QUALITY CONTROL OF PROTEOMICS DATA FROM LIQUID CHROMATOGRAPHY - MASS SPECTROMETER

Submitted by

Kishore Tumarada

Table of Contents

1. Abstract

1.1. Quality control of Liquid Chromatography - Mass Spectrometer system is essential for appropriate statistical analysis of output data. To monitor the system performance, a set of standard peptides and fragment ions have been extracted based on conditions of maximum intensity, uniformly sampled across whole retention time, and lowest retention time CV. In particular, a total of 21 peptides have been extracted along with 5 fragment ions with non-neutral losses. These standards are used as the basis for analyzing successive LC-MS outputs for different Raw files over a period of time. Finally, visualizations of various parameters - m/z, intensity, resolution, reduced mobility, ccs, retention time, retention length (FWHM), ion mobility index length (FWHM), fragment ion intensities - have been created to see the pattern of peptides and monitor the performance of the LC-MS system.

2. Introduction (background of the internship project, and the motivation of the study)

2.1. Proteomics refers to the large-scale experimental analysis of proteins and proteomes. It has enabled identification of ever-increasing number of proteins. It covers exploration of proteomes from overall level of protein composition, structure, and activity.

2.2. It is a crucial domain in modern biological and biomedical research. Presently, Liquid chromatography (LC) followed by Mass spectrometry (MS) is the preferred method to identify and quantify complex protein samples. The importance of these techniques is demonstrated by their use in large-scale research initiatives, such as the ongoing Human Proteome Project (HPP).

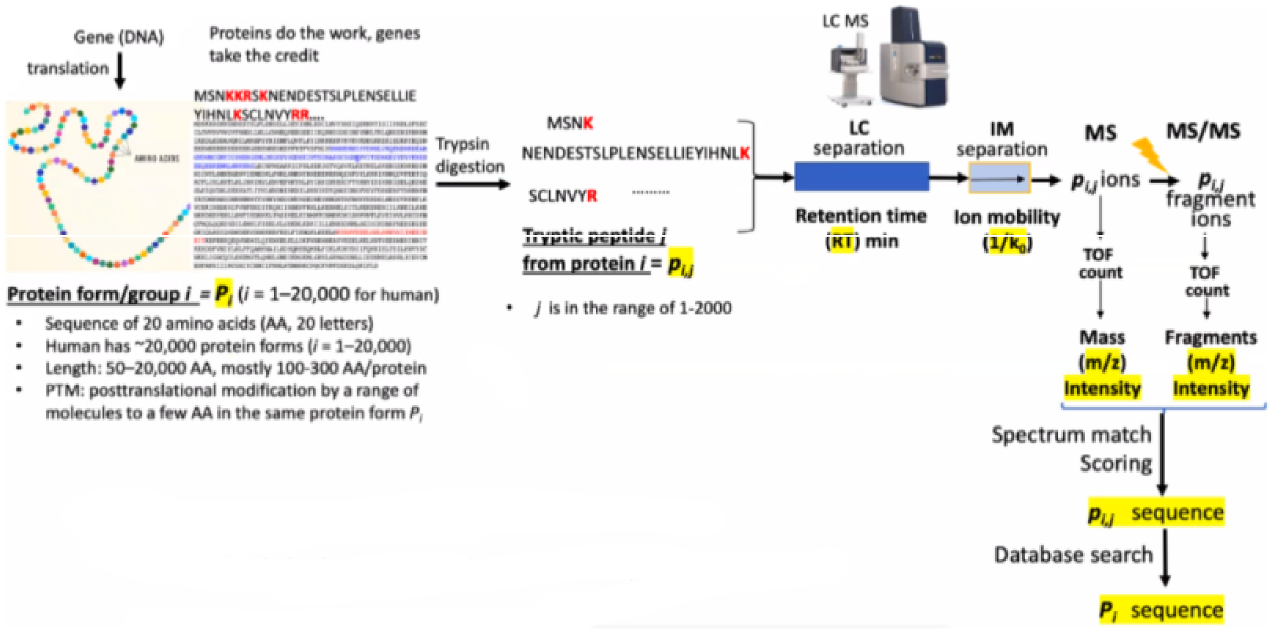

Gene (DNA)
translation
Proteins do the work, genes take the credit
MSNKKRSKNENDESTSLPLENSELLIE YIHNLKSCLNVYRR....
Trypsin digestion
MSNK
NENDESTSLPLENSELLIEYIHNLK
SCLNVYR
Tryptic peptide j from protein i = $p_{i,j}$
Protein form/group i = P_i (i = 1–20,000 for human)
Sequence of 20 amino acids (AA, 20 letters)
Human has ~20,000 protein forms (i = 1–20,000)
Length: 50–20,000 AA, mostly 100-300 AA/protein
PTM: posttranslational modification by a range of molecules to a few AA in the same protein form P_i
j is in the range of 1-2000
LC MS
LC separation
Retention time (RT) min
IM separation
Ion mobility (1/k0)
MS
$p_{i,j}$ ions
TOF count
Mass (m/z) Intensity
MS/MS
$p_{i,j}$ fragment ions
TOF count
Fragments (m/z) Intensity
Spectrum match Scoring
$p_{i,j}$ sequence
Database search
P_i sequence

Figure 1 Workflow of LC-MS system

2.3. Figure 1 shows the workflow of LC-MS system. Firstly, proteins in a biological sample are digested into peptides by tryptic digestion. In the next stage, peptides are separated based on retention time and ion mobility. In the last stage, based on spectrum match scoring and database search, proteins are identified.

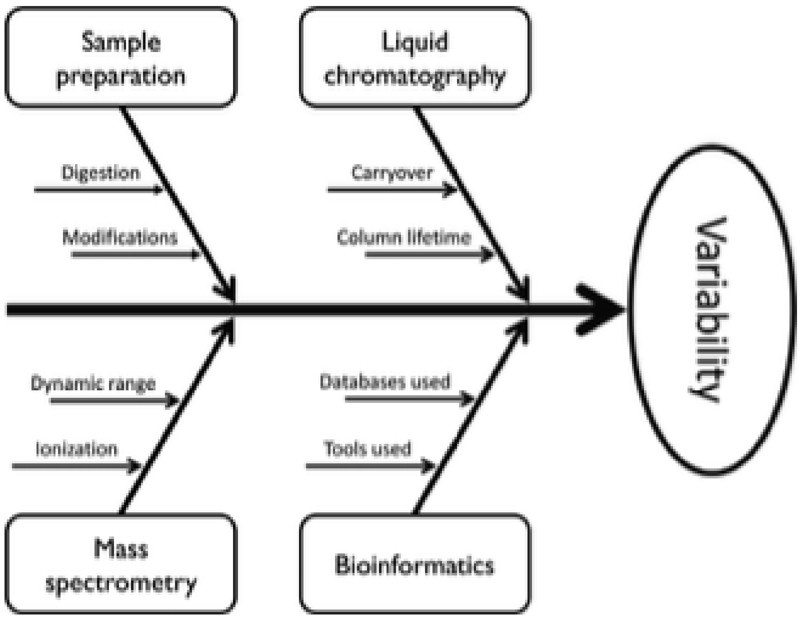


Figure 2. A diagram highlighting some of the major sources of variability in each of the stages of an LC-MS experiment.

2.4. However, the results of an experiment can still be subject to significant variability. This variability can originate from multiple sources as shown in Figure 2. Hence quality control is key for proteomic workflow.

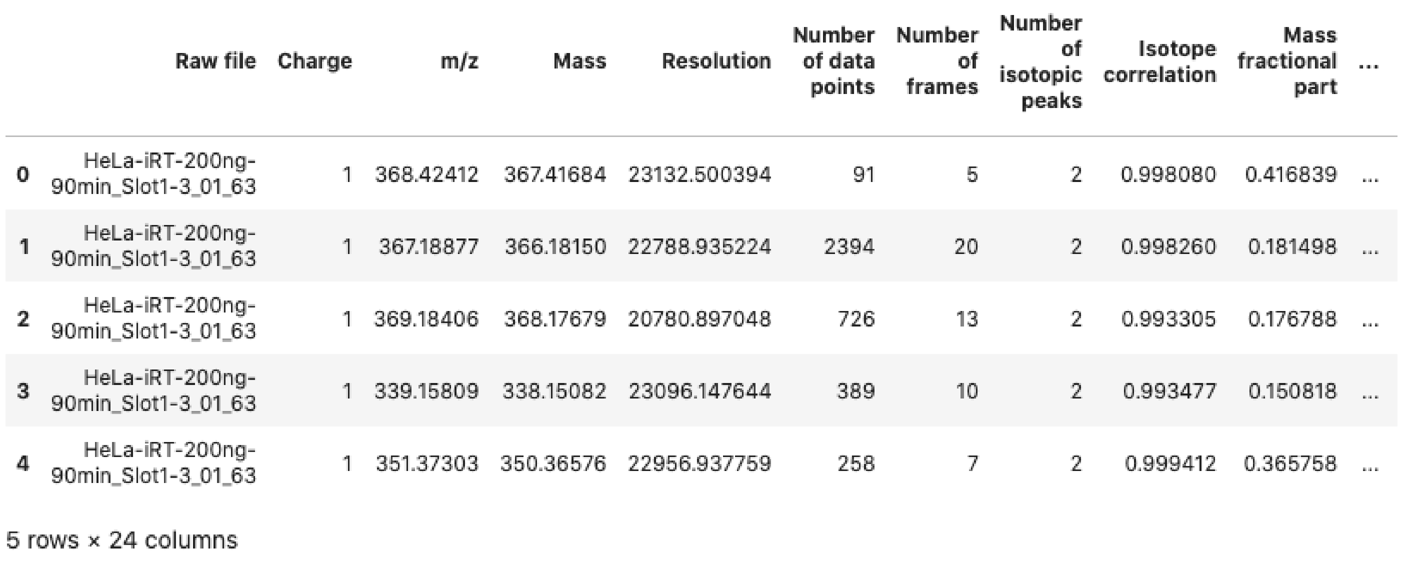
2.5. Goals of this project are three-fold. Firstly, extract a series of peptides to serve as standards. Secondly, extract information only for this group of peptides from QC (Quality control) runs obtained periodically. Lastly, visualize the parameters of each of these standard peptides for subsequent QC runs overtime to monitor the performance of this system.

3. Data and methods

3.1. LC-MS output Raw data is high-resolution data without data labeling. MaxQuant, a quantitative proteomics software package, is used for analyzing and data labeling the raw data. The output is used as input for this project.

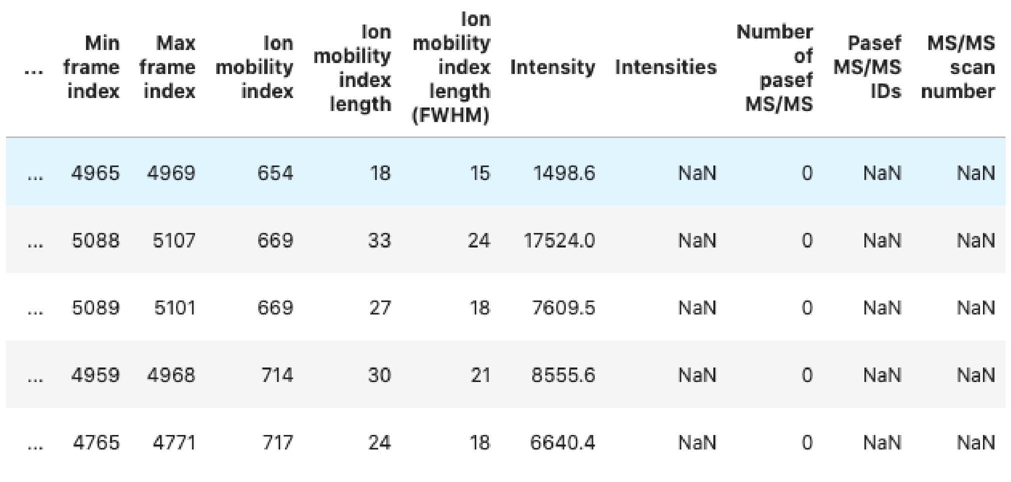
3.2. The following datasets from MaxQuant output are considered for creating benchmark peptides:

3.2.1. **Allpeptides file** - all detected LC-MS features (feature characteristics; can be e.g. used to plot all features against the ones targeted for MS/MS and identified ones)



	Raw file	Charge	m/z	Mass	Resolution	Number of data points	Number of frames	Number of isotopic peaks	Isotope correlation	Mass fractional part	...
0	HeLa-iRT-200ng-90min_Slot1-3_01_63	1	368.42412	367.41684	23132.500394	91	5	2	0.998080	0.416839	...
1	HeLa-iRT-200ng-90min_Slot1-3_01_63	1	367.18877	366.18150	22788.935224	2394	20	2	0.998260	0.181498	...
2	HeLa-iRT-200ng-90min_Slot1-3_01_63	1	369.18406	368.17679	20780.897048	726	13	2	0.993305	0.176788	...
3	HeLa-iRT-200ng-90min_Slot1-3_01_63	1	339.15809	338.15082	23096.147644	389	10	2	0.993477	0.150818	...
4	HeLa-iRT-200ng-90min_Slot1-3_01_63	1	351.37303	350.36576	22956.937759	258	7	2	0.999412	0.365758	...

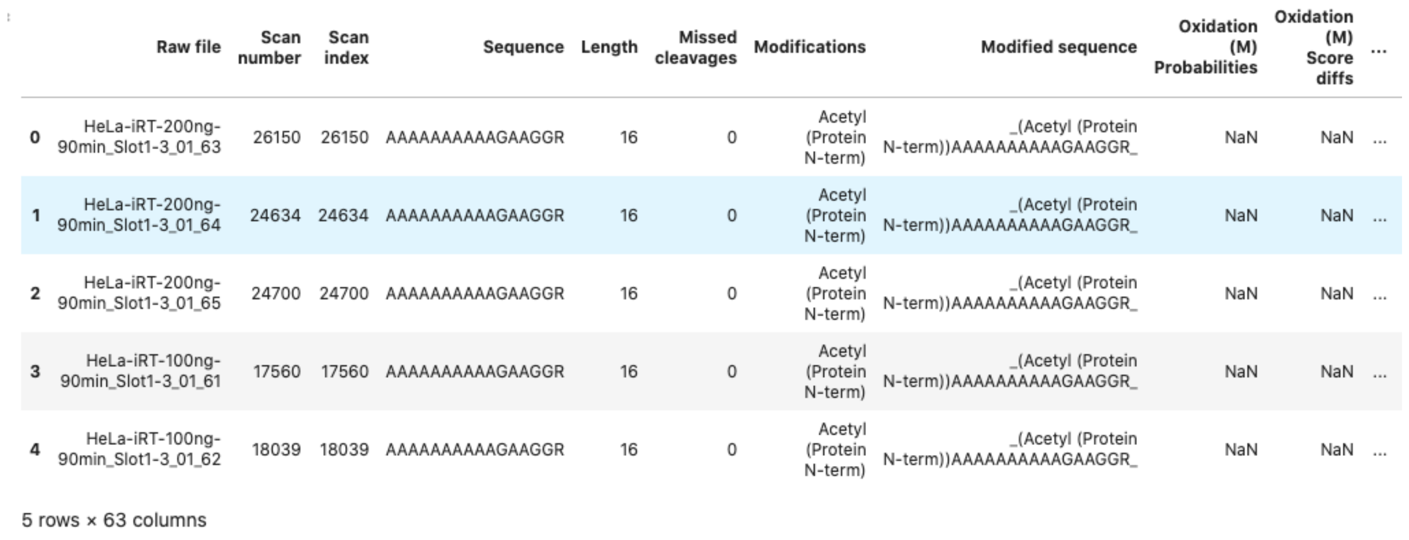
5 rows × 24 columns



...	Min frame index	Max frame index	Ion mobility index	Ion mobility index length	Ion mobility index length (FWHM)	Intensity	Intensities	Number of pasef MS/MS	Pasef MS/MS IDs	MS/MS scan number
...	4965	4969	654	18	15	1498.6	NaN	0	NaN	NaN
...	5088	5107	669	33	24	17524.0	NaN	0	NaN	NaN
...	5089	5101	669	27	18	7609.5	NaN	0	NaN	NaN
...	4959	4968	714	30	21	8555.6	NaN	0	NaN	NaN
...	4765	4771	717	24	18	6640.4	NaN	0	NaN	NaN

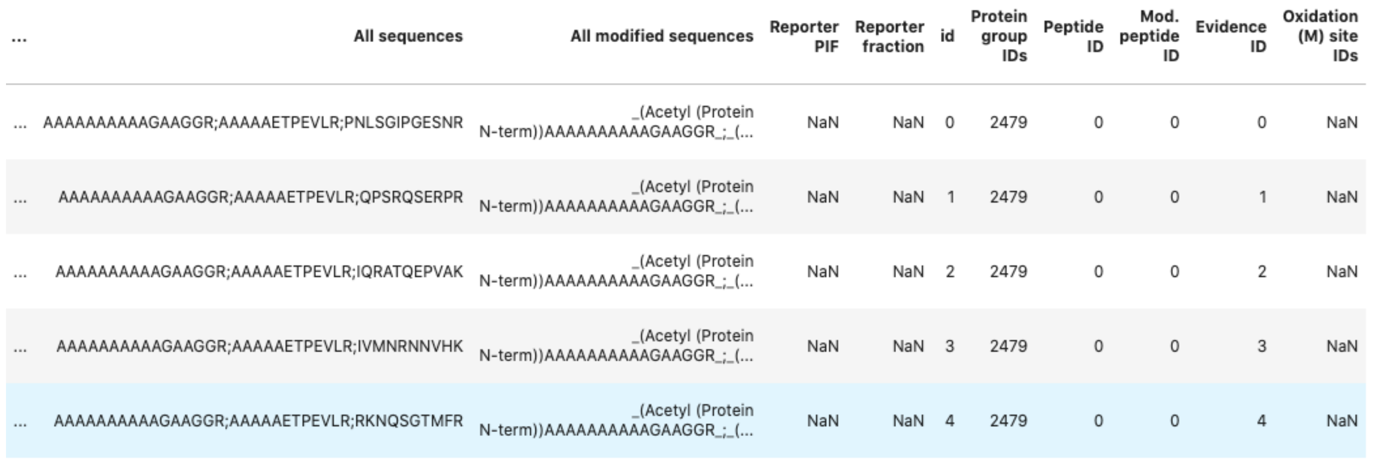
Table 1. Top 10 rows of Allpeptides dataset

3.2.2. **Msms file:** identified MS/MS events



	Raw file	Scan number	Scan index	Sequence	Length	Missed cleavages	Modifications	Modified sequence	Oxidation (M) Probabilities	Oxidation (M) Score diffs	...
0	HeLa-iRT-200ng-90min_Slot1-3_01_63	26150	26150	AAAAAAAAAAGAAGGR	16	0	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	...
1	HeLa-iRT-200ng-90min_Slot1-3_01_64	24634	24634	AAAAAAAAAAGAAGGR	16	0	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	...
2	HeLa-iRT-200ng-90min_Slot1-3_01_65	24700	24700	AAAAAAAAAAGAAGGR	16	0	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	...
3	HeLa-iRT-100ng-90min_Slot1-3_01_61	17560	17560	AAAAAAAAAAGAAGGR	16	0	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	...
4	HeLa-iRT-100ng-90min_Slot1-3_01_62	18039	18039	AAAAAAAAAAGAAGGR	16	0	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	...

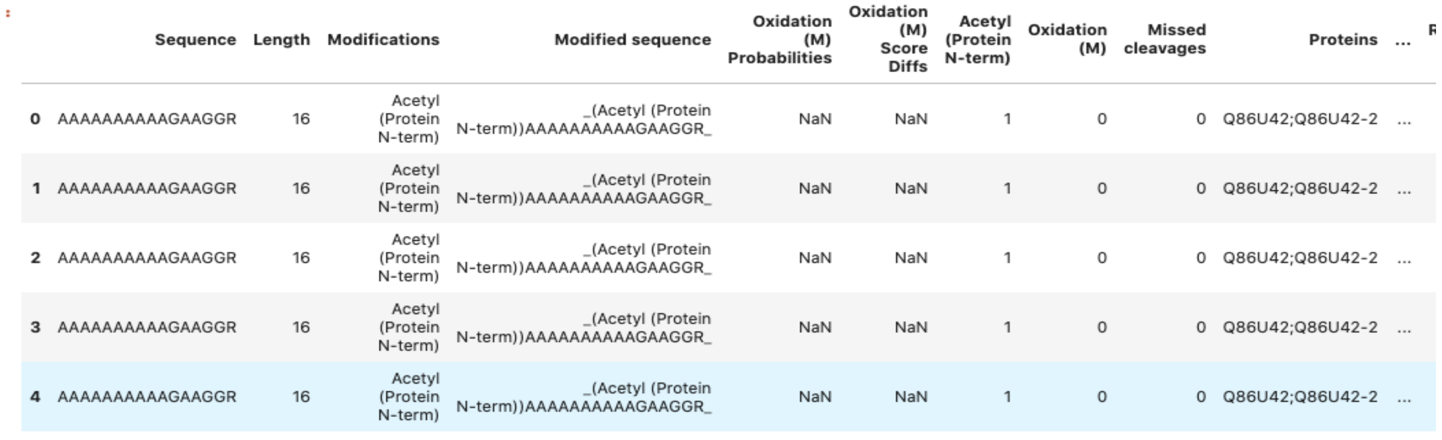
5 rows × 63 columns



...	All sequences	All modified sequences	Reporter PIF	Reporter fraction	id	Protein group IDs	Peptide ID	Mod. peptide ID	Evidence ID	Oxidation (M) site IDs
...	AAAAAAAAAAGAAGGR;AAAAAETPEVLR;PNLSGIPGESNR	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_;_(...	NaN	NaN	0	2479	0	0	0	NaN
...	AAAAAAAAAAGAAGGR;AAAAAETPEVLR;QPSRQSERPR	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_;_(...	NaN	NaN	1	2479	0	0	1	NaN
...	AAAAAAAAAAGAAGGR;AAAAAETPEVLR;IQRATQEPVAK	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_;_(...	NaN	NaN	2	2479	0	0	2	NaN
...	AAAAAAAAAAGAAGGR;AAAAAETPEVLR;IVMNRNNVHK	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_;_(...	NaN	NaN	3	2479	0	0	3	NaN
...	AAAAAAAAAAGAAGGR;AAAAAETPEVLR;RKNQSGTMFR	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_;_(...	NaN	NaN	4	2479	0	0	4	NaN

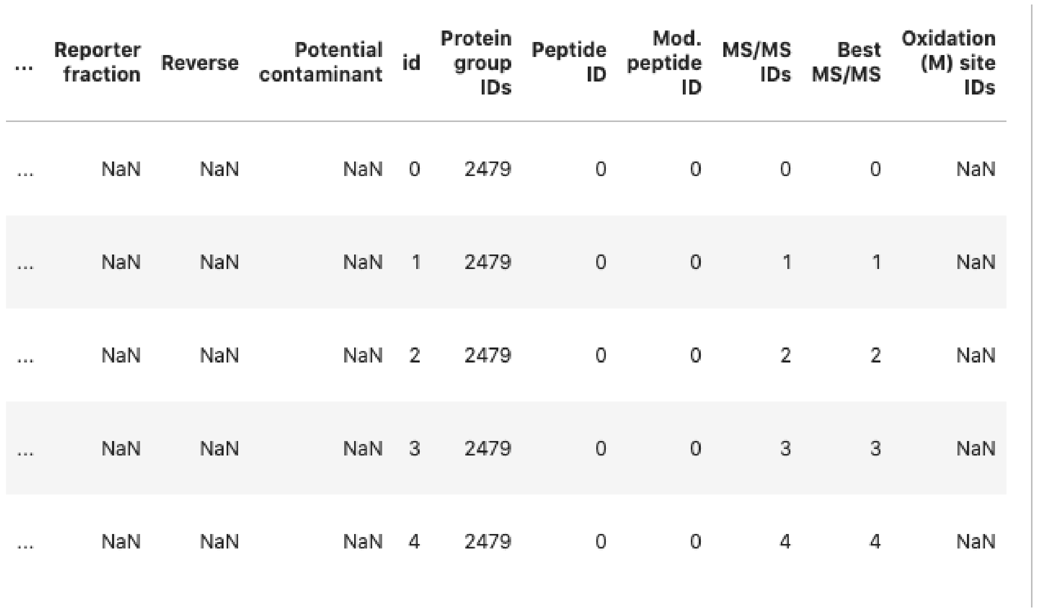
Table 2. Top 10 rows of msms dataset

3.2.3. **Evidence file**: all identified LC-MS-features, identified by MS/MS or by matching between runs, MS/MS events per identified feature, scores, mass deviation, feature information, useful table also for troubleshooting.



	Sequence	Length	Modifications	Modified sequence	Oxidation (M) Probabilities	Oxidation (M) Score Diffs	Acetyl (Protein N-term)	Oxidation (M)	Missed cleavages	Proteins	...
0	AAAAAAAAAAGAAGGR	16	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	1	0	0	Q86U42;Q86U42-2	...
1	AAAAAAAAAAGAAGGR	16	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	1	0	0	Q86U42;Q86U42-2	...
2	AAAAAAAAAAGAAGGR	16	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	1	0	0	Q86U42;Q86U42-2	...
3	AAAAAAAAAAGAAGGR	16	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	1	0	0	Q86U42;Q86U42-2	...
4	AAAAAAAAAAGAAGGR	16	Acetyl (Protein N-term)	_(Acetyl (Protein N-term))AAAAAAAAAAGAAGGR_	NaN	NaN	1	0	0	Q86U42;Q86U42-2	...

5 rows × 71 columns



...	Reporter fraction	Reverse	Potential contaminant	id	Protein group IDs	Peptide ID	Mod. peptide ID	MS/MS IDs	Best MS/MS	Oxidation (M) site IDs
...	NaN	NaN	NaN	0	2479	0	0	0	0	NaN
...	NaN	NaN	NaN	1	2479	0	0	1	1	NaN
...	NaN	NaN	NaN	2	2479	0	0	2	2	NaN
...	NaN	NaN	NaN	3	2479	0	0	3	3	NaN
...	NaN	NaN	NaN	4	2479	0	0	4	4	NaN

Table 3 Top 10 rows of Evidence dataset

4. Results and discussion

4.1. **Part 1 – Extraction of standard peptides**

4.1.1. **General principles for extracting Peptide sequences:**

To pick these standard peptides, we need to generate a pool of suitable ones. Following are the general principles adopted to extract data:

4.1.1.1. Limit peptides with a charge of 2. Column "charge" in both msms.txt and evidence.txt.

4.1.1.2. Standard peptides should be uniformly distributed along the entire gradient, i.e., across total retention time taken by a solute to pass through the liquid chromatography column. For example, if the gradient is 120 minutes, we can choose peptide every 5 minutes, which corresponds to at least 24 peptides. The following figure shows the histogram for retention time across the entire data

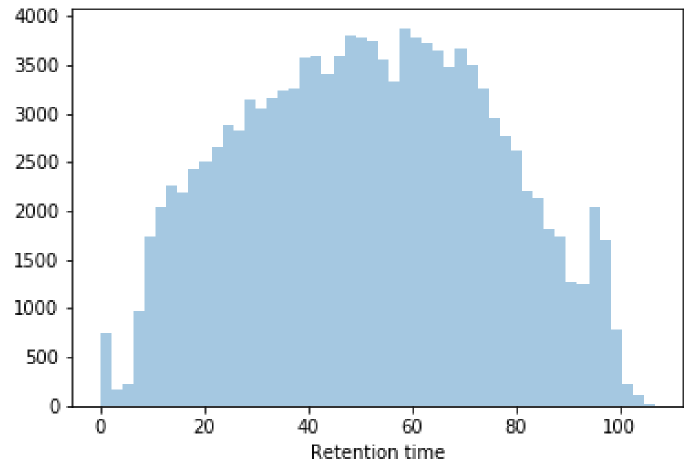
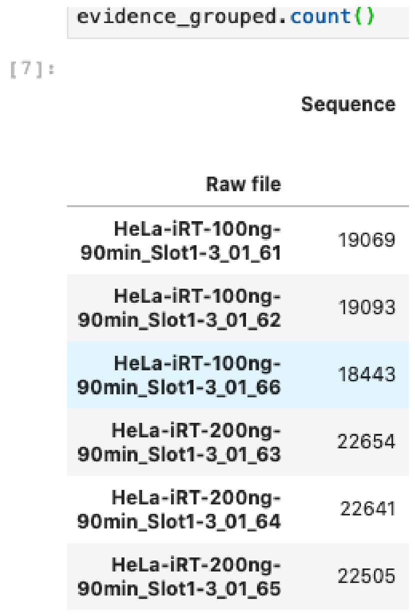


Figure 3 Histogram of Retention time of all Raw files

4.1.1.3. As the same peptide can be found multiple times in multiple files, the retention time has a bit of variation. For this, we have evaluated the Coefficient of Variations (CV) and decide a proper CV value to filter the peptides. (The coefficient of variation (CV), which is calculated by dividing the standard deviation of peptide profiles by the mean, is reported as percentages.)

4.1.1.4. Peptides need to be present in every sample or Raw file. For instance, in the present data, MQ output is generated from 6 raw files. Table 4 shows the number of peptide sequences present in each raw file.



```
evidence_grouped.count()
```

[7]:

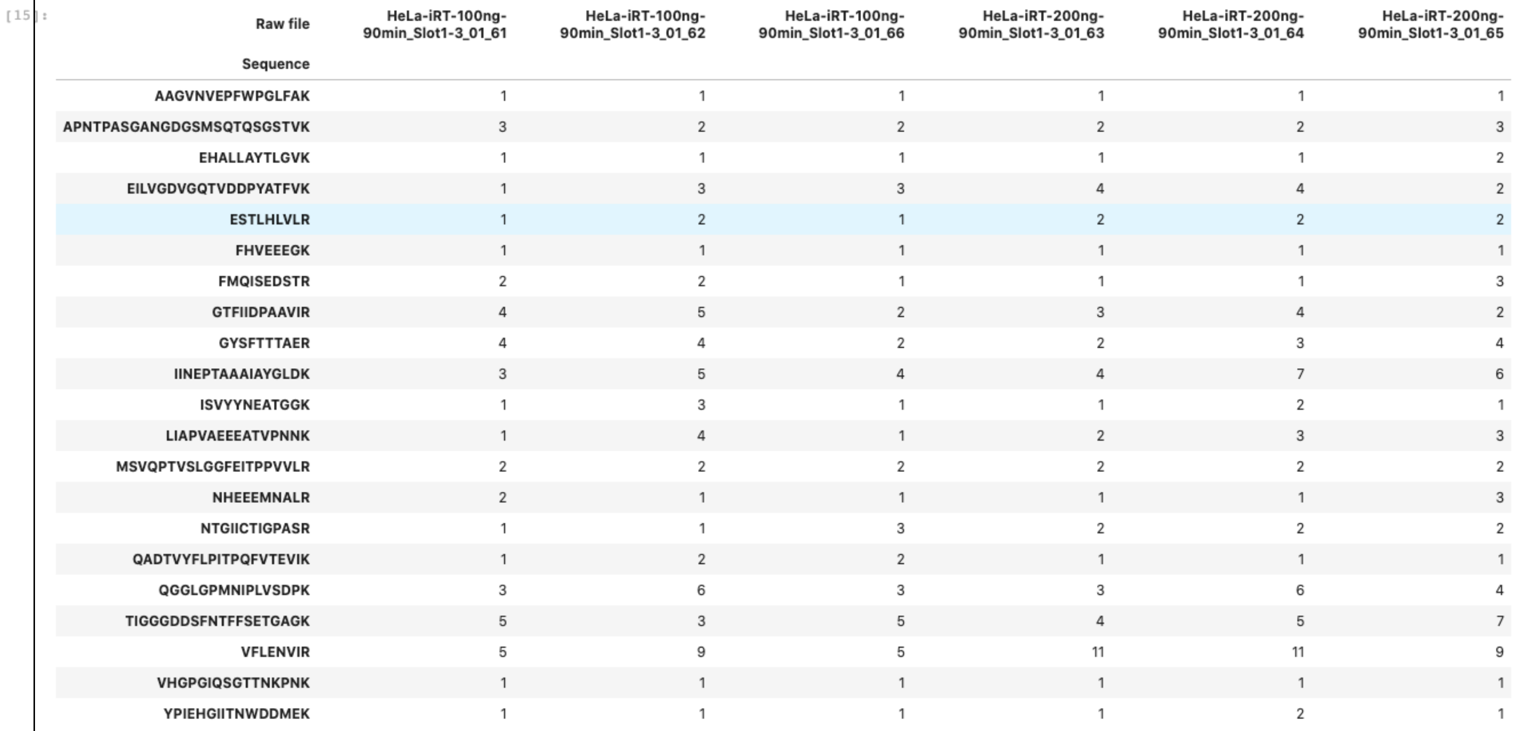
Raw file	Sequence
HeLa-iRT-100ng-90min_Slot1-3_01_61	19069
HeLa-iRT-100ng-90min_Slot1-3_01_62	19093
HeLa-iRT-100ng-90min_Slot1-3_01_66	18443
HeLa-iRT-200ng-90min_Slot1-3_01_63	22654
HeLa-iRT-200ng-90min_Slot1-3_01_64	22641
HeLa-iRT-200ng-90min_Slot1-3_01_65	22505

Table 4 number of peptide sequences across each raw file

4.1.1.5. Peptides need to have good signal intensity.

4.1.2. Based on these principles, out of all peptide sequences in every 5 minutes interval, peptides present in all raw files are filtered. From these peptides, those with the highest intensity and lowest Retention time CV is selected. As a result, a maximum of 24 peptides for 120 minutes retention time interval can be obtained.

4.1.3. However, in the selected MQ output, only 21 peptides met the conditions. Table 5 shows the distribution of these peptides across raw files:


[15]:

Raw file / Sequence	HeLa-iRT-100ng-90min_Slot1-3_01_61	HeLa-iRT-100ng-90min_Slot1-3_01_62	HeLa-iRT-100ng-90min_Slot1-3_01_66	HeLa-iRT-200ng-90min_Slot1-3_01_63	HeLa-iRT-200ng-90min_Slot1-3_01_64	HeLa-iRT-200ng-90min_Slot1-3_01_65
AAGVNVEPFWPGLFAK	1	1	1	1	1	1
APNTPASGANGDGSMSQTQSGSTVK	3	2	2	2	2	3
EHALLAYTLGVK	1	1	1	1	1	2
EILVGDVGQTVDDPYATFVK	1	3	3	4	4	2
ESTLHLVLR	1	2	1	2	2	2
FHVEEEGK	1	1	1	1	1	1
FMQISEDSTR	2	2	1	1	1	3
GTFIIDPAAVIR	4	5	2	3	4	2
GYSFTTTAER	4	4	2	2	3	4
IINEPTAAAIAYGLDK	3	5	4	4	7	6
ISVYYNEATGGK	1	3	1	1	2	1
LIAPVAEEEATVPNNK	1	4	1	2	3	3
MSVQPTVSLGGFEITPPVVLR	2	2	2	2	2	2
NHEEEMNALR	2	1	1	1	1	3
NTGIICTIGPASR	1	1	3	2	2	2
QADTVYFLPITPQFVTEVIK	1	2	2	1	1	1
QGGLGPMNIPLVSDPK	3	6	3	3	6	4
TIGGGDDSFNTFFSETGAGK	5	3	5	4	5	7
VFLENVIR	5	9	5	11	11	9
VHGPGIQSGTTNKPNK	1	1	1	1	1	1
YPIEHGIITNWDDMEK	1	1	1	1	2	1

Table 5. Distribution of peptides across all raw files

4.1.4. **<u>General principles for fragment ions extraction:</u>**

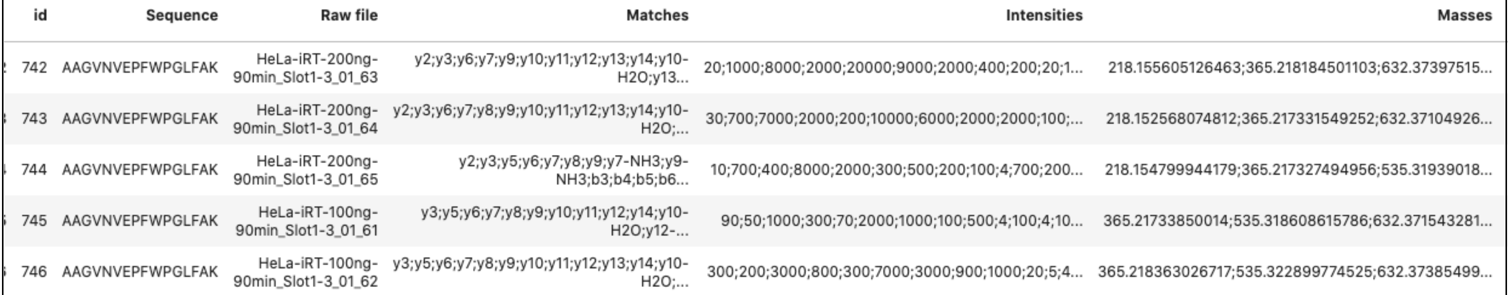
These standard Peptides need to have good fragmentation ion signals (also called MS2 signals).

4.1.4.1. This info on fragmentation ions can be found in columns “Matches”, “Intensities”, and “Masses” in file msms file. Column “Matches” contains the types of fragmented ions, and column “Intensities” and “Masses” contain the intensity and mass of each ion in the column “Matches”, respectively.

4.1.4.2. Only list the type, intensity, and mass of the fragment ions above a certain intensity value, for example 200.

4.1.4.3. Also eliminate those with neutral losses, such as y5-H2O, or y3-NH3, etc. only include those ions with types in the form of a letter (y or b) with a number (single or double digits).

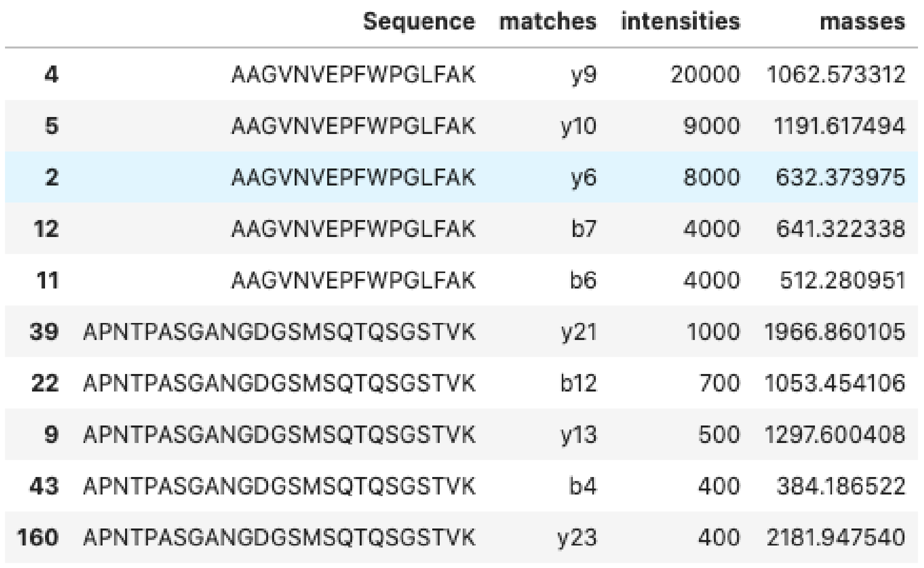
4.1.5. Table 6 shows the top 5 rows of msms data, after filtering for standard peptides:



id	Sequence	Raw file	Matches	Intensities	Masses
742	AAGVNVEPFWPGLFAK	HeLa-iRT-200ng-90min_Slot1-3_01_63	y2;y3;y6;y7;y9;y10;y11;y12;y13;y14;y10-H2O;y13...	20;1000;8000;2000;20000;9000;2000;400;200;20;1...	218.155605126463;365.218184501103;632.37397515...
743	AAGVNVEPFWPGLFAK	HeLa-iRT-200ng-90min_Slot1-3_01_64	y2;y3;y6;y7;y8;y9;y10;y11;y12;y13;y14;y10-H2O;...	30;700;7000;2000;200;10000;6000;2000;2000;100;...	218.152568074812;365.217331549252;632.37104926...
744	AAGVNVEPFWPGLFAK	HeLa-iRT-200ng-90min_Slot1-3_01_65	y2;y3;y5;y6;y7;y8;y9;y7-NH3;y9-NH3;b3;b4;b5;b6...	10;700;400;8000;2000;300;500;200;100;4;700;200...	218.154799944179;365.217327494956;535.31939018...
745	AAGVNVEPFWPGLFAK	HeLa-iRT-100ng-90min_Slot1-3_01_61	y3;y5;y6;y7;y8;y9;y10;y11;y12;y14;y10-H2O;y12-...	90;50;1000;300;70;2000;1000;100;500;4;100;4;10...	365.21733850014;535.318608615786;632.371543281...
746	AAGVNVEPFWPGLFAK	HeLa-iRT-100ng-90min_Slot1-3_01_62	y3;y5;y6;y7;y8;y9;y10;y11;y12;y13;y14;y10-H2O;...	300;200;3000;800;300;7000;3000;900;1000;20;5;4...	365.218363026717;535.322899774525;632.37385499...

Table 6. Top 5 rows of msms data after filtering standard peptides

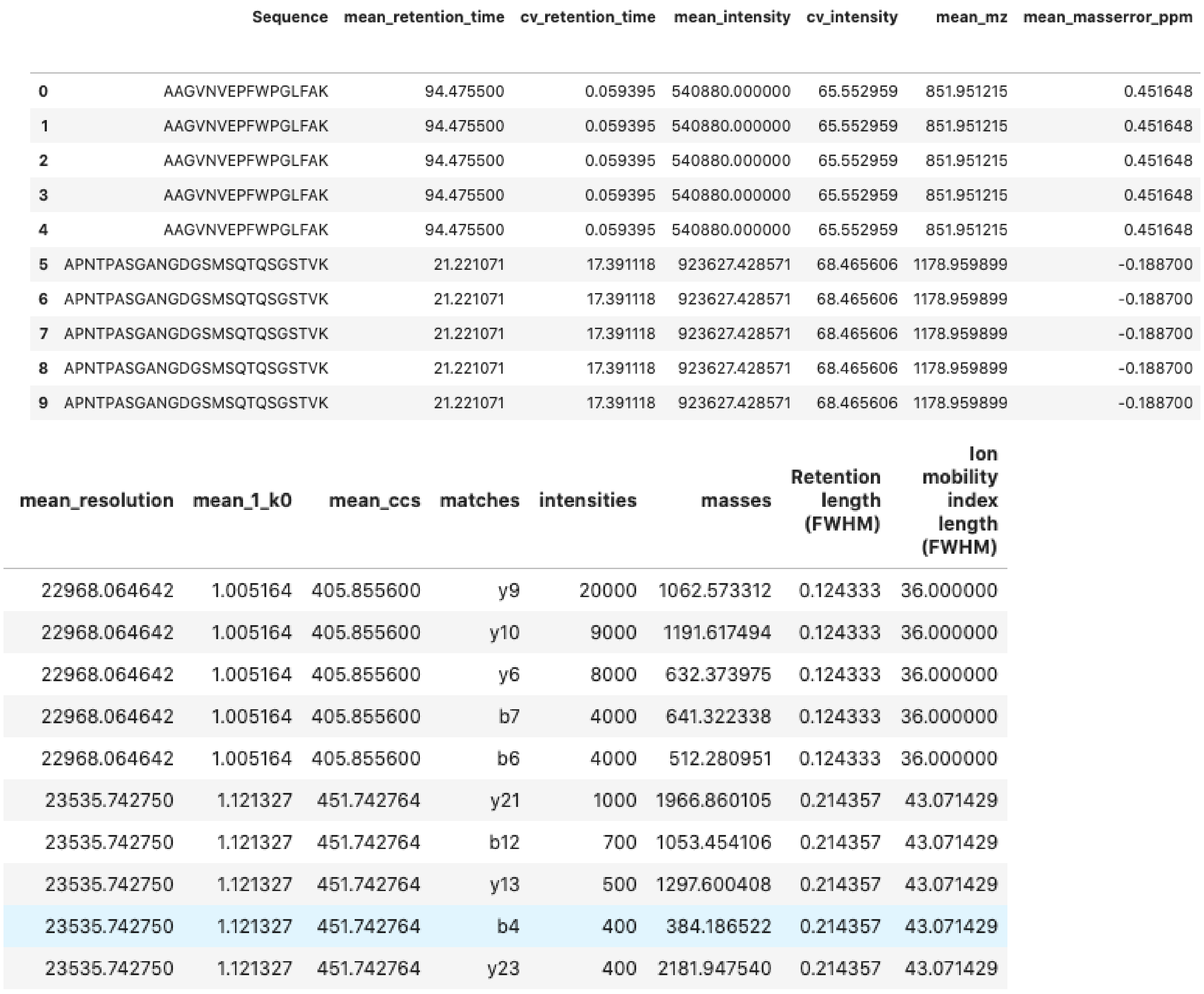
4.1.6. From the Filtered msms data, Top 5 fragment ions with highest intensities, without non-neutral losses, for each of 21 peptide sequences have been extracted. Table 7 shows the top 10 rows of the final fragment ions table.



	Sequence	matches	intensities	masses
4	AAGVNVEPFWPGLFAK	y9	20000	1062.573312
5	AAGVNVEPFWPGLFAK	y10	9000	1191.617494
2	AAGVNVEPFWPGLFAK	y6	8000	632.373975
12	AAGVNVEPFWPGLFAK	b7	4000	641.322338
11	AAGVNVEPFWPGLFAK	b6	4000	512.280951
39	APNTPASGANGDGSMSQTQSGSTVK	y21	1000	1966.860105
22	APNTPASGANGDGSMSQTQSGSTVK	b12	700	1053.454106
9	APNTPASGANGDGSMSQTQSGSTVK	y13	500	1297.600408
43	APNTPASGANGDGSMSQTQSGSTVK	b4	400	384.186522
160	APNTPASGANGDGSMSQTQSGSTVK	y23	400	2181.947540

Table 7. Top 10 rows of Processed msms data with 5 fragment ions for each standard peptide

4.1.7. Table 8 shows the final data for standard peptides, after extracting Retention length and Ion mobility index length (FWHM) from allpeptides file.



	Sequence	mean_retention_time	cv_retention_time	mean_intensity	cv_intensity	mean_mz	mean_masserror_ppm	mean_resolution	mean_1_k0	mean_ccs	matches	intensities	masses	Retention length (FWHM)	Ion mobility index length (FWHM)
0	AAGVNVEPFWPGLFAK	94.475500	0.059395	540880.000000	65.552959	851.951215	0.451648	22968.064642	1.005164	405.855600	y9	20000	1062.573312	0.124333	36.000000
1	AAGVNVEPFWPGLFAK	94.475500	0.059395	540880.000000	65.552959	851.951215	0.451648	22968.064642	1.005164	405.855600	y10	9000	1191.617494	0.124333	36.000000
2	AAGVNVEPFWPGLFAK	94.475500	0.059395	540880.000000	65.552959	851.951215	0.451648	22968.064642	1.005164	405.855600	y6	8000	632.373975	0.124333	36.000000
3	AAGVNVEPFWPGLFAK	94.475500	0.059395	540880.000000	65.552959	851.951215	0.451648	22968.064642	1.005164	405.855600	b7	4000	641.322338	0.124333	36.000000
4	AAGVNVEPFWPGLFAK	94.475500	0.059395	540880.000000	65.552959	851.951215	0.451648	22968.064642	1.005164	405.855600	b6	4000	512.280951	0.124333	36.000000
5	APNTPASGANGDGSMSQTQSGSTVK	21.221071	17.391118	923627.428571	68.465606	1178.959899	-0.188700	23535.742750	1.121327	451.742764	y21	1000	1966.860105	0.214357	43.071429
6	APNTPASGANGDGSMSQTQSGSTVK	21.221071	17.391118	923627.428571	68.465606	1178.959899	-0.188700	23535.742750	1.121327	451.742764	b12	700	1053.454106	0.214357	43.071429
7	APNTPASGANGDGSMSQTQSGSTVK	21.221071	17.391118	923627.428571	68.465606	1178.959899	-0.188700	23535.742750	1.121327	451.742764	y13	500	1297.600408	0.214357	43.071429
8	APNTPASGANGDGSMSQTQSGSTVK	21.221071	17.391118	923627.428571	68.465606	1178.959899	-0.188700	23535.742750	1.121327	451.742764	b4	400	384.186522	0.214357	43.071429
9	APNTPASGANGDGSMSQTQSGSTVK	21.221071	17.391118	923627.428571	68.465606	1178.959899	-0.188700	23535.742750	1.121327	451.742764	y23	400	2181.947540	0.214357	43.071429

Table 8. Top 10 rows of dataset with standard peptides and their fragment ions

4.1.8. Following are histograms of CV for Retention time and Intensities:

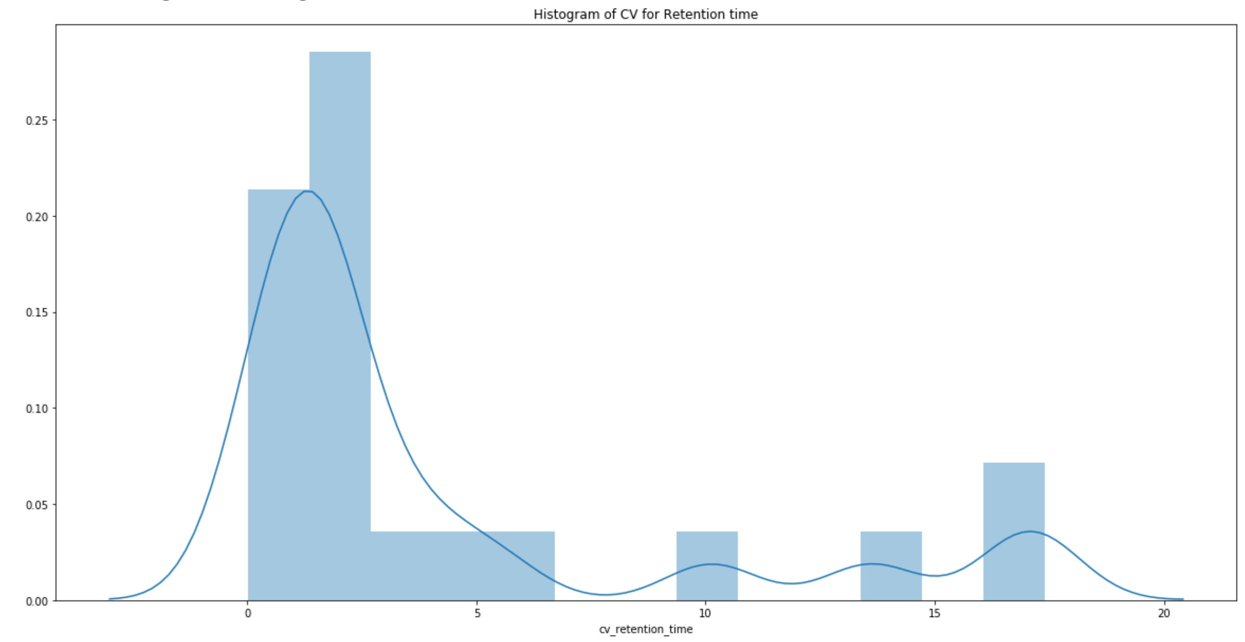


Figure 4 Histogram of CV for Retention time

Figure 4 shows that highest frequency of CV of Retention time (in %) is less than 3%.

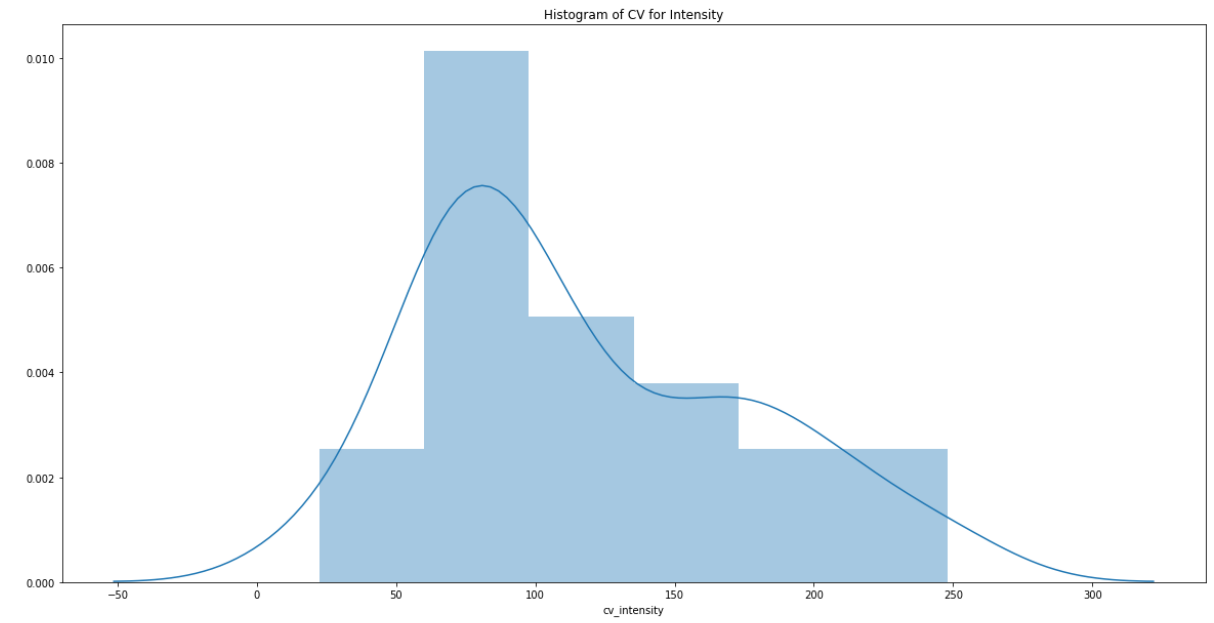
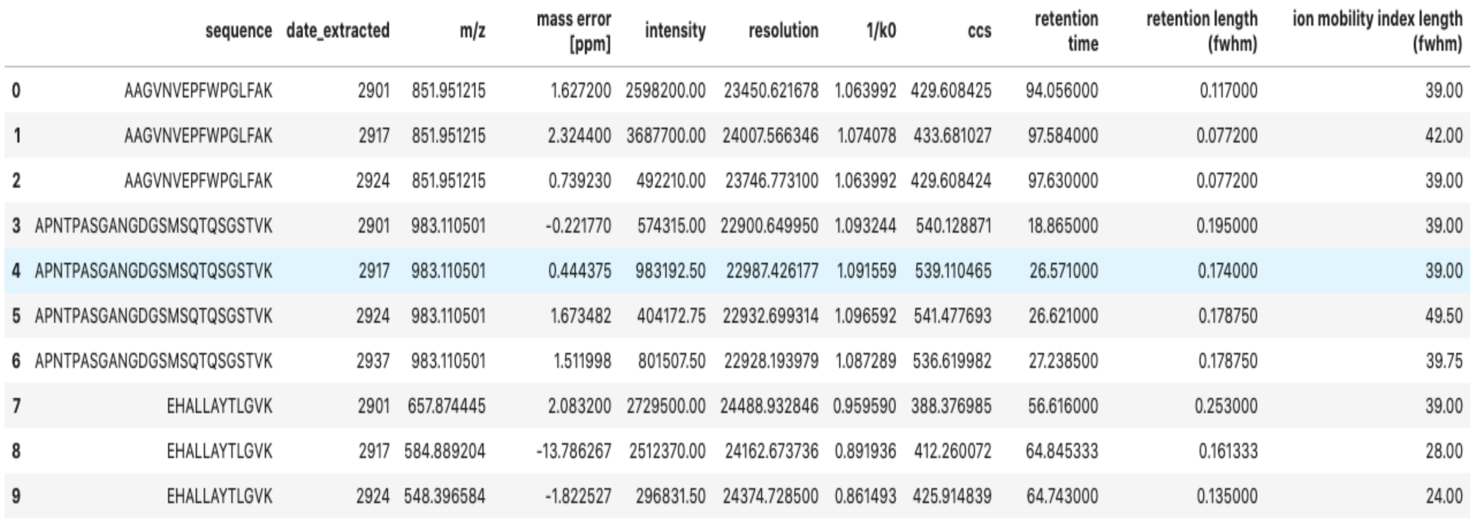


Figure 5 Histogram of CV of intensity

Figure 5 shows that highest frequency of CV of intensity (in %) is just less than 100%, which indicates that there is very high variation in intensity for peptides.

4.2. **<u>Part 2 – Processing of QC run files based on standard peptides</u>**

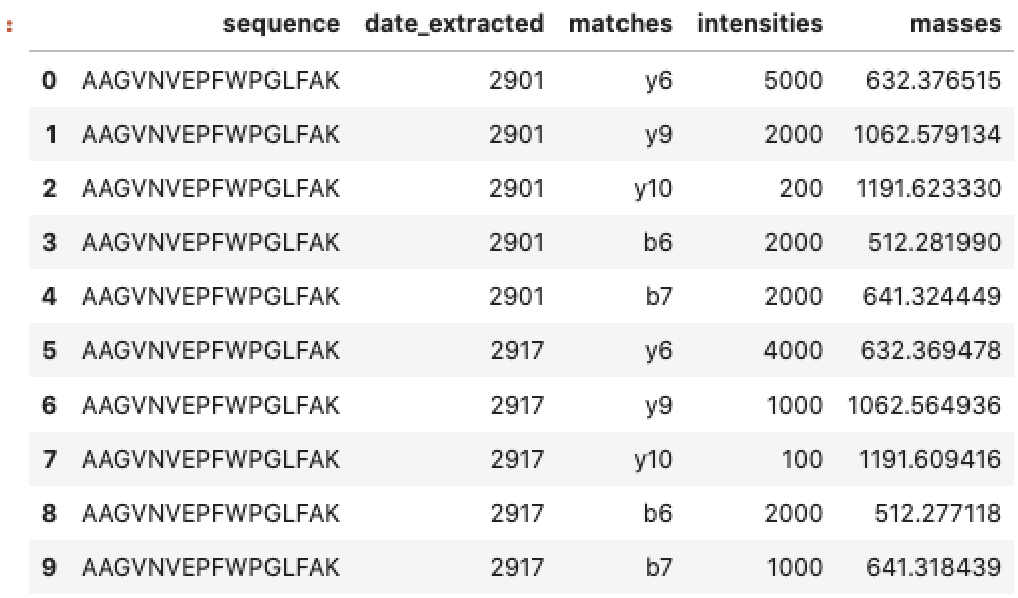
4.2.1. In the second part of the project, the goal is to extract data related to each peptide and its corresponding 5 fragment ions for every date (encoded form), which is extracted from raw file name. Table 9 shows the top 10 rows related to standard peptides.



	sequence	date_extracted	m/z	mass error [ppm]	intensity	resolution	1/k0	ccs	retention time	retention length (fwhm)	ion mobility index length (fwhm)
0	AAGVNVEPFWPGLFAK	2901	851.951215	1.627200	2598200.00	23450.621678	1.063992	429.608425	94.056000	0.117000	39.00
1	AAGVNVEPFWPGLFAK	2917	851.951215	2.324400	3687700.00	24007.566346	1.074078	433.681027	97.584000	0.077200	42.00
2	AAGVNVEPFWPGLFAK	2924	851.951215	0.739230	492210.00	23746.773100	1.063992	429.608424	97.630000	0.077200	39.00
3	APNTPASGANGDGSMSQTQSGSTVK	2901	983.110501	-0.221770	574315.00	22900.649950	1.093244	540.128871	18.865000	0.195000	39.00
4	APNTPASGANGDGSMSQTQSGSTVK	2917	983.110501	0.444375	983192.50	22987.426177	1.091559	539.110465	26.571000	0.174000	39.00
5	APNTPASGANGDGSMSQTQSGSTVK	2924	983.110501	1.673482	404172.75	22932.699314	1.096592	541.477693	26.621000	0.178750	49.50
6	APNTPASGANGDGSMSQTQSGSTVK	2937	983.110501	1.511998	801507.50	22928.193979	1.087289	536.619982	27.238500	0.178750	39.75
7	EHALLAYTLGVK	2901	657.874445	2.083200	2729500.00	24488.932846	0.959590	388.376985	56.616000	0.253000	39.00
8	EHALLAYTLGVK	2917	584.889204	-13.786267	2512370.00	24162.673736	0.891936	412.260072	64.845333	0.161333	28.00
9	EHALLAYTLGVK	2924	548.396584	-1.822527	296831.50	24374.728500	0.861493	425.914839	64.743000	0.135000	24.00

Table 9. Top 10 rows of standard peptides info extracted cumulatively from successive QC runs data

4.2.2. Similarly, data related to fragment ions, with highest intensities, of each standard peptide for each date (encoded form) has also been extracted as shown in Table 10.



	sequence	date_extracted	matches	intensities	masses
0	AAGVNVEPFWPGLFAK	2901	y6	5000	632.376515
1	AAGVNVEPFWPGLFAK	2901	y9	2000	1062.579134
2	AAGVNVEPFWPGLFAK	2901	y10	200	1191.623330
3	AAGVNVEPFWPGLFAK	2901	b6	2000	512.281990
4	AAGVNVEPFWPGLFAK	2901	b7	2000	641.324449
5	AAGVNVEPFWPGLFAK	2917	y6	4000	632.369478
6	AAGVNVEPFWPGLFAK	2917	y9	1000	1062.564936
7	AAGVNVEPFWPGLFAK	2917	y10	100	1191.609416
8	AAGVNVEPFWPGLFAK	2917	b6	2000	512.277118
9	AAGVNVEPFWPGLFAK	2917	b7	1000	641.318439

Table 10. Top 10 rows of fragment ions info of standard peptides extracted cumulatively from successive QC runs data

4.3. Part 3 – Visualization of QC runs data

4.3.1. In the last part, data is visualized to identify the changes in parameters over a period of time. For illustration, we have taken 3 peptide sequences to measure the change in parameters over a period of time.

4.3.2. Figure 6 shows that mass-over-charge ratio vs date for different peptides. It shows that m/z remains constant over a period of time for all 3 peptides.

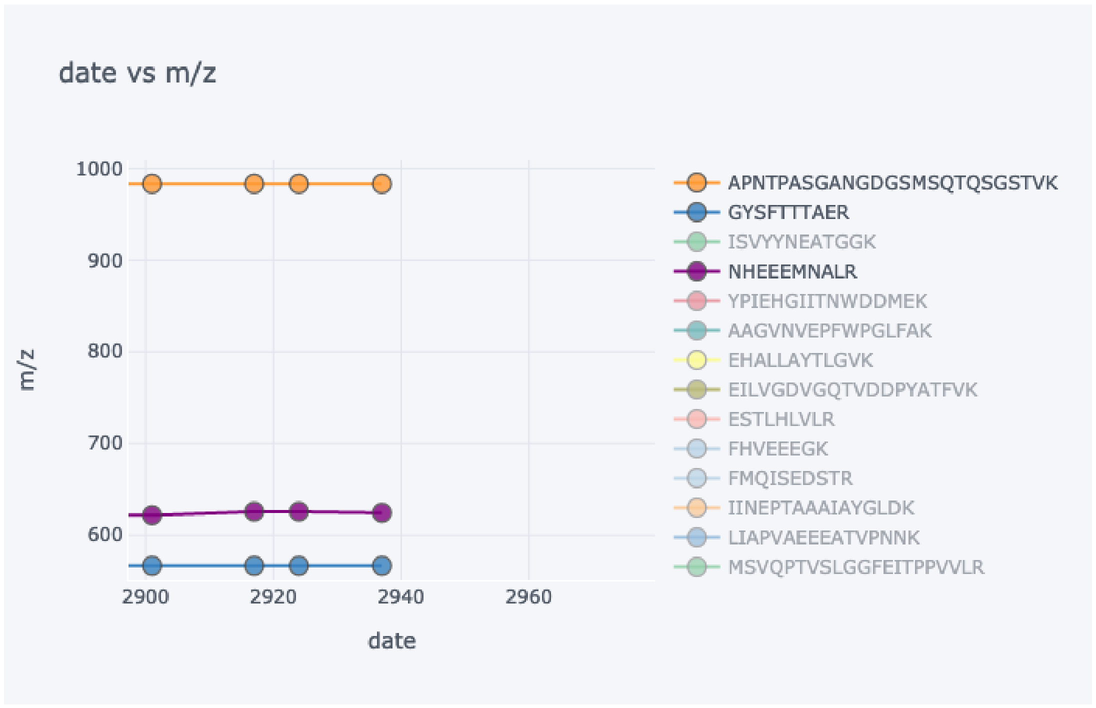


Figure 6 Line plot of date vs mass-to-charge ratio

4.3.3. Figure 7 shows that ion mobility index length increasing on 3rd day and then falling for all 3 peptides.

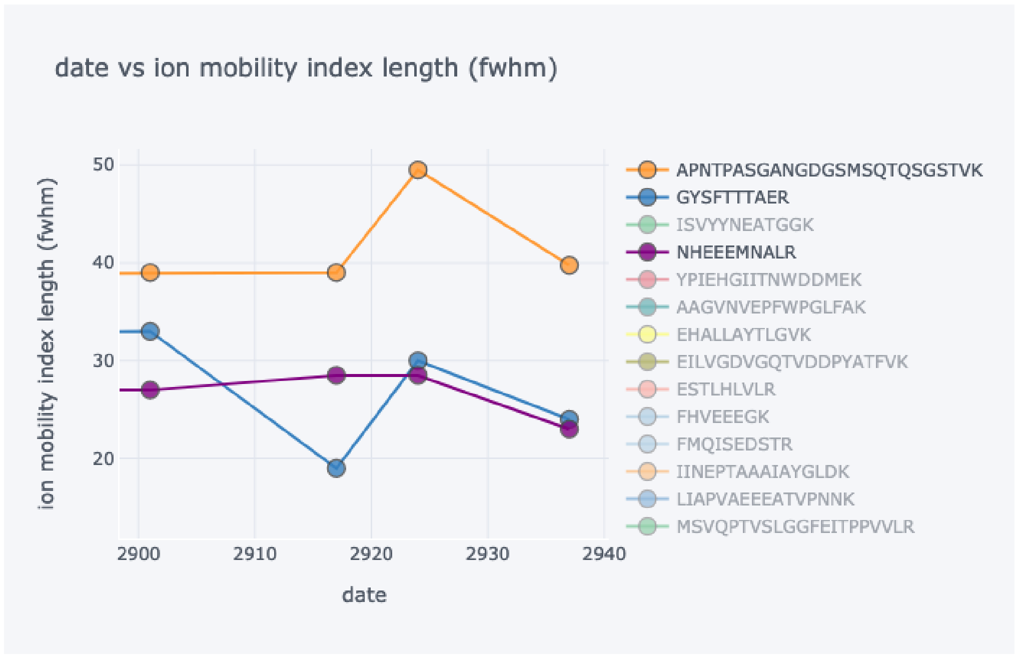


Figure 7 Line plot of date vs ion mobility index length

4.3.4. Figure 8 shows that retention length decreases on the second day but again increases on next day. However, GYS-peptide (blue line) does not reach its previous length.

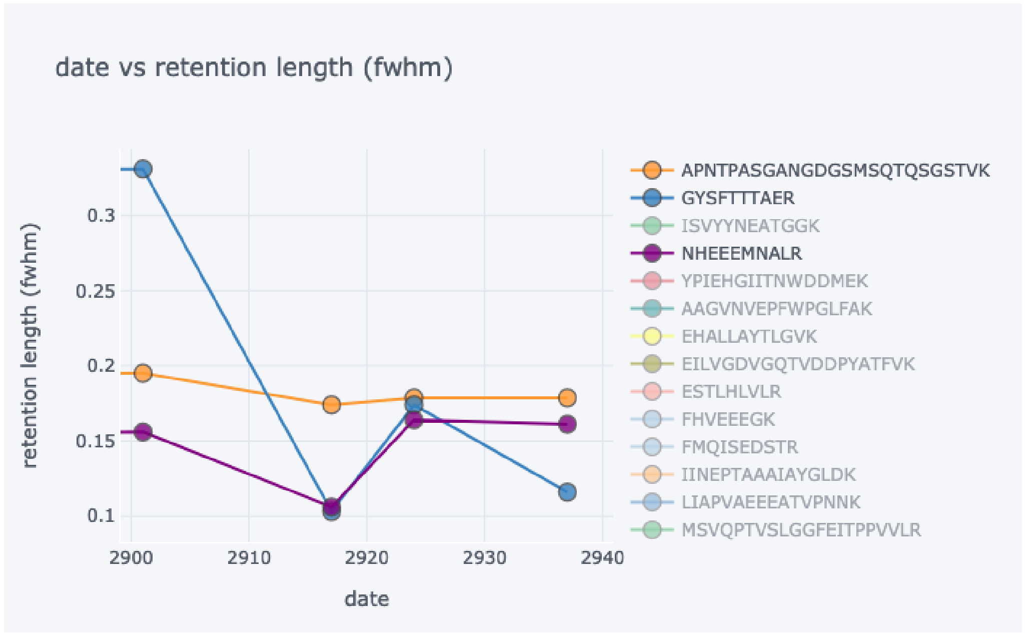


Figure 8 Line plot of date vs retention length

4.3.5. Figure 9 shows that retention time increases for all peptides. Ideally, if we use the same experimental conditions, the sample's retention time should not change significantly.

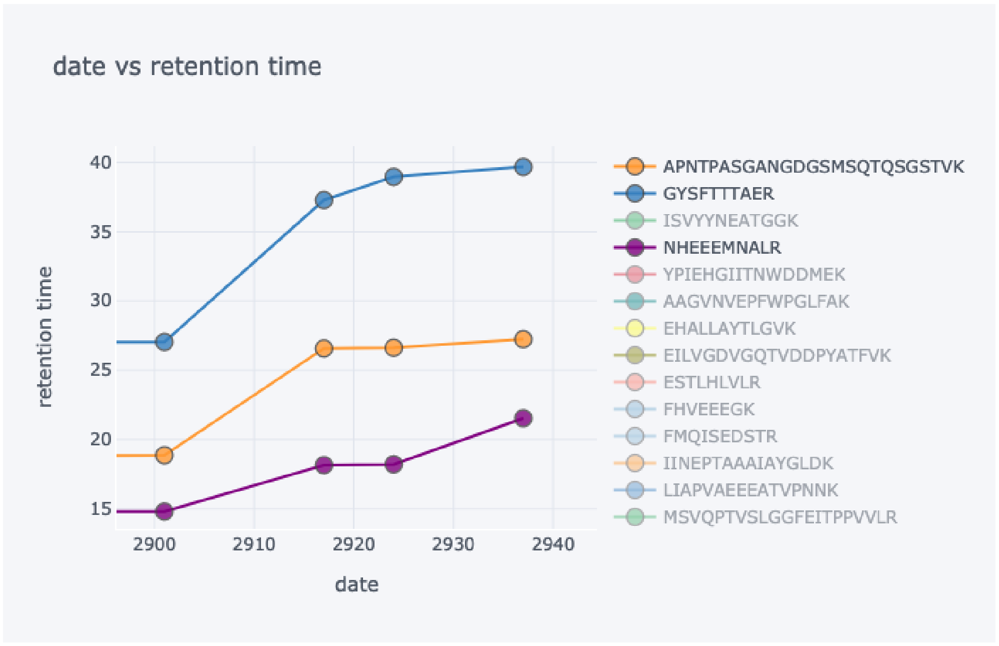


Figure 9 Line plot of date vs retention time

4.3.6. Figures 10 and 11 show that ion mobility (1 / K0) and CCS (Collision Cross Section) of all three peptides have stable values.

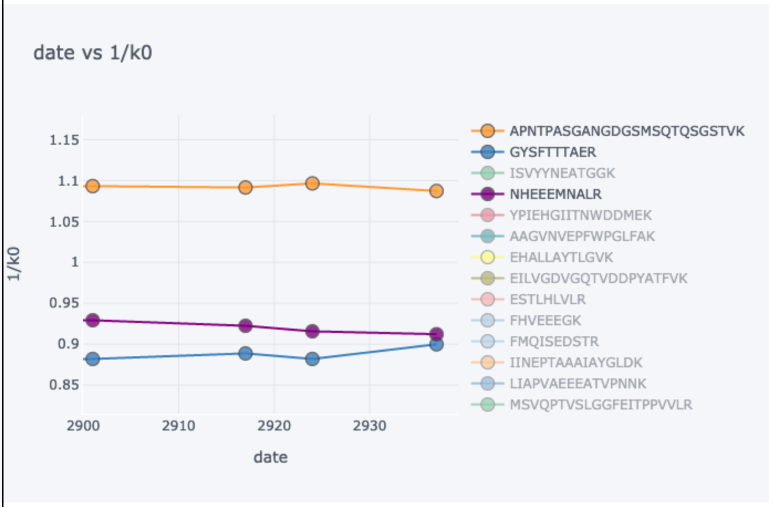


Figure 10 Line plot of date vs inverse reduced mobility

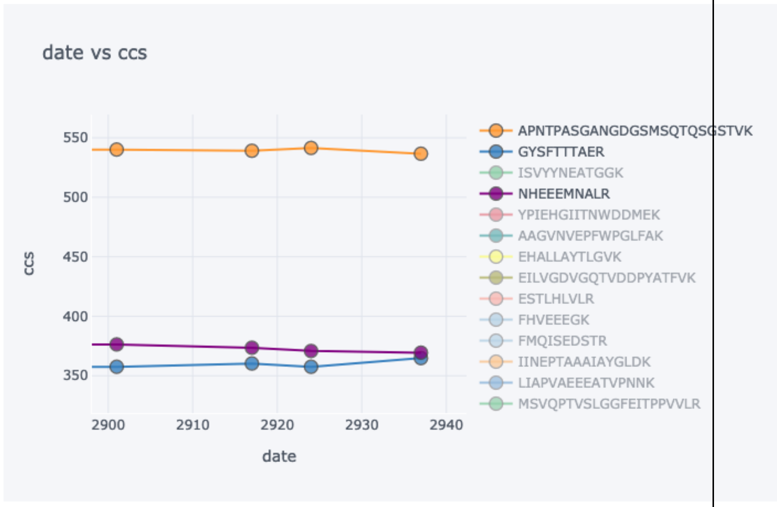


Figure 11. Line plot of date vs ccs

4.3.7. Figure 12 shows that the resolution of APN-peptide is almost stable, whereas the resolution of GYS-peptide has very high variation over time.

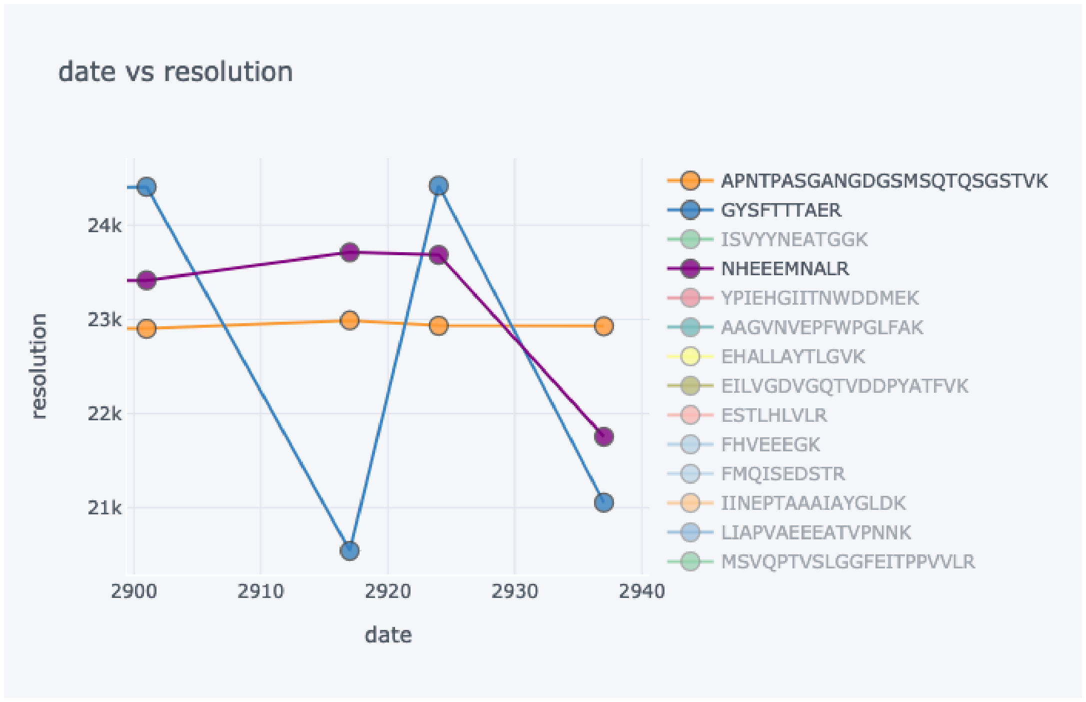
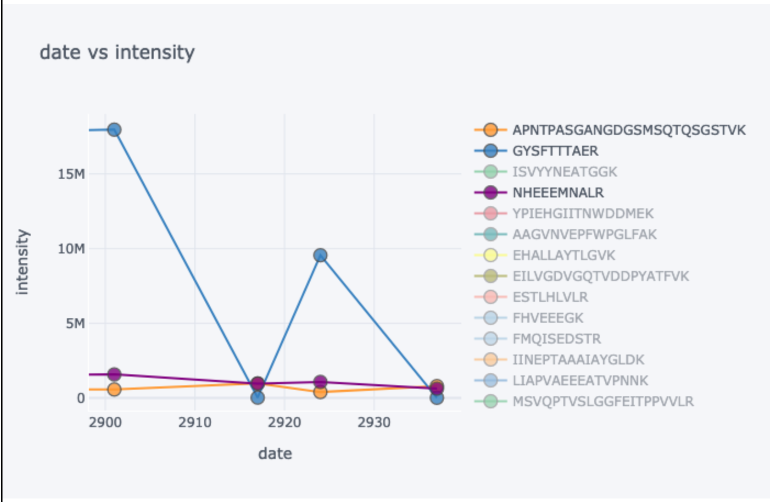


Figure 12. Line plot of date vs resolution

4.3.8. Figures 13 and 14 also show that intensity and mass errors of GYS-peptide also have very high variation compared to the other two peptides.



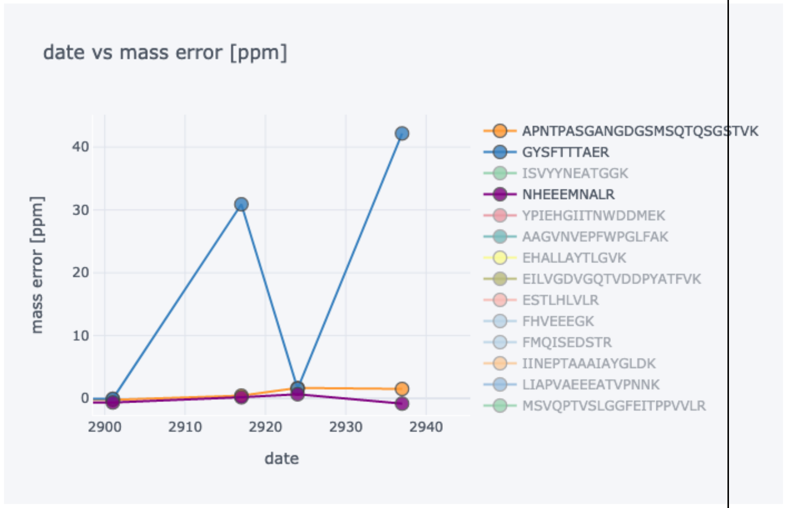


Figure 13. Line plot of date vs intensity

Figure 14. Line plot of date vs mass error

4.3.9. Figures 15-17 show changes in intensities for the top 5 fragment ions (based on standard peptides data) for each of these peptides. For APN-peptide, only 4 out of 5 fragment ions are present in QC runs. Out of these intensities of b4 and b12 ions increase over time. In the case of GYS-peptide, intensities of all 5 ions fall on 2nd date and again raise after that. However, in the case of last NHE-peptide, ion intensities increase on the same day and decrease later.

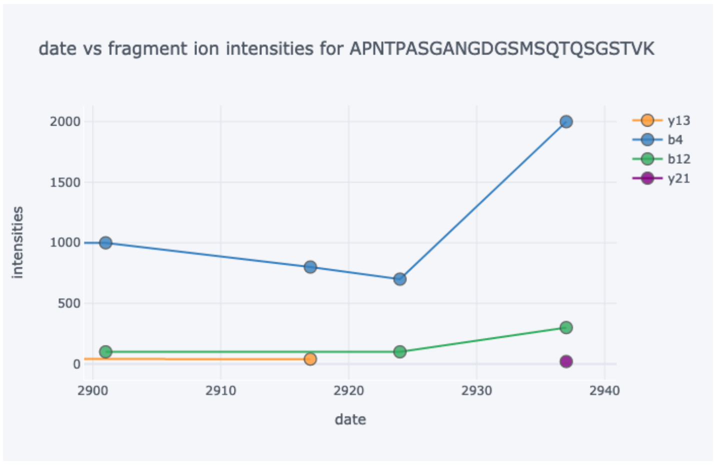


Figure 15. Line plot of date vs fragment ions intensities for a selected peptide

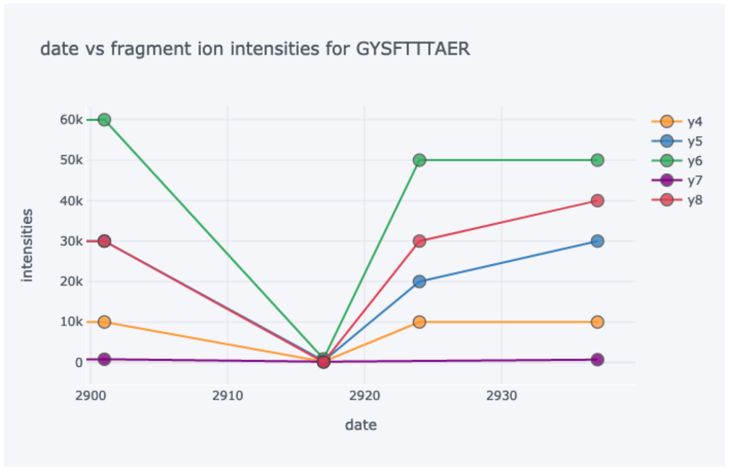


Figure 16. Line plot of date vs fragment ions intensities for a selected peptide

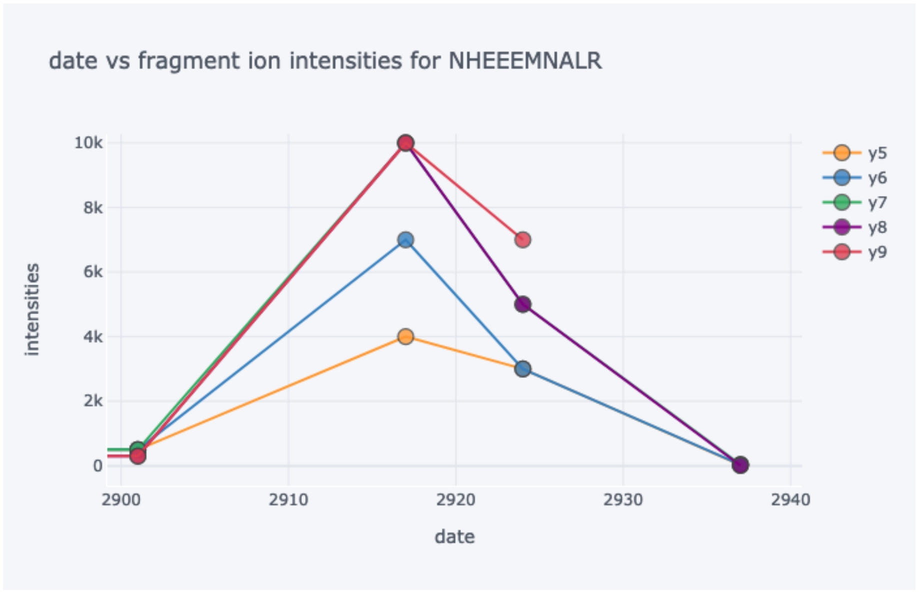


Figure 17. Line plot of date vs fragment ions intensities for a selected peptide

4.4. In this way, we can look at the patterns of any of extracted standard peptides, and their corresponding fragment ions of all QC Runs data obtained by Mass spectrometer.

4.5. In summary, this whole workflow effectively monitors the performance of LC-MS system through the lens of selected standard peptides and aids in tuning the system for best performance.

5. Conclusion

5.1. LC-MS system is prone to many variabilities, which decrease the reliability of the system in identifying the proteins. To control the quality of the system, a series of peptides and fragment ions have been chosen as standards, and various parameters such as intensities, m/z, ion mobilities have been monitored for these standards across successive runs of the system. This would help in tuning different parameters of the LC-MS system to improve quality.

6. References

6.1. Quality control in mass spectrometry-based proteomics by Wout Bittremieux, David Tabb et al.

6.2. *pmartR*: Quality Control and Statistics for Mass Spectrometry-Based Biological DataKelly G. Stratton, Bobbie-Jo M. Webb-Robertson et al.

7. Table and figures of data analysis

7.1. List of Figures:

7.2. List of tables

8. Appendix – Python code

8.1. **Part 1**

```
#!/usr/bin/env python
# coding: utf-8
# # Code to extract benchmark peptides from Spectrometer data
# In[1]:
import pandas as pd
import numpy as np
import os
from pandas import ExcelWriter
import seaborn as sns
import matplotlib.pyplot as plt
import re
import warnings
# # Reading all files in the test/txt directory
# # Please enter the location of Maxquant output in test_directory variable
# In[2]:
test_directory = r'/Volumes/Transcend/OneDrive/UoH DS/Summer 2020/Spectrometry
lab internship/new_MQoutput/MaxQuantOutput/test/txt'
# In[3]:
#display all files in the directory
for root, dirs, files in os.walk(test_directory, topdown=False):
    file_names = []
    print('root: ', root)
    print('filenames--')
    for name in files:
        if name[0] is not '.':
            print(os.path.join(root, name))
            file_names.append(os.path.join(root, name))

# In[4]:
warnings.filterwarnings("ignore")
#Reading filenames of evidence and msms files
for name in file_names:
    if name.endswith('evidence.txt'):
        file_evidence = name
    elif name.endswith('msms.txt'):
        file_msms = name
    elif name.endswith('allPeptides.txt'):
        file_allpeptides = name

# In[5]:
print(file_evidence)
```

```
df_evidence = pd.read_table(file_evidence)
df_evidence.head()

# In[6]:
print(file_msms)
df_msms = pd.read_table(file_msms)
df_msms.head()

# In[7]:
print(file_allpeptides)
df_allpeptides = pd.read_table(file_allpeptides)
df_allpeptides.head()

# ## Info needed for each peptide:
#         Peptide sequence - To be extracted from evidence file
#         m/z             - Evidence file
#         mass error (ppm) - Evidence file
#         Retention time (FWHM) - allpeptides
#         Retention length - Evidence file
#         Intensity       - Evidence file
#         Resolution      - Evidence file
#         1/k0            - Evidence file
#         CCS             - Evidence file
#         Ion mobility index length (FWHM) - allpeptides
#
# ## The following info needed for fragment ions:
#         Matches - msms file
#         Intensities- msms file
#         Masses- msms file

# # Evidence file processing

# ## Grouping content to Raw files

# In[8]:
evidence_grouped = df_evidence.groupby(['Raw file'])
evidence_grouped.count()

# In[9]:
evidence_grouped['Retention time'].agg([ 'mean', 'min','max'])

# ## Extracting evidence data for charge = 2
# In[10]:
```

```
#Filter for charge
df_evidence_q2 = df_evidence.query('Charge == 2')
df_evidence_q2.describe()

# ## Convert Retention time column to date time format
# In[11]:

df_evidence_q2['RT_timeform']  = pd.to_timedelta(df_evidence_q2['Retention time'],
unit='minutes')
df_evidence_q2.head()

# ## Resample for every 5 mins and extract peptides list
# In[12]:
#Time resampling of evidence files
resampled = df_evidence_q2.resample(rule = '5T',on = 'RT_timeform')
num_rawfiles = df_evidence_q2['Raw file'].unique().size

#aggregation function for grouped and resampled data
def fun_count_rawfiles(series):
    return series.unique().size

#aggregate dfunction for calculation of cv
def func_cv(df):
    return np.abs(df.std()*100/df.mean())

#creating empty list for saving standard peptides later
selected_peptides = []

#Read each group of 5 mins of peptides and extract 1 peptide from each
for name, group in resampled:
    print("Group: ", name)
    print("-" * 27)
    sample_group = group[['Sequence', 'Raw file', 'Intensity','Retention time' ]]
    print(sample_group, end="\n\n")

    #grouping data by Sequence and aggregating based on count of raw files, max for
Intensity and cv for RT
    grouped = sample_group.groupby("Sequence")
    result = grouped.agg( {'Raw file':fun_count_rawfiles,
                      'Intensity':np.max,
                     'Retention time': func_cv}).rename(columns = {'Intensity': 'max_intensity',
                                                   'Retention time': 'cv_retentiontime'})
#     print("\nresult: \n",result)
```

```
    #filtering peptides which are present in all Raw files
    result_interm = result[result["Raw file"] == num_rawfiles ]
#     print("\nresult_interm: \n",result_interm)
    #selecting top 5 peptides with highest intensities and choosing peptide with lowest cv
    num_peptides = result_interm.shape[0]

    if num_peptides > 0:
        if num_peptides > 5:
            filter_top_n = 5
        else:
            filter_top_n = num_peptides

        print('\n filter_top_n: ',filter_top_n)
        result_interm_topn = result_interm.nlargest( filter_top_n,['max_intensity'])
        print('\nresult_interm_topn: \n', result_interm_topn)
        result_selected = result_interm_topn['cv_retentiontime'].idxmin()

        #if cv_retention time is NaN, choosing peptide with highest intensity
        result_selected = np.where( result_selected is np.NaN,
result_interm['max_intensity'].idxmax(), result_selected )
        print("\n result_selected: ",result_selected)
        selected_peptides.append(result_selected.item())


# In[13]:
print("number of selected_peptides: ",len(selected_peptides))
print("selected_peptides: ",selected_peptides)


# ## Verifying the presence of selected peptides across all Raw files

# In[14]:
df_evidence_sample = df_evidence_q2[
df_evidence_q2['Sequence'].isin(selected_peptides) ]
df_evidence_sample.groupby(['Sequence', 'Raw file']).count()['id'].unstack()


# ### Above table shows the distribution of peptides across all Raw files

# ## Info needed for each peptide:
#        Peptide sequence - To be extracted from evidence file
#        m/z            - Evidence file
#        mass error (ppm) - Evidence file
#         Retention length - Evidence file
#         Intensity       - Evidence file
#         Resolution      - Evidence file
#         1/k0           - Evidence file
#         CCS            - Evidence file

# In[15]:
evidence_columns = ['id', 'Sequence', 'Raw file','m/z', 'Mass error [ppm]', 'Intensity',
                'Resolution', '1/K0', 'CCS','Retention time' ]
df_evidence_mod = df_evidence_sample[evidence_columns]
df_evidence_mod.head()


# ## Calculating CV and mean for Retention time, Intensity and other columns
# In[16]:
final_evidence_columns = ['Sequence','m/z', 'Mass error [ppm]','Intensity', 'Resolution',
'1/K0', 'CCS',
                    'Retention time']

df_evidence_mod_final =
df_evidence_mod[final_evidence_columns].groupby('Sequence').agg(
   mean_retention_time = ('Retention time','mean'), cv_retention_time = ('Retention
time', func_cv),
   mean_intensity = ('Intensity','mean'), cv_intensity = ( 'Intensity', func_cv),
   mean_mz = ( 'm/z', 'mean'),
   mean_masserror_ppm = ( 'Mass error [ppm]', 'mean'),
   mean_resolution = ( 'Resolution', 'mean'),
   mean_1_k0 = ( '1/K0', 'mean'),
   mean_ccs = ( 'CCS', 'mean')
)
df_evidence_mod_final


# # Extracting Fragment ion info from msms data
# In[17]:
df_msms.describe()


# ### Filtering msms file with selected peptides list and charge = 2
# In[18]:
msms_columns = ['id','Sequence', 'Raw file', 'Matches', 'Intensities', 'Masses', 'Evidence
ID']
df_msms_filtered = df_msms.query("Sequence in @selected_peptides and Charge ==
2")[msms_columns]
```

```
df_msms_filtered.head()


# In[19]:
df_msms_filtered.describe()

# In[20]:
def fun_matches_pattern(element_matches):
   list_matches = element_matches.split(sep = ';')

   #regex pattern to detect "y/b<1 or 2 digit numbers>"
   regex = re.compile('^[y,b][0-9]{1,2}$')
   pattern_list_matches = []

   for word in list_matches:
      pattern = regex.findall(word)
      if len(pattern) == 0:
         pattern = [0]
      pattern_list_matches.append(pattern[0])

   return pattern_list_matches


# In[21]:
df_msms_filtered.loc[:, 'pattern_matches'] =
df_msms_filtered['Matches'].map(fun_matches_pattern)
df_msms_filtered.head()


# In[22]:
# function to filter masses and intensities for ions with filtered pattern
def func_filter_cols(df_input ):
   list_intensities = df_input[ 'Intensities'].split(sep = ';')
   list_masses = df_input[ 'Masses'].split(sep = ';')
   sel_list_intensities = []
   sel_list_matches = []
   sel_list_masses = []
   for index, item in enumerate(df_input['pattern_matches']):
      if item != 0:
         sel_list_matches.append(item)
         sel_list_intensities.append(list_intensities[index])
         sel_list_masses.append(list_masses[index])

   return (sel_list_matches,sel_list_intensities,sel_list_masses)
```

```
df_msms_filtered['filtered_cols'] = df_msms_filtered.apply(func_filter_cols, axis = 1)
df_msms_filtered.head()


# ## Processing filtered msms file to extract top 5 ions with highest intensity for each
sequence

# In[24]:
#function to transform matches, intensities and masses from single column to
#multiple columns
def fun_create_matches_df(input):
    print( 'input:\n',input)
    df_input = pd.DataFrame({ 'matches': input[0],
                        'intensities': input[1],
                        'masses': input[2]})
    return df_input

#create final msms file
df_msms_final = pd.DataFrame(index = df_evidence_mod_final.index)

#create dataframe with separate data for each fragment ion
df_frag_ions = pd.DataFrame(columns = ['Sequence', 'matches','intensities', 'masses'])

#grouping the msms file to extract fragment ion info
grouped_sequence = df_msms_filtered.groupby('Sequence')

for index, group in grouped_sequence:
    print("-" * 27)
    print('peptide sequence : ',index)
    print("-" * 27)
    group_mod = group[['Sequence', 'filtered_cols']].reset_index(drop = True)
#     print('group_mod: \n',group_mod )
    #creating dataframe to extract all fragment ions data for a peptide
    column_names = ['matches','intensities', 'masses']
    df_required = pd.DataFrame( columns = column_names)

    #concatenate all fragment ions info for a peptide
    for element in group_mod['filtered_cols']:
#         print('element: \n', element)
        df_input = pd.DataFrame({ 'matches': element[0],
                        'intensities': element[1],
                        'masses': element[2]})
#         print(df_input)
```

```
        df_required = pd.concat([df_required, df_input])

    df_required.reset_index(drop = True, inplace = True)
    print('\nfragment ions list: \n',(df_required))

    #filtering fragment ions with max intensities and creating new dataframe
    df_required['intensities'] = pd.to_numeric(df_required['intensities'])
    df_required['masses'] = pd.to_numeric(df_required['masses'])
    df_required_with_max =
df_required.groupby('matches')['intensities'].transform(np.max)
    max_intensity_idx = df_required_with_max == df_required[ 'intensities']
    df_max_intensity = df_required[ max_intensity_idx]
#     print('grouped df_required: \n', df_max_intensity  )
    df_test = df_max_intensity.drop_duplicates('matches').sort_values('intensities',
ascending=False).iloc[:5]
    df_test['Sequence'] = index
    print('\nchosen fragment ions: \n', df_test)
    #appending fragment ion data for all peptides
    df_frag_ions = df_frag_ions.append( df_test)
    df_msms_final.loc[index, 'fragment_ions'] = str(df_test.to_dict(orient = 'list'))


# In[25]:
df_frag_ions.head(10)


# # Combining evidence file data with msms file

# In[26]:
df_evid_fragions_join = df_evidence_mod_final.merge(df_frag_ions, left_on =
'Sequence', right_on='Sequence' )
df_evid_fragions_join


# ## Extracting Retention length (FWHM) and Ion mobility index length (FWHM) from
allpeptides file.

# In[27]:
df_allpeptides.head()


# In[28]:
df_allpeptides.columns
```

```
# In[29]:
msms_columns = ['id','Sequence', 'Raw file','Scan number']
df_msms_test = df_msms.query("Sequence in @selected_peptides and Charge ==
2")[msms_columns]
df_msms_test.head()


# In[30]:
allpept_columns = ['Sequence','Raw file', 'Scan number', 'Retention length (FWHM)', 'Ion
mobility index length (FWHM)']
df_allpept_merged = df_msms_test.merge( df_allpeptides, how = 'left', left_on= ['Scan
number', 'Raw file'],
                              right_on = ['MS/MS scan number', 'Raw file'])[allpept_columns]
df_allpept_merged.head()


# In[31]:
df_allpept_final = df_allpept_merged.groupby('Sequence').mean()[['Retention length
(FWHM)','Ion mobility index length (FWHM)']]
df_allpept_final


# In[32]:
df_benchmark_peptides = df_evid_fragions_join.merge(df_allpept_final, left_on=
'Sequence', right_index = True)
df_benchmark_peptides


# In[33]:
df_benchmark_peptides.head(10)


# # Creating new directory and saving final benchmarks peptides excel file

# In[34]:
# detect the current working directory and print it
current_path = os.getcwd()
print ("The current working directory is %s" % current_path)

# define the name of the directory to be created
new_path = "LCMS_data"
try:
   os.mkdir(new_path)
except OSError as error:
    print ("Creation of the directory {0} failed because of error {1}".format( new_path,
error))
else:
    print ("Successfully created the directory %s " % new_path)
finally:
    os.chdir( os.path.join(current_path, new_path))


# In[35]:
#write to excel dataset
filepath = 'Benchmark_peptides.xlsx'
with ExcelWriter(filepath) as writer:
    df_benchmark_peptides.to_excel(writer, sheet_name = ( 'df_benchmark_peptides') )
    writer.save()


# ## Histograms of CV for Retention time and Intensity

# In[36]:
plt.figure(figsize= (20,10))
histogram_cv_RT =
sns.distplot(df_benchmark_peptides['cv_retention_time']).set_title('Histogram of CV for
Retention time')
histogram_cv_RT.get_figure().savefig('histogram_cv_RT.png')
histogram_cv_RT


# In[37]:
plt.figure(figsize= (20,10))

histogram_cv_intensity =
sns.distplot(df_benchmark_peptides['cv_intensity']).set_title('Histogram of CV for
Intensity')
histogram_cv_intensity.get_figure().savefig('histogram_cv_intensity.png')
histogram_cv_intensity
```

8.2. Part 2

```
#!/usr/bin/env python
# coding: utf-8
# # Code to extract data from test set spectrometer data based on benchmarked peptides

# In[1]:
import pandas as pd
import numpy as np
import os
from pandas import ExcelWriter
import seaborn as sns
import matplotlib.pyplot as plt
import re
import warnings


# # Please enter input file path in test_directory variable

# In[2]:
###MQoutput path
test_directory = r'/Volumes/Transcend/OneDrive/UoH DS/Summer 2020/Spectrometry lab
internship/new_MQoutput/TestDataSet-2'


# In[3]:
print ("The current working directory is %s" % os.getcwd())

###benchmark peptides path
new_path = r"/LCMS_data"
current_dir = str(os.getcwd())
if current_dir.endswith(new_path):
   working_dir = current_dir
else:
   working_dir = current_dir + new_path

#changing working directory to benchmark peptides path
os.chdir(working_dir)
print ("The new working directory is %s" % os.getcwd())


# # Reading all files in the test/txt directory
# In[4]:
```

```
for root, dirs, files in os.walk(test_directory, topdown=False):
    file_names = []
    print('root: ', root)
    print('filenames--')
    for name in files:
        if name[0] is not '.':
            print(os.path.join(root, name))
            file_names.append(os.path.join(root, name))


# In[5]:
#Reading filenames of evidence and msms files
for name in file_names:
    if name.endswith('evidence.txt'):
        file_evidence = name
    elif name.endswith('msms.txt'):
        file_msms = name
    elif name.endswith('allPeptides.txt'):
        file_allpeptides = name

warnings.filterwarnings("ignore")

# In[6]:
print(file_evidence)
df_evidence = pd.read_table(file_evidence)
df_evidence.columns = (df_evidence.columns).str.lower()
df_evidence.head()

# In[7]:
print(file_msms)
df_msms = pd.read_table(file_msms)
df_msms.columns = (df_msms.columns).str.lower()
df_msms.head()

# In[8]:
print(file_allpeptides)
df_allpeptides = pd.read_table(file_allpeptides)
df_allpeptides.columns = (df_allpeptides.columns).str.lower()
df_allpeptides.head()

# # Read extracted benchmark peptides file
# In[9]:
file_benchmark_peptides = 'Benchmark_peptides.xlsx'
```

```
df_extracted_peptides = pd.read_excel(file_benchmark_peptides, index_col=0)
df_extracted_peptides.columns = (df_extracted_peptides.columns).str.lower()

df_extracted_peptides.head()


# # Process peptides data for visualization

# In[10]:
#showing list of benchmark peptide sequences
selected_peptides = df_extracted_peptides['sequence'].unique()
selected_peptides


# ## Filter evidence and allpeptides

# In[11]:
evidence_columns = [ 'sequence', 'raw file','m/z', 'mass error [ppm]', 'intensity',
                 'resolution', '1/k0', 'ccs','retention time' ]
df_evidence_viz = df_evidence.query("sequence in @selected_peptides")[evidence_columns]
df_evidence_viz


# In[12]:
msms_columns = ['id','sequence', 'raw file','scan number']
df_msms_test = df_msms.query("sequence in @selected_peptides")[msms_columns]
df_msms_test.head()


# In[13]:
allpept_columns = ['sequence','raw file', 'scan number', 'retention length (fwhm)', 'ion mobility
index length (fwhm)']
df_allpept_merged = df_msms_test.merge( df_allpeptides, how = 'left', left_on= ['scan
number', 'raw file'],
                              right_on = ['ms/ms scan number', 'raw file'])[allpept_columns]
df_allpept_merged.head()


# ## Reframing evidence file to include Rawfile name and date of processing

# In[14]:
#extracting date from Raw file name
def fun_date_extract(rawfile_names):
   #regex pattern to detect last numbers after underscore
```

```
    regex = re.compile('_([0-9]+)')
    date_extract = regex.findall( rawfile_names)

    return date_extract[-1]


# In[15]:
df_evidence_viz['date_extracted'] = df_evidence_viz['raw file'].map( fun_date_extract)
df_evidence_viz


# In[16]:
df_evidence_aggviz = df_evidence_viz.groupby( ['sequence', 'date_extracted']).mean()
df_evidence_aggviz


# ## Extracting FWHM from allpeptides file and separating date of extraction

# In[17]:
df_allpept_merged['date_extracted'] =  df_allpept_merged['raw file'].map( fun_date_extract)
df_allpept_merged


# In[18]:
df_allpept_aggviz = df_allpept_merged.groupby( ['sequence', 'date_extracted']).mean()[
['retention length (fwhm)', 'ion mobility index length (fwhm)']]
df_allpept_aggviz


# ## Merging evidence and all peptides files

# In[19]:
df_evid_allpept_viz = df_evidence_aggviz.merge(df_allpept_aggviz, left_index=True,
right_index=True)
df_evid_allpept_viz


# In[20]:
df_peptides_finalviz = df_evid_allpept_viz.reset_index().copy()
df_peptides_finalviz


# In[21]:
df_peptides_finalviz.head(10)
```

```
# # Process fragment ion data for visualization

# ## Filter msms file and extract fragment ions

# In[22]:
msms_columns = ['id','sequence', 'raw file', 'matches', 'intensities', 'masses']
df_msms_filtered = df_msms.query("sequence in @selected_peptides")[msms_columns]
df_msms_filtered


# In[23]:
df_msms_filtered['date_extracted'] = df_msms_filtered['raw file'].map( fun_date_extract)
df_msms_filtered


# In[24]:
#grouping the msms file to extract fragment ion info
grouped_msms = df_msms_filtered.groupby(['sequence', 'date_extracted'])

#creating dataframe to extract all fragment ions data for a peptide
column_names = ['sequence','date_extracted','matches','intensities', 'masses']
df_fragment_ions = pd.DataFrame( columns = column_names)

#Read each group and extract fragment ions parameters and create a dataframe
for index, group in grouped_msms:
    print("-" * 27)
    print('peptide sequence: ',index)
    print("-" * 27)
    group_mod = group[['sequence', 'date_extracted','matches', 'intensities',
'masses']].reset_index(drop = True)

    #extract fragment ion parameters into list variables
    list_matches = group_mod[ 'matches'].str.split(';').tolist()[0]
    list_intensities = group_mod[ 'intensities'].str.split( ';').tolist()[0]
    list_masses = group_mod[ 'masses'].str.split(';').tolist()[0]

    req_sequence = index[0]
    benchmark_ions = list(df_extracted_peptides.query('sequence ==
@req_sequence')['matches'])
    print('benchmark_ions: \n',benchmark_ions )

    #concatenate all fragment ions info for a peptide
    for ion in list_matches:
#        print('ion: \n', ion)
        if ion in benchmark_ions:
            req_index = list_matches.index(ion)
            df_input = pd.DataFrame({'sequence':index[0],
                                     'date_extracted' : index[1],
                                     'matches': [list_matches[req_index]],
                                     'intensities': [list_intensities[req_index]],
                                     'masses': [list_masses[req_index]]
                                    })

            df_fragment_ions = pd.concat([df_fragment_ions, df_input])

    df_fragment_ions.reset_index(drop = True, inplace = True)
#    print('df_fragment_ions: \n', df_fragment_ions)

#convert columns to float
df_fragment_ions['intensities'] = pd.to_numeric(df_fragment_ions['intensities'])
df_fragment_ions['masses'] = pd.to_numeric(df_fragment_ions['masses'])

df_frag_ions_finalviz = df_fragment_ions

# In[25]:
df_frag_ions_finalviz.head(10)

# # Update existing peptides and ions sheets with new data

# In[26]:
current_dir = os.getcwd()
required_file = 'Peptides_viz.xlsx'

#Read all filenames in current directory and create dataframes for old data
for *rest, files in os.walk(current_dir, topdown=False):
    print('All files in current directory: ', files)
    #Read existing peptides file if it exists
    if required_file in files:
        print('FOUND REQUIRED FILE: ', required_file)
        df_peptides_finalviz_old = pd.read_excel(required_file, sheet_name= 'peptides_viz_data',
index_col=0)
        df_frag_ions_finalviz_old = pd.read_excel(required_file, sheet_name=
'fragment_ions_vizdata', index_col=0)
    else:
        print('NOT FOUND REQUIRED FILE: ', required_file)
        df_peptides_finalviz_old = pd.DataFrame( columns= df_peptides_finalviz.columns)
    df_frag_ions_finalviz_old = pd.DataFrame( columns= df_frag_ions_finalviz.columns)

print('\nOld peptides_finalviz data: \n', df_peptides_finalviz_old.head())
print('\nOld frag_ions_finalviz data: \n', df_frag_ions_finalviz_old.head())

# In[27]:
#create new dataframes by combining old data with new data
df_peptides_finalviz_new = pd.concat( [df_peptides_finalviz_old, df_peptides_finalviz])
df_frag_ions_finalviz_new = pd.concat( [df_frag_ions_finalviz_old, df_frag_ions_finalviz])

print('\n updated peptides_finalviz : \n', df_peptides_finalviz_new.head())
print('\n updated frag_ions_finalviz : \n', df_frag_ions_finalviz_new.head())

# In[28]:
#write new files into excel file
filepath = 'Peptides_viz.xlsx'
with ExcelWriter(filepath) as writer:
  df_peptides_finalviz_new.to_excel(writer, sheet_name = ( 'peptides_viz_data') )
  df_frag_ions_finalviz_new.to_excel(writer, sheet_name = ( 'fragment_ions_vizdata') )
  writer.save()
```

8.3. <u>Part 3</u>

```
#!/usr/bin/env python
# coding: utf-8

# # Code to plot peptides and fragment ions visualization

# In[1]:
import pandas as pd
import numpy as np
import os
from pandas import ExcelWriter
import cufflinks as cf
import plotly.offline
import warnings

# # Read Peptides data file
# In[2]:
print ("The current working directory is %s" % os.getcwd())

###Processed peptides data path
working_dir = r'/Users/kishore/LCMS_data'
#changing working directory to above path
os.chdir(working_dir)
```

```
#Reading the file
peptides_finalviz = 'Peptides_viz.xlsx'
df_peptides_finalviz = pd.read_excel(peptides_finalviz, sheet_name='peptides_viz_data', index_col= 0)
df_frag_ions_finalviz = pd.read_excel(peptides_finalviz, sheet_name='fragment_ions_vizdata',
index_col= 0)
print('df_extracted_peptides: \n', df_peptides_finalviz.head())
print('df_frag_ions_finalviz: \n', df_frag_ions_finalviz.head())

# # Data visualizations
# ## Visualization of date vs parameters using plotly library

# In[3]:
cf.go_offline()
cf.set_config_file(offline=True, world_readable=True)
plotly.offline.init_notebook_mode()

# In[4]:
warnings.filterwarnings("ignore")
for col in df_peptides_finalviz.columns[2:]:
   df_peptides_finalviz.iplot(kind = 'scatter', mode = 'lines+markers',
            x = 'date_extracted', y = str(col), categories = 'sequence',
           title = 'date vs '+str(col), xTitle = 'date', yTitle = str(col))


# # Extracting Fragment ion info from msms data
# In[5]:
df_frag_ions_finalviz.head()


# In[6]:
warnings.filterwarnings("ignore")

#showing list of benchmark peptide sequences
selected_peptides = df_frag_ions_finalviz['sequence'].unique()

for peptide in selected_peptides:
   df_peptide = df_frag_ions_finalviz.query('sequence == @peptide')[['sequence', 'date_extracted',
'matches', 'intensities']]
   print('Cross table for above petide: \n', df_peptide)
   if df_peptide.empty:
      print('\n#######No fragment ion data for this peptide########\n')
   else:
      df_peptide.iplot(kind = 'scatter', mode = 'lines+markers',
            x = 'date_extracted', y = 'intensities', categories = 'matches',
           title = 'date vs fragment ion intensities for '+peptide, xTitle = 'date', yTitle = 'intensities')
```