IoCor Documentation

Release 2.2.2

Baudoin Delépine, Matthieu Guionnet, Pierre Millard

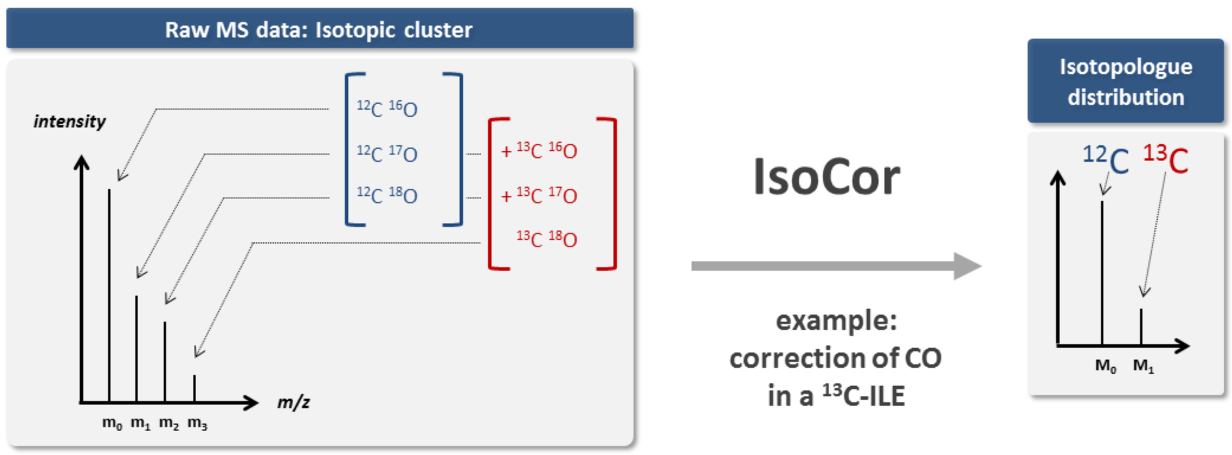
Nov 02, 2023

USAGE

CHAPTER

ONE

WELCOME TO ISOCOR DOCUMENTATION!



IsoCor is a scientific software dedicated to the correction of mass spectrometry (MS) data for naturally occuring isotopes. IsoCor corrects raw MS data (*mass fractions*) for naturally-occurring isotopes of all elements and purity of the *isotopic tracer*. The output of IsoCor is the *isotopologue distribution* (i.e. the relative fractions of molecular entities differing only in the number of isotopic substitutions of the tracer) of the molecule. IsoCor also calculates the mean enrichment (i.e. the mean isotopic content in the molecule) in metabolites.

It is one of the routine tools that we use at the MetaSys team and MetaToul platform in isotopic studies of metabolic systems.

The code is open-source, and available on GitHub under a *GPLv3 license*.

This documentation is available on Read the Docs (https://isocor.readthedocs.io) and can be downloaded as a PDF file.

Key features

- **correction of naturally occuring isotopes**, both for non-tracer and tracer elements,
- **correction of tracer purity**,
- shipped as a library with both a **graphical and command line interface**,
- mass-spectrometer and *resolution* agnostic,
- can be applied to singly- and multiply-charged ions
- can be used with any tracer element (having two or more isotopes)
- account for the contribution of derivatization steps (if any),
- generate InChIs of isotopically-resolved (tracer) isotopologues,

- open-source, free and easy to install everywhere where Python 3 and pip run,
- biologist-friendly.

See also:

We strongly encourage you to read the *Tutorials* before using IsoCor.

1.1 Quick start

1.1.1 Installation

IsoCor requires Python 3.5 or higher. If you do not have a Python environment configured on your computer, we recommend that you follow the instructions from Anaconda.

Then, open a terminal (e.g. run *Anaconda Prompt* if you have installed Anaconda) and type:

```
pip install isocor
```

You are now ready to start IsoCor.

If this method does not work, you should ask your local system administrator or the IT department "how to install a Python 3 package from PyPi" on your computer.

Alternatives & update

If you know that you do not have permission to install software systemwide, you can install IsoCor into your user directory using the `--user` flag:

```
pip install --user isocor
```

If you already have a previous version of IsoCor installed, you can upgrade it to the latest version with:

```
pip install --upgrade isocor
```

Alternatively, you can also download all sources in a tarball from GitHub, but it will be more difficult to update IsoCor later on.

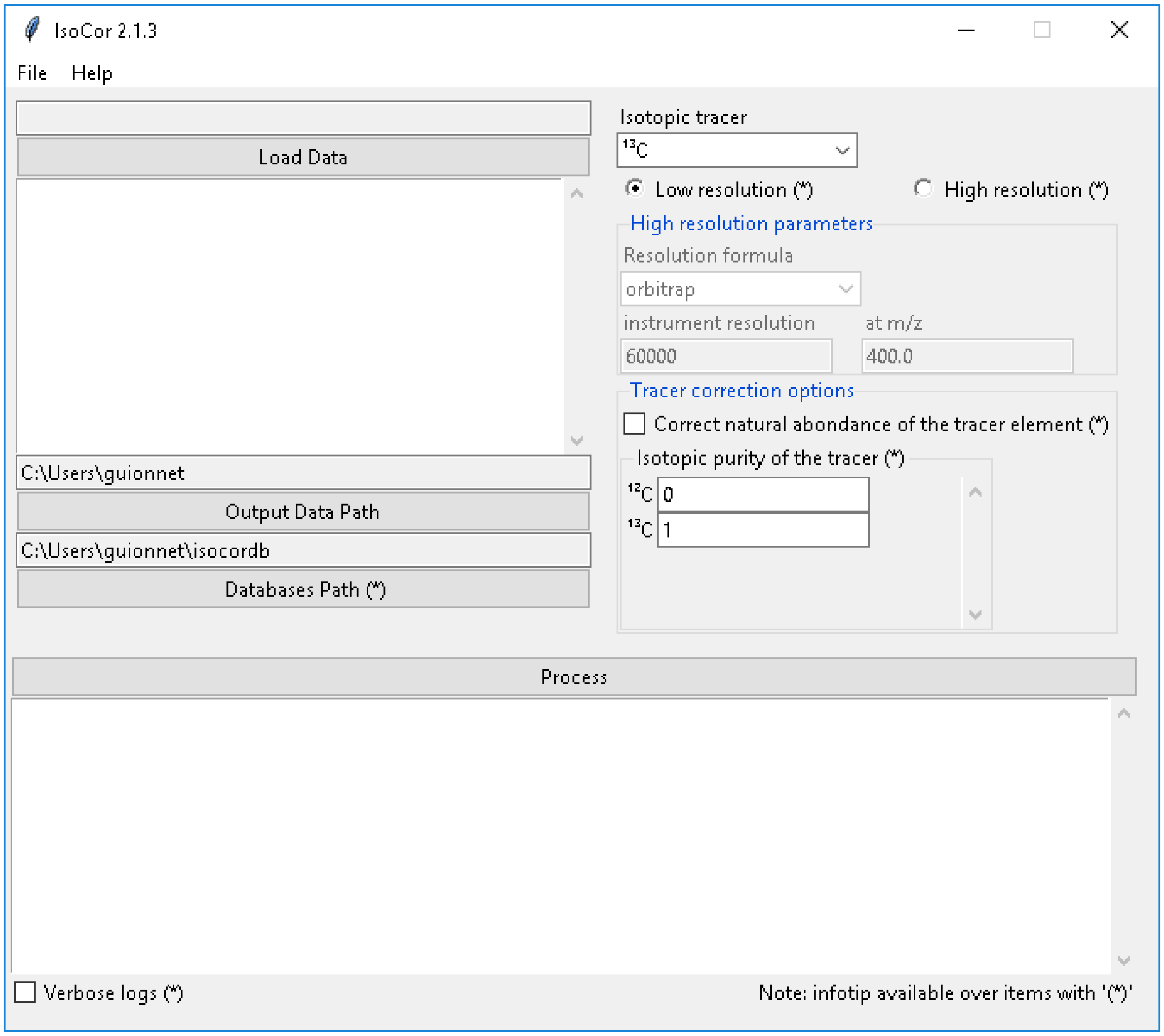
1.1.2 Usage

Graphical User Interface

To start the Graphical User Interface, type in a terminal (Windows: *Anaconda Prompt*):

```
isocor
```

The IsoCor window will open. If the window fails to open, have a look at our *dedicated troubleshooting procedure* to solve the problem.



Select the measurements file, modify the correction parameters (isotopic tracer, resolution, etc) according to your experiment, and click on `Process`. IsoCor proceeds automatically to the corrections and display its progress and important messages.

Warning: The correction options must be carefully selected to ensure reliable interpretations of labeling data, as detailed in the *Tutorials*.

The output of the calculations (i.e. isotopologue distributions) will be written in a text file along a log file.

Note: IsoCor silently overwrites (results and log) files if they already exist. So take care to copy your results elsewhere if you want to protect them from overwriting.

See also:

Tutorial *First time using IsoCor* has example data that you can use to test your installation.

Command Line Interface

To process your data, type in a terminal:

```
isocorcli [command line options]
```

Here after the available options with their full names are enumerated and detailed.

```
usage: isocorcli [-h] [-M M] [-D D] [-I I] -t TRACER [-r RESOLUTION]
                 [-m MZ_OF_RESOLUTION]
                 [-f {orbitrap,ft-icr,constant,datafile}] [-p TRACER_PURITY]
                 [-n] [-v]
                 inputdata
```

Positional Arguments

inputdata	measurements file to process

Named Arguments

-M	path to metabolites database
-D	path to derivatives database
-I	path to isotopes database
-t, --tracer	the isotopic tracer (e.g. “13C”)
-r, --resolution	HR only: resolution of the mass spectrometer (e.g. “1e4”)
-m, --mz_of_resolution	HR only: mz at which resolution is given (e.g. “400”)
-f, --resolution_formula_code	Possible choices: orbitrap, ft-icr, constant, datafile HR only: spectrometer formula code
-p, --tracer_purity	purity vector of the tracer
-n, --correct_NA_tracer	flag to correct tracer natural abundance
-v, --verbose	flag to enable verbose logs

IsoCor proceeds automatically to the corrections and display its progress and important messages.

Warning: The correction options must be carefully selected to ensure reliable interpretations of labeling data, as detailed in the *Tutorials*.

See also:

Tutorial *First time using IsoCor* has example data that you can use to test your installation.

Library

IsoCor is also available as a library (a Python module) that you can import directly in your Python scripts:

```
import isocor
```

See also:

Have a look at our *library showcase* if you are interested into this experimental feature.

1.2 Tutorials

See also:

If you have a question that is not covered in the tutorials, have a look at the *Frequently asked questions*.

1.2.1 First time using IsoCor

Input data

IsoCor takes as input the raw MS data, i.e. *mass fractions* of the *isotopic cluster*, to calculate the corresponding *isotopologue distribution*, hence providing quantitative information on the incorporation of labeling into metabolites.

The raw MS data and the information required to perform the correction (i.e. the natural abundance and exact mass of isotopes, and a list of metabolites and derivatives moieties with their elemental formulas) are provided in flat text files.

At first start, IsoCor creates in the user directory a IsoCor data folder *isocordb* containing default database files. These files can be edited and implemented according to the user's needs, as detailed below. Different database files can also be created (e.g. to have specific project-related databases), as detailed below.

Measurements file

This file contains the raw MS data for each metabolite of each sample, i.e. the *mass fractions* of the measured *isotopic cluster* that contain information on the *tracer isotopologues*. For each metabolite you should always measure $n + 1$ mass fractions, where n is the number of atoms of the tracer element in the metabolite.

The measurement file is a TSV file with one row by *isotopologue* and the following columns:

sample
The sample name (optional); e.g. "Cloverfield 10".

metabolite
The metabolite name that represents the metabolite moiety, as it is referred in the metabolite database (*metabolites.dat*); e.g. "PEP".

derivative
The derivative name (optional) that represents the derivative moiety, as it is referred in the derivative database (*derivatives.dat*); e.g. "TMS".

isotopologue
The index of the peak measured, as an integer; e.g. '0' for the M0 peak that does not have any mass shift.

area
The measured *mass fractions*; e.g. "4242.42".

resolution (optional)
The MS resolution of the corresponding *mass fractions*; e.g. "60000". Note the *all* mass fractions of a given isotopic cluster must have the same resolution.

`Example file`.

Note: An example file is provided with IsoCor. It is created at the first run of IsoCor in your user directory (*<youruserdirectory>/isocordb/Data_example.tsv*).

About derivatives

The derivative field is optional and should be declared only if:

1. a derivatization step was performed before MS analysis,
2. some atoms of the derivative remains in the molecular entity that gives rise to measured *isotopic cluster*.

See also:

Declaration of elemental formulas: "metabolite" and "derivative" moieties

Database files

The exact mass and natural abundance of each isotope and the elemental formulas used for correction have to be defined carefully, otherwise the correction will be wrong.

IsoCor rely on several flat-files to store this information. Pre-configured files are shipped with IsoCor and created at the first run of IsoCor. Those database should be modified according to the user needs. They are **located in IsoCor data directory**, in user main directory: *<youruserdirectory>/isocordb/*.

Note: IsoCor is case sensitive; i.e. two metabolites or derivatives with the same name but different cases will be considered as two distinct entities.

Isotopes database (Isotopes.dat)

This file stores **the exact mass and natural abundance of all stable isotopes of each element**, given as relative fractions.

It is a TSV file with one row by isotope and the following columns:

element
The element symbol of the isotope; e.g. "C".

mass
The exact mass of this isotope; e.g. "13.003354835" for ^{13}C.

abundance
The relative abundance of this isotope normalized to 1; e.g. "0.0107" for ^{13}C.

`Example file`.

A pre-configured isotopes database can be found in IsoCor data directory and should be edited according to the users needs. It is located in user main directory at *<youruserdirectory>/isocordb/Isotopes.dat*.

Warning: The isotopes database is always loaded from IsoCor data directory, i.e. from *<youruserdirectory>/isocordb/Isotopes.dat*.

Note: All elements should be declared, including elements with only one isotope (with its abundance set to 1). This is required for accurate correction of high-resolution data.

Note: For elements with gaps in the list of nominal mass of isotopes (e.g. for sulfur with isotopes ^{33}S, ^{34}S, ^{36}S, but not ^{35}S), declare the missing isotope(s), with the exact mass set at the missing integer(s), and an abundance of 0 (as done in the example file for sulfur).

Metabolites database (Metabolites.dat)

This file stores **elemental formulas of the metabolites**.

It is a TSV file with the following columns:

name
: Metabolite name or abbreviation; e.g. "pyruvic acid" or "PYR".

formula
: Elemental formula of the metabolite moiety of the molecular entity that gives rise to the measured *isotopic cluster*; e.g. "$C_3H_4O_3$". See also *Declaration of elemental formulas: metabolite and derivative moieties*.

charge
: Charge state of the detected ion; e.g. "-1" for singly-charge ions or "-2" for doubly-charge ions.

inchi
: InChI (may refer to the metabolite, the detected ion, or any other chemical substance); e.g. "InChI=1S/C4H4O4/c5-3(6)1-2-4(7)8/h1-2H,(H,5,6)(H,7,8)/p-2/b2-1+" for fumarate. This field is optional.

`Example file`.

A pre-configured metabolites database can be found in IsoCor data directory and should be edited according to the users needs. It is located in user main directory at *<youruserdirectory>/isocordb/Metabolites.dat*.

Derivatives database (Derivatives.dat)

This file stores **elemental formulas of chemical derivatives** that have to be considered for the isotopic correction of metabolites derivatized prior to MS analysis.

It is a TSV file with the following columns:

name
: Derivative name or abbrevation; e.g. "t-butyldimethyl-silylation" or "M-57".

formula
: Elemental formula of the derivative moiety of the molecular entity that gives rise to the measured *isotopic cluster*; e.g. "$Si_2C_8H_{21}$". See also *Declaration of elemental formulas: metabolite and derivative moieties*.

`Example file`.

A pre-configured derivatives database can be found in IsoCor data directory and should be edited according to the users needs. It is located in the user main directory at *<youruserdirectory>/isocordb/Derivatives.dat*.

Custom databases

IsoCor data directory is created at the first run of IsoCor with pre-configured databases files in the user main directory (*<youruserdirectory>/isocordb/*). These files should be edited according to the users needs, e.g. to add some metabolites and derivatives formulas.

Alternatively, users can select at runtime a custom folder from which metabolites and derivatives will be loaded ('Metabolites.dat' and 'Derivatives.dat') with the 'Databases Path' button. It is especially useful to define project-based database files.

Warning: Importantly, 'Isotopes.dat' is always loaded from IsoCor data directory ('<youruserdirectory>/isocordb/Isotopes.dat') and will not be loaded from a custom databases folder.

Correction parameters

IsoCor provides several options to adapt to many situations that can be encountered in terms of isotopic tracer, sample processing, *resolution* of the MS analyzer, etc.

Measurements file
Path to the *Measurements file*.

Isotopic tracer
The tracer used for your experiment. Available tracers are imported from *isotopes.dat* database file.

Resolution
Resolution of the MS analyzer.

Resolution measured at
m/z at which the *resolution* is given.

Resolution formula
The relationship between the operating *resolution* and the resolution at m/z of the measured metabolite moiety depends on the MS analyzer, which has to be selected. If 'datafile' is selected, resolution should be provided for all mass fractions in the measurements file.

Tracer purity
Correct for the presence of unlabeled atoms at labeled positions, using the relative abundance of each isotope of the tracer element at labeled positions. Default is to assume a perfect purity (i.e. tracer isotope=1).

Correct natural abundance of the tracer element
Correct for natural abundance of the tracer element at unlabeled positions. Default is no correction.

Output data path
Path to the *Output files*. A log file with the same name will be created in the same directory, with a '.log' extension.

Verbose logs
If set, the log-file will contain all information necessary to check intermediate results of the correction process.

See also:

Tutorial: *Isotopic purity and natural abundance of the tracer*.

Output files

Result file

The result file is a TSV file with the following columns:

sample
: Name of the sample, as it was provided in the *Measurements file*.

metabolite
: Name of the metabolite, as it was provided in the *Measurements file*.

derivative
: Name of the derivative, as it was provided in the *Measurements file*.

isotopologue
: The index of the peak measured, as an integer; e.g. '0' for the M_0 peak that does not have any mass shift, as it was provided in the *Measurements file*.

isotopic_inchi
: Isotopic InChI of the corresponding tracer isotopologue (or just the isotopic layer if no InChI has been provided in the *Metabolites database (Metabolites.dat)* file), as detailed *here*; e.g. with isotopic layer '/a(C1+1),(C3+0)' for the M_1 ^{13}C-isotopologue of fumarate.

area
: The measured peak intensity; e.g. '42.5', as it was provided in the *Measurements file*.

corrected_area
: The corrected area.

isotopologue_fraction
: The abundance of each *isotopologue* (corrected area normalized to 1).

residuum
: Residuum of the fit (difference between experimental and optimal isotopologue distribution, normalized to 1).

mean_enrichment
: Mean molecular content in isotopic tracer in the metabolite.

Log file

A log file is created in the same directory as the Result file to store correction parameters (for reproducibility), with a '.log' extension.

Extensive information on the correction process (correction vector, correction matrix, intermediary results, etc.) can be found in the log file if 'Verbose logs' option has been checked.

Warning and error messages

Error messages are explicit. You should examine carefully any warning/error message. After correcting the problem, you might have to restart IsoCor (to reload databases files) and perform the correction again.

1.2.2 Declaration of elemental formulas: metabolite and derivative moieties

This section provides guidelines for the definition of elemental formulas of "metabolite" and "derivative" moieties. It also provides representative examples to cover a large panel of MS and MS/MS methods dedicated to quantitative isotopic analysis.

What is in the elemental formula

Elemental formulas must be defined according to the molecular entity that gives rise to the measured *isotopic cluster*. It may correspond (but not necessarily) to the elemental formula of the detected ion.

For instance, in the following situations, the formulas should include:

- for MS measurements: all atoms of the detected ion
- for MS/MS measurements, with all tracer atoms in the detected ion: only atoms of the detected ion
- for MS/MS measurements, with no tracer atoms in the detected ion: only atoms of the complement (neutral fragment)

Metabolite vs. derivative formulas

All atoms of the molecular entity that gives rise to the measured *isotopic cluster* **should be declared strictly once in a formula, either as a "metabolite" or a "derivative" moiety.**

Atoms that originate from the metabolite should be declared in the file "*metabolites.dat*", and atoms that originate from the derivative (if any) should be declared in the file "*derivatives.dat*".

A derivative moiety should thus be declared only if a derivatization step was performed before MS analysis. Importantly, we consider that *the derivative moiety do not contain any tracer atom*. Therefore, all its atoms (including atoms of the tracer element) are expected to be at natural isotope abundance and will be corrected as such. This is obviously not the case for the metabolite moiety that do incorporate tracer atoms and is thus corrected differently. It follows that, to ensure the accurate correction of the measured *isotopic cluster*, the atoms originated from the derivative moiety must be declared separately from those originated from the metabolite moiety (respectively into *derivatives.dat* and *metabolites.dat*).

Example 1 - MS analysis: Pyruvate

Pyruvic acid ($C_3H_4O_3$) can be analyzed by LC-MS using multiple ion monitoring (MIM) in the negative mode, and the measured *isotopic cluster* originates from the molecular ion $[C_3H_3O_3]^-$, then the formula to use for correction is $C_3H_3O_3$. This formula must be set into *metabolites.dat* and referred to by its associated name into the measurements file.

Example 2 - MS/MS analysis, with no tracer atoms in the detected ion: PEP

Phosphoenolpyruvate (PEP) can be analyzed using the MS/MS method developed by Kiefer et al. (2007). The fragmentation of phosphorylated metabolites results in the efficient release of $[PO_3]^-$ or $[H_2PO_4]^-$ ions, allowing highly sensitive measurement of *isotopologue distributions* in these compounds in the multiple reaction monitoring (MRM) mode. This is achieved by selecting MRM transitions in which phosphate ions are detected but which encode the *isotopic cluster* of the complement, i.e., the part of the molecule that remains after loss of the phosphate ion that is actually detected. In the case of PEP ($C_3H_5O_6P$), for which the molecular ion that is analyzed is $[C_3H_4O_6P]^-$, the analysis is based on MRM transitions in which $[PO_3]^-$ ions are used, meaning that the *isotopic cluster* is actually measured for the complement fragment $C_3H_4O_3$. Hence, the formula to enter in *metabolites.dat* is $C_3H_4O_3$.

Example 3 - MS analysis of derivatized metabolites with in source fragmentation, with all tracer atoms in the detected ion: TBDMS-derivatized Alanine

Alanine ($C_3H_7O_2N$) can be analyzed by GC-MS after t-butyldimethyl-silylation (TBDMS derivatization). A fragment that is classically used for ^{13}C-metabolic flux analysis is the ‘M-57’ fragment that contains all atoms the compound of interest and two TBDMS groups, one of which lose the fragment $[C_4H_9]$. The elemental formula of the two TBDMS groups excluding the latter fragment (i.e. $[Si_2C_8H_{21}]$) must be declared into *derivatives.dat* since it will be present in the molecular entity that gives rise to the measured *isotopic cluster*. Meanwhile, the elemental composition of the alanine moiety of the detected ion (i.e. $[C_3H_5O_2N]$) must be declared as the “metabolite moiety”, thus into *metabolites.dat*.

Example 4 - MS/MS analysis, with all tracer atoms in the detected ion

In this situation where the fragment ion which is detected gives rise to the measured *isotopic cluster*, the elemental formula to declare in IsoCor is the formula of the fragment ion. Atoms of the fragment that originate from the metabolite should be declared into *metabolites.dat*, and atoms that originate from the derivative should be declared into *derivatives.dat*.

1.2.3 Resolution of the MS analyzer

This section provides guidelines to account for the *resolution* of the MS analyzer.

Low-resolution

For low *resolution* datasets collected at unitary resolution (i.e. typically R<1000), select “Low resolution”.

High-resolution

For high *resolution* datasets, accurate correction requires to know the resolution of the MS analyzer at the particular m/z of the molecular entity that gives rise to the experimental *isotopic cluster*. It is used to identify the correct set of isotopic species that overlap with the masses of the tracer isotopologues in the *isotopic cluster*, and ultimately remove their contribution.

Typically, the *resolution* of the MS analyzer is given at a specific m/z (defined during instrument calibration). IsoCor estimates the resolution at the appropriate m/z, provided this relationship is known. This relationship depends on each instrument and was implemented for FT-ICR and Orbitrap analyzers.

We have also implemented an option to set a “constant resolution”, i.e. which is considered to be independent of the m/z.

Finally, the option “datafile” allows users to provide resolution of each mass fraction directly in the measurements file. Note that resolution must be the same for *all* peaks of a given isotopic cluster.

Note: If you want to use IsoCor with a high-resolution MS instrument that is not currently supported (and for which you have the mathematical relationship to calculate the *resolution* at a given m/z from the resolution at the calibration mass), please contact us.

1.2.4 Isotopic purity and natural abundance of the tracer

IsoCor provides options to correct (or not) for isotopic purity of the tracer and natural abundance of the tracer elements. Ideally, you should correct the data for both isotopic purity of the tracer and natural abundance of the tracer elements. By doing so, the output data will readily reflect the incorporation of labeling and will be comparable between metabolites.

However, this is not always possible (e.g. if the isotopic purity is not known it cannot be corrected), nor desirable (e.g. if a tool downstream in your analysis pipeline will force you to perform some corrections). In the end, the correction options must always be taken into account when interpreting the data so you should choose them carefully.

Warning: The choice to correct isotopic purity and/or natural abundance of the tracer is absolutely critical for accurate interpretations of the output data (isotopologues distributions)!

Isotopic purity of the tracer

Labelled substrates are not isotopically pure, i.e. they are not 100 % enriched at the 'labelled' position(s). The latter contain small fractions of non-tracer isotopes for which MS data must be corrected. To do so, the fractions of each isotope into the 'labelled' positions must be provided. For example, if the content in ^{13}C atoms in each position of a U-^{13}C-labeled compound is 99 %, other 1 % being ^{12}C atoms, the purity must be entered as *12C=0.01* and *13C=0.99*.

Note: If you do not want to correct *isotopic clusters* for the isotopic purity of the substrate, or if you do not know it, just let the default value (purity = 1).

Warning: Tracer purity correction is only valid if *all* the labelled positions of the substrate(s) have the same isotopic purity. It should be checked from the manufacturers or determined experimentally.

When different labeled substrates are mixed, tracer purity correction also requires that all their labeled positions have the same isotopic purity.

Example: Unknown purity

If the purity of the label input(s) is not known you will not be able to correct it, despite the fact that it could be significant. Therefore, you should take special care in the interpretation of mean enrichment which will be overestimated.

Example: Several inputs with distinct purity

If two or more labeled inputs have highly different isotopic purity you will not be able to correct it properly. Therefore, you should take special care in the interpretation of mean enrichment.

Natural abundance of the tracer

When the label input is not uniformly labelled, it contains ‘unlabelled’ positions in which the tracer isotope is usually occurring at its natural abundance. The MS data can be corrected for the contribution of these naturally occurring isotopes.

Warning: Correction for natural abundance of the tracer element is only valid when the isotopes of the tracer element occur at natural abundance into the unlabeled positions of the input substrate(s). It is typically the case but should be checked from the manufacturer or determined experimentally.

Example: Natural abundance and downstream analysis

You must be aware of the corrections performed by downstream analysis tools and make sure that you do not correct something twice.

In a ^{13}C-metabolic flux analysis experiment, *if the raw data has already been corrected for natural abundance of the tracer element*, the unlabeled position(s) of all carbon sources must be declared as unlabeled with a perfect purity when calculating fluxes (e.g. CO_2 input should be declared as: *12C=1.0*), which might be counter-intuitive since you knew they were at natural abundance.

In contrast, *if the raw data was not corrected for natural abundance of the tracer element*, the unlabeled position(s) of all carbon sources must be declared at natural abundance when calculating fluxes (e.g. CO_2 input should be declared as: *12C=0.9893, 13C=0.0107*).

1.3 How to cite

Thank you for using IsoCor and citing us in your work! It means a lot to us and encourage us to continue its development.

IsoCor: isotope correction for high-resolution MS labeling experiments

Pierre Millard, Baudoin Delépine, Matthieu Guionnet, Maud Heuillet, Floriant Bellvert and Fabien Létisse

Bioinformatics, 2019, doi: 10.1093/bioinformatics/btz209

1.4 Frequently asked questions

See also:

If you wonder how to make the most out of IsoCor, have a look at the *Tutorials*.

1.4.1 Wait, I needed to correct something?

If you don’t fully understand why you should correct your MS isotopic data, we invite you to have a look at Midani et al. 2017 excellent review.

Finally, practical examples are provided in the *Tutorials*.

1.4.2 What are the alternatives to IsoCor?

You will find scripts for correction of HRMS data out there, but we won't recommend any since, to the best of our knowledge, they implement algorithms that partly fail for high-resolution datasets (see IsoCor v2 publication).

Tool	Has GUI?	MS resolution?	Tracers?	Reference	Comment
AccuCor	No	All	^{2}H, ^{13}C, ^{15}N	Su et al. 2017	Faulty at High-resolution
PyNAC	No	UltraHigh only	^{2}H, ^{13}C, ^{15}N	Carreer et al. 2013	none
IsoCorrectoR	No	Low & UltraHigh only	All	Heinrich et al. 2018	none
ElemCor	Yes	All	^{2}H, ^{13}C, ^{15}N, ^{18}O, ^{34}S	Du et al. 2019	Faulty at High-resolution
IsoCor v1	Yes	Low only	All	Millard et al. 2012	none
IsoCor v2	Yes	All	All	Millard et al. 2019	none

Note: If you would like your software to appear in this list, please get in touch with us.

1.4.3 How many peaks should I measure?

For a compound with n atoms of the tracer element, you should measure $n + 1$ peaks.

1.4.4 What mass fractions should I measure?

For a compound with n atoms of the tracer element, you should measure the *mass fractions* corresponding to the compound having incorporated $0, 1, ..., n$ isotopic tracers.

1.4.5 Can I perform correction in case of missing values?

Mass fractions cannot be corrected by IsoCor in case of missing measurement(s).

For instance, in a C_2 compound for which M_2 is not measured, one cannot estimate the contribution of the corresponding isotopologue – through (im)purity of the isotopic tracer – to M_1 or M_0. Hence, these mass fractions cannot be corrected for purity. As another example, in a C_3 compound for which M_2 is not measured, one cannot estimate its contribution – through natural abundance – to M_3. Here again, these mass fractions cannot be corrected for natural abundance of isotopes.

To avoid misinterpretation of partially corrected data, we have prefered to not allow correction to be applied in case of missing values.

1.4.6 How to add a new metabolite/derivative into the database?

Open the relevant database file with a rustic text editor (e.g. *Notepad++*) and add a new row in your file following the format described in *Input data*.

Database files are created by default at the first run of isocor in '/user/isocordb'. Additional metabolites and derivatives databases can also be defined, as described in *Input data*.

Take care not to modify the file format, nor its structure. A typical error comes from Excel replacing '.' to ',' in floats.

1.4.7 What elemental formula should I declare into the database?

Obviously, errors in elemental formulas will result in erroneous *isotopologue distributions*; thus special care must be taken when defining these formulas. Details on the elemental formulas to be declared in IsoCor can be found in *Tutorial section on formulas*.

1.4.8 Is it possible to correct multiply-charged molecules?

Yes, IsoCor takes into account the charge when constructing the correction matrix. The charge state of each metabolite should be declared in the corresponding database file, as detailed in *Input data*.

1.4.9 Should I tailor natural abundance of isotopes for my experiment?

The abundance of each isotope in natural samples depends on their origin. For instance, marine organisms have been reported to have slightly less ^{13}C than land plants [IUPAC2016]. Ideally, you should measure the exact abundance of each isotope present in an unlabeled sample prior to the labeling experiment. However, most of the time such an experiment would require too much resources for a negligible gain in precision, as we previously found [Millard2014]. The default values should be good enough for most users, unless you work with strongly exotic material.

1.4.10 Where does the default values for natural abundance and mass come from?

From IUPAC [IUPAC2016].

1.4.11 Should I correct the tracer purity for my experiment?

Yes, if you know it. The purity of your tracer should be available from your provider of labeled compound.

See also:

Isotopic purity and natural abundance of the tracer

1.4.12 What is the default value for the tracer purity?

By default, we assume a perfect tracer purity.

1.4.13 Should I correct natural abundance of the tracer for my experiment?

Yes, you should correct for the presence of isotopes at natural abundance in unlabeled positions of non-uniformally labeled nutrients.

See also:

Isotopic purity and natural abundance of the tracer

1.4.14 How does IsoCor performs its corrections?

Please have a look at the examples in the Tutorials section. If you are looking for something more detailed, we invite you to review our source code from our git repository. Also, have a look at the logs in Verbose logs mode; all the intermediate results (correction vector used to construct the correction matrix, correction matrix, etc) will allow you to reproduce the results with pen and paper.

1.4.15 How is computed the mean enrichment?

The mean isotopic enrichment of a metabolite refers to the mean content in isotopic tracer in the metabolite, expressed as the relative fraction of total atoms of its element in the metabolite. This information is particularly useful for the quantification of split ratios between two metabolic pathways resulting in different content of tracer. IsoCor calculates the mean enrichment (ME) using the formula $ME = \frac{\sum_{i=1}^{n} M_i.i}{n}$, where M_i is the proportion of isotopologues with i ^{13}C atoms for a metabolite containing n carbon atoms.

1.4.16 What is the isotopic InChI?

The IUPAC International Chemical Identifier (InChI) is a textual identifier for chemical substances, designed to provide a standard way to encode molecular information and to facilitate the search for such information in databases and on the web.

The identifiers describe chemical substances in terms of layers of information. IsoCor generates an isotopic layer that specifies the isotopologue of the tracer element, following the extended representation proposed by the InChI Isotopologue and Isotopomer Development Team:

- Simple definition: /a(Ee#<+|->#…)
- Complete definition:

  /a(<element><isotope_count><isotope_designation>[,<atom_number>])

  - <element> - one or two letter Element code (Ee).
  - <isotope_count> - number of atoms with the designated isotope (#).
  - <isotope_designation> - isotope designation indicated by a sign (+ or -) and number indicating the unit mass difference from the rounded average atomic mass of the element. For example, the average atomic mass of Sn (118.710) is rounded to 119. We specify two ^{118}Sn atoms as "/a(Sn2-1)".
- **Examples:**
  - $^{13}C_2$-isotopologue of alpha-D-glucopyranose:

    "InChI=1/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h2-11H,1H2/t2-,3-,4+,5-,6+/m1/s1/a(C2+1),(C4+0)"
  - $^{16}O_1{}^{18}O_3$-isotopologue of fumarate:

    "InChI=1S/C4H4O4/c5-3(6)1-2-4(7)8/h1-2H,(H,5,6)(H,7,8)/p-2/b2-1+/a(O3+2),(O1+0)"

Warning: This is an experimental feature: isotopic inchis may be subject to change according to the evolution of the IUPAC specifications.

1.4.17 I cannot start IsoCor graphical user interface, can you help me?

If you installed IsoCor following our standard procedure and that you are unable to start IsoCor by opening a terminal and typing `isocor`, then there is indeed something wrong. Do not panic, we are here to help! Please follow this simple procedure:

1. The first step of the debugging process will be to get a *traceback*, i.e. a message telling us what is actually going wrong:
   - On Unix-based systems, you should already see it in the terminal you opened.
   - On Windows, you will have to open IsoCor from your Anaconda prompt with `python.exe -m isocor` to display the traceback.
2. Read the traceback and try to understand what is going wrong:
   - If it is related to your system or your Python installation, you will need to ask some help from your local system administrator or your IT department so they could guide you toward a clean installation. Tell them that you wanted "to use the graphical user interface of IsoCor, a Python 3.5 software" and what you did so far (installation), give them the traceback and a link toward the documentation. They should know what to do.
   - If you believe the problem is in IsoCor or that your local system administrator told you so, then you probably have found a bug! We would greatly appreciate if you could open a new issue on our issue tracker. One of the developers will help you.

1.4.18 I would like a new feature.

We would be glad to improve IsoCor. Please get in touch with us so we could discuss your problem.

1.5 Definitions

Some important definitions are provided in this section.

1.5.1 Isotopologues

Molecular entities that differ only in their isotopic composition (number of isotopic substitutions), according to its IUPAC definition. It consists of isotopic forms that either:

i. do not have the same isotopic composition but have the same nominal mass (e.g., $^{13}CH_3NH_2$and CH_2DNH_2)

ii. or have neither the same isotopic composition nor the same nominal mass (e.g., CH_4, CH_3D, CH_2D_2). The ability of the MS analyzer to distinguish different isotopologues is determined only by the elemental formula of the molecule and the resolution of the MS analyzer.

1.5.2 Tracer isotopologues

Isotopologues that differ only in their isotopic composition (number of isotopic substitutions) of the tracer element (e.g. CH_3NH_2, CH_2DNH_2 and CHD_2NH_2 are considered as tracer isotopologues in a ^{2}H-labeling experiments). A molecular entity containing n tracer atoms has $n+1$ tracer isotopologues. Tracer isotopologues have different nominal masses, hence they can be distinguished both at low and high MS resolution.

1.5.3 Isotopic cluster

Group of MS peaks that originate from a unique molecular entity, i.e. with the same elemental composition but different isotopic compositions, according to its IUPAC definition. A given peak of the isotopic cluster typically contains several isotopologues. The number and the nature of isotopologues overlapped in each peak is determined only by the elemental formula of the molecule and the resolution of the MS analyzer. The intensity of each peak depends on the abundance of each isotopologue, and thus on the incorporation of tracer.

1.5.4 Mass fractions

The intensity or area of the peaks of the isotopic cluster that contain information on the incorporation of tracer, i.e. of the MS peaks originating from the molecular entity containing $0, 1, ..., n$ tracer atoms, where n is the number of atoms of tracer element in this entity. Mass fractions also contains the contribution of other isotopic species which must be removed. The set of overlapping isotopic species is determined only by the elemental formula of the molecule and the resolution of the MS analyzer.

1.5.5 Isotopologue distribution

Abundance of tracer isotopologues relative to the total pool of the tracer isotopologues. It contains labeling information which is specific to the incorporation of tracer atoms. Isotopologue distribution, obtained after correcting mass fractions for naturally occuring isotopes, is the output of IsoCor.

1.5.6 Resolution

Resolution measures of the ability of the MS analyzer to distinguish two peaks of slightly different mass-to-charge ratios $\delta(M)$ in a mass spectrum. It is classically described as a fixed value of $M/\delta(M)$ (the *nominal resolution*, defined as Full Width at Half Maximum (FWHM) at a particular m/z). In order for the isotopic peaks to be well separated, the mass difference should be greater than 1.66 FWHM.

1.6 Library documentation

IsoCor is also a Python module that you can import directly, for instance in Jupyter Notebooks or in your own software.

Warning: This is an experimental feature: methods signatures may be subject to change between IsoCor versions.

1.6.1 Showcase: isocor Python package

```
[1]: import isocor
```

The correction of a measurement vector starts by instantiating a metabolite-specific *corrector* object and calling its `.correct()` method.

Instantiation of a *corrector*

We recommend that you use `MetaboliteCorrectorFactory` to choose the best correction strategy depending on your resolution:

```
[2]: corrector = isocor.mscorrectors.MetaboliteCorrectorFactory("C3PO", tracer="13C")
     corrector_HR = isocor.mscorrectors.MetaboliteCorrectorFactory("C3PO", tracer="13C",␣
     ↪resolution=1e4, mz_of_resolution=400, charge=1)
```

```
[3]: corrector
```

```
[3]: <isocor.mscorrectors.LowResMetaboliteCorrector at 0x7fdb77eb1a30>
```

```
[4]: corrector_HR
```

```
[4]: <isocor.mscorrectors.HighResMetaboliteCorrector at 0x7fdb77eb1970>
```

Watch out for default parameters!

You should pay attention to the default parameters of `MetaboliteCorrectorFactory`, such as the mass and natural abundance stored for each isotope, the formula used to interpret the resolution, etc.

```
[5]: corrector.data_isotopes
```

```
[5]: {'C': {'abundance': [0.9893, 0.0107],
       'mass': [Decimal('12.0'), Decimal('13.003354835')]},
      'H': {'abundance': [0.999885, 0.000115],
       'mass': [Decimal('1.0078250322'), Decimal('2.0141017781')]},
      'N': {'abundance': [0.99636, 0.00364],
       'mass': [Decimal('14.003074004'), Decimal('15.000108899')]},
      'P': {'abundance': [1.0], 'mass': [Decimal('30.973761998')]},
      'O': {'abundance': [0.99757, 0.00038, 0.00205],
       'mass': [Decimal('15.99491462'),
        Decimal('16.999131757'),
        Decimal('17.999159613')]},
      'S': {'abundance': [0.9499, 0.0075, 0.0425, 0.0, 0.0001],
       'mass': [Decimal('31.972071174'),
        Decimal('32.971458910'),
        Decimal('33.9678670'),
        Decimal('35.0'),
        Decimal('35.967081')]},
      'Si': {'abundance': [0.92223, 0.04685, 0.03092],
       'mass': [Decimal('27.976926535'),
        Decimal('28.976494665'),
        Decimal('29.9737701')]}}
```

```
[6]: corrector.tracer_purity  # perfect purity: [0% of 12C, 100% of 13C]
```

```
[6]: [0.0, 1.0]
```

Direct instantiation

If you don't want to use the factory or want to create your own *corrector*, you can use `LowResMetaboliteCorrector` and `HighResMetaboliteCorrector` directly for respectively low and high resolution data.

But keep in mind that we do not use the same algorithm at low and high resolution.

```
[7]: corrector_LR  = isocor.mscorrectors.LowResMetaboliteCorrector(formula="C3PO", tracer="13C
     ↪",
                                                               derivative_formula=None,
                                                               tracer_purity=[0.0, 1.0],
                                                               data_isotopes=isocor.
     ↪mscorrectors.LowResMetaboliteCorrector.DEFAULT_ISODATA,
                                                               correct_NA_tracer= False)
     corrector_LR
```

```
[7]: <isocor.mscorrectors.LowResMetaboliteCorrector at 0x7fdb77eb1b20>
```

Correction of a measurement vector

The correction is performed on a vector of measurements through the `.correct()` method:

```
[8]: corrected_area, iso_fraction, res, m_enr = corrector.correct([0., 4000., 200., 0.])
     corrected_area
```

```
[8]: array([1.30186754e-05, 4.00972659e+03, 1.98956608e+02, 0.00000000e+00])
```

In addition of the corrected area, `.correct()` returns:

- isotopologue_fraction, that is the abundance of each tracer isotopologue (corrected area normalized to 1)
- residuum
- mean enrichment

One corrector can safely be re-used for other measurements vectors (given that it is the same metabolite/derivative/parameters), for instance in another sample:

```
[9]: corrector.correct([0., 4000, 2000, 1000])
```

```
[9]: (array([1.16470462e-11, 4.00974368e+03, 2.00334442e+03, 9.93432796e+02]),
      [1.6623152066774086e-15,
       0.5722874070659756,
       0.2859257045808499,
       0.14178688835317282],
      [-1.6598205588440532e-15,
       1.9489172180848461e-16,
       -8.120488408686859e-16,
       7.633259104165648e-16],
      0.523166493762398)
```

Empty vectors will give:

[10]:
```
corrector.correct([0., 0., 0., 0.])
```

[10]:
```
(array([0., 0., 0., 0.]), [nan, nan, nan, nan], [nan, nan, nan, nan], nan)
```

IsoCor check the inputs you provide and warn you if you did something unexpected:

[11]:
```
try:
    corrector.correct([0., 0., 0.])  # our corrector was defined for a C3, so we expect␣
↪4 "peaks" (n+1)
except Exception as e:
    print(e.__class__, e)
```

```
<class 'ValueError'> The length of the measured isotopic cluster (3) is different than␣
↪the required number of measurements: 4 (i.e. N + 1, where N is the number of atoms␣
↪that could be traced)
```

Readable attributes

Everything is accessible from the *corrector*.

[12]:
```
corrector.formula
```

[12]:
```
Counter({'C': 3, 'P': 1, 'O': 1})
```

The correction matrix is always available (computed on-the-fly at first access):

[13]:
```
corrector.correction_matrix
```

[13]:
```
array([[9.9757e-01, 0.0000e+00, 0.0000e+00, 0.0000e+00],
       [3.8000e-04, 9.9757e-01, 0.0000e+00, 0.0000e+00],
       [2.0500e-03, 3.8000e-04, 9.9757e-01, 0.0000e+00],
       [0.0000e+00, 2.0500e-03, 3.8000e-04, 9.9757e-01]])
```

[14]:
```
corrector_HR.correction_matrix
```

[14]:
```
array([[9.9757e-01, 0.0000e+00, 0.0000e+00, 0.0000e+00],
       [3.8000e-04, 9.9757e-01, 0.0000e+00, 0.0000e+00],
       [0.0000e+00, 3.8000e-04, 9.9757e-01, 0.0000e+00],
       [0.0000e+00, 0.0000e+00, 3.8000e-04, 9.9757e-01]])
```

Of course, other attributes and parameters are available too:

[15]:
```
corrector.molecular_weight
```

[15]:
```
Decimal('82.968676618')
```

[16]:
```
corrector.correct_NA_tracer
```

[16]:
```
False
```

The correction matrix is unique to a corrector object. **You should never have to set or reset the correction matrix yourself**.

If you need a new correction matrix, you should re-instanciate a new corrector object. IsoCor forbids you to change directly the correction matrix:

```
[17]: try:
          corrector.correction_matrix = [45]
      except Exception as e:
          print(e.__class__, e)
```

```
<class 'AttributeError'> can't set attribute
```

In the same vein, you should never change parameters of the corrector at runtime. IsoCor might even forbid you to do so:

```
[18]: try:
          corrector.formula = {'H': 2, 'O': 1}
      except Exception as e:
          print(e.__class__, e)
```

```
<class 'AttributeError'> can't set attribute
```

As a rule of thumbs, always instantiate a new corrector if anything else than the measurement vector changed in your analysis.

Catch the logs

You can use *logging* to catch the logs. By default, logs go to the `stderr` (that is printed in Jupyter notebooks):

```
[19]: import logging

      logging.basicConfig(format='%(asctime)s - %(levelname)s - %(message)s', datefmt='%H:%M',␣
      ↪level=logging.DEBUG)
```

```
[20]: # Let's do something silly
      isocor.MetaboliteCorrectorFactory("C3PO", '13C', resolution=0.1, mz_of_resolution= 400,␣
      ↪charge=1)
```

```
15:47 - DEBUG - MetaboliteCorrectorFactory chose to use a HighResMetaboliteCorrector for␣
↪C3PO.
15:47 - WARNING - Improper usage of HighResMetaboliteCorrector by␣
↪MetaboliteCorrectorFactory. Falling back to LowResMetaboliteCorrector. Reason: The␣
↪correction limit is expected to be sufficent to distinguish peaks with a delta-mass of␣
↪1 amu (627.2625176486116>0.5).
15:47 - DEBUG - New LowResMetaboliteCorrector, C3PO||13C: [('_data_isotopes', {'C': {
↪'abundance': [0.9893, 0.0107], 'mass': [Decimal('12.0'), Decimal('13.003354835')]}, 'H
↪': {'abundance': [0.999885, 0.000115], 'mass': [Decimal('1.0078250322'), Decimal('2.
↪0141017781')]}, 'N': {'abundance': [0.99636, 0.00364], 'mass': [Decimal('14.003074004
↪'), Decimal('15.000108899')]}, 'P': {'abundance': [1.0], 'mass': [Decimal('30.973761998
↪')]}, 'O': {'abundance': [0.99757, 0.00038, 0.00205], 'mass': [Decimal('15.99491462'),␣
↪Decimal('16.999131757'), Decimal('17.999159613')]}, 'S': {'abundance': [0.9499, 0.0075,
↪ 0.0425, 0.0, 0.0001], 'mass': [Decimal('31.972071174'), Decimal('32.971458910'),␣
↪Decimal('33.9678670'), Decimal('35.0'), Decimal('35.967081')]}, 'Si': {'abundance': [0.
↪92223, 0.04685, 0.03092], 'mass': [Decimal('27.976926535'), Decimal('28.976494665'),␣
↪Decimal('29.9737701')]}}), ('_molecular_weight', None), ('_tracer_purity', [0.0, 1.0]),
↪ ('_correct_NA_tracer', False), ('_formula', Counter({'C': 3, 'P': 1, 'O': 1})), ('_
↪inchi', ''), ('_isotopic_inchi', None), ('_derivative_formula', None), ('_correction_
↪formula', None), ('_mzshift_tracer', None), ('_str_formula', 'C3PO'), ('_str_
```

(continues on next page)

(continued from previous page)

```
↪derivative_formula', ''), ('_str_tracer_code', '13C'), ('_charge', None), ('_tracer_el
↪', 'C'), ('_idx_tracer', 1), ('label', 'C3PO||13C'), ('_correction_matrix', None)].
15:47 - DEBUG - C3PO||13C additionally uses: []
```

```
[20]: <isocor.mscorrectors.LowResMetaboliteCorrector at 0x7fdba45958e0>
```

```
[21]: # it is forbidden to half-define a isocorrector
try:
    isocor.MetaboliteCorrectorFactory("C3PO", '13C', mz_of_resolution= 400, charge=1)
except Exception as e:
    print(e.__class__, e)
```

```
15:47 - ERROR - MetaboliteCorrectorFactory was unable to select a correction strategy.␣
↪Please check your inputs.
```

```
<class 'ValueError'> MetaboliteCorrectorFactory was unable to select a correction␣
↪strategy. Please check your inputs.
```

1.6.2 Reference

`base.py`

Parent classes used by IsoCor to define specific *correctors* sub-classes.

There should be no need to instanciate any of those classes directly. However, all *correctors* classes should inherit from the classes:

- *`LabelledChemical`* that stores data related to the metabolite to correct and checks the inputs,
- *`InterfaceMSCorrector`* that contractualize what any corrector should be able to do.

class `isocor.base.`**`InterfaceMSCorrector`**

Bases: `object`

Interface for Mass Spectrometry correctors.

This class defines the minimal methods that a corrector should implement. Keep in mind that a corrector must inherit from both *`InterfaceMSCorrector`* and *`LabelledChemical`*.

exception **`ImproperUsageError`**

Bases: `Exception`

Raised when the corrector is used incorrectly.

`compute_correction_matrix``()`

Returns the correction matrix taking into account all parameters.

Warning: Does not set *`correction_matrix`* if called directly.

Returns
the correction matrix

Return type
array

`correct`(*measurement*)

Return corrected measurement vector.

Parameters

`measurement` (*`list`*) – measured areas

Returns

- corrected_area (array): Corrected area for each peak.
- isotopologue_fraction (list): The abundance of each tracer isotopologue (corrected area normalized to 1).
- residuum
- mean enrichment

Return type

tuple

`property correction_matrix`

Correction matrix.

The correction matrix will be set once and for all using `compute_correction_matrix()` during the first access. Use *del x.correction_matrix* if you wish to reset it.

`class isocor.base.LabelledChemical`(*formula*, *tracer*, *derivative_formula*, *tracer_purity*, *correct_NA_tracer*, *data_isotopes*, *charge=None*, *label=None*, *inchi=None*)

Bases: `object`

A labeled chemical considered for isotope correction.

This class defines and check the parameters (formulas, tracer, etc.). needed to model a chemical undergoing MS correction.

Default isotopic data come from:

Isotopic Compositions of the Elements 2013, Pure Appl. Chem., 2016, Vol. 88, No. 3, pp. 293-306, https://doi.org/10.1515/pac-2015-0503

Warning: Except specified otherwise, you should never set any attribute at runtime. Always make a new instance if you wish to change the parameters.

Parameters

- **`formula`** (*`str`*) – elemental formula of the metabolite moiety (e.g. “C3H7O6P”)
- **`inchi`** (*`str`*) – InChI of the metabolite (e.g. “InChI=1/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/ h2-11H,1H2/t2-,3-,4+,5-,6+/m1/s1” for alpha- D -glucopyranose). Note that the InChI might represents the metabolite moiety (e.g. a fragment ion) or the metabolite, hence its formula may differ from `formula`.
- **`tracer`** (*`str`*) – the isotopic tracer (e.g. “13C”)
- **`charge`** (*`int`*) – charge of the detected ion
- **`label`** (*`str`*) – metabolite abbreviation (e.g. “G3P”)
- **`data_isotopes`** (*`dict`*) – isotopic data with mass and abundance as in `DEFAULT_ISODATA`. If set to *None*, default isotopic data are used.
- **`derivative_formula`** (*`str`*) – elemental formula of the derivative moiety

- **tracer_purity** (*list*) – proportion of each isotope of the tracer element (metabolite moiety) The list must have the same length as the list of isotopes for the relevant isotope in `data_isotopes`. Must be normalized to one. If set to *None*, a perfect purity is assumed.
- **correct_NA_tracer** (*bool*) – flag to correct tracer natural abundance or not. If set to True, tracer elements of the metabolite moiety will be corrected for the natural isotopic abundance of the tracer. Note that tracer elements from the derivative moiety will always be corrected at natural abundance (by definition of a derivative moiety).

```
DEFAULT_ISODATA = {'C': {'abundance': [0.9893, 0.0107], 'mass': [Decimal('12.0'),
Decimal('13.003354835')]}, 'H': {'abundance': [0.999885, 0.000115], 'mass':
[Decimal('1.0078250322'), Decimal('2.0141017781')]}, 'N': {'abundance': [0.99636,
0.00364], 'mass': [Decimal('14.003074004'), Decimal('15.000108899')]}, 'O':
{'abundance': [0.99757, 0.00038, 0.00205], 'mass': [Decimal('15.99491462'),
Decimal('16.999131757'), Decimal('17.999159613')]}, 'P': {'abundance': [1.0],
'mass': [Decimal('30.973761998')]}, 'S': {'abundance': [0.9499, 0.0075, 0.0425, 0.0,
0.0001], 'mass': [Decimal('31.972071174'), Decimal('32.971458910'),
Decimal('33.9678670'), Decimal('35.0'), Decimal('35.967081')]}, 'Si': {'abundance':
[0.92223, 0.04685, 0.03092], 'mass': [Decimal('27.976926535'),
Decimal('28.976494665'), Decimal('29.9737701')]}}
```

property charge

absolute value of the charge of the metabolite.

Type
int

property correct_NA_tracer

correct tracer natural abundance if set to True

Type
bool

property correction_formula

molecular formula on which the correction will be applied

This formula is the one on which the correction vector is based. Remember that for the correction vector we need **non-tracers atoms** from the measured metabolite (all elements excluding the tracer element) and **all atoms** from the derivative (including the tracer element).

Type
Counter

property data_isotopes

isotopic data with mass and abundance

See `DEFAULT_ISODATA` for an example.

Type
dict of dicts

property derivative_formula

elemental formula of the derivative moiety.

Type
Counter

property formula

elemental formula of the metabolite moiety

Type
Counter

`property inchi`

inchi of the metabolite.

Type
str

`property isotopic_inchi`

Generate isotopic inchis of the corrected fractions, or just the isotopic layer if no InChI has been provided.

Standard proposed by the InChI Isotopologue and Isotopomer Development Team:

Simple Definition: /a(Ee#<+|->#. . .) Complete Definition:

/a(<element><isotope_count><isotope_designation>[,<atom_number>]) <element> - one or two letter Element code (Ee). <isotope_count> - number of atoms with the designated isotope (#). <isotope_designation> - isotope designation indicated by a sign (+ or -) and number

indicating the unit mass difference from the rounded average atomic mass of the element. For example, the average atomic mass of Sn (118.710) is rounded to 119. We specify two 118 Sn atoms as "/a(Sn2-1)".

Example

13C2 isotopologue of alpha-D-glucopyranose: InChI=1/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h2-11H,1H2/t2-,3-,4+,5-,6+/m1/s1 /a(C2+1),(C4+0)

Returns
isotopic inchis

Return type
list

`property molecular_weight`

exact molecular weight of the metabolite.

Including the derivative and the tracer.

Type
Decimal

`property mzshift_tracer`

mass shift of the tracer

Warning: Mass shift is computed from the first isotope of the list in `data_isotopes`.

Type
Decimal

`property tracer_purity`

proportion of each isotope for the tracer element of the metabolite

Type
list

mscorrectors.py

Correctors classes of IsoCor for mass spectrometry data.

Metabolite *correctors* are metabolite-centered objects used to correct mass spectrometry data for natural abundance and tracer purity.

Metabolite *correctors* can be instanciated directly or through their factory (recommended).

class isocor.mscorrectors.**HighResMetaboliteCorrector**(*formula*, *tracer*, *resolution*, *mz_of_resolution*, *resolution_formula_code*, *charge*, ***kwargs*)

Bases: *LowResMetaboliteCorrector*

Metabolite *corrector* for high-resolution mass-spectrometry data.

At high-resolution, we consider that the measured peak **may not** encompass all isotopologues due to isotope natural abundance, depending on the operating resolution of the mass spectrometer at a given m/z.

This is essentially the same correction pipeline as for low-resolution data, except that the calculation of the correction vector must be more precise and therefore requires a more complex (and more time-consumming) algorithm.

Parameters

- **formula** (*str*) – elemental formula of the metabolite moiety (e.g. "C3H7O6P")
- **tracer** (*str*) – the isotopic tracer (e.g. "13C")
- **label** (*str*) – metabolite abbreviation (e.g. "G3P")
- **inchi** (*str*) – InChI of the metabolite (e.g. "InChI=1/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/ h2-11H,1H2/t2-,3-,4+,5-,6+/m1/s1" for alpha- D -glucopyranose). Note that the InChI might represents the metabolite moiety (e.g. a fragment ion) or the metabolite, hence its formula may differ from `formula`.
- **data_isotopes** (*dict*) – isotopic data with mass and abundance as in `DEFAULT_ISODATA`
- **derivative_formula** (*str*) – elemental formula of the derivative moiety
- **tracer_purity** (*list*) – proportion of each isotope of the tracer element (metabolite moiety) The list must have the same length as the list of isotopes for the relevant isotope in `data_isotopes`. Must be normalized to one.
- **correct_NA_tracer** (*bool*) – flag to correct tracer natural abundance or not. If set to True, tracer elements of the metabolite moiety will be corrected for the natural isotopic abundance of the tracer. Note that tracer elements from the derivative moiety will always be corrected at natural abundance (by definition of a derivative moiety).
- **resolution** (*int*) – resolution of the mass spectrometer (e.g. "1e4")
- **mz_of_resolution** (*int*) – m/z at which the resolution was measured (e.g. "400"). This is **not** the m/z of the metabolite.
- **resolution_formula_code** (*str*) – code for the resolution formula you wish to use to compute the correction limit among the presets: "orbitrap", "ft-icr", and "constant". This formula depends on your mass spectrometer.
- **resolution_formula** (*func*) – (EXPERIMENTAL) if no code suits you, you can use this parameter to provide a function returning the resolution at the mz of the metabolite. Use the same format as in *RES_FORMULAS*. `resolution_formula` has precedence over `resolution_formula_code`.
- **charge** (*int*) – charge state of the metabolite (e.g. "-2").

`RES_FORMULAS = {'constant': <function HighResMetaboliteCorrector.<lambda>>, 'datafile': <function HighResMetaboliteCorrector.<lambda>>, 'ft-icr': <function HighResMetaboliteCorrector.<lambda>>, 'orbitrap': <function HighResMetaboliteCorrector.<lambda>>}`

`compute_correction_matrix()`

Returns the correction matrix taking into account all parameters.

Attention: Does not set `correction_matrix` if called directly.

Returns
the correction matrix

Return type
array

`property correction_limit`

Operative correction limit (in Da).

Depends on the calibration resolution, the m/z at which the instrument was calibrated, the m/z of the meabolite and of course the formula that gives the relation between those values (that depends on the spectrometer).

`get_isotopic_cluster()`

Return the full isotopic cluster of the compound (for non-tracer elements).

Parameters
`threshold_p` (*float*) – Probability threshold under which a peak will be ignored (default: keep all peaks).

Returns
isotopologues masses mapped to their respective expected abundance (e.g. *{83.9959772546: 0.966808533640457, …}*).

Return type
dict

`get_mass_distribution_vector()`

Get mass distribution vector (at natural abundancy) for non-tracers elements.

Based on the elemental compositions of both metabolite's and derivative's moieties.

Important: This methods needs to compute the full isotopic cluster, which is an expensive operation.

Returns
mass distribution vector

Return type
list

`get_peaks_around`(*isotopic_cluster*, *masses*)

Sum unresolved peaks of isotopic cluster according to the MS resolution.

Parameters

- `isotopic_cluster` (*dict*) – isotopic cluster as returned by *:meth:~get_isotopic_cluster*

- **masses** (*list*) – m/z around which peaks will be pooled

Returns
abundances for each input mass, after sorting the mass and pooling nearby peaks within the resolution limit (*correction_limit*)

Return type
list

get_tracershifted_peaks_between(*mz_min*, *mz_max*)

Return the mass of all the peaks that originate from the tracer.

Those peaks are the 'main' peaks around which we will find other peaks due to the incorporation of natural isotopes of non-tracer elements (different from the most abundant isotope).

Parameters

- **mz_min** (*float*) – minimum range
- **mz_max** (*float*) – maximum range

Returns
mass of all peaks that originate from the tracer

Return type
list

class isocor.mscorrectors.**LowResMetaboliteCorrector**(*formula*, *tracer*, ***kwargs*)

Bases: *LabelledChemical*, *InterfaceMSCorrector*

Metabolite *corrector* for low-resolution mass-spectrometry data.

At low-resolution, all isotopologues with the same nominal mass are considered to be measured together.

Parameters

- **formula** (*str*) – elemental formula of the metabolite moiety (e.g. "C3H7O6P")
- **inchi** (*str*) – InChI of the metabolite (e.g. "InChI=1/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/ h2-11H,1H2/t2-,3-,4+,5-,6+/m1/s1" for alpha- D -glucopyranose). Note that the InChI might represents the metabolite moiety (e.g. a fragment ion) or the metabolite, hence its formula may differ from `formula`.
- **tracer** (*str*) – the isotopic tracer (e.g. "13C")
- **label** (*str*) – metabolite abbreviation (e.g. "G3P")
- **data_isotopes** (*dict*) – isotopic data with mass and abundance as in `DEFAULT_ISODATA`
- **derivative_formula** (*str*) – elemental formula of the derivative moiety
- **tracer_purity** (*list*) – proportion of each isotope of the tracer element (metabolite moiety) The list must have the same length as the list of isotopes for the relevant isotope in `data_isotopes`. Must be normalized to one. Default is perfect purity.
- **correct_NA_tracer** (*bool*) – flag to correct tracer natural abundance or not. If set to True, tracer elements of the metabolite moiety will be corrected for the natural isotopic abundance of the tracer. Note that tracer elements from the derivative moiety will always be corrected at natural abundance (by definition of a derivative moiety).

compute_correction_matrix()

Returns the correction matrix taking into account all parameters.

Attention: Does not set `correction_matrix` if called directly.

Returns
the correction matrix

Return type
array

`correct`(*measurement*)

Return corrected measurement vector.

Parameters
`measurement` (*`list`*) – measured areas

Returns

- corrected_area (array): Corrected area for each peak.
- isotopologue_fraction (list): The abundance of each tracer isotopologue (corrected area normalized to 1).
- residuum
- mean enrichment

Return type
tuple

`get_mass_distribution_vector`()

Get low resolution mass distribution vector (at natural abundance) for non-tracers elements.

Based on the elemental compositions of both metabolite's and derivative's moieties. The element corresponding to the isotopic tracer is ignored.

Calculation is based on convolution of isotopic vectors of individual elements (IsoCor, Millard et al., 2012), which is much faster than the combinatorial approach implemented for simulation at high resolution.

Returns
mass distribution vector

Return type
list

`class isocor.mscorrectors.`**`MetaboliteCorrectorFactory`**(*formula*, *tracer*, ***kwargs*)

Bases: `object`

A Factory that returns the right *metabolite corrector* given the correction options.

Parameters

- **`formula`** (*`str`*) – elemental formula of the metabolite moiety (e.g. "C3H7O6P")
- **`inchi`** (*`str`*) – InChI of the metabolite (e.g. "InChI=1/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/ h2-11H,1H2/t2-,3-,4+,5-,6+/m1/s1" for alpha- D -glucopyranose). Note that the InChI might represents the metabolite moiety (e.g. a fragment ion) or the metabolite, hence its formula may differ from `formula`.
- **`tracer`** (*`str`*) – the isotopic tracer (e.g. "13C")
- **`label`** (*`str`*) – metabolite abbreviation (e.g. "G3P")

- **`data_isotopes`** (*`dict`*) – isotopic data with mass and abundance as in `DEFAULT_ISODATA` (default values)
- **`derivative_formula`** (*`str`*) – elemental formula of the derivative moiety (default: no derivative)
- **`tracer_purity`** (*`list`*) – proportion of each isotope of the tracer element (metabolite moiety) The list must have the same length as the list of isotopes for the relevant isotope in `data_isotopes`. Must be normalized to one. Default is perfect purity.
- **`correct_NA_tracer`** (*`bool`*) – flag to correct tracer natural abundance or not. If set to True, tracer elements of the metabolite moiety will be corrected for the natural isotopic abundance of the tracer. Note that tracer elements from the derivative moiety will always be corrected at natural abundance (by definition of a derivative moiety). Default is False.
- **`resolution`** (*`int`*) – resolution of the mass spectrometer (e.g. "1e4"); default "None" (low-resolution).
- **`mz_of_resolution`** (*`int`*) – m/z at which the resolution was measured (e.g. "400"). This is **not** the m/z of the metabolite.
- **`resolution_formula_code`** (*`str`*) – code for the resolution formula you wish to use to compute the correction limit among the presets: "orbitrap" (default), "ft-icr", and "constant". This formula depends on your mass spectrometer.
- **`charge`** (*`int`*) – charge state of the metabolite (e.g. "-2").

Raises

- **`ValueError`** – wrong input
- ***`ImproperUsageError`*** – incoherent input

1.6.3 Unit tests

Isotope correction is a complex task and we use (some) unit tests to make sure that critical features are not compromised during development.

A total of 318 tests has been designed to test IsoCor from individual steps (e.g. calculation of theoretical *mass fractions* or correction matrices) to the entire correction process. Importantly, most of the tests compare intermediate (e.g. correction matrix) or final (e.g. *isotopologue distribution*) calculations of IsoCor to theoretical results, with algebraic equations provided in the code for manual check.

All situations that may be encountered (e.g. in terms of tracer, metabolite or MS *resolution*) should be covered by these tests.

You can run all tests by calling `pytest` in the shell at project's root directory (it must be installed beforehand):

```
pytest
```

To add a new test-case, see pytest documentation.

Tests are stored in `tests/` folder.

Isotopic clusters

Test the calculation of low-resolution isotopic clusters (for metabolites with different number of elements and isotopes) against a brute force implementation.

`isocor.tests.test_isotopic_cluster_LowRes.`**`get_isoclust_bruteforce`**`(`*formula*, *data_iso*`)`

Return the low-resolution isotopic cluster of a compound using convolution.

`isocor.tests.test_isotopic_cluster_LowRes.`**`test_isoclust_against_bruteforce`**`(`*formula*, *data_iso*, *usr_tolerance*`)`

Check the low resolution isotopic clusters generation against a simple implementation.

Test the calculation of high-resolution isotopic clusters (for metabolites with different number of elements and isotopes) against a brute force implementation.

`isocor.tests.test_isotopic_cluster_HighRes.`**`get_isoclust_bruteforce`**`(`*formula*, *data_iso*`)`

Return the isotopic cluster for a compound using a brute-force method.

This method is based on systematic computation of all isotopic species of the molecule. It should be easier to understand than the optimized version but is much slower.

`isocor.tests.test_isotopic_cluster_HighRes.`**`test_isoclust_against_bruteforce`**`(`*formula*, *data_iso*, *usr_tolerance*`)`

Check the high-resolution isotopic clusters generation against a brute force implementation.

Minimal mass difference to distinguish two isotopic species

Test calculation of the minimal *mass* difference required to resolve two isotopic species of a labeled chemical.

The minimal *mass* difference required to separate two isotopic species of a labeled chemical correspond to the minimal difference of mz to distinguish two MS peaks at the operating resolution multiplied by the charge. The minimal mz difference is calculated at the operating resolution using 'resolution_formula' 'constant' at the mz of the labeled chemical.

`isocor.tests.test_m_min_constant.`**`test_m_min_constant`**`(`*data*, *data_iso*`)`

Calculation of the minimal *mass* difference (m_min) required to resolve two isotopic species of a labeled chemical at different resolution (10000, 100000) & charge (1, 2) for 'constant' resolution. m_min calculated by IsoCor is compared to the expected value (provided in the fixture) and to the theoretical value (equation provided in the code). This is checked for correctors instantiated through the Factory or directly as HighResMetaboliteCorrector.

Test calculation of the minimal *mass* difference required to resolve two isotopic species of a labeled chemical.

The minimal *mass* difference required to separate two isotopic species of a labeled chemical correspond to the minimal difference of mz to distinguish two MS peaks at the operating resolution multiplied by the charge. The minimal mz difference is calculated at the operating resolution using 'resolution_formula' of orbitrap (Su et al., 2017) at the mz of the labeled chemical.

`isocor.tests.test_m_min_orbitrap.`**`test_m_min_orbitrap`**`(`*data*, *data_iso*`)`

Calculation of the minimal *mass* difference (m_min) required to resolve two isotopic species of a labeled chemical at different resolution (10000, 100000) & charge (1, 2) on an *Orbitrap*. m_min calculated by IsoCor is compared to the expected value (provided in the fixture) and to the theoretical value (equation provided in the code). This is checked for correctors instantiated through the Factory or directly as HighResMetaboliteCorrector.

Correction matrix

Test low-resolution correction matrix when no isotopic species are resolved from the tracer isotopologues.

In this situation, the contribution of all elements and isotopes should be removed. The low-resolution correction matrix calculated by IsoCor is compared to a theoretical correction matrix (with algebraic equations provided in the code for manual check).

`isocor.tests.test_correction_matrix_LowRes.`**`test_low_res_cor_matrix`**`(`*data*, *data_iso*, *usr_tolerance*`)`

Correction of low resolution MS data.

Test simulates the correction of C2 compounds in a 13C-tracer experiment.

Test high-resolution correction matrix when some or all isotopic species are resolved from the tracer isotopologues.

In this situation, the contribution of only some isotopic species should be removed, depending on the resolution. The high-resolution correction matrix calculated by IsoCor is compared to a theoretical correction matrix (with algebraic equations provided in the code for manual check).

`isocor.tests.test_correction_matrix_HighRes_resolved.`**`test_high_res_cor_matrix`**`(`*data*, *data_iso*, *usr_tolerance*`)`

Correction of high-resolution MS data.

Test calculates the correction matrix of labeled metabolites containing C, H, N and O elements, and compares this matrix to a theoretical matrix.

Test high-resolution correction matrix when no isotopic species are resolved from the tracer isotopologues.

In this situation, the contribution of all elements and isotopes should be removed, hence the high-resolution and low-resolution correction matrices should be the same.

The high-resolution correction matrix calculated by IsoCor is compared to

1. a theoretical correction matrix (with algebraic equations provided in the code for manual check), and
2. the low resolution correction matrix.

`isocor.tests.test_correction_matrix_HighRes_unresolved.`**`test_high_res_cor_matrix_unresolved`**`(`*data*, *data_iso*, *usr_tolerance*`)`

Correction of high resolution MS data.

This test is designed to evaluate the correction of high-resolution MS data where none of the isotopic species are resolved from the tracer isotopologues, i.e. the high-resolution correction algorithm should provides the same correction matrix as the one computed using the low-resolution correction algorithm.

Correction process examples

Test the entire low-resolution correction process at low-resolution.

Compare mass fractions corrected by IsoCor to theoretical isotopologues distributions (with algebraic equations provided in the code for manual check).

`isocor.tests.test_correction_process_LowRes.`**`test_low_res_correction`**`(`*data*, *data_iso*, *usr_tolerance*`)`

Correction of non-tracer elements depending on resolution and formula.

Measured values should be corrected for the incorporation of natural isotopes of non-tracers elements if the resolution was not high enough to distinguish them. The resolution depends on m/z ratio and thus the formula of the metabolite. The higher the mass, the higher the resolution should be used to distinguish all isotopomers and avoid to need correction.

Test the entire high-resolution correction process at high-resolution.

Comparison of mass fractions corrected by IsoCor to theoretical isotopologues distributions (with algebraic equations provided in the code for manual check).

`isocor.tests.test_correction_process_HighRes.`**`test_high_res_correction`**`(`*data*, *data_iso*`)`

Correction of non-tracer elements depending on resolution and formula.

Measured values should be corrected for the incorporation of natural isotopes of non-tracers elements if the resolution was not high enough to distinguish them. The resolution depends on m/z ratio and thus the formula of the metabolite. The higher the mass, the higher the resolution should be used to distinguish all isotopomers and avoid to need correction.

Test the entire correction process at low- and high-resolution to cover all the situations that can be encountered.

This aims to verify that no error will be raised during normal usage. This set of tests correct a measurement vector (set to unity, i.e. [1,1,1,1] for a compound with 3 atoms of the tracer element) in all the possible combinations of the following situations:

- formulas: C1, C2, O1, O2, O3P3N3H3C3
- tracer elements: 13C, 17O, 18O
- derivative: H, H2, O, O2
- purity: 100%
- resolution (for high-resolution correction only): 10000, 100000, 1000000
- correct NA: True, False

Total number of tests: 240

`isocor.tests.test_all_cases.`**`test_correction_process_misc`**`(`*data*, *data_iso*`)`

Tests the entire correction process at high resolution.

Combination of all the following situations:

- formulas: O3P3N3H3C3, C1, O1, O2, C2, O1
- tracer elements: 13C (2 isotopes), 17O (3 isotopes), 18O (3 isotopes)
- derivative: H, H2, O, O2
- purity: 100%
- resolution: 10000, 100000, 1000000
- correct NA: True, False

Total number of tests: 240

Misc.

Test the MetaboliteCorrectorFactory.

`isocor.tests.test_factory.`**`test_MetaboliteCorrectorFactory`**`(`*data_iso*`)`

Test that the Factory can be instanciated safely.

`isocor.tests.test_factory.`**`test_badinput`**`(`*kwargs*, *data_iso*`)`

Test a wrong type of parameter value (Factory).

`isocor.tests.test_factory.`**`test_isotopedata`**`(`*bad_dataiso*`)`

Tests expected to fail because of a badly formatted data_isotopes.

`isocor.tests.test_factory.`**`test_typo_factory`**`(`*data_iso*`)`

Test a typo in the name of one of the parameters (Factory).

`isocor.tests.test_factory.`**`test_typo_parameter`**`(`*data_iso*`)`

Test a typo in the name of one of the parameters (LowRes).

Configuration variables for the tests

Shared pre-defined parameters The return value of fixture function will be available as a predefined parameter for all test functions. The test's parameter name must be the same as the fixture function's name.

`isocor.tests.conftest.`**`data_iso`**`()`

Typical isotopic data object.

`isocor.tests.conftest.`**`usr_tolerance`**`()`

Platform-dependent tolerance.

1.7 License

BIBLIOGRAPHY

[IUPAC2016] Isotope-abundance variations and atomic weights of selected elements: 2016 (IUPAC Technical Report) https://doi.org/10.1515/pac-2016-0302

[Millard2014] Isotopic studies of metabolic systems by mass spectrometry: using Pascal's triangle to produce biological standards with fully controlled labeling patterns, 2014, Anal. Chem., 86(20):10288-10295, https://doi.org/10.1021/ac502490g

PYTHON MODULE INDEX

INDEX