Protein Identification Target/Decoy and False Discover Rate

COMPUTATIONAL BIOLOGY 1

PROF.DR/IBRAHIM AL-AWADY ALSAMMAN

Team Names:

- Omnia Tarek Farghaly Moahemd (sec 3 group 1)
- Afnan Eid Mahmoud (sec 2 group 1)
- Sohaila Essam El-Den Mohamed (sec 4 group 2)
- Arwa Mostafa Kamal Abd El-Wahab (sec 2 group 1)

References

- Fasta File

Uniprot link: https://www.uniprot.org/uniprot/P48444

- Raw Data File(mzML)

Pridelink : https://www.ebi.ac.uk/pride/archive/projects/PXD024124

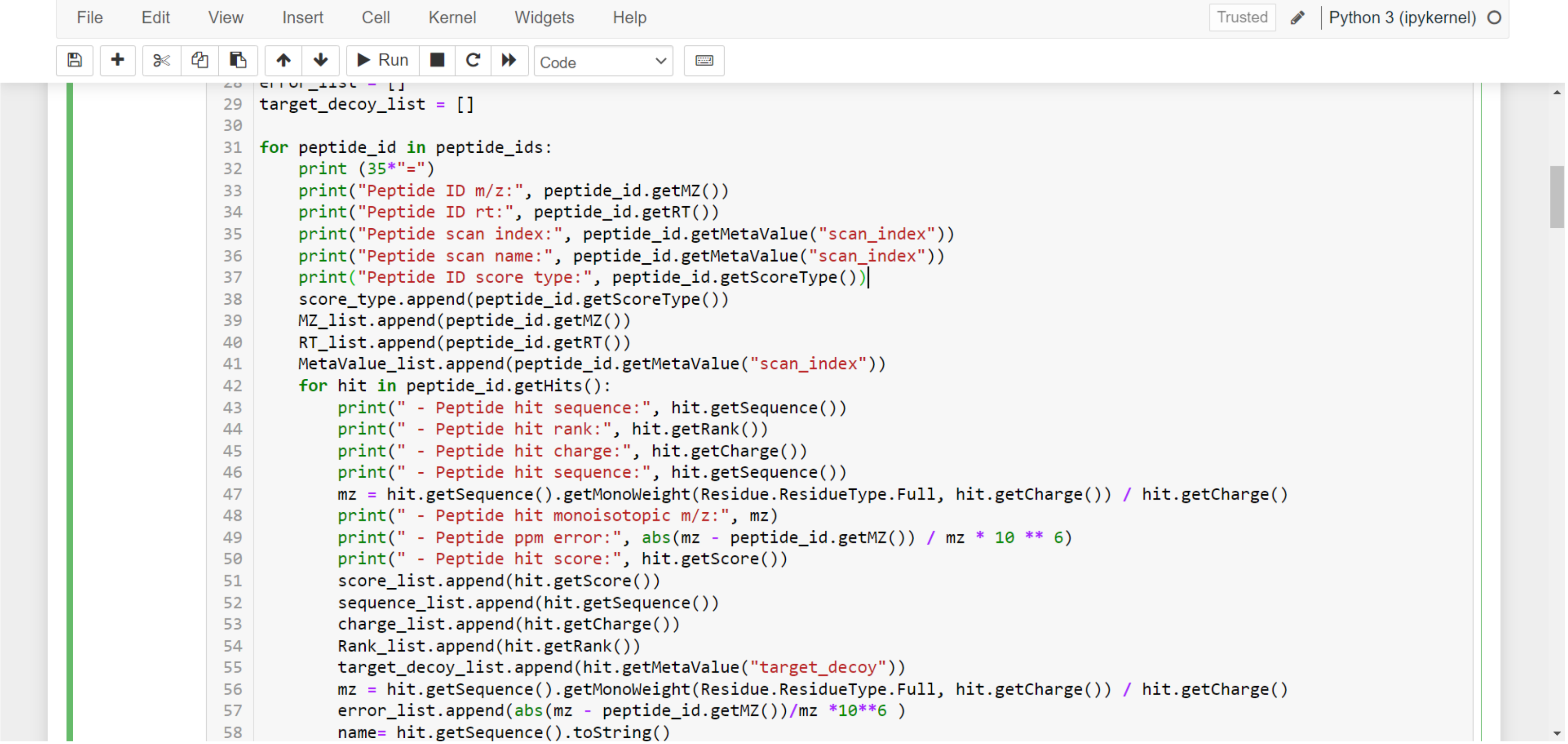
Fasta sequence(Homo sapiens)

- >sp|P48444|COPD_HUMAN Coatomer subunit delta OS=Homo sapiens OX=9606 GN=ARCN1 PE=1 SV=1MVLLAAAVCTKAGKAIVSRQFVEMTRTRIEGLLAAFPKLMNTGKQHTFVETESVRYVYQPMEKLYMVLITTKNSNILEDLETLRLFSRVIPEYCRALEENEISEHCFDLIFAFDEIVALGYRENVNLAQIRTFTEMDSHEEKVFRAVRETQEREAKAEMRRKAKELQQARRDAERQGKKAPGFGGFGSSAVSGGSTAAMITETIIETDKPKVAPAPARPSGPSKALKLGAKGKEVDNFVDKLKSEGETIMSSSMGKRTSEATKMHAPPINMESVHMKIEEKITLTCGRDGGLQNMELHGMIMLRISDDKYGRIRLHVENEDKKGVQLQTHPNVDKKLFTAESLIGLKNPEKSFPVNSDVGVLKWRLQTTEESFIPLTINCWPSESGNGCDVNIEYELQEDNLELNDVVITIPLPSGVGAPVIGEIDGEYRHDSRRNTLEWCLPVIDAKNKSGSLEFSIAGQPNDFFPVQVSFVSKKNYCNIQVTKVTQVDGNSPVRFSTETTFLVDKYEIL

Identifications Data

- In OpenMS, identifications of peptides, proteins and small molecules are stored in dedicated data structures. These data structures are typically stored to disc as idXML or mzIdentML file. The highest-level structure is ProteinIdentification. It stores all identified proteins of an identification run as ProteinHit objects plus additional metadata (search parameters, etc.). Each ProteinHit contains the actual protein accession, an associated score, and (optionally) the protein sequence
- A PeptideIdentification object stores the data corresponding to a single identified spectrum or feature. It has members for the retention time, m/z, and a vector of PeptideHit objects. Each PeptideHit stores the information of a specific peptide-to-spectrum match or PSM (e.g., the score and the peptide sequence). Each PeptideHit also contains a vector of PeptideEvidence objects which store the reference to one or more (in the case the peptide maps to multiple proteins) proteins and the position therein

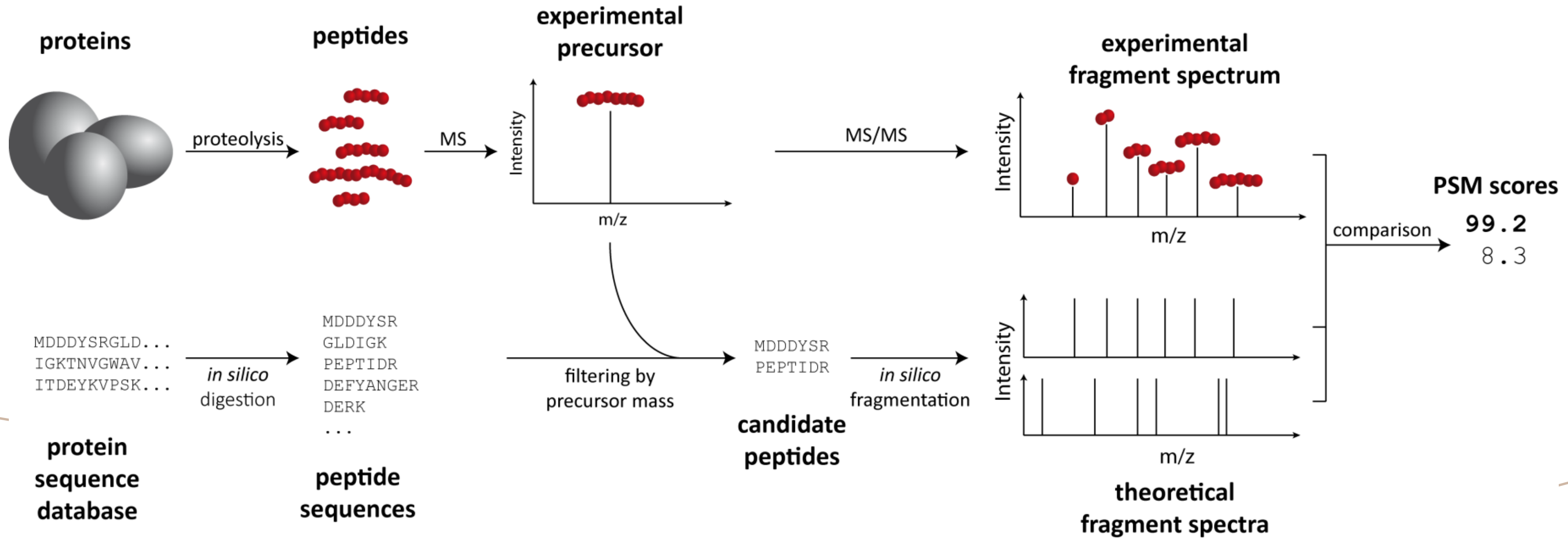
Protein and Peptide identification



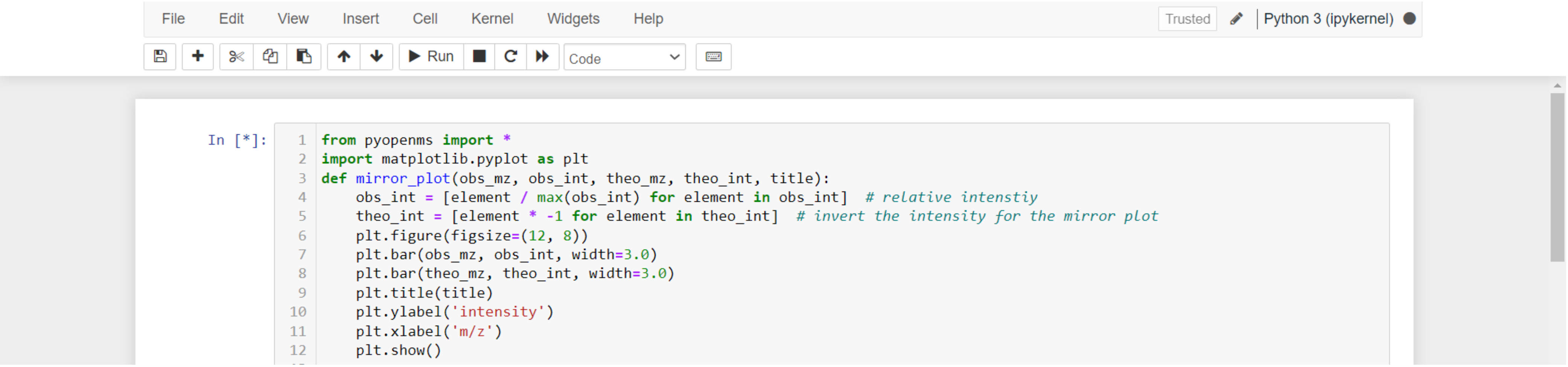
```python
error_list = []
target_decoy_list = []

for peptide_id in peptide_ids:
    print (35*"=")
    print("Peptide ID m/z:", peptide_id.getMZ())
    print("Peptide ID rt:", peptide_id.getRT())
    print("Peptide scan index:", peptide_id.getMetaValue("scan_index"))
    print("Peptide scan name:", peptide_id.getMetaValue("scan_index"))
    print("Peptide ID score type:", peptide_id.getScoreType())
    score_type.append(peptide_id.getScoreType())
    MZ_list.append(peptide_id.getMZ())
    RT_list.append(peptide_id.getRT())
    MetaValue_list.append(peptide_id.getMetaValue("scan_index"))
    for hit in peptide_id.getHits():
        print(" - Peptide hit sequence:", hit.getSequence())
        print(" - Peptide hit rank:", hit.getRank())
        print(" - Peptide hit charge:", hit.getCharge())
        print(" - Peptide hit sequence:", hit.getSequence())
        mz = hit.getSequence().getMonoWeight(Residue.ResidueType.Full, hit.getCharge()) / hit.getCharge()
        print(" - Peptide hit monoisotopic m/z:", mz)
        print(" - Peptide ppm error:", abs(mz - peptide_id.getMZ()) / mz * 10 ** 6)
        print(" - Peptide hit score:", hit.getScore())
        score_list.append(hit.getScore())
        sequence_list.append(hit.getSequence())
        charge_list.append(hit.getCharge())
        Rank_list.append(hit.getRank())
        target_decoy_list.append(hit.getMetaValue("target_decoy"))
        mz = hit.getSequence().getMonoWeight(Residue.ResidueType.Full, hit.getCharge()) / hit.getCharge()
        error_list.append(abs(mz - peptide_id.getMZ())/mz *10**6 )
        name= hit.getSequence().toString()
```

Peptide Search

- In MS-based proteomics, fragment ion spectra (MS2 spectra) are often interpreted by comparing them against a theoretical set of spectra generated from a FASTA database. OpenMS contains a (simple) implementation of such a "search engine" that compares experimental spectra against theoretical spectra generated from an enzymatic or chemical digest of a proteome (e.g. tryptic digest).

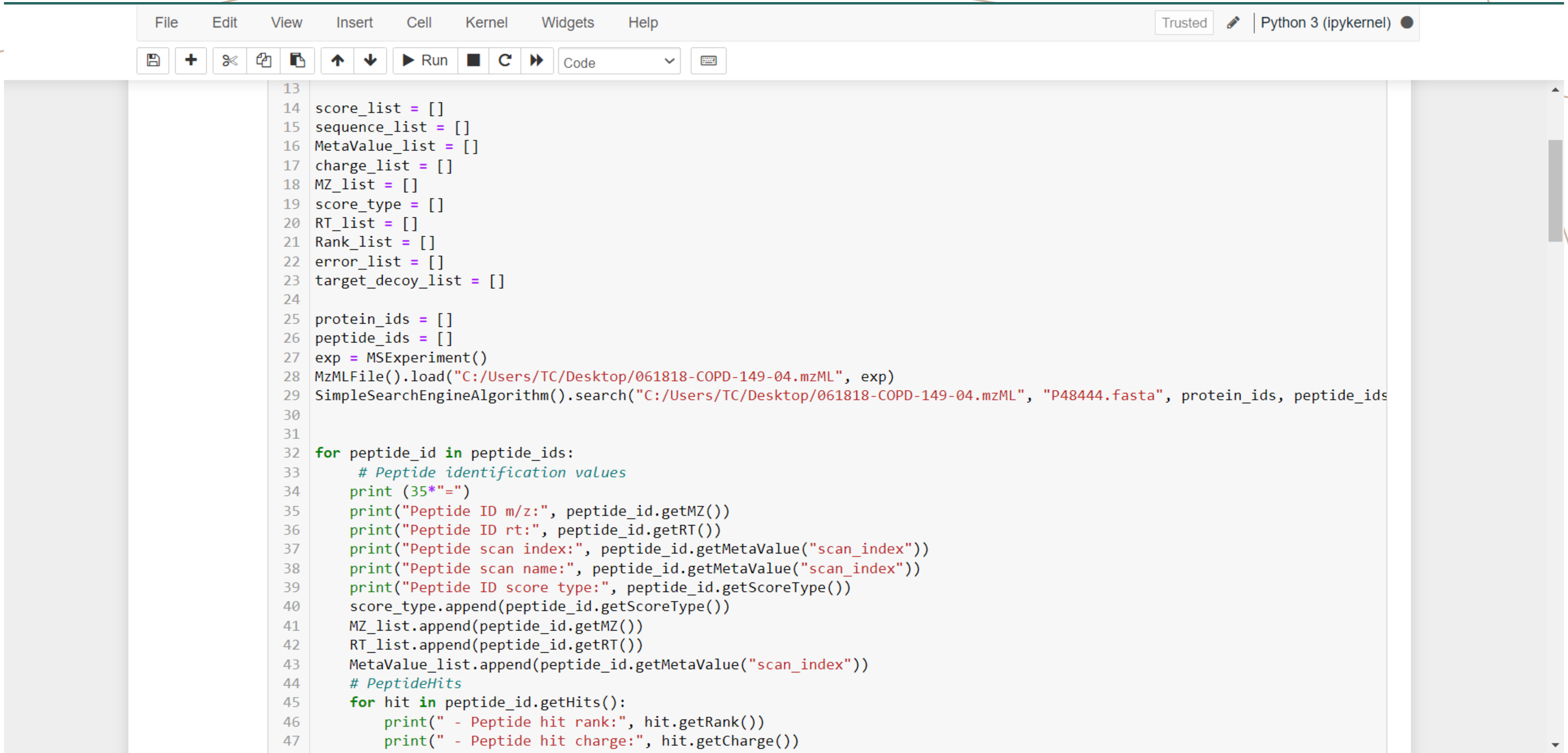


The whole code



```python
from pyopenms import *
import matplotlib.pyplot as plt
def mirror_plot(obs_mz, obs_int, theo_mz, theo_int, title):
    obs_int = [element / max(obs_int) for element in obs_int]  # relative intenstiy
    theo_int = [element * -1 for element in theo_int]  # invert the intensity for the mirror plot
    plt.figure(figsize=(12, 8))
    plt.bar(obs_mz, obs_int, width=3.0)
    plt.bar(theo_mz, theo_int, width=3.0)
    plt.title(title)
    plt.ylabel('intensity')
    plt.xlabel('m/z')
    plt.show()
```

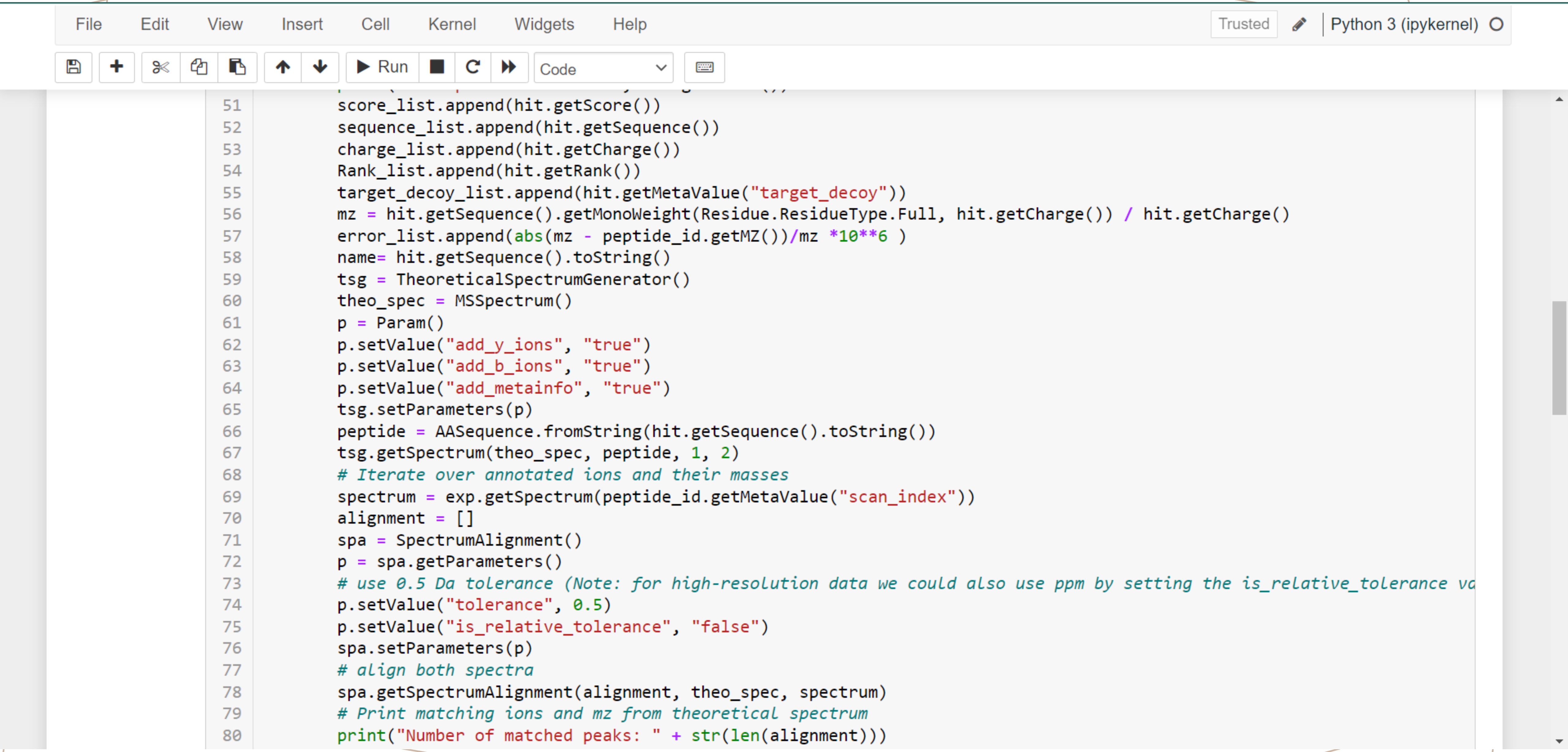
Here we start with writing a function called mirror_plot to draw the mirror between the theoretical and experimental data



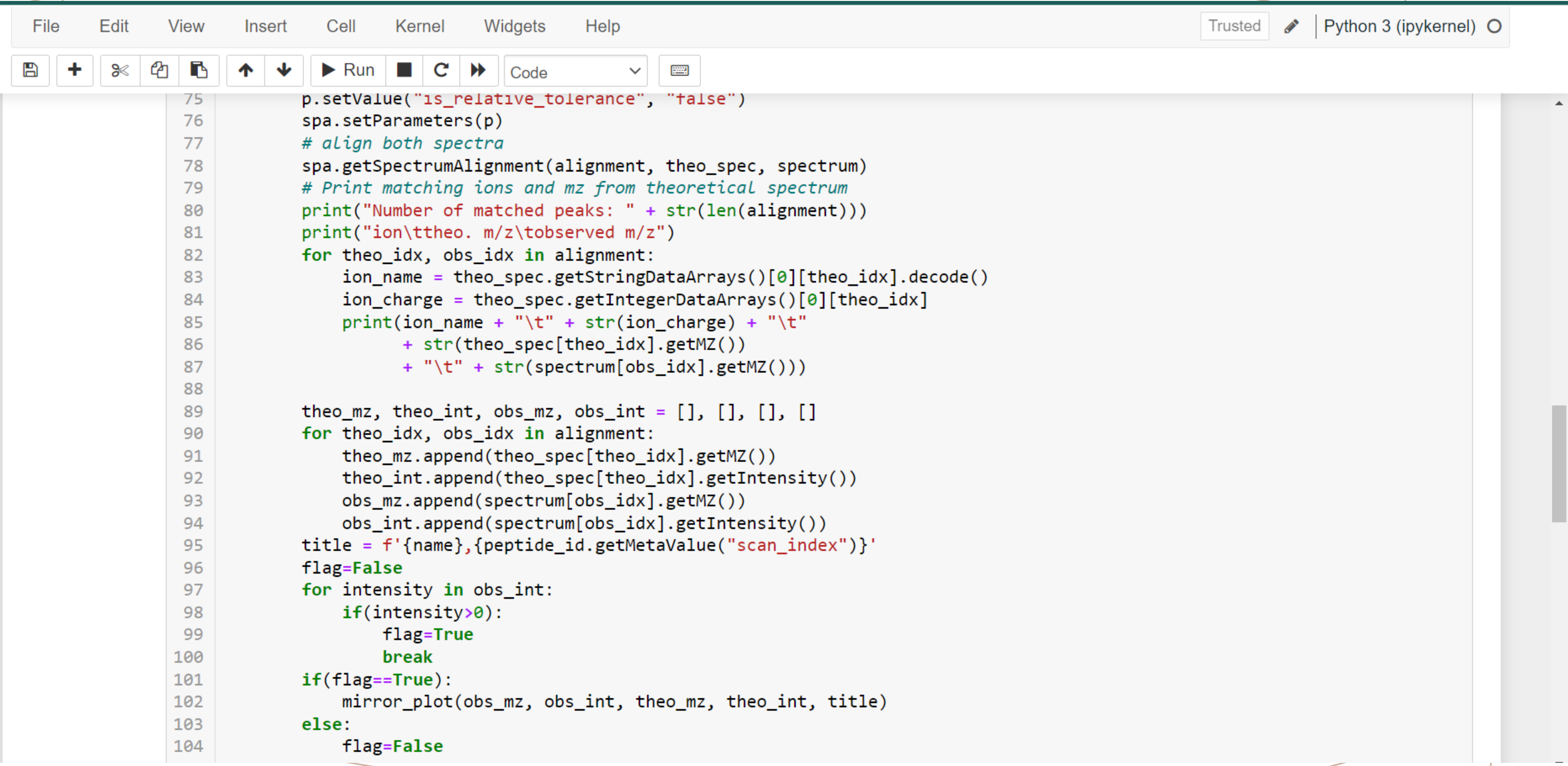
```python

score_list = []
sequence_list = []
MetaValue_list = []
charge_list = []
MZ_list = []
score_type = []
RT_list = []
Rank_list = []
error_list = []
target_decoy_list = []

protein_ids = []
peptide_ids = []
exp = MSExperiment()
MzMLFile().load("C:/Users/TC/Desktop/061818-COPD-149-04.mzML", exp)
SimpleSearchEngineAlgorithm().search("C:/Users/TC/Desktop/061818-COPD-149-04.mzML", "P48444.fasta", protein_ids, peptide_ids


for peptide_id in peptide_ids:
     # Peptide identification values
    print (35*"=")
    print("Peptide ID m/z:", peptide_id.getMZ())
    print("Peptide ID rt:", peptide_id.getRT())
    print("Peptide scan index:", peptide_id.getMetaValue("scan_index"))
    print("Peptide scan name:", peptide_id.getMetaValue("scan_index"))
    print("Peptide ID score type:", peptide_id.getScoreType())
    score_type.append(peptide_id.getScoreType())
    MZ_list.append(peptide_id.getMZ())
    RT_list.append(peptide_id.getRT())
    MetaValue_list.append(peptide_id.getMetaValue("scan_index"))
    # PeptideHits
    for hit in peptide_id.getHits():
        print(" - Peptide hit rank:", hit.getRank())
        print(" - Peptide hit charge:", hit.getCharge())
```

-In the above code we make lists to store the valued we mentioned to retrieve it in a data frame after we get the values

-Construction of decoy databases

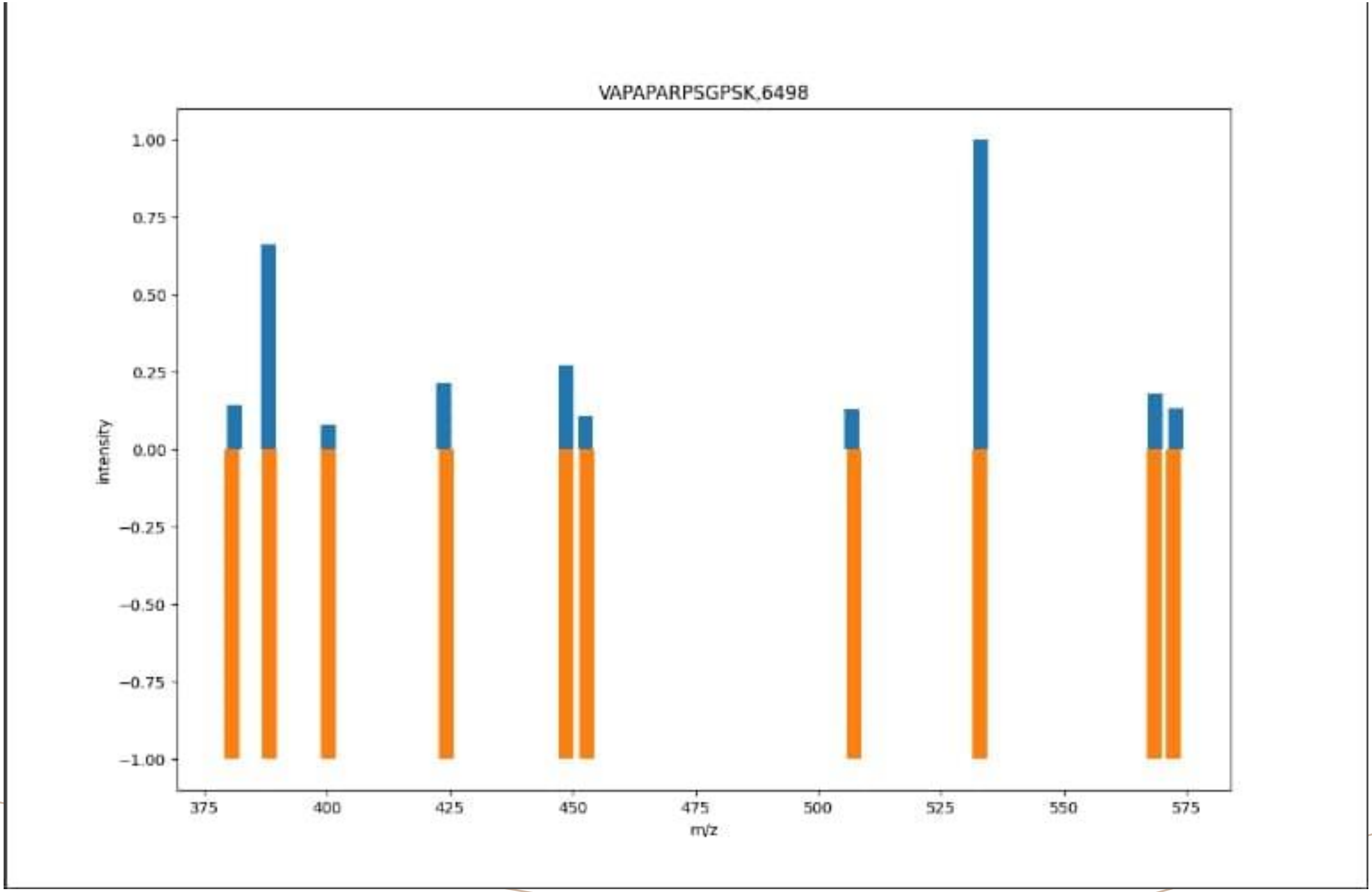


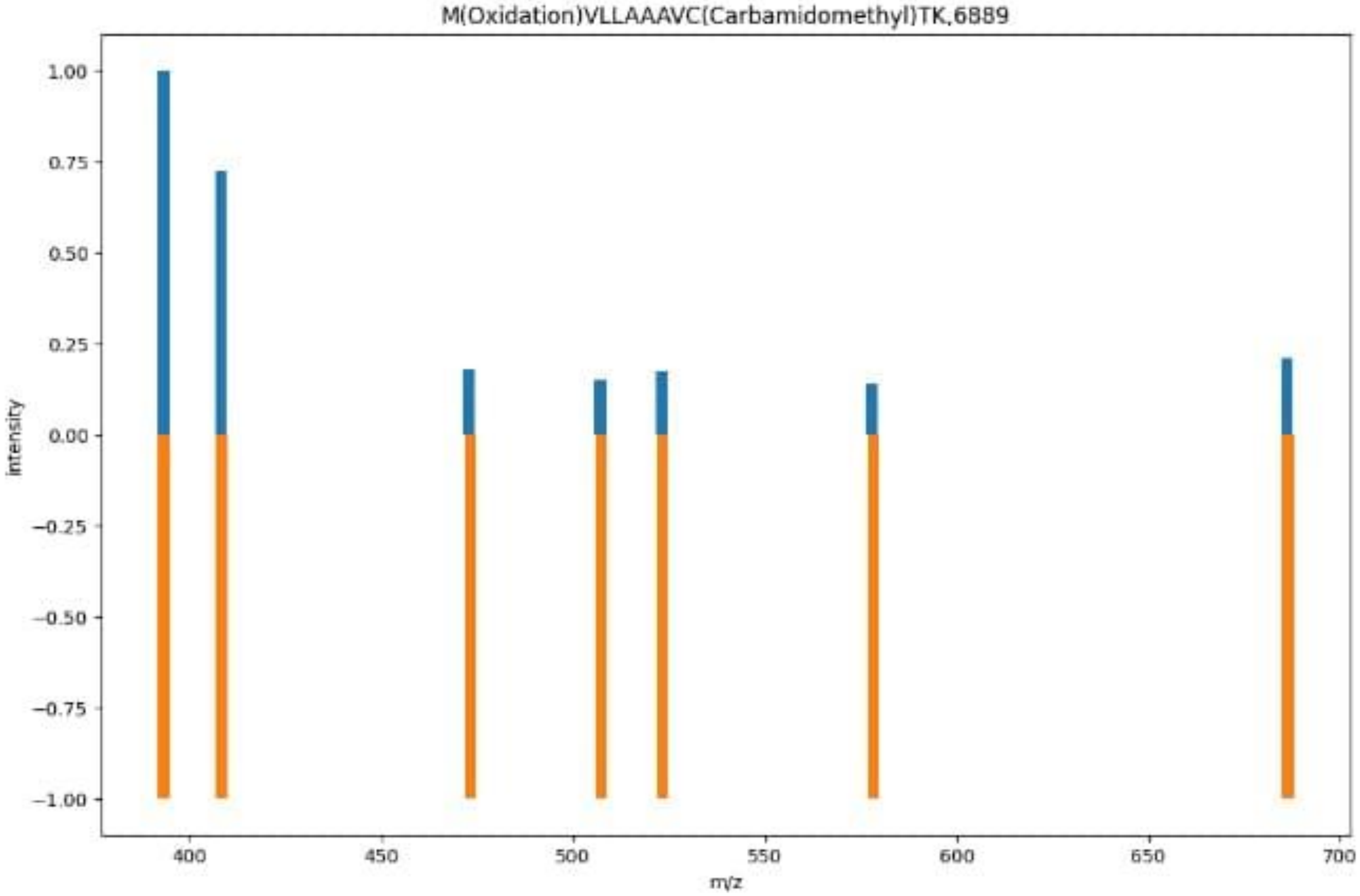
```
        score_list.append(hit.getScore())
        sequence_list.append(hit.getSequence())
        charge_list.append(hit.getCharge())
        Rank_list.append(hit.getRank())
        target_decoy_list.append(hit.getMetaValue("target_decoy"))
        mz = hit.getSequence().getMonoWeight(Residue.ResidueType.Full, hit.getCharge()) / hit.getCharge()
        error_list.append(abs(mz - peptide_id.getMZ())/mz *10**6 )
        name= hit.getSequence().toString()
        tsg = TheoreticalSpectrumGenerator()
        theo_spec = MSSpectrum()
        p = Param()
        p.setValue("add_y_ions", "true")
        p.setValue("add_b_ions", "true")
        p.setValue("add_metainfo", "true")
        tsg.setParameters(p)
        peptide = AASequence.fromString(hit.getSequence().toString())
        tsg.getSpectrum(theo_spec, peptide, 1, 2)
        # Iterate over annotated ions and their masses
        spectrum = exp.getSpectrum(peptide_id.getMetaValue("scan_index"))
        alignment = []
        spa = SpectrumAlignment()
        p = spa.getParameters()
        # use 0.5 Da tolerance (Note: for high-resolution data we could also use ppm by setting the is_relative_tolerance va
        p.setValue("tolerance", 0.5)
        p.setValue("is_relative_tolerance", "false")
        spa.setParameters(p)
        # align both spectra
        spa.getSpectrumAlignment(alignment, theo_spec, spectrum)
        # Print matching ions and mz from theoretical spectrum
        print("Number of matched peaks: " + str(len(alignment)))
```

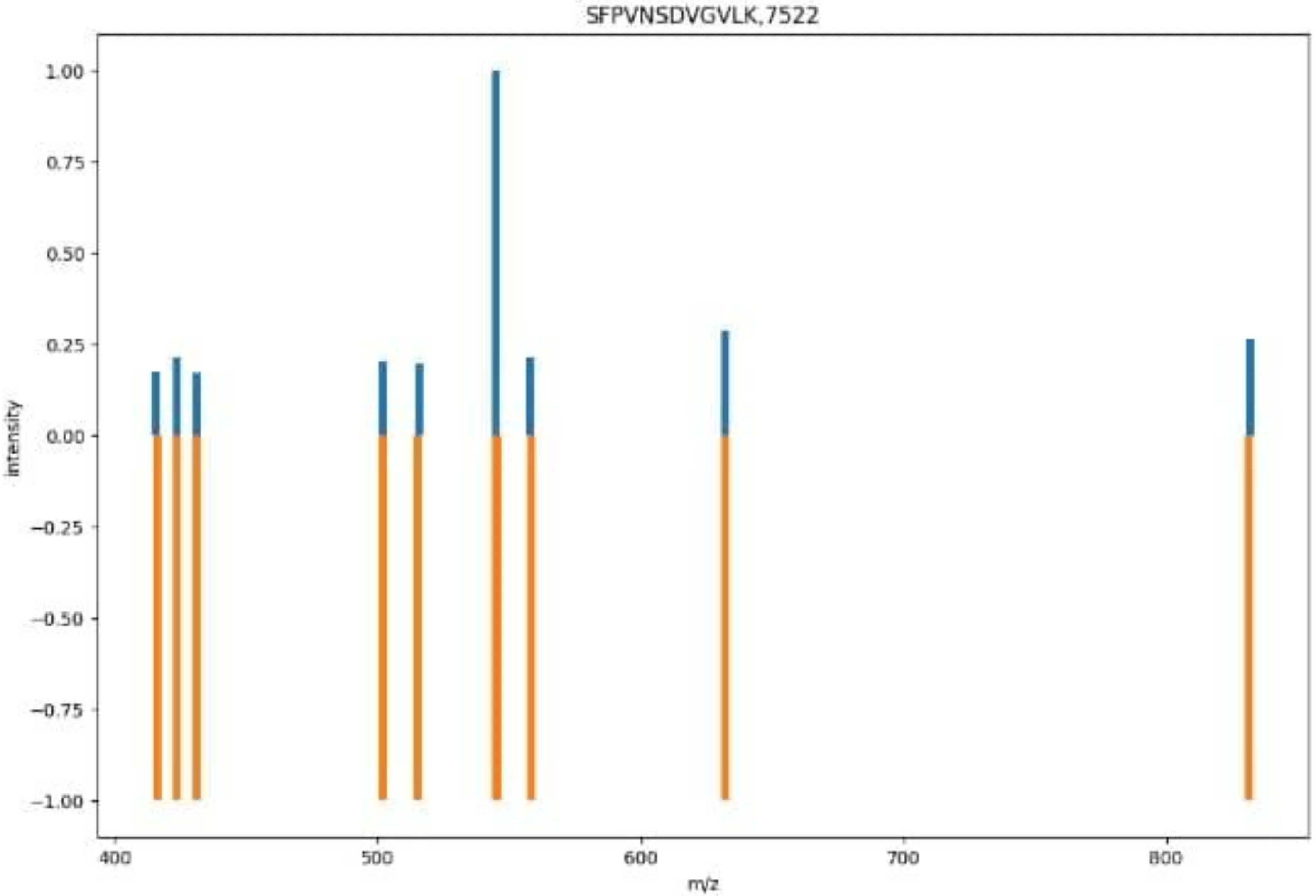


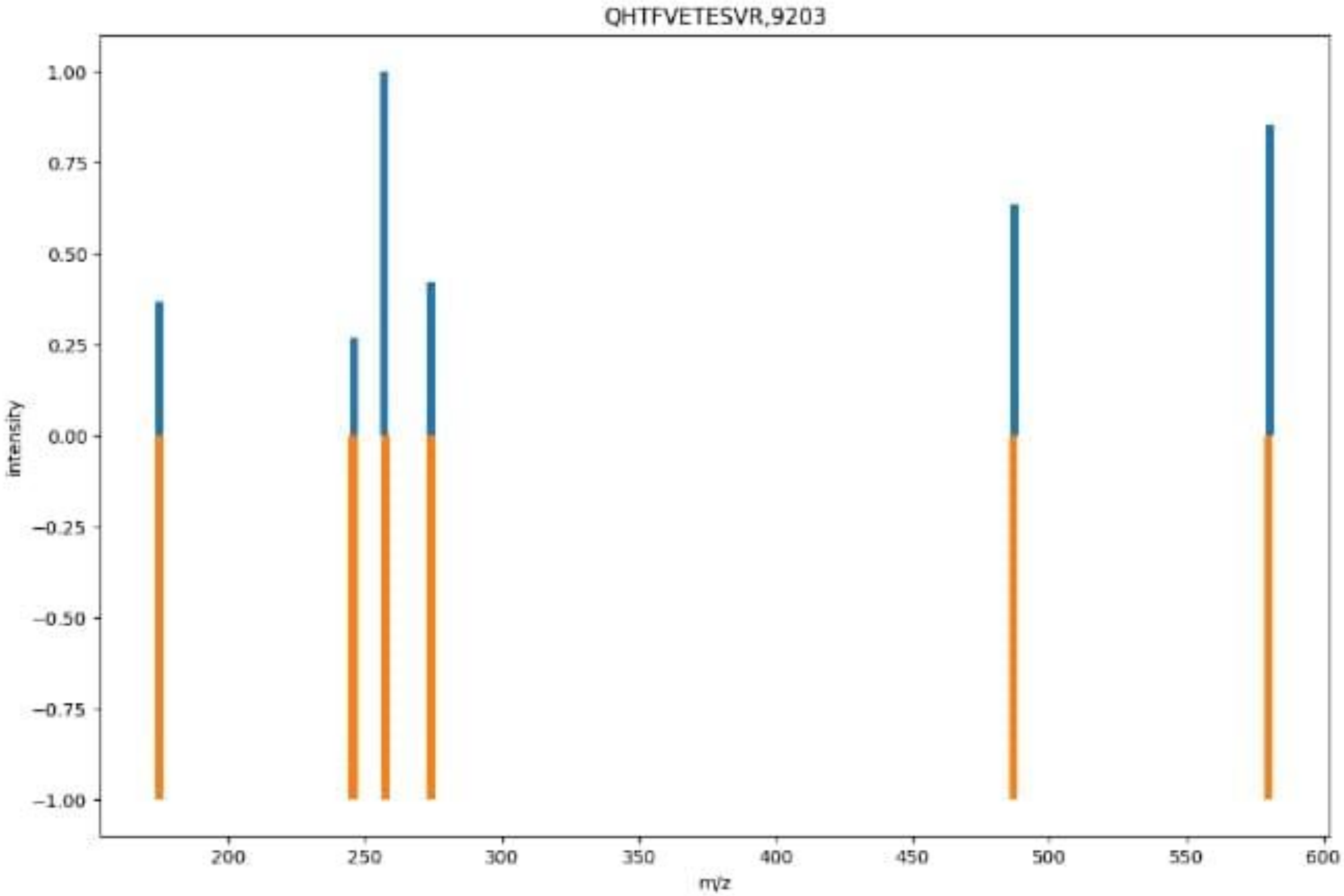
```
        p.setValue("is_relative_tolerance", "false")
        spa.setParameters(p)
        # align both spectra
        spa.getSpectrumAlignment(alignment, theo_spec, spectrum)
        # Print matching ions and mz from theoretical spectrum
        print("Number of matched peaks: " + str(len(alignment)))
        print("ion\ttheo. m/z\tobserved m/z")
        for theo_idx, obs_idx in alignment:
            ion_name = theo_spec.getStringDataArrays()[0][theo_idx].decode()
            ion_charge = theo_spec.getIntegerDataArrays()[0][theo_idx]
            print(ion_name + "\t" + str(ion_charge) + "\t"
                  + str(theo_spec[theo_idx].getMZ())
                  + "\t" + str(spectrum[obs_idx].getMZ()))

        theo_mz, theo_int, obs_mz, obs_int = [], [], [], []
        for theo_idx, obs_idx in alignment:
            theo_mz.append(theo_spec[theo_idx].getMZ())
            theo_int.append(theo_spec[theo_idx].getIntensity())
            obs_mz.append(spectrum[obs_idx].getMZ())
            obs_int.append(spectrum[obs_idx].getIntensity())
        title = f'{name},{peptide_id.getMetaValue("scan_index")}'
        flag=False
        for intensity in obs_int:
            if(intensity>0):
                flag=True
                break
        if(flag==True):
            mirror_plot(obs_mz, obs_int, theo_mz, theo_int, title)
        else:
            flag=False
```

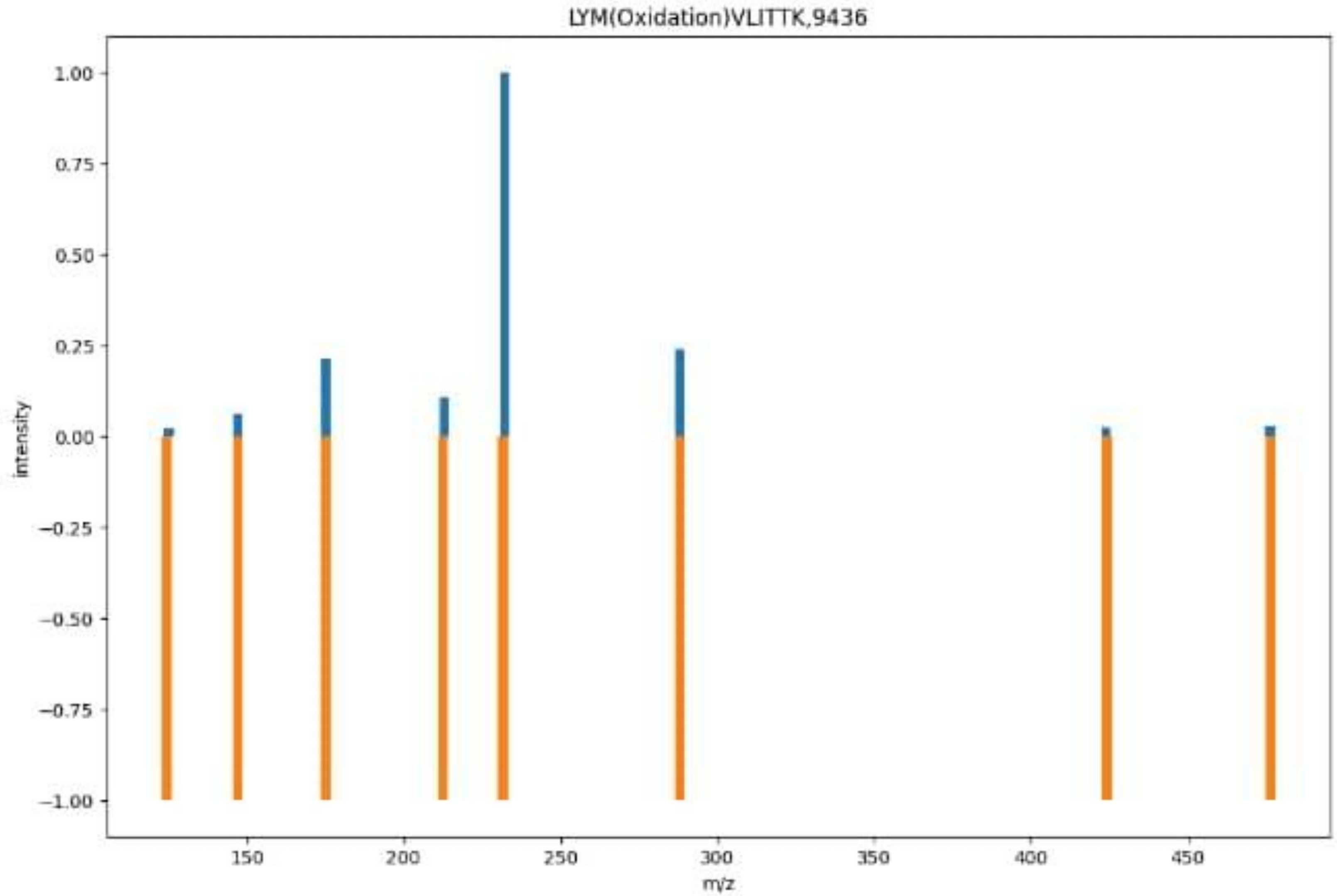
File :061818-COPD-152-02.mzML

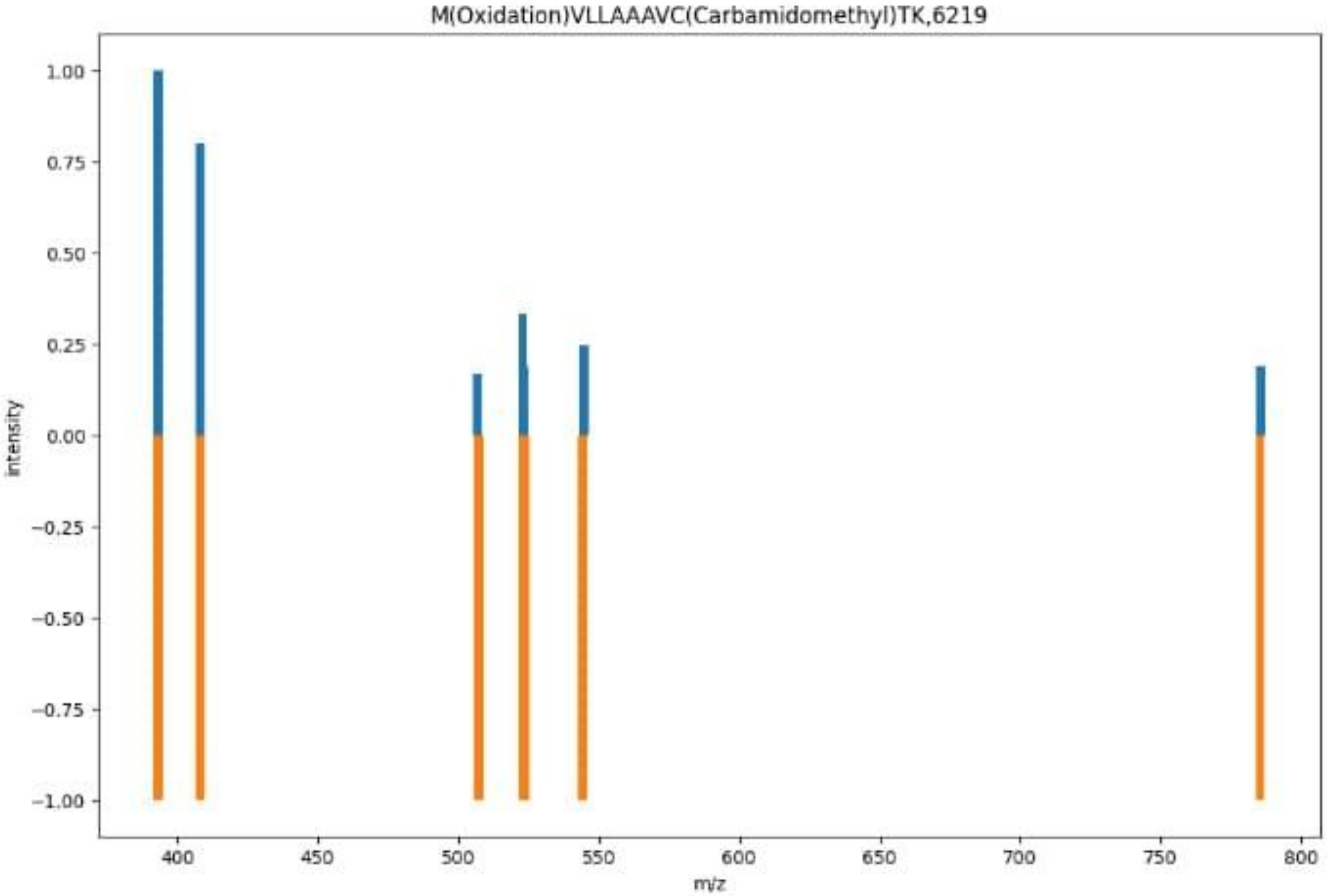



M(Oxidation)VLLAAAVC(Carbamidomethyl)TK,6889
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
450
500
550
600
650
700
m/z

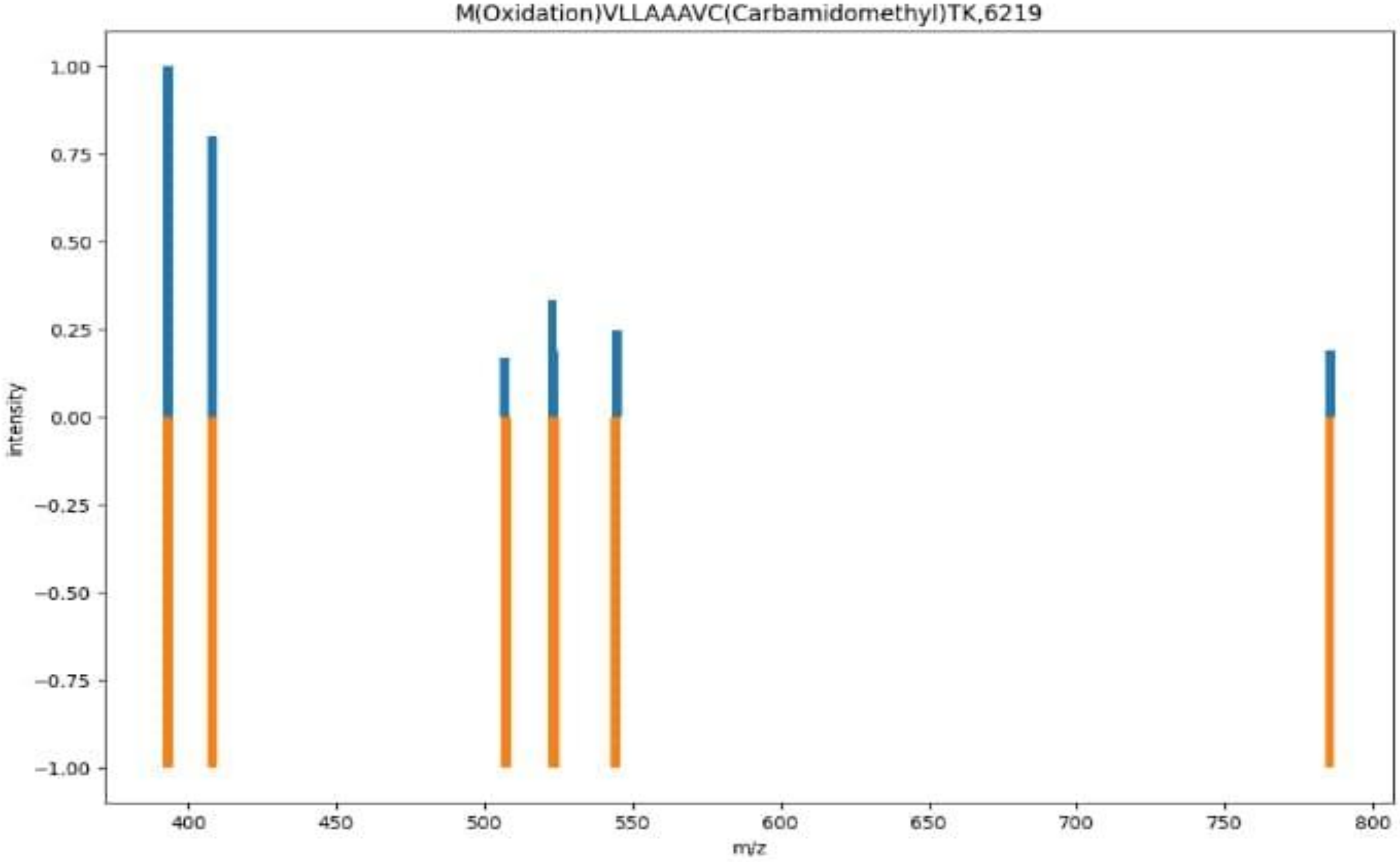

SFPVNSDVGVLK,7522
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
intensity
400
500
600
700
800
m/z

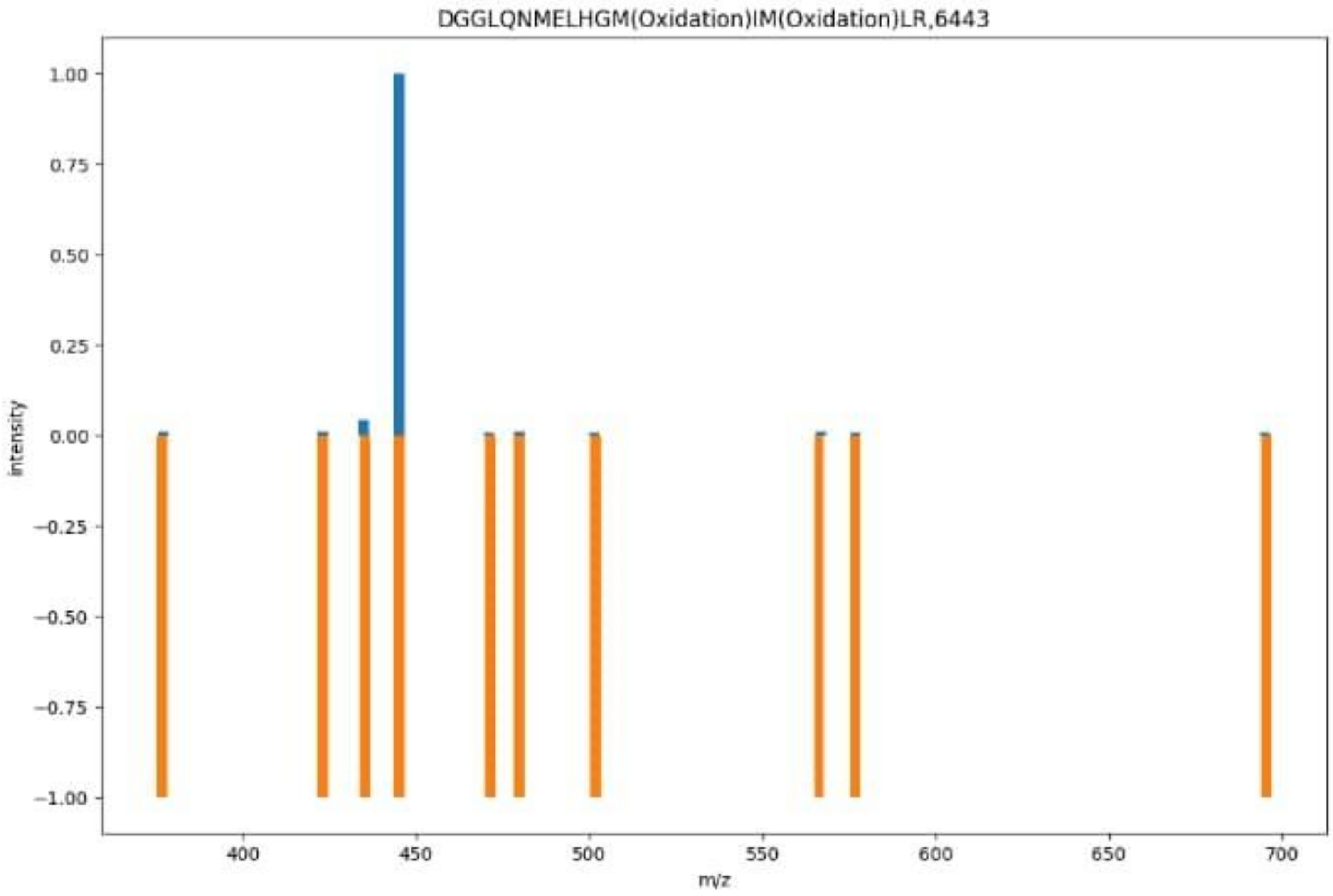

QHTFVETESVR,9203
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
intensity
200
250
300
350
400
450
500
550
600
m/z

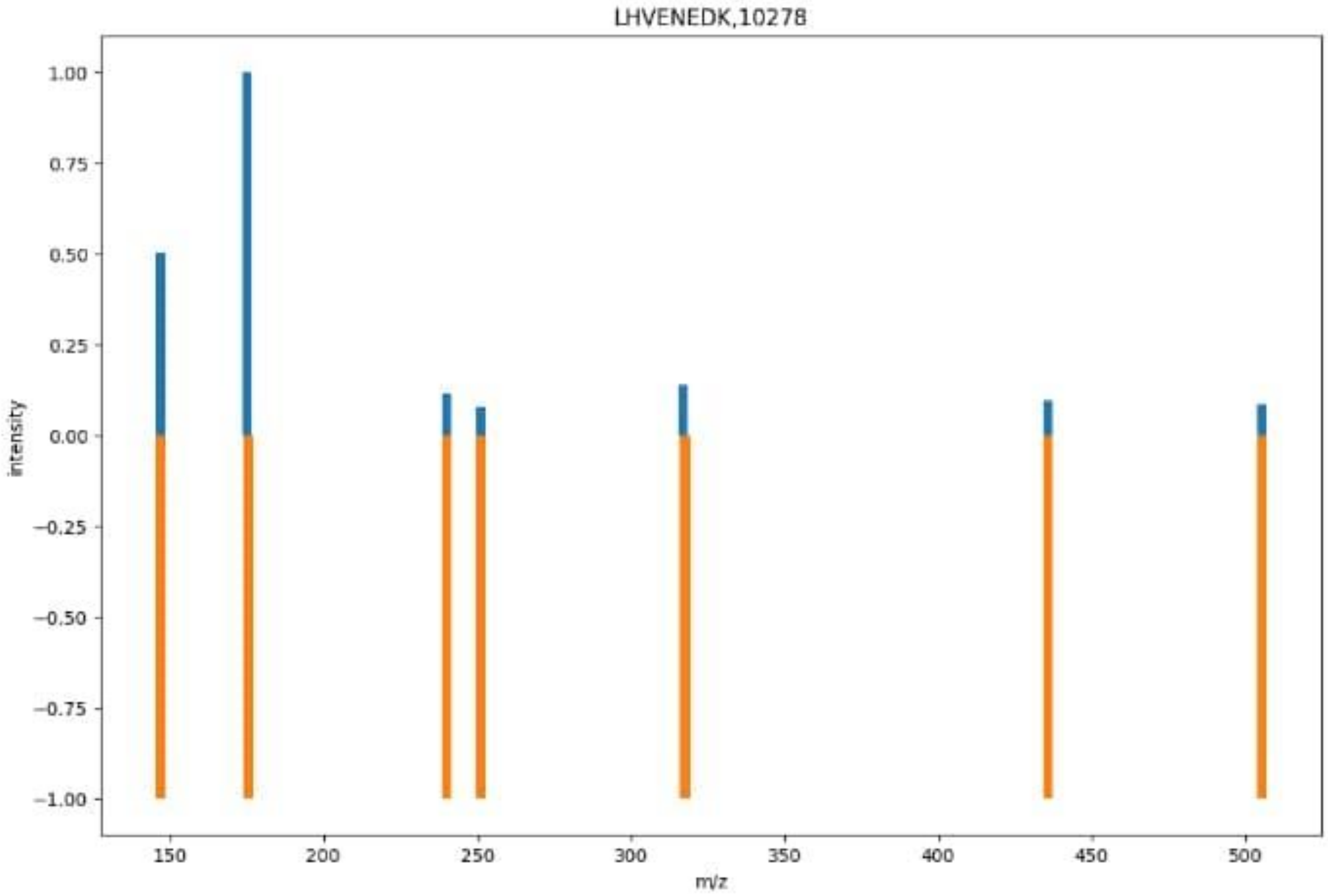

LYM(Oxidation)VLITTK,9436
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
150
200
250
300
350
400
450
m/z

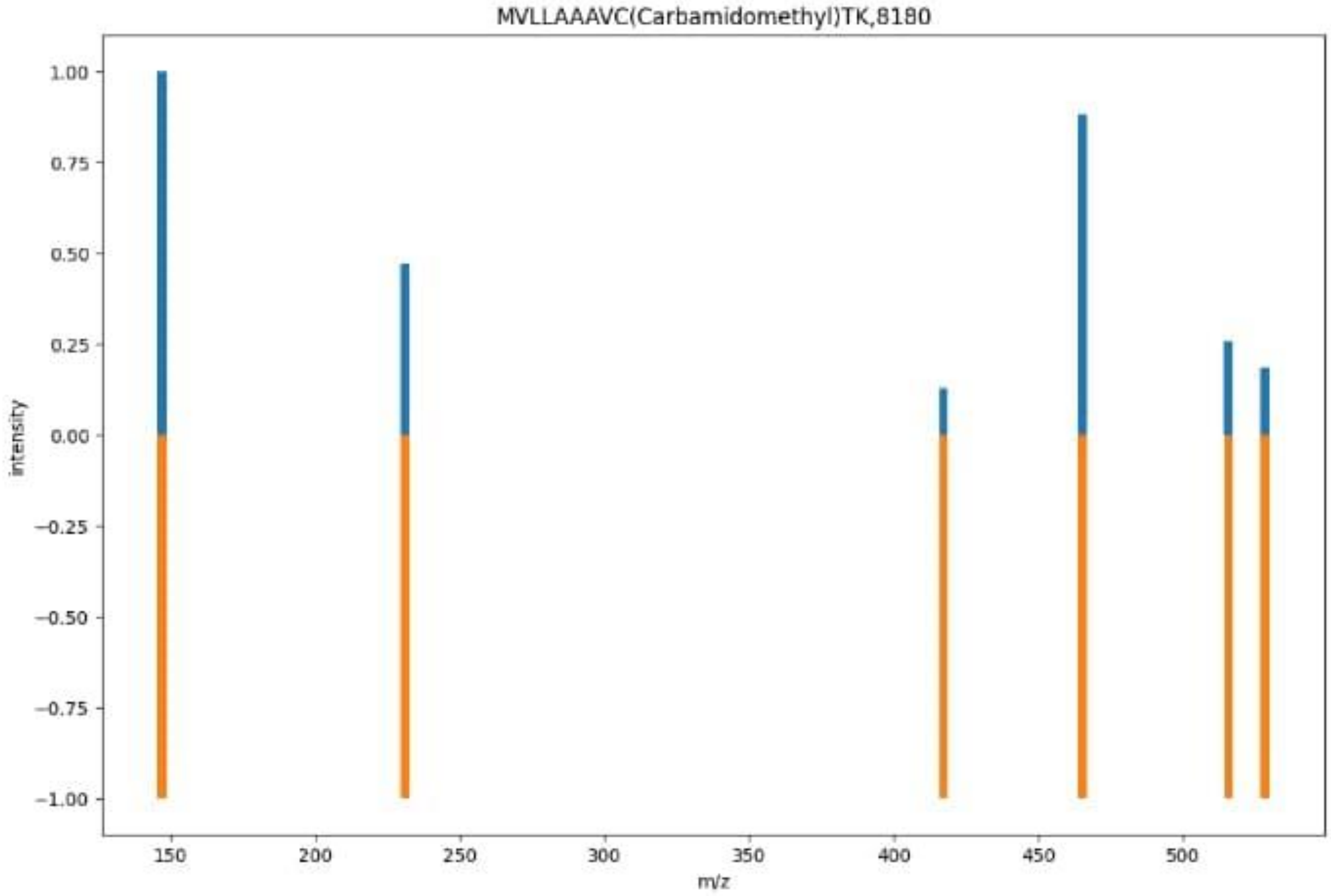

M(Oxidation)VLLAAAVC(Carbamidomethyl)TK,6219
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
450
500
550
600
650
700
750
800
m/z

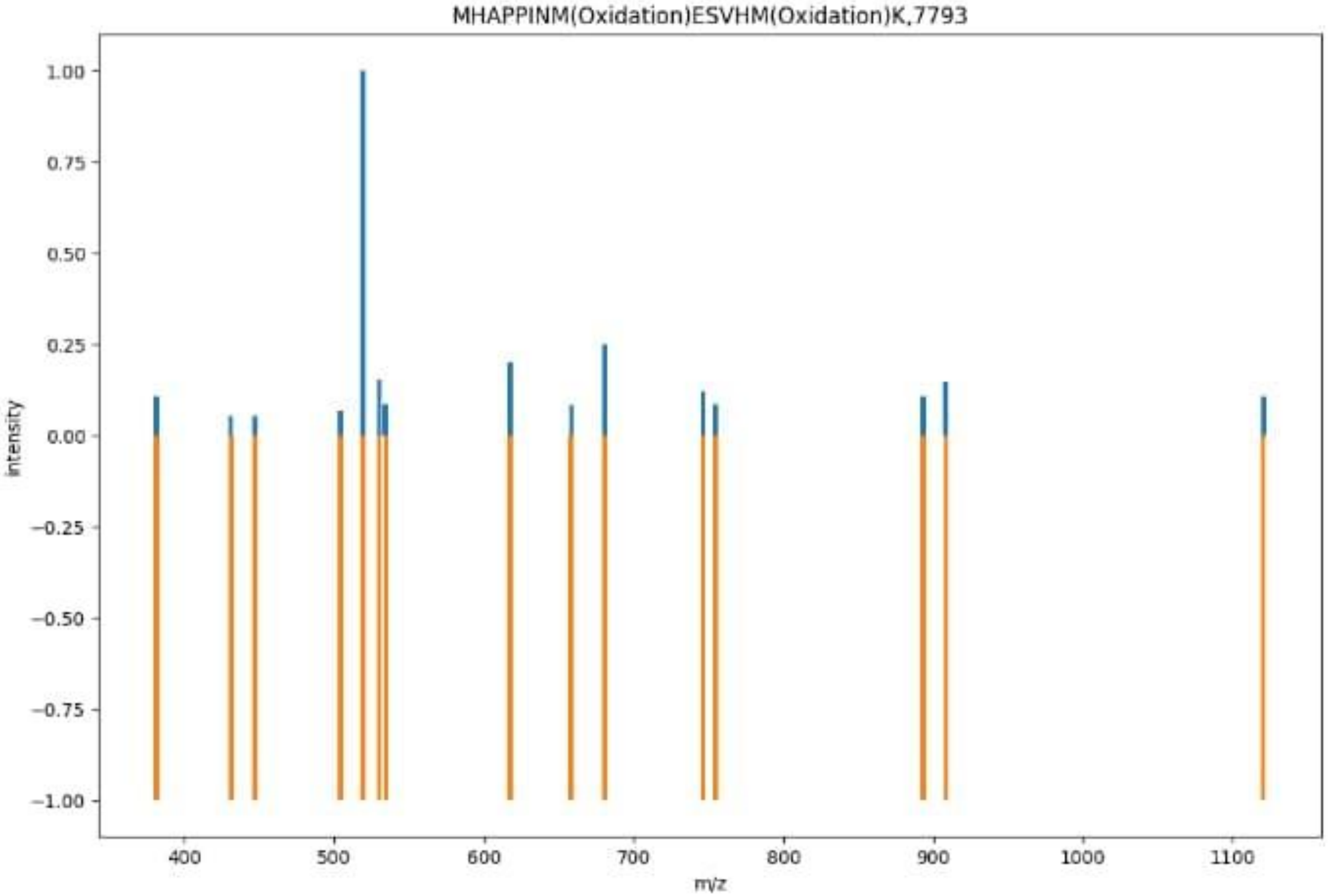
File : 061818-COPD-152-01.mzML




DGGLQNMELHGM(Oxidation)IM(Oxidation)LR,6443
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
450
500
550
600
650
700
m/z

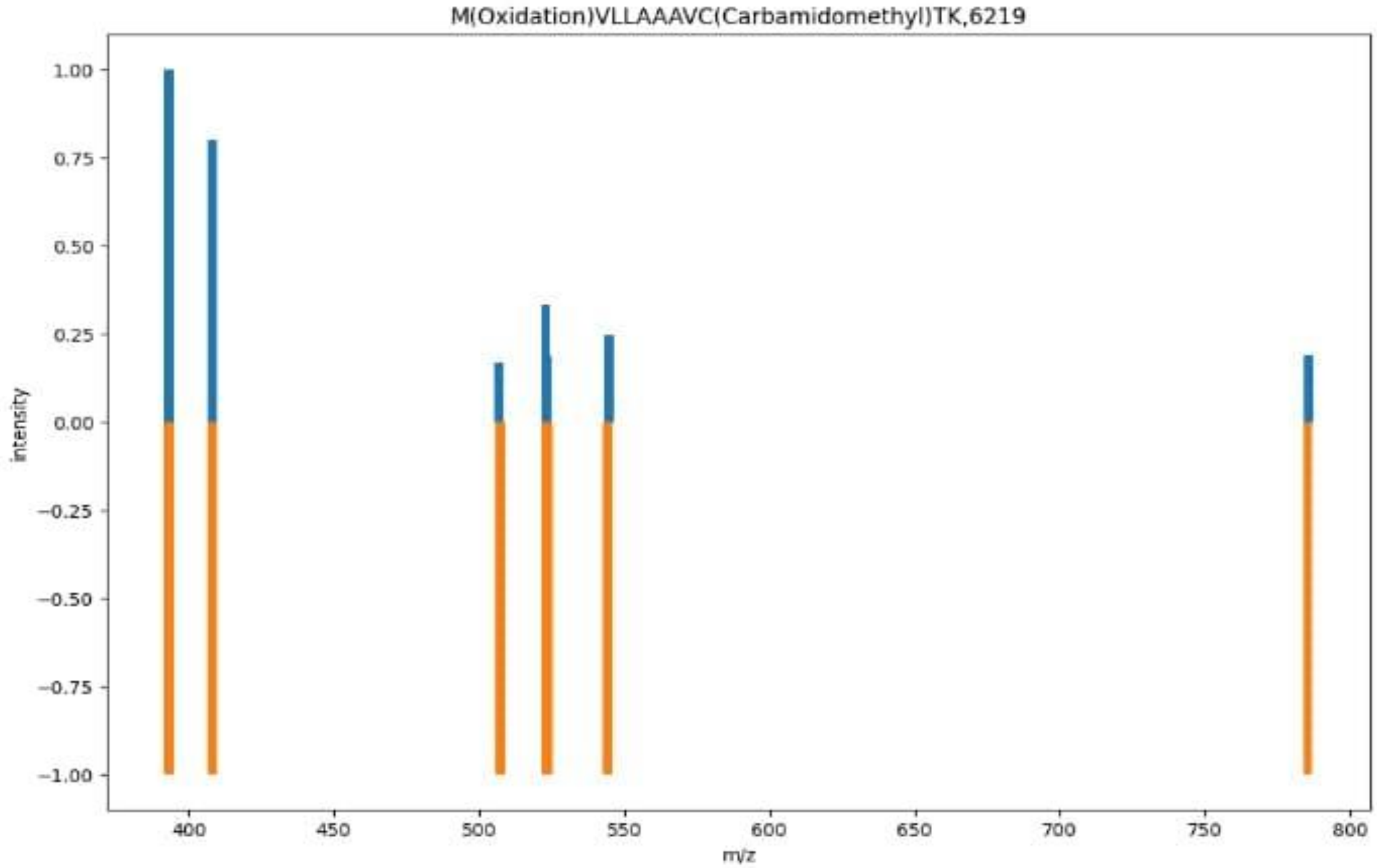

LHVENEDK,10278
intensity
m/z
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
150
200
250
300
350
400
450
500

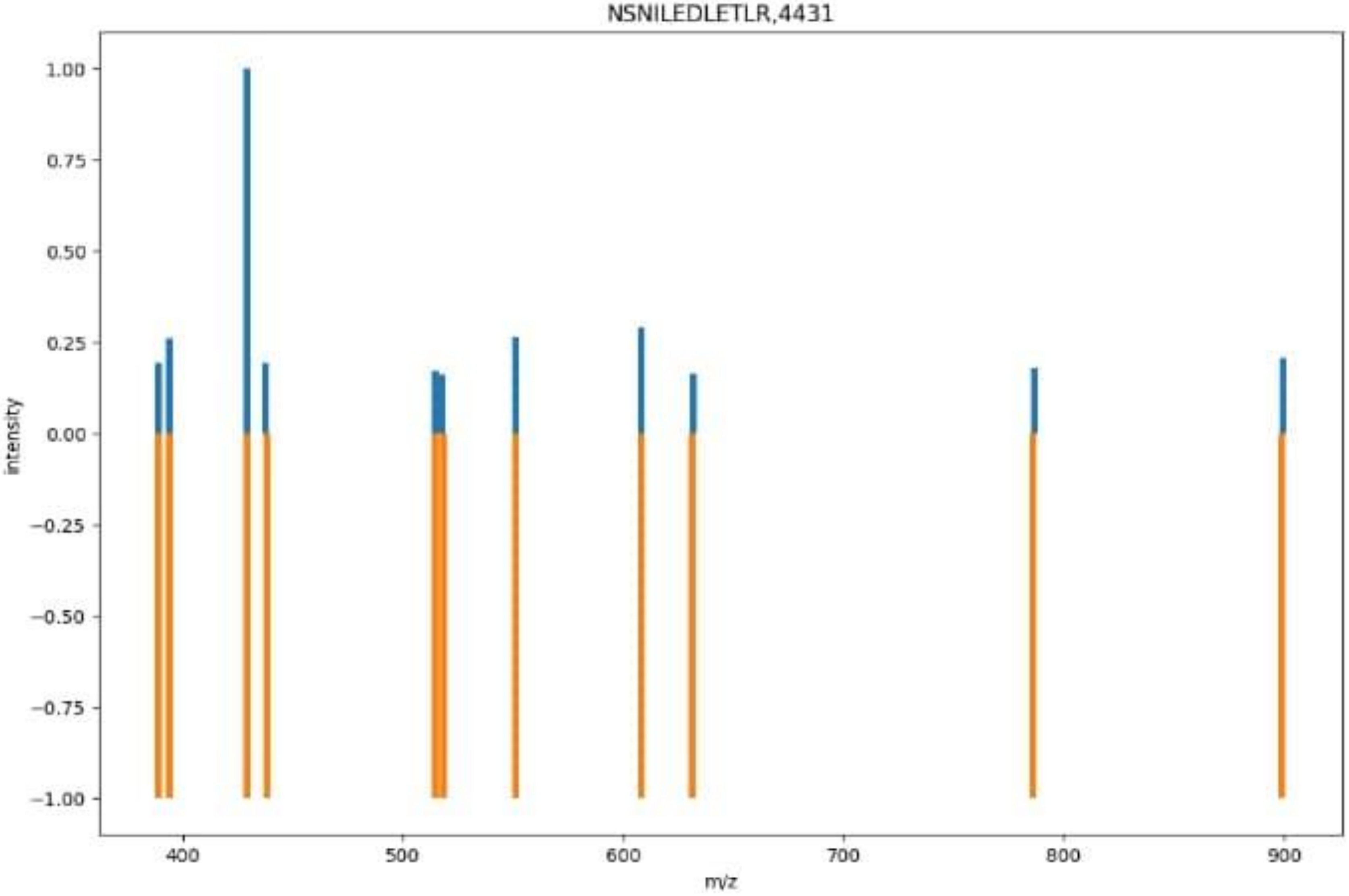

MVLLAAAVC(Carbamidomethyl)TK,8180
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
150
200
250
300
350
400
450
500
m/z

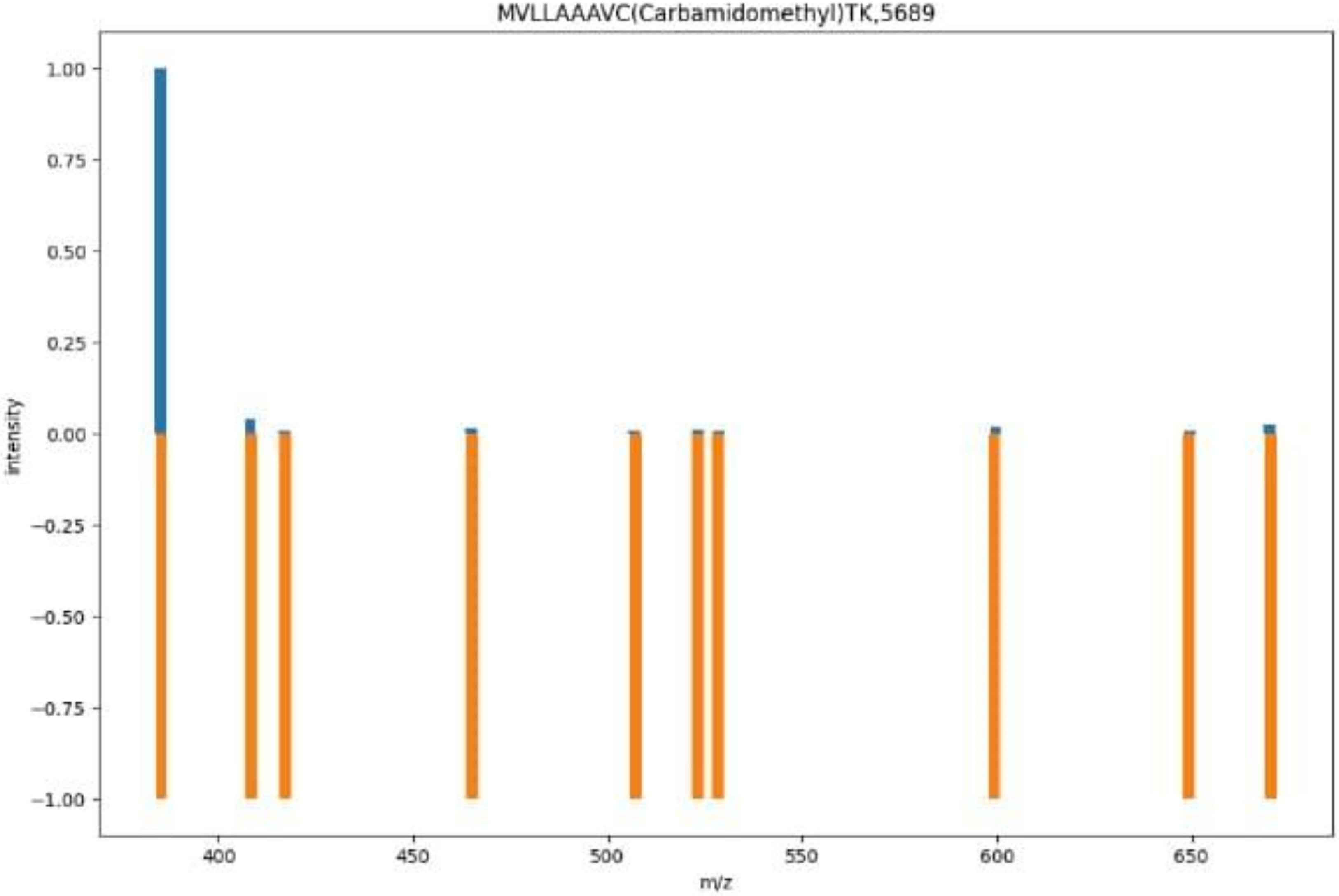


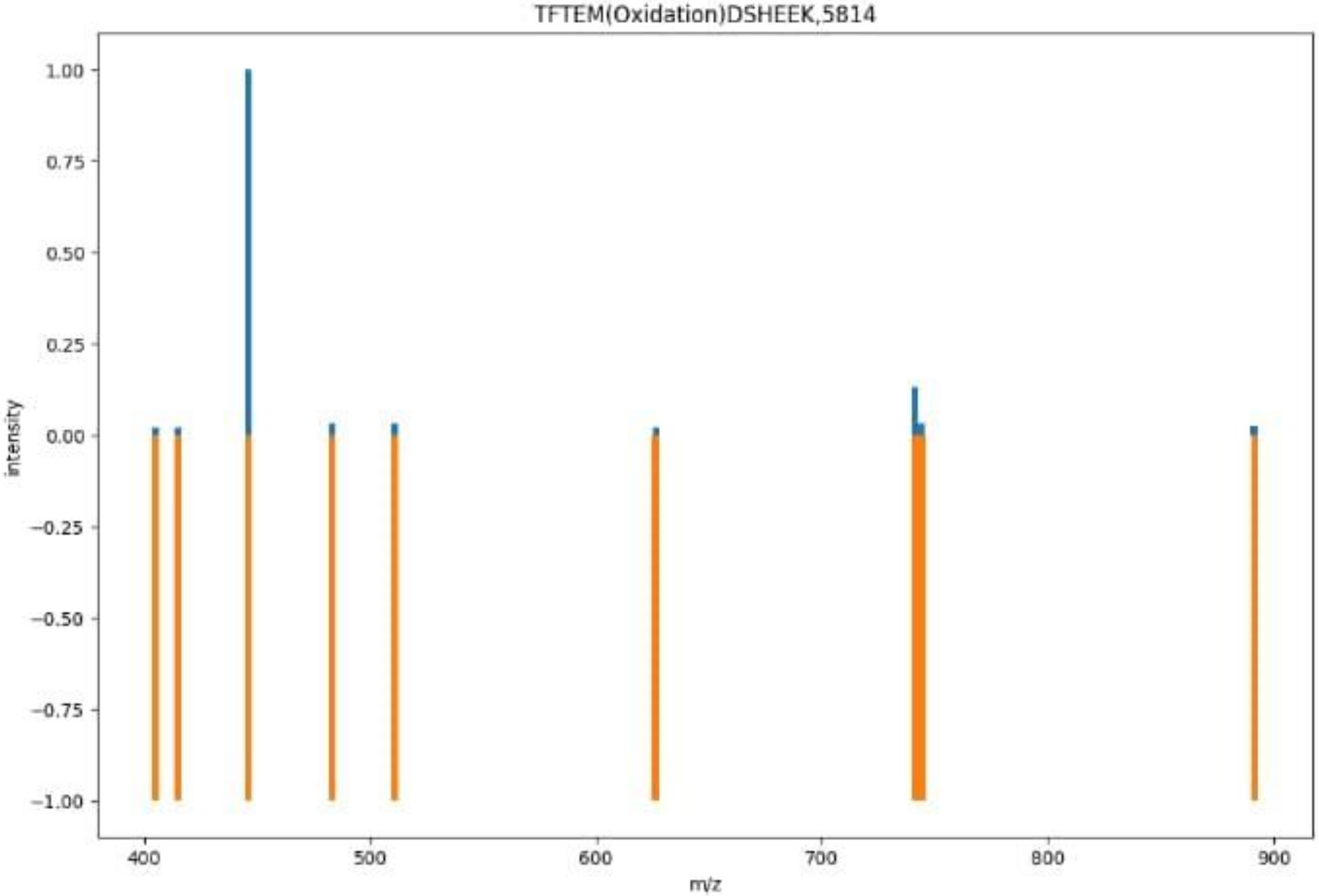
MHAPPINM(Oxidation)ESVHM(Oxidation)K,7793

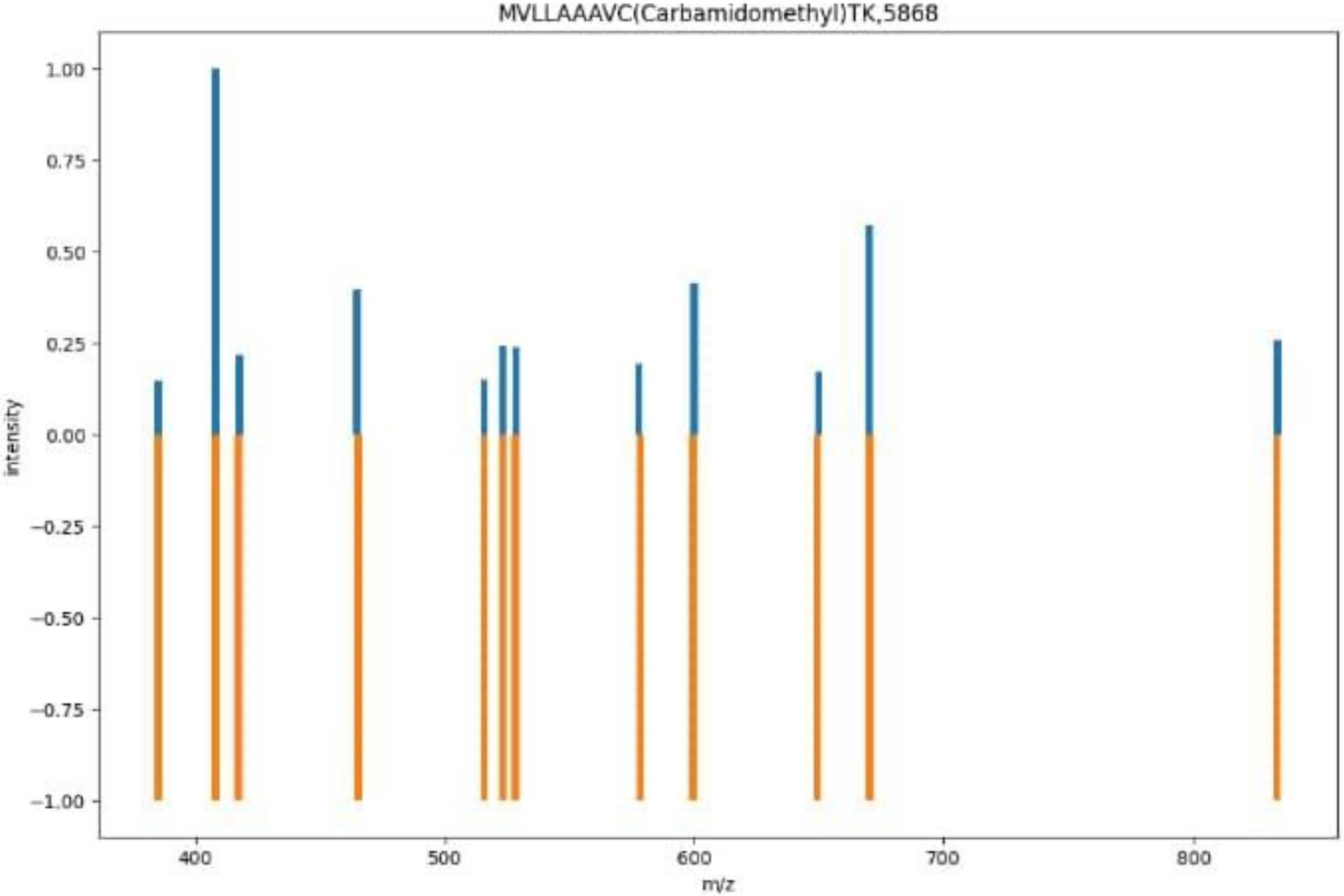
File:061818-COPD-151-04.mzML



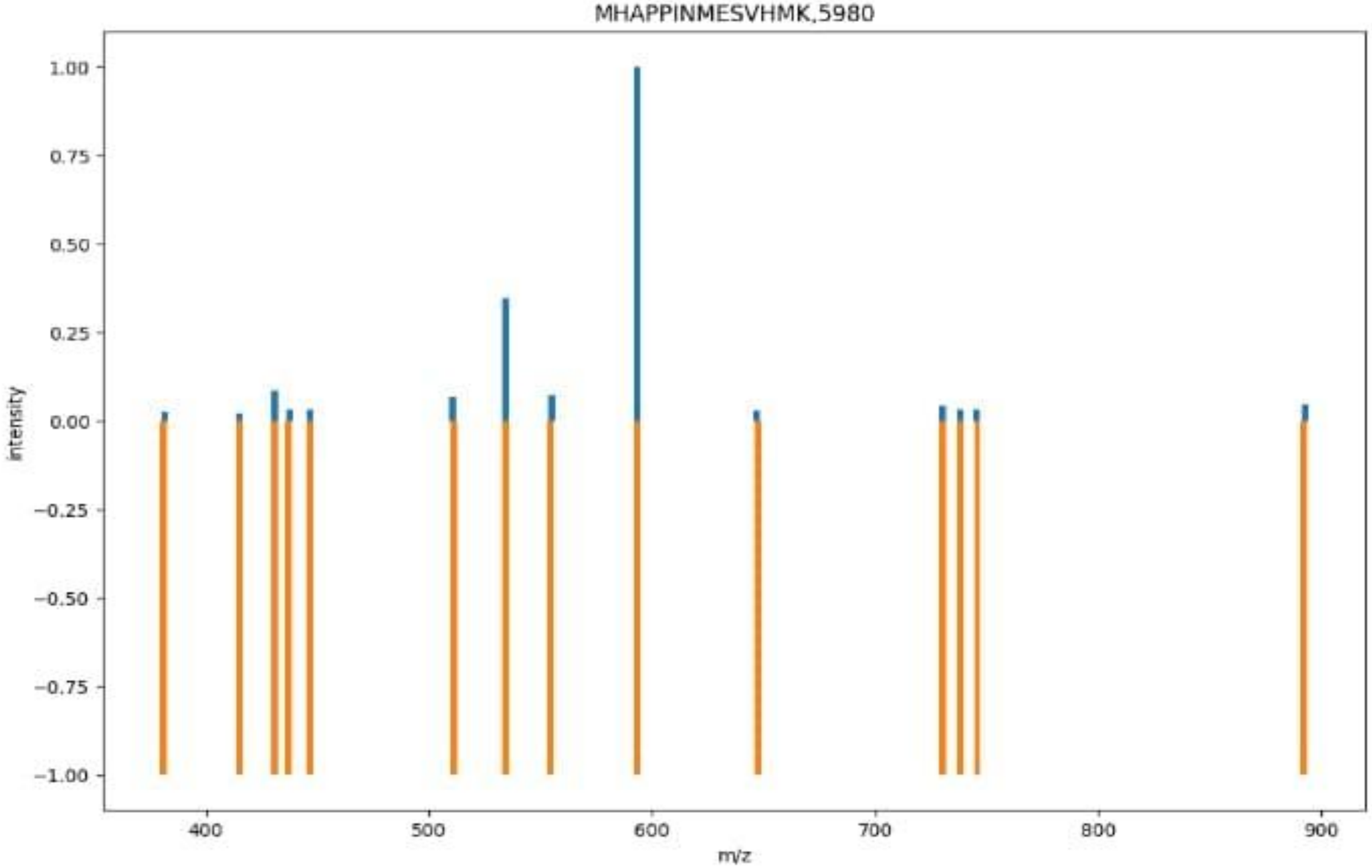

NSNILEDLETLR,4431
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
500
600
700
800
900
m/z

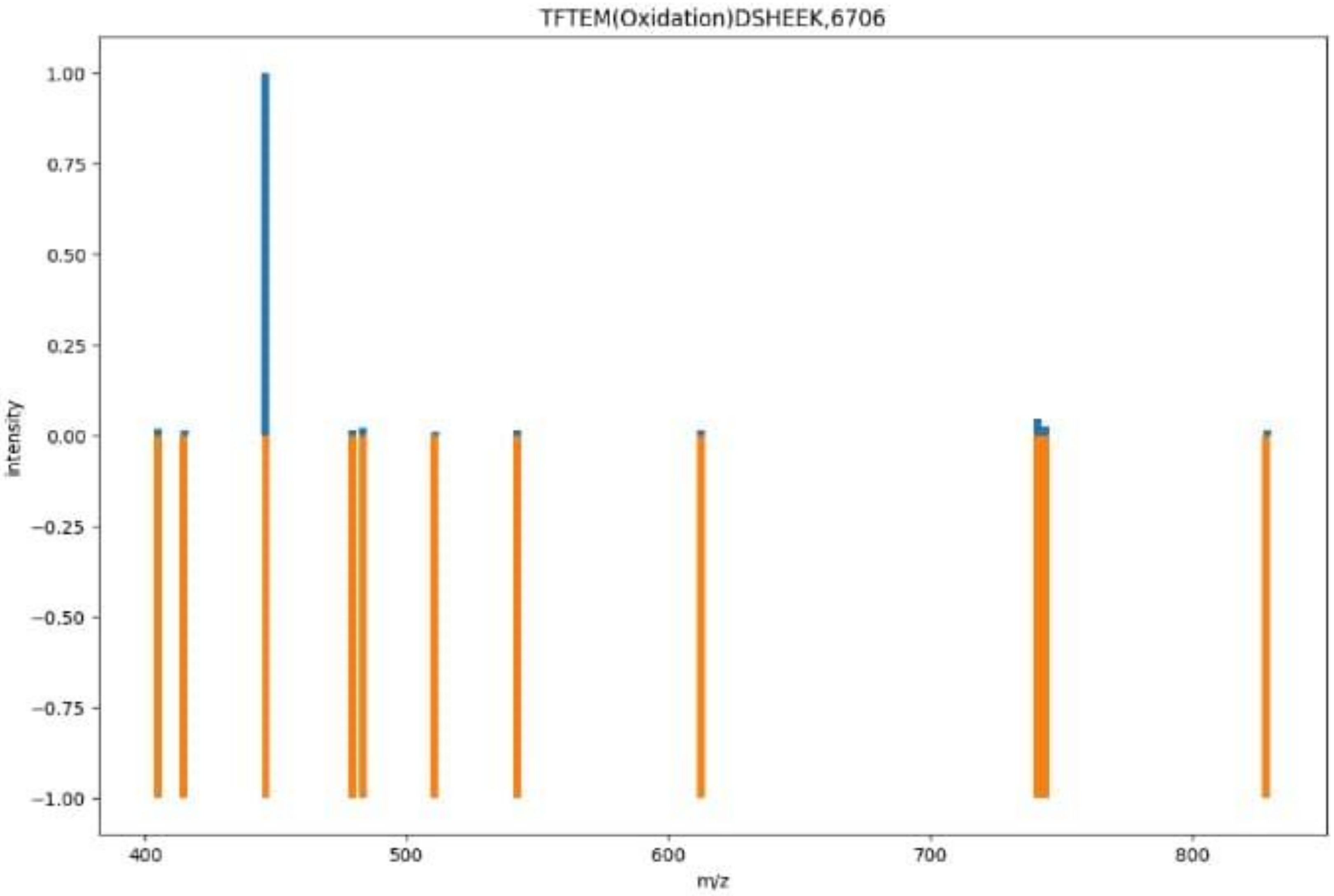

MVLLAAAVC(Carbamidomethyl)TK,5689
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
450
500
550
600
650
m/z

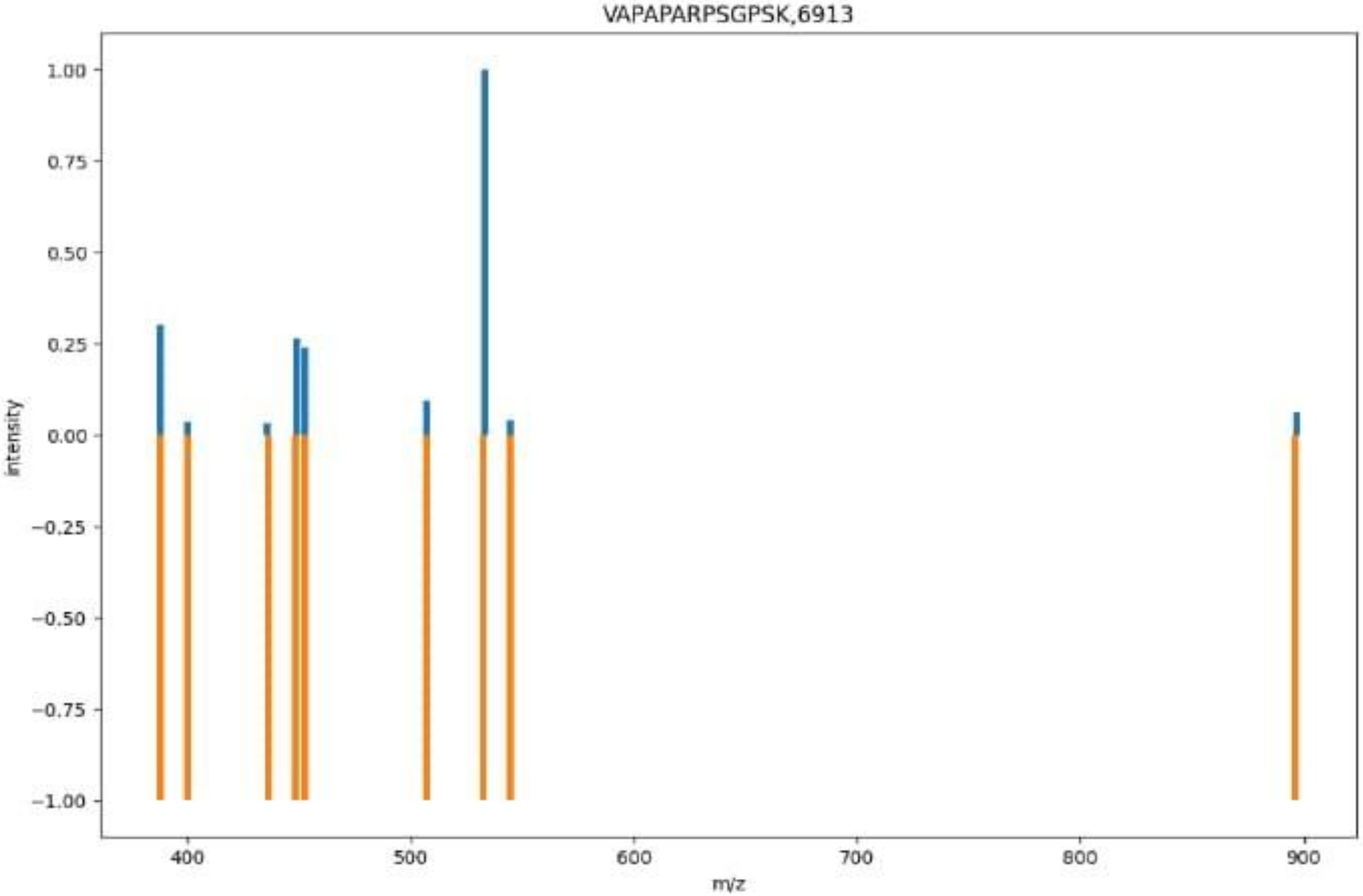

TFTEM(Oxidation)DSHEEK,5814
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
500
600
700
800
900
m/z

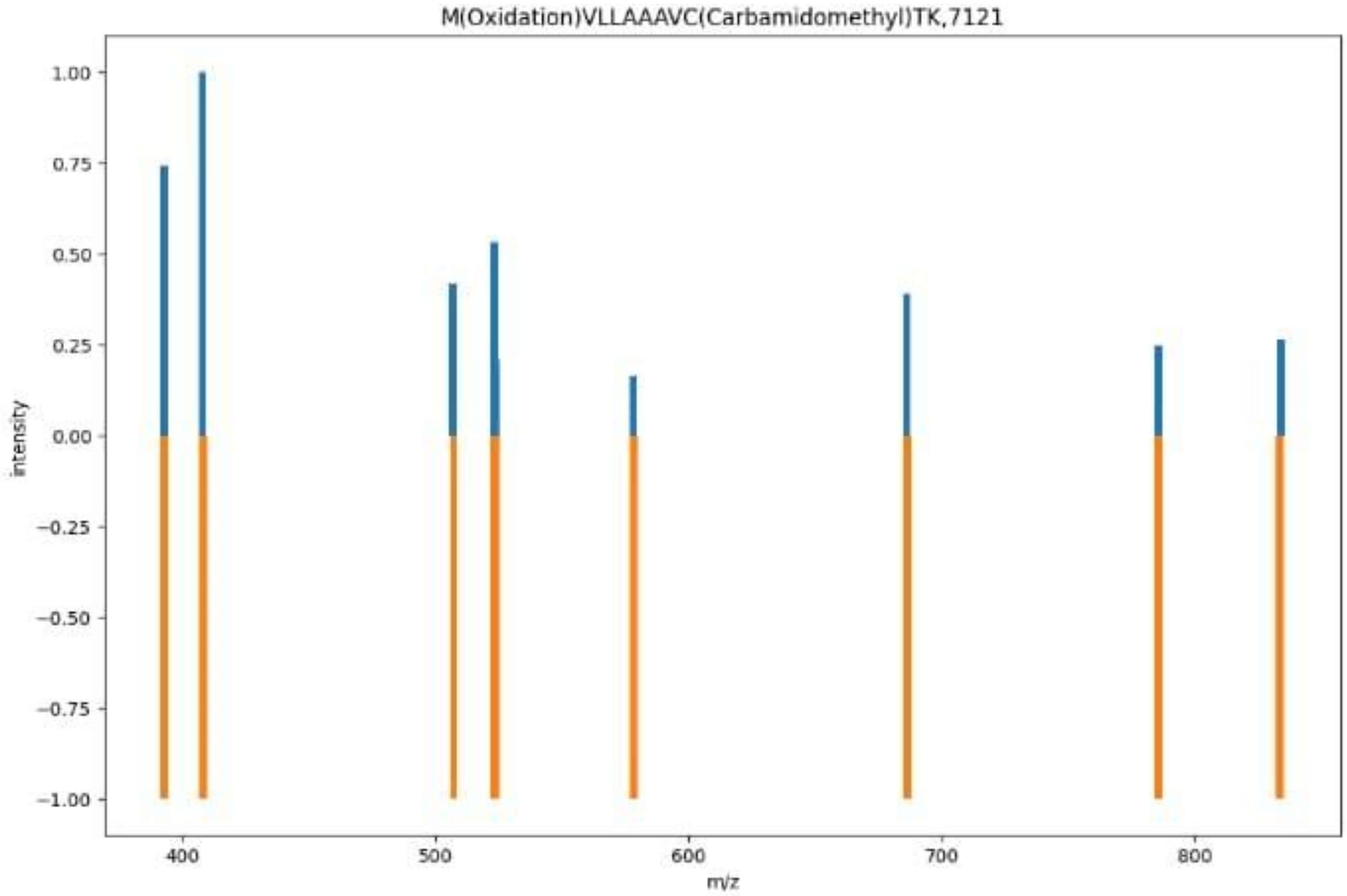

MVLLAAAVC(Carbamidomethyl)TK,5868
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
intensity
400
500
600
700
800
m/z

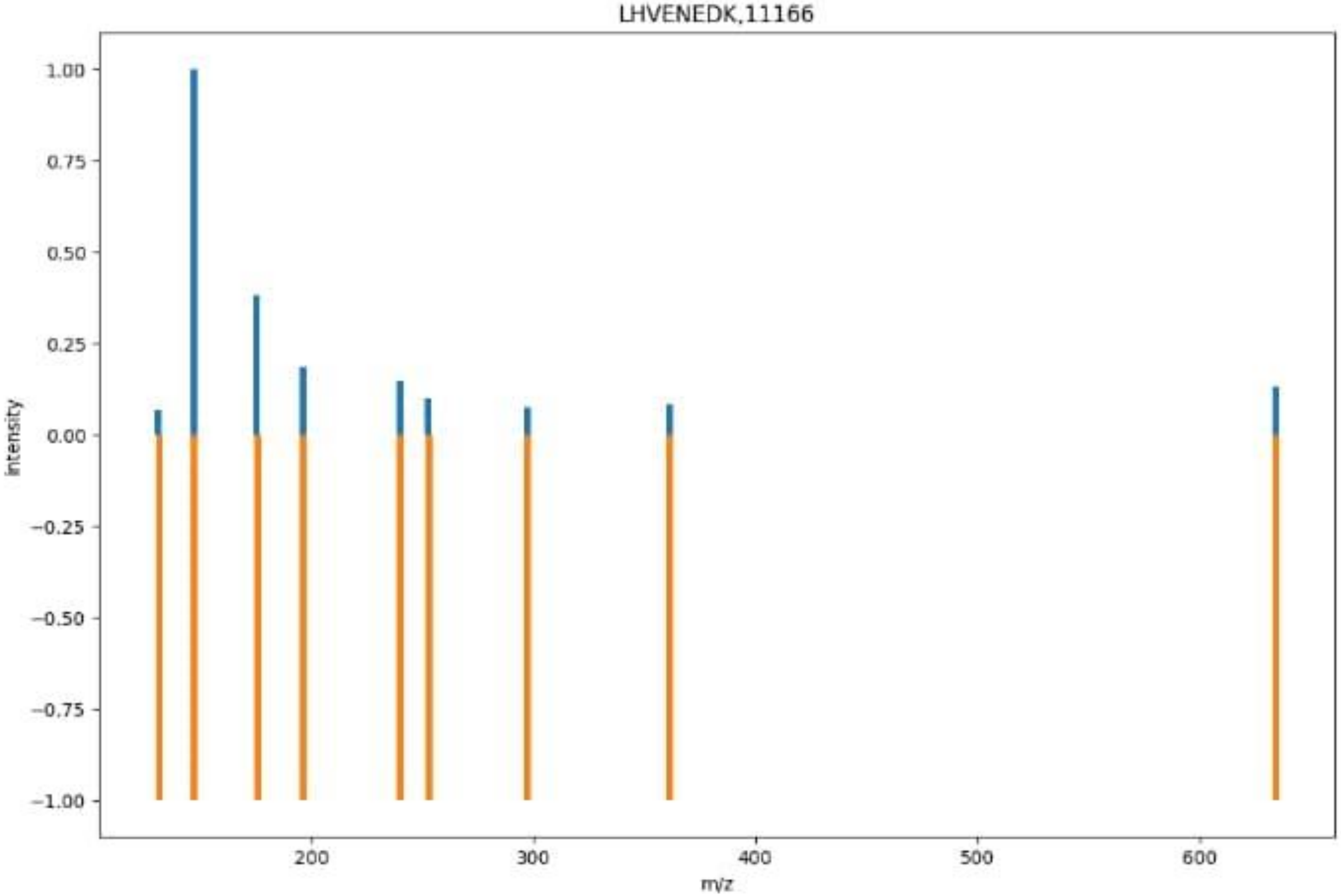
File: 061818-COPD-150-05.mzML



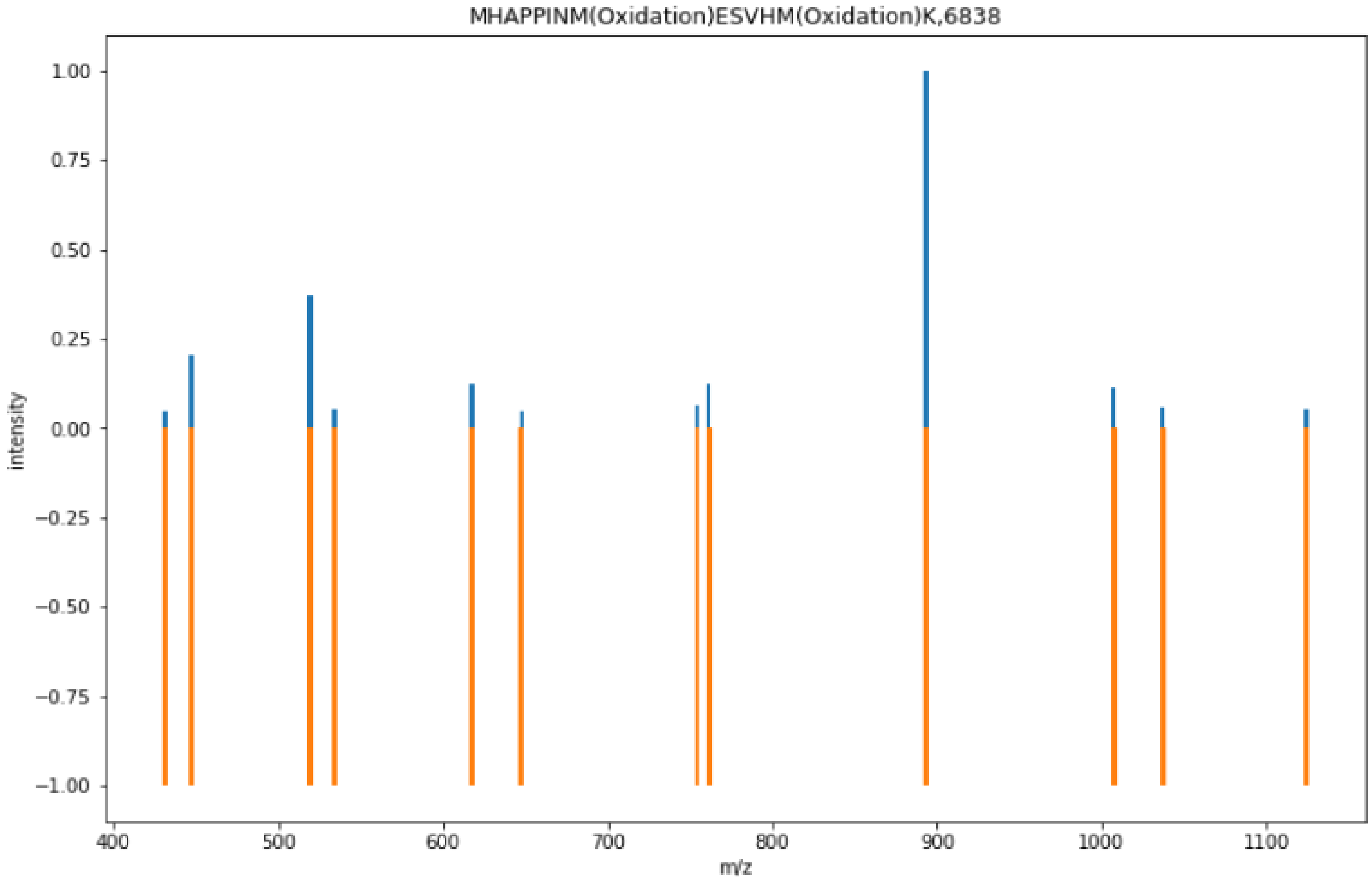

TFTEM(Oxidation)DSHEEK,6706
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
500
600
700
800
m/z

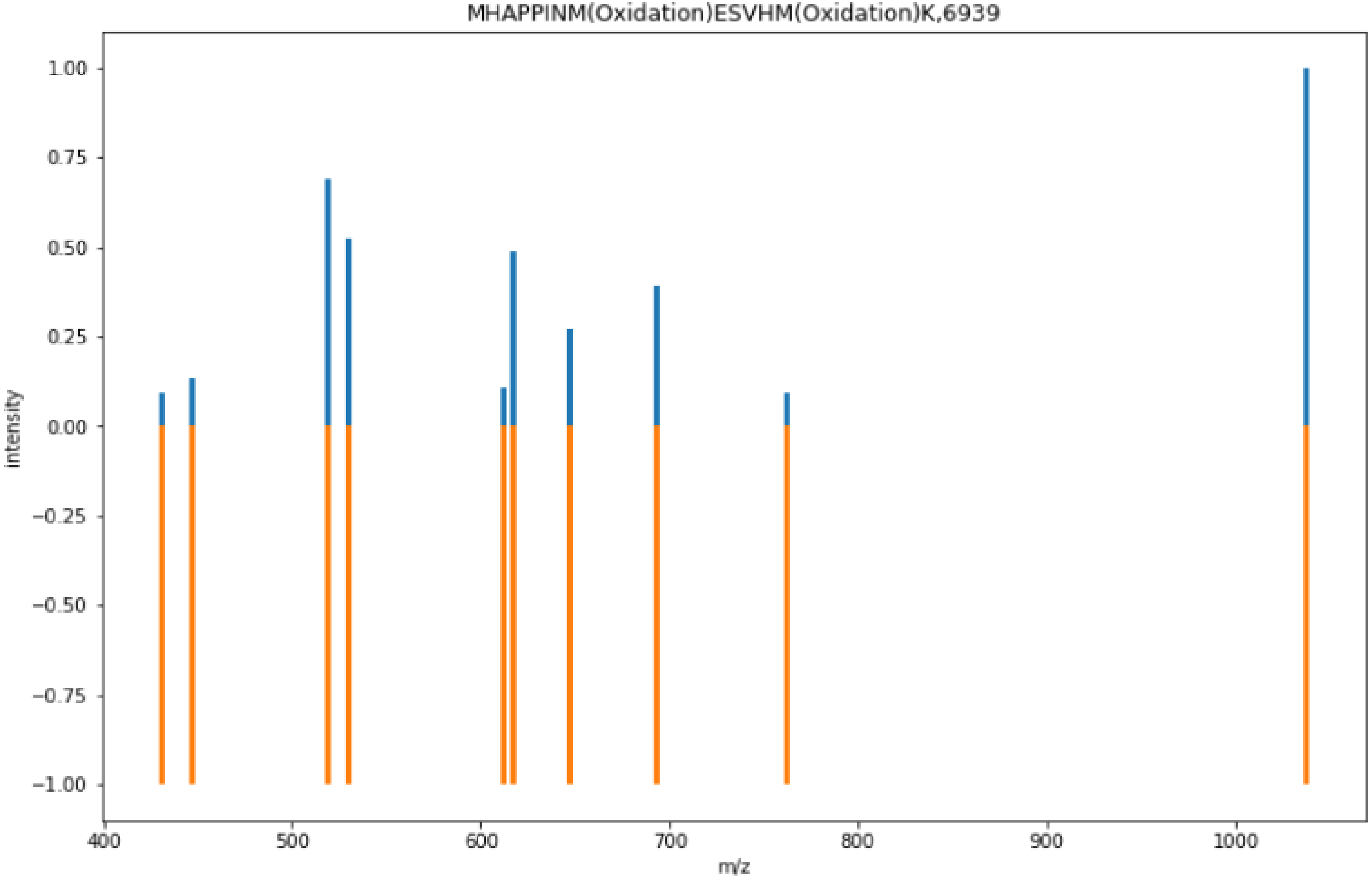

VAPAPARPSGPSK,6913
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
500
600
700
800
900
m/z

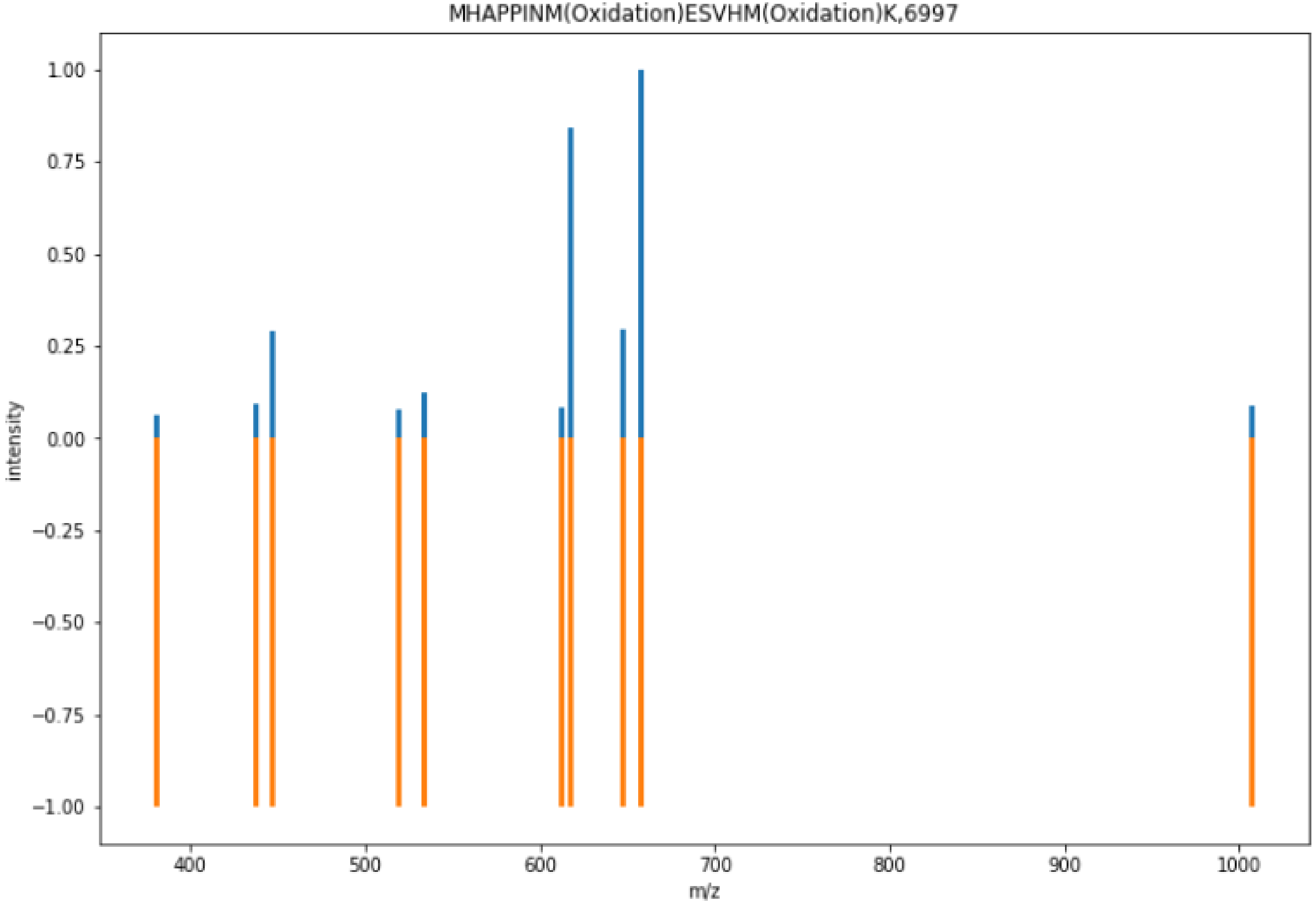

M(Oxidation)VLLAAAVC(Carbamidomethyl)TK,7121
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
500
600
700
800
m/z

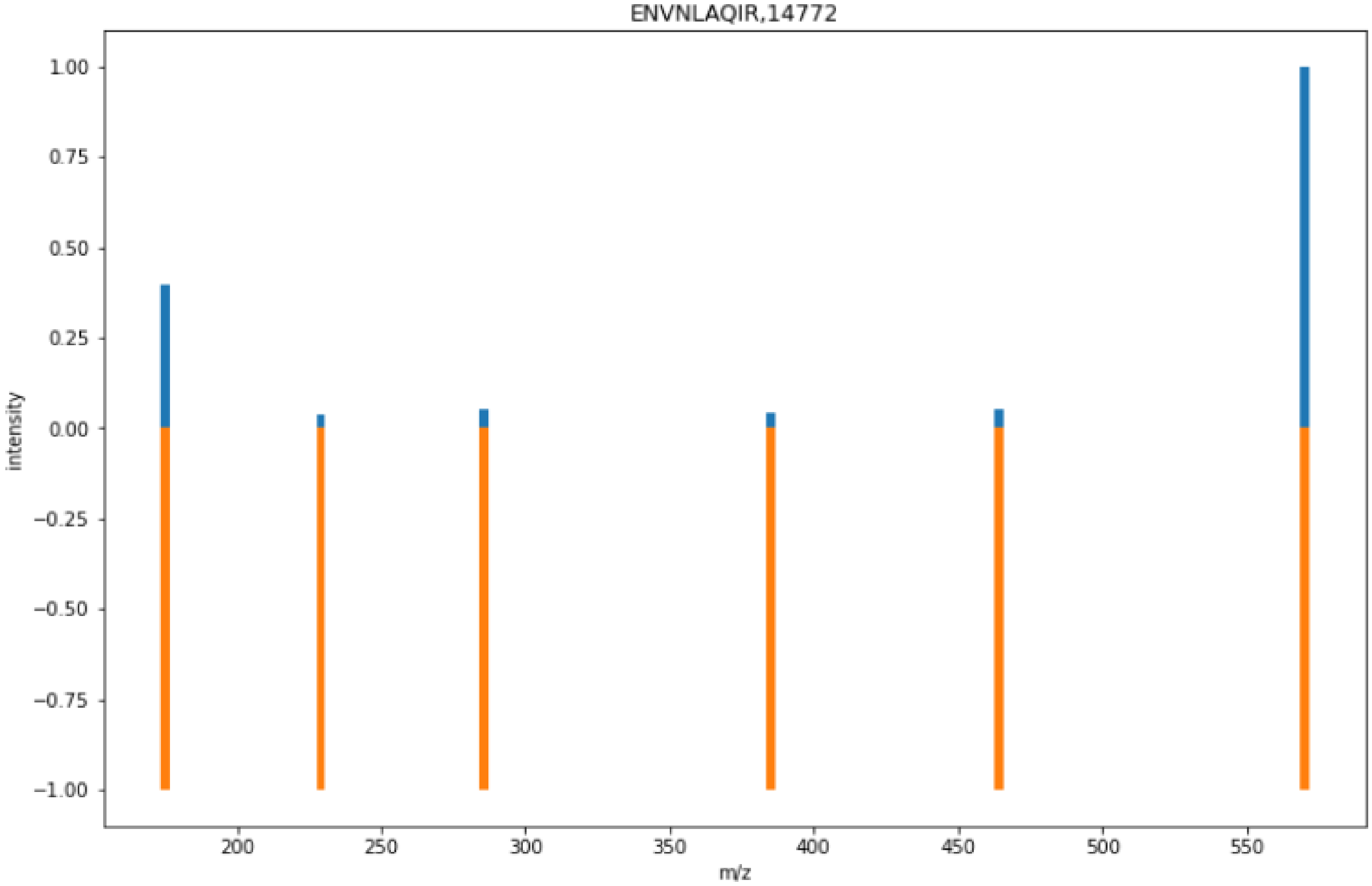

LHVENEDK,11166
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
intensity
200
300
400
500
600
m/z

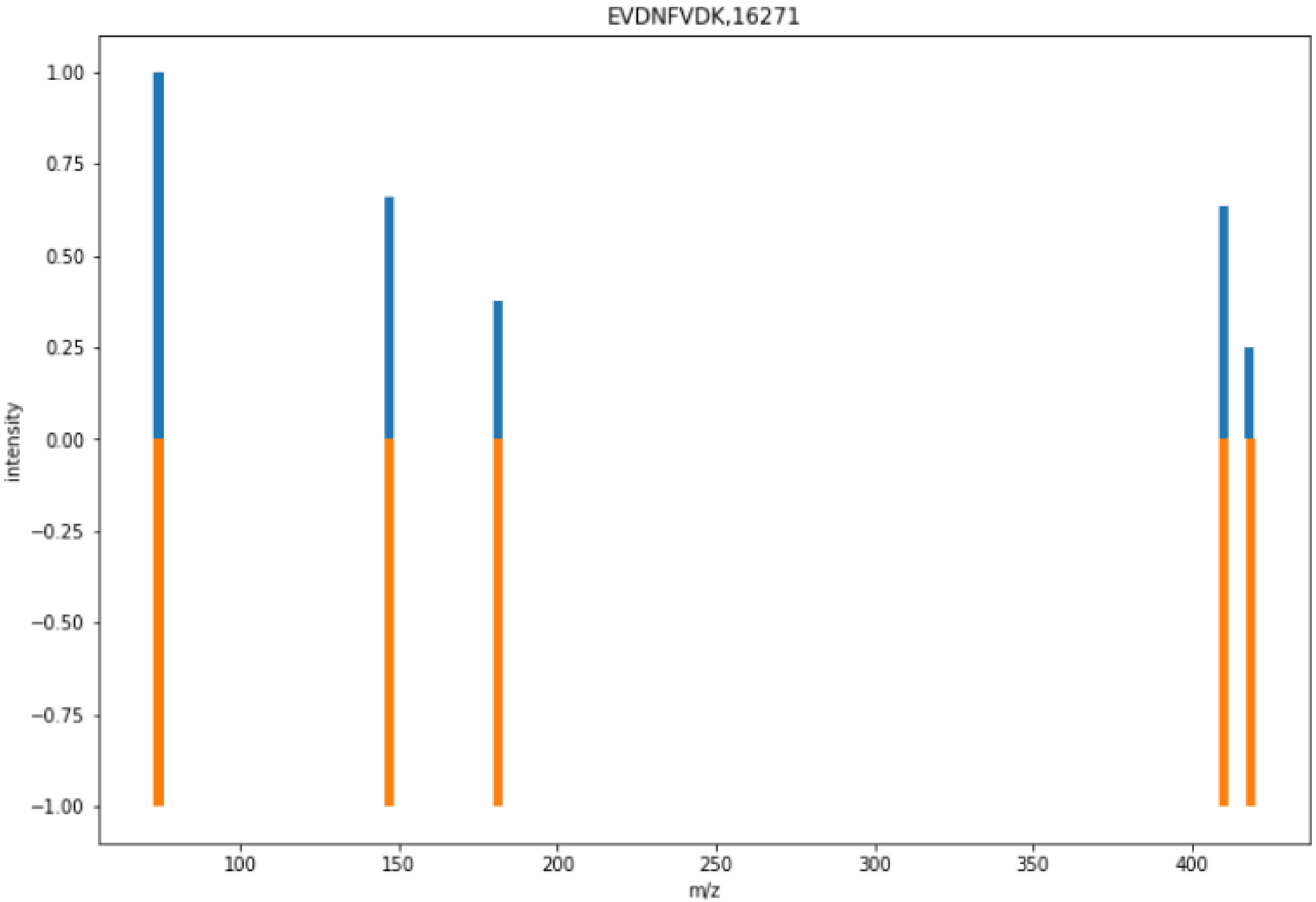
File:061818-COPD-147-02.mzML

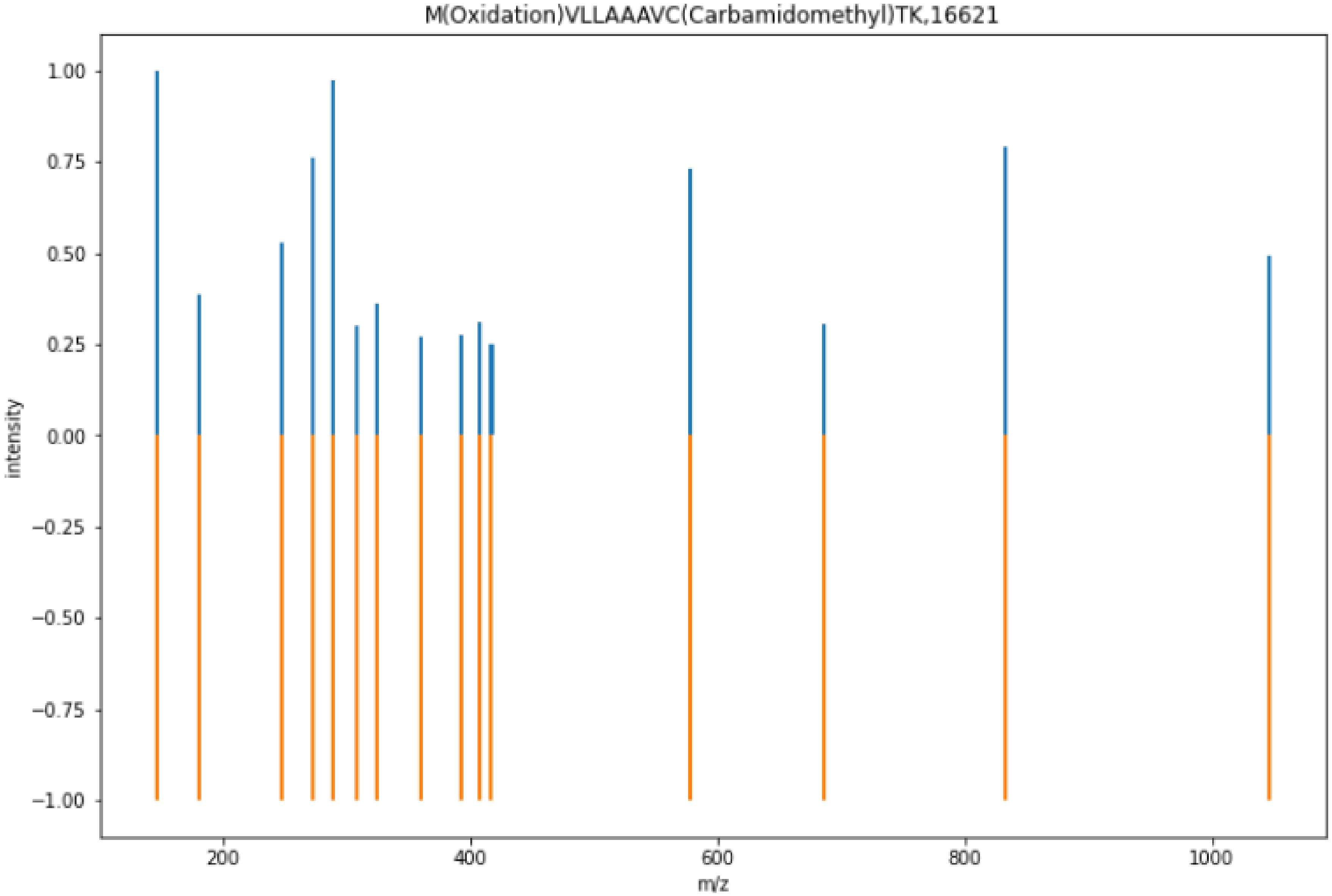



MHAPPINM(Oxidation)ESVHM(Oxidation)K,6939
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
500
600
700
800
900
1000
m/z

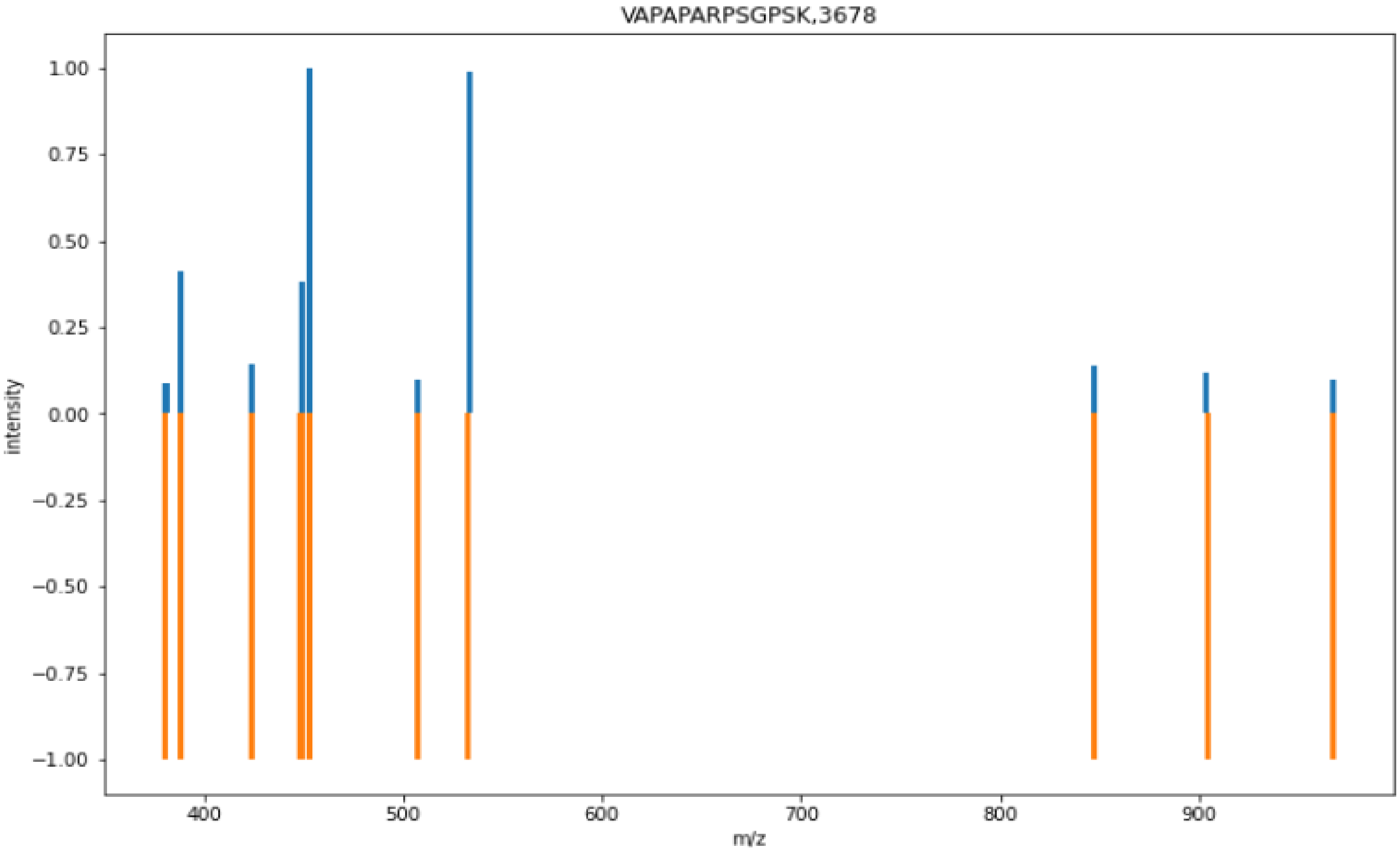

MHAPPINM(Oxidation)ESVHM(Oxidation)K,6997
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
500
600
700
800
900
1000
m/z

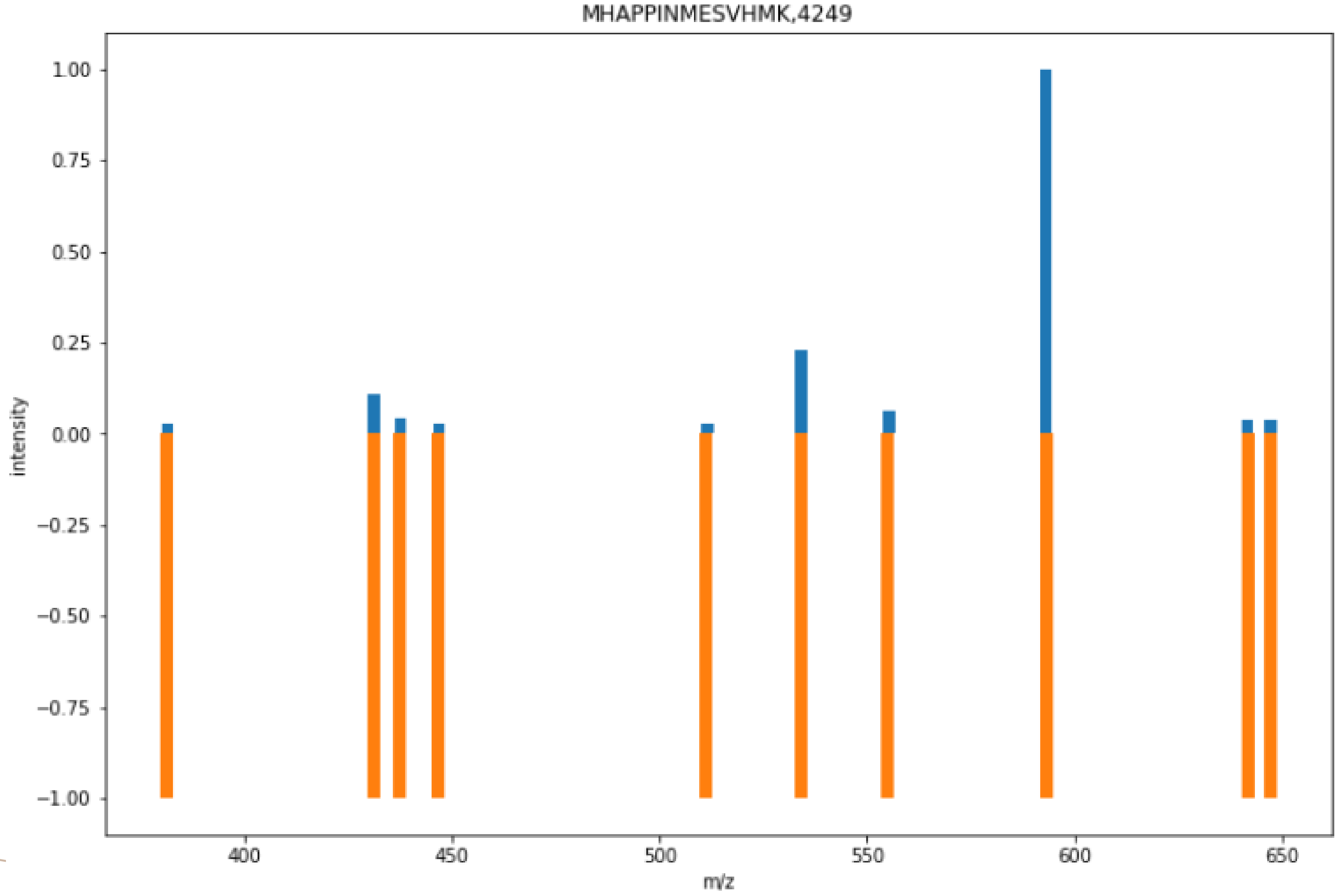

ENVNLAQIR,14772
intensity
m/z
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
200
250
300
350
400
450
500
550

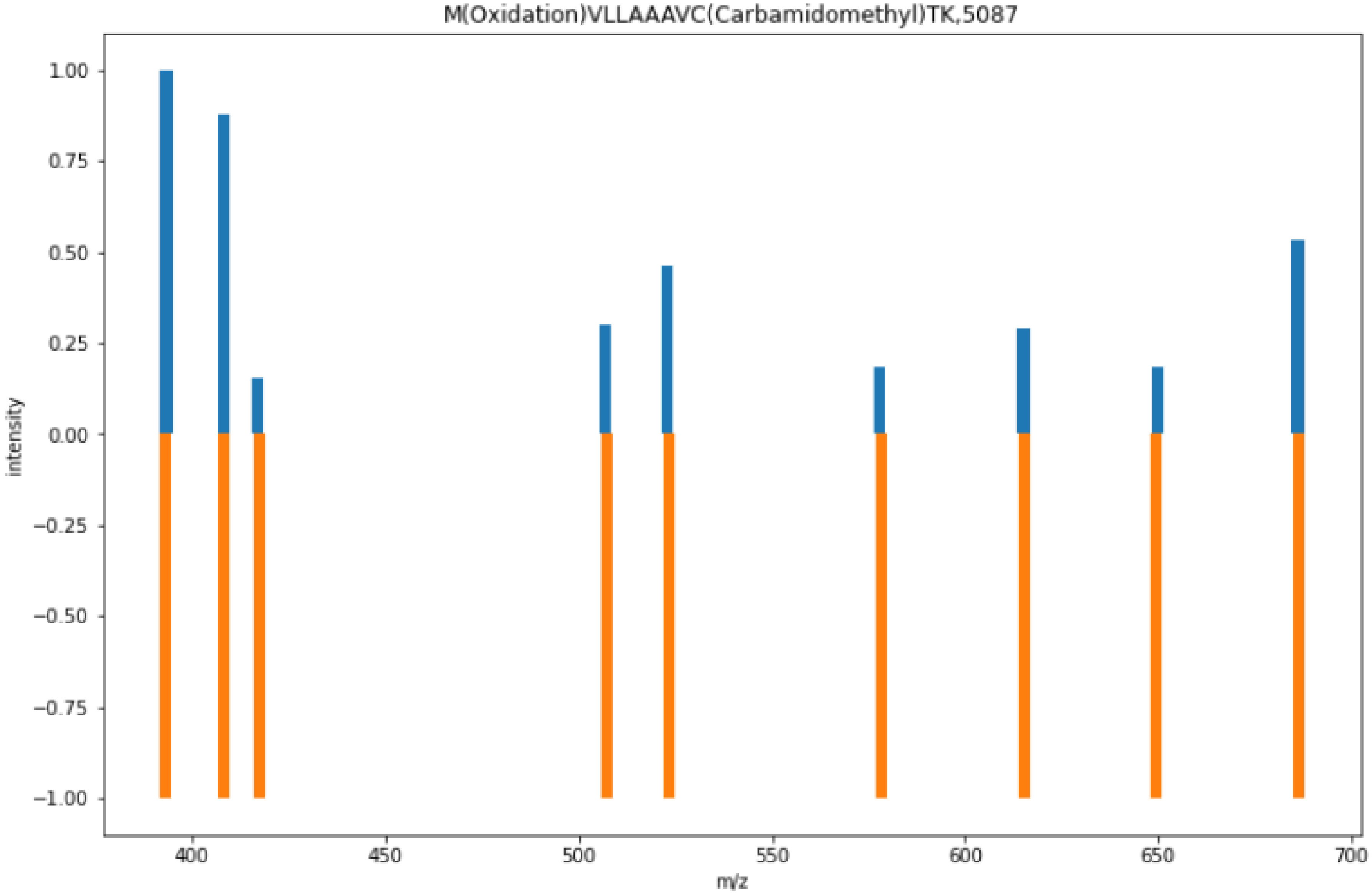

EVDNFVDK,16271
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
intensity
100
150
200
250
300
350
400
m/z

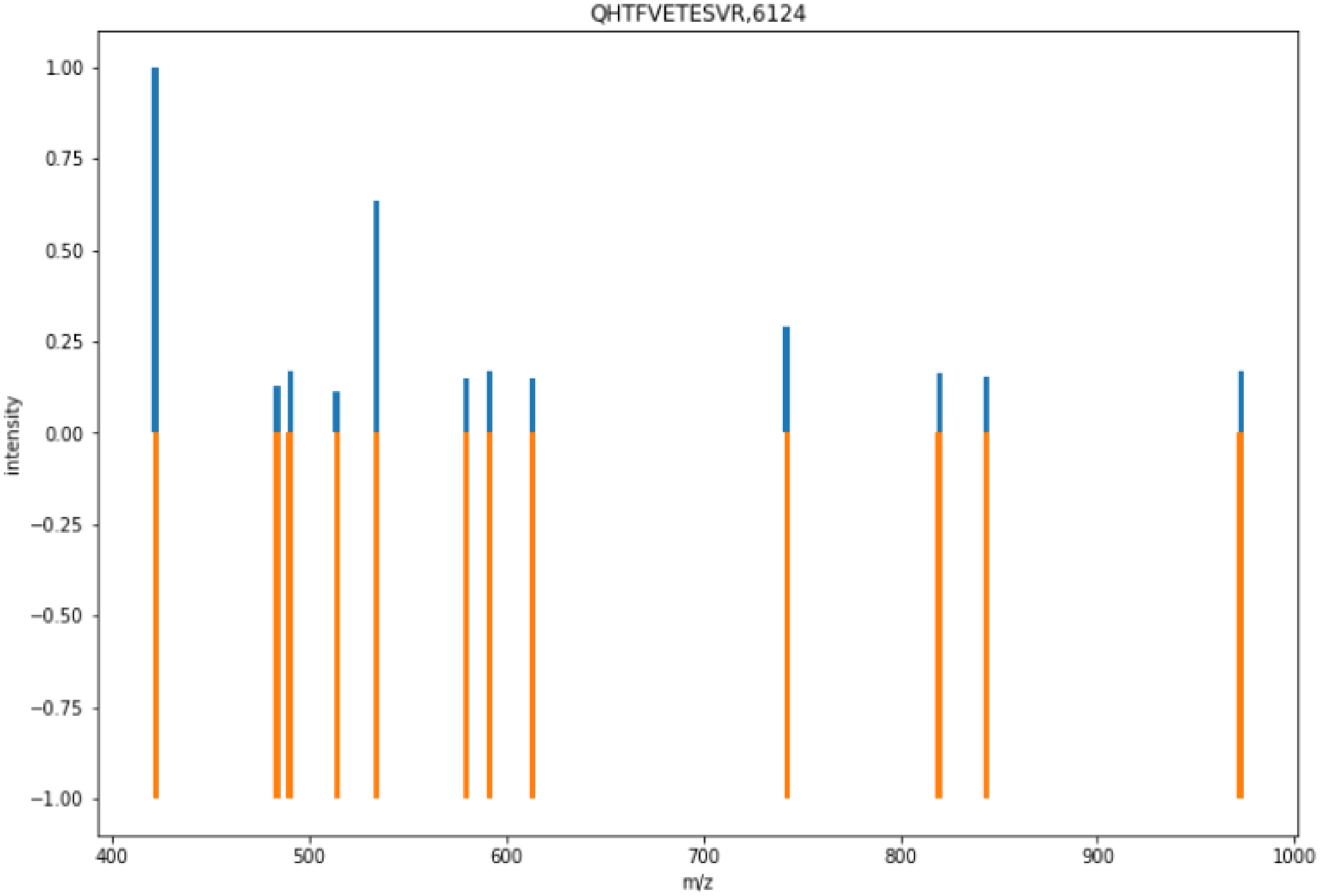

M(Oxidation)VLLAAAVC(Carbamidomethyl)TK,16621
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
200
400
600
800
1000
m/z

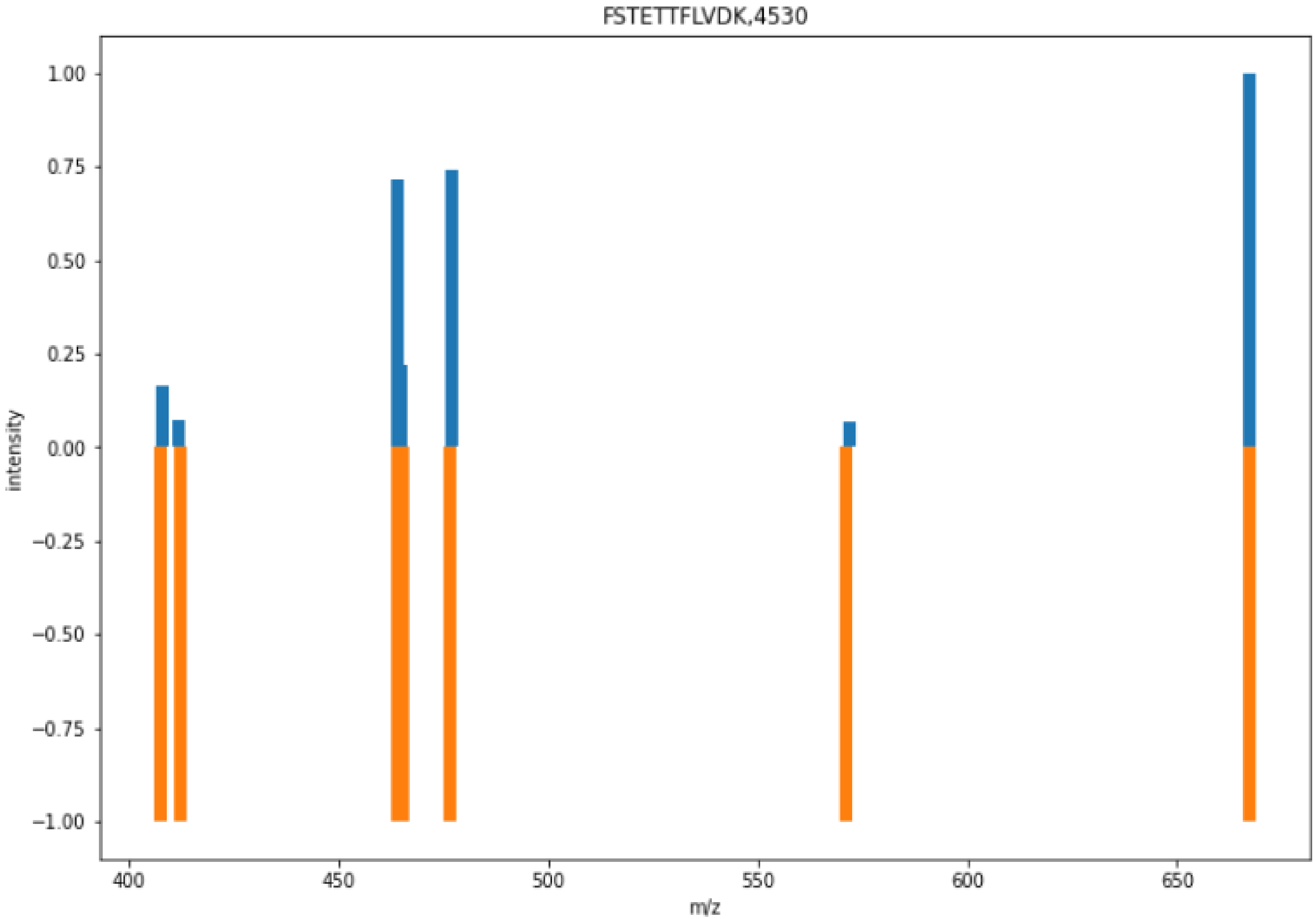
File: a061818-COPD-149-04.mzML

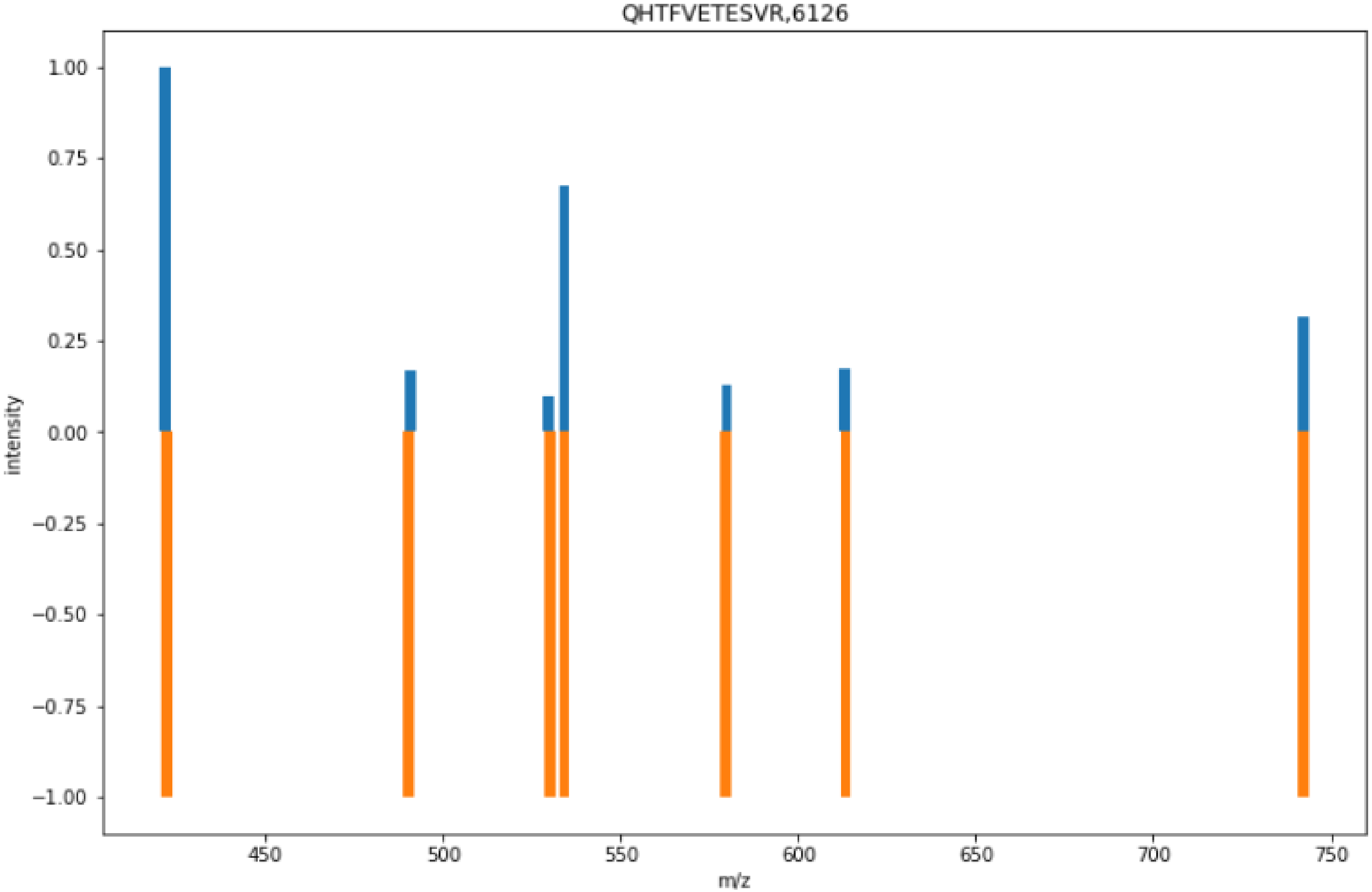



MHAPPINMESVHMK,4249
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
450
500
550
600
650
m/z


M(Oxidation)VLLAAAVC(Carbamidomethyl)TK,5087
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
450
500
550
600
650
700
m/z


QHTFVETESVR,6124
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
intensity
400
500
600
700
800
900
1000
m/z


FSTETTFLVDK,4530
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
400
450
500
550
600
650
m/z


QHTFVETESVR,6126
intensity
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
450
500
550
600
650
700
750
m/z

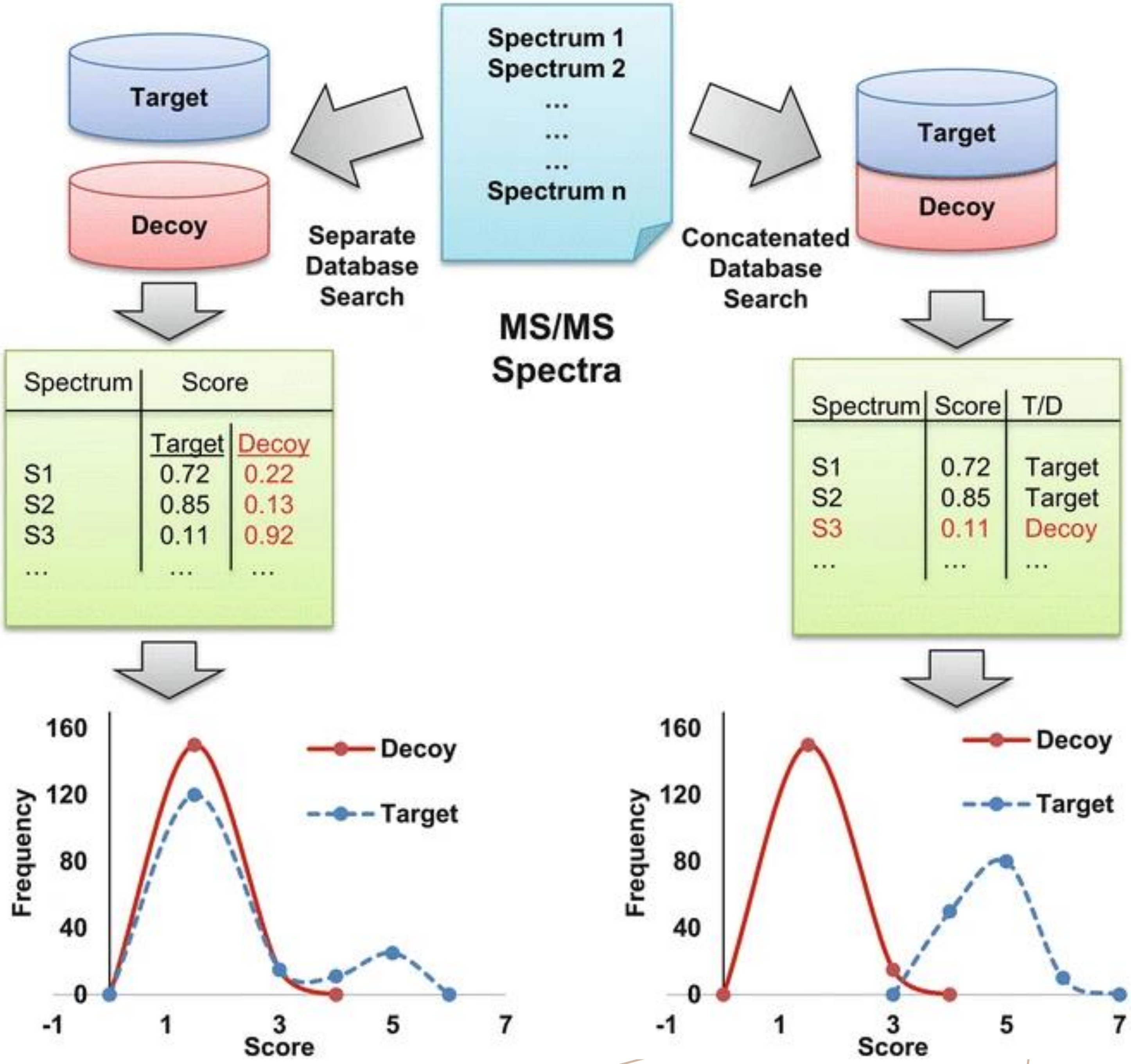
False Rate Discovery

- of statistical hypothesis are tested, usually independently, for significance [1]. Each hypothesis tested (a gene, a transcript, a peptide, etc.) yields a p-value, e-value, or a score that depicts the quantitative measure of that hypothesis being correct. In proteomics, shotgun proteomics data is searched using a database search algorithm that provides such confidence metrics after searching spectra against peptides in a given FASTA database.
- While such metrics can reflect the “goodness of fit” of an experimental spectrum to the assigned peptide and related chances of error in its identification, it cannot reflect the associated error in the whole dataset. For example, selecting peptide spectral matches (PSMs) or hits with p-value ≤ 0.05 means that each PSM has a 5 % or less chance of incorrectly being assigned as a significant match. This, however, does not mean that all PSMs passing this threshold will have a collective error of 5 %. Any PSM, even with a low estimated error of 5 %, could turn out to be wrong.

False Discovery Rate

- Adjusting for multiple comparisons can be achieved by applying Bonferroni correction which readjusts significance threshold ($\alpha = 0.5$) to control the false positives. For n spectra, the population-level significance threshold becomes 0.5/n, to adjust for the error rate of 5 % globally. This method is very stringent and false positives are extremely low when n is large. But this occurs at the cost of false negatives. To avoid false positives, this method excludes many true positives with good scores and p-values. False discovery rate (FDR) is a measure of the incorrect PSMs among all accepted PSMs

There are two database search strategies—separate or combined database search. In separate search target and decoy databases are searched separately and FDR is estimated using Kall’s method .Each spectrum has one target and one decoy best score. In combined/concatenated approach, one unified target-decoy database is searched in which both TD peptides compete with each other. Each spectrum has one best score, either from target or decoy but not both. This also changes the score distributions

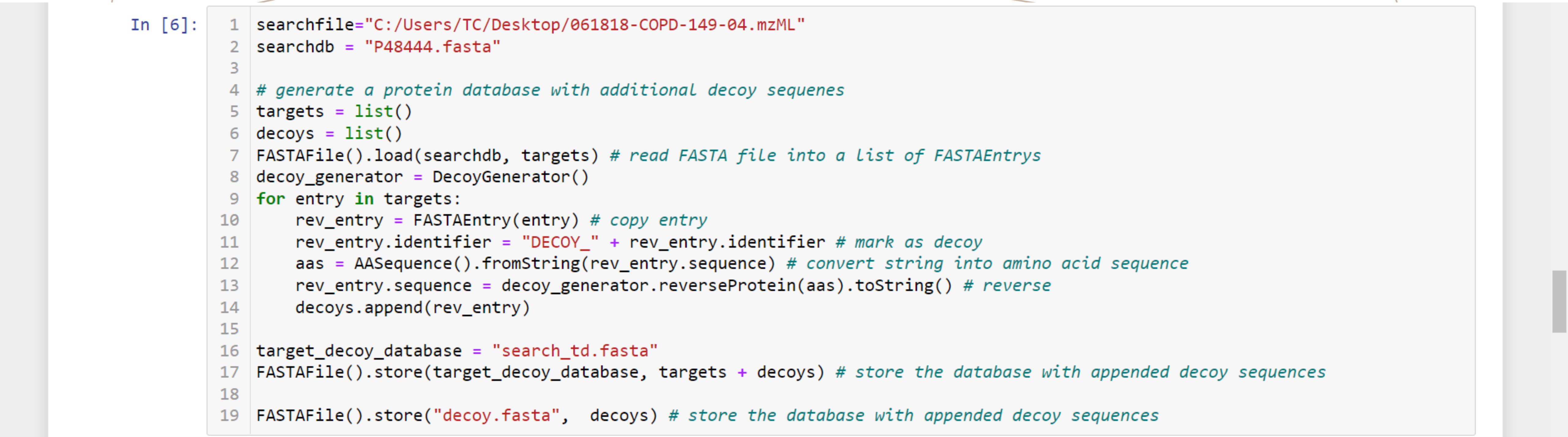


Spectrum	Score	
	Target	Decoy
S1	0.72	0.22
S2	0.85	0.13
S3	0.11	0.92
...	...	...

Spectrum	Score	T/D
S1	0.72	Target
S2	0.85	Target
S3	0.11	Decoy
...	...	...

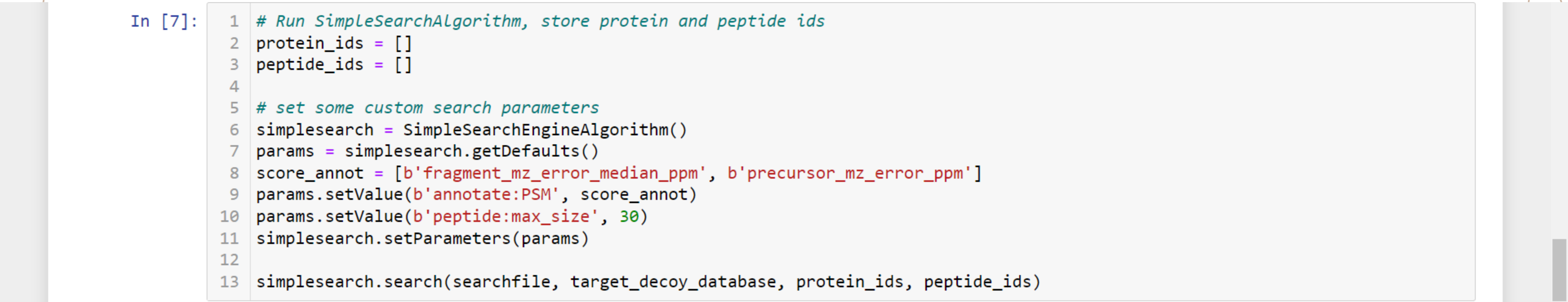
Q-Value

- While FDR is a global measure of population error rate, this communicates nothing about confidence of individual PSMs. A PSM-specific metric is needed which conveys the confidence measure of a particular PSM after FDR correction has been applied. Thus, q-value was introduced by Storey and Tibshirani [1], which is defined as the minimum FDR cutoff at which a particular PSM can be accepted.



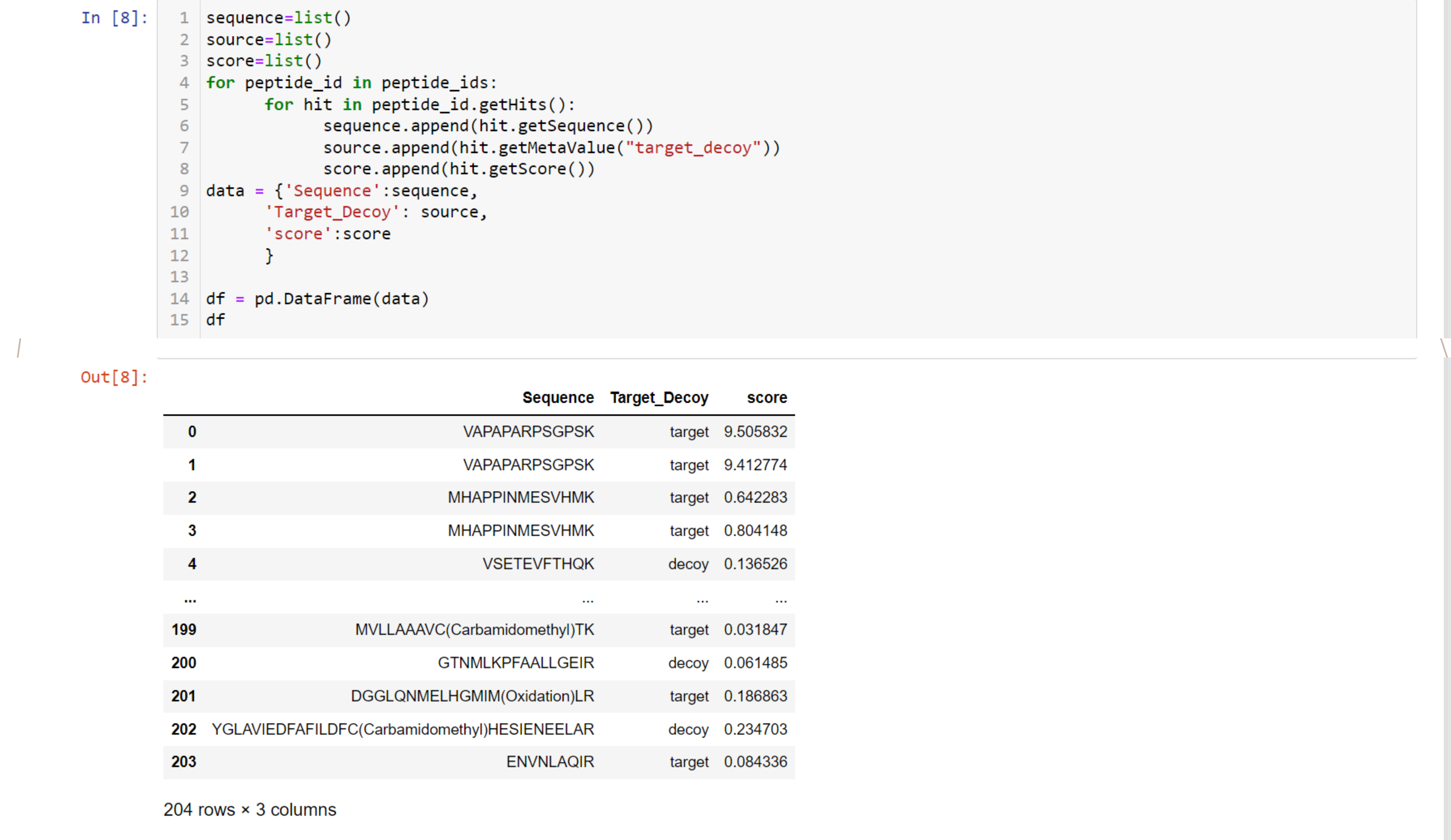
In [6]:

```
searchfile="C:/Users/TC/Desktop/061818-COPD-149-04.mzML"
searchdb = "P48444.fasta"

# generate a protein database with additional decoy sequenes
targets = list()
decoys = list()
FASTAFile().load(searchdb, targets) # read FASTA file into a list of FASTAEntrys
decoy_generator = DecoyGenerator()
for entry in targets:
    rev_entry = FASTAEntry(entry) # copy entry
    rev_entry.identifier = "DECOY_" + rev_entry.identifier # mark as decoy
    aas = AASequence().fromString(rev_entry.sequence) # convert string into amino acid sequence
    rev_entry.sequence = decoy_generator.reverseProtein(aas).toString() # reverse
    decoys.append(rev_entry)

target_decoy_database = "search_td.fasta"
FASTAFile().store(target_decoy_database, targets + decoys) # store the database with appended decoy sequences

FASTAFile().store("decoy.fasta",  decoys) # store the database with appended decoy sequences
```



In [7]:

```
# Run SimpleSearchAlgorithm, store protein and peptide ids
protein_ids = []
peptide_ids = []

# set some custom search parameters
simplesearch = SimpleSearchEngineAlgorithm()
params = simplesearch.getDefaults()
score_annot = [b'fragment_mz_error_median_ppm', b'precursor_mz_error_ppm']
params.setValue(b'annotate:PSM', score_annot)
params.setValue(b'peptide:max_size', 30)
simplesearch.setParameters(params)

simplesearch.search(searchfile, target_decoy_database, protein_ids, peptide_ids)
```



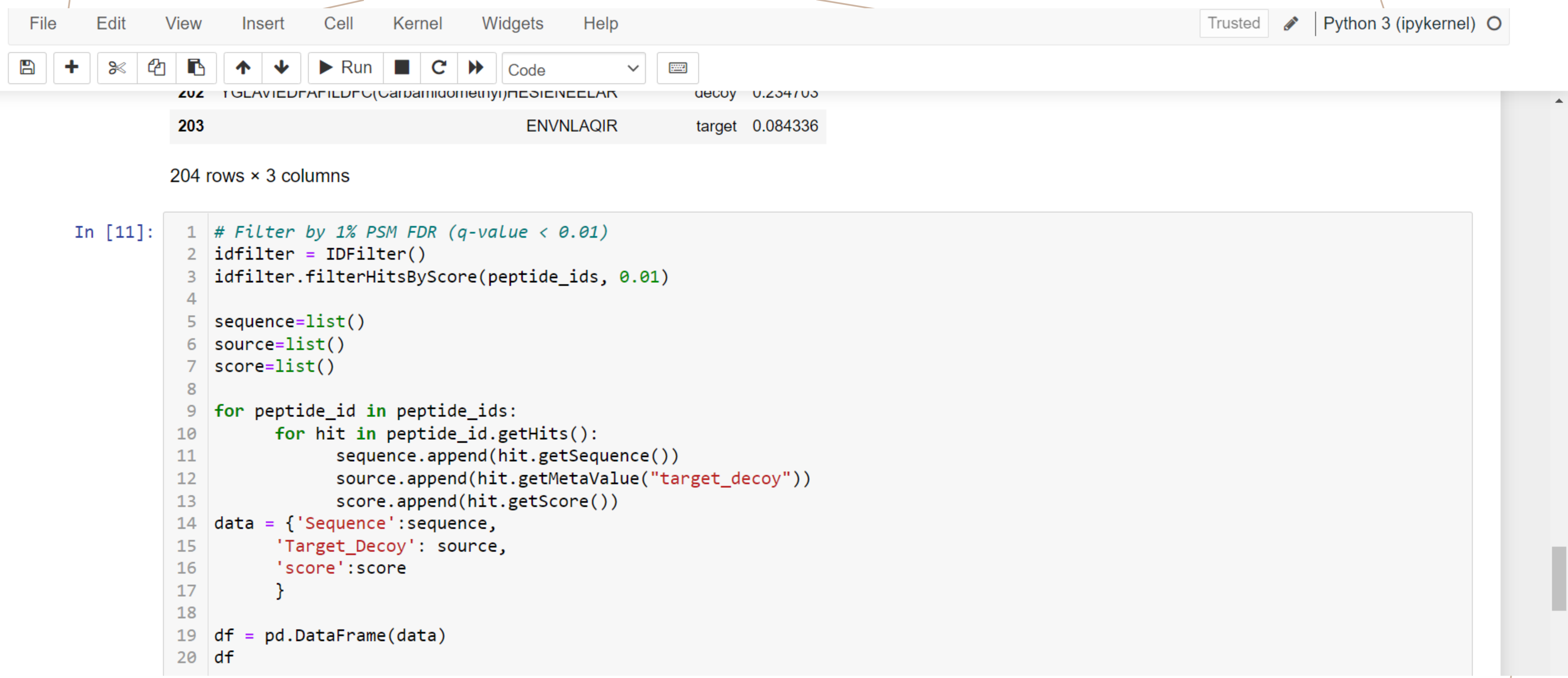
In [8]:

```python
sequence=list()
source=list()
score=list()
for peptide_id in peptide_ids:
      for hit in peptide_id.getHits():
            sequence.append(hit.getSequence())
            source.append(hit.getMetaValue("target_decoy"))
            score.append(hit.getScore())
data = {'Sequence':sequence,
      'Target_Decoy': source,
      'score':score
      }

df = pd.DataFrame(data)
df
```

Out[8]:

	Sequence	Target_Decoy	score
0	VAPAPARPSGPSK	target	9.505832
1	VAPAPARPSGPSK	target	9.412774
2	MHAPPINMESVHMK	target	0.642283
3	MHAPPINMESVHMK	target	0.804148
4	VSETEVFTHQK	decoy	0.136526
...	...	...	...
199	MVLLAAAVC(Carbamidomethyl)TK	target	0.031847
200	GTNMLKPFAALLGEIR	decoy	0.061485
201	DGGLQNMELHGMIM(Oxidation)LR	target	0.186863
202	YGLAVIEDFAFILDFC(Carbamidomethyl)HESIENEELAR	decoy	0.234703
203	ENVNLAQIR	target	0.084336

204 rows × 3 columns

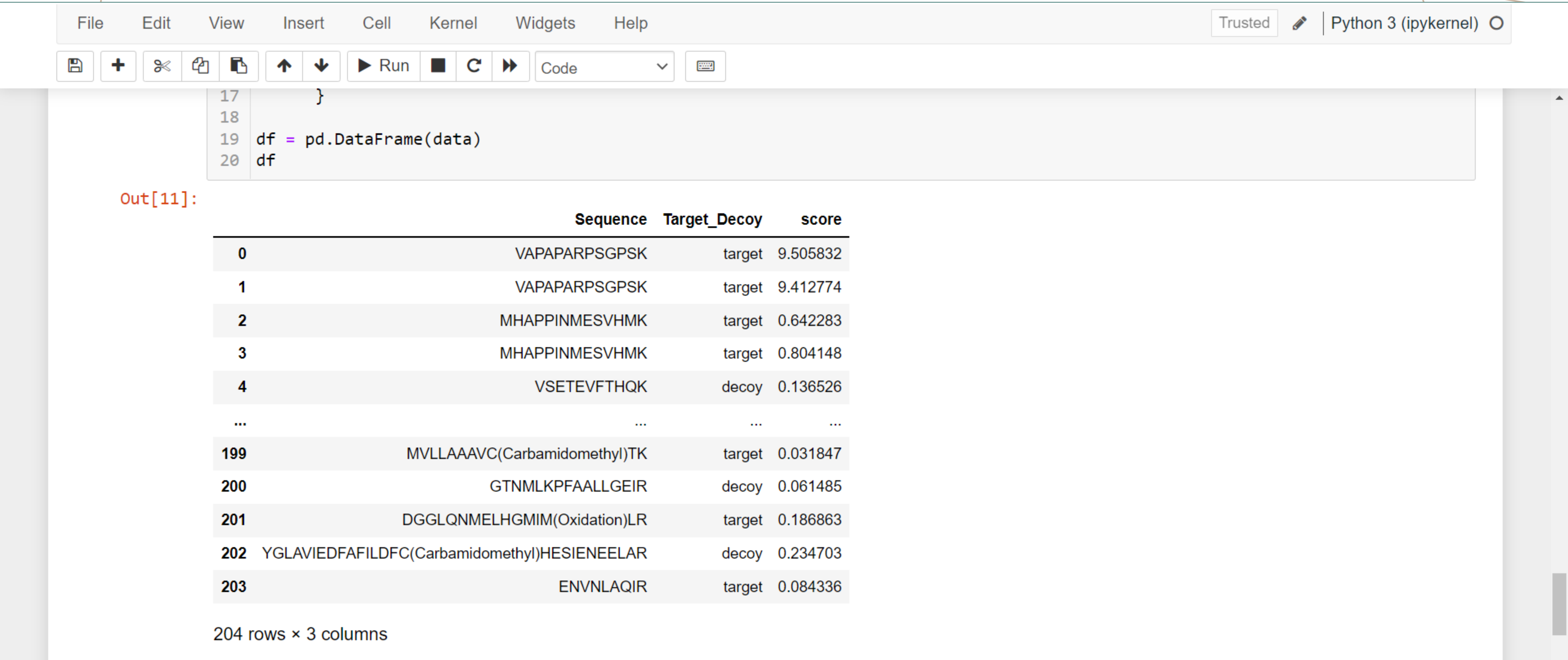


202	YGLAVIEDFAFILDFC(Carbamidomethyl)HESIENEELAR	decoy	0.234703
203	ENVNLAQIR	target	0.084336

204 rows × 3 columns

In [11]:

```
# Filter by 1% PSM FDR (q-value < 0.01)
idfilter = IDFilter()
idfilter.filterHitsByScore(peptide_ids, 0.01)

sequence=list()
source=list()
score=list()

for peptide_id in peptide_ids:
      for hit in peptide_id.getHits():
            sequence.append(hit.getSequence())
            source.append(hit.getMetaValue("target_decoy"))
            score.append(hit.getScore())
data = {'Sequence':sequence,
      'Target_Decoy': source,
      'score':score
      }

df = pd.DataFrame(data)
df
```

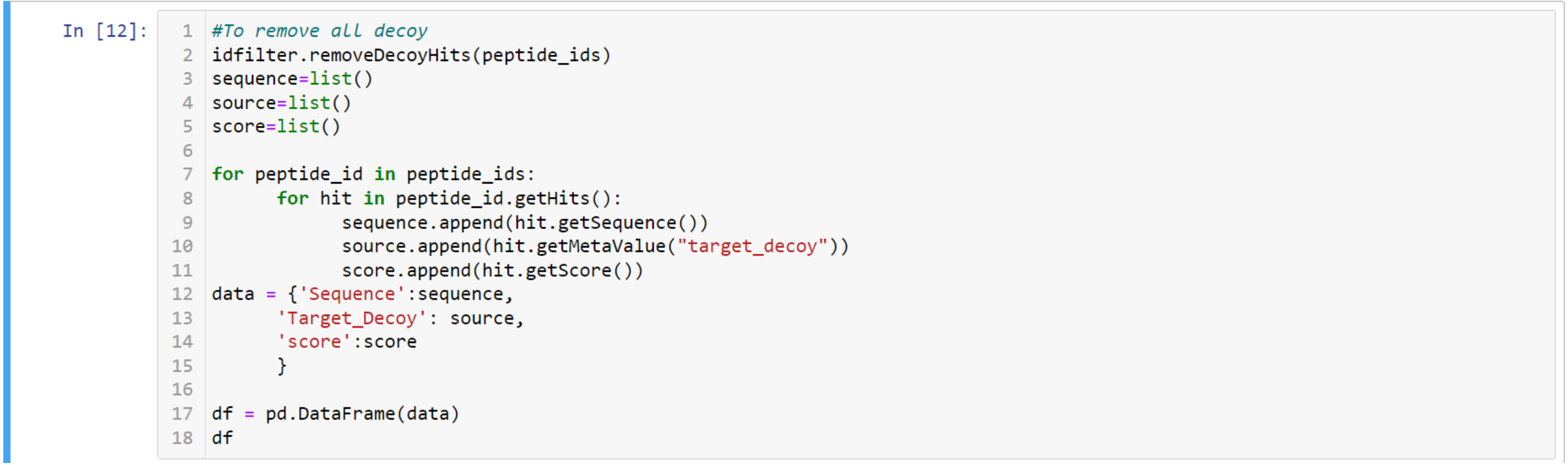


```python
        }

df = pd.DataFrame(data)
df
```

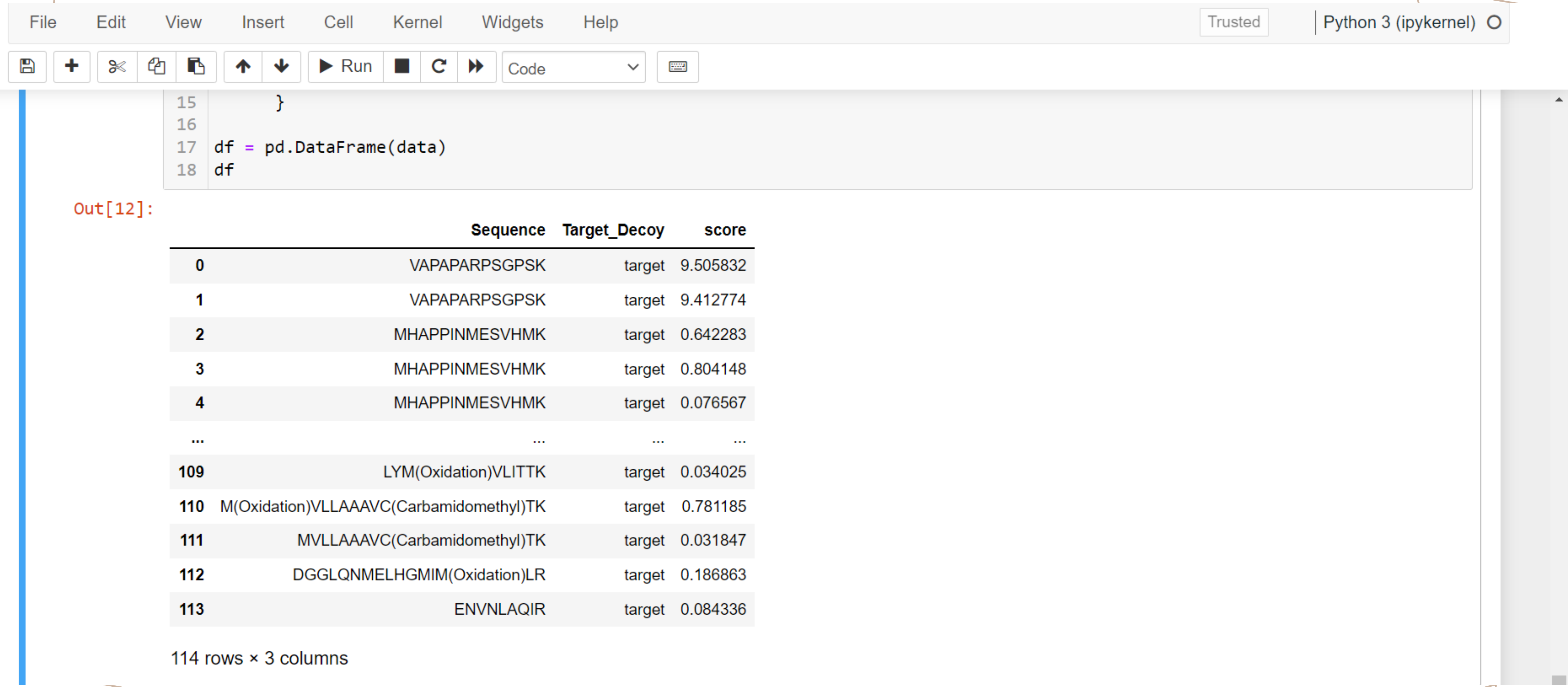
Out[11]:

	Sequence	Target_Decoy	score
0	VAPAPARPSGPSK	target	9.505832
1	VAPAPARPSGPSK	target	9.412774
2	MHAPPINMESVHMK	target	0.642283
3	MHAPPINMESVHMK	target	0.804148
4	VSETEVFTHQK	decoy	0.136526
...	...	...	...
199	MVLLAAAVC(Carbamidomethyl)TK	target	0.031847
200	GTNMLKPFAALLGEIR	decoy	0.061485
201	DGGLQNMELHGMIM(Oxidation)LR	target	0.186863
202	YGLAVIEDFAFILDFC(Carbamidomethyl)HESIENEELAR	decoy	0.234703
203	ENVNLAQIR	target	0.084336

204 rows × 3 columns



```
#To remove all decoy
idfilter.removeDecoyHits(peptide_ids)
sequence=list()
source=list()
score=list()

for peptide_id in peptide_ids:
      for hit in peptide_id.getHits():
            sequence.append(hit.getSequence())
            source.append(hit.getMetaValue("target_decoy"))
            score.append(hit.getScore())
data = {'Sequence':sequence,
      'Target_Decoy': source,
      'score':score
      }

df = pd.DataFrame(data)
df
```



```
        }

df = pd.DataFrame(data)
df
```

Out[12]:

	Sequence	Target_Decoy	score
0	VAPAPARPSGPSK	target	9.505832
1	VAPAPARPSGPSK	target	9.412774
2	MHAPPINMESVHMK	target	0.642283
3	MHAPPINMESVHMK	target	0.804148
4	MHAPPINMESVHMK	target	0.076567
...	...	...	...
109	LYM(Oxidation)VLITTK	target	0.034025
110	M(Oxidation)VLLAAAVC(Carbamidomethyl)TK	target	0.781185
111	MVLLAAAVC(Carbamidomethyl)TK	target	0.031847
112	DGGLQNMELHGMIM(Oxidation)LR	target	0.186863
113	ENVNLAQIR	target	0.084336

114 rows × 3 columns

THANK YOU