lamp_quick_start

October 10, 2024

1 Quick Start

In this vignette we will demonstrate how to use `lamp` python package.

1.1 1. Setup

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

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

1.2 2. Data Loading

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

[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` needs to indicates the locations of peak name, m/z value, retention time and starting points of data matrix. 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 3. Metabolite annotation

To performance metabolite annotation, users shoudl provide their own reference file. If not, `lamp` will use its default reference file for annotation.

[4]:
```
ppm = 5.0
ion_mode = "pos"

ref_path = ""    # if empty, use default reference file for matching

# load refernce 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`. 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`.

[5]:
```
match = anno.comp_match_mass(df, ppm, ref)
match
```

[5]:
```
             id          mz  compound_id molecular_formula  \
0      M152T40  152.043607        19682         C4H12N2S2
1      M153T40  153.055906         3589         C4H12NO3P
2      M154T37  154.062402         7777           C8H10O3
3      M154T37  154.062402         3920           C8H10O3
4      M154T37  154.062402        13366           C8H10O3
..         …         …           …                …
128  M278T42_2  278.153135         9966          C16H22O4
129  M278T42_2  278.153135        10947          C16H22O4
130    M280T38  280.124684        32692       C14H20N2O2S
131    M281T35  281.036224        11165       C14H13Cl2NO
132    M283T47  283.110871        31572        C16H14FN3O

                             compound_name  exact_mass  ppm_error
0                                 cystamine      152.04      -3.84
1          N-Dimethyl-2-aminoethylphosphonate      153.06       2.78
2                       2,6-Dimethoxyphenol      154.06      -3.84
3        4-Hydroxy-3-methoxy-benzenemethanol      154.06      -3.84
4                             Hydroxytyrosol      154.06      -3.84
..                                       …         …        …
128                     Dibutyl phthalate      278.15       4.77
129                   Diisobutyl phthalate      278.15       4.77
130                          Azabon (USAN)      280.12       0.48
```

```
131  2-Amino-1,2-bis(p-chlorophenyl)ethanol      281.04      -4.25
132                   Afloqualone (JP15/INN)      283.11      -4.30

[133 rows x 7 columns]
```

`match` gives the counpound 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 default adducts library are:

[6]:
```
path = 'lib/adducts.txt'
filename = os.path.join(
    os.path.dirname(os.path.abspath(lamp.__file__)), path
)
lib_df = pd.read_csv(filename, sep="\t")
lib_df
```

[6]:

	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+NaFormate]	67.987400	1	pos
10	[M+NaFormate+NaFormate]	135.974800	1	pos
11	[M+NaCl]	57.958600	1	pos
12	[M+Formic acid]	46.000500	1	pos
13	[M+Acetonitrile]	41.026500	1	pos
14	[M+CaFormate]	84.960300	1	pos
15	[M-H]-	-1.007276	1	neg
16	[M+35Cl]-	34.969401	1	neg
17	[M+Formate]-	44.998203	1	neg
18	[M+Acetate]-	59.013853	1	neg
19	[M+NaFormate]	66.979600	1	neg
20	[M+NaFormate+NaFormate]	134.967000	1	neg
21	[M+NaCl]	56.950800	1	neg
22	[M+Formic acid]	44.992700	1	neg
23	[M+Acetonitrile]	40.018700	1	neg
24	[M+CaFormate]	83.952500	1	neg

[8]:
```
ion_mode = "pos"
# if empty, use default adducts library
add_path = ""
```

```
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
9	[M+NaFormate]	67.987400	1
10	[M+NaFormate+NaFormate]	135.974800	1
11	[M+NaCl]	57.958600	1
12	[M+Formic acid]	46.000500	1
13	[M+Acetonitrile]	41.026500	1
14	[M+CaFormate]	84.960300	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	compound_id	molecular_formula	\
0	M151T40	151.040235	18293	C2H7O3P	
1	M152T40	152.043607	1138	C5H8N2O2	
2	M152T40	152.043607	3613	C5H8N2O2	
3	M152T40	152.043607	125	C5H8N2O2	
4	M153T36	153.019474	4021	CH5O4P	
…	…	…	…	…	
1698	M283T60	283.191869	35629	C19H23NO	
1699	M284T60	284.195294	37078	C11H26N2O6	
1700	M284T108	284.223499	6237	C19H26N2	
1701	M284T108	284.223499	18303	C16H34	
1702	M284T108	284.223499	18302	C16H34	

	compound_name	exact_mass	\
0	ethylphosphonate	151.04	
1	5,6-Dihydrothymine	152.04	
2	alpha-Amino-gamma-cyanobutanoate	152.04	
3	gamma-Amino-gamma-cyanobutanoate	152.04	
4	Hydroxymethylphosphonate	153.02	
…	…	…	
1698	Diphenylpyraline (INN)	283.19	
1699	2-[3-(2-Hydroxy-1,1-Dihydroxymethyl-Ethylamino…	284.19	
1700	(-)-Quebrachamine	284.22	

```
1701                              3-methyl-pentadecane      284.22
1702                                        hexadecane      284.22

                adduct  ppm_error
0     [M+Acetonitrile]       3.01
1              [M+Mg]+       3.52
2              [M+Mg]+       3.52
3              [M+Mg]+       3.52
4     [M+Acetonitrile]       2.79
…                 …          …
1698           [M+2H]+      -4.23
1699           [M+2H]+       3.89
1700           [M+2H]+      -4.23
1701          [M+NaCl]      -4.05
1702          [M+NaCl]      -4.05

[1703 rows x 8 columns]
```

1.4 4. Correlation analysis

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

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

[10]:
```
thres_rt = 1.0
thres_corr = 0.7
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: 1.4457974433898926 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	M171T34	0.75	5.545024e-19	0.25

```
4      M151T34   M181T34     0.73  3.471998e-18     0.53
..         …       …      …           …       …
865   M281T287  M282T287     0.99  1.570424e-99     0.04
866    M282T61   M283T61     0.95  1.362657e-50     0.01
867  M283T34_1   M285T34     0.82  5.937139e-26     0.08
868    M283T60   M284T60     0.86  1.033010e-29     0.15
869   M283T339  M284T339     0.91  4.031333e-39     0.04

[870 rows x 5 columns]
```

Based on the correlation analsys, 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  \
0      M231T34            35
1      M216T35            35
2      M215T35            35
3      M217T35            35
4      M239T34            34
..         …           …
264  M158T37_1             1
265   M279T233             1
266   M255T275             1
267   M280T233             1
268   M206T573             1

                                                 cor_grp
0    M233T34::M239T34::M241T34::M249T33::M256T35::M…
1    M217T35::M218T35::M219T34::M219T35::M221T34::M…
2    M216T35::M217T35::M218T35::M219T34::M219T35::M…
3    M218T35::M219T34::M219T35::M221T34::M223T34::M…
4    M241T34::M249T33::M256T35::M259T35::M261T35::M…
..                                                 …
264                                          M156T37
265                                         M280T233
266                                         M256T275
267                                         M279T233
268                                         M208T573

[269 rows x 3 columns]
```

1.5 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	compound_id	exact_mass	ppm_error	\
0	M151T34	150.886715	34.152700	NaN	NaN	NaN	
1	M151T40	151.040235	39.838172	NaN	NaN	NaN	
2	M152T40	152.043607	40.303700	19682.0	152.04	-3.84	
3	M153T34	152.883824	34.174647	NaN	NaN	NaN	
4	M153T36	153.019474	35.785847	NaN	NaN	NaN	
..	…	…	…	…	…	…	
395	M283T61	283.068474	60.739869	NaN	NaN	NaN	
396	M284T108	284.223499	108.406389	NaN	NaN	NaN	
397	M284T339	284.267962	338.725056	NaN	NaN	NaN	
398	M284T60	284.195294	59.593561	NaN	NaN	NaN	
399	M285T34	284.775031	34.079641	NaN	NaN	NaN	

	molecular_formula	compound_name
0	NaN	NaN
1	NaN	NaN
2	C4H12N2S2	cystamine
3	NaN	NaN
4	NaN	NaN
..	…	…
395	NaN	NaN
396	NaN	NaN
397	NaN	NaN
398	NaN	NaN
399	NaN	NaN

[400 rows x 8 columns]

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

[17]:
```
mr
```

[17]:

	name	mz	rt	compound_id	molecular_formula	\
0	M151T34	150.886715	34.152700	NaN	NaN	
1	M151T40	151.040235	39.838172	NaN	NaN	
2	M152T40	152.043607	40.303700	19682.0	C4H12N2S2	
3	M153T34	152.883824	34.174647	NaN	NaN	
4	M153T36	153.019474	35.785847	NaN	NaN	
..	…	…	…	…	…	
480	M283T61	283.068474	60.739869	NaN	NaN	

```
481  M284T108  284.223499  108.406389          NaN               NaN
482  M284T339  284.267962  338.725056          NaN               NaN
483   M284T60  284.195294   59.593561          NaN               NaN
484   M285T34  284.775031   34.079641          NaN               NaN

    compound_name  exact_mass  ppm_error
0             NaN         NaN        NaN
1             NaN         NaN        NaN
2       cystamine      152.04      -3.84
3             NaN         NaN        NaN
4             NaN         NaN        NaN
..            …           …          …
480           NaN         NaN        NaN
481           NaN         NaN        NaN
482           NaN         NaN        NaN
483           NaN         NaN        NaN
484           NaN         NaN        NaN

[485 rows x 8 columns]
```

Now we merges single format results with correlation results:

[14]:
```
# merge summery table with correlation analysis
res = anno.comp_summ_corr(sr, corr_df)
res
```

[14]:
```
         name          mz          rt  compound_id  exact_mass  ppm_error  \
0     M231T34  230.899863   33.939162      34582.0      230.90      -2.86
5     M229T34  228.902797   33.943804      19352.0      228.90       3.66
33    M186T36  186.045606   36.486115      37273.0      186.05      -3.72
46    M276T36  276.077397   36.385373      13041.0      276.08      -2.16
61    M263T35  263.044237   35.178853      32165.0      263.04       4.75
..        …        …           …           …          …          …
393   M279T79  279.163910   78.758079          NaN         NaN        NaN
395   M280T67  280.192145   67.298802          NaN         NaN        NaN
396   M282T85  282.207859   84.719202          NaN         NaN        NaN
397   M283T37  283.103695   36.796242          NaN         NaN        NaN
399  M284T108  284.223499  108.406389          NaN         NaN        NaN

                     molecular_formula  \
0                            C3Cl3N3O3
5                             C6H3Cl4N
33   C7H10N2O2S::C7H10N2O2S::C7H10N2O2S
46                         C10H16N2O5S
61                            C12H9NO6
..                                  …
393                               NaN
395                               NaN
```

```
396                                    NaN
397                                    NaN
399                                    NaN

                                        compound_name  cor_grp_size  \
0                                 Symclosene (USAN/INN)          35.0
5                                             nitrapyrin          34.0
33   (4s)-2-[(1e)-1-Aminoprop-1-Enyl]-4,5-Dihydro-1…          12.0
46                                        Biotin sulfone          12.0
61                                     Miloxacin (INN)          11.0
..                                                 …           …
393                                              NaN           NaN
395                                              NaN           NaN
396                                              NaN           NaN
397                                              NaN           NaN
399                                              NaN           NaN

                                                  cor_grp
0    M233T34::M239T34::M241T34::M249T33::M256T35::M…
5    M231T34::M233T34::M239T34::M241T34::M249T33::M…
33   M187T36::M188T36::M189T36::M200T36::M201T36::M…
46   M277T36_2::M278T36::M173T36_2::M186T36::M187T3…
61   M277T36_1::M279T35::M281T35::M215T35::M216T35:…
..                                                 …
393                                              NaN
395                                              NaN
396                                              NaN
397                                              NaN
399                                              NaN

[400 rows x 10 columns]
```

You can save all results in different forms, such as `sqlite3` database:

[15]:
```python
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)
```