lamp_quick_start

October 13, 2024

1 Quick Start

In this vignette we will demonstrate how to use `lamp` python package. The input data and reference files are located in https://github.com/wanchanglin/lamp/tree/master/examples/data.

1.1 Setup

To use `lamp`, the first step is to import some python libraries including `lamp`.

[1]:
```
import pandas as pd
from lamp import anno, stats, utils
```

1.2 Data loading

`lamp` supports text files separated by comma (`,`) or tab (`\t`). The Microsoft's XLSX is also supported, presuming that data are in the first sheet.

Here we use a small example data set with TSV format. Load it into python and have a look of data format:

[2]:
```
# data set
d_data = "./data/df_pos_2.tsv"
data = pd.read_table(d_data, header=0, sep="\t")
data
```

[2]:
```
         name     namecustom          mz       mzmin       mzmax          rt  \
0     M151T34   M150.8867T34  150.886715  150.886592  150.886863   34.152700
1     M151T40   M151.0402T40  151.040235  151.040092  151.040350   39.838172
2     M152T40   M152.0436T40  152.043607  152.043451  152.043737   40.303700
3     M153T34   M152.8838T34  152.883824  152.883678  152.883959   34.174647
4     M153T36   M153.0195T36  153.019474  153.019331  153.019633   35.785847
..        …            …          …           …           …          …
395  M283T339  M283.2646T339  283.264583  283.264341  283.264809  338.763489
396   M284T60   M284.1953T60  284.195294  284.194939  284.195536   59.593561
397  M284T108  M284.2235T108  284.223499  284.223156  284.223692  108.406389
398  M284T339   M284.268T339  284.267962  284.267634  284.268204  338.725056
399   M285T34    M284.775T34  284.775031  284.774635  284.775287   34.079641

          rtmin       rtmax  npeaks    .  …          X210          X209  \
```

```
0     33.637595   35.465548      97  97  …  4.224942e+06  3.946599e+06
1     37.556072   40.532315      95  95  …  1.419062e+06  1.251606e+06
2     38.092678   40.909428      81  81  …  1.203919e+05  9.970442e+04
3     33.637595   35.465548      98  98  …  5.592065e+06  5.761380e+06
4     34.130244   36.287354      98  98  …  7.284938e+06  1.083289e+07
..          …         …     …   ..  …           …            …
395  338.398380  339.165948      94  94  …  3.509767e+05  4.117633e+05
396   58.844217   60.107058      59  59  …           NaN           NaN
397  107.880510  108.971046      72  72  …  7.477652e+04  7.482219e+04
398  338.268300  339.370098      84  84  …  3.697604e+04  5.398264e+04
399   33.667172   35.198181      97  97  …  3.439330e+06  3.359842e+06

              X208          X207          X206          X205          X204  \
0    3.668948e+06  3.754321e+06  3.853724e+06  3.787350e+06  3.584464e+06
1    1.214826e+06  8.143028e+05  5.331963e+05  1.930928e+06  1.479001e+06
2    9.384000e+04  4.186335e+04           NaN  2.115447e+05  1.285713e+05
3    5.845419e+06  5.576013e+06  5.552878e+06  6.132789e+06  5.891378e+06
4    1.140072e+07  8.220552e+06  9.255154e+06  7.648211e+06  7.723814e+06
..            …             …             …             …             …
395  3.948000e+05  4.338804e+05  5.335221e+05  6.224684e+05  7.009340e+05
396           NaN           NaN           NaN  2.558004e+04  4.020517e+04
397  3.399667e+04  7.233564e+04  1.043879e+05  2.506785e+04  2.753769e+04
398  5.340109e+04  6.557698e+04  7.656575e+04  1.040606e+05  1.063727e+05
399  3.375577e+06  3.789056e+06  3.478506e+06  3.391588e+06  5.067802e+06

              X203          X202          X201
0    3.499711e+06  3.623205e+06  4.145770e+06
1    1.076354e+06  9.293218e+05  5.298062e+05
2    9.389346e+04  7.163655e+04  4.916483e+04
3    5.418082e+06  5.036840e+06  5.733794e+06
4    5.571163e+06  5.362560e+06  9.259675e+06
..            …             …             …
395  3.005173e+05  3.133173e+05  8.204783e+05
396           NaN  3.162670e+04  5.446684e+04
397           NaN           NaN           NaN
398           NaN  3.059370e+04  1.358056e+05
399  3.497546e+06  3.316025e+06  3.906000e+06

[400 rows x 110 columns]
```

This data set includes peak list and intensity data matrix. `lamp` will use peak list's name, m/z value and retention time. Hence you needs to indicates the locations of peak name, m/z value, retention time and starting points of data matrix from input data. Here they are 1, 3, 6 and 11, respectively.

[3]:
```
cols = [1, 3, 6, 11]
# get the input data set for `lamp`
```

```
df = anno.read_peak(d_data, cols, sep='\t')
df
```

[3]:

	name	mz	rt	QC9	QC5	\
0	M151T34	150.886715	34.152700	3.664879e+06	3.735147e+06	
1	M151T40	151.040235	39.838172	7.406381e+05	7.524075e+05	
2	M152T40	152.043607	40.303700	6.105241e+04	5.335546e+04	
3	M153T34	152.883824	34.174647	5.141479e+06	5.496344e+06	
4	M153T36	153.019474	35.785847	5.336758e+06	5.558265e+06	
..	…	…	…	…	…	
395	M283T339	283.264583	338.763489	7.330602e+05	8.243956e+05	
396	M284T60	284.195294	59.593561	2.310932e+04	NaN	
397	M284T108	284.223499	108.406389	3.748444e+04	2.993283e+04	
398	M284T339	284.267962	338.725056	1.161886e+05	1.476514e+05	
399	M285T34	284.775031	34.079641	4.063268e+06	3.807148e+06	

	QC4	QC3	QC26	QC25	QC24	\
0	5.190263e+06	2.742966e+06	3.824723e+06	3.722932e+06	3.804188e+06	
1	NaN	6.429245e+05	1.167016e+06	1.175981e+06	1.122533e+06	
2	NaN	NaN	6.875157e+04	7.807399e+04	8.943068e+04	
3	8.335846e+06	3.860588e+06	5.316874e+06	5.988232e+06	5.844917e+06	
4	1.118557e+07	6.876715e+06	9.967314e+06	9.073822e+06	9.328573e+06	
..	…	…	…	…	…	
395	NaN	1.159506e+06	4.294760e+05	4.641813e+05	4.570657e+05	
396	NaN	NaN	1.759336e+04	2.645392e+04	2.727266e+04	
397	NaN	NaN	3.175596e+04	3.879604e+04	4.299529e+04	
398	NaN	NaN	NaN	6.753490e+04	5.436219e+04	
399	4.645099e+06	2.232221e+06	4.576754e+06	4.533339e+06	4.559356e+06	

	…	X210	X209	X208	X207	\
0	…	4.224942e+06	3.946599e+06	3.668948e+06	3.754321e+06	
1	…	1.419062e+06	1.251606e+06	1.214826e+06	8.143028e+05	
2	…	1.203919e+05	9.970442e+04	9.384000e+04	4.186335e+04	
3	…	5.592065e+06	5.761380e+06	5.845419e+06	5.576013e+06	
4	…	7.284938e+06	1.083289e+07	1.140072e+07	8.220552e+06	
..	…	…	…	…	…	
395	…	3.509767e+05	4.117633e+05	3.948000e+05	4.338804e+05	
396	…	NaN	NaN	NaN	NaN	
397	…	7.477652e+04	7.482219e+04	3.399667e+04	7.233564e+04	
398	…	3.697604e+04	5.398264e+04	5.340109e+04	6.557698e+04	
399	…	3.439330e+06	3.359842e+06	3.375577e+06	3.789056e+06	

	X206	X205	X204	X203	X202	\
0	3.853724e+06	3.787350e+06	3.584464e+06	3.499711e+06	3.623205e+06	
1	5.331963e+05	1.930928e+06	1.479001e+06	1.076354e+06	9.293218e+05	
2	NaN	2.115447e+05	1.285713e+05	9.389346e+04	7.163655e+04	
3	5.552878e+06	6.132789e+06	5.891378e+06	5.418082e+06	5.036840e+06	

```
4    9.255154e+06  7.648211e+06  7.723814e+06  5.571163e+06  5.362560e+06
..            …             …             …             …             …
395  5.335221e+05  6.224684e+05  7.009340e+05  3.005173e+05  3.133173e+05
396           NaN  2.558004e+04  4.020517e+04           NaN  3.162670e+04
397  1.043879e+05  2.506785e+04  2.753769e+04           NaN           NaN
398  7.656575e+04  1.040606e+05  1.063727e+05           NaN  3.059370e+04
399  3.478506e+06  3.391588e+06  5.067802e+06  3.497546e+06  3.316025e+06

             X201
0    4.145770e+06
1    5.298062e+05
2    4.916483e+04
3    5.733794e+06
4    9.259675e+06
..            …
395  8.204783e+05
396  5.446684e+04
397           NaN
398  1.358056e+05
399  3.906000e+06

[400 rows x 103 columns]
```

Data frame `df` now includes only `name`, `mz`, `rt` and intensity data matrix.

1.3 Metabolite annotation

To performance metabolite annotation, users should provide their own reference file. Otherwise, `lamp` will use its default reference file for annotation.

[4]:
```
ref_path = ""    # if empty, use default reference file for matching

# load reference library
cal_mass = False
ref = anno.read_ref(ref_path, calc=cal_mass)
ref
```

[4]:
```
       compound_id molecular_formula                       compound_name  \
0             1638          C10Cl10O                          Chlordecone
1            38485       C10H10Br2O2                 Dibromothymoquinone
2            32427       C10H10BrNO2                Brofoxine (USAN/INN)
3            39834      C10H10Cl2N2O                  Fenmetozole (USAN)
4            10156       C10H10Cl2O3  4-(2,4-Dichlorophenoxy)butyric acid
…              …                 …                                   …
31639        80256          H5O10P3                                PPPi
31640        37374          H6NO9P3   (Diphosphono)Aminophosphonic Acid
31641        32626          H9N2O4P             Ammonium phosphate (NF)
31642          735             HNO3                             Nitrate
```

```
31643        40762              HNO3                          Peroxynitrite

       exact_mass
0       485.683441
1       319.904755
2       254.989491
3       244.017018
4       248.000700
…              …
31639   257.909557
31640   256.925542
31641   132.029994
31642    62.995643
31643    62.995643

[31644 rows x 4 columns]
```

The reference file must have two columns: `molecular_formula` and `compound_name` (or `name`). The `exact_mass` is optional. if absent, `lamp` will calculates it based on NIST database. If your reference file has `exact_mass` and want to calculate it using NIST database, set `calc` as True. The `exact_mass` is used to match against a range of `mz`, controlled by `ppm` in data frame `df`.

Now we have a look another reference file:

[5]:
```
ref_path = "./data/hmdb_urine_v4_0_20200910_v1.tsv"

# load reference library
cal_mass = True    # there is no exact mass in reference file, so calculate
ref = anno.read_ref(ref_path, calc=cal_mass)
ref
```

[5]:
```
                id molecular_formula                    molecular_name  \
0     HMDB0000001         C7H11N3O2                 1-Methylhistidine
1     HMDB0000002           C3H10N2                1,3-Diaminopropane
2     HMDB0000005            C4H6O3                2-Ketobutyric acid
3     HMDB0000008            C4H8O3             2-Hydroxybutyric acid
4     HMDB0000010          C19H24O3                  2-Methoxyestrone
…             …                 …                                 …
1606  HMDB0012308            C8H8O3                       Vanillin
1607  HMDB0012322           C10H8O                       2-Naphthol
1608  HMDB0012325           C5H10O5                   Arabinofuranose
1609  HMDB0012451          C20H28O3  all-trans-5,6-Epoxyretinoic acid
1610  HMDB0012467         C15H13O9S           (-)-Epicatechin sulfate

                                                  inchi  \
0     InChI=1S/C7H11N3O2/c1-10-3-5(9-4-10)2-6(8)7(11…
1                   InChI=1S/C3H10N2/c4-2-1-3-5/h1-5H2
2       InChI=1S/C4H6O3/c1-2-3(5)4(6)7/h2H2,1H3,(H,6,7)
```

```
3     InChI=1S/C4H8O3/c1-2-3(5)4(6)7/h3,5H,2H2,1H3,(…
4     InChI=1S/C19H24O3/c1-19-8-7-12-13(15(19)5-6-18…
…                                                   …
1606  InChI=1S/C8H8O3/c1-11-8-4-6(5-9)2-3-7(8)10/h2-…
1607  InChI=1S/C10H8O/c11-10-6-5-8-3-1-2-4-9(8)7-10/…
1608  InChI=1S/C5H10O5/c6-1-2-3(7)4(8)5(9)10-2/h2-9H…
1609  InChI=1S/C20H28O3/c1-15(8-6-9-16(2)14-17(21)22…
1610  InChI=1S/C15H14O9S/c16-9-3-8-5-13(24-25(20,21)…

                        inchi_key  exact_mass
0     BRMWTNUJHUMWMS-LURJTMIESA-N  169.085127
1     XFNJVJPLKCPIBV-UHFFFAOYSA-N   74.084398
2     TYEYBOSBBBHJIV-UHFFFAOYSA-N  102.031694
3     AFENDNXGAFYKQO-VKHMYHEASA-N  104.047344
4     WHEUWNKSCXYKBU-QPWUGHHJSA-N  300.172545
…                             …          …
1606  MWOOGOJBHIARFG-UHFFFAOYSA-N  152.047344
1607  JWAZRIHNYRIHIV-UHFFFAOYSA-N  144.057515
1608  HMFHBZSHGGEWLO-HWQSCIPKSA-N  150.052823
1609  KEEHJLBAOLGBJZ-WEDZBJJJSA-N  316.203845
1610  WTXWEAXATVSZQX-AFYYWNPRSA-M  369.028028

[1611 rows x 6 columns]
```

Next we use HMDB reference file for compounds match. Here function argument `ppm` is used to control the m/z value matching tolerance or range.

[6]:
```
ppm = 5.0
match = anno.comp_match_mass(df, ppm, ref)
match
```

[6]:
```
           id          mz molecular_formula             molecular_name  \
0     M154T37  154.062402           C8H10O3              Hydroxytyrosol
1    M164T119  164.046774            C9H8O3         Phenylpyruvic acid
2    M164T119  164.046774            C9H8O3            m-Coumaric acid
3    M164T119  164.046774            C9H8O3     4-Hydroxycinnamic acid
4    M164T119  164.046774            C9H8O3     2-Hydroxycinnamic acid
5    M164T233  164.046832            C9H8O3         Phenylpyruvic acid
6    M164T233  164.046832            C9H8O3            m-Coumaric acid
7    M164T233  164.046832            C9H8O3     4-Hydroxycinnamic acid
8    M164T233  164.046832            C9H8O3     2-Hydroxycinnamic acid
9     M164T53  164.046825            C9H8O3         Phenylpyruvic acid
10    M164T53  164.046825            C9H8O3            m-Coumaric acid
11    M164T53  164.046825            C9H8O3     4-Hydroxycinnamic acid
12    M164T53  164.046825            C9H8O3     2-Hydroxycinnamic acid
13    M167T35  167.021095            C7H5NO4            Quinolinic acid
14  M173T36_3  173.104423           C8H15NO3            Hexanoylglycine
15    M174T35  174.088395            C8H14O4               Suberic acid
```

```
16    M181T36  181.060407          C6H7N5O2  8-Hydroxy-7-methylguanine
17    M212T39  212.067866          C10H12O5           Vanillactic acid
18    M276T36  276.077397       C10H16N2O5S             Biotin sulfone

                                                  inchi  \
0   InChI=1S/C8H10O3/c9-4-3-6-1-2-7(10)8(11)5-6/h1…
1   InChI=1S/C9H8O3/c10-8(9(11)12)6-7-4-2-1-3-5-7/…
2   InChI=1S/C9H8O3/c10-8-3-1-2-7(6-8)4-5-9(11)12/…
3   InChI=1S/C9H8O3/c10-8-4-1-7(2-5-8)3-6-9(11)12/…
4   InChI=1S/C9H8O3/c10-8-4-2-1-3-7(8)5-6-9(11)12/…
5   InChI=1S/C9H8O3/c10-8(9(11)12)6-7-4-2-1-3-5-7/…
6   InChI=1S/C9H8O3/c10-8-3-1-2-7(6-8)4-5-9(11)12/…
7   InChI=1S/C9H8O3/c10-8-4-1-7(2-5-8)3-6-9(11)12/…
8   InChI=1S/C9H8O3/c10-8-4-2-1-3-7(8)5-6-9(11)12/…
9   InChI=1S/C9H8O3/c10-8(9(11)12)6-7-4-2-1-3-5-7/…
10  InChI=1S/C9H8O3/c10-8-3-1-2-7(6-8)4-5-9(11)12/…
11  InChI=1S/C9H8O3/c10-8-4-1-7(2-5-8)3-6-9(11)12/…
12  InChI=1S/C9H8O3/c10-8-4-2-1-3-7(8)5-6-9(11)12/…
13  InChI=1S/C7H5NO4/c9-6(10)4-2-1-3-8-5(4)7(11)12…
14  InChI=1S/C8H15NO3/c1-2-3-4-5-7(10)9-6-8(11)12/…
15  InChI=1S/C8H14O4/c9-7(10)5-3-1-2-4-6-8(11)12/h…
16  InChI=1S/C6H7N5O2/c1-11-2-3(9-6(11)13)8-5(7)10…
17  InChI=1S/C10H12O5/c1-15-9-5-6(2-3-7(9)11)4-8(1…
18  InChI=1S/C10H16N2O5S/c13-8(14)4-2-1-3-7-9-6(5-…

                      inchi_key  exact_mass  ppm_error
0   JUUBCHWRXWPFFH-UHFFFAOYSA-N      154.06      -3.84
1   BTNMPGBKDVTSJY-UHFFFAOYSA-N      164.05      -3.47
2   KKSDGJDHHZEWEP-SNAWJCMRSA-N      164.05      -3.47
3   NGSWKAQJJWESNS-ZZXKWVIFSA-N      164.05      -3.47
4   PMOWTIHVNWZYFI-AATRIKPKSA-N      164.05      -3.47
5   BTNMPGBKDVTSJY-UHFFFAOYSA-N      164.05      -3.12
6   KKSDGJDHHZEWEP-SNAWJCMRSA-N      164.05      -3.12
7   NGSWKAQJJWESNS-ZZXKWVIFSA-N      164.05      -3.12
8   PMOWTIHVNWZYFI-AATRIKPKSA-N      164.05      -3.12
9   BTNMPGBKDVTSJY-UHFFFAOYSA-N      164.05      -3.16
10  KKSDGJDHHZEWEP-SNAWJCMRSA-N      164.05      -3.16
11  NGSWKAQJJWESNS-ZZXKWVIFSA-N      164.05      -3.16
12  PMOWTIHVNWZYFI-AATRIKPKSA-N      164.05      -3.16
13  GJAWHXHKYYXBSV-UHFFFAOYSA-N      167.02      -4.57
14  UPCKIPHSXMXJOX-UHFFFAOYSA-N      173.11      -4.45
15  TYFQFVWCELRYAO-UHFFFAOYSA-N      174.09      -4.67
16  VHPXSVXJBWZORQ-UHFFFAOYSA-N      181.06       2.39
17  SVYIZYRTOYHQRE-UHFFFAOYSA-N      212.07      -2.86
18  QPFQYMONYBAUCY-ZKWXMUAHSA-N      276.08      -2.16
```

`match` gives the compound matching results. `lamp` also provides a mass adjust option by adduct library. You can provide your own adducts library otherwise `lamp` uses its default adducts library.

The adducts library looks like:

[7]:
```
add_path = './data/adducts_short.tsv'
lib_df = pd.read_csv(add_path, sep="\t")
lib_df
```

[7]:

	label	exact_mass	charge	ion_mode
0	[M+H]+	1.007276	1	pos
1	[M+NH4]+	18.033826	1	pos
2	[M+Na]+	22.989221	1	pos
3	[M+Mg]+	23.984493	1	pos
4	[M+K]+	38.963158	1	pos
5	[M+Fe]+	55.934388	1	pos
6	[M+Cu]+	62.929049	1	pos
7	[M+2H]+	2.015101	1	pos
8	[M+3H]+	3.022926	1	pos
9	[M-H]-	-1.007276	1	neg
10	[M+35Cl]-	34.969401	1	neg
11	[M+Formate]-	44.998203	1	neg
12	[M+Acetate]-	59.013853	1	neg

We use this addcuts file to adjust mass:

[8]:
```
ion_mode = "pos"
# if empty, use default adducts library
add_path = "./data/adducts_short.tsv"

lib_add = anno.read_lib(add_path, ion_mode)
lib_add
```

[8]:

	label	exact_mass	charge
0	[M+H]+	1.007276	1
1	[M+NH4]+	18.033826	1
2	[M+Na]+	22.989221	1
3	[M+Mg]+	23.984493	1
4	[M+K]+	38.963158	1
5	[M+Fe]+	55.934388	1
6	[M+Cu]+	62.929049	1
7	[M+2H]+	2.015101	1
8	[M+3H]+	3.022926	1

Now use this function to match compounds:

[9]:
```
match_1 = anno.comp_match_mass_add(df, ppm, ref, lib_add)
match_1
```

[9]:

	id	mz	molecular_formula	molecular_name	\
0	M152T40	152.043607	C5H8N2O2	Dihydrothymine	
1	M154T37	154.062402	C8H8O3	p-Hydroxyphenylacetic acid	

2	M154T37	154.062402	C8H8O3	3-Hydroxyphenylacetic acid
3	M154T37	154.062402	C8H8O3	ortho-Hydroxyphenylacetic acid
4	M154T37	154.062402	C8H8O3	Mandelic acid
5	M154T37	154.062402	C8H8O3	3-Cresotinic acid
6	M154T37	154.062402	C8H8O3	4-Hydroxy-3-methylbenzoic acid
7	M154T37	154.062402	C8H8O3	Vanillin
8	M157T35	157.036819	C4H10N2O2	2,4-Diaminobutyric acid
9	M157T35	157.036819	C4H10N2O2	L-2,4-diaminobutyric acid
10	M167T35	167.021095	C5H8N2O2	Dihydrothymine
11	M174T35	174.088395	C9H13NO	Phenylpropanolamine
12	M174T35	174.088395	C10H14O	Thymol
13	M174T35	174.088395	C10H14O	(S)-Carvone
14	M174T35	174.088395	C8H12O4	2-Octenedioic acid
15	M174T35	174.088395	C8H12O4	cis-4-Octenedioic acid
16	M181T36	181.060407	C8H8N2O3	Nicotinuric acid
17	M184T38	184.097942	C10H13N2	Nicotine imine
18	M185T39_2	185.082034	C5H15NO4P	Phosphorylcholine
19	M186T36	186.045606	C6H14N2O	N-Acetylputrescine
20	M187T38	187.097642	C5H15NO4P	Phosphorylcholine
21	M193T40	193.050761	C5H14N4	Agmatine
22	M200T36	200.061328	C7H16N2O	N-Acetylcadaverine
23	M201T39_1	201.051849	C10H10O3	4-Methoxycinnamic acid
24	M203T36_1	203.002108	C9H9NO	Indole-3-carbinol
25	M212T39	212.067866	C8H15NO3	Hexanoylglycine
26	M212T39	212.067866	C10H10O5	Vanilpyruvic acid
27	M217T37_1	217.018279	C10H11NO	Tryptophol
28	M221T37	221.012328	C9H11NO2	L-Phenylalanine
29	M223T38	223.008162	C4H10NO6P	O-Phosphothreonine
30	M223T40	223.096863	C12H14O4	Monoisobutyl phthalic acid
31	M226T44	226.128007	C8H18N4O2	Asymmetric dimethylarginine
32	M226T44	226.128007	C8H18N4O2	Symmetric dimethylarginine
33	M227T36	227.066175	C9H10N2O5	3-Nitrotyrosine
34	M229T38	229.069418	C4H10N3O5P	Phosphocreatine
35	M233T38	233.043479	C8H10N4O2	Caffeine
36	M245T44	245.045772	C7H15N3O3	Homocitrulline
37	M245T37_2	245.093315	C13H18O2	Ibuprofen
38	M249T38	249.038309	C8H10N4O3	1,3,7-Trimethyluric acid
39	M261T43	260.972975	C10H7NO4	Xanthurenic acid
40	M269T37_2	269.088048	C10H12N4O5	Inosine
41	M275T168	275.201932	C18H24O2	Estradiol
42	M275T168	275.201932	C18H24O2	17a-Estradiol
43	M277T181	277.217564	C18H28O2	19-Norandrosterone
44	M277T181	277.217564	C18H28O2	19-Noretiocholanolone
45	M278T71	278.148195	C11H20N2O6	Saccharopine
46	M279T233	279.233232	C18H30O2	alpha-Linolenic acid
47	M279T233	279.233232	C18H28O2	19-Norandrosterone
48	M279T233	279.233232	C18H28O2	19-Noretiocholanolone

49	M281T287	281.248903	C18H32O2	Linoleic acid
50	M281T287	281.248903	C18H30O2	alpha-Linolenic acid
51	M282T61	282.070271	C10H14N2O6	Ribothymidine
52	M282T61	282.070271	C10H14N2O6	3-Methyluridine
53	M283T37	283.103695	C11H14N4O5	1-Methylinosine

	inchi \
0	InChI=1S/C5H8N2O2/c1-3-2-6-5(9)7-4(3)8/h3H,2H2…
1	InChI=1S/C8H8O3/c9-7-3-1-6(2-4-7)5-8(10)11/h1-…
2	InChI=1S/C8H8O3/c9-7-3-1-2-6(4-7)5-8(10)11/h1-…
3	InChI=1S/C8H8O3/c9-7-4-2-1-3-6(7)5-8(10)11/h1-…
4	InChI=1S/C8H8O3/c9-7(8(10)11)6-4-2-1-3-5-6/h1-…
5	InChI=1S/C8H8O3/c1-5-3-2-4-6(7(5)9)8(10)11/h2-…
6	InChI=1S/C8H8O3/c1-5-4-6(8(10)11)2-3-7(5)9/h2-…
7	InChI=1S/C8H8O3/c1-11-8-4-6(5-9)2-3-7(8)10/h2-…
8	InChI=1S/C4H10N2O2/c5-2-1-3(6)4(7)8/h3H,1-2,5-…
9	InChI=1S/C4H10N2O2/c5-2-1-3(6)4(7)8/h3H,1-2,5-…
10	InChI=1S/C5H8N2O2/c1-3-2-6-5(9)7-4(3)8/h3H,2H2…
11	InChI=1S/C9H13NO/c1-7(10)9(11)8-5-3-2-4-6-8/h2…
12	InChI=1S/C10H14O/c1-7(2)9-5-4-8(3)6-10(9)11/h4…
13	InChI=1S/C10H14O/c1-7(2)9-5-4-8(3)10(11)6-9/h4…
14	InChI=1S/C8H12O4/c9-7(10)5-3-1-2-4-6-8(11)12/h…
15	InChI=1S/C8H12O4/c9-7(10)5-3-1-2-4-6-8(11)12/h…
16	InChI=1S/C8H8N2O3/c11-7(12)5-10-8(13)6-2-1-3-9…
17	InChI=1S/C10H13N2/c1-12-7-3-5-10(12)9-4-2-6-11…
18	InChI=1S/C5H14NO4P/c1-6(2,3)4-5-10-11(7,8)9/h4…
19	InChI=1S/C6H14N2O/c1-6(9)8-5-3-2-4-7/h2-5,7H2,…
20	InChI=1S/C5H14NO4P/c1-6(2,3)4-5-10-11(7,8)9/h4…
21	InChI=1S/C5H14N4/c6-3-1-2-4-9-5(7)8/h1-4,6H2,(…
22	InChI=1S/C7H16N2O/c1-7(10)9-6-4-2-3-5-8/h2-6,8…
23	InChI=1S/C10H10O3/c1-13-9-5-2-8(3-6-9)4-7-10(1…
24	InChI=1S/C9H9NO/c11-6-7-5-10-9-4-2-1-3-8(7)9/h…
25	InChI=1S/C8H15NO3/c1-2-3-4-5-7(10)9-6-8(11)12/…
26	InChI=1S/C10H10O5/c1-15-9-5-6(2-3-7(9)11)4-8(1…
27	InChI=1S/C10H11NO/c12-6-5-8-7-11-10-4-2-1-3-9(…
28	InChI=1S/C9H11NO2/c10-8(9(11)12)6-7-4-2-1-3-5-…
29	InChI=1S/C4H10NO6P/c1-2(3(5)4(6)7)11-12(8,9)10…
30	InChI=1S/C12H14O4/c1-8(2)7-16-12(15)10-6-4-3-5…
31	InChI=1S/C8H18N4O2/c1-12(2)8(10)11-5-3-4-6(9)7…
32	InChI=1S/C8H18N4O2/c1-10-8(11-2)12-5-3-4-6(9)7…
33	InChI=1S/C9H10N2O5/c10-6(9(13)14)3-5-1-2-8(12)…
34	InChI=1S/C4H10N3O5P/c1-7(2-3(8)9)4(5)6-13(10,1…
35	InChI=1S/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)…
36	InChI=1S/C7H15N3O3/c8-5(6(11)12)3-1-2-4-10-7(9…
37	InChI=1S/C13H18O2/c1-9(2)8-11-4-6-12(7-5-11)10…
38	InChI=1S/C8H10N4O3/c1-10-4-5(9-7(10)14)11(2)8(…
39	InChI=1S/C10H7NO4/c12-7-3-1-2-5-8(13)4-6(10(14…

40	InChI=1S/C10H12N4O5/c15-1-4-6(16)7(17)10(19-4)…
41	InChI=1S/C18H24O2/c1-18-9-8-14-13-5-3-12(19)10…
42	InChI=1S/C18H24O2/c1-18-9-8-14-13-5-3-12(19)10…
43	InChI=1S/C18H28O2/c1-18-9-8-14-13-5-3-12(19)10…
44	InChI=1S/C18H28O2/c1-18-9-8-14-13-5-3-12(19)10…
45	InChI=1S/C11H20N2O6/c12-7(10(16)17)3-1-2-6-13-…
46	InChI=1S/C18H30O2/c1-2-3-4-5-6-7-8-9-10-11-12-…
47	InChI=1S/C18H28O2/c1-18-9-8-14-13-5-3-12(19)10…
48	InChI=1S/C18H28O2/c1-18-9-8-14-13-5-3-12(19)10…
49	InChI=1S/C18H32O2/c1-2-3-4-5-6-7-8-9-10-11-12-…
50	InChI=1S/C18H30O2/c1-2-3-4-5-6-7-8-9-10-11-12-…
51	InChI=1S/C10H14N2O6/c1-4-2-12(10(17)11-8(4)16)…
52	InChI=1S/C10H14N2O6/c1-11-6(14)2-3-12(10(11)17…
53	InChI=1S/C11H14N4O5/c1-14-3-13-9-6(10(14)19)12…

	inchi_key	exact_mass	adduct	ppm_error
0	NBAKTGXDIBVZOO-VKHMYHEASA-N	152.04	[M+Mg]+	3.52
1	XQXPVVBIMDBYFF-UHFFFAOYSA-N	154.06	[M+2H]+	-0.28
2	FVMDYYGIDFPZAX-UHFFFAOYSA-N	154.06	[M+2H]+	-0.28
3	CCVYRRGZDBSHFU-UHFFFAOYSA-N	154.06	[M+2H]+	-0.28
4	IWYDHOAUDWTVEP-ZETCQYMHSA-N	154.06	[M+2H]+	-0.28
5	WHSXTWFYRGOBGO-UHFFFAOYSA-N	154.06	[M+2H]+	-0.28
6	LTFHNKUKQYVHDX-UHFFFAOYSA-N	154.06	[M+2H]+	-0.28
7	MWOOGOJBHIARFG-UHFFFAOYSA-N	154.06	[M+2H]+	-0.28
8	OGNSCSPNOLGXSM-UHFFFAOYSA-N	157.04	[M+K]+	-3.61
9	OGNSCSPNOLGXSM-VKHMYHEASA-N	157.04	[M+K]+	-3.61
10	NBAKTGXDIBVZOO-VKHMYHEASA-N	167.02	[M+K]+	-3.83
11	DLNKOYKMWOXYQA-VXNVDRBHSA-N	174.09	[M+Na]+	-3.10
12	MGSRCZKZVOBKFT-UHFFFAOYSA-N	174.09	[M+Mg]+	-3.23
13	ULDHMXUKGWMISQ-VIFPVBQESA-N	174.09	[M+Mg]+	-3.23
14	BNTPVRGYUHJFHN-HWKANZROSA-N	174.09	[M+2H]+	-1.52
15	LQVYKEXVMZXOAH-UPHRSURJSA-N	174.09	[M+2H]+	-1.52
16	ZBSGKPYXQINNGF-UHFFFAOYSA-N	181.06	[M+H]+	-1.99
17	GTQXYYYOJZZJHL-UHFFFAOYSA-N	184.10	[M+Na]+	4.61
18	YHHSONZFOIEMCP-UHFFFAOYSA-O	185.08	[M+H]+	4.80
19	KLZGKIDSEJWEDW-UHFFFAOYSA-N	186.05	[M+Fe]+	3.25
20	YHHSONZFOIEMCP-UHFFFAOYSA-O	187.10	[M+3H]+	4.52
21	QYPPJABKJHAVHS-UHFFFAOYSA-N	193.05	[M+Cu]+	-0.69
22	RMOIHHAKNOFHOE-UHFFFAOYSA-N	200.06	[M+Fe]+	3.39
23	AFDXODALSZRGIH-QPJJXVBHSA-N	201.05	[M+Na]+	-1.82
24	IVYPNXXAYMYVSP-UHFFFAOYSA-N	203.00	[M+Fe]+	-3.42
25	UPCKIPHSXMXJOX-UHFFFAOYSA-N	212.07	[M+K]+	-2.29
26	YGQHQTMRZPHIBB-UHFFFAOYSA-N	212.07	[M+2H]+	-0.28
27	MBBOMCVGYCRMEA-UHFFFAOYSA-N	217.02	[M+Fe]+	-0.79
28	COLNVLDHVKWLRT-QMMMGPOBSA-N	221.01	[M+Fe]+	-4.70
29	USRGIUJOYOXOQJ-GBXIJSLDSA-N	223.01	[M+Mg]+	-4.06
30	RZJSUWQGFCHNFS-UHFFFAOYSA-N	223.10	[M+H]+	1.69

31	YDGMGEXADBMOMJ-LURJTMIESA-N	226.13	[M+Mg]+	2.38
32	HVPFXCBJHIIJGS-LURJTMIESA-N	226.13	[M+Mg]+	2.38
33	FBTSQILOGYXGMD-LURJTMIESA-N	227.07	[M+H]+	-0.32
34	DRBBFCLWYRJSJZ-UHFFFAOYSA-N	229.07	[M+NH4]+	-0.94
35	RYYVLZVUVIJVGH-UHFFFAOYSA-N	233.04	[M+K]+	-0.23
36	XIGSAGMEBXLVJJ-YFKPBYRVSA-N	245.05	[M+Fe]+	0.17
37	HEFNNWSXXWATRW-UHFFFAOYSA-N	245.09	[M+K]+	-2.13
38	BYXCFUMGEBZDDI-UHFFFAOYSA-N	249.04	[M+K]+	-0.56
39	FBZONXHGGPHHIY-UHFFFAOYSA-N	260.97	[M+Fe]+	4.13
40	UGQMRVRMYYASKQ-KQYNXXCUSA-N	269.09	[M+H]+	0.01
41	VOXZDWNPVJITMN-ZBRFXRBCSA-N	275.20	[M+3H]+	5.00
42	VOXZDWNPVJITMN-SFFUCWETSA-N	275.20	[M+3H]+	5.00
43	UOUIARGWRPHDBX-CQZDKXCPSA-N	277.22	[M+H]+	4.90
44	UOUIARGWRPHDBX-DHMVHTBWSA-N	277.22	[M+H]+	4.90
45	ZDGJAHTZVHVLOT-YUMQZZPRSA-N	278.15	[M+2H]+	3.44
46	DTOSIQBPPRVQHS-PDBXOOCHSA-N	279.23	[M+H]+	4.93
47	UOUIARGWRPHDBX-CQZDKXCPSA-N	279.23	[M+3H]+	4.93
48	UOUIARGWRPHDBX-DHMVHTBWSA-N	279.23	[M+3H]+	4.93
49	OYHQOLUKZRVURQ-HZJYTTRNSA-N	281.25	[M+H]+	4.97
50	DTOSIQBPPRVQHS-PDBXOOCHSA-N	281.25	[M+3H]+	4.97
51	DWRXFEITVBNRMK-JXOAFFINSA-N	282.07	[M+Mg]+	2.10
52	UTQUILVPBZEHTK-UHFFFAOYSA-N	282.07	[M+Mg]+	2.10
53	WJNGQIYEQLPJMN-IOSLPCCCSA-N	283.10	[M+H]+	-0.01

1.4 Correlation analysis

Next step is correlation analysis, based on intensity data matrix along all peaks. All results are filtered by the correlation coefficient, p-values and retention time difference. That is: keep correlation results in an retention time differences/windows(such as 1 seconds) with correlation coefficient larger than a threshold(such as 0.5) and their correlation p-values less than a threshold (such as 0.05).

`lamp` uses one of correlation methods, either `pearson` or `spearman`. Also parameter `positive` allows user to select only positive correlation results.

Two functions, `_tic` and `_toc`, record the correlation computation time in seconds.

[10]:
```
thres_rt = 1.0
thres_corr = 0.5
thres_pval = 0.05
method = "spearman"    # "pearson"
positive = True

utils._tic()
corr = stats.comp_corr_rt(df, thres_rt, thres_corr, thres_pval, method,
                          positive)
utils._toc()
corr
```

```
Elapsed time: 4.283773899078369 seconds.
```

[10]:

	name_a	name_b	r_value	p_value	rt_diff
0	M151T34	M153T34	0.80	1.267076e-23	0.02
1	M151T34	M155T34	0.71	1.752854e-16	0.20
2	M151T34	M161T34	0.78	1.869949e-21	0.14
3	M151T34	M163T34	0.69	3.239594e-15	0.20
4	M151T34	M167T35	0.51	5.776482e-08	0.73
…	…	…	…	…	…
1783	M283T34_1	M283T34_2	0.62	4.214876e-12	0.29
1784	M283T34_1	M285T34	0.82	5.937139e-26	0.08
1785	M283T34_2	M285T34	0.66	7.898957e-14	0.37
1786	M283T60	M284T60	0.86	1.033010e-29	0.15
1787	M283T339	M284T339	0.91	4.031333e-39	0.04

[1788 rows x 5 columns]

Based on the correlation analysis, we can extract the groups and their size by:

[11]:
```
# get correlation group and size
corr_df = stats.corr_grp_size(corr)
corr_df
```

[11]:

	name	cor_grp_size	cor_grp
0	M219T35	52	M221T34::M223T34::M225T35::M226T35::M229T34::M…
1	M216T35	52	M217T35::M218T35::M219T34::M219T35::M221T34::M…
2	M217T35	52	M218T35::M219T34::M219T35::M221T34::M223T34::M…
3	M215T35	52	M216T35::M217T35::M218T35::M219T34::M219T35::M…
4	M218T35	51	M219T34::M219T35::M221T34::M223T34::M225T35::M…
..	…	…	…
335	M173T119	1	M171T119
336	M277T71	1	M278T71
337	M259T233	1	M191T233
338	M284T60	1	M283T60
339	M266T66	1	M265T66

[340 rows x 3 columns]

1.5 Summarize results

The final step gets the summary table in different format and save for the further analysis.

[12]:
```
# get summary of metabolite annotation
sr, mr = anno.comp_summ(df, match)
```

This function combines peak table with compound matching results and returns two results in different formats. `sr` is single row results for each peak id in peak table `df`:

[13]:
```
sr
```

[13]:
```
          name          mz          rt  exact_mass  ppm_error  \
0     M151T34  150.886715   34.152700         NaN        NaN
1     M151T40  151.040235   39.838172         NaN        NaN
2     M152T40  152.043607   40.303700         NaN        NaN
3     M153T34  152.883824   34.174647         NaN        NaN
4     M153T36  153.019474   35.785847         NaN        NaN
..        …         …          …          …         …
395   M283T61  283.068474   60.739869         NaN        NaN
396  M284T108  284.223499  108.406389         NaN        NaN
397  M284T339  284.267962  338.725056         NaN        NaN
398   M284T60  284.195294   59.593561         NaN        NaN
399   M285T34  284.775031   34.079641         NaN        NaN

    molecular_formula molecular_name inchi inchi_key
0                 NaN            NaN   NaN       NaN
1                 NaN            NaN   NaN       NaN
2                 NaN            NaN   NaN       NaN
3                 NaN            NaN   NaN       NaN
4                 NaN            NaN   NaN       NaN
..                …              …     …         …
395               NaN            NaN   NaN       NaN
396               NaN            NaN   NaN       NaN
397               NaN            NaN   NaN       NaN
398               NaN            NaN   NaN       NaN
399               NaN            NaN   NaN       NaN

[400 rows x 9 columns]
```

`mr` is multiple rows format if the match more than once from the reference file:

[14]:
```
mr
```

[14]:
```
          name          mz          rt molecular_formula molecular_name inchi  \
0     M151T34  150.886715   34.152700               NaN            NaN   NaN
1     M151T40  151.040235   39.838172               NaN            NaN   NaN
2     M152T40  152.043607   40.303700               NaN            NaN   NaN
3     M153T34  152.883824   34.174647               NaN            NaN   NaN
4     M153T36  153.019474   35.785847               NaN            NaN   NaN
..        …         …          …                 …              …     …
404   M283T61  283.068474   60.739869               NaN            NaN   NaN
405  M284T108  284.223499  108.406389               NaN            NaN   NaN
406  M284T339  284.267962  338.725056               NaN            NaN   NaN
407   M284T60  284.195294   59.593561               NaN            NaN   NaN
408   M285T34  284.775031   34.079641               NaN            NaN   NaN

    inchi_key  exact_mass  ppm_error
0         NaN         NaN        NaN
1         NaN         NaN        NaN
```

```
2         NaN          NaN         NaN
3         NaN          NaN         NaN
4         NaN          NaN         NaN
..        …            …           …
404       NaN          NaN         NaN
405       NaN          NaN         NaN
406       NaN          NaN         NaN
407       NaN          NaN         NaN
408       NaN          NaN         NaN

[409 rows x 9 columns]
```

Now we merges single format results with correlation results:

[15]:
```
# merge summery table with correlation analysis
res = anno.comp_summ_corr(sr, corr_df)
res
```

[15]:
```
         name          mz          rt  exact_mass  ppm_error  \
0     M167T35  167.021095   34.882147      167.02      -4.57
1     M276T36  276.077397   36.385373      276.08      -2.16
2     M154T37  154.062402   37.183625      154.06      -3.84
3     M174T35  174.088395   35.001130      174.09      -4.67
4     M181T36  181.060407   35.734801      181.06       2.39
..        …         …           …          …          …
395   M279T50  279.159930   50.055451         NaN        NaN
396   M279T79  279.163910   78.758079         NaN        NaN
397   M282T85  282.207859   84.719202         NaN        NaN
398   M283T47  283.110871   46.822069         NaN        NaN
399  M284T108  284.223499  108.406389         NaN        NaN

    molecular_formula             molecular_name  \
0             C7H5NO4            Quinolinic acid
1         C10H16N2O5S             Biotin sulfone
2             C8H10O3             Hydroxytyrosol
3             C8H14O4               Suberic acid
4            C6H7N5O2  8-Hydroxy-7-methylguanine
..                …                          …
395             NaN                        NaN
396             NaN                        NaN
397             NaN                        NaN
398             NaN                        NaN
399             NaN                        NaN

                                                 inchi  \
0    InChI=1S/C7H5NO4/c9-6(10)4-2-1-3-8-5(4)7(11)12…
1    InChI=1S/C10H16N2O5S/c13-8(14)4-2-1-3-7-9-6(5-…
2    InChI=1S/C8H10O3/c9-4-3-6-1-2-7(10)8(11)5-6/h1…
```

```
3    InChI=1S/C8H14O4/c9-7(10)5-3-1-2-4-6-8(11)12/h…
4    InChI=1S/C6H7N5O2/c1-11-2-3(9-6(11)13)8-5(7)10…
..                                                 …
395                                                NaN
396                                                NaN
397                                                NaN
398                                                NaN
399                                                NaN

                       inchi_key  cor_grp_size  \
0    GJAWHXHKYYXBSV-UHFFFAOYSA-N          25.0
1    QPFQYMONYBAUCY-ZKWXMUAHSA-N          13.0
2    JUUBCHWRXWPFFH-UHFFFAOYSA-N          12.0
3    TYFQFVWCELRYAO-UHFFFAOYSA-N           9.0
4    VHPXSVXJBWZORQ-UHFFFAOYSA-N           9.0
..                           …            …
395                          NaN           NaN
396                          NaN           NaN
397                          NaN           NaN
398                          NaN           NaN
399                          NaN           NaN

                                                  cor_grp
0    M171T34::M197T36::M209T34::M211T34::M213T34::M…
1    M277T36_2::M278T36::M173T36_2::M186T36::M187T3…
2    M155T38::M158T37_2::M164T36::M171T37_2::M173T3…
3    M211T34::M213T34::M219T34::M221T34::M229T35::M…
4    M224T36::M225T35::M226T35::M227T36::M269T37_2:…
..                                                 …
395                                                NaN
396                                                NaN
397                                                NaN
398                                                NaN
399                                                NaN

[400 rows x 11 columns]
```

The result data frame `res` is re-arranged as four parts from top to bottom:

- 1st part: identified metabolites, satisfied with correlation analysis
- 2nd part: identified metabolites, not satisfied with correlation
- 3rd part: no identified metabolites, satisfied with correlation
- 4th part: no identified metabolites, not satisfied with correlation

The users should focus on the first part and perform their further analysis.

You can save all results in different forms, such as text format TSV or CSV. You can also save all results `sqlite3` database and use DB Browser for SQLite to view:

[16]:
```
import sqlite3

f_save = False   # here we do NOT save results
db_out = "test.db"
sr_out = "test_s.tsv"

if f_save:
    # save all results into a sqlite3 database
    conn = sqlite3.connect(db_out)
    df[["name", "mz", "rt"]].to_sql("peaklist", conn,
                                    if_exists="replace", index=False)
    corr_df.to_sql("corr_grp", conn, if_exists="replace", index=False)
    corr.to_sql("corr_pval_rt", conn, if_exists="replace", index=False)
    match.to_sql("match", conn, if_exists="replace", index=False)
    mr.to_sql("anno_mr", conn, if_exists="replace", index=False)
    res.to_sql("anno_sr", conn, if_exists="replace", index=False)

    conn.commit()
    conn.close()

    # save final results
    res.to_csv(sr_out, sep="\t", index=False)
```

1.6 End user usages

For end users, `lamp` has two options: command line interface or graphical user interface.

To use GUI, open a terminal and type in:

```
$ lamp gui
```

To use CLI, open a terminal and type in something like:

```
$ lamp cli \
  --sep "tab" \
  --input-data "./data/df_pos_3.tsv" \
  --col-idx "1, 2, 3, 4" \
  --add-path "" \
  --ref-path "" \
  --ion-mode "pos" \
  --cal-mass \
  --thres-rt "1.0" \
  --thres-corr "0.5" \
  --thres-pval "0.05" \
  --method "pearson" \
  --positive \
  --ppm "5.0" \
  --save-db \
  --save-mr \
  --db-out "./res/test.db" \
```

```
  --sr-out "./res/test_s.tsv" \
  --mr-out "./res/test_m.tsv"
```

Or you can create a bash script `lamp_cli.sh` (Linux and MacOS) or Windows script `lamp_cli.bat` to contain these CLI arguments and run:

- For Linux and MacOS terminal:

  ```
  $ chmod +x lamp_cli.sh
  $ ./lamp_cli.sh
  ```

- For Windows terminal:

  ```
  $ lamp_cli.bat
  ```