | 92/113 [00:00<00:00, 171.63it/s]
Lu-Er @ 24 rows       :  97%|#########7| 110/113 [00:00<00:00, 171.36it/s]
Lu-Tm @ 4 rows        :  97%|#########7| 110/113 [00:00<00:00, 171.36it/s]
Lu-Yb @ 28 rows       :  97%|#########7| 110/113 [00:00<00:00, 171.36it/s]
Lu-Lu @ 6 rows        :  97%|#########7| 110/113 [00:00<00:00, 171.36it/s]
Lu-Lu @ 6 rows        : 100%|##########| 113/113 [00:00<00:00, 173.21it/s]
```

From this table, we can create our plot, here limiting the labelling to the five peaks with highest estimated intensity:

```
ax = df.mz.stemplot(window=window, max_labels=5, figsize=(8, 4))
plt.show()
```



Out:

```
/home/docs/checkouts/readthedocs.org/user_builds/interferences/envs/latest/lib/python3.7/
↪site-packages/numpy/core/fromnumeric.py:43: VisibleDeprecationWarning: Creating an␣
↪ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or␣
↪ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you␣
↪must specify 'dtype=object' when creating the ndarray.
  result = getattr(asarray(obj), method)(*args, **kwds)
```

While the production of the doubly-charged double-REE ions is likely less significant than shown here (no penalisation for higher charges/larger molecules is included in generating these spectra), we can see that $^{153}\mathrm{Eu}^{16}\mathrm{O}$ could be a potential interference issue if the conditions are relatively oxidised, and if there's sufficient hydrogen, $^{168}\mathrm{Er}^{1}\mathrm{H}$ may similarly contribute to problems.

Notably, there's a number of other potential ions in vicinity of $^{169}\mathrm{Tm}$. However, most of these are doubly-charged double-REE ions. Given the highly-correlated nature of the REE, these may not pose as significant issues for standardisation as the hydride and oxide ions.

Total running time of the script: (0 minutes 7.653 seconds)

2.2 Building Ion Tables

`interferences` core funciton is to generate, filter and visualise tables of isotope-specified small inorganic ionic molecules. This example demonstrates how to build a small table of ions, and some of the options available. Note that as `interferences` is largely built around `pandas`, you can expect to be working with `DataFrame` objects most of the time.

```
import pandas as pd
from interferences.table import build_table
```

In the simplest case, where you have a list of elments you want to find viable set of ions which they may produce, you can simply specify these elements in a call to `build_table()`:

```
df = build_table(["Ca", "O", "Ar", "H"])
df.info()
```

Out:

```
  0%|          | 0/34 [00:00<?, ?it/s]
                        :   0%|          | 0/34 [00:00<?, ?it/s]
O @ 6 rows              :   0%|          | 0/34 [00:00<?, ?it/s]
H @ 4 rows              :   0%|          | 0/34 [00:00<?, ?it/s]
Ca @ 12 rows            :   0%|          | 0/34 [00:00<?, ?it/s]
Ar @ 6 rows             :   0%|          | 0/34 [00:00<?, ?it/s]
O-O @ 12 rows           :   0%|          | 0/34 [00:00<?, ?it/s]
H-O @ 12 rows           :   0%|          | 0/34 [00:00<?, ?it/s]
H-H @ 6 rows            :   0%|          | 0/34 [00:00<?, ?it/s]
Ca-O @ 36 rows          :   0%|          | 0/34 [00:00<?, ?it/s]
Ca-H @ 24 rows          :   0%|          | 0/34 [00:00<?, ?it/s]
Ca-Ca @ 42 rows         :   0%|          | 0/34 [00:00<?, ?it/s]
Ar-O @ 18 rows          :   0%|          | 0/34 [00:00<?, ?it/s]
Ar-H @ 12 rows          :   0%|          | 0/34 [00:00<?, ?it/s]
Ar-Ca @ 36 rows         :   0%|          | 0/34 [00:00<?, ?it/s]
Ar-Ar @ 12 rows         :   0%|          | 0/34 [00:00<?, ?it/s]
O-O-O @ 20 rows         :   0%|          | 0/34 [00:00<?, ?it/s]
H-O-O @ 24 rows         :   0%|          | 0/34 [00:00<?, ?it/s]
H-H-O @ 18 rows         :   0%|          | 0/34 [00:00<?, ?it/s]
H-H-H @ 8 rows          :   0%|          | 0/34 [00:00<?, ?it/s]
H-H-H @ 8 rows          :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ca-O-O @ 72 rows        :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ca-H-O @ 72 rows        :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ca-H-H @ 36 rows        :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ca-Ca-O @ 126 rows      :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ca-Ca-H @ 84 rows       :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ca-Ca-Ca @ 112 rows     :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-O-O @ 36 rows        :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-H-O @ 36 rows        :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-H-H @ 18 rows        :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-Ca-O @ 108 rows      :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-Ca-H @ 72 rows       :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-Ca-Ca @ 126 rows     :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-Ar-O @ 36 rows       :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-Ar-H @ 24 rows       :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
```

(continues on next page)

(continued from previous page)

```
Ar-Ar-Ca @ 72 rows       :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-Ar-Ar @ 20 rows       :  53%|#####2    | 18/34 [00:00<00:00, 178.23it/s]
Ar-Ar-Ar @ 20 rows       : 100%|##########| 34/34 [00:00<00:00, 151.69it/s]
<class 'pandas.core.frame.DataFrame'>
Index: 1344 entries, H[1]++ to Ca[48]Ca[48]Ca[48]+
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   m_z          1344 non-null   float64
 1   mass         1344 non-null   float64
 2   charge       1344 non-null   int64
 3   iso_product  1344 non-null   float64
dtypes: float64(3), int64(1)
memory usage: 52.5+ KB
```

Note that the table is indexed by the ions themselves, with the respective mass/charge ratio found under the *m_z* column. Even with a small set of elements, with the combination of isotopes, we've generated a fair number of potential ions:

```
df.index.size
```

Out:

```
1344
```

As this is probably more ions than you want to consider, let's narrow the foucs using a mass window (here for $39 \geq m/z \leq 41$):

```
df = build_table(["Ca", "O", "Ar", "H"], window=(39, 41))
df.index.size
```

Out:

```
48
```

While you can use m/z ratios if you know them specifically, it's likely more useful to specify an ion mass and a mass-window either side, which you can do as follows:

```
df = build_table(["Ca", "O", "Ar", "H"], window=("Ca[40]", 0.01))
df.index.size
```

Out:

```
6
```

By default, `interferences` builds ions for molecules with up to three atoms with ionic charges of either *+1* or *+2* (note the sign of the charge is largely irrelevant, given a mass spectrometer will be set up for either positive or negative ions). If you wanted to use different parameters to generate a table, you can use the *max_atoms* and *charges* keyword arguemnts:

```
df = build_table(["Ca", "O", "Ar", "H"], charges=[1], max_atoms=2)
df.index.size
```

Out:

```
119
```

Also, to save time for futher computation, `interferences` uses a local HDFStore to cache results. You can disable this behaviour and gain some speed if you're generating one-off large tables by using `cache_results=False`:

```
df = build_table(["Ca", "O", "Ar", "H"], cache_results=False)
```

If you're finding that you end up with a table which includes minor ions which likely have too low an isotopic abundance to influence results, you can use the *threshold* keyword argument to adjust the isotopic abundance threshold for isotopes used to build the table. Note that these won't be added to the cached reference:

```
df = build_table(["N", "K"], threshold=0.5)
df.index.size
```

Out:

```
  0%|          | 0/9 [00:00<?, ?it/s]
                        :   0%|          | 0/9 [00:00<?, ?it/s]
N @ 2 rows              :   0%|          | 0/9 [00:00<?, ?it/s]
K @ 4 rows              :   0%|          | 0/9 [00:00<?, ?it/s]
N-N @ 2 rows            :   0%|          | 0/9 [00:00<?, ?it/s]
K-N @ 4 rows            :   0%|          | 0/9 [00:00<?, ?it/s]
K-K @ 6 rows            :   0%|          | 0/9 [00:00<?, ?it/s]
N-N-N @ 2 rows          :   0%|          | 0/9 [00:00<?, ?it/s]
K-N-N @ 4 rows          :   0%|          | 0/9 [00:00<?, ?it/s]
K-K-N @ 6 rows          :   0%|          | 0/9 [00:00<?, ?it/s]
K-K-K @ 8 rows          :   0%|          | 0/9 [00:00<?, ?it/s]
K-K-K @ 8 rows          : 100%|##########| 9/9 [00:00<00:00, 194.34it/s]

35
```

Finally, if you're likely to do some plotting with the ion data, you can specify this using the *add_labels* keyword argument, which will add nicely formatted labels comapatible with `matplotlib`:

```
df = build_table(["Ca", "O", "Ar", "H"], add_labels=True)
```

Total running time of the script: (0 minutes 3.216 seconds)

2.3 Spectra : spectra

If you would like something that looks similar to your peak scans, you can use `spectra()`, which you can also access from your dataframe using the `mz` accessor:

```
import matplotlib.pyplot as plt
from interferences import build_table
from pyrolite.geochem.ind import REE
```

Here build a table based on some low-mass isotopes, and focus in on the BO+ ion:

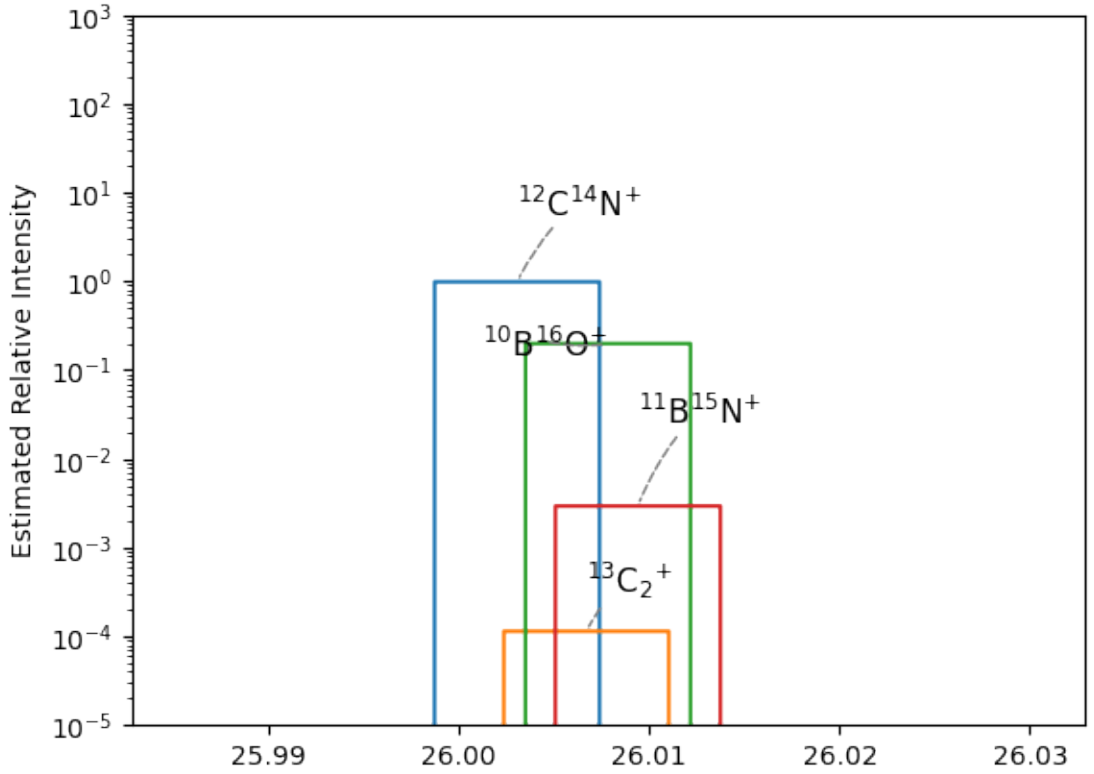
```
window = ("B[10]O[16]", 0.05)
df = build_table(["C", "B", "N", "O"], window=window, max_atoms=2)
```

Out:

```
  0%|          | 0/6 [00:00<?, ?it/s]
                    :   0%|          | 0/6 [00:00<?, ?it/s]
N-N @ 6 rows        :   0%|          | 0/6 [00:00<?, ?it/s]
C-O @ 12 rows       :   0%|          | 0/6 [00:00<?, ?it/s]
C-N @ 8 rows        :   0%|          | 0/6 [00:00<?, ?it/s]
C-C @ 6 rows        :   0%|          | 0/6 [00:00<?, ?it/s]
B-O @ 12 rows       :   0%|          | 0/6 [00:00<?, ?it/s]
B-N @ 8 rows        :   0%|          | 0/6 [00:00<?, ?it/s]
B-N @ 8 rows        : 100%|##########| 6/6 [00:00<00:00, 186.34it/s]
```

From this table, we can create our plot, limiting the labelling to the five peaks with highest estimated intensity. Note we should specify the mass resolution for the simulated peaks:

```
ax = df.mz.spectra(window=window, mass_resolution=3000, max_labels=5, figsize=(8, 4))
plt.show()
```



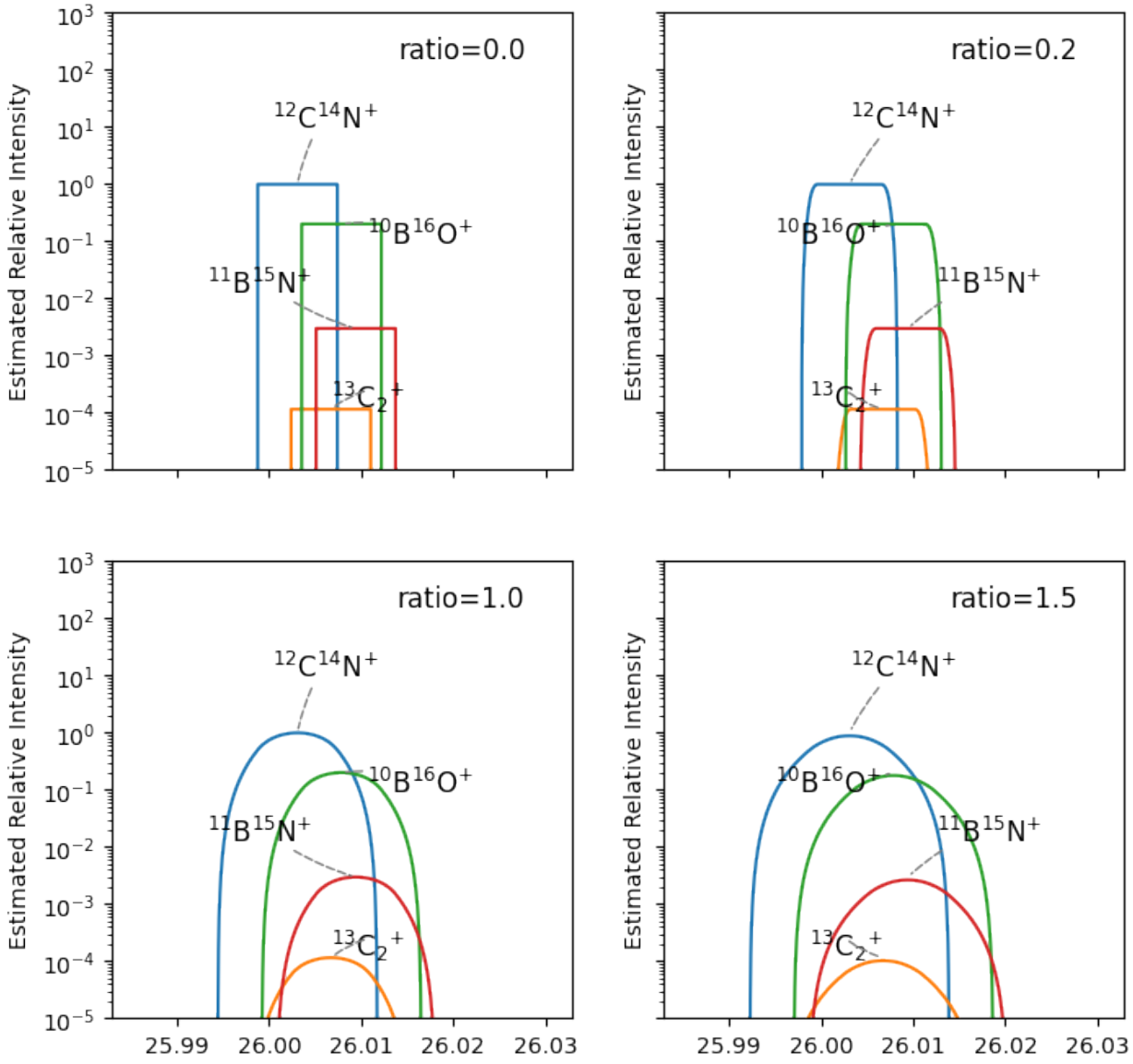
Out:

```
/home/docs/checkouts/readthedocs.org/user_builds/interferences/envs/latest/lib/python3.7/
↪site-packages/numpy/core/fromnumeric.py:43: VisibleDeprecationWarning: Creating an␣
↪ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or␣
↪ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you␣
↪must specify 'dtype=object' when creating the ndarray.
  result = getattr(asarray(obj), method)(*args, **kwds)
```

Thesse peaks better show the 'interference' aspect of these ions at relatively low mass resolution, but are notably unnaturally square. To simulate some shoulders for your peaks (i.e. a non-zero-width image of your source, and a signal limited by a collector slit or similar) you can specify a ratio for the *image_ratio* keyword argument. Here we

explore the effect this parameter with a few different values (0, <1, 1, and >1):

```
fig, ax = plt.subplots(2, 2, sharex=True, sharey=True, figsize=(8, 8))
for a, ratio in zip(ax.flat, [0, 0.2, 1, 1.5]):
    df.mz.spectra(
        ax=a, window=window, mass_resolution=3000, image_ratio=ratio, max_labels=5
    )
    a.annotate(
        "ratio={:.1f}".format(ratio),
        xy=(0.9, 0.9),
        xycoords=a.transAxes,
        ha="right",
        fontsize=12,
    )
plt.show()
```



Total running time of the script: (0 minutes 24.328 seconds)

CHAPTER

THREE

TUTORIALS

This page is home to longer examples which incorporate multiple components of `interferences` or provide examples of how to integrate these into your own workflows.

Note: This page is a work in progress. Feel free to request tutorials or examples with a feature request.

CHAPTER

FOUR

API

4.1 interferences.table

Functions for working with tables of molecular ions.

4.1.1 interferences.table.build

interferences.table.build.**build_table**(*elements=None*, *max_atoms=3*, *sortby=['m_z', 'charge', 'mass']*, *charges=[1, 2]*, *add_labels=False*, *threshold=None*, *window=None*, *cache_results=True*)

Build the interferences table.

Parameters

- **elements** (list) – List of elements to include in the table.
- **max_atoms** (int) – Largest size of molecule to build, in atoms.
- **sortby** (str | list) – Column or list of columns to sort the final table by.
- **charges** (list (int)) – Ionic charges to include in the model.
- **add_labels** (bool) – Whether to produce molecule names which are nicely formatted. This takes additional computation time.
- **threshold** (float) – Threshold for isotopic abundance for inclusion of low-abudance/non-stable isotopes.
- **mass_window** (tuple) – Window of interest to filter out irrelevant examples (here a mass window, which directly translates to m/z window with z=1).
- **cache_results** (bool) – Whether to store the results on disk for

---

Todo: Consider options for parellizing this to reduce build time. This would allow larger molecules to be included.

Invalid molecules (e.g. *H{2+}*) will currently be present, but will ideally be filtered out

In some cases, mass peaks will be duplicated, and we want to keep the simplest version (e.g. *Ar[40]+* and *Ar[40]2{2+}*). We here remove duplicate mass peaks before sorting (i.e. take the first one, as higher charges would be penalised), but we could potentially add a check that both contain the same isotopic components for verificaiton (this would be slow..).

While "m/z" would be an appropriate column name, it can't be used in HDF indexes.

4.1.2 interferences.table.combinations

Functions for calculating combinations (in the combinatorics sense) of elements and isotopes into isotope-specified molecular ions.

interferences.table.combinations.**get_elemental_combinations**(*elements*, *max_atoms=3*)

Combine a list of elements into lists of molecular combinations up to a maximum number of atoms per molecule. Successively adds smaller molecules until down to single atoms.

Parameters

- **elements** (list) – Elements or isotopes to combine into molecules.
- **max_atoms** (int) – Maximum number of atoms per molecule. This limits the number of molecules returned to the generally most relevant simple molecules.

Todo: Check that isotopes supplied to this function are propogated

---

interferences.table.combinations.**get_isotopic_combinations**(*element_comb*, *threshold=None*)

Take a combination of elements and expand it to generate the potential combinations of elements.

Parameters

- **element_comb** (list) – List of elements for which to combine lists of isotopes.
- **threshold** (float) – Threshold below which to ignore low-abundance isotopes.

Return type list

interferences.table.combinations.**component_subtable**(*components*, *charges=[1, 2]*, *threshold=None*)

Build a sub-table from a set of elemental components.

Parameters

- **components** (list) – List of elements to combine in the subtable.
- **charges** (list (int)) – Ionic charges to include in the model.
- **threshold** (float) – Threshold for isotopic abundance for inclusion of low-abudance/non-stable isotopes.

Return type pandas.DataFrame

4.1.3 interferences.table.intensity

Functions to threshold, combine and estimate intensities of elements and isotopes based on their abundances.

interferences.table.intensity.**isotope_abundance_threshold**(*isotopes*, *threshold=None*)

Remove isotopes from a list which have no or zero abundance.

Parameters

- **isotopes** (list) – List of isotopes to filter.
- **threshold** (float) – Minimum isotope abundance for inclusion.

Return type list

interferences.table.intensity.**get_isotopic_abundance_product**(*components*)

Estimates the abundance of a molecule based on the abundance of the isotopic components.

Return type float

Notes

This is essentially a simplistic activity model. Isotopic abundances from periodictable are in %, and are hence divded by 100 here.

4.1.4 interferences.table.molecules

Functions for creating, formatting and serialising representaitons of molecules.

interferences.table.molecules.**components_from_index_value**(*idx*)

interferences.table.molecules.**deduplicate**(*df*, *charges=None*, *multiples=True*)

De-duplicate a dataframe index based on index values and and molecule-multiples.

Parameters

- **df** (pandas.DataFrame) – Dataframe to check the index of.
- **charges** (list) – List of valid charges for the frame.
- **multiples** (bool) – Whether to remove molecule-multiples.

Return type pandas.DataFrame

interferences.table.molecules.**repr_formula**(*molecule*)

Get a string representation of a formula which preserves element and isotope information.

interferences.table.molecules.**get_formatted_formula**(*molecule*, *sorted=False*)

Construct a formatted name for a molecule.

Parameters

- **molecule** (Formula) – Molecule to name.
- **sorted** (bool) – Whether a molecular formula is already sorted, so sorting can be skipped.

Return type str

interferences.table.molecules.**get_molecule_labels**(*df*, ***kwargs*)

Get labels for molecules based on their composition and charge.

Parameters **df** (pandas.DataFrame)

Return type pandas.Series

interferences.table.molecules.**molecule_from_components**(*components*)

Builds a Formula from a list of atom or isotope components.

Parameters **components** (list) – Atomic, isotope or molecular components to construct an ionic molecule from.

Return type Formula

Todo:

- Modify to accept consumption of molecular components (e.g. Fe2O3+)

See also:

pyrolite.mineral.transform.merge_formulae()

4.1.5 interferences.table.store

interferences.table.store.**load_store**(*path=None*, *complevel=4*, *complib='lzo'*, ***kwargs*)

Load the interferences HDF store.

Parameters

- **path** (str | pathlib.Path) – Path to the store.
- **complevel** (int) – Compression level option for the HDF store. Uncompressed tables can easily reach a few hundred MB - this isn't an issue on a local disk, but can be limiting for web transfer.
- **complib** (str) – Which compression library to use.

Return type pandas.HDFStore

interferences.table.store.**lookup_components**(*identifier*, *path=None*, *key='table'*, *window=None*, ***kwargs*)

Look up a a list of components from the store based on their identifiers.

Parameters

- **identifiers** (str) – Identifiers for the components to look up.
- **path** (str | pathlib.Path) – Path to store to search.
- **key** (str) – Key for the table within the store.
- **window** (tuple) – Window for indexing along m/z to return a subset of results.
- **drop_first_level** (bool) – Whether to drop the first level of the index for simplicity.

Return type pandas.DataFrame

interferences.table.store.**get_store_index**(*path*, *drop_first_level=True*, ***kwargs*)

interferences.table.store.**process_subtables**(*dfs*, *charges=None*, *dump=True*, *path=None*, *mode='a'*, *data_columns=['elements', 'm_z', 'iso_abund_product']*, *complevel=4*, *complib='lzo'*, ***kwargs*)

Process and optionally dump a set of subtables to file, appending to the hierarchically-indexed table.

Parameters

- **dfs** (list`(:class:`pandas.DataFrame)) – Dataframes to dump.
- **charges** (list) – Charges used to create for the table.
- **path** (str | pathlib.Path) – Path to the file to add the table to.
- **mode** (str) – Mode for accessing the HDF file.
- **data_columns** (list) – List of columns to create an indexes for to allow query-by-data.
- **complevel** (int) – Compression level option for the HDF store. Uncompressed tables can easily reach a few hundred MB - this isn't an issue on a local disk, but can be limiting for web transfer.
- **complib** (str) – Which compression library to use.

Returns De-duplicated concatenated version of new tables.

Return type pandas.DataFrame

interferences.table.store.**reset_table**(*path=None*, *remove=True*, *key='table'*, *format='table'*, *complevel=4*, *complib='lzo'*, ***kwargs*)

Reset or remove a HDF store.

Parameters
- **path** (str | pathlib.Path) – Path to store.
- **remove** (bool) – Whether to remove the table from disk, if possible.
- **format** (str) – Format to set for the new tables.
- **complevel** (int) – Compression level option for the HDF store. Uncompressed tables can easily reach a few hundred MB - this isn't an issue on a local disk, but can be limiting for web transfer.
- **complib** (str) – Which compression library to use.

4.2 interferences.plot

Submodule for visualisation and visual interrogation of mass spectra and potential interference patterns.

4.2.1 interferences.plot.ptable

interferences.util.ptable.**get_periodic_frame**()

Construct a simple periodic table dataframe organised by group and row. Note that the lanthanides and actinides are each found in a single cell.

Return type pandas.DataFrame

4.2.2 interferences.plot.ptable

Visualisation using of a periodic table.

4.3 interferences.util

4.3.1 interferences.util.sorting

interferences.util.sorting.**get_first_atom**(*molecule*)

Get the first atom in a molecular formula.

Parameters **molecule** (Element | Formula) – Molecule to check.

Returns Element or isotope.

Return type Element

interferences.util.sorting.**get_relative_electronegativity**(*element*, *reverse=True*)

Get an index of the relative electronegativity of an element, for use in sorting elements (e.g. for chemical formulae). If a list of elements is supplied, a list will be returned.

Parameters **element** (str | periodictable.core.Element | list)

Return type int | list

Note: Electronegativity check uses numbers as these are provided by both Element and Isotope objects.

4.3.2 interferences.util.mz

interferences.util.mz.**process_window**(*window*)

Process the two allowable verions of a mass window (element/isotope and width, or a low- and high-mass).

Parameters **window** (tuple) – Window parameters to process.

Return type tuple

4.3.3 interferences.util.ptable

interferences.util.ptable.**get_periodic_frame**()

Construct a simple periodic table dataframe organised by group and row. Note that the lanthanides and actinides are each found in a single cell.

Return type pandas.DataFrame

4.3.4 interferences.util.meta

interferences.util.meta.**interferences_datafolder**(*subfolder=None*)

Returns the path of the interferences data folder.

Parameters **subfolder** (str) – Subfolder within the interferences data folder.

Return type pathlib.Path

4.3.5 interferences.util.log

interferences.util.log.**Handle**(*logger*, *handler=<NullHandler (NOTSET)>*, *formatter='%(asctime)s %(name)s - %(levelname)s: %(message)s'*, *level=None*)

Handle a logger with a standardised formatting.

Parameters

- **logger** (logging.Logger | str) – Logger or module name to source a logger from.
- **handler** (logging.Handler) – Handler for the logging messages.
- **formatter** (str | logging.Formatter) – Formatter for the logging handler. Strings will be passed to the logging.Formatter constructor.
- **level** (str) – Logging level for the handler.

Returns Configured logger.

Return type logging.Logger

CHAPTER

FIVE

DEVELOPMENT

5.1 Development History and Planning

- Changelog

5.2 Contributing

- Contributing
- Contributors
- Code of Conduct

5.3 Development Installation

To access and use the development version, you can either clone the repository or install via pip directly from GitHub:

```
pip install git+git://github.com/morganjwilliams/interferences.git@develop
↪#egg=interferences
```

5.4 Tests

If you clone the source repository, unit tests can be run using pytest from the root directory after installation with development dependencies (`pip install -e .[dev]`):

```
python setup.py test
```

If instead you only want to test a subset, you can call `pytest` directly from within the interferences repository:

```
pytest ./test/<path to test or test folder>
```

CHAPTER

SIX

CHANGELOG

All notable changes to this project will be documented here.

6.1 Development

Note: Changes noted in this subsection are to be released in the next version. If you're keen to check something out before its released, you can use a development install.

CHAPTER

SEVEN

CODE OF CONDUCT

7.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

7.2 Our Standards

Examples of behaviour that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behaviour by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

7.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behaviour and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behaviour.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviours that they deem inappropriate, threatening, offensive, or harmful.

7.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

7.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behaviour may be reported by contacting the project admins. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

7.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant Version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html. The Contributor Covenant is released under the Creative Commons Attribution 4.0 License.

For answers to common questions about this code of conduct, see https://www.contributor-covenant.org/faq.

CHAPTER

EIGHT

CONTRIBUTING

The long-term aim of this project is to be designed, built and supported by (and for) the geochemistry community. Requests for features and bug reports are particularly valuable contributions, in addition to code and expanding the documentation. All individuals contributing to the project are expected to follow the Code of Conduct, which outlines community expectations and responsibilities.

Also, be sure to add your name or GitHub username to the contributors list.

Note: This project is currently in *beta*, and as such there's much work to be done.

8.1 Feature Requests

If you're new to Python, and want to implement a specific process, plot or framework as part of `interferences`, you can submit a Feature Request. Perhaps also check the Issues Board first to see if someone else has suggested something similar (or if something is in development), and comment there.

8.2 Bug Reports

If you've tried to do something with `interferences`, but it didn't work, and googling error messages didn't help (or, if the error messages are full of `interferences.XX.xx`), you can submit a Bug Report . Perhaps also check the Issues Board first to see if someone else is having the same issue, and comment there.

8.3 Contributing to Documentation

The documentation and examples for `interferences` are gradually being developed, and any contributions or corrections would be greatly appreciated. Currently the examples are patchy, and any 'getting started' guides would be a helpful addition.

These pages serve multiple purposes:

- A human-readable reference of the source code (compiled from docstrings).
- A set of simple examples to demonstrate use and utility.
- A place for developing extended examples

8.4 Contributing Code

Code contributions are always welcome, whether it be small modifications or entire features. As the project gains momentum, check the Issues Board for outstanding issues, features under development. If you'd like to contribute, but you're not so experienced with Python, look for `good first issue` tags or email the maintainer for suggestions.

To contribute code, the place to start will be forking the source for `interferences` from GitHub. Once forked, clone a local copy and from the repository directory you can install a development (editable) copy via `python setup.py develop`. To incorporate suggested changes back to into the project, push your changes to your remote fork, and then submit a pull request onto interferences/develop or a relevant feature branch.

Note:

- See Installation for directions for installing extra dependencies for development, and Development for information on development environments and tests.
- `interferences` development roughly follows a gitflow workflow. `interferences/master` is only used for releases, and large separable features should be build on `feature` branches off `develop`.
- Contributions introducing new functions, classes or entire features should also include appropriate tests where possible (see *Writing Tests*, below).
- `interferences` uses Black for code formatting, and submissions which have passed through `Black` are appreciated, although not critical.

8.5 Writing Tests

There is currently a minimal unit test suite for `interferences`, which guards against breaking changes and assures baseline functionality. `interferences` uses continuous integration via Travis, where the full suite of tests are run for each commit and pull request, and test coverage output to Coveralls.

Adding or expanding tests is a helpful way to ensure `interferences` does what is meant to, and does it reproducibly. The unit test suite one critical component of the package, and necessary to enable sufficient trust to use `interferences` for scientific purposes.

CHAPTER

NINE

CONTRIBUTORS

This list includes people who have contributed to the project in the form of code, comments, testing, bug reports, or feature requests.

- Morgan Williams

Note: This documentation is a work in progress and is updated regularly. Contact the maintainer with any specific questions/requests.

PYTHON MODULE INDEX

i

INDEX

P

R