IsoSolve Documentation

Release 1.0.3

Serguei Sokol

Mar 08, 2021

Contents:

IsoSolve is a Python module realizing an integrative framework for isotope labeling measurements. It can provide both symbolic and numeric solutions for integrating and consolidating heterogeneous measurements, e.g. NMR and MS measuring ^{13}C labeling, to cite the most wide spread ones. This integrative framework helps to

i) clarify and improve the coverage of the isotopic space;

ii) evaluate the complementarity and redundancy of different techniques;

iii) consolidate isotopic datasets;

iv) design experiments, and

v) guide future analytical developments.

This framework can be applied to any labeled element, isotopic tracer, metabolite, and analytical platform.

IsoSolve is available as a Python library with a command line interface. It is one of the routine tools that we use at the MetaSys team and MetaToul platform in isotopic studies of metabolic systems.

A paper on this subject is under preparation.

If you have an idea on how we could improve IsoSolve please submit a new issue to our GitHub issue tracker.

CHAPTER 1

User's manual

1.1 Installation

To install the latest stable release run in a shell

```
pip install --user isosolve
```

Use `pip3` instead of `pip` when python2 and python3 are both accessible. An executable script may be installed in a directory which is not in your `PATH` variable. If it is the case, you will be advertised accordingly during the installation process. Add the corresponding directory to the PATH to be able to launch `isosolve` from any directory on your computer.

To test if the installation worked as expected run:

```
isosolve -v
```

It should just print the program name followed by its version number, e.g.

```
isosolve 1.0
```

1.2 Uninstall

```
pip uninstall isosolve
```

1.3 Upgrade

```
pip install --user --upgrade isosolve
```

1.4 Dependencies

IsoSolve depends on following Python modules:

- sympy
- nlsic
- numpy
- scipy
- pandas
- mdutils
- markdown

For the first time installation or upgrade of nlsic module, a Fortran compiler is required. If it is not present on your system, you'll have to install it by your own means. Here are some indications for different systems. If the yourth is not present, ask for help your local computer guru.

1.4.1 Conda (multi-platform)

```
conda install -c conda-forge fortran-compiler
```

1.4.2 Linux

On Debian-like distributions you can do:

```
apt-get install gfortran
```

On RPM-like:

```
yum install gcc-gfortran
```

On Mageia:

```
urpmi gcc-gfortran
```

1.4.3 Windows

On windows platform we used conda solution. If you don't have conda you can try to install cygwin and choose a package `gcc-fortran` (not tested)

1.4.4 MacOS (not tested)

If you have Homebrew installed then you can try

```
brew update
brew install gcc
```

`gfortran` is part of `gcc` package.

If Python dependencies are lacking on your system they will be automatically installed. On the other hand, if you uninstall IsoSolve and don't need anymore dependencies, you'll have to uninstall them manually.

1.5 Quick startup

To test IsoSolve on prepared example data, download two files : ALA_mapping.tsv and ALA_measurements.tsv. The first file describes a mapping of different measurement techniques on isotopomers of Alanine amino acid labeled with ^{13}C and the second one (optional) provides measured data and their standard deviations (SD).

Run the following command from the directory having these files:

```
isosolve -d ALA_measurements.tsv ALA_mapping.tsv
```

or just

```
isosolve ALA_mapping.tsv
```

if you wish only symbolic formulas.

1.5.1 Input/Output files

IsoSolve takes as input isotopic mapping for a given metabolite, e.g. carbon isotopologue distribution measured by MS or specific enrichments measured by NMR, which must be mapped to the isotopic space. This mapping file is a tab-separated-values (TSV) file and is organized as shown in the following table:

Table 1: Example of input file

bcmer	HSQC_Ca	HSQC_Cb	HACO	JRES_Ha	JRES_Hb	HACO-DIPSY	GC-MS	LC-MS
000				e	t		o	g
001		k		e	u		p	h
100			r	e	t		o	h
101		k	r	e	u		p	i
010	a			f	t	m	p	h
011	b	l		f	u	m	q	i
110	c		s	f	t	n	p	i
111	d	l	s	f	u	n	q	j

where the entry meaning is following:

- each row correspond to a given binary cumomer, e.g. `010x` or `010`;
- each column corresponds to a given experiment type, e.g. `HSQC_Cb`;
- each cell contains a variable name, e.g. `a`, `b` etc. or empty, i.e. NA;
- each variable name can appear multiple times in a given column but not in different columns. If a variable appears in several rows, these rows are considered as mixed in equal parts in a given measurement method. E.g. in HSQC_Cb, contributions of patterns `001` and `101` are mixed in one variable `k`, and contributions of `011` and `111` are mixed in another one, `l`. Other rows in this column must be left empty as they don't contribute to any measurement.

In the above example, running IsoSolve will produce two files: `ALA_mapping.md` (plain text, MarkDown format) and `ALA_mapping.html`. The latter should automatically open in your browser. If not, open it manually. This output file contains formulas for definition of isotopomers, cumomers, and EMUs (ICE) depending on provided measurements. An analysis is made to determine how many of each ICE can be defined by measurements and which are still undefined. The file content is quite self-explanatory. If everything worked as expected it should be similar to ALA_mapping.html. Small differences are allowed, e.g. the redundant measurements maybe not the same as in the example file but their number (10) should coincide.

If `--inchi` option is activated, it will produce a series of additional TSV files having `_inchi_` in their names. They will contain International Chemical Identifiers (InChI) for involved isotopomers, cumomers and EMUs as well as for elementary measurable combinations. It is important to note that atom numbers used in those InChI are relative to only carbon atoms numbered from left to right in the label masks provided by user in his input file. It is up to user to renumber them according to conventional numbering appropriate to the used molecule.

For elementary measurable combinations, InChI file can contain entries with multiple isotopomers in them. To be able to put one isotopomer per row, an additional column `Group Id` is introduced. The rows having identical entries in `Group Id` must be grouped together.

For fine-tuning `isosolve` usage, read the following section about command line options. For programmatic use, see the section *API documentation*.

1.6 `isosolve` command line options

usage: isosolve.py [-h] [-c COLSEL] [-t] [-d DATA] [-w] [-s SEED] [-r] [-p PATH] [-f] [-v] mm

calculate isotopomer/cumomer/EMU expressions (symbolic and optionally numeric) by a combination of isotope labeling measurements (NMR, MS, . . .)

positional arguments:

mm file name (tsv) providing measure matrix which is organized as follows:

- each row corresponds to a given binary cumomer, e.g. '010x'
- each column corresponds to a given experiment type, e.g. 'HSQC-C_beta'
- each cell contains a variable name, e.g. 'a', 'b' etc. or empty, i.e. NA
- each variable name can appear multiple times in a given column but not in different columns. If a variable appears in several rows, these rows are considered as mixed in equal parts in a given measurement method. E.g. in HSQC-C_beta, contributions of patterns '001x' and '101x' are mixed in one variable, say 'k', and contributions of '011x' and '111x' are mixed in another one, say 'l'. Other rows in this column must be left empty as they don't contribute to any measurement.

optional arguments:

-h, --help show this help message and exit

-c COLSEL, --colsel COLSEL **column selection (full set by default).** Can be a slice, comma separated list of integers or names, regular expression, or a mix of all this. In a slice, negative number means counting from the end. In a list, if given a negative numbers or names starting with '-', the corresponding columns are excluded from treatment. Names can be given as Python regular expressions. The order of column selection is irrelevant.

Examples:

- '1:3' - first 3 columns;

- ':3' - the same;
- ':-1' - all columns but the last;
- '2::2' - even columns;
- '1,3,6' - first, third and sixth columns;
- 'HSQC.*' - columns with names started by 'HSQC';
- '-HN.*' - all columns except starting with 'HN'

-t, --TIMEME activate or not (default) CPU time printing. Useful only for debugging or issue reporting.

-d DATA, --data DATA file name with 3 columns: name, value, and sd. Numeric values in columns 'value' and 'sd' must be non-negative and positive respectively. Fields are tab-separated, comments start with sharp-sign '#'

-w, --write force .md and .html file writing even in non CLI mode

-s SEED, --seed SEED integer value used as a seed for pseudo-random drawing. Useful only for debugging or issue reporting.

-r, --rand make random draws for numerical tests of formulas. Useful only for debugging or issue reporting.

-p PATH, --path PATH path for .md and html files. If it ends by '/' it is interpreted as a directory path which is created if nonexistent. Otherwise, it is interpreted as a base before .md and .html extensions. If not given, .md and .html are written in the same directory with the same basename (before the extension) as 'MM' file

-f, --fast skip calculations of measurable combinations

-i, --inchi write InChi files

-v, --version print version number on stdout and exit. Useful only for debugging or issue reporting.

1.7 Jupyter notebook

IsoSolve can be used as a Python module that you can import directly, for instance in Jupyter notebooks or in your own software. We showcase IsoSolve usage in a Jupyter notebook distributed via this repository. It is authored by Pierre Millard (TBI/INRAE, France).

If not yet done, you can install Jupyter by:

```
pip3 install --user jupyter
```

Some dependencies are specific to our notebook, not to IsoSolve itself. If not available on your system, they can be installed with:

```
pip3 install --user seaborn matplotlib
```

Download and unpack the notebook's tarball and go in a shell to the notebook's directory.

After that, you are ready to examine and execute the notebook by launching:

```
jupyter notebook IsoSolve.ipynb
```

After launching, the notebook will open in your web browser where in each cell you can read/modify/execute a proposed code as well as read accompanying comments. Some cells can take a while to execute, so we distribute also an HTML file showing the whole notebook's output after execution. Your own output should be similar to this one.

CHAPTER 2

API documentation

2.1 isosolve module

Take symbolic measure matrix and reconstruct possible isotopomer/cumomer/emu groups (as elementary as possible, idealy each group composed of only one i/c/e) for each combination of measurement methods

class isosolve.**Object**
Bases: object

Object to which user can add fields

isosolve.**assign**(*name*, *value*, *d*)
assign a 'value' to 'name' in dictionary 'd', e.g. 'd' can be locals(). Return 'value'.

isosolve.**dhms**(*f*, *dig=2*)
Convert float number 'f' representing seconds to a 'D days Hh Mm Ss

isosolve.**e2li**(*e*)
transform expression 'e' in a nested list. Each nested level corresponds to Add, Mul and Pow arguments, Add being the highest level.

isosolve.**ech**(*m*, *d={}*, *c={}*, *sc={}*, *nd={}*, *ld={}*, *fast=False*)
reduce augmented matrix m to echelon form applying substitutions from d. Return echelon matrix and permutation list of col-index couples. No check is made for non zero terms in the last column in all zero rows.

isosolve.**echsol**(*m*, **unk*, *pe=[]*, *d={}*, *c={}*, *sc={}*, *nsub={}*, *fast=False*)
solves augmented linear system in echelon form resulted from ech(...). Unknown symbols are taken from *unk. pe is a possible permutation list of ij-tuples from ech(...). Return a symbolic column vector

isosolve.**hascumo**(*s1*, *s2*)
return a cumulated string (or None if not conformant) for binary cumomers in s1 and s2. E.g. 01x+11x => x1x

isosolve.**iainb**(*a*, *b*)
return a list of indexes of a in b or None if a is not strictly in b. An empty list is returned for empty a. Repeated values in a must be repeated at least the same number of times in b.

isosolve.**is_intstr**(*s*)
check if a string 's' represents exactly an integer number

isosolve.**isstr**(*s*)
return True if 's' is of type 'str'

isosolve.**lfact**(*e*, *esub={}*, *cache={}*, *scache={}*, *pr=False*, *nsub={}*, *fast=False*)
factorize terms in expression 'e'

isosolve.**lico2tb**(*lico*, *mdFile=None*, *text_align='center'*)
transform list of columns in lico to a flat list of strings suitable for md table writing. Write md table if mdFile is set.

isosolve.**limi**(*li*, *negative=False*)
apply or not sign '-' to all terms of the list li

isosolve.**main**(*li=None*, *mm=None*, *colsel=None*, *tim=None*, *data=None*, *w=None*, *s=None*, *rd=None*, *path=None*, *fast=False*, *vers=None*, *inchi=False*)
Resolve symbolically and numerically iso-, cumo- and EMU-mers from isotope labeling measurements (NMR, MS, ...)

Parameters

- **li** (*list, optional*) – list of argument to be processed, typically a `sys.argv[1:]`, cf. `isosolve -h` or *isosolve command line options* section for parameter significance. Defaults to None.
- **mm** (*str or pandas.FataFrame, optional*) – mapping matrix describes symbolically the mapping of each measurements method to isotopic space. Can be a file name in tsv format or a pandas data.frame. Either this parameter must be set or the only positional argument in li. Defaults to None.
- **colsel** (*str or iterable, optional*) – if str, coma separated column selection from mm. The syntax is described in `isosolve -h`. If iterable, a collection of items describing column selection. Each item can be an integer (positive or negative for exclusion), slice or string with column name. Column numbers are 1-based (not 0-based as usual for Python). Defaults to None which is equivalent to proceeding all the columns from mm.
- **tim** (*logical, optional*) – if True, print CPU time after some key steps. Defaults to None.
- **data** (*str or pandas.DataFrame, optional*) – if DataFrame, numeric values and sd of measurements described in mm. If str, a file name in tsv format with such data in three columns: name, value, sd. Defaults to None.
- **w** (*logical, optional*) – if True, write formatted results in .md (plain text, MarkDown format) and .html files. The basis of file name is the same as in mm when mm is string, or `mm` if mm is a DataFrame. The name `mm` can be overwritten with path parameter (cf. hereafter). Defaults to None which is equivalent to True when used in command line and to False when used programmatically.
- **s** (*int, optional*) – seed for pseudo random generation. When used, randomly drawn values become reproducible. Only useful for debugging or issue reporting. Defaults to None.
- **rd** (*logical, optional*) – if True, use randomly generated values for checking obtained formulas. In the output HTML file, numerical values that are not coinciding are typeset in **bold**. Only useful for debugging and issue reporting. Defaults to None.
- **path** (*str, optional*) – directory (when ends with "/") of file name (otherwise) for formatted output. Can be useful when mm is a DataFrame. Defaults to None which is equivalent to the current working directory.

- **fast** (*logical, optional*) – if True, skip calculation of elementary measurable combinations.
- **vers** (*logical, optional*) – if True, print the version number and return without any result.
- **inchi** – if True, writes TSV files with International Chemical Identifiers (InChI) for isotopomers, cumomers, EMUs and elementary measurable combinations.

Returns

dictionary with the following entries:

xbc 1-column DataFrame, randomly drawn values for isotopomers;

xv dict, key=measurement name and value=numeric value deduced from xbc;

am sympy Matrix, augmented matrix of system equations. The last column is the right hand side;

rdn_mes list, names of redundant measurements;

smm DataFrame, mapping matrix after column selection;

mmd dict, key=measurement name, value=numpy array with isotopomer indexes mapped onto this measurements;

uvals dict, key=method name (i.e. column name in mm), value=list of unique measurement names involved in this method;

sy_meas list, symbolic equations defining measurements;

sbc dict, key=isotopomer name (i.e. row name in mm), value=isotopomer symbolic variable;

sols list, symbolic solution for isotopomers. The order is the same as in mm;

vsubs dict, key=full symbolic expression, value=simplified symbolic expression. Used for simplification procedures;

metrics dict, the fields are:

idef list, defined isotopomer indexes;

idef_c set, indexes of defined cumomers;

idef_e dict, key=EMU name, value=set of defined M+i mass isotopologues in this EMU;

emusol dict, key=EMU name, value=list of symbolic solutions for M+i;

cusol dict, key=cumomer name, value=symbolic solution;

ls dict, least square solution. The fields are:

iso list, isotopomer values and sd;

cov isotopomer covariance matrix;

cumo list, cumomer values and sd;

emu list of dicts. Each dict has a key=EMU name and value=list of values (the first dict) or sd (the second one);

mec list, measurable elementary combinations values and sd;

inchi dict, InChI information. The fields are:

iso DataFrame for isotopomers;

cumo DataFrame for cumomers;

emu DataFrame for EMUs;

mcomb DataFrame for measurable elementary combinations;

Return type dict

isosolve.**main_arg**()
Wrapper for *main(li=sys.argv[1:])* call when used from executable script

isosolve.**new_header**(*level=None*, *title=''*, *mdFile=None*)

isosolve.**partcomb**(*x*, *p*)
return iterator of iterator tuples ([p distinct elements of x], [the rest len(x)-p distinct elements of x])

isosolve.**pdiff**(*expr*, *vs*)
partial derivatives of expr by vars in vs

isosolve.**revenum**(*x*)
reversed enumerated iterable x

isosolve.**rsubs**(*e*, *d*, *c={}*)
restricted substitution of keys from 'd' found in 'e' by d[key]. c is an optional cache dict

isosolve.**setsub**(*l*, *it*, *v*)
in a list l set values with indexes from it to v, i.e. l[it]=v where it is iterator. v is recycled if needed

isosolve.**sumi2sd**(*covmat*, *i*, *tol=2.842170943040401e-14*)
calculate sd for a new variable defined as sum of components in 'i'

isosolve.**timeme**(*s=''*, *dig=2*)
if a global variable TIMEME is True and another global variable _T0 is set, print current CPU time relative to _T0. This printing is preceded by a message from 's'

isosolve.**useq**(*seq*)
return iterator of unique elements in a sequence preserving initial order

The principal function to call is `isosolve.main(...)`

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

Index