Supporting Information for

Dynamic Cluster Analysis: An Unbiased Method for Identifying A+2 Element Containing Compounds in Liquid Chromatographic High-Resolution TOF Mass Spectrometric Data

Author Names

Aaron John Christian Andersen,[1,2] Per Juel Hansen,[3] Kevin Jørgensen,[1] and Kristian Fog Nielsen[2]

Author Affiliations

[1] National Food Institute, Technical University of Denmark, Lyngby, Denmark

[2] Department of Biotechnology and Biomedicine, Technical University of Denmark, Lyngby, Denmark

[3] Marine Biology Section, University of Copenhagen, Helsingør, Denmark

Table of Contents

Text S-1: *Database Decision Boundary Modelling* 3

Text S-2: *Biomass Extraction Protocol* 4

Figure S-1: *Inter-isotopic Mass Accuracy* 5

Figure S-2: *Illustrated Polynomial Midpoint Equations* 6

Figure S-3: *Graphical user interface of DCAnalysis v1.07* 7

Figure S-4: *Comparison of ions found by MeHaloCoA and DCA-Hal as EIC* 8

Table S-1: *Reference Starndard Molecular Formulas and Structures* 13

Table S-2: *Strain Information of* P. parvum 14

Table S-3: *DCA-Hal and DCA-Sul results from simulated isotope patterns of the Antibase* 15

Table S-4: *Complete table of A+2 Elements* 17

Table S-5: *DCA-Hal and DCA-Sul Reference Standards Results* 24

Table S-6a: *DCA-Hal comparison to MeHaloCoA: amphotericin B Results* 25

Table S-6b: *DCA-Hal comparison to MeHaloCoA: BE 52440A isomer Results* 26

Table S-6c: *DCA-Hal Comparison to MeHaloCoA: bis-sclerotioramin Results* 27

Table S-6d: *DCA-Hal comparison to MeHaloCoA: cephalosporin-C Results* 28

Table S-6e: *DCA-Hal comparison to MeHaloCoA: citreo isocumarin Results* 29

Table S-6f: *DCA-Hal comparison to MeHaloCoA: dichlorodiaportin Results* 30

Table S-6g: *DCA-Hal comparison to MeHaloCoA: enniatin-A Results* 31

Table S-6h: *DCA-Hal comparison to MeHaloCoA: folic acid Results* 32

Table S-6i: *DCA-Hal comparison to MeHaloCoA: malformin-A Results* 33

Table S-6j: *DCA-Hal comparison to MeHaloCoA: neosolaniol Results* 34

Table S-6k: *DCA-Hal comparison to MeHaloCoA: nidulin Results* 35

Table S-6l: *DCA-Hal comparison to MeHaloCoA: ochratoxin alpha Results* 36

Table S-6m: *DCA-Hal comparison to MeHaloCoA: penicillin-G Results* 37

Table S-6n: *DCA-Hal comparison to MeHaloCoA: penitrem A Results* 38

Table S-6o: *DCA-Hal comparison to MeHaloCoA: phalloidin Results* 39

Table S-6p: *DCA-Hal comparison to MeHaloCoA: phomopsin A Results* 40

Table S-6q: *DCA-Hal comparison to MeHaloCoA: roridin A Results*.....41
Table S-6r: *DCA-Hal comparison to MeHaloCoA: vitamin B1 Results*.....42
Table S-7: *Mass Accuracy Comparison*.....48
Table S-8: *Isotope Pattern and Data Extraction Accuracy Comparison* 51
Table S-9: *Inter-Isotopic Mass Accuracy Comparison*.....61
Table S-10*: Bruker DCA-Hal Results*.....63
Table S-11: *Agilent DCA-Hal Results*.....65
Table S-12: *Identified Prymnesin-like Chemical Features* 67
Table S-13: *The Ten Chemical Features With the Highest Intensity Identified by MeHaloCoA* 68
Equation S-1: *A to A+2 isotope cluster spacing boundary equation for Group-Cl and Group-C*.....69
Equation S-2: *A to A+2 isotope cluster spacing boundary equation for Group-S and Group-C* 69
Equation S-3: *A to A+2 isotope cluster spacing lower boundary equation for Group-S and Group-Cl* 69
Code S-1: *Source code for DCAnalysis v1.07*.....90
Code S-2: *Source code for FormExtract v1.01* 118
Code S-3: *Source code for PatExtract v1.01*.....141
Code S-4: *Source code for IsoPat v1.09* 181
References 182

Text S-1: *Database Decision Boundary Modelling*

The Marinlit natural products database was analyzed to determine the maximum O to C ratio (O:C), the maximum N to C ratio (N:C), and the minimum H to C ratio (H:C) that includes 99.8% of the database for each ratio.

The decision boundaries for the A+1 to A+2 isotope cluster spacings were calculated by analysing realistic hypothetical molecular formulas based on these ratios. The spread in the A+1 to A+2 isotope cluster spacing within each group was mostly influenced by the O:C, N:C, and the degree of saturation (H:C), therefore by simulating the isotope patterns of hypothetical molecular formulas with realistic degrees of these parameters it was possible to determine upper and lower limits of each of these groups at mass ranges not covered by the database. The upper limits of Group-Cl and Group-S were compared with the lower limit of Group-C and the average differences between these limits were calculated to give two equations for the isotope cluster spacing decision boundaries between Group-Cl and Group-C and between Group-S and Group-C.

In addition to these two decision boundaries, an equation for the lower limit for the isotope spacing of Group-Cl and Group-S was set to the minimum isotope cluster spacing expected for saturated molecular formulas containing S or Cl, 0.9936, minus a mass error of 5 ppm.

The decision boundaries for the A:A+2 intensity ratios were determined by comparing the derived hypothetical molecular formulas to the Marinlit database. The equations for the minimum boundaries of Group-Cl and Group-S were derived from the A:A+2 intensity ratios of the best fit molecular formulas, less an acceptable error of 5% to account to analyzer inaccuracies. 5% is a value that has been suggested previously for molecular formula calculations.[4]

Text S-2: *Biomass Extraction Protocol*

The biomass samples of *P. parvum* in 50 mL falcon tubes were washed three times with 1 mL acetone, each time vortexed for 5 sec, placed in an ultrasonication bath for 10 min, centrifuged at 4000 g, at 15 °C for 10 min, supernatant then discarded. The acetone extracted biomass was then extracted with 1 mL of MeOH, vortexed for 5 sec, placed in an ultrasonication bath for 10 min, centrifuged at 4000 g, at 15 °C for 10 min. The supernatant was transferred to 1.5 mL Eppendorf tubes. These were then centrifuged at 15000 g for 5 min, and 800 µL of supernatant was transferred to HPLC vials for analysis by Dionex/Bruker UHPLC-QTOF system. For analysis on the Agilent UHPLC-QTOF the extracts were up concentrated to half the volume. All extracts were analyzed with 3 µL injections on the Dionex/Bruker UHPLC-QTOF system and 10 µL on the Agilent UHPLC-QTOF system.

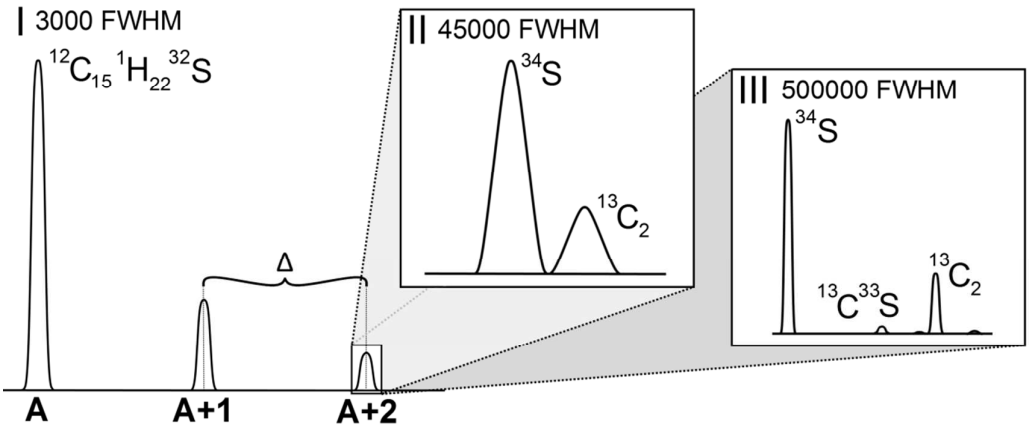


Figure S-1: *Inter-isotopic Mass Accuracy*

I: Simulated isotope pattern of erectathiol ($C_{15}H_{22}S$) at a full width at half maximum (FWHM) resolution of 3000. Δ is the A+1 to A+2 isotope cluster spacing, the error for this value, termed here, the inter-isotopic mass accuracy. II: The A+2 isotope cluster for erectathiol at a FWHM resolution of 45000. III: The individual isotopic ions of the A+2 isotope cluster of erectathiol, simulated at a FWHM resolution of 500000. Due to the large influence of the ^{34}S isotopic ion, a weighted average of all isotopologues of the A+2 isotope cluster will cause the A+1 to A+2 isotope cluster spacing to be reduced compared to a compound that does not contain S. Using a static value as a decision boundary for metabolite classification based on the A+1 to A+2 isotope cluster spacing, such as 1.001 u, results in poor classification compared to using a dynamic boundary. For example, gadusol ($C_8H_{12}O_6$), has an A+1 to A+2 isotope cluster spacing of 1.0014, above the 1.001 u limit so can therefore be easily assigned to a group not containing S, Cl or Br. To be able to accurately distinguish a mass difference of 0.0004 u for a metabolite with a monoisotopic mass of 204.0634 u, as in this example, requires an inter-isotopic mass accuracy of approximately 2 ppm, however if a dynamic decision boundary is used the inter-isotopic mass accuracy required is reduced to close to 10 ppm.

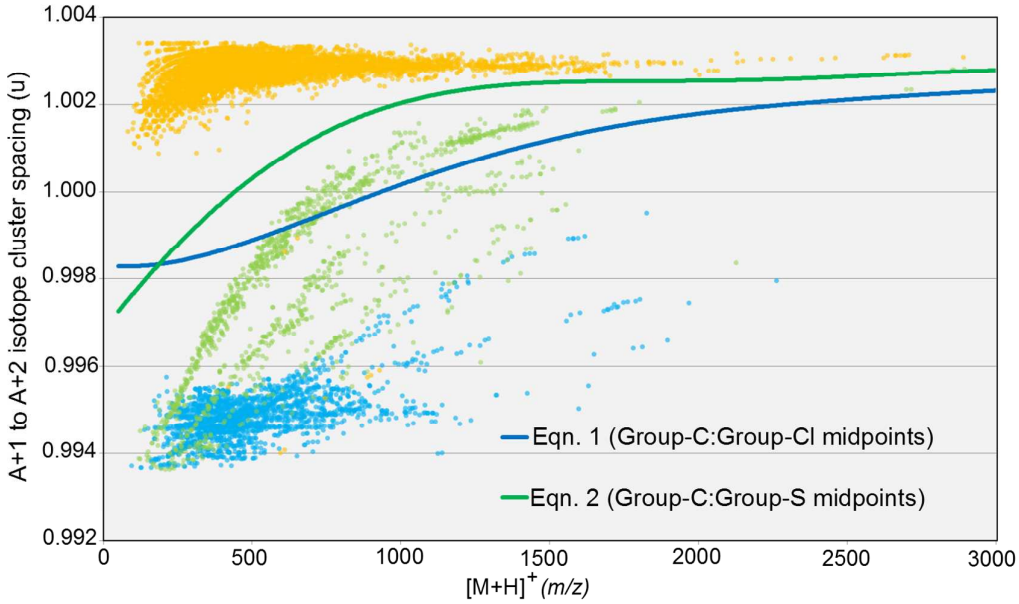


Figure S-2: *Illustrated Polynomial Midpoint Equations*

The polynomial midpoint equations derived from realistic hypothetical molecular formulas in comparison to the molecular formulas of the Marinlit database plotted as their isotopic properties *m/z* of $[M+H]^+$ on the x-axis, and A+1 to A+2 isotope cluster spacing on the y-axis, split into three groups: Group-Cl (blue), metabolites containing Cl and/or Br; Group-S (green), metabolites containing S and no Cl/Br; and Group-C (red), the remaining metabolites of the Marinlit database.

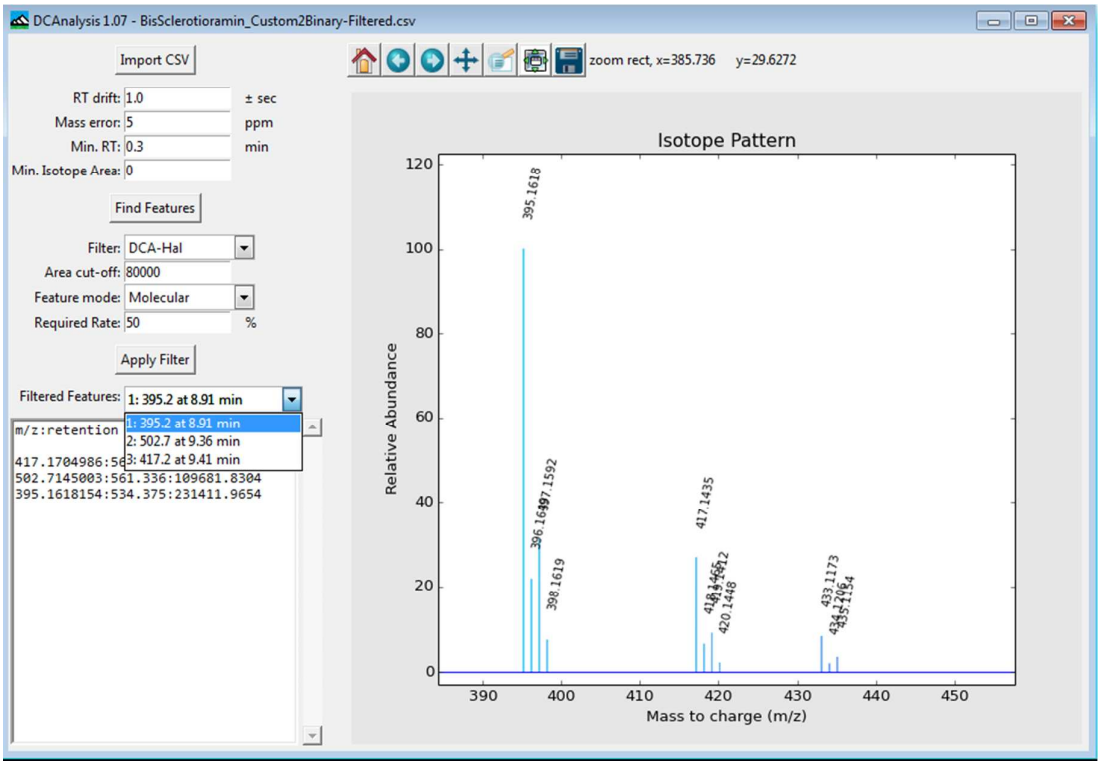


Figure S-3: *Graphical user interface of DCAnalysis v1.07*

A CSV containing isotopomer information from XCMS extraction is uploaded into the program. Default parameters, RT drift, Mass error, Min. RT, and Min. Isotope Area are set to 1.0 sec, 5 ppm, 0.3 min, and 0, respectively, for compiling chemical and molecular features, these values can be adjusted depending on the instrument used and to reduce processing time. The option for either DCA-Hal or DCA-Sul analysis are selected from a drop down menu, the minimum area of the most abundant isotope from a molecular feature can be set to remove molecular features with large noise influences, default 80000. Selection of filter by chemical or molecular features can be set from a drop down menu, and the required percent of chemical features identified by DCA-Hal or DCA-Sul in a molecular feature for a feature to be considered a correct identification. Results of DCA-Hal or DCA-Sul are selected from a drop down menu and the corresponding spectrum is displayed.

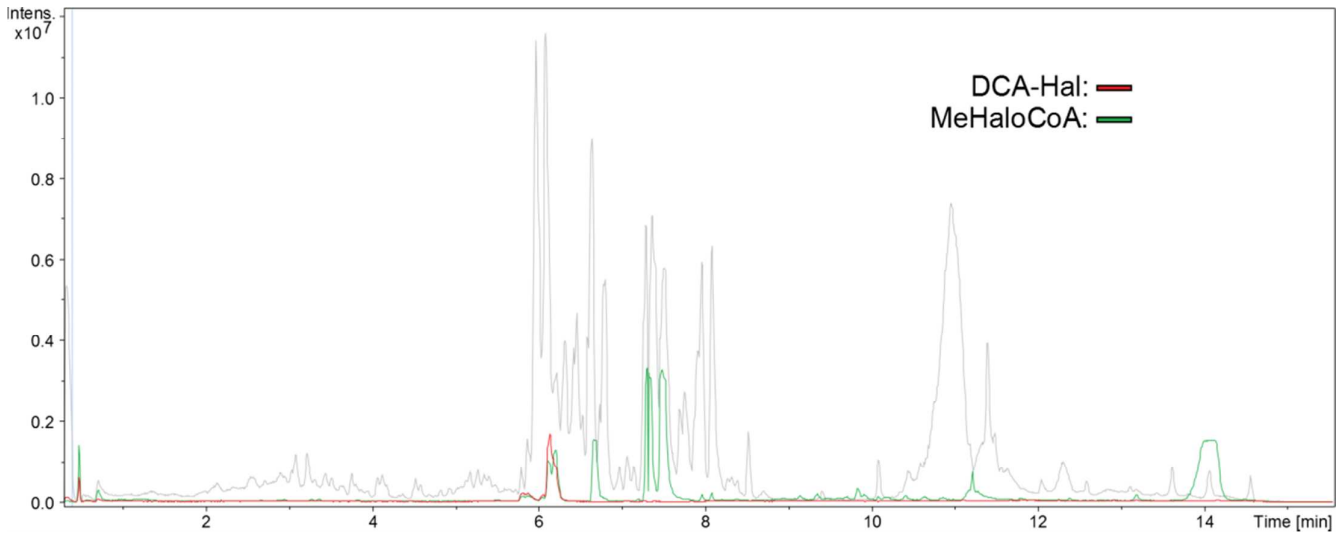
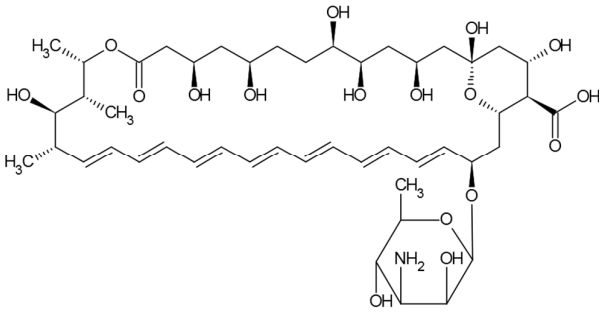
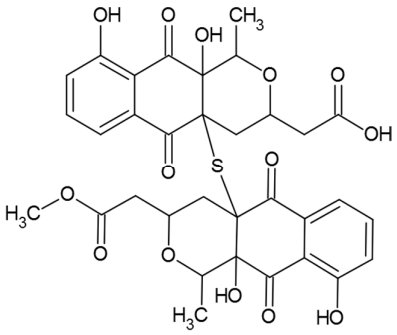
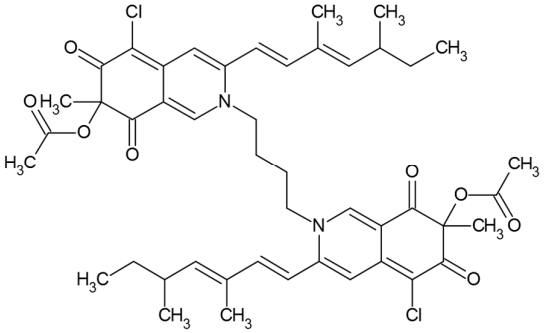
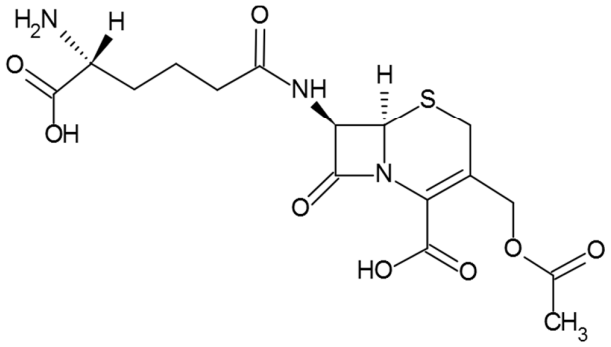
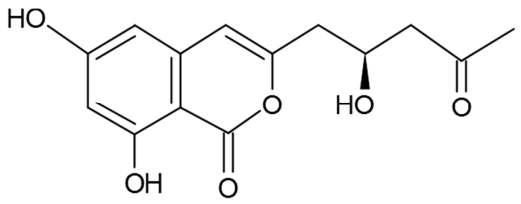
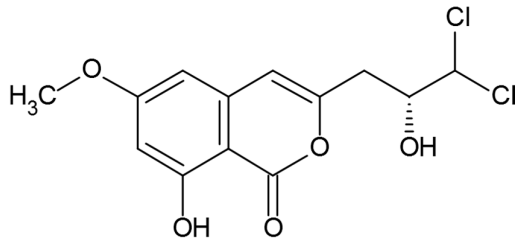
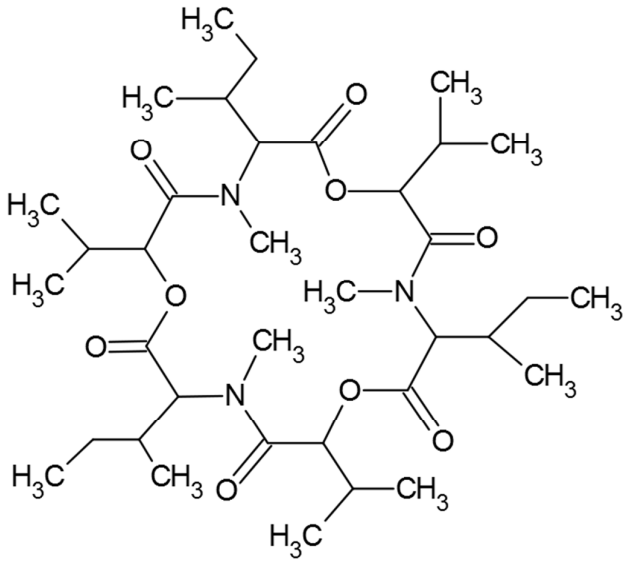
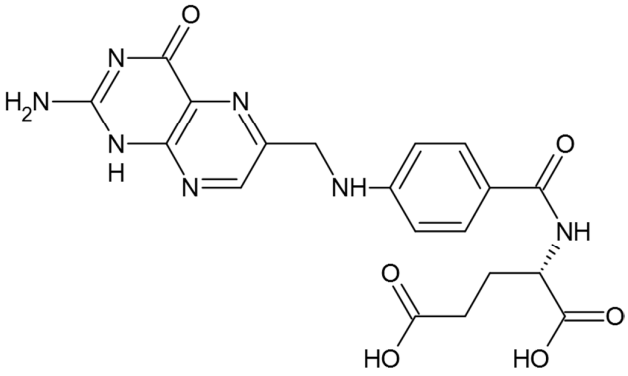
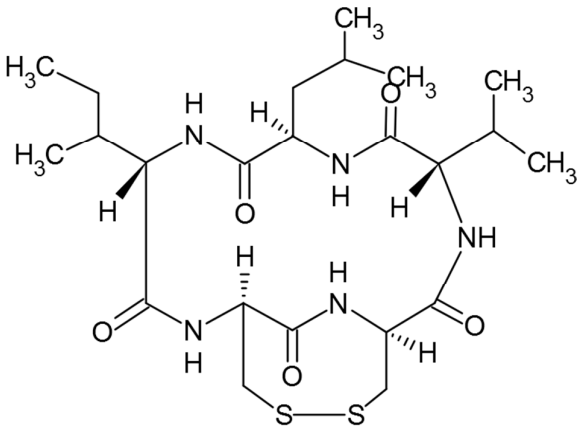
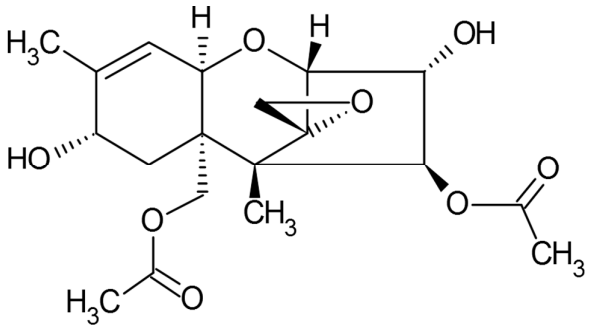
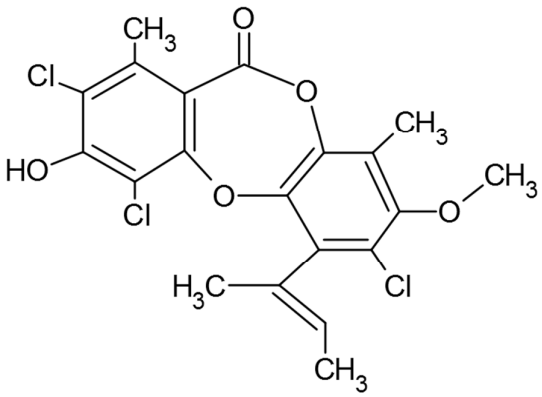


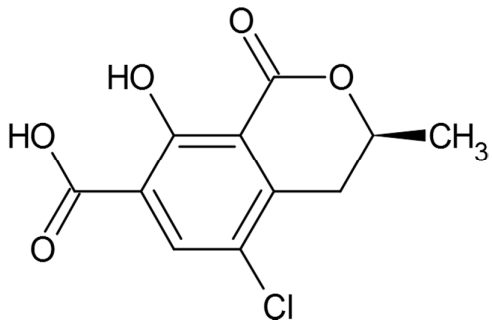
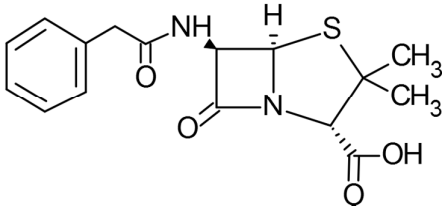
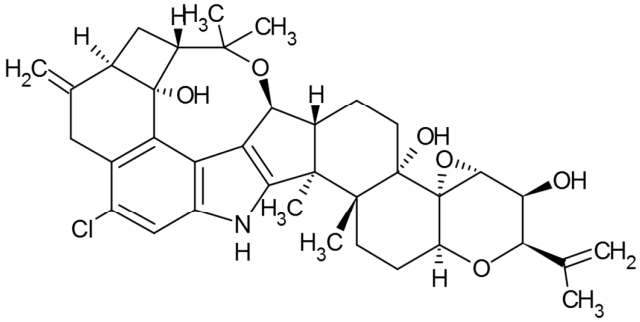
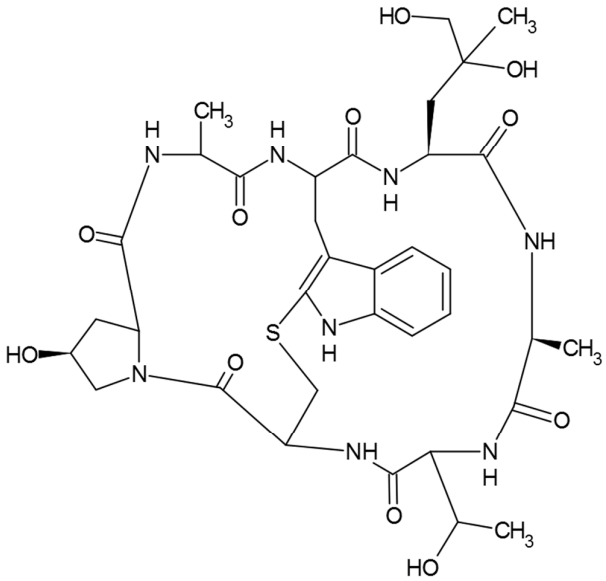
Figure S-4: *Comparison of ions found by MeHaloCoA and DCA-Hal as EIC*

The grey chromatogram is the EIC of all chemical features compiled by DCAnalysis v1.07. The green chromatogram is the EIC of all ions indentified by MeHaloCoA as being halogenated. The red chromatogram is the EIC of all chemical features identified by DCA-Hal as being halogenated. The EICs of the identified ions show significantly more identifications in the MeHaloCoA analysis. Inspection of the larger peaks in the MeHaloCoA chromatogram which are absent in the DCA-Hal analysis revieled that these ions were misidentified because of peak saturation, or were over 800 Da. All comparisons between DCA-Hal and MeHaloCoA were performed on the same XML data files and extracted using the same XCMS centwave settings. Default MeHaloCoA filtering settings were used in the comparisons.

Reference Standard	Molecular formula	*Mono isotopic mass* ($[M+H]^+$)	Structure
amphotericin B	$C_{47}H_{73}NO_{17}$	924.49513	
BE 52440A isomer	$C_{33}H_{32}O_{14}S$	685.15855	
bis-sclerotioramin	$C_{46}H_{54}Cl_2N_2O_8$	833.33300	
cephalosporin-C	$C_{16}H_{21}N_3O_8S$	416.11221	

Reference Standard	Molecular formula	*Mono isotopic mass* ([M+H]$^+$)	Structure
citreo-isocoumarin	$C_{14}H_{14}O_6$	279.086315	
dichlorodiaportin	$C_{13}H_{12}Cl_2O_5$	319.013455	
enniatin-A	$C_{36}H_{63}N_3O_9$	682.463707	
folic acid	$C_{19}H_{19}N_7O_6$	442.146958	

Reference Standard	Molecular formula	*Mono isotopic mass* ($[M+H]^+$)	Structure
malformin-A	$C_{23}H_{39}N_5O_5S_2$	530.246536	
neosolaniol	$C_{19}H_{26}O_8$	383.170044	
nidulin	$C_{20}H_{17}Cl_3O_5$	443.021433	

Reference Standard	Molecular formula	*Mono isotopic mass* ([M+H]$^+$)	Structure
ochratoxin alpha	$C_{11}H_9ClO_5$	257.021128	
penicillin-G	$C_{16}H_{18}N_2O_4S$	335.106003	
penitrem A	$C_{37}H_{44}ClNO_6$	634.292992	
phalloidin	$C_{35}H_{48}N_8O_{11}S$	789.323601	

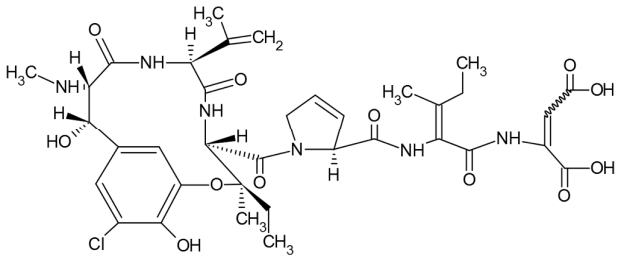
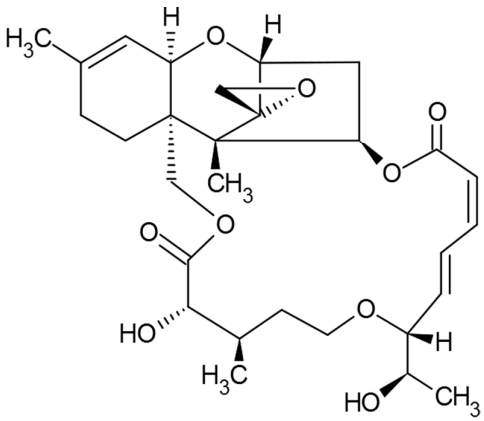
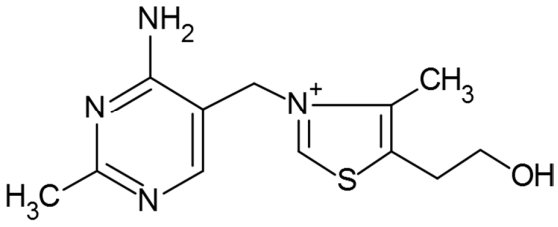
Reference Standard	Molecular formula	*Mono isotopic mass* ([M+H]$^+$)	Structure
phomopsin A	$C_{36}H_{45}ClN_6O_{12}$	789.285675	
roridin A	$C_{29}H_{40}O_9$	533.274509	
vitamin B1	$C_{12}H_{17}N_4OS$	265.11176*	

Table S-1: *Reference Starndard Molecular Formulas and Structures*

The molecular formulas, structures and m/z of [M+H]$^+$ for the 18 reference standards used for DCA trial. * = indicates monoisotopic mass of [M+]$^+$.

Species	Strain	Date	Location
P. parvum	K-0081	1985	Thy, Denmark
P. patelliferum	K-0252	1987	Victoria, Australia
P. patelliferum	K-0374	1989	Norway
P. parvum	KAC-39	-	Oslo fjord, Norway
P. parvum	N-1017	-	Japan
P. patelliferum	RCC-191	1976	S. Coast, England
P. parvum	RCC-1433	1981	English Channel
P. parvum	RCC-1435	-	La Réunion, France
P. parvum	RCC-1436	1977	France
P. parvum	UTEX-2797	2001	Texas, USA

Table S-2: *Strain Information of* P. parvum

K strains were obtained from the Scandinavian Culture Collection for Algae and Protozoa (Marine Biological Section, University of Copenhagen, Denmark). KAC-39 strain was obtained from the Kalmar Algal Collection (Linnaeus University, Sweden). N-1017 strain was obtained from the Microbial Culture Collection (National Institute for Environmental Studies, Japan). RCC strains were obtained from the Roscoff Culture Collection (France). UTEX-2797 strain was obtained from UTEX Culture Collection of Algae (University of Texas, Austin). *P. parvum* and *P. patelliferum* are now considered the same species and are both referred to as *P. parvum* throughout. All cultures were harvested at the beginning of the stationary phase. Cultures were maintained at 15 °C with an irradiance of 250 mmol photons $m^{-2}s^{-1}$ in f/2 media made with pasteurised seawater with a salinity of 30 PSU.

	Group-Cl	Group-S	Group-C
Group-Cl	100%	0%	0%
Group-S	2.8%	96%	1.2%
Group-C	0.1%	0.1%	99.9%

Table S-3: *DCA-Hal and DCA-Sul results from simulated isotope patterns of the Antibase*

Results of applying the Marinlit modelled decision boundaries to the molecular formulas of Antibase. 100% of Group-Cl was assigned correctly. 96.0% of Group-S was assigned correctly, 2.8% of Group-S were incorrectly assigned to Group-Cl, the metabolites incorrectly assigned to Group-Cl either had six or more S atoms, had monoisotopic ion masses above the theoretical limit for Group-S classification (1081 u), or contained a transition metal, zinc, copper, or nickel. The 1.2% of metabolites from Group-S that were incorrectly assigned to Group-C either had monoisotopic ion masses above the theoretical limit for Group-S classification or they contained iron. 99.9% of Group-C metabolites were assigned correctly. The 0.1% of metabolites from Group-C that were incorrectly assigned to Group-Cl all contained atypical elements with unusual isotopic patterns, specifically, Zn, Ni, Mg, Fe, Cu, or Se. The 0.1% of metabolites from Group-C that were incorrectly assigned to Group-S all contained Mg. Molecular formulas which were not detected by DCA-Hal and DCA-Sul were assigned to Group-C.

	Ag	Br	Ca	Cd	Ce	Cl	Cr	Cu
A+1-A+2:	0.99626	0.99455	0.99263	0.99800	1.0004	0.99365	0.99497	0.99479
A:A+2:	0.929	0.973	0.007	0.261	0.126	0.320	0.028	0.446
DCA Asignment:	Hal	Hal	Abundance too low	Sul	Spacing too small	Hal	Abundance too low	Hal
Similar to:	Br	Br	-	S_6	-	Cl	-	Cl
Resolving isotope/s:	None	Not Required	-	A-1, A-2, A-3, A-4	-	Not Required	-	None

	Er	Fe	Ga	Gd	Ge	Hg	K	Kr
A+1-A+2:	0.99969	0.99494	0.99573	0.99955	0.99682	0.99945	0.99472	0.99571
A:A+2:	0.553	0.003	0.664	0.880	0.213	0.230	0.072	0.303
DCA Asignment:	Hal	Abundance too low	Hal	Hal	Sul	Sul	Sul	Hal
Similar to:	Cl	-	Cl_2	Cl_2	S_5	S_5	S	Cl
Resolving isotope/s:	A-1, A-2	-	A+4 (Cl_2)	A-1, A-2, A-3, A-5	A-1, A-2, A-3, A-4	A-1, A-2, A-3, A-4	None	A-1, A-2, A-4

	Mg	Mo	Nd	Ne	Ni	O	Pd	Rb
A+1-A+2:	0.99415	0.99867	0.99896	0.99554	0.99204	1.00085	0.99701	0.99399
A:A+2:	0.139	0.401	0.875	0.102	0.385	0.002	0.968	0.386
DCA Asignment:	Sul	Hal	Hal	Sul	Hal	Abundance too low	Hal	Hal
Similar to:	S_3	Cl_2	Br	S_2	Cl	-	Br	Cl
Resolving isotope/s:	A+1	A-1, A-2, A-3, A-4, A-6	A+1, A+3, A+4, A+6, A+8	None	A+4	-	A-1, A-2, A+4	None

	Re	Ru	S	Sb	Se	Si	Sm	Sn
A+1-A+2:	0.99940	0.99768	0.99240	0.99700	0.99700	0.99344	0.99910	0.99784
A:A+2:	1.674	0.590	0.045	0.748	0.176	0.034	0.850	0.142
DCA Asignment:	Hal	Hal	Sul	Hal	Sul	Sul	Hal	Sul
Similar to:	Br_2	Cl_2	S	Cl_2	S_3	S	Cl_3	S_3
Resolving isotope/s:	A+4 (Br_2)	A-6, A-4, A-3, A-2, A-1	Not Required	A+4 (Cl_2)	A-4, A-3, A-2	None	A-8, A-5, A-4, A-3, A-2	A-4, A-3, A-2, A-1, A+4

	Ti	W	Xe	Yb	Zn	Zr
A+1-A+2:	0.99344	1.00003	0.99784	1.00031	0.99349	0.99694
A:A+2:	0.070	0.928	0.388	0.401	0.564	0.333
DCA Asignment:	Sul	Spacing too small	Hal	Spacing too small	Hal	Hal
Similar to:	S	-	Cl	-	Cl_2	Cl
Resolving isotope/s:	A-2, A-1, A+1	-	A-4, A-3, A-2, A-1, A+4	-	A+4	A+1, A+4

Table S-4: *Complete table of A+2 Elements*

Summary of likely DCA classification of A+2 elements and their approximate A+1 to A+2 isotope cluster spacing and A:A+2 intensity ratios for saturated compounds at 400 Da. Of the 38 A+2 elements, 7 (Ca, Ce, Cr, Fe, O, W and Yb) can not be detected by DCA because of either low abundance of the A+2 isotope, or because of a large A+1 to A+2 isotope spacing. Of the remaining 31 elements, 25 can be identified by either DCA-Hal, or DCA-Sul, and with the use of different isotopes and be distinguished from compounds containing S, Cl, or Br. The remaining 6 (Si, Rb, Ne, K, Cu, and Ag) can be detected by either DCA-Hal or DCA-Sul, however, they can not be simply distinguished from compounds containing S, Cl or Br with the use of adtional isotopes.

	BE 52440A isomer		bis-sclerotioramin		cephalosporin-C		dichlorodiaportin	
	$C_{33}H_{32}O_{14}S$		$C_{46}H_{54}Cl_2N_2O_8$		$C_{16}H_{21}N_3O_8S$		$C_{13}H_{12}Cl_2O_5$	
QTOF:	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent
Retention time (min):	5.63	8.40	9.41	13.09	0.65	2.52	6.44	9.28
Data processing time (sec):	1.3	2.0	5.2	1.7	3.5	1.9	4.8	4.1
Observed ion:	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+2H]^{2+}$**	**$[M+2H]^{2+}$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**
Observed *m/z*:	685.1600	685.1595	417.1705	417.1718	416.1126	416.1139	319.0138	319.0145
Predicted *m/z*:	685.1586	685.1586	417.1701	417.1701	416.1122	416.1122	319.0135	319.0135
error (ppm):	2.1	1.4	0.9	4.0	0.9	4.1	1.1	3.3
Observed ion:	**$[M+NH_4]^+$**	**$[M+NH_4]^+$**	**$[M+H]^+$**	**$[M+H]^+$**			**$[M+Na]^+$**	**$[M+Na]^+$**
Observed *m/z*:	702.1863	702.1858	833.3323	833.3359			340.9954	340.9957
Predicted *m/z*:	702.1851	702.1851	833.3330	833.3330			340.9954	340.9954
error (ppm):	1.7	1.0	0.8	3.5			0.0	0.9
Observed ion:	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**			**$[M+K]^+$**	**$[M+K]^+$**
Observed *m/z*:	707.1410	707.1410	855.3137	855.3163			356.9693	356.9695
Predicted *m/z*:	707.1405	707.1405	855.3149	855.3149			356.9693	356.9693
error (ppm):	0.7	0.7	1.4	1.6			0.1	0.4
Observed ion:		**$[2M+H]^+$**	**$[M+K]^+$**				**$[2M+Na]^+$**	**$[2M+Na]^+$**
Observed *m/z*:		1369.3104	871.2877				659.0005	659.0015
Predicted *m/z*:		1369.3098	871.2889				659.0016	659.0016
error (ppm):		0.4	1.4				1.6	0.1
Observed ion:		**$[2M+NH_4]^+$**					**$[2M+K]^+$**	
Observed *m/z*:		1386.3365					674.9722	
Predicted *m/z*:		1386.3364					674.9755	
error (ppm):		0.1					4.9	
Observed ion:		**$[2M+Na]^+$**					**$[4M+K+H]^{2+}$**	

	BE 52440A isomer		bis-sclerotioramin		cephalosporin-C		dichlorodiaportin	
	$C_{33}H_{32}O_{14}S$		$C_{46}H_{54}Cl_2N_2O_8$		$C_{16}H_{21}N_3O_8S$		$C_{13}H_{12}Cl_2O_5$	
Observed *m/z*:		1391.2936					655.9930	
Predicted *m/z*:		1391.2918					655.9976	
error (ppm):		1.3					7.0	
Observed ion:		**$[2M+K]^+$**						
Observed *m/z*:		1407.2629						
Predicted *m/z*:		1407.2657						
error (ppm):		2.0						
Total number of chemical features:	3	7	4	3	1	1	6	4
Percent of CF correctly assigned:	100	100	100	100	100	100	83	100
Average error (ppm):	1.5	1.0	1.1	3.0	0.9	4.1	2.5	1.2

	malformin-A		nidulin		ochratoxin-alpha		penicillin-G	
	$C_{23}H_{39}N_5O_5S_2$		$C_{20}H_{17}Cl_3O_5$		$C_{11}H_9ClO_5$		$C_{16}H_{18}N_2O_4S$	
QTOF:	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent
Retention time (min):	6.12	8.76	9.41	12.82	4.18	6.15	4.80	7.00
Data processing time (sec):	2.2	1.5	10.6	4.3	2.6	1.2	2.5	1.3
Observed ion:	**$[M+H]^+$**	**$[M+H]^+$**	**$[M-HCl+H]^+$**		**$[M-H_2O+H]^+$**	**$[M-H_2O+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**
Observed *m/z*:	530.2467	530.2478	407.0442		239.0108	239.0108	335.1061	335.1061
Predicted *m/z*:	530.2465	530.2465	407.0448		239.0106	239.0106	335.1060	335.1060
error (ppm):	0.3	2.4	1.4		1.0	1.0	0.3	0.3
Observed ion:	**$[M+NH_4]^+$**	**$[M+NH_4]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+NH_4]^+$**	
Observed *m/z*:	547.2727	547.2734	443.0218	443.0220	257.0214	257.0218	352.1322	
Predicted *m/z*:	547.2731	547.2731	443.0214	443.0214	257.0211	257.0211	352.1326	

	malformin-A		nidulin		ochratoxin-alpha		penicillin-G	
	$C_{23}H_{39}N_5O_5S_2$		$C_{20}H_{17}Cl_3O_5$		$C_{11}H_9ClO_5$		$C_{16}H_{18}N_2O_4S$	
error (ppm):	0.7	0.6	0.8	1.3	1.1	2.6	1.0	
Observed ion:	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+NH_4]^+$**	**$[M+NH_4]^+$**	**$[M-H_2O+Na]^+$**		**$[M+Na]^+$**	**$[M+Na]^+$**
Observed *m/z*:	552.2285	552.2310	460.0474	460.0479	260.9924		357.0879	357.0882
Predicted *m/z*:	552.2285	552.2285	460.0480	460.0480	260.9925		357.0880	357.0880
error (ppm):	0.0	4.6	1.3	0.2	0.4		0.1	0.7
Observed ion:	**$[M+K]^+$**	**$[M+K]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+NH_4]^+$**		**$[M+K]^+$**	**$[M+K]^+$**
Observed *m/z*:	568.2016	568.2024	465.0028	465.0040	274.0475		373.0616	373.0615
Predicted *m/z*:	568.2024	568.2024	465.0034	465.0034	274.0477		373.0619	373.0619
error (ppm):	1.4	0.0	1.2	1.3	0.7		0.8	1.0
Observed ion:	**$[2M+H]^+$**	**$[2M+H]^+$**		**$[M+K]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[2M+H]^+$**	
Observed *m/z*:	1059.4854	1059.4871		480.9775	279.0033	279.0033	669.2041	
Predicted *m/z*:	1059.4858	1059.4858		480.9773	279.0031	279.0031	669.2047	
error (ppm):	0.4	1.2		0.4	0.8	0.8	0.9	
Observed ion:		**$[2M+NH_4]^+$**	**$[M+H_2O+Na]^+$**		**$[M+K]^+$**		**$[2M+NH_4]^+$**	
Observed *m/z*:		1076.5135	483.0130		294.9769		686.2308	
Predicted *m/z*:		1076.5124	483.0139		294.9770		686.2313	
error (ppm):		1.1	1.9		0.4		0.7	
Observed ion:	**$[2M+Na]^+$**	**$[2M+Na]^+$**	**$[2M+Na]^+$**	**$[2M+Na]^+$**	**$[M+H_2O+Na]^+$**		**$[2M+Na]^+$**	**$[2M+Na]^+$**
Observed *m/z*:	1081.4677	1081.4699	907.0164	907.0188	297.0131		691.1867	691.1869
Predicted *m/z*:	1081.4677	1081.4677	907.0175	907.0175	297.0136		691.1867	691.1867
error (ppm):	0.0	2.0	1.2	1.4	1.8		0.0	0.3
Observed ion:	**$[2M+K]^+$**	**$[2M+K]^+$**				**$[M-H+2Na]^+$**		
Observed *m/z*:	1097.4405	1097.4422				300.9851		
Predicted *m/z*:	1097.4417	1097.4417				300.9850		

	malformin-A		nidulin		ochratoxin-alpha		penicillin-G	
	$C_{23}H_{39}N_5O_5S_2$		$C_{20}H_{17}Cl_3O_5$		$C_{11}H_9ClO_5$		$C_{16}H_{18}N_2O_4S$	
error (ppm):	1.1	0.5				0.3		
Observed ion:					**$[2M+Na]^+$**	**$[2M+Na]^+$**		
Observed *m/z*:					535.0169	535.0168		
Predicted *m/z*:					535.0169	535.0169		
error (ppm):					0.0	0.2		
Observed ion:						**$[2M-H+2Na]^+$**		
Observed *m/z*:						556.9985		
Predicted *m/z*:						556.9989		
error (ppm):						0.7		
Observed ion:						**$[2M-2H+3Na]^+$**		
Observed *m/z*:						578.9802		
Predicted *m/z*:						578.9808		
error (ppm):						1.1		
Observed ion:						**$[4M-4H+5Na]^+$**		
Observed *m/z*:						1134.9687		
Predicted *m/z*:						1134.9724		
error (ppm):						3.3		
Total number of chemical features:	7	8	5	5	8	8	7	4
Percent of CF correctly assigned:	86	87.5	100	100	75	100	100	100
Average error (ppm):	0.6	1.5	1.3	0.9	0.8	1.2	0.6	0.6

	penitrem A	phalloidin	phomopsin A	vitamin B1
	$C_{37}H_{44}ClNO_6$	$C_{35}H_{48}N_8O_{11}S$	$C_{36}H_{45}ClN_6O_{12}$	$C_{12}H_{17}N_4OS$

	penitrem A		phalloidin		phomopsin A		vitamin B1	
	$C_{37}H_{44}ClNO_6$		$C_{35}H_{48}N_8O_{11}S$		$C_{36}H_{45}ClN_6O_{12}$		$C_{12}H_{17}N_4OS$	
QTOF:	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent
Retention time (min):	8.51	12.11	3.46	4.94	3.65	5.24	0.50	1.48
Data processing time (sec):	3.1	1.9	2.4	1.2	3.4	2.1	2.7	0.7
Observed ion:	**$[M-H_2O+H]^+$**	**$[M-H_2O+H]^+$**	**$[M-H_2O+2H]^{2+}$**		**$[M-H_2O+K+2H]^{3+}$**		**$[M-H_2O+H]^{2+}$**	
Observed *m/z*:	616.2812	616.2824	386.1597		270.4119		124.0542	
Predicted *m/z*:	616.2824	616.2824	386.1602		270.4152		124.0542	
error (ppm):	2.0	0.0	1.2		12.1		0.2	
Observed ion:	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+2H]^{2+}$**		**$[M+K+2H]^{3+}$**		**$[M+H]^{2+}$**	
Observed *m/z*:	634.2926	634.2935	395.1652		276.4155		133.0596	
Predicted *m/z*:	634.2930	634.2930	395.1654		276.4187		133.0595	
error (ppm):	0.6	0.8	0.6		11.6		0.6	
Observed ion:	**$[M+Na]^+$**		**$[M+Na+H]^{2+}$**		**$[M+H_2O+K+2H]^{3+}$**		**$[M]^+$**	**$[M]^+$**
Observed *m/z*:	656.2730		406.1555		282.4188		265.1118	265.1125
Predicted *m/z*:	656.2749		406.1564		282.4222		265.1118	265.1118
error (ppm):	3.0		2.2		12.1		0.2	2.8
Observed ion:	**$[M+H_2O+Na]^+$**	**$[M+H_2O+Na]^+$**	**$[M+2Na]^{2+}$**		**$[M-H_2O+2H]^{2+}$**			
Observed *m/z*:	674.2833	674.2853	417.1469		386.1406			
Predicted *m/z*:	674.2855	674.2855	417.1474		386.1412			
error (ppm):	3.3	0.3	1.2		1.5			
Observed ion:		**$[2M+Na]^+$**	**$[M-H+Fe(III)]^{2+}$**		**$[M+2H]^{2+}$**	**$[M+2H]^{2+}$**		
Observed *m/z*:		1289.5616	420.6229		395.1460	395.1469		
Predicted *m/z*:		1289.56065	420.6235		395.1465	395.1465		
error (ppm):		0.7	1.5		1.2	1.1		
Observed ion:			**$[M+H]^+$**	**$[M+H]^+$**	**$[M+Na+H]^{2+}$**	**$[M+Na+H]^{2+}$**		

	penitrem A	phalloidin		phomopsin A		vitamin B1
	$C_{37}H_{44}ClNO_6$	$C_{35}H_{48}N_8O_{11}S$		$C_{36}H_{45}ClN_6O_{12}$		$C_{12}H_{17}N_4OS$
Observed *m/z*:		789.3239	789.3280	406.1370	406.1352	
Predicted *m/z*:		789.3236	789.3236	406.1375	406.1375	
error (ppm):		0.4	5.6	1.1	5.5	
Observed ion:		**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+Xa]^{2+}$**		
Observed *m/z*:		811.3052	811.3069	413.1445		
Predicted *m/z*:		811.3056	811.3056			
error (ppm):		0.4	1.7			
Observed ion:		**$[M+K]^+$**		**$[M+2Na]^{2+}$**		
Observed *m/z*:		827.2779		417.1276		
Predicted *m/z*:		827.2795		417.1284		
error (ppm):		1.9		2.0		
Observed ion:		**$[2M+Na+H]^{2+}$**		**$[M+Xb]^{2+}$**		
Observed *m/z*:		800.3082		419.6045		
Predicted *m/z*:		800.3146				
error (ppm):		8.0				
Observed ion:		**$[2M+K+H]^{2+}$**		**$[M+Xc]^{2+}$**		
Observed *m/z*:		808.2969		426.1065		
Predicted *m/z*:		808.3015				
error (ppm):		5.7				
Observed ion:				**$[M+H]^+$**	**$[M+H]^+$**	
Observed *m/z*:				789.2867	789.2870	
Predicted *m/z*:				789.2857	789.2857	
error (ppm):				1.3	1.7	
Observed ion:				**$[M+Xd]^+$**		

	penitrem A		phalloidin		phomopsin A		vitamin B1	
	$C_{37}H_{44}ClNO_6$		$C_{35}H_{48}N_8O_{11}S$		$C_{36}H_{45}ClN_6O_{12}$		$C_{12}H_{17}N_4OS$	
Observed *m/z*:					803.3006			
Predicted *m/z*:								
error (ppm):								
Observed ion:					**$[M+Na]^+$**	**$[M+Na]^+$**		
Observed *m/z*:					811.2666	811.2676		
Predicted *m/z*:					811.2676	811.2676		
error (ppm):					1.3	0.0		
Observed ion:					**$[2M+K+H]^{2+}$**			
Observed *m/z*:					808.2591			
Predicted *m/z*:					808.2636			
error (ppm):					5.6			
Observed ion:						**$[M-H+2Na]^+$**		
Observed *m/z*:						833.2489		
Predicted *m/z*:						833.2496		
error (ppm):						0.8		
Total number of chemical features:	4	4	10	2	14	5	3	1
Percent of CF correctly assigned:	75	100	60	100	71	80	100	100
Average error (ppm):	2.2	0.5	2.3	3.6	5.0	1.8	0.3	2.8

Table S-5: *DCA-Hal and DCA-Sul Reference Standards Results*

Results from DCA-Hal and DCA-Sul analysis of reference standards in Group-Cl and Group-S. Xa-d denotes an adduct which is unknown.

amphotericin B ($C_{47}H_{73}NO_{17}$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	946.4769	1	-	FP

Table S-6a: *DCA-Hal comparison to MeHaloCoA: amphotericin B Results*

Analysis of amphotericin B with MeHaloCoA resulted in 1 false positives.

BE 52440A isomer ($C_{33}H_{32}O_{14}S$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	1341.3520	1	-	FP
				1363.3350	1	-	FP
				1377.3130	1	FP	-
				1395.2690	1	-	FP

Table S-6b: *DCA-Hal comparison to MeHaloCoA: BE 52440A isomer Results*

Analysis of BE 52440A isomer with MeHaloCoA resulted in 3 false positives, and analysis with DCA-Hal resulted in 1 false positive. The false positive of *m/z* 1377.3130 of DCA-Hal was caused by a misclassification of the chemical feature as being halogenated.

bis-sclerotioramin ($C_{46}H_{54}Cl_2N_2O_8$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
833.3323	1	x	x	833.3359	1	x	x
855.3137	1	x	x	855.3163	1	x	x
871.2877	1	x	x	923.3029	1	x	x

Table S-6c: *DCA-Hal Comparison to MeHaloCoA: bis-sclerotioramin Results*

Analysis of bis-sclerotioramin resulted in no false negative.

cephalosporin-C ($C_{16}H_{21}N_3O_8S$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	-	-	-	-

Table S-6d: *DCA-Hal comparison to MeHaloCoA: cephalosporin-C Results*

Analysis of cephalosporin-C resulted in no false positives.

citreo isocumarin ($C_{14}H_{14}O_6$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	-	-	-	-

Table S-6e: *DCA-Hal comparison to MeHaloCoA: citreo isocumarin Results*

Analysis of citreo isocumarin resulted in no false positives.

dichlorodiaportin ($C_{13}H_{12}Cl_2O_5$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
319.0138	1	x	x	319.0145	1	x	x
340.9958	1	FN	x	340.9957	1	x	x
356.9693	1	x	x	356.9695	1	x	x
659.0005	1	x	FN	659.0015	1	x	x
689.9313	1	x	x	691.9294	1	x	x

Table S-6f: *DCA-Hal comparison to MeHaloCoA: dichlorodiaportin Results*

Analysis of dichlorodiaportin with MeHaloCoA resulted in 1 false negative, and analysis with DCA-Hal resulted in 1 false negative. The false negative *m/z* 340.9958 of DCA-Hal was caused by the compiling of a doubly charged dimer into the chemical formula of the singly charge species.

enniatin-A ($C_{36}H_{63}N_3O_9$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	-	-	-	-

Table S-6g: *DCA-Hal comparison to MeHaloCoA: enniatin-A Results*

Analysis of enniatin-A resulted in no flase positives.

folic acid ($C_{19}H_{19}N_7O_6$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	-	-	-	-

Table S-6h: *DCA-Hal comparison to MeHaloCoA: folic acid Results*

Analysis of folic acid resulted in no flase positives.

malformin-A ($C_{23}H_{39}N_5O_5S_2$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
1081.4680	1	-	FP	568.2020	1	-	FP
1097.4405	1	FP	-	1059.4870	1	-	FP
				1076.5140	1	-	FP
				1081.4700	1	-	FP
				1097.4420	1	FP	FP
				1139.4260	1	FP	FP
				1143.4360	1	-	FP
				1149.4560	1	-	FP
				1165.4290	1	-	FP
				1217.4430	1	-	FP

Table S-6i: *DCA-Hal comparison to MeHaloCoA: malformin-A Results*

Analysis of malformin-A with MeHaloCoA resulted in 11 false positives, and analysis with DCA-Hal resulted in 3 false positives. The false positives of *m/z* 1097.4405, 1097.4420 and 1139.4260 for DCA-Hal were due to the misclassification of these chemical features as being halogenated.

neosolaniol ($C_{19}H_{26}O_8$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
790.3223*	1	-	FP^{A+3}	-	-	-	-
804.2901*	1	-	FP^{A+2}				
809.2963	1	-	FP				
839.2547	1	FP	FP				
849.2834	1	-	FP				
855.2269	1	-	FP				

Table S-6j: *DCA-Hal comparison to MeHaloCoA: neosolaniol Results*

Analysis of neosolaniol with MeHaloCoA resulted in 6 false positives, and analysis with DCA-Hal resulted in 1 false positive. In addition, MeHaloCoA did not detect the A isotopomer of 2 chemical features (indicated with *), rather the identification was performed on the A+2, or A+3 isotopomers. The false positive of *m/z* 839.2547 for DCA-Hal was due to the misclassification of this chemical features as being halogenated.

nidulin ($C_{20}H_{17}Cl_3O_5$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
407.0442	1	x	x	371.1012	1	FN	x
443.0218	1	x	x	443.0220	1	x	x
460.0474	1	x	x	460.0479	1	x	FN
465.0028	1	x	FN	465.0040	1	x	x
483.0130	1	x	FN	480.9775	1	x	FN
907.0164*	1	x	x^{A+2}	519.1387	1	x	x
				907.0188	1	x	x
				1349.0320	1	x	x

Table S-6k: *DCA-Hal comparison to MeHaloCoA: nidulin Results*

Analysis of nidulin with MeHaloCoA resulted in 4 false negatives, and analysis with DCA-Hal resulted in 1 false negative. In addition, MeHaloCoA did not detect the A isotopomer of 1 chemical feature (indicated with *), rather the identification was performed on the A+2 isotopomer. The false negative of *m/z* 371.1012 for DCA-Hal was due to a chemical feature compiling error which missed the A+2 isotopomer.

ochratoxin alpha ($C_{11}H_9ClO_5$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
239.0108	1	FN	x	239.0108	1	x	x
257.0214	1	x	x	257.0218	1	x	x
260.9924	1	x	x	279.0033	1	x	x
274.0475	1	x	FN	300.9851	1	x	x
279.0033	1	FN	x	392.9929	1	x	FN
294.9769	1	x	x	535.0168	1	x	x
535.0169	1	x	x	556.9985	1	x	FN
556.9985	1	x	x	565.9464*	1	x	x^{A+1}
566.9536	1	FN	x	578.9802	1	x	x
583.9566	1	x	x	793.0071	1	x	x
806.9952	1	x	x	806.995	1	x	FN
821.9602	1	x	x	821.9604	1	x	x
844.9431*	1	x	X^{A+1}	834.9928*	1	x	x^{A+2}
				856.9759	1	x	x
				1134.969	1	x	FN

Table S-6l: *DCA-Hal comparison to MeHaloCoA: ochratoxin alpha Results*

Analysis of ochratoxin alpha with MeHaloCoA resulted in 5 false negatives, and analysis with DCA-Hal resulted in 3 false negatives. In addition, MeHaloCoA did not detect the A isotopomer of 3 chemical features (indicated with *), rather the identification was performed on the A+1, or A+2 isotopomer. The false negatives of *m/z* 239.0108 and 279.0033 for DCA-Hal were due to the misclassification of these chemical features as containing sulfur. The false negative of *m/z* 566.9536 for DCA-Hal was due to a chemical feature compiling error.

penicillin-G ($C_{16}H_{18}N_2O_4S$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	-	-	-	-

Table S-6m: *DCA-Hal comparison to MeHaloCoA: penicillin-G Results*

Analysis of penicillin-G resulted in no false positives.

penitrem A ($C_{37}H_{44}ClNO_6$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
558.2391	1	x	x	614.3109	1	FN	x
616.2812	1	x	x	616.2824	1	x	x
634.2926	1	FN	x	634.2935	1	x	x
656.273	1	x	x	664.3114	1	FN	x
674.2833	1	x	FN	666.2837	1	x	x
				674.2853	1	x	x
				688.2643	1	x	FN
				1267.578	1	FN	x
				1289.562	1	x	x
				1307.563*	1	FN	x^{A+2}
				1357.548	1	x	x
				1425.534	1	x	x

Table S-6n: *DCA-Hal comparison to MeHaloCoA: penitrem A Results*

Analysis of penitrem A with MeHaloCoA resulted in 2 false negatives, and analysis with DCA-Hal resulted in 5 false negatives. In addition, MeHaloCoA did not detect the A isotopomer of 1 chemical feature (indicated with *), rather the identification was performed on the A+2 isotopomer. Also, DCA-Hal did not detect the A isotopomer of 1 chemical feature (indicated with #), rather the identification was performed on the A+2 isotopomer. The false negative of *m/z* 634.2926 for DCA-Hal was due to the misclassification of this chemical features as containing sulfur. The false negative of *m/z* 614.3109, 664.3114, 1267.5780, 1307.5630 for DCA-Hal were due to a chemical feature compiling errors, missing the A or A+1 isotopomers.

phalloidin ($C_{35}H_{48}N_8O_{11}S$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	789.3280	1	-	FP
				814.3099	1	-	FP
				1577.642	1	-	FP
				1594.665	1	-	FP

Table S-6o: *DCA-Hal comparison to MeHaloCoA: phalloidin Results*

Analysis of phalloidin with MeHaloCoA resulted in 4 false positives.

phomopsin A ($C_{36}H_{45}ClN_6O_{12}$)							
Bruker				Agilent			
CF (m/z)	Charge	DCA	MeHaloCoA	CF (m/z)	Charge	DCA	MeHaloCoA
519.1996	1	x	x	658.2635	1	x	x
658.2633	1	FN	x	789.2870	1	x	x
789.2867	1	FN	x	803.3012	1	x	x
803.3004	1	x	x	811.2676	1	x	x
811.2666	1	x	x	833.2489	1	x	x

Table S-6p: *DCA-Hal comparison to MeHaloCoA: phomopsin A Results*

Analysis of phomopsin A with MeHaloCoA resulted in no false negatives, and analysis with DCA-Hal resulted in 2 false negatives. The false negatives of *m/z* 658.2633 and 789.2867 for DCA-Hal were due to the misclassification of these chemical features as containing sulfur.

roridin A ($C_{29}H_{40}O_9$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	-	-	-	-

Table S-6q: *DCA-Hal comparison to MeHaloCoA: roridin A Results*

Analysis of roridin A resulted in no false positives.

vitamin B1 ($C_{12}H_{17}N_4OS$)							
Bruker				Agilent			
CF (*m/z*)	Charge	DCA	MeHaloCoA	CF (*m/z*)	Charge	DCA	MeHaloCoA
-	-	-	-	565.1924	1	FP	FP

Table S-6r: *DCA-Hal comparison to MeHaloCoA: vitamin B1 Results*

Analysis of vitamin B1 with MeHaloCoA resulted in 1 false positive, and analysis with DCA-Hal resulted in 1 false positive. The false positive of *m/z* 565.1924 for DCA-Hal was due to the misclassification of this chemical feature as being halogenated.

										A+1 to A+2		A to A+2	
	QTOF		Theoretical (*m/z*)	LCMS (*m/z*)	XCMS (*m/z*)	LCMS-Theoretical (u)	LCMS-Theoretical (± ppm)	LCMS-XCMS (u)	LCMS-XCMS (± ppm)	LCMS-Theoretical (± ppm)	LCMS-XCMS (± ppm)	LCMS-Theoretical (± ppm)	LCMS-XCMS (± ppm)
amphotericin B		**A:**	924.4951	924.4953	924.4952	0.0002	0.1	0.0001	0.1				
$C_{47}H_{73}NO_{17}$	**Bruker:**	**A+1:**	925.4985	925.4983	925.4982	0.0002	0.1	0.0001	0.0	0.3	0.1	0.4	0.0
		A+2:	926.5014	926.5007	926.5005	0.0007	0.4	0.0002	0.1				
		A:	924.4951	924.4955	924.4958	0.0004	0.2	0.0003	0.2				
	Agilent:	**A+1:**	925.4985	925.4989	925.4994	0.0004	0.2	0.0005	0.3	0.0	0.1	0.0	0.0
		A+2:	926.5014	926.5017	926.5019	0.0003	0.2	0.0002	0.1				
BE 52440A isomer		**A:**	671.1793	671.1801	671.1807	0.0008	0.6	0.0006	0.5				
$C_{33}H_{34}O_{13}S$	**Bruker:**	**A+1:**	672.1826	672.1838	672.1837	0.0012	0.9	0.0001	0.1	0.1	0.0	0.2	0.5
		A+2:	673.1820	673.1830	673.1829	0.0010	0.8	0.0001	0.1				
		A:	671.1793	671.1802	671.1805	0.0009	0.7	0.0003	0.2				
	Agilent:	**A+1:**	672.1826	672.1833	672.1835	0.0007	0.5	0.0002	0.2	0.5	0.2	0.4	0.1
		A+2:	673.1820	673.1834	673.1838	0.0014	1.1	0.0004	0.3				
bis-sclerotioramin		**A:**	417.1701	417.1706	417.1705	0.0005	0.6	0.0001	0.1				
$C_{46}H_{54}Cl_2N_2O_8$	**Bruker:**	**A+1:**	417.6718	417.6724	417.6723	0.0006	0.7	0.0001	0.1	0.3	0.0	0.1	0.0
		A+2:	418.1695	418.1699	418.1698	0.0004	0.4	0.0001	0.1				
		A:	833.3330	833.3359	833.3359	0.0029	1.7	0.0000	0.0				
	Agilent:	**A+1:**	834.3363	834.3383	834.3383	0.0020	1.2	0.0000	0.0	0.2	0.1	0.3	0.1
		A+2:	835.3318	835.3341	835.3343	0.0023	1.4	0.0002	0.1				
cephalosporin-C		**A:**	416.1122	416.1126	416.1126	0.0004	0.5	0.0000	0.0				
$C_{16}H_{21}N_3O_8S$	**Bruker:**	**A+1:**	417.1151	417.1152	417.1152	0.0001	0.1	0.0000	0.0	0.8	0.0	1.1	0.0
		A+2:	418.1120	418.1115	418.1115	0.0005	0.6	0.0000	0.0				
	Agilent:	**A:**	416.1122	416.1142	416.1151	0.0020	2.4	0.0009	1.1	0.9	0.8	1.0	1.5
		A+1:	417.1151	417.1155	417.1159	0.0004	0.5	0.0004	0.5				

	QTOF		Theoretical (*m/z*)	LCMS (*m/z*)	XCMS (*m/z*)	LCMS-Theoretical (u)	LCMS-Theoretical (± ppm)	LCMS-XCMS (u)	LCMS-XCMS (± ppm)	A+1 to A+2 LCMS-Theoretical (± ppm)	A+1 to A+2 LCMS-XCMS (± ppm)	A to A+2 LCMS-Theoretical (± ppm)	A to A+2 LCMS-XCMS (± ppm)
		A+2:	418.1120	418.1132	418.1129	0.0012	1.4	0.0003	0.3				
citreo-isocoumarin		**A:**	279.0863	279.0862	279.0862	0.0001	0.2	0.0000	0.0				
$C_{14}H_{14}O_6$	**Bruker:**	**A+1:**	280.0897	280.0904	280.0904	0.0007	1.2	0.0000	0.0	6.6	0.0	8.1	0.1
		A+2:	281.0918	281.0962	281.0962	0.0044	7.9	0.0000	0.1				
		A:	279.0863	279.0868	279.0869	0.0005	0.9	0.0001	0.2				
	Agilent:	**A+1:**	280.0897	280.0910	280.0910	0.0013	2.3	0.0000	0.1	6.8	0.5	8.2	0.5
		A+2:	281.0918	281.0969	281.0967	0.0051	9.1	0.0002	0.4				
dichlorodiaportin		**A:**	319.0135	319.0139	319.0138	0.0005	0.7	0.0001	0.1				
$C_{13}H_{12}Cl_2O_5$	**Bruker:**	**A+1:**	320.0169	320.0170	320.0170	0.0002	0.2	0.0000	0.0	0.3	0.1	0.2	0.0
		A+2:	321.0108	321.0111	321.0110	0.0003	0.5	0.0001	0.1				
		A:	319.0135	319.0144	319.0145	0.0010	1.5	0.0001	0.1				
	Agilent:	**A+1:**	320.0169	320.0173	320.0173	0.0005	0.7	0.0000	0.0	0.3	0.0	0.5	0.0
		A+2:	321.0108	321.0114	321.0114	0.0006	1.0	0.0000	0.1				
enniatin-A		**A:**	704.4457	704.4470	704.4469	0.0014	1.0	0.0001	0.1				
$C_{36}H_{63}N_3O_9$	**Bruker:**	**A+1:**	705.4489	705.4500	705.4499	0.0011	0.8	0.0001	0.1	0.4	0.0	0.6	0.0
		A+2:	706.4517	706.4523	706.4522	0.0006	0.4	0.0001	0.1				
		A:	704.4457	704.4486	704.4490	0.0030	2.1	0.0004	0.3				
	Agilent:	**A+1:**	705.4489	705.4506	705.4507	0.0017	1.2	0.0001	0.1	0.7	0.0	1.5	0.2
		A+2:	706.4517	706.4525	706.4526	0.0008	0.5	0.0001	0.0				
folic acid		**A:**	442.1470	442.1475	442.1475	0.0005	0.6	0.0000	0.0				
$C_{19}H_{19}N_7O_6$	**Bruker:**	**A+1:**	443.1497	443.1501	443.1501	0.0004	0.5	0.0000	0.0	0.5	0.0	0.6	0.1
		A+2:	444.1520	444.1520	444.1521	0.0000	0.0	0.0001	0.1				
	Agilent:	**A:**	442.1470	442.1470	442.1470	0.0000	0.0	0.0000	0.0	0.4	0.2	0.4	0.1

										A+1 to A+2		A to A+2	
	QTOF		Theoretical (*m/z*)	LCMS (*m/z*)	XCMS (*m/z*)	LCMS-Theoretical (u)	LCMS-Theoretical (± ppm)	LCMS-XCMS (u)	LCMS-XCMS (± ppm)	LCMS-Theoretical (± ppm)	LCMS-XCMS (± ppm)	LCMS-Theoretical (± ppm)	LCMS-XCMS (± ppm)
		A+1:	443.1497	443.1497	443.1498	0.0000	0.0	0.0001	0.1				
		A+2:	444.1520	444.1517	444.1517	0.0003	0.3	0.0000	0.1				
malformin-A		**A:**	530.2465	530.2467	530.2467	0.0002	0.2	0.0000	0.0				
$C_{23}H_{39}N_5O_5S_2$	**Bruker:**	**A+1:**	531.2493	531.2494	531.2493	0.0001	0.1	0.0001	0.1	1.1	0.0	1.1	0.0
		A+2:	532.2458	532.2447	532.2446	0.0011	1.0	0.0001	0.1				
		A:	552.2285	552.2311	552.2310	0.0026	2.4	0.0001	0.1				
	Agilent:	**A+1:**	553.2313	553.2324	553.2324	0.0011	1.0	0.0000	0.0	0.0	0.0	1.4	0.0
		A+2:	554.2277	554.2288	554.2287	0.0011	1.0	0.0001	0.0				
neosolaniol		**A:**	400.1966	400.1967	400.1967	0.0001	0.1	0.0000	0.0				
$C_{19}H_{26}O_8$	**Bruker:**	**A+1:**	401.1999	401.1997	401.1997	0.0002	0.2	0.0000	0.0	0.5	0.0	0.9	0.0
		A+2:	402.2022	402.2016	402.2016	0.0006	0.7	0.0000	0.0				
		A:	405.1520	405.1536	405.1539	0.0016	2.0	0.0003	0.4				
	Agilent:	**A+1:**	406.1554	406.1559	406.1560	0.0005	0.6	0.0001	0.2	0.7	0.4	0.6	0.2
		A+2:	407.1577	407.1588	407.1593	0.0011	1.4	0.0005	0.6				
nidulin		**A:**	443.0214	443.0218	443.0218	0.0004	0.4	0.0000	0.0				
$C_{20}H_{17}Cl_3O_5$	**Bruker:**	**A+1:**	444.0248	444.0249	444.0249	0.0001	0.1	0.0000	0.0	0.3	0.1	0.1	0.1
		A+2:	445.0188	445.0191	445.0190	0.0003	0.4	0.0001	0.1				
		A:	907.0175	907.0189	907.0188	0.0014	0.8	0.0001	0.1				
	Agilent:	**A+1:**	908.0209	908.0221	908.0220	0.0012	0.6	0.0001	0.0	0.5	0.0	0.4	0.0
		A+2:	909.0151	909.0172	909.0171	0.0021	1.1	0.0001	0.0				
ochratoxin-alpha		**A:**	257.0211	257.0214	257.0214	0.0003	0.5	0.0000	0.1				
$C_{11}H_9ClO_5$	**Bruker:**	**A+1:**	258.0245	258.0246	258.0246	0.0001	0.2	0.0000	0.0	0.3	0.0	0.7	0.0
		A+2:	259.0186	259.0185	259.0185	0.0001	0.2	0.0000	0.0				

	QTOF		Theoretical (*m/z*)	LCMS (*m/z*)	XCMS (*m/z*)	LCMS-Theoretical (u)	LCMS-Theoretical (± ppm)	LCMS-XCMS (u)	LCMS-XCMS (± ppm)	A+1 to A+2 LCMS-Theoretical (± ppm)	A+1 to A+2 LCMS-XCMS (± ppm)	A to A+2 LCMS-Theoretical (± ppm)	A to A+2 LCMS-XCMS (± ppm)
		A:	257.0211	257.0217	257.0218	0.0006	1.1	0.0001	0.2				
	Agilent:	**A+1:**	258.0245	258.0248	258.0249	0.0003	0.5	0.0001	0.1	0.1	0.1	0.7	0.2
		A+2:	259.0186	259.0188	259.0188	0.0002	0.4	0.0000	0.0				
penicillin-G		**A:**	335.1060	335.1062	335.1061	0.0002	0.3	0.0001	0.1				
$C_{16}H_{18}N_2O_4S$	**Bruker:**	**A+1:**	336.1090	336.1089	336.1089	0.0001	0.1	0.0000	0.1	2.2	0.1	2.6	0.1
		A+2:	337.1053	337.1037	337.1037	0.0016	2.3	0.0000	0.0				
		A:	335.1060	335.1062	335.1061	0.0002	0.3	0.0001	0.1				
	Agilent:	**A+1:**	336.1090	336.1093	336.1093	0.0003	0.5	0.0000	0.0	0.2	0.0	0.3	0.1
		A+2:	337.1053	337.1057	337.1057	0.0004	0.6	0.0000	0.0				
penitrem A		**A:**	634.2930	634.2927	634.2926	0.0003	0.2	0.0001	0.1				
$C_{37}H_{44}ClNO_6$	**Bruker:**	**A+1:**	635.2963	635.2958	635.2958	0.0005	0.4	0.0000	0.0	0.2	0.1	0.4	0.0
		A+2:	636.2922	636.2914	636.2913	0.0008	0.6	0.0001	0.1				
		A:	634.2930	634.2934	634.2935	0.0004	0.3	0.0001	0.1				
	Agilent:	**A+1:**	635.2963	635.2965	635.2966	0.0002	0.1	0.0001	0.1	0.4	0.0	0.3	0.0
		A+2:	636.2922	636.2929	636.2930	0.0007	0.6	0.0001	0.1				
phalloidin		**A:**	789.3236	789.3240	789.3239	0.0004	0.3	0.0001	0.0				
$C_{35}H_{48}N_8O_{11}S$	**Bruker:**	**A+1:**	790.3265	790.3266	790.3265	0.0001	0.1	0.0001	0.1	0.2	0.0	0.4	0.1
		A+2:	791.3263	791.3260	791.3258	0.0003	0.2	0.0002	0.1				
		A:	789.3236	789.3275	789.3280	0.0039	2.5	0.0005	0.3				
	Agilent:	**A+1:**	790.3265	790.3289	790.3291	0.0024	1.5	0.0002	0.1	0.3	0.1	1.2	0.3
		A+2:	791.3263	791.3282	791.3282	0.0019	1.2	0.0000	0.0				
phomopsin A	**Bruker:**	**A:**	789.2857	789.2867	789.2867	0.0010	0.7	0.0000	0.0	0.0	0.0	0.3	0.0
$C_{36}H_{45}ClN_6O_{12}$		**A+1:**	790.2888	790.2893	790.2893	0.0005	0.3	0.0000	0.0				

	QTOF		Theoretical (*m/z*)	LCMS (*m/z*)	XCMS (*m/z*)	LCMS-Theoretical (u)	LCMS-Theoretical (± ppm)	LCMS-XCMS (u)	LCMS-XCMS (± ppm)	A+1 to A+2 LCMS-Theoretical (± ppm)	A+1 to A+2 LCMS-XCMS (± ppm)	A to A+2 LCMS-Theoretical (± ppm)	A to A+2 LCMS-XCMS (± ppm)
		A+2:	791.2850	791.2855	791.2855	0.0005	0.3	0.0000	0.0				
		A:	789.2857	789.2869	789.2870	0.0012	0.8	0.0001	0.1				
	Agilent:	**A+1:**	790.2888	790.2895	790.2895	0.0007	0.5	0.0000	0.0	0.3	0.0	0.1	0.1
		A+2:	791.2850	791.2861	791.2861	0.0012	0.7	0.0000	0.0				
roridin A		**A:**	533.2745	533.2747	533.2746	0.0002	0.2	0.0001	0.1				
$C_{29}H_{40}O_9$	**Bruker:**	**A+1:**	534.2779	534.2779	534.2778	0.0000	0.0	0.0001	0.1	0.5	0.0	0.7	0.0
		A+2:	535.2806	535.2801	535.2801	0.0005	0.5	0.0000	0.0				
		A:	555.2565	555.2595	555.2600	0.0031	2.7	0.0005	0.5				
	Agilent:	**A+1:**	556.2599	556.2613	556.2616	0.0014	1.3	0.0003	0.2	0.5	0.3	1.9	0.5
		A+2:	557.2626	557.2635	557.2635	0.0009	0.8	0.0000	0.0				
vitamin B1		**A:**	133.0595	133.0597	133.0596	0.0002	0.7	0.0001	0.3				
$C_{12}H_{17}N_4OS$	**Bruker:**	**A+1:**	133.5608	133.5609	133.5609	0.0001	0.3	0.0000	0.2	2.1	0.2	2.4	0.2
		A+2:	134.0585	134.0580	134.0580	0.0005	1.7	0.0000	0.1				
		A:	265.1118	265.1123	265.1125	0.0005	1.0	0.0002	0.3				
	Agilent:	**A+1:**	266.1144	266.1150	266.1150	0.0007	1.2	0.0000	0.1	1.3	0.1	1.1	0.1
		A+2:	267.1096	267.1096	267.1097	0.0000	0.1	0.0001	0.2				
	Bruker av.:					0.0005	0.6	0.0001	0.1	0.9	0.1	1.2	0.1
	Agilent av.:					0.0012	1.1	0.0002	0.2	0.8	0.2	1.1	0.2
	Total av.:					0.0008	0.9	0.0001	0.1	0.9	0.1	1.1	0.2

Table S-7: *Mass Accuracy Comparison*

Analysis of XCMS extraction of LCMS data of dominant ion from each reference standard and comparison of acquired data to theoretically calculated.

	QTOF		Theoretical (%)	LCMS (%)	XCMS (%)	LCMS-Theoretical (± %)	LCMS-XCMS (± %)	A:A+2 Ratio LCMS-Theoretical (± %)	A:A+2 Ratio LCMS-XCMS (± %)
amphotericin B		**A:**	100.0	100.0	100.0				
$C_{47}H_{73}NO_{17}$	**Bruker:**	**A+1:**	52.7	45.9	46.5	3.4	0.3		
		A+2:	17.1	12.6	13.1	2.3	0.2	1.0	0.2
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	52.7	52.2	59.5	0.3	3.6		
		A+2:	17.1	17.7	17.3	0.3	0.2	0.1	0.1
BE 52440A isomer		**A:**	100.0	100.0	100.0				
$C_{33}H_{34}O_{13}S$	**Bruker:**	**A+1:**	37.4	28.8	29.2	4.3	0.2		
		A+2:	14.0	7.9	8.2	3.1	0.2	2.8	0.2
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	37.4	36.3	36.3	0.6	0.0		
		A+2:	14.0	12.0	12.0	1.0	0.0	0.6	0.0
bis-sclerotioramin		**A:**	100.0	100.0	100.0				
$C_{46}H_{54}Cl_2N_2O_8$	**Bruker:**	**A+1:**	51.4	37.6	39.0	6.9	0.7		
		A+2:	78.6	66.7	67.3	6.0	0.3	0.1	0.0
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	51.4	55.8	55.6	2.2	0.1		
		A+2:	78.6	79.6	79.5	0.5	0.0	0.0	0.0
cephalosporin-C		**A:**	100.0	100.0	100.0				
$C_{16}H_{21}N_3O_8S$	**Bruker:**	**A+1:**	19.8	13.5	13.8	3.1	0.2		
		A+2:	8.0	3.3	3.4	2.4	0.0	9.0	0.4
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	19.8	26.3	26.6	3.3	0.2		
		A+2:	8.0	8.6	9.2	0.3	0.3	0.4	0.4
citreo-isocoumarin		**A:**	100.0	100.0	100.0				
$C_{14}H_{14}O_6$	**Bruker:**	**A+1:**	15.5	18.7	18.8	1.6	0.0		
		A+2:	2.4	3.5	3.2	0.6	0.1	6.8*	1.1
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	15.5	21.3	21.5	2.9	0.1		
		A+2:	2.4	5.6	5.5	1.6	0.1	12.3*	0.2
dichlorodiaportin		**A:**	100.0	100.0	100.0				
$C_{13}H_{12}Cl_2O_5$	**Bruker:**	**A+1:**	14.4	10.6	10.8	1.9	0.1		
		A+2:	66.0	57.4	58.0	4.3	0.3	0.1	0.0
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	14.4	15.4	15.5	0.5	0.0		
		A+2:	66.0	66.8	66.9	0.4	0.1	0.0	0.0
enniatin-A		**A:**	100.0	100.0	100.0				
$C_{36}H_{63}N_3O_9$	**Bruker:**	**A+1:**	41.1	33.5	34.0	3.8	0.3		
		A+2:	10.1	6.6	7.0	1.7	0.2	2.6	0.4
	Agilent:	**A:**	100.0	100.0	100.0				

	QTOF		Theoretical (%)	LCMS (%)	XCMS (%)	LCMS-Theoretical (± %)	LCMS-XCMS (± %)	A:A+2 Ratio LCMS-Theoretical (± %)	A:A+2 Ratio LCMS-XCMS (± %)
		A+1:	41.1	51.1	51.0	5.0	0.1		
		A+2:	10.1	13.4	13.3	1.7	0.1	1.2	0.0
folic acid		**A:**	100.0	100.0	100.0				
$C_{19}H_{19}N_7O_6$	**Bruker:**	**A+1:**	23.6	17.4	17.7	3.1	0.2		
		A+2:	3.9	2.5	2.6	0.7	0.1	7.3	0.9
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	23.6	22.0	22.0	0.8	0.0		
		A+2:	3.9	3.4	3.3	0.2	0.0	1.9	0.3
malformin-A		**A:**	100.0	100.0	100.0				
$C_{23}H_{39}N_5O_5S_2$	**Bruker:**	**A+1:**	29.0	22.7	23.3	3.1	0.3		
		A+2:	14.1	8.0	8.3	3.1	0.2	2.7	0.3
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	29.0	36.9	36.8	4.0	0.1		
		A+2:	14.1	16.1	16.0	1.0	0.0	0.4	0.0
neosolaniol		**A:**	100.0	100.0	100.0				
$C_{19}H_{26}O_8$	**Bruker:**	**A+1:**	21.6	16.2	16.4	2.7	0.1		
		A+2:	3.9	2.3	2.5	0.8	0.1	8.5	1.1
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	21.2	23.8	23.6	1.3	0.1		
		A+2:	3.8	4.0	4.0	0.1	0.0	0.8	0.2
nidulin		**A:**	100.0	92.3	95.8	3.8	1.8		
$C_{20}H_{17}Cl_3O_5$	**Bruker:**	**A+1:**	22.0	18.2	18.4	1.9	0.1		
		A+2:	99.3	100.0	100.0	0.3		0.0	0.0
		A:	49.1	49.6	49.5	0.2	0.1		
	Agilent:	**A+1:**	21.6	22.3	22.2	0.3	0.0		
		A+2:	100.0	100.0	100.0			0.0	0.0
ochratoxin-alpha		**A:**	100.0	100.0	100.0				
$C_{11}H_9ClO_5$	**Bruker:**	**A+1:**	12.2	9.0	9.4	1.6	0.2		
		A+2:	33.7	29.3	29.4	2.2	0.1	0.2	0.0
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	12.2	11.5	11.6	0.3	0.0		
		A+2:	33.7	33.3	33.3	0.2	0.0	0.0	0.0
penicillin-G		**A:**	100.0	100.0	100.0				
$C_{16}H_{18}N_2O_4S$	**Bruker:**	**A+1:**	19.2	13.7	14.0	2.8	0.2		
		A+2:	7.1	3.0	3.2	2.0	0.1	9.4	1.1
		A:	100.0	100.0	100.0				
	Agilent:	**A+1:**	19.2	18.3	18.3	0.4	0.0		
		A+2:	7.1	5.4	5.4	0.8	0.0	2.2	0.0
penitrem A	**Bruker:**	**A:**	100.0	100.0	100.0				
$C_{37}H_{44}ClNO_6$		**A+1:**	41.1	37.5	37.4	1.8	0.1		

	QTOF		Theoretical (%)	LCMS (%)	XCMS (%)	LCMS-Theoretical (± %)	LCMS-XCMS (± %)	A:A+2 Ratio LCMS-Theoretical (± %)	A:A+2 Ratio LCMS-XCMS (± %)
		A+2:	41.5	31.4	31.3	5.0	0.1	0.4	0.0
		A:	100.0	100.0	100.0				
	Agilent:	A+1:	41.1	40.0	40.0	0.6	0.0		
		A+2:	41.5	36.1	36.0	2.7	0.0	0.2	0.0
phalloidin		A:	100.0	100.0	100.0				
$C_{35}H_{48}N_8O_{11}S$	Bruker:	A+1:	42.6	34.6	35.1	4.0	0.3		
		A+2:	15.6	9.6	10.0	3.0	0.2	2.0	0.2
		A:	100.0	100.0	100.0				
	Agilent:	A+1:	42.6	52.2	51.9	4.8	0.2		
		A+2:	15.6	20.2	20.0	2.3	0.1	0.7	0.0
phomopsin A		A:	100.0	100.0	100.0				
$C_{36}H_{45}ClN_6O_{12}$	Bruker:	A+1:	42.1	41.3	40.4	0.4	0.5		
		A+2:	43.1	35.8	35.3	3.6	0.3	0.2	0.0
		A:	100.0	100.0	100.0				
	Agilent:	A+1:	42.1	42.8	42.7	0.4	0.0		
		A+2:	43.1	41.9	41.8	0.6	0.1	0.0	0.0
roridin A		A:	100.0	100.0	100.0				
$C_{29}H_{40}O_9$	Bruker:	A+1:	32.2	24.6	25.3	3.8	0.4		
		A+2:	6.9	4.3	4.5	1.3	0.1	4.5	0.8
		A:	100.0	100.0	100.0				
	Agilent:	A+1:	32.2	41.9	41.0	4.9	0.5		
		A+2:	6.9	9.2	9.0	1.2	0.1	1.9	0.1
vitamin B1		A:	100.0	100.0	100.0				
$C_{12}H_{17}N_4OS$	Bruker:	A+1:	15.5	12.4	12.1	1.5	0.2		
		A+2:	5.8	3.6	3.6	1.1	0.0	5.5	0.1
		A:	100.0	100.0	100.0				
	Agilent:	A+1:	15.5	14.5	14.5	0.5	0.0		
		A+2:	5.8	4.6	4.6	0.6	0.0	2.3	0.0
		Bruker av.:				2.7	0.2	3.3	0.4
		Agilent av.:				1.4	0.2	0.8	0.1
		Total av.:				2.0	0.2	2.0	0.2

Table S-8: *Isotope Pattern and Data Extraction Accuracy Comparison*

Analysis of XCMS extraction of LCMS data of dominant ion from each reference standard and comparison of acquired data to theoretically calculated. * = values excluded from average calculations due to interference from coeluting dihydro isomer.

	BE 52440A isomer		bis-sclerotioramin		cephalosporin-C		dichlorodiaportin	
	$C_{33}H_{32}O_{14}S$		$C_{46}H_{54}Cl_2N_2O_8$		$C_{16}H_{21}N_3O_8S$		$C_{13}H_{12}Cl_2O_5$	
QTOF:	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent
Retention time (min):	5.63	8.40	9.41	13.09	0.65	2.52	6.44	9.28
Observed ion:	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+2H]^{2+}$**	**$[M+2H]^{2+}$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**
A+1:	686.1629	686.1628	417.6723	417.6730	417.1152	417.1155	320.0170	320.0173
A+2:	687.1622	687.1628	418.1698	418.1710	418.1115	418.1132	321.0110	321.0114
A+1 to A+2:	0.9993	1.0000	0.4975	0.4980	0.9963	0.9977	0.9940	0.9941
Predicted A+1 to A+2:	0.9994	0.9994	0.4977	0.4977	0.9969	0.9969	0.9939	0.9939
Inter-isotopic accuracy (ppm):	0.1	0.9	0.6	0.6	1.5	1.8	0.3	0.6
Observed ion:	**$[M+NH_4]^+$**	**$[M+NH_4]^+$**	**$[M+H]^+$**	**$[M+H]^+$**			**$[M+Na]^+$**	**$[M+Na]^+$**
A+1:	703.1890	703.1885	834.3354	834.3383			341.9989	341.9987
A+2:	704.1879	704.1869	835.3306	835.3343			342.9929	342.9928
A+1 to A+2:	0.9989	0.9984	0.9952	0.9960			0.9940	0.9941
Predicted A+1 to A+2:	0.9994	0.9994	0.9955	0.9955			0.9939	0.9939
Inter-isotopic accuracy (ppm):	0.7	1.5	0.3	0.6			0.3	0.6
Observed ion:	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**			**$[M+K]^+$**	**$[M+K]^+$**
A+1:	708.1440	708.1438	856.3173	856.3194			357.9726	357.9729
A+2:	709.1432	709.1438	857.3126	857.3151			358.9667	358.9668
A+1 to A+2:	0.9992	1.0000	0.9953	0.9957			0.9941	0.9939
Predicted A+1 to A+2:	0.9994	0.9994	0.9955	0.9955			0.9940	0.9940
Inter-isotopic accuracy (ppm):	0.3	0.9	0.2	0.3			0.3	0.3
Observed ion:		**$[2M+H]^+$**	**$[M+K]^+$**				**$[2M+Na]^+$**	**$[2M+Na]^+$**
A+1:		1370.3133	872.2901				660.0028	660.0038
A+2:		1371.3139	873.2860				660.9989	660.9993
A+1 to A+2:		1.0006	0.9959				0.9961	0.9955

	BE 52440A isomer	bis-sclerotioramin	cephalosporin-C	dichlorodiaportin	
	$C_{33}H_{32}O_{14}S$	$C_{46}H_{54}Cl_2N_2O_8$	$C_{16}H_{21}N_3O_8S$	$C_{13}H_{12}Cl_2O_5$	
Predicted A+1 to A+2:	1.0007	0.9954		0.9941	0.9941
Inter-isotopic accuracy (ppm):	0.1	0.6		3.1	2.2
Observed ion:	**$[2M+NH_4]^+$**			**$[2M+K]^+$**	
A+1:	1387.3395			675.9733	
A+2:	1388.3385			676.9691	
A+1 to A+2:	0.9990			0.9958	
Predicted A+1 to A+2:	1.0007			0.9941	
Inter-isotopic accuracy (ppm):	1.2			2.5	
Observed ion:	**$[2M+Na]^+$**			**$[4M+K+H]^{2+}$**	
A+1:	1392.2971			656.4958	
A+2:	1393.2976			656.9918	
A+1 to A+2:	1.0005			0.4960	
Predicted A+1 to A+2:	1.0007			0.4972	
Inter-isotopic accuracy (ppm):	0.1			1.8	
Observed ion:	**$[2M+K]^+$**				
A+1:	1408.2659				
A+2:	1409.2665				
A+1 to A+2:	1.0006				
Predicted A+1 to A+2:	0.9998				
Inter-isotopic accuracy (ppm):	0.5				

	malformin-A		nidulin		ochratoxin-alpha		penicillin-G	
	$C_{23}H_{39}N_5O_5S_2$		$C_{20}H_{17}Cl_3O_5$		$C_{11}H_9ClO_5$		$C_{16}H_{18}N_2O_4S$	
QTOF:	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent
Retention time (min):	6.12	8.76	9.41	12.82	4.18	6.15	4.80	7.00
Observed ion:	**$[M+H]^+$**	**$[M+H]^+$**	**$[M-HCl+H]^+$**		**$[M-H_2O+H]^+$**	**$[M-H_2O+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**
A+1:	531.2493	531.2500	408.0477		240.0140	240.0141	336.1089	336.1093
A+2:	532.2446	532.2469	409.0418		241.0079	241.0081	337.1037	337.1057
A+1 to A+2:	0.9953	0.9969	0.9941		0.9939	0.9940	0.9948	0.9964
Predicted A+1 to A+2:	0.9964	0.9964	0.9941		0.9940	0.9940	0.9963	0.9963
Inter-isotopic accuracy (ppm):	2.1	0.9	0.0		0.5	0.1	4.4	0.3
Observed ion:	**$[M+NH_4]^+$**	**$[M+NH_4]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+NH_4]^+$**	
A+1:	548.2754	548.2762	444.0249	444.0251	258.0246	258.02	353.1352	
A+2:	549.2710	549.2727	445.0190	445.0194	259.0185	259.0188	354.1297	
A+1 to A+2:	0.9956	0.9965	0.9941	0.9943	0.9939	0.9939	0.9945	
Predicted A+1 to A+2:	0.9965	0.9965	0.9940	0.9940	0.9941	0.9941	0.9964	
Inter-isotopic accuracy (ppm):	1.6	0.0	0.3	0.8	0.7	0.7	5.4	
Observed ion:	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+NH_4]^+$**	**$[M+NH_4]^+$**	**$[M-H_2O+Na]^+$**		**$[M+Na]^+$**	**$[M+Na]^+$**
A+1:	553.2310	553.2323	461.0504	461.0514	261.996		358.0907	358.0913
A+2:	554.2268	554.2287	462.0447	462.0452	262.9896		359.0855	359.0877
A+1 to A+2:	0.9958	0.9964	0.9943	0.9938	0.9936		0.9948	0.9964
Predicted A+1 to A+2:	0.9964	0.9964	0.9941	0.9941	0.9940		0.9963	0.9963
Inter-isotopic accuracy (ppm):	1.1	0.0	0.5	0.6	1.7		4.1	0.3
Observed ion:	**$[M+K]^+$**	**$[M+K]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+NH_4]^+$**		**$[M+K]^+$**	**$[M+K]^+$**
A+1:	569.2045	569.2049	466.0061	466.0072	275.0508		374.0646	374.0643
A+2:	570.2	570.2013	467.0000	467.0013	276.0448		375.0595	375.0605
A+1 to A+2:	0.9955	0.9964	0.9939	0.9941	0.9940		0.9949	0.9962

	malformin-A		nidulin		ochratoxin-alpha		penicillin-G	
	$C_{23}H_{39}N_5O_5S_2$		$C_{20}H_{17}Cl_3O_5$		$C_{11}H_9ClO_5$		$C_{16}H_{18}N_2O_4S$	
Predicted A+1 to A+2:	0.9961	0.9961	0.9940	0.9940	0.9942		0.9957	0.9957
Inter-isotopic accuracy (ppm):	1.0	0.6	0.1	0.3	0.9		2.2	1.3
Observed ion:	**$[2M+H]^+$**	**$[2M+H]^+$**		**$[M+K]^+$**	**$[M+Na]^+$**	**$[M+Na]^+$**	**$[2M+H]^+$**	
A+1:	1060.49	1060.4899		481.9807	280.0066	280.0064	670.2071	
A+2:	1061.49	1061.4877		482.9749	281.0005	281.0007	671.2041	
A+1 to A+2:	0.9966	0.9978		0.9942	0.9939	0.9943	0.9970	
Predicted A+1 to A+2:	0.9979	0.9979		0.9940	0.9941	0.9941	0.9977	
Inter-isotopic accuracy (ppm):	1.2	0.1		0.4	0.6	0.8	1.0	
Observed ion:		**$[2M+NH_4]^+$**	**$[M+H_2O+Na]^+$**		**$[M+K]^+$**		**$[2M+NH_4]^+$**	
A+1:		1077.5158	484.0162		295.9803		687.2333	
A+2:		1078.5134	485.0105		296.9743		688.2316	
A+1 to A+2:		0.9976	0.9943		0.9940		0.9983	
Predicted A+1 to A+2:		0.9979	0.9940		0.9942		0.9977	
Inter-isotopic accuracy (ppm):		0.3	0.7		0.6		0.8	
Observed ion:	**$[2M+Na]^+$**	**$[2M+Na]^+$**	**$[2M+Na]^+$**	**$[2M+Na]^+$**	**$[M+H_2O+Na]^+$**		**$[2M+Na]^+$**	**$[2M+Na]^+$**
A+1:	1082.4710	1082.4723	908.0169	908.022	298.0147		692.1893	692.1894
A+2:	1083.4682	1083.4709	909.0144	909.0171	299.0106		693.1864	693.1872
A+1 to A+2:	0.9972	0.9986	0.9975	0.9951	0.9959		0.9971	0.9978
Predicted A+1 to A+2:	0.9979	0.9979	0.9942	0.9942	0.9941		0.9977	0.9977
Inter-isotopic accuracy (ppm):	0.6	0.7	3.6	1.0	6.0		0.8	0.2
Observed ion:	**$[2M+K]^+$**	**$[2M+K]^+$**				**$[M-H+2Na]^+$**		
A+1:	1098.4446	1098.4446				301.9885		
A+2:	1099.4414	1099.4419				302.9823		
A+1 to A+2:	0.9968	0.9973				0.9938		

	malformin-A		nidulin	ochratoxin-alpha		penicillin-G
	$C_{23}H_{39}N_5O_5S_2$		$C_{20}H_{17}Cl_3O_5$	$C_{11}H_9ClO_5$		$C_{16}H_{18}N_2O_4S$
Predicted A+1 to A+2:	0.9975	0.9975			0.9941	
Inter-isotopic accuracy (ppm):	0.6	0.1			0.9	
Observed ion:				**$[2M+Na]^+$**	**$[2M+Na]^+$**	
A+1:				536.0198	536.0196	
A+2:				537.0144	537.0123	
A+1 to A+2:				0.9946	0.9927	
Predicted A+1 to A+2:				0.9943	0.9943	
Inter-isotopic accuracy (ppm):				0.6	2.9	
Observed ion:					**$[2M-H+2Na]^+$**	
A+1:					558.0007	
A+2:					558.9955	
A+1 to A+2:					0.9948	
Predicted A+1 to A+2:					0.9943	
Inter-isotopic accuracy (ppm):					0.9	
Observed ion:					**$[2M-2H+3Na]^+$**	
A+1:					579.9825	
A+2:					580.9779	
A+1 to A+2:					0.9954	
Predicted A+1 to A+2:					0.9943	
Inter-isotopic accuracy (ppm):					1.9	
Observed ion:					**$[4M-4H+5Na]^+$**	
A+1:					1135.9697	
A+2:					1136.9682	
A+1 to A+2:					0.9985	

	malformin-A	nidulin	ochratoxin-alpha	penicillin-G
	$C_{23}H_{39}N_5O_5S_2$	$C_{20}H_{17}Cl_3O_5$	$C_{11}H_9ClO_5$	$C_{16}H_{18}N_2O_4S$
Predicted A+1 to A+2:			0.9947	
Inter-isotopic accuracy (ppm):			3.4	

	penitrem A		phalloidin		phomopsin A		vitamin B1	
	$C_{37}H_{44}ClNO_6$		$C_{35}H_{48}N_8O_{11}S$		$C_{36}H_{45}ClN_6O_{12}$		$C_{12}H_{17}N_4OS$	
QTOF:	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent	Bruker	Agilent
Retention time (min):	8.51	12.11	3.46	4.94	3.65	5.24	0.50	1.48
Observed ion:	**$[M-H_2O+H]^+$**	**$[M-H_2O+H]^+$**	**$[M-H_2O+2H]^{2+}$**		**$[M-H_2O+K+2H]^{3+}$**		**$[M-H_2O+H]^{2+}$**	
A+1:	617.2843	617.2858	386.6612		270.7462		124.5555	
A+2:	618.2798	618.2815	387.1610		271.0788		125.0527	
A+1 to A+2:	0.9955	0.9957	0.4998		0.3326		0.4972	
Predicted A+1 to A+2:	0.9958	0.9958	0.4999		0.3320		0.4975	
Inter-isotopic accuracy (ppm):	0.5	0.2	0.2		2.2		0.5	
Observed ion:	**$[M+H]^+$**	**$[M+H]^+$**	**$[M+2H]^{2+}$**		**$[M+K+2H]^{3+}$**		**$[M+H]^{2+}$**	
A+1:	635.2958	635.2966	395.6667		276.7494		133.5609	
A+2:	636.2913	636.2930	396.1666		277.0818		134.0580	
A+1 to A+2:	0.9955	0.9964	0.4999		0.3324		0.4971	
Predicted A+1 to A+2:	0.9958	0.9958	0.4999		0.3320		0.4976	
Inter-isotopic accuracy (ppm):	0.5	0.9	0.1		1.4		4.1	
Observed ion:	**$[M+Na]^+$**		**$[M+Na+H]^{2+}$**		**$[M+H_2O+K+2H]^{3+}$**		**$[M]^+$**	**$[M]^+$**
A+1:	657.2766		406.1555		282.7539		266.1144	266.1150
A+2:	658.2720		**		283.0854		267.1079	267.1097
A+1 to A+2:	0.9954		**		0.3315		0.9935	0.9947
Predicted A+1 to A+2:	0.9958		0.4999		0.3320		0.9953	0.9953

	penitrem A		phalloidin		phomopsin A		vitamin B1	
	$C_{37}H_{44}ClNO_6$		$C_{35}H_{48}N_8O_{11}S$		$C_{36}H_{45}ClN_6O_{12}$		$C_{12}H_{17}N_4OS$	
Inter-isotopic accuracy (ppm):	0.7		**		1.9		6.7	2.2
Observed ion:	**$[M+H_2O+Na]^+$**	**$[M+H_2O+Na]^+$**	**$[M+2Na]^{2+}$**		**$[M-H_2O+2H]^{2+}$**			
A+1:	675.2866	675.2890	417.6483		386.6424			
A+2:	676.2811	676.2843	418.1484		387.1399			
A+1 to A+2:	0.9945	0.9953	0.5001		0.4975			
Predicted A+1 to A+2:	0.9959	0.9959	0.4999		0.4981			
Inter-isotopic accuracy (ppm):	2.0	0.8	0.5		1.5			
Observed ion:		**$[2M+Na]^+$**	**$[M-H+Fe(III)]^{2+}$**		**$[M+2H]^{2+}$**	**$[M+2H]^{2+}$**		
A+1:		1290.5645	421.1264		395.6475	395.6483		
A+2:		1291.5624	421.621		396.1456	396.1466		
A+1 to A+2:		0.9979	0.4946		0.4981	0.4983		
Predicted A+1 to A+2:		0.9971	0.4963		0.4981	0.4981		
Inter-isotopic accuracy (ppm):		0.6	3.9		0.0	0.5		
Observed ion:			**$[M+H]^+$**	**$[M+H]^+$**	**$[M+Na+H]^{2+}$**	**$[M+Na+H]^{2+}$**		
A+1:			790.3265	790.3291	406.6386	**		
A+2:			791.3258	791.3282	407.1358	407.1271		
A+1 to A+2:			0.9993	0.9991	0.4972	**		
Predicted A+1 to A+2:			0.9998	0.9998	0.4981	0.4981		
Inter-isotopic accuracy (ppm):			0.6	0.9	2.2	**		
Observed ion:			**$[M+Na]^+$**	**$[M+Na]^+$**	**$[M+Xa]^{2+}$**			
A+1:			811.8053	812.3093	413.6464			
A+2:			**	813.3096	414.143			
A+1 to A+2:			**	1.0003	0.4966			
Predicted A+1 to A+2:			0.9998	0.9998	*			

	penitrem A	phalloidin		phomopsin A		vitamin B1
	$C_{37}H_{44}ClNO_6$	$C_{35}H_{48}N_8O_{11}S$		$C_{36}H_{45}ClN_6O_{12}$		$C_{12}H_{17}N_4OS$
Inter-isotopic accuracy (ppm):		**	0.7	*		
Observed ion:		**$[M+K]^+$**		**$[M+2Na]^{2+}$**		
A+1:		828.2809		417.6292		
A+2:		829.2788		418.1271		
A+1 to A+2:		0.9979		0.4979		
Predicted A+1 to A+2:		0.9983		0.4981		
Inter-isotopic accuracy (ppm):		0.5		0.5		
Observed ion:		**$[2M+Na+H]^{2+}$**		**$[M+Xb]^{2+}$**		
A+1:		**		420.1061		
A+2:		801.3050		420.6037		
A+1 to A+2:		**		0.4976		
Predicted A+1 to A+2:		0.5005		*		
Inter-isotopic accuracy (ppm):		**		*		
Observed ion:		**$[2M+K+H]^{2+}$**		**$[M+Xc]^{2+}$**		
A+1:		808.7981		**		
A+2:		809.2988		427.1053		
A+1 to A+2:		0.5007		**		
Predicted A+1 to A+2:		0.5001		*		
Inter-isotopic accuracy (ppm):		0.8		*		
Observed ion:				**$[M+H]^+$**	**$[M+H]^+$**	
A+1:				790.2893	790.2895	
A+2:				**	791.2861	
A+1 to A+2:				**	0.9966	
Predicted A+1 to A+2:				0.9962	0.9962	

	penitrem A	phalloidin	phomopsin A		vitamin B1
	$C_{37}H_{44}ClNO_6$	$C_{35}H_{48}N_8O_{11}S$	$C_{36}H_{45}ClN_6O_{12}$		$C_{12}H_{17}N_4OS$
Inter-isotopic accuracy (ppm):			**	0.5	
Observed ion:			**$[M+Xd]^+$**		
A+1:			804.3038		
A+2:			805.3003		
A+1 to A+2:			0.9965		
Predicted A+1 to A+2:			*		
Inter-isotopic accuracy (ppm):			*		
Observed ion:			**$[M+Na]^+$**	**$[M+Na]^+$**	
A+1:			812.2692	812.2701	
A+2:			813.2656	813.2666	
A+1 to A+2:			0.9964	0.9965	
Predicted A+1 to A+2:			0.9962	0.9962	
Inter-isotopic accuracy (ppm):			0.2	0.4	
Observed ion:			**$[2M+K+H]^{2+}$**		
A+1:			808.7601		
A+2:			809.2591		
A+1 to A+2:			0.4990		
Predicted A+1 to A+2:			0.4986		
Inter-isotopic accuracy (ppm):			0.5		
Observed ion:				**$[M-H+2Na]^+$**	
A+1:				834.2509	
A+2:				835.2476	
A+1 to A+2:				0.9967	
Predicted A+1 to A+2:				0.9962	

	penitrem A	phalloidin	phomopsin A	vitamin B1
	$C_{37}H_{44}ClNO_6$	$C_{35}H_{48}N_8O_{11}S$	$C_{36}H_{45}ClN_6O_{12}$	$C_{12}H_{17}N_4OS$
Inter-isotopic accuracy (ppm):			0.6	

Table S-9: *Inter-Isotopic Mass Accuracy Comparison*

Analysis of the inter-isotopic mass accuracy of the dynamic cluster analysis identified ions produced by reference standards. Xa-d denotes an adduct which is unknown. * = a value unable to be determined due to an unknown adduct. ** = a value unable to be determined due to a miss isotope cluster from ions with low abundance.

$[M+2H]^{+2}$ (*m/z*)	Retention time (min)	UTEX-2797	K-0374	KAC-39	K-0252	RCC-1436	N-1017	RCC-1435	RCC-191	RCC-1433	K-0081
644.2413*	5.18	x				x	x	x	x	x	
828.8967‡	6.16		x	x							
855.8522	5.12				x						
857.8681	4.39								x		
868.8795	5.59					x					
874.8555	4.92							x			
885.8599	5.67					x					
894.9172‡	5.81		x	x							
894.9182‡	6.13		x	x							x
909.9198‡	6.08										x
911.8990	5.87		x								
911.8997	6.20		x								
913.8753‡	5.17							x			
913.8755‡	5.32							x		x	
914.8844‡	5.00					x					
914.8856**‡	5.37					x					
914.9346	6.32			x							
917.9165	5.51										x
926.9205	4.97										x
928.8998	5.58		x								
930.8955‡	4.66								x		
931.8660‡	5.73									x	
931.8674‡	5.08							x	x	x	
948.8457‡	5.15					x			x	x	
948.8469‡	5.51									x	
948.8663‡	4.83									x	
951.8803	5.35							x			
952.8873	5.72								x		
953.8413	5.49									x	
953.8433	5.13								x		
967.9221‡	6.35	x									
970.8251	5.21									x	
977.8336	5.22									x	
977.8338	5.59									x	
979.8993	5.28				x		x				
979.8993	4.87						x				
980.9058‡	5.32					x					
984.9049‡	6.4	x									
996.8700**	5.35				x						
997.8858‡	5.05				x	x	x	x			
997.8859‡	5.4					x					

$[M+2H]^{+2}$ (*m/z*)	Retention time (min)	UTEX-2797	K-0374	KAC-39	K-0252	RCC-1436	N-1017	RCC-1435	RCC-191	RCC-1433	K-0081
1004.918	6.59	x									
1014.867	5.46				x						
1014.868	5.12				x						
1017.901	5.61					x					
1078.9128‡	5.28									x	
1078.9138‡	4.95									x	
1095.894	5.01								x	x	
1095.895	5.34									x	
1108.931	6.92						x				
1114.968	6.18	x									
1131.9526‡	6.24	x									
Data processing time (sec.):		**52.8**	**58.1**	**23.2**	**38.5**	**9.4**	**47.6**	**20.4**	**42.5**	**101.3**	**34.8**

Table S-10*: Bruker DCA-Hal Results*

Results from DCA-Hal analysis of crude extracts from 10 strains of *Prymnesin parvum* analysed on Bruker Maxis HD QTOF. * = Non-prymnesin-like molecular feature, displaying $[M+H]^{+}$. ** = Non-observed, inferred $[M+2H]^{2+}$.‡ = a *m/z* of the $[M+2H]^{+2}$ ion which has been previously reported as a prymnesin or prymnesin-like feature.[1-3]

$[M+2H]^{+2}$ (m/z)	Retention time (min)	UTEX-2797	K-0374	KAC-39	K-0252	RCC-1436	N-1017	RCC-1435	RCC-191	RCC-1433	K-0081
815.8903	9.09										x
828.8965‡	9.06		x	x							
828.8965‡	9.52		x	x							x
832.8720	9.20										x
848.8627	7.79					x					
894.9177‡	9.01			x							
894.9182‡	9.49		x	x							x
896.9172	9.01										x
909.9319‡	9.38										x
911.8992	9.59		x								x
913.8766‡	8.22							x		x	
913.8766‡	8.4									x	
913.8767‡	7.58							x			
913.8976‡	9.11										x
923.8902	6.63								x		
924.8684	8.23				x						
926.9167	9.49										x
930.8978‡	7.10								x		
931.8635‡	7.91					x		x	x	x	
931.8641‡	8.41							x			
931.8647‡	8.90							x		x	
942.8571	7.91				x						
948.8464‡	8.03									x	
948.8466‡	9.01									x	
953.8414	8.52									x	
975.9450‡	9.23										x
977.8340	8.69									x	
977.8341	8.16									x	
979.8975	7.50				x						
979.8976	8.16				x						
990.9512‡	9.19										x
997.8839‡	8.36				x	x			x		
997.8864‡	7.84				x						
1007.9315	9.27										x
1078.9120‡	8.15									x	
1078.9127‡	7.65									x	
1095.8933	7.74									x	
1110.0068	9.65		x								
1187.0197	9.57		x								
Data processing time (sec.):		**9.7**	**9.3**	**3.3**	**8.5**	**4.9**	**6.6**	**3.9**	**6.3**	**15.7**	**15.3**

Table S-11: *Agilent DCA-Hal Results*

Results from DCA-Hal analysis of crude extracts from 10 strains of *Prymnesin parvum* analysed on Agilent 6545 Q-TOF. ‡ = a *m/z* of the $[M+2H]^{+2}$ ion which has been previously reported as a prymnesin or prymnesin-like feature.[1-3]

RT	m/z	z	DCA-Hal	MeHaloCoA
5.51	911.9198	2	+	+
5.51	1822.8359	1	+	-
5.52	845.8984	2	+	-
5.53	1690.7900	1	-	+
5.54	902.9150	2	+	-
5.58	928.8998	2	+	-
5.58	1856.7998	1	+	-
5.60	862.8787	2	+	-
5.80	609.5960	3	+	-
5.80	905.9084	2	+	+
5.81	894.9172	2	+	+
5.82	565.5821	3	+	-
5.83	828.8965	2	-	+
5.83	839.8871	2	-	+
5.87	911.8990	2	+	+
5.87	1656.7862	1	+	-
5.88	620.9182	3	+	-
5.88	922.8907	2	+	-
5.89	930.8775	2	+	-
5.90	845.8784	2	+	-
5.91	856.8699	2	+	+
5.93	855.3818	2	+	-
6.07	826.3903	2	+	+
6.11	1810.8140	1	-	+
6.11	806.0381	2	+	-
6.12	552.9331	3	+	-
6.12	560.2581	3	+	-
6.12	609.5970	3	+	-
6.12	894.9191	2	+	-
6.12	903.4315	2	+	+
6.12	904.6561	2	+	-
6.12	905.9096	2	+	-
6.12	914.8947	2	-	+
6.12	1788.8332	1	+	+
6.13	546.9297	3	+	-
6.14	534.9227	3	+	-
6.14	540.9264	3	+	-
6.14	1032.9750	2	-	+
6.15	570.9066	3	+	-
6.15	694.3078	3	+	-
6.15	1032.9747	2	+	-
6.15	1657.2893	1	+	+
6.16	565.5827	3	+	-
6.16	828.8980	2	+	-
6.16	837.4101	2	+	-
6.16	839.8881	2	+	-
6.16	855.3556	2	+	-
6.16	1656.7900	1	+	-
6.16	1678.7680	1	+	+
6.18	1115.9751	2	+	-
6.19	820.8983	2	+	-
6.19	879.8837	2	+	-

RT	m/z	z	DCA-Hal	MeHaloCoA
6.20	564.2538	3	+	-
6.20	620.9173	3	+	-
6.20	626.2415	3	+	+
6.20	911.8997	2	+	+
6.20	920.4110	2	+	+
6.20	921.6378	2	+	-
6.20	922.8901	2	+	+
6.20	930.8738	2	+	+
6.20	953.8753	2	+	-
6.20	1640.7942	1	+	+
6.20	1822.7950	1	+	+
6.21	534.2361	3	+	-
6.21	540.2396	3	+	-
6.21	558.2497	3	+	-
6.21	872.8349	2	+	-
6.21	1049.9523	2	+	-
6.22	547.5706	3	+	-
6.22	845.8780	2	+	+
6.22	856.8685	2	+	+
6.22	1690.7514	1	+	+
6.23	576.9030	3	+	+
6.23	854.3911	2	+	-
6.23	855.6139	2	+	-
6.23	864.8518	2	+	+
6.23	1712.7273	1	+	+
6.31	914.9338	2	+	-
6.31	925.9238	2	+	-
6.33	848.9130	2	+	-
6.38	942.9057	2	+	-
6.39	865.8933	2	+	-
6.78	1629.7550	1	-	+
		Total:	76	30

Table S-12: *Identified Prymnesin-like Chemical Features*

Table of prymnesin-like chemical features identified by either DCA-Hal or MeHaloCoA as being halogenated.

RT	Intensity	M	DCA-Hal	MeHaloCoA
7.50	1501140	592.3510	-	FP
7.32	1499492	592.3524	-	FP
6.68	1498676	724.4298	-	FP
14.05	1490876	903.5654	-	FP
6.12	257248	1788.8330	+	+
0.49	217852	640.8836	+	+
9.84	208968	1218.7390	-	FP
6.19	179276	1824.7960	+	+
6.22	178192	1692.7530	+	+
7.30	128652	1176.6450	-	FP

Table S-13: *The Ten Chemical Features With the Highest Intensity Identified by MeHaloCoA*

Of the ten most intense chemical features highlighted by MeHaloCoA, 6 were false positives. The four most intense ions were above the detectors saturation limit. FP = False positive

Equation S-1: *A to A+2 isotope cluster spacing boundary equation for Group-Cl and Group-C*

$$V = 7.55 \times 10^{-20}(mz)^5 - 9.436 \times 10^{-16}(mz)^4 + 4.616 \times 10^{-12}(mz)^3 - 1.11 \times 10^{-8}(mz)^2 + 1.334 \times 10^{-5}(mz) + 1.9989$$

Equation S-2: *A to A+2 isotope cluster spacing boundary equation for Group-S and Group-C*

$$V = -6.471 \times 10^{-20}(mz)^5 + 7.337 \times 10^{-16}(mz)^4 - 2.948 \times 10^{-12}(mz)^3 + 4.53 \times 10^{-9}(mz)^2 - 2.502 \times 10^{-7}(mz) + 2.001$$

Equation S-3: *A to A+2 isotope cluster spacing lower boundary equation for Group-S and Group-Cl*

$$V = 1.9936 - \left(\frac{(mz)}{1000000} \times 5\right)$$

The loss of the A+1 isotopomer due to signal-to-noise within Cl and Br containing chemical features is a possibility, and may be compensated for this by utilising the A to A+2 isotope cluster spacing equations (**Supporting Equations S-1**, **S-2**, and **S-3**), a feature not yet incorporated into DCAnalysis.

```
from Tkinter import *
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from pylab import figure, show
import matplotlib.pyplot as plt
import Tkinter
import numpy
import math
import ttk
import tkFileDialog
import csv
import copy
import string
iconData = '''R0lGODlhEAAQAPcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArMwArZgArmQArzAAr/wBVAABVMwBV
        ZgBVmQBVzABV/wCAAACAMwCAZgCAmQCAzACA/wCqAACqMwCqZgCqmQCqzACq/wDVAADVMwDVZgDV
        mQDVzADV/wD/AAD/MwD/ZgD/mQD/zAD//zMAADMAMzMAZjMAmTMAzDMA/zMrADMrMzMrZjMrmTMr
        zDMr/zNVADNVMzNVZjNVmTNVzDNV/zOAADOAMzOAZjOAmTOAzDOA/zOqADOqMzOqZjOqmTOqzDOq
        /zPVADPVMzPVZjPVmTPVzDPV/zP/ADP/MzP/ZjP/mTP/zDP//2YAAGYAM2YAZmYAmWYAzGYA/2Yr
        AGYrM2YrZmYrmWYrzGYr/2ZVAGZVM2ZVZmZVmWZVzGZV/2aAAGaAM2aAZmaAmWaAzGaA/2aqAGaq
        M2aqZmaqmWaqzGaq/2bVAGbVM2bVZmbVmWbVzGbV/2b/AGb/M2b/Zmb/mWb/zGb//5kAAJkAM5kA
        ZpkAmZkAzJkA/5krAJkrM5krZpkrmZkrzJkr/5lVAJlVM5lVZplVmZlVzJlV/5mAAJmAM5mAZpmA
        mZmAzJmA/5mqAJmqM5mqZpmqmZmqzJmq/5nVAJnVM5nVZpnVmZnVzJnV/5n/AJn/M5n/Zpn/mZn/
        zJn//8wAAMwAM8wAZswAmcwAzMwA/8wrAMwrM8wrZswrmcwrzMwr/8xVAMxVM8xVZsxVmcxVzMxV
        /8yAAMyAM8yAZsyAmcyAzMyA/8yqAMyqM8yqZsyqmcyqzMyq/8zVAMzVM8zVZszVmczVzMzV/8z/
        AMz/M8z/Zsz/mcz/zMz///8AAP8AM/8AZv8Amf8AzP8A//8rAP8rM/8rZv8rmf8rzP8r//9VAP9V
        M/9VZv9Vmf9VzP9V//+AAP+AM/+AZv+Amf+AzP+A//+qAP+qM/+qZv+qmf+qzP+q///VAP/VM//V
        Zv/Vmf/VzP/V////AP//M///Zv//mf//zP///wAAAAAAAAAAAAAAACH5BAEAAPwALAAAAAAQABAA
        AAiKADt04CCQ4MCCCAlaOLiQwwWCDiEyHMihYoeHBzs0pFiRA4CCDy0OvFAQgEmKAjUe9Ghyn0mL
        EgWaBLCv5kuFHlm2rOkSAByCM2n25NnzQsudQ3kGrRlDaFKbAAwEnUq1alUiGjRkyDBEA5GvQ4Zk
        wOr169YPRJSYFUsEBIivH5KETfI1idywXwMCADs=
        '''
ionChoices1 = []
ionChoices1 = [
    'NA'
]
modeChoices1 = []
modeChoices1 = [
    'DCA-Hal',
    'DCA-Sul',
    #'DCA-C',
    #'DCA-Bor', #For compounds containing boron
    #'All Features'
]
featureChoices1 = []
featureChoices1 = [
    'Molecular',
    'Chemical'
]
class simpleapp_tk(Tkinter.Tk):
    def __init__(primary,parent):
        Tkinter.Tk.__init__(primary,parent)
        primary.parent = parent
        primary.initialize()
        primary.update()
        primary.minsize(primary.winfo_width(), primary.winfo_height())
    def initialize(primary):
        primary.grid()
        primary.grid_rowconfigure(25,weight=1)
        primary.grid_columnconfigure(10,weight=1)
        primary.massList = []
        primary.abundanceList = []
        primary.masterList = []
```

```
        primary.rtVar = StringVar()
        primary.mzVar = StringVar()
        primary.areaVar = StringVar()
        primary.requireVar = StringVar()
        primary.sepVar = StringVar()
        primary.rawMasterList = []
        primary.resultsMasterList = []
        primary.resultsMasterRateList = []
        primary.selectedSep = StringVar()
        primary.selectedFeature = StringVar()
        primary.radioSelect = StringVar()
        primary.rSel = StringVar()
        primary.selectedMode = StringVar()
        primary.minRTVar = StringVar()
        primary.minIIVar = StringVar()
        color = '0.9'
        plotFrame = Frame(primary, width=654, height=500)
        plotFrame.pack(fill="both", expand=True)
        plotFrame.grid_propagate(False)
        plotFrame.grid_rowconfigure(25, weight=1)
        plotFrame.grid_columnconfigure(10, weight=1)
        plotFrame.grid(row=2, column=10, columnspan=2, rowspan=25, sticky="neswn")
        fig = Figure()
        rect = fig.patch
        rect.set_facecolor(color)
        primary.canvas = FigureCanvasTkAgg(fig, plotFrame)
        primary.canvas.get_tk_widget().grid(row=2, column=10, columnspan=2, rowspan=25,
sticky="neswn")
        toolbar = NavigationToolbar2TkAgg(primary.canvas,primary)
        toolbar.grid(row=1, column=10, columnspan=5, sticky=W)
        primary.ax2 = fig.add_subplot(1,1,1)
        primary.ax2.set_xlabel('Mass to charge (m/z)')
        primary.ax2.set_title('Isotope Pattern')
        primary.ax2.set_ylabel('Relative Abundance')
        primary.canvas.draw()
        primary.CSVBtn = Button(primary, text='Import CSV', command=primary.getFileName)
        primary.CSVBtn.grid(row=1, column=1, columnspan=5, pady=(10,10))
        primary.rtVarLabel = Label(primary, text="RT drift:")
        primary.rtVarLabel.grid(row=3, column=1, sticky=E)
        primary.rtVarVal = Entry(primary, textvariable=primary.rtVar, width=15)
        primary.rtVarVal.grid(row=3, column=2, sticky=W)
        primary.rtVarVal.insert(0, 1.0)
        primary.rtVarValMinLabel = Label(primary, text="± sec")
        primary.rtVarValMinLabel.grid(row=3, column=3, sticky=W)
        primary.mzVarLabel = Label(primary, text="Mass error:")
        primary.mzVarLabel.grid(row=4, column=1, sticky=E)
        primary.mzVarVal = Entry(primary, textvariable=primary.mzVar, width=15)
        primary.mzVarVal.grid(row=4, column=2, sticky=W)
        primary.mzVarVal.insert(0, 5)
        primary.mzVarValDaLabel = Label(primary, text="ppm")
        primary.mzVarValDaLabel.grid(row=4, column=3, sticky=W)
        primary.minRTVarLabel = Label(primary, text="Min. RT:")
        primary.minRTVarLabel.grid(row=5, column=1, sticky=E)
        primary.minRTVarVal = Entry(primary, textvariable=primary.minRTVar, width=15)
        primary.minRTVarVal.grid(row=5, column=2, sticky=W)
        primary.minRTVarVal.insert(0, 0.3)
        primary.minRTVarValMinLabel = Label(primary, text="min")
        primary.minRTVarValMinLabel.grid(row=5, column=3, sticky=W)
        primary.minIIVarLabel = Label(primary, text="Min. Isotope Area:")
        primary.minIIVarLabel.grid(row=6, column=1, sticky=E)
        primary.minIIVarVal = Entry(primary, textvariable=primary.minIIVar, width=15)
        primary.minIIVarVal.grid(row=6, column=2, sticky=W)
        primary.minIIVarVal.insert(0, 0)
```

```
        primary.findBtn = Button(primary, text='Find Features',
command=primary.findIsoAndComps)
        primary.findBtn.grid(row=7, column=1, columnspan=5, pady=(10,10))
        primary.modeLabel = Label(primary, text="Filter:")
        primary.modeLabel.grid(row=8, column=1, sticky=E)
        primary.modeOptions = ttk.Combobox(primary, state='readonly', width=15,
textvariable=primary.selectedMode)
        primary.modeOptions["values"] = modeChoices1
        primary.modeOptions.set("DCA-Hal")
        primary.modeOptions.grid(row=8, column=2, columnspan=4, sticky=SW)
        primary.areaVarLabel = Label(primary, text="Area cut-off:")
        primary.areaVarLabel.grid(row=9, column=1, sticky=E)
        primary.areaVarVal = Entry(primary, textvariable=primary.areaVar, width=15)
        primary.areaVarVal.grid(row=9, column=2, sticky=W)
        primary.areaVarVal.insert(0, 80000)
        primary.requireVarLabel = Label(primary, text="Required Rate:")
        primary.requireVarLabel.grid(row=11, column=1, sticky=E)
        primary.requireVarVal = Entry(primary, textvariable=primary.requireVar, width=15)
        primary.requireVarVal.grid(row=11, column=2, sticky=W)
        primary.requireVarVal.insert(0, 50)
        primary.requireVarValLabel = Label(primary, text="%")
        primary.requireVarValLabel.grid(row=11, column=3, sticky=W)
        primary.featureVarLabel = Label(primary, text="Feature mode:")
        primary.featureVarLabel.grid(row=10, column=1, sticky=E)
        primary.selectedFeature.trace('w', primary.FeatureRefresh)
        primary.featureOptions = ttk.Combobox(primary, state='readonly', width=15,
textvariable=primary.selectedFeature)
        primary.featureOptions["values"] = featureChoices1
        primary.featureOptions.set("Molecular")
        primary.featureOptions.grid(row=10, column=2, columnspan=4, sticky=SW)
        primary.filterBtn = Button(primary, text='Apply Filter',
command=primary.applyFilter)
        primary.filterBtn.grid(row=12, column=1, columnspan=5, pady=(10,0))
        primary.sepLabel = Label(primary, text="Filtered Features:")
        primary.sepLabel.grid(row=13, column=1, pady=(5,0), sticky=E)
        primary.sepOptions = ttk.Combobox(primary, state='readonly', width=22,
textvariable=primary.selectedSep)
        primary.sepOptions["values"] = ionChoices1
        primary.sepOptions.set("NA")
        primary.sepOptions.grid(row=13, column=2, columnspan=4, pady=(10,0), sticky=SW)
        primary.sepOptions.bind("<<ComboboxSelected>>", primary.grph)
        primary.txt = Text(primary, borderwidth=3, relief="sunken")
        primary.txt.config(font=("consolas", 9), undo=True, wrap='word', width=35,
height=20)
        primary.txt.grid(row=20, column=1, columnspan=4, rowspan=4, sticky=E, padx=2,
pady=5)
        primary.txt.configure(state="disabled")
        scrollb = Scrollbar(primary, command=primary.txt.yview)
        scrollb.grid(row=20, column=6, rowspan=4, sticky=W+N+S, padx=(0,20), pady=5)
        primary.txt['yscrollcommand'] = scrollb.set
    def FeatureRefresh(primary, *args):
        featureMode = primary.selectedFeature.get()
        if featureMode == "Molecular":
            result = primary.requireVarVal.configure(state="normal")
        if featureMode == "Chemical":
            result = primary.requireVarVal.configure(state="disabled")
        return
    def getFileName(primary):
        fileName = tkFileDialog.askopenfilename()
        justTheFile = StringVar
        if fileName != "":
            justTheFile = fileName.split('/')
            justTheFile = justTheFile[len(justTheFile)-1]
```

```
            app.title('DCAnalysis 1.07 - ' + justTheFile)
            csvRaw = csv.reader(open(fileName, 'U'), dialect='excel')
            ionIDList = []
            mzList = []
            RTList = []
            peakAList = []
            completeList = []
            listLen = 0
            maxRT = 0
            for row in csvRaw:
                ionIDList.append(row[0])
                mzList.append(row[7])
                RTList.append(row[1])
                peakAList.append(row[2])
            listLen = len(ionIDList)
            ionIDList = ionIDList[1:listLen]
            mzList = mzList[1:listLen]
            RTList = RTList[1:listLen]
            peakAList = peakAList[1:listLen]
            n = 0
            for each in RTList:
                RTList[n] = float(RTList[n])
                n = n + 1
            n = 0
            maxRT = max(RTList)
            if maxRT < 100:
                for i in RTList:
                    RTList[n] = float(RTList[n]) * 60
                    n = n + 1
            for i in range(len(ionIDList)):
                completeList.append([float(ionIDList[i]), float(mzList[i]),
float(RTList[i]), float(peakAList[i]), 0, 0])
            completeList = sorted(completeList,key=lambda l:l[1], reverse=False) #Sorted by
[1] MZ (RT = 2, ID = 0, PA = 3)
            primary.txt.configure(state="normal")
            primary.txt.delete(1.0, END)
            for each in completeList:
                primary.txt.insert('1.0', str(each) + '\n')
            primary.txt.configure(state="disabled")
            primary.rawMasterList = completeList
        return
    #Sorts isotopomers into chemical features
    def findIsoAndComps(primary):
        RTChange = DoubleVar
        MZChange = DoubleVar
        areaCutOff = DoubleVar
        SepToFind = DoubleVar
        currentRT = DoubleVar
        currentMZ = DoubleVar
        currentAbd = DoubleVar
        testStr = StringVar
        halogenDevMax = DoubleVar
        halogenDevMin = DoubleVar
        halogenDevMax = 1.00584801
        halogenDevMin = 0.99080011
        rawList = []
        rawList = copy.deepcopy(primary.rawMasterList)
        editedList = []
        n = 0
        n1 = 1
        targetIonList = []
        targetIon = 0
        ppmError = float(primary.mzVar.get())
```

```
        RTChange = float(primary.rtVar.get())
        minRT = float(primary.minRTVar.get())
        minII = float(primary.minIIVar.get())
        while n < len(rawList):
            if float(rawList[n][2]) > (minRT*60) and float(rawList[n][3]) > minII:
                currentMZ = float(rawList[n][1])
                currentRT = float(rawList[n][2])
                currentAbd = float(rawList[n][3])
                MZChange = currentMZ / 1000000 * ppmError / 2
                while n1 < len(rawList):
                    if float(rawList[n1][2]) < (currentRT + RTChange):
                        if float(rawList[n1][2]) > (currentRT - RTChange):
                            if rawList[n][5] == 0:
                                #Triply charged features
                                if ((float(rawList[n1][1]) < ((currentMZ + (halogenDevMax /
3)) + MZChange) and \
                                    float(rawList[n1][1]) > ((currentMZ + (halogenDevMin /
3)) - MZChange)) or \
                                    (float(rawList[n1][1]) < ((currentMZ - (halogenDevMax /
3)) + MZChange) and \
                                    float(rawList[n1][1]) > ((currentMZ - (halogenDevMin /
3)) - MZChange))):

                                    if float(rawList[n1][3]) / currentAbd < 8 and
float(rawList[n1][3]) / currentAbd > 0.0000078*currentMZ**1.1098:
                                        rawList[n][4] = 3
                                        rawList[n][5] = rawList[n1][0]
                                #Doubly charged features
                                elif ((float(rawList[n1][1]) < ((currentMZ + (halogenDevMax
/ 2)) + MZChange) and \
                                    float(rawList[n1][1]) > ((currentMZ + (halogenDevMin /
2)) - MZChange)) or \
                                    (float(rawList[n1][1]) < ((currentMZ - (halogenDevMax /
2)) + MZChange) and \
                                    float(rawList[n1][1]) > ((currentMZ - (halogenDevMin /
2)) - MZChange))):

                                    if float(rawList[n1][3]) / currentAbd < 8 and
float(rawList[n1][3]) / currentAbd >  0.0000078*currentMZ**1.1098:
                                        rawList[n][4] = 2
                                        rawList[n][5] = rawList[n1][0]
                                #Singly charged features
                                elif ((float(rawList[n1][1]) < ((currentMZ + halogenDevMax)
+ MZChange) and \
                                    float(rawList[n1][1]) > ((currentMZ + halogenDevMin) -
MZChange)) or \
                                    (float(rawList[n1][1]) < ((currentMZ - halogenDevMax) +
MZChange) and \
                                    float(rawList[n1][1]) > ((currentMZ - halogenDevMin) -
MZChange))):

                                    if float(rawList[n1][3]) / currentAbd < 8 and
float(rawList[n1][3]) / currentAbd > 0.0000078*currentMZ**1.1098:
                                        rawList[n][4] = 1
                                        rawList[n][5] = rawList[n1][0]

                    n1 = n1 + 1
            n = n + 1
            n1 = n + 1
        n = n - 1
        n1 = 0
        counter = 0
        editedList = rawList[:][:]
```

```
        while n != -1:
            if editedList[n][5] != 0:
                targetIon = editedList[n][5]
                counter = counter + 1
                while float(editedList[n1][0]) != float(targetIon):
                    n1 = n1 + 1
                if float(editedList[n][4]) < float(editedList[n1][4]):
                    editedList[n][4] = editedList[n1][4]
                if editedList[n1][5] == 0:
                    editedList[n][5] =  str(editedList[n1][1]) + "-" +
str(editedList[n1][2]) + "-" + str(editedList[n1][3]) + ":"
                else:
                    editedList[n][5] = str(editedList[n1][1]) + "-" +
str(editedList[n1][2]) + "-" + str(editedList[n1][3]) + ":" + str(editedList[n1][5])
                editedList[n1][1] = "0"
                editedList[n1][2] = "0"
                editedList[n1][3] = "0"
                editedList[n1][4] = "0"
                editedList[n1][5] = 0
            n1 = 0
            n = n - 1
        counter = counter - 1
        filterList = []
        n = 1
        numberOfIsos = 0
        while n < len(editedList):
            if editedList[n][5] != 0:
                numberOfIsos = editedList[n][5].count(':')
            if numberOfIsos < 2:
                editedList[n][1] = "0"
                editedList[n][2] = "0"
                editedList[n][3] = "0"
                editedList[n][4] = "0"
                editedList[n][5] = 0
            if editedList[n][5] == 0:
                editedList[n][1] = "0"
                editedList[n][2] = "0"
                editedList[n][3] = "0"
                editedList[n][4] = "0"
                editedList[n][5] = 0
            n = n + 1
            numberOfIsos = 50
        n = 0
        while n < len(editedList):
            if editedList[n][1] != "0":
                filterList.append(editedList[n])
            n = n + 1
        primary.masterList = filterList
        primary.compileMF()
        return
    #Compiles chemical features into molecular features
    def compileMF(primary):
        filterList = primary.masterList
        RTChange = float(primary.rtVar.get())
        ppmError = float(primary.mzVar.get())
        targetIonList = []
        currentIon = DoubleVar
        currentRT = DoubleVar
        MZChange = DoubleVar
        filterFind = 0
        mFNum = 1
        n = 0
        while n < len(filterList):
```

```
            filterList[n][0] = 0
            n = n + 1
        n = 0
        n1 = 0
        #Loop to search for the mass difference between a chemical feature and common
adducts and fragments, K, Na, NH4, and -H2O etc.
        while n < len(filterList):
            mFNum = mFNum + 1
            if filterList[n][0] == 0 and filterList[n][4] != 0:
                currentIon = float(filterList[n][1]) * float(filterList[n][4])
                filterList[n][0] = mFNum
                MZChange = currentIon / 1000000 * ppmError
                currentRT = filterList[n][2]
            else:
                n1 = len(filterList)
            while n1 < len(filterList):
                if filterList[n][4] != 0:
                    compareIon = float(filterList[n1][1]) * float(filterList[n1][4])
                else:
                    compareIon = float(filterList[n1][1])

                if float(filterList[n1][2]) < (currentRT + RTChange) and \
                        float(filterList[n1][2]) > (currentRT - RTChange):
                    if float(compareIon) > (float(currentIon) + 4.95539) - MZChange and
float(compareIon) < (float(currentIon) + 4.95539) + MZChange or \
                       float(currentIon) > (float(compareIon) + 4.95539) - MZChange and
float(currentIon) < (float(compareIon) + 4.95539) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "Na-NH4 _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 15.97394) - MZChange and
float(compareIon) < (float(currentIon) + 15.97394) + MZChange or \
                       float(currentIon) > (float(compareIon) + 15.97394) - MZChange and
float(currentIon) < (float(compareIon) + 15.97394) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "K-Na _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 17.02655) - MZChange and
float(compareIon) < (float(currentIon) + 17.02655) + MZChange or \
                       float(currentIon) > (float(compareIon) + 17.02655) - MZChange and
float(currentIon) < (float(compareIon) + 17.02655) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "NH4-M _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 17.96611) - MZChange and
float(compareIon) < (float(currentIon) + 17.96611) + MZChange or \
                       float(currentIon) > (float(compareIon) + 17.96611) - MZChange and
float(currentIon) < (float(compareIon) + 17.96611) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "H2O-HCl _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 18.01057) - MZChange and
float(compareIon) < (float(currentIon) + 18.01057) + MZChange or \
                       float(currentIon) > (float(compareIon) + 18.01057) - MZChange and
float(currentIon) < (float(compareIon) + 18.01057) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "M-H2O _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 20.92933) - MZChange and
float(compareIon) < (float(currentIon) + 20.92933) + MZChange or \
                       float(currentIon) > (float(compareIon) + 20.92933) - MZChange and
float(currentIon) < (float(compareIon) + 20.92933) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "K-NH4 _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 21.98194) - MZChange and
float(compareIon) < (float(currentIon) + 21.98194) + MZChange or \
                       float(currentIon) > (float(compareIon) + 21.98194) - MZChange and
float(currentIon) < (float(compareIon) + 21.98194) + MZChange:
```

```
                        filterList[n1][0] = mFNum
                        #print "Na-M _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 35.03712) - MZChange and
float(compareIon) < (float(currentIon) + 35.03712) + MZChange or \
                       float(currentIon) > (float(compareIon) + 35.03712) - MZChange and
float(currentIon) < (float(compareIon) + 35.03712) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "NH4-H2O _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 35.97668) - MZChange and
float(compareIon) < (float(currentIon) + 35.97668) + MZChange or \
                       float(currentIon) > (float(compareIon) + 35.97668) - MZChange and
float(currentIon) < (float(compareIon) + 35.97668) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "M-HCl _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 37.95588) - MZChange and
float(compareIon) < (float(currentIon) + 37.95588) + MZChange or \
                       float(currentIon) > (float(compareIon) + 37.95588) - MZChange and
float(currentIon) < (float(compareIon) + 37.95588) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "K-M _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 39.99251) - MZChange and
float(compareIon) < (float(currentIon) + 39.99251) + MZChange or \
                       float(currentIon) > (float(compareIon) + 39.99251) - MZChange and
float(currentIon) < (float(compareIon) + 39.99251) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "Na-H2O _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 53.00323) - MZChange and
float(compareIon) < (float(currentIon) + 53.00323) + MZChange or \
                       float(currentIon) > (float(compareIon) + 53.00323) - MZChange and
float(currentIon) < (float(compareIon) + 53.00323) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "NH4-HCl _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 55.96645) - MZChange and
float(compareIon) < (float(currentIon) + 55.96645) + MZChange or \
                       float(currentIon) > (float(compareIon) + 55.96645) - MZChange and
float(currentIon) < (float(compareIon) + 55.96645) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "K-H2O _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 57.95862) - MZChange and
float(compareIon) < (float(currentIon) + 57.95862) + MZChange or \
                       float(currentIon) > (float(compareIon) + 57.95862) - MZChange and
float(currentIon) < (float(compareIon) + 57.95862) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "Na-HCl _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 61.9156) - MZChange and
float(compareIon) < (float(currentIon) + 61.9156) + MZChange or \
                       float(currentIon) > (float(compareIon) + 61.9156) - MZChange and
float(currentIon) < (float(compareIon) + 61.9156) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "H2O-HBr _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 61.94625) - MZChange and
float(compareIon) < (float(currentIon) + 61.94625) + MZChange or \
                       float(currentIon) > (float(compareIon) + 61.94625) - MZChange and
float(currentIon) < (float(compareIon) + 61.94625) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "H2O-SO3 _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 73.93256) - MZChange and
float(compareIon) < (float(currentIon) + 73.93256) + MZChange or \
                       float(currentIon) > (float(compareIon) + 73.93256) - MZChange and
float(currentIon) < (float(compareIon) + 73.93256) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "K-HCl _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 79.92617) - MZChange and
```

```
float(compareIon) < (float(currentIon) + 79.92617) + MZChange or \
                    float(currentIon) > (float(compareIon) + 79.92617) - MZChange and
float(currentIon) < (float(compareIon) + 79.92617) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "M-HBr _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 79.95682) - MZChange and
float(compareIon) < (float(currentIon) + 79.95682) + MZChange or \
                    float(currentIon) > (float(compareIon) + 79.95682) - MZChange and
float(currentIon) < (float(compareIon) + 79.95682) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "M-SO3 _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 96.95272) - MZChange and
float(compareIon) < (float(currentIon) + 96.95272) + MZChange or \
                    float(currentIon) > (float(compareIon) + 96.95272) - MZChange and
float(currentIon) < (float(compareIon) + 96.95272) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "NH4-HBr _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 96.98337) - MZChange and
float(compareIon) < (float(currentIon) + 96.98337) + MZChange or \
                    float(currentIon) > (float(compareIon) + 96.98337) - MZChange and
float(currentIon) < (float(compareIon) + 96.98337) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "NH4-SO3 _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 101.90811) - MZChange and
float(compareIon) < (float(currentIon) + 101.90811) + MZChange or \
                    float(currentIon) > (float(compareIon) + 101.90811) - MZChange and
float(currentIon) < (float(compareIon) + 101.90811) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "Na-HBr _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 101.93876) - MZChange and
float(compareIon) < (float(currentIon) + 101.93876) + MZChange or \
                    float(currentIon) > (float(compareIon) + 101.93876) - MZChange and
float(currentIon) < (float(compareIon) + 101.93876) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "Na-SO3 _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 117.88205) - MZChange and
float(compareIon) < (float(currentIon) + 117.88205) + MZChange or \
                    float(currentIon) > (float(compareIon) + 117.88205) - MZChange and
float(currentIon) < (float(compareIon) + 117.88205) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "K-HBr _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 117.9127) - MZChange and
float(compareIon) < (float(currentIon) + 117.9127) + MZChange or \
                    float(currentIon) > (float(compareIon) + 117.9127) - MZChange and
float(currentIon) < (float(compareIon) + 117.9127) + MZChange:
                        filterList[n1][0] = mFNum
                        #print "K-SO3 _ " + str(currentIon) + " _ " + str(compareIon)
                n1 = n1 + 1
            n = n + 1
            n1 = 0
        n = 0
        n1 = 1
        n2 = 0
        #Loop to search for the mass difference between a double charged ion and adducts of
the single charged ion
        while n < len(filterList):
            if filterList[n][4] != 0 and  filterList[n][0] != 0:
                currentIon = float(filterList[n][1]) * float(filterList[n][4]) - (1.00727 *
(float(filterList[n][4]) - 1))
                MZChange = currentIon / 1000000 * ppmError
                currentRT = filterList[n][2]
                mFNum = filterList[n][0]
            else:
```

```
                n1 = len(filterList)
            while n1 < len(filterList):
                compareIon = float(filterList[n1][1])
                if float(filterList[n1][2]) < (currentRT + RTChange) and \
                        float(filterList[n1][2]) > (currentRT - RTChange):
                    if float(compareIon) > (float(currentIon) + 18.01057) - MZChange and
float(compareIon) < (float(currentIon) + 18.01057) + MZChange or \
                       float(currentIon) > (float(compareIon) + 18.01057) - MZChange and
float(currentIon) < (float(compareIon) + 18.01057) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - H2O-M _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 17.02655) - MZChange and
float(compareIon) < (float(currentIon) + 17.02655) + MZChange or \
                       float(currentIon) > (float(compareIon) + 17.02655) - MZChange and
float(currentIon) < (float(compareIon) + 17.02655) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - M-NH4 _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 4.95539) - MZChange and
float(compareIon) < (float(currentIon) + 4.95539) + MZChange or \
                       float(currentIon) > (float(compareIon) + 4.95539) - MZChange and
float(currentIon) < (float(compareIon) + 4.95539) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - NH4-Na _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 15.97394) - MZChange and
float(compareIon) < (float(currentIon) + 15.97394) + MZChange or \
                       float(currentIon) > (float(compareIon) + 15.97394) - MZChange and
float(currentIon) < (float(compareIon) + 15.97394) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - Na-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 35.03712) - MZChange and
float(compareIon) < (float(currentIon) + 35.03712) + MZChange or \
                       float(currentIon) > (float(compareIon) + 35.03712) - MZChange and
float(currentIon) < (float(compareIon) + 35.03712) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - H2O-NH4 _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 21.98194) - MZChange and
float(compareIon) < (float(currentIon) + 21.98194) + MZChange or \
                       float(currentIon) > (float(compareIon) + 21.98194) - MZChange and
```

```
float(currentIon) < (float(compareIon) + 21.98194) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - M-Na _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 20.92933) - MZChange and
float(compareIon) < (float(currentIon) + 20.92933) + MZChange or \
                       float(currentIon) > (float(compareIon) + 20.92933) - MZChange and
float(currentIon) < (float(compareIon) + 20.92933) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - NH4-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 39.99251) - MZChange and
float(compareIon) < (float(currentIon) + 39.99251) + MZChange or \
                       float(currentIon) > (float(compareIon) + 39.99251) - MZChange and
float(currentIon) < (float(compareIon) + 39.99251) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - H2O-Na _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 37.95588) - MZChange and
float(compareIon) < (float(currentIon) + 37.95588) + MZChange or \
                       float(currentIon) > (float(compareIon) + 37.95588) - MZChange and
float(currentIon) < (float(compareIon) + 37.95588) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - M-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 55.96645) - MZChange and
float(compareIon) < (float(currentIon) + 55.96645) + MZChange or \
                       float(currentIon) > (float(compareIon) + 55.96645) - MZChange and
float(currentIon) < (float(compareIon) + 55.96645) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2H - H2O-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                n1 = n1 + 1
            n = n + 1
            n1 = 0
        n = 0
        n1 = 1
        n2 = 0
        #Loop to search for the mass difference between a dimer and adducts of the monomer
        while n < len(filterList):
            if filterList[n][4] != 0 and  filterList[n][0] != 0:
                currentIon = float(filterList[n][1]) * float(filterList[n][4]) - (1.00727 *
(float(filterList[n][4]) - 1))
```

```
                MZChange = currentIon / 1000000 * ppmError
                currentRT = filterList[n][2]
                mFNum = filterList[n][0]
            else:
                n1 = len(filterList)
            while n1 < len(filterList):
                compareIon = ((float(filterList[n1][1]) * float(filterList[n1][4]) -
(1.00727 * (float(filterList[n1][4]) - 1))) - 1.00727) / 2 + 1.00727
                if float(filterList[n1][2]) < (currentRT + RTChange) and \
                        float(filterList[n1][2]) > (currentRT - RTChange):
                    if float(compareIon) > (float(currentIon) + 9.005284) - MZChange and
float(compareIon) < (float(currentIon) + 9.005284) + MZChange or \
                       float(currentIon) > (float(compareIon) + 9.005284) - MZChange and
float(currentIon) < (float(compareIon) + 9.005284) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - H2O-M _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 8.513275) - MZChange and
float(compareIon) < (float(currentIon) + 8.513275) + MZChange or \
                       float(currentIon) > (float(compareIon) + 8.513275) - MZChange and
float(currentIon) < (float(compareIon) + 8.513275) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - M-NH4 _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 2.4776949) - MZChange and
float(compareIon) < (float(currentIon) + 2.4776949) + MZChange or \
                       float(currentIon) > (float(compareIon) + 2.4776949) - MZChange and
float(currentIon) < (float(compareIon) + 2.4776949) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - NH4-Na _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 7.98697) - MZChange and
float(compareIon) < (float(currentIon) + 7.98697) + MZChange or \
                       float(currentIon) > (float(compareIon) + 7.98697) - MZChange and
float(currentIon) < (float(compareIon) + 7.98697) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - Na-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 17.51856) - MZChange and
float(compareIon) < (float(currentIon) + 17.51856) + MZChange or \
                       float(currentIon) > (float(compareIon) + 17.51856) - MZChange and
float(currentIon) < (float(compareIon) + 17.51856) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
```

```
                        #print "2M - H2O-NH4 _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 10.99097) - MZChange and
float(compareIon) < (float(currentIon) + 10.99097) + MZChange or \
                       float(currentIon) > (float(compareIon) + 10.99097) - MZChange and
float(currentIon) < (float(compareIon) + 10.99097) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - M-Na _ " + str(currentIon) + " _ " + str(compareIon)
                    elif float(compareIon) > (float(currentIon) + 10.464665) - MZChange and
float(compareIon) < (float(currentIon) + 10.464665) + MZChange or \
                       float(currentIon) > (float(compareIon) + 10.464665) - MZChange and
float(currentIon) < (float(compareIon) + 10.464665) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - NH4-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 19.996255) - MZChange and
float(compareIon) < (float(currentIon) + 19.996255) + MZChange or \
                       float(currentIon) > (float(compareIon) + 19.996255) - MZChange and
float(currentIon) < (float(compareIon) + 19.996255) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - H2O-Na _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 18.97794) - MZChange and
float(compareIon) < (float(currentIon) + 18.97794) + MZChange or \
                       float(currentIon) > (float(compareIon) + 18.97794) - MZChange and
float(currentIon) < (float(compareIon) + 18.97794) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - M-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                    elif float(compareIon) > (float(currentIon) + 27.983225) - MZChange and
float(compareIon) < (float(currentIon) + 27.983225) + MZChange or \
                       float(currentIon) > (float(compareIon) + 27.983225) - MZChange and
float(currentIon) < (float(compareIon) + 27.983225) + MZChange:
                        while n2 < len(filterList):
                            if filterList[n2][0] == filterList[n1][0]:
                                filterList[n2][0] = mFNum
                            n2 = n2 + 1
                        n2 = 0
                        #print "2M - H2O-K _ " + str(filterList[n][1]) + " _ " +
str(filterList[n1][1])
                n1 = n1 + 1
            n = n + 1
            n1 = 0
        primary.masterList = filterList
        return
    def applyFilter(primary):
```

```
        filterList = primary.masterList
        filterMode = primary.selectedMode.get()
        areaCutOff = float(primary.areaVar.get())
        minRT = float(primary.minRTVar.get())
        minHitRate = float(primary.requireVar.get())
        currentMode = primary.selectedFeature.get()
        n = 0
        n1 = 1
        resultList = []
        i = 0
        listForID = []
        listForMZ = []
        listForRT = []
        listForPA = []
        comboList = []
        MFList = []
        testMF = 0
        tempTestMF = ""
        tempCurrentMF = ""
        testMFAll = ""
        currentMF = 0
        ID = 0
        PA = 0
        MZ = 0
        RT = 0
        counter = 0
        peakA1 = 0
        peakA2 = 0
        peakA3 = 0
        massA1 = 0
        massA2 = 0
        massA3 = 0
        isotopeCount = 0
        peakCompare3 = 0
        peakCompare2 = 0
        maxCompare2 = 0
        minCompare2 = 0
        maxCompare3 = 0
        minCompare3 = 0
        minMassComp = 0
        maxMassComp = 0
        maxPeakComp = 0
        minPeakComp = 0
        massCompare = 0
        massCompareA1 = 0
        sulphMinPeakComp = 0
        n = 0
        haloHit = 0
        while n < len(filterList):
            peakA1 = float(filterList[n][3])
            if float(filterList[n][4]) != 0:
                massA1 = (float(filterList[n][1])*float(filterList[n][4]))-
((float(filterList[n][4])-1)*1.00728)
                if filterList[n][5].count(':') > 1:
                    isotopeCount = filterList[n][5].count(':')
                    peakA2 = filterList[n][5]
                    peakA2 = peakA2.split(':')
                    peakA2 = peakA2[0]
                    peakA2 = peakA2.split('-')
                    peakA2 = peakA2[2]
                    peakA3 = filterList[n][5]
                    peakA3 = peakA3.split(':')
                    peakA3 = peakA3[1]
```

```
                    peakA3 = peakA3.split('-')
                    peakA3 = peakA3[2]
                    massA1 = (float(filterList[n][1])*float(filterList[n][4]))-
((float(filterList[n][4])-1)*1.00728)
                    massA2 = filterList[n][5]
                    massA2 = massA2.split(':')
                    massA2 = massA2[0]
                    massA2 = massA2.split('-')
                    massA2 = (float(massA2[0])*float(filterList[n][4]))-
((float(filterList[n][4])-1)*1.00728)
                    massA3 = filterList[n][5]
                    massA3 = massA3.split(':')
                    massA3 = massA3[1]
                    massA3 = massA3.split('-')
                    massA3 = (float(massA3[0])*float(filterList[n][4]))-
((float(filterList[n][4])-1)*1.00728)
                    peakCompare3 = float(peakA3)/float(peakA1)
                    massCompare = float(massA3)-float(massA2)
                    massCompareA1 = float(massA2)-float(massA1)
                    minMassComp = 0
                    maxMassComp = 0
                    if "0-0-0" in str(filterList[n][5]):
                        filterList[n][5] = string.replace(str(filterList[n][5]),"0-0-
0:","")
                    if filterMode == "DCA-Hal":
                        maxPeakComp = 1.611 * 10**(-7) * float(massA1)**2 - 1.319 * 10**(-
5) * float(massA1) + 12.05
                        minPeakComp = 1.611 * 10**(-7) * float(massA1)**2 - 1.319 * 10**(-
5) * float(massA1) + 0.2702 #Original 5% Error
                        maxMassComp = 1.5644 * 10**(-23) * float(massA1)**6 - 2.46 * 10**(-
19) * float(massA1)**5 + 1.5135 * 10**(-15) * float(massA1)**4 - \
                            4.485 * 10**(-12) * float(massA1)**3 + 5.954 * 10**(-9) *
float(massA1)**2 - 9.019 * 10**(-7) * float(massA1) + 0.99832
                        minMassComp = 0.9936 - (float(massA1) / 1000000 * 5)
                    elif  filterMode == "DCA-Sul":
                        maxPeakComp = 1.611 * 10**(-7) * float(massA1)**2 - 1.319 * 10**(-
5) * float(massA1) + 0.2702 #Original 5% Error
                        minPeakComp = 1.611 * 10**(-7) * float(massA1)**2 - 7.982 * 10**(-
6) * float(massA1) + 0.00471
                        maxMassComp = 4.288 * 10**(-23) * float(massA1)**6 - 4.975 * 10**(-
19) * float(massA1)**5 + 2.0086 * 10**(-15) * float(massA1)**4 - \
                            2.735 * 10**(-12) * float(massA1)**3 - 2.07 * 10**(-9) *
float(massA1)**2 + 8.454 * 10**(-6) * float(massA1) + 0.99684
                        minMassComp = 0.9936 - (float(massA1) / 1000000 * 5)
                    elif  filterMode == "DCA-C":
                        maxPeakComp = 1.611 * 10**(-7) * float(massA1)**2 - 1.319 * 10**(-
5) * float(massA1) + 0.2702
                        minPeakComp = 0.0005
                        maxMassComp = 1.006
                        minMassComp = 4.288 * 10**(-23) * float(massA1)**6 - 4.975 * 10**(-
19) * float(massA1)**5 + 2.0086 * 10**(-15) * float(massA1)**4 - \
                            2.735 * 10**(-12) * float(massA1)**3 - 2.07 * 10**(-9) *
float(massA1)**2 + 8.454 * 10**(-6) * float(massA1) + 0.99684
                    elif  filterMode == "DCA-Bor":
                        maxPeakComp = 100
                        minPeakComp = 0
                        if float(massA1) < 2000:
                            maxMassComp = -
0.0000000008575*float(massA1)**2+0.000004535*float(massA1) +0.996
                        else:
                            maxMassComp = float(massA1)*0.00000108+0.9995
                        minMassComp = 0.995
                        massCompare = massCompareA1
```

```
                    #if filterMode == "All Features":
                    #    if currentMode == "Molecular":
                    #        resultList.append(str(filterList[n][0]) + "_" +
str(filterList[n][1]) + "-" + str(filterList[n][2]) + \
                    #                "-" + str(filterList[n][3]) + ":" +
str(filterList[n][5]))
                    #    else:
                    #        resultList.append(str(filterList[n][1]) + "-" +
str(filterList[n][2]) + \
                    #                "-" + str(filterList[n][3]) + ":" +
str(filterList[n][5]))
                    #    haloHit == 1
                    if haloHit == 0: #elif, if using All Features
                        if peakCompare3 < maxPeakComp:
                            if peakCompare3 > minPeakComp:
                                if massCompare < maxMassComp:
                                    if massCompare > minMassComp:
                                        if currentMode == "Molecular":
                                            resultList.append(str(filterList[n][0]) + "_" +
str(filterList[n][1]) + "-" + str(filterList[n][2]) + \
                                                    "-" + str(filterList[n][3]) + ":" +
str(filterList[n][5]))
                                        else:
                                            resultList.append(str(filterList[n][1]) + "-" +
str(filterList[n][2]) + \
                                                    "-" + str(filterList[n][3]) + ":" +
str(filterList[n][5]))
                                        haloHit = 1
                    #Use below to invert results. Also need to remove the results appended
above....
                    #if haloHit == 0:
                        #resultList.append(str(filterList[n][1]) + "-" +
str(filterList[n][2]) + \
                                #"-" + str(filterList[n][3]) + ":" + str(filterList[n][5]))

            haloHit = 0
            n = n + 1
        if currentMode == "Molecular":
            n = 0
            n1 = 1
            totalNumFromFilter = 1
            while n < len(resultList):
                currentMF = resultList[n]
                currentMF = currentMF.split('_')
                currentMF = currentMF[0]
                while n1 < len(resultList):
                    testMF = resultList[n1]
                    testMF = testMF.split('_')
                    testMF = testMF[0]
                    if testMF == currentMF:
                        tempTestMF = resultList[n1]
                        tempTestMF = tempTestMF.split('_')
                        tempTestMF = tempTestMF[1]
                        testMFAll = testMFAll + tempTestMF
                        resultList[n1] = "0_0-0-0:"
                        totalNumFromFilter = totalNumFromFilter + 1
                    n1 = n1 + 1
                if float(currentMF) > 0:
                    tempCurrentMF = resultList[n]
                    MFList.append(str(totalNumFromFilter) + "+" + tempCurrentMF +
testMFAll)
                testMFAll = ""
                totalNumFromFilter = 1
```

```
                n = n + 1
                n1 = n + 1
            resultList = MFList
            n = 0
            n1 = 0
            totalNumInMF = 0
            probabilityOfHit = 0
            MissedCFList = []
            hitRateList = []
            #Loop through resultsList to compare molecular feature number to that of the
filterList
            #to see if any chemical features were missed by the filter equations
            while n < len(resultList):
                currentMF = resultList[n]
                currentMF = currentMF.split('_')
                currentMF = currentMF[0]
                currentMF = currentMF.split('+')
                currentMF = currentMF[1]
                while n1 < len(filterList):
                    testMF = filterList[n1][0]
                    if str(testMF) == str(currentMF):
                        totalNumInMF = totalNumInMF + 1
                        tempTestMF = (str(filterList[n1][1]) + "-" + str(filterList[n1][2])
+ \
                                      "-" + str(filterList[n1][3]) + ":" +
str(filterList[n1][5]))
                        testMFAll = testMFAll + tempTestMF
                    n1 = n1 + 1
                totalNumFromFilter = resultList[n]
                totalNumFromFilter = totalNumFromFilter.split('_')
                totalNumFromFilter = totalNumFromFilter[0]
                totalNumFromFilter = totalNumFromFilter.split('+')
                totalNumFromFilter = totalNumFromFilter[0]
                probabilityOfHit = float(totalNumFromFilter) / float(totalNumInMF) * 100
                probabilityOfHit = '%.1f' % (float(probabilityOfHit))
                if float(probabilityOfHit) >= minHitRate:
                    MissedCFList.append(testMFAll)
                    hitRateList.append(float(probabilityOfHit))
                n1 = 0
                n = n + 1
                testMFAll = ""
                totalNumFromFilter = 0
                totalNumInMF = 0
            resultList = MissedCFList
        n = 0
        n1 = 0
        currentPA = 0
        currentMaxPA = 0
        isoNum = 0
        while n < len(resultList):
            currentMaxPA = 0
            isoNum = resultList[n]
            isoNum = isoNum.split(':')
            isoNum = len(isoNum) - 1
            while n1 < isoNum:
                currentPA = resultList[n]
                currentPA = currentPA.split(':')
                currentPA = currentPA[n1]
                currentPA = currentPA.split('-')
                currentPA = currentPA[2]
                if float(currentPA) > float(currentMaxPA):
                    currentMaxPA = currentPA
                n1 = n1 + 1
```

```
            if float(currentMaxPA) < float(areaCutOff):
                resultList[n] = 0
                if currentMode == "Molecular":
                    hitRateList[n] = 0
            n = n + 1
            n1 = 0
        n = 0
        n1 = 1
        currentRT = 0
        testRT = 0
        resultList[:] = (value for value in resultList if value != 0)
        if currentMode == "Molecular":
            hitRateList[:] = (value for value in hitRateList if value != 0)
        resultMZ = []
        resultRT = []
        resultPA = []
        final1 = []
        n = 0
        n1 = 0
        splitTheList = []
        completeRT = []
        currentAbund = 0
        testAbund = 0
        isotopeCount = 0
        printList = []
        i = 0
        pLCount = 0
        printList.append("0")
        while n < len(resultList):
            splitTheList = resultList[n]
            splitTheList = splitTheList.split(':')
            while n1 < (len(splitTheList)-1):
                testAbund = splitTheList[n1]
                testAbund = testAbund.split('-')
                testAbund = testAbund[2]
                if float(testAbund) > float(currentAbund):
                    currentAbund = testAbund
                    resultMZ = splitTheList[n1]
                    resultMZ = resultMZ.split('-')
                    resultMZ = resultMZ[0]
                n1 = n1 + 1

            resultRT = resultList[n]
            resultRT = resultRT.split(':')
            resultRT = resultRT[0]
            resultRT = resultRT.split('-')
            resultRT = resultRT[1]
            completeRT.append(float(resultRT))
            resultPA = resultList[n]
            resultPA = resultPA.split(':')
            resultPA = resultPA[0]
            resultPA = resultPA.split('-')
            resultPA = resultPA[2]
            resultMZ = str(resultMZ) + ":" + str(resultRT) + ":" + str(resultPA)
            final1.append(resultMZ)
            testAbund = 0
            currentAbund = 0
            n1 = 0
            n = n + 1
        if len(final1) > 1:
            ## Sorts the two lists by RT
            if currentMode == "Molecular":
                completeRT, final1, resultList, hitRateList = zip(*sorted(zip(completeRT,
```

```
final1, resultList, hitRateList)))
            else:
                completeRT, final1, resultList = zip(*sorted(zip(completeRT, final1,
resultList)))
        n = 0
        primary.txt.configure(state="normal")
        primary.txt.delete(1.0, END)
        while n < len(final1):
            primary.txt.insert('1.0', str(final1[n]) + '\n')# + '\n')
            n = n + 1
        primary.txt.insert('1.0', "m/z:retention time:peak area" + '\n' +  '\n')
        primary.txt.configure(state="disabled")
        simplifyRT = ""
        simplifyMZ = ""
        simplifyMZ1 = ""
        ionChoices1 = []
        del ionChoices1[:]
        n = 0
        for each in final1:
            simplifyMZ = final1[n].split(':')
            simplifyMZ = simplifyMZ[0]
            simplifyMZ = '%.1f' % (float(simplifyMZ))
            resultSimplify = final1[n].split(':')
            resultSimplify = resultSimplify[1]
            resultSimplify = float(resultSimplify) / 60
            resultSimplify = '%.2f' % (float(resultSimplify))
            ionChoices1.append(str(simplifyMZ) + " at " + str(resultSimplify))
            n = n + 1
        n = 0
        for each in ionChoices1:
            ionChoices1[n] = str(n+1) + ": " + str(ionChoices1[n]) + " min"
            n = n + 1
        if len(ionChoices1) == 0:
            ionChoices1.append("None Detected")
        primary.sepOptions.set(ionChoices1[0])
        primary.sepOptions["values"] = ionChoices1
        primary.resultsMasterList = resultList
        if currentMode == "Molecular":
            primary.resultsMasterRateList = hitRateList
        return
    def grph(primary, event):
        currentMode = primary.selectedFeature.get()
        resultsForPrint = []
        resultsForPrint = copy.deepcopy(primary.resultsMasterList)
        filtetedList = []
        if currentMode == "Molecular":
            rateList = []
            rateList = copy.deepcopy(primary.resultsMasterRateList)
            selectedRate = 0
        selectedResult = primary.selectedSep.get()
        if selectedResult == "NA":
            return
        elif selectedResult == "None Detected":
            return
        selectedResult = selectedResult.split(':')
        selectedResult = selectedResult[0]
        selectedResult = int(selectedResult) - 1
        filtetedList.append(resultsForPrint[int(selectedResult)])
        if currentMode == "Molecular":
            selectedRate = rateList[int(selectedResult)]
        massList = []
        abundanceList = []
        mzResult = []
```

```
        abResult = []
        i = 0
        n = 0
        while n < len(filtetedList):
            isoCount = filtetedList[n].count(':')
            while i < isoCount:
                mzResult = filtetedList[n]
                mzResult = mzResult.split(':')
                mzResult = mzResult[i]
                mzResult = mzResult.split('-')
                mzResult = mzResult[0]
                abResult = filtetedList[n]
                abResult = abResult.split(':')
                abResult = abResult[i]
                abResult = abResult.split('-')
                abResult = abResult[2]
                massList.append(mzResult)
                abundanceList.append(abResult)
                i = i + 1
            i = 0
            n = n + 1
        currentMass = 0
        resNum = 15000
        resNum = 1/float(resNum)*225
        maxCounts = 0
        n = 0
        for each in abundanceList:
            if maxCounts < float(abundanceList[n]):
                maxCounts = float(abundanceList[n])
            n = n + 1
        n = 0
        for each in abundanceList:
            abundanceList[n] = float(abundanceList[n]) / float(maxCounts) * 100
            n = n + 1
        intNum = 0
        n = 0
        maxCounts = 100
        maxMass = 0
        minMass = 1000000000
        resVal = StringVar()
        resVal = 15000
        labelHeight = 0
        primary.ax2.clear()
        colorNum = 0.2
        colorChange = 0
        intNum = 0
        while intNum < len(massList):
            if float(massList[intNum]) > float(maxMass):
                maxMass = float(massList[intNum])  + 2
            if float(massList[intNum]) < float(minMass):
                minMass = float(massList[intNum])  - 1
            intNum = intNum + 1
        labelHeight = (100+100/4)*0.14
        intNum = 0
        colorsMap = [(0.976,0.333,0.518), (0.976,0.333,0.518), (0.976,0.333,0.518)]
        while intNum < len(abundanceList):
            primary.ax2.plot([float(massList[intNum]) -
(150/float(resVal)),float(massList[intNum])], [0,(float(abundanceList[intNum]))], color =
(colorNum,1-colorNum,0.952)) #"b")
            primary.ax2.plot([float(massList[intNum]) +
(150/float(resVal)),float(massList[intNum])], [0,(float(abundanceList[intNum]))], color =
(colorNum,1-colorNum,0.952)) #"b")
            primary.ax2.plot([0,30000], [0,0], "b", linewidth=0.5)
```

```
            primary.ax2.text(float(massList[intNum]),
(float(abundanceList[intNum]))+labelHeight, str(format(float(massList[intNum]),'.4f')),
fontsize=9, rotation=80)
            primary.ax2.set_xlabel('Mass to charge (m/z)')
            primary.ax2.set_title('Isotope Pattern')
            primary.ax2.set_ylabel('Relative Abundance')
            if colorChange == 0:
                colorNum = colorNum + 0.02
            if colorChange == 1:
                colorNum = colorNum - 0.02
            if colorNum > 1:
                colorNum = colorNum - 0.04
                colorChange = 1
            if colorNum < 0.2:
                colorNum = colorNum + 0.04
                colorChange = 0
            intNum = intNum + 1
        primary.ax2.set_ylim( ((-(100+100/4))/50,maxCounts+maxCounts/4) )
        primary.ax2.set_xlim( (minMass - ((maxMass-minMass)*0.1), maxMass + ((maxMass-
minMass)*0.1)) )
        if currentMode == "Molecular":
            primary.ax2.text(maxMass - ((maxMass-minMass)*0.15), 110, "Hit rate:" +
str(selectedRate), fontsize=12) #, rotation=80)
        primary.canvas.draw()
        return
if __name__ == "__main__":
    app = simpleapp_tk(None)
    app.title('DCAnalysis 1.07')
    img = PhotoImage(data=iconData)
    app.tk.call('wm', 'iconphoto', app._w, img)
    app.mainloop()
```

Code S-1: *Source code for DCAnalysis v1.07*

This programme performs dynamic cluster analysis on chromatographic mass spectrometric data in the form of CSV files.

```
from Tkinter import *
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from pylab import figure, show
import matplotlib.pyplot as plt
import Tkinter
import numpy
import math
import ttk
import tkFileDialog
import csv
import copy
import string
iconData = '''R0lGODlhEAAQAPcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArMwArZgArmQArzAAr/wBVAABVMwBV
        ZgBVmQBVzABV/wCAAACAMwCAZgCAmQCAzACA/wCqAACqMwCqZgCqmQCqzACq/wDVAADVMwDVZgDV
        mQDVzADV/wD/AAD/MwD/ZgD/mQD/zAD//zMAADMAMzMAZjMAmTMAzDMA/zMrADMrMzMrZjMrmTMr
        zDMr/zNVADNVMzNVZjNVmTNVzDNV/zOAADOAMzOAZjOAmTOAzDOA/zOqADOqMzOqZjOqmTOqzDOq
        /zPVADPVMzPVZjPVmTPVzDPV/zP/ADP/MzP/ZjP/mTP/zDP//2YAAGYAM2YAZmYAmWYAzGYA/2Yr
        AGYrM2YrZmYrmWYrzGYr/2ZVAGZVM2ZVZmZVmWZVzGZV/2aAAGaAM2aAZmaAmWaAzGaA/2aqAGaq
        M2aqZmaqmWaqzGaq/2bVAGbVM2bVZmbVmWbVzGbV/2b/AGb/M2b/Zmb/mWb/zGb//5kAAJkAM5kA
        ZpkAmZkAzJkA/5krAJkrM5krZpkrmZkrzJkr/5lVAJlVM5lVZplVmZlVzJlV/5mAAJmAM5mAZpmA
        mZmAzJmA/5mqAJmqM5mqZpmqmZmqzJmq/5nVAJnVM5nVZpnVmZnVzJnV/5n/AJn/M5n/Zpn/mZn/
        zJn//8wAAMwAM8wAZswAmcwAzMwA/8wrAMwrM8wrZswrmcwrzMwr/8xVAMxVM8xVZsxVmcxVzMxV
        /8yAAMyAM8yAZsyAmcyAzMyA/8yqAMyqM8yqZsyqmcyqzMyq/8zVAMzVM8zVZszVmczVzMzV/8z/
        AMz/M8z/Zsz/mcz/zMz///8AAP8AM/8AZv8Amf8AzP8A//8rAP8rM/8rZv8rmf8rzP8r//9VAP9V
        M/9VZv9Vmf9VzP9V//+AAP+AM/+AZv+Amf+AzP+A//+qAP+qM/+qZv+qmf+qzP+q///VAP/VM//V
        Zv/Vmf/VzP/V////AP//M///Zv//mf//zP///wAAAAAAAAAAAAAAACH5BAEAAPwALAAAAAAQABAA
        AAiKADt04CCQ4MCCCAlaOLiQwwWCDiEyHMihYoeHBzs0pFiRA4CCDy0OvFAQgEmKAjUe9Ghyn0mL
        EgWaBLCv5kuFHlm2rOkSAByCM2n25NnzQsudQ3kGrRlDaFKbAAwEnUq1alUiGjRkyDBEA5GvQ4Zk
        wOr169YPRJSYFUsEBIivH5KETfI1idywXwMCADs=
        '''
class simpleapp_tk(Tkinter.Tk):
    def __init__(primary,parent):
        Tkinter.Tk.__init__(primary,parent)
        primary.parent = parent
        primary.initialize()
        primary.update()
        primary.minsize(primary.winfo_width(), primary.winfo_height())
    def initialize(primary):
        primary.grid()
        primary.grid_rowconfigure(10,weight=1)
        primary.grid_columnconfigure(3,weight=1)
        primary.massList = []
        primary.abundanceList = []
        primary.rawMasterList = []
        primary.resultsMasterList = []
        primary.rSel = StringVar()
        primary.resVar = StringVar()
        primary.selectedMode = StringVar()
        color = '0.9'
        plotFrame = Frame(primary, width=300, height=300)
        plotFrame.pack(fill="both", expand=True)
        plotFrame.grid_propagate(False)
        plotFrame.grid_rowconfigure(10, weight=1)
        plotFrame.grid_columnconfigure(3, weight=1)
        plotFrame.grid(row=6, column=1, columnspan=2, rowspan=4, sticky="neswn")
        fig = Figure()
        rect = fig.patch
        rect.set_facecolor(color)
        primary.CSVBtn = Button(primary, text='Import CSV', command=primary.getFileName)
        primary.CSVBtn.grid(row=1, column=1, columnspan=2, pady=(5,5))
```

```
        primary.resVarLabel = Label(primary, text="Resolution:")
        primary.resVarLabel.grid(row=3, column=1, sticky=E, pady=(0,5))
        primary.resVarVal = Entry(primary, textvariable=primary.resVar, width=15)
        primary.resVarVal.grid(row=3, column=2, sticky=W)
        primary.resVarVal.insert(0, 50000)
        primary.modeLabel = Label(primary, text="Mode:")
        primary.modeLabel.grid(row=2, column=1, sticky=E)
        primary.selectedMode.trace('w', primary.ModeRefresh)
        primary.modeOptions = ttk.Combobox(primary, state='readonly', width=15,
textvariable=primary.selectedMode)
        primary.modeOptions["values"] = ('Centroid') #, 'Gaussian')
        primary.modeOptions.set("Centroid")
        primary.modeOptions.grid(row=2, column=2, columnspan=1, sticky=SW)
        primary.findBtn = Button(primary, text='Extract Formula List',
command=primary.findIsoAndComps)
        primary.findBtn.grid(row=5, column=1, columnspan=2, pady=(0,0))
        primary.txt = Text(primary, borderwidth=3, relief="sunken")
        primary.txt.config(font=("consolas", 9), undo=True, wrap='word', width=20,
height=20)
        primary.txt.grid(row=6, column=1, columnspan=3, rowspan=5, sticky="neswn", padx=2,
pady=5)
        primary.txt.configure(state="disabled")
        scrollb = Scrollbar(primary, command=primary.txt.yview)
        scrollb.grid(row=6, column=4, rowspan=5, padx=(0,10), pady=0, sticky="neswn")
        primary.txt['yscrollcommand'] = scrollb.set
    def ModeRefresh(primary, *args):
        msMode = primary.selectedMode.get()
        if msMode == "Centroid":
            result = primary.resVarVal.configure(state="disabled")
        if msMode == "Gaussian":
            result = primary.resVarVal.configure(state="normal")
        return
    def getFileName(primary):
        fileName = tkFileDialog.askopenfilename()
        justTheFile = StringVar
        if fileName != "":
            justTheFile = fileName.split('/')
            justTheFile = justTheFile[len(justTheFile)-1]
            app.title('FormExtract 1.01 - ' + justTheFile)
            csvRaw = csv.reader(open(fileName, 'U'), dialect='excel')
            formList = []
            completeList = []
            listLen = 0
            maxRT = 0
            for row in csvRaw:
                formList.append(row[1])
            listLen = len(formList)
            formList = formList[1:listLen]
            n = 0
            primary.txt.configure(state="normal")
            primary.txt.delete(1.0, END)
            for each in formList:
                primary.txt.insert('1.0', str(each) + '\n')
            primary.txt.configure(state="disabled")
            primary.rawMasterList = formList
        return
    def findIsoAndComps(primary):
        rawList = []
        rawList = copy.deepcopy(primary.rawMasterList)
        msResolution = float(primary.resVar.get())
        msMode = primary.selectedMode.get()
        editedList = []
        elementList = []
```

```
        numElement = []
        atomNumList = []
        atomList = []
        currentAbndList = []
        currentMzList = []
        tempAbndList = []
        tempMzList = []
        mzList = []
        relAbndList = []
        resultList = []
        pattern = '([A-Z])([a-z]*)(\d*)'
        pattern2 = '(\d*)'
        n = 0
        x = 0
        y = 0
        primary.txt.configure(state="normal")
        primary.txt.delete(1.0, END)
        primary.txt.configure(state="disabled")
        n1 = 1
        def nCr(n,r):
            f = math.factorial
            return f(n) / f(r) / f(n-r)
        def nCrThree(n,b,c):
            f = math.factorial
            return f(n) / (f(n-(b+c)) * f(b) * f(c))
        def nCrFour(n,b,c,d):
            f = math.factorial
            return f(n) / (f(n-(b+c+d)) * f(b) * f(c) * f(d))
        def nCrFive(n,b,c,d,e):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e)) * f(b) * f(c) * f(d) * f(e))
        def nCrSix(n,b,c,d,e,g):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g)) * f(b) * f(c) * f(d) * f(e) * f(g))
        def nCrSeven(n,b,c,d,e,g,h):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h))
        def nCrEight(n,b,c,d,e,g,h,i):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h) *
f(i))
        def nCrNine(n,b,c,d,e,g,h,i,j):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i+j)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h)
* f(i) * f(j))
        def nCrTen(n,b,c,d,e,g,h,i,j,k):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i+j+k)) * f(b) * f(c) * f(d) * f(e) * f(g) *
f(h) * f(i) * f(j) * f(k))
        while n < len(rawList):
            compound = rawList[n]
            compound = compound + "H"
            numOfHe = 0
            numOfD = 0
            numOfBe = 0
            numOfF = 0
            numOfNa = 0
            numOfAl = 0
            numOfP = 0
            numOfSc = 0
            numOfMn = 0
            numOfCo = 0
            numOfAs = 0
```

```
            numOfY = 0
            numOfNb = 0
            numOfRh = 0
            numOfI = 0
            numOfCs = 0
            numOfPr = 0
            numOfTb = 0
            numOfHo = 0
            numOfTm = 0
            numOfAu = 0
            numOfBi = 0
            numOfAc = 0
            numOfTh = 0
            numOfPa = 0
            oxyNum = 0
            nitNum = 0
            carbNum = 0
            broNum = 0
            chloNum = 0
            sulphNum = 0
            borNum = 0
            ironNum = 0
            phosNum = 0
            ioNum = 0
            fluNum = 0
            nikNum = 0
            magNum = 0
            copNum = 0
            zincNum = 0
            selNum = 0
            telNum = 0
            hydroNum = 0
            t = 0
            t1 = 0
            countEle = 0
            countEle3 = 0
            countEle4 = 0
            countEle5 = 0
            countEle6 = 0
            countEle7 = 0
            countEle8 = 0
            countEle9 = 0
            countEle10 = 0
            abund1 = 1
            abund2 = 0
            abund3 = 0
            abund4 = 0
            abund5 = 0
            abund6 = 0
            abund7 = 0
            abund8 = 0
            abund9 = 0
            abund10 = 0
            mass1 = 0
            mass2 = 0
            mass3 = 0
            mass4 = 0
            mass5 = 0
            mass6 = 0
            mass7 = 0
            mass8 = 0
            mass9 = 0
            mass10 = 0
```

```
        for match in re.finditer(pattern, compound):
            s = match.start()
            e = match.end()
            elementComp = compound[s:e]
            elementList.append(compound[s:e])
        for each in elementList:
            element = ''.join([i for i in elementList[x] if not i.isdigit()])
            numElement = ''.join([i for i in elementList[x] if i.isdigit()])
            if numElement == "":
                numElement = "1"
            elementList[x] = element + "." + numElement
            x = x + 1
        for each in elementList:
            sepEleList = elementList[y].split(".")
            atomNumList.append(int(sepEleList[1]))
            atomList.append(sepEleList[0])
            y = y + 1
        x = 0
        y = 1
        for each in atomList:
            while y < len(atomList):
                if atomNumList[x] == 0:
                    atomList[x] = ""
                if atomList[x] == atomList[y]:
                    atomList[y] = ""
                    atomNumList[x] = atomNumList[x] + atomNumList[y]
                    atomNumList[y] = 0
                y = y + 1
            x = x + 1
            y = x + 1
        x = 0
        for each in atomList:
            if atomList[x] == "He": #4.00260325415
                numOfHe = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "D": #2.0141017778
                numOfD = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Be": #9.0121822
                numOfBe = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "F": #18.99840
                numOfF = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
                fluNum = numOfF
            elif atomList[x] == "Na": #22.98977
                numOfNa = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Al": #26.98153863
                numOfAl = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "P": #30.97376
                numOfP = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
                phosNum = numOfP
```

```
            elif atomList[x] == "Sc": #44.9559119
                numOfSc = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Mn": #54.9380451
                numOfMn = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Co": #58.9331950
                numOfCo = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "As": #74.92160
                numOfAs = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Y": #88.9058483
                numOfY = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Nb": #92.9063781
                numOfNb = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Rh": #102.905504
                numOfRh = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "I": #126.90447
                numOfI = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
                ioNum = numOfI
            elif atomList[x] == "Cs": #132.905451933
                numOfCs = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Pr": #140.9076528
                numOfPr = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Tb": #158.9253468
                numOfTb = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Ho": #164.9303221
                numOfHo = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Tm": #168.9342133
                numOfTm = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Au": #196.9665687
                numOfAu = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Bi": #208.9803987
                numOfBi = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Ac": #227.0277521
```

```
            numOfAc = atomNumList[x]
            atomNumList[x] = 0
            atomList[x] = ""
        elif atomList[x] == "Th": #232.0380553
            numOfTh = atomNumList[x]
            atomNumList[x] = 0
            atomList[x] = ""
        elif atomList[x] == "Pa": #231.0358840
            numOfPa = atomNumList[x]
            atomNumList[x] = 0
            atomList[x] = ""
        x = x + 1
    while 0 in atomNumList: atomNumList.remove(0)
    while "" in atomList: atomList.remove("")
    x = 0
    y = 0
    for each in atomList:
        x = atomNumList[y]
        if atomList[y] == "C":
            abund1 = 0.9893
            abund2 = 0.0107
            mass1 = 12
            mass2 = 13.0033548378
            carbNum = x
        elif atomList[y] == "H":
            abund1 = 0.999885
            abund2 = 0.000115
            mass1 = 1.00782503207
            mass2 = 2.0141017778
            hydroNum = x
        elif atomList[y] == "Li":
            abund1 = 0.0759
            abund2 = 0.9241
            mass1 = 6.015122795
            mass2 = 7.01600455
        elif atomList[y] == "B":
            abund1 = 0.199
            abund2 = 0.801
            mass1 = 10.0129370
            mass2 = 11.0093054
            borNum = x
        elif atomList[y] == "N":
            abund1 = 0.99636
            abund2 = 0.00364
            mass1 = 14.0030740048
            mass2 = 15.0001088982
            nitNum = x
        elif atomList[y] == "Cl":
            abund1 = 0.7576
            abund2 = 0.2424
            mass1 = 34.96885268
            mass2 = 36.96590259
            chloNum = x
        elif atomList[y] == "V":
            abund1 = 0.00250
            abund2 = 0.99750
            mass1 = 49.9471585
            mass2 = 50.9439595
        elif atomList[y] == "Cu":
            abund1 = 0.6915
            abund2 = 0.3085
            mass1 = 62.9295975
```

```
                    mass2 = 64.9277895
                    copNum = x
                elif atomList[y] == "Ga":
                    abund1 = 0.60108
                    abund2 = 0.39892
                    mass1 = 68.9255736
                    mass2 = 70.9247013
                elif atomList[y] == "Br":
                    abund1 = 0.5069
                    abund2 = 0.4931
                    mass1 = 78.9183371
                    mass2 = 80.9162906
                    broNum = x
                elif atomList[y] == "Rb":
                    abund1 = 0.7217
                    abund2 = 0.2783
                    mass1 = 84.911789738
                    mass2 = 86.909180527
                elif atomList[y] == "Ag":
                    abund1 = 0.51839
                    abund2 = 0.48161
                    mass1 = 106.905097
                    mass2 = 108.904752
                elif atomList[y] == "In":
                    abund1 = 0.0429
                    abund2 = 0.9571
                    mass1 = 112.904058
                    mass2 = 114.903878
                elif atomList[y] == "Sb":
                    abund1 = 0.5721
                    abund2 = 0.4279
                    mass1 = 120.903815
                    mass2 = 122.9042140
                elif atomList[y] == "La":
                    abund1 = 0.0009
                    abund2 = 0.99910
                    mass1 = 137.907112
                    mass2 = 138.9063533
                elif atomList[y] == "Eu":
                    abund1 = 0.4781
                    abund2 = 0.5219
                    mass1 = 150.9198502
                    mass2 = 152.9212303
                elif atomList[y] == "Lu":
                    abund1 = 0.9741
                    abund2 = 0.0259
                    mass1 = 174.9407718
                    mass2 = 175.9426863
                elif atomList[y] == "Ta":
                    abund1 = 0.00012
                    abund2 = 0.99988
                    mass1 = 179.9474648
                    mass2 = 180.9479958
                elif atomList[y] == "Re":
                    abund1 = 0.3740
                    abund2 = 0.6260
                    mass1 = 184.9529550
                    mass2 = 186.9557531
                elif atomList[y] == "Ir":
                    abund1 = 0.373
                    abund2 = 0.627
                    mass1 = 190.9605940
                    mass2 = 192.9629264
```

```
                elif atomList[y] == "Tl":
                    abund1 = 0.2952
                    abund2 = 0.7048
                    mass1 = 202.9723442
                    mass2 = 204.9744275
                #########THREE#########
                elif atomList[y] == "O":
                    abund1 = 0.99757
                    abund2 = 0.00205
                    abund3 = 0.00038
                    mass1 = 15.99491461956
                    mass2 = 17.9991610
                    mass3 = 16.99913170
                    oxyNum = x
                elif atomList[y] == "Ne":
                    abund1 = 0.9048
                    abund2 = 0.0027
                    abund3 = 0.0925
                    mass1 = 19.9924401754
                    mass2 = 20.99384668
                    mass3 = 21.991385114
                elif atomList[y] == "Mg":
                    abund1 = 0.7899
                    abund2 = 0.1000
                    abund3 = 0.1101
                    mass1 = 23.985041700
                    mass2 = 24.98583692
                    mass3 = 25.982592929
                    magNum = x
                elif atomList[y] == "Si":
                    abund1 = 0.92223
                    abund2 = 0.04685
                    abund3 = 0.03092
                    mass1 = 27.9769265325
                    mass2 = 28.97649470
                    mass3 = 29.97377017
                elif atomList[y] == "Ar":
                    abund1 = 0.003336
                    abund2 = 0.000629
                    abund3 = 0.996035
                    mass1 = 35.967545106
                    mass2 = 37.9627324
                    mass3 = 39.9623831225
                elif atomList[y] == "K":
                    abund1 = 0.932581
                    abund2 = 0.067302
                    abund3 = 0.000117
                    mass1 = 38.96370668
                    mass2 = 40.96182576
                    mass3 = 39.96399848
                elif atomList[y] == "U":
                    abund1 = 0.000054
                    abund2 = 0.007204
                    abund3 = 0.992742
                    mass1 = 234.0409521
                    mass2 = 235.0439299
                    mass3 = 238.0507882
                #########FOUR#########
                elif atomList[y] == "S":
                    abund1 = 0.9493
                    abund2 = 0.0076
                    abund3 = 0.0429
                    abund4 = 0.0002
```

```
                    mass1 = 31.97207100
                    mass2 = 32.97145876
                    mass3 = 33.96786690
                    mass4 = 35.96708076
                    sulphNum = x
                elif atomList[y] == "Cr":
                    abund1 = 0.04345
                    abund2 = 0.83789
                    abund3 = 0.09501
                    abund4 = 0.02365
                    mass1 = 49.9460442
                    mass2 = 51.9405075
                    mass3 = 52.9406494
                    mass4 = 53.9388804
                elif atomList[y] == "Fe":
                    abund1 = 0.05845
                    abund2 = 0.91754
                    abund3 = 0.02119
                    abund4 = 0.00282
                    mass1 = 53.9396105
                    mass2 = 55.9349375
                    mass3 = 56.9353940
                    mass4 = 57.9332756
                    ironNum = x
                elif atomList[y] == "Sr":
                    abund1 = 0.0056
                    abund2 = 0.0986
                    abund3 = 0.0700
                    abund4 = 0.8258
                    mass1 = 83.913425
                    mass2 = 85.9092607309
                    mass3 = 86.9088774970
                    mass4 = 87.905612257
                elif atomList[y] == "Ce":
                    abund1 = 0.00185
                    abund2 = 0.00251
                    abund3 = 0.88450
                    abund4 = 0.11114
                    mass1 = 135.907172
                    mass2 = 137.905991
                    mass3 = 139.9054387
                    mass4 = 141.909244
                elif atomList[y] == "Pb":
                    abund1 = 0.014
                    abund2 = 0.241
                    abund3 = 0.221
                    abund4 = 0.524
                    mass1 = 203.9730436
                    mass2 = 205.9744653
                    mass3 = 206.9758969
                    mass4 = 207.9766521
                #########FIVE#########
                elif atomList[y] == "Ti":
                    abund1 = 0.0825
                    abund2 = 0.0744
                    abund3 = 0.7372
                    abund4 = 0.0541
                    abund5 = 0.0518
                    mass1 = 45.9526316
                    mass2 = 46.9517631
                    mass3 = 47.9479463
                    mass4 = 48.9478700
                    mass5 = 49.9447912
```

```
            elif atomList[y] == "Ni":
                abund1 = 0.680769
                abund2 = 0.262231
                abund3 = 0.011399
                abund4 = 0.036345
                abund5 = 0.009256
                mass1 = 57.9353429
                mass2 = 59.9307864
                mass3 = 60.9310560
                mass4 = 61.9283451
                mass5 = 63.9279660
                nikNum = x
            elif atomList[y] == "Zn":
                abund1 = 0.4917
                abund2 = 0.2773
                abund3 = 0.0404
                abund4 = 0.1845
                abund5 = 0.0061
                mass1 = 63.9291422
                mass2 = 65.9260334
                mass3 = 66.9271273
                mass4 = 67.9248442
                mass5 = 69.9253193
                zincNum = x
            elif atomList[y] == "Ge":
                abund1 = 0.2038
                abund2 = 0.2731
                abund3 = 0.0776
                abund4 = 0.3672
                abund5 = 0.0783
                mass1 = 69.9242474
                mass2 = 71.9220758
                mass3 = 72.9234589
                mass4 = 73.9211778
                mass5 = 75.9214026
            elif atomList[y] == "Zr":
                abund1 = 0.5145
                abund2 = 0.1122
                abund3 = 0.1715
                abund4 = 0.1738
                abund5 = 0.0280
                mass1 = 89.9047044
                mass2 = 90.9056458
                mass3 = 91.9050408
                mass4 = 93.9063152
                mass5 = 95.9082734
            elif atomList[y] == "W":
                abund1 = 0.0012
                abund2 = 0.2650
                abund3 = 0.1431
                abund4 = 0.3064
                abund5 = 0.2843
                mass1 = 179.946704
                mass2 = 181.9482042
                mass3 = 182.9502230
                mass4 = 183.9509312
                mass5 = 185.9543641
            #########SIX#########
            elif atomList[y] == "Ca":
                abund1 = 0.96941
                abund2 = 0.00647
                abund3 = 0.00135
```

```
                    abund4 = 0.02086
                    abund5 = 0.00004
                    abund6 = 0.00187
                    mass1 = 39.96259098
                    mass2 = 41.95861801
                    mass3 = 42.9587666
                    mass4 = 43.9554818
                    mass5 = 45.9536926
                    mass6 = 47.952534
                elif atomList[y] == "Se":
                    abund1 = 0.0089
                    abund2 = 0.0937
                    abund3 = 0.0763
                    abund4 = 0.2377
                    abund5 = 0.4961
                    abund6 = 0.0873
                    mass1 = 73.9224764
                    mass2 = 75.9192136
                    mass3 = 76.9199140
                    mass4 = 77.9173091
                    mass5 = 79.9165213
                    mass6 = 81.9166994
                    selNum = x
                elif atomList[y] == "Kr":
                    abund1 = 0.00355
                    abund2 = 0.02286
                    abund3 = 0.11593
                    abund4 = 0.11500
                    abund5 = 0.56987
                    abund6 = 0.17279
                    mass1 = 77.9203648
                    mass2 = 79.9163790
                    mass3 = 81.9134836
                    mass4 = 82.914136
                    mass5 = 83.91150
                    mass6 = 85.91061073
                elif atomList[y] == "Pd":
                    abund1 = 0.0102
                    abund2 = 0.1114
                    abund3 = 0.2233
                    abund4 = 0.2733
                    abund5 = 0.2646
                    abund6 = 0.1172
                    mass1 = 101.905609
                    mass2 = 103.904036
                    mass3 = 104.905085
                    mass4 = 105.903486
                    mass5 = 107.903892
                    mass6 = 109.905153
                elif atomList[y] == "Er":
                    abund1 = 0.00139
                    abund2 = 0.01601
                    abund3 = 0.33503
                    abund4 = 0.22869
                    abund5 = 0.26978
                    abund6 = 0.14910
                    mass1 = 161.928778
                    mass2 = 163.929200
                    mass3 = 165.9302931
                    mass4 = 166.9320482
                    mass5 = 167.932370
                    mass6 = 169.9354643
                elif atomList[y] == "Hf":
```

```
                    abund1 = 0.0016
                    abund2 = 0.0526
                    abund3 = 0.1860
                    abund4 = 0.2728
                    abund5 = 0.1362
                    abund6 = 0.3508
                    mass1 = 173.940046
                    mass2 = 175.9414086
                    mass3 = 176.9432207
                    mass4 = 177.9436988
                    mass5 = 178.9458161
                    mass6 = 179.9465500
                elif atomList[y] == "Pt":
                    abund1 = 0.00014
                    abund2 = 0.00782
                    abund3 = 0.32967
                    abund4 = 0.33832
                    abund5 = 0.25242
                    abund6 = 0.07163
                    mass1 = 189.959932
                    mass2 = 191.9610380
                    mass3 = 193.9626803
                    mass4 = 194.9647911
                    mass5 = 195.9649515
                    mass6 = 197.967893
                #########SEVEN#########
                elif atomList[y] == "Mo":
                    abund1 = 0.14649
                    abund2 = 0.09187
                    abund3 = 0.15873
                    abund4 = 0.16673
                    abund5 = 0.09582
                    abund6 = 0.24292
                    abund7 = 0.09744
                    mass1 = 91.906811
                    mass2 = 93.9050883
                    mass3 = 94.9058421
                    mass4 = 95.9046795
                    mass5 = 96.9060215
                    mass6 = 97.9054082
                    mass7 = 99.907477
                elif atomList[y] == "Ru":
                    abund1 = 0.0554
                    abund2 = 0.0187
                    abund3 = 0.1276
                    abund4 = 0.1260
                    abund5 = 0.1706
                    abund6 = 0.3155
                    abund7 = 0.1862
                    mass1 = 95.907598
                    mass2 = 97.905287
                    mass3 = 98.9059393
                    mass4 = 99.904219
                    mass5 = 100.9055821
                    mass6 = 101.9043493
                    mass7 = 103.905433
                elif atomList[y] == "Ba":
                    abund1 = 0.00106
                    abund2 = 0.00101
                    abund3 = 0.02417
                    abund4 = 0.06592
                    abund5 = 0.07854
                    abund6 = 0.11232
```

```
                    abund7 = 0.71698
                    mass1 = 129.9063208
                    mass2 = 131.9050613
                    mass3 = 133.9045084
                    mass4 = 134.9056886
                    mass5 = 135.9045759
                    mass6 = 136.9058274
                    mass7 = 137.9052472
                elif atomList[y] == "Nd":
                    abund1 = 0.272
                    abund2 = 0.122
                    abund3 = 0.238
                    abund4 = 0.083
                    abund5 = 0.172
                    abund6 = 0.057
                    abund7 = 0.056
                    mass1 = 141.9077233
                    mass2 = 142.9098143
                    mass3 = 143.9100873
                    mass4 = 144.9125736
                    mass5 = 145.9131169
                    mass6 = 147.916893
                    mass7 = 149.920891
                elif atomList[y] == "Sm":
                    abund1 = 0.0307
                    abund2 = 0.1499
                    abund3 = 0.1124
                    abund4 = 0.1382
                    abund5 = 0.0738
                    abund6 = 0.2675
                    abund7 = 0.2275
                    mass1 = 143.911999
                    mass2 = 146.9148979
                    mass3 = 147.9148227
                    mass4 = 148.9171847
                    mass5 = 149.9172755
                    mass6 = 151.9197324
                    mass7 = 153.9222093
                elif atomList[y] == "Gd":
                    abund1 = 0.0020
                    abund2 = 0.0218
                    abund3 = 0.1480
                    abund4 = 0.2047
                    abund5 = 0.1565
                    abund6 = 0.2484
                    abund7 = 0.2186
                    mass1 = 151.9197910
                    mass2 = 153.9208656
                    mass3 = 154.9226220
                    mass4 = 155.9221227
                    mass5 = 156.9239601
                    mass6 = 157.9241039
                    mass7 = 159.9270541
                elif atomList[y] == "Dy":
                    abund1 = 0.00056
                    abund2 = 0.00095
                    abund3 = 0.02329
                    abund4 = 0.18889
                    abund5 = 0.25475
                    abund6 = 0.24896
                    abund7 = 0.28260
                    mass1 = 155.924283
                    mass2 = 157.924409
```

```
                mass3 = 159.9251975
                mass4 = 160.9269334
                mass5 = 161.9267984
                mass6 = 162.9287312
                mass7 = 163.9291748
            elif atomList[y] == "Yb":
                abund1 = 0.0013
                abund2 = 0.0304
                abund3 = 0.1428
                abund4 = 0.2183
                abund5 = 0.1613
                abund6 = 0.3183
                abund7 = 0.1276
                mass1 = 167.933897
                mass2 = 169.9347618
                mass3 = 170.9363258
                mass4 = 171.9363815
                mass5 = 172.9382108
                mass6 = 173.9388621
                mass7 = 175.9425717
            elif atomList[y] == "Os":
                abund1 = 0.0002
                abund2 = 0.0159
                abund3 = 0.0196
                abund4 = 0.1324
                abund5 = 0.1615
                abund6 = 0.2626
                abund7 = 0.4078
                mass1 = 183.9524891
                mass2 = 185.9538382
                mass3 = 186.9557505
                mass4 = 187.9558382
                mass5 = 188.9581475
                mass6 = 189.9584470
                mass7 = 191.9614807
            elif atomList[y] == "Hg":
                abund1 = 0.0015
                abund2 = 0.0997
                abund3 = 0.1687
                abund4 = 0.2310
                abund5 = 0.1318
                abund6 = 0.2986
                abund7 = 0.0687
                mass1 = 195.965833
                mass2 = 197.9667690
                mass3 = 198.9682799
                mass4 = 199.9683260
                mass5 = 200.9703023
                mass6 = 201.9706430
                mass7 = 203.9734939
            #########EIGHT#########
            elif atomList[y] == "Cd":
                abund1 = 0.0125
                abund2 = 0.0089
                abund3 = 0.1249
                abund4 = 0.1280
                abund5 = 0.2413
                abund6 = 0.1222
                abund7 = 0.2873
                abund8 = 0.0749
                mass1 = 105.906459
                mass2 = 107.904184
                mass3 = 109.9030021
```

```
                mass4 = 110.9041781
                mass5 = 111.9027578
                mass6 = 112.9044017
                mass7 = 113.9033585
                mass8 = 115.904756
            elif atomList[y] == "Te":
                abund1 = 0.0009
                abund2 = 0.0255
                abund3 = 0.0089
                abund4 = 0.0474
                abund5 = 0.0707
                abund6 = 0.1884
                abund7 = 0.3174
                abund8 = 0.3408
                mass1 = 119.90402
                mass2 = 121.903043
                mass3 = 122.9042700
                mass4 = 123.9028179
                mass5 = 124.9044307
                mass6 = 125.9033117
                mass7 = 127.9044631
                mass8 = 129.9062244
                telNum = x
            #########NINE#########
            elif atomList[y] == "Xe":
                abund1 = 0.000952
                abund2 = 0.000890
                abund3 = 0.019102
                abund4 = 0.264006
                abund5 = 0.040710
                abund6 = 0.212324
                abund7 = 0.269086
                abund8 = 0.104357
                abund9 = 0.088573
                mass1 = 123.905893
                mass2 = 125.904274
                mass3 = 127.9035313
                mass4 = 128.9047794
                mass5 = 129.9035080
                mass6 = 130.9050824
                mass7 = 131.9041535
                mass8 = 133.9053945
                mass9 = 135.907219
            #########TEN#########
            elif atomList[y] == "Sn":
                abund1 = 0.0097
                abund2 = 0.0066
                abund3 = 0.0034
                abund4 = 0.1454
                abund5 = 0.0768
                abund6 = 0.2422
                abund7 = 0.0859
                abund8 = 0.3258
                abund9 = 0.0463
                abund10 = 0.0579
                mass1 = 111.904818
                mass2 = 113.902779
                mass3 = 114.903342
                mass4 = 115.901741
                mass5 = 116.902952
                mass6 = 117.901603
                mass7 = 118.903308
                mass8 = 119.9021947
```

```
                    mass9 = 121.9034390
                    mass10 = 123.9052739
                else:
                    errorBool = 1
                    errorEle = atomList[y]
                if len(relAbndList) == 0:
                    if (atomList[y] == "O") or (atomList[y] == "U") or (atomList[y] == "K")
or (atomList[y] == "Mg") or \
                       (atomList[y] == "Si") or (atomList[y] == "Ne") or (atomList[y] ==
"Ar"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                relAbndList.append(((abund1**(x-countEle-
countEle3))*(abund2**countEle)*(abund3**countEle3))*(nCrThree(x, countEle, countEle3)))
                                mzList.append((mass1*(x-countEle-countEle3)) +
(mass2*(countEle)) + (mass3*(countEle3)))
                                countEle3 = countEle3 + 1
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Fe") or (atomList[y] == "S") or (atomList[y] ==
"Cr") or (atomList[y] == "Sr") or (atomList[y] == "Pb") or \
                        (atomList[y] == "Ce"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    relAbndList.append(((abund1**(x-countEle-countEle3-
countEle4))*(abund2**countEle)*(abund3**countEle3)*(abund4**countEle4))* \
                                                                    (nCrFour(x, countEle, countEle3,
countEle4)))
                                    mzList.append((mass1*(x-countEle-countEle3-countEle4))
+ (mass2*(countEle)) + (mass3*(countEle3)) + (mass4*(countEle4)))
                                    countEle4 = countEle4 + 1
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Ti") or (atomList[y] == "Ni") or (atomList[y] ==
"Zn") or (atomList[y] == "Ge") or (atomList[y] == "Zr") or \
                        (atomList[y] == "W"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        relAbndList.append(((abund1**(x-countEle-countEle3-
countEle4-countEle5))*(abund2**countEle)*(abund3**countEle3)* \

(abund4**countEle4)*(abund5**countEle5))*(nCrFive(x, countEle, countEle3, countEle4,
countEle5)))
                                        mzList.append((mass1*(x-countEle-countEle3-
countEle4-countEle5)) + (mass2*(countEle)) + (mass3*(countEle3)) + \

(mass4*(countEle4))+(mass5*(countEle5)))
                                        countEle5 = countEle5 + 1
                                    countEle4 = countEle4 + 1
                                    countEle5 = 0
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle + 1
                            countEle3 = 0
                    elif (atomList[y] == "Ca") or (atomList[y] == "Se") or (atomList[y] ==
"Pd") or (atomList[y] == "Er") or (atomList[y] == "Pt") or \
                        (atomList[y] == "Kr") or (atomList[y] == "Hf"):
```

```
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                            relAbndList.append(((abund1**(x-countEle-
countEle3-countEle4-countEle5-countEle6))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6))*\
                                                                (nCrSix(x, countEle,
countEle3, countEle4, countEle5, countEle6)))
                                            mzList.append((mass1*(x-countEle-countEle3-
countEle4-countEle5-countEle6)) + (mass2*(countEle)) + \
                                                                (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6)))
                                            countEle6 = countEle6 + 1
                                        countEle5 = countEle5 + 1
                                        countEle6 = 0
                                    countEle4 = countEle4 + 1
                                    countEle5 = 0
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Mo") or (atomList[y] == "Ru") or (atomList[y] ==
"Ba") or (atomList[y] == "Nd") or (atomList[y] == "Sm") or \
                         (atomList[y] == "Gd") or (atomList[y] == "Dy") or (atomList[y] ==
"Yb") or (atomList[y] == "Os") or (atomList[y] == "Hg"):

                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                            while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                                relAbndList.append(((abund1**(x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*\
                                                                (abund7**countEle7))*\
                                                                (nCrSeven(x, countEle,
countEle3, countEle4, countEle5, countEle6, countEle7)))
                                                mzList.append((mass1*(x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7)) + (mass2*(countEle)) + \
                                                                (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7)))
                                                countEle7 = countEle7 + 1
                                            countEle6 = countEle6 + 1
                                            countEle7 = 0
                                        countEle5 = countEle5 + 1
                                        countEle6 = 0
                                    countEle4 = countEle4 + 1
                                    countEle5 = 0
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Te") or (atomList[y] == "Cd"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
```

```
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                            while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                                while countEle8 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7):
                                                    relAbndList.append(((abund1**(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8))*\
                                                                    (nCrEight(x, countEle,
countEle3, countEle4, countEle5, countEle6, countEle7, countEle8)))
                                                    mzList.append((mass1*(x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8)) + (mass2*(countEle)) + \
                                                                    (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8)))
                                                    countEle8 = countEle8 + 1
                                                countEle7 = countEle7 + 1
                                                countEle8 = 0
                                            countEle6 = countEle6 + 1
                                            countEle7 = 0
                                        countEle5 = countEle5 + 1
                                        countEle6 = 0
                                    countEle4 = countEle4 + 1
                                    countEle5 = 0
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Xe"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                            while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                                while countEle8 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7):
                                                    while countEle9 <= (x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8):
                                                        relAbndList.append(((abund1**(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-
countEle9))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9))*\
                                                                    (nCrNine(x, countEle,
countEle3, countEle4, countEle5, countEle6, countEle7, countEle8, countEle9)))
                                                    mzList.append((mass1*(x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9)) +
(mass2*(countEle)) + \
                                                                    (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9)))
                                                    countEle9 = countEle9 + 1
                                                countEle8 = countEle8 + 1
                                                countEle9 = 0
```

```
                                            countEle7 = countEle7 + 1
                                            countEle8 = 0
                                        countEle6 = countEle6 + 1
                                        countEle7 = 0
                                    countEle5 = countEle5 + 1
                                    countEle6 = 0
                                countEle4 = countEle4 + 1
                                countEle5 = 0
                            countEle3 = countEle3 + 1
                            countEle4 = 0
                        countEle = countEle +1
                        countEle3 = 0
                elif (atomList[y] == "Sn"):
                    while countEle <= x:
                        while countEle3 <= (x-countEle):
                            while countEle4 <= (x-countEle-countEle3):
                                while countEle5 <= (x-countEle-countEle3-countEle4):
                                    while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                        while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                            while countEle8 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7):
                                                while countEle9 <= (x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8):
                                                    while countEle10 <= (x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9):

relAbndList.append(((abund1**(x-countEle-countEle3-countEle4-countEle5-countEle6-countEle7-
countEle8-countEle9-countEle10))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9)*\

(abund10**countEle10))*(nCrTen(x, countEle, countEle3, countEle4, countEle5, countEle6,
countEle7, countEle8, countEle9, countEle10)))
                                                    mzList.append((mass1*(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9-countEle10))
+ (mass2*(countEle)) + \
                                                        (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9))+\
                                                        (mass10*(countEle10)))
                                                    countEle10 = countEle10 + 1
                                                countEle9 = countEle9 + 1
                                                countEle10 = 0
                                            countEle8 = countEle8 + 1
                                            countEle9 = 0
                                        countEle7 = countEle7 + 1
                                        countEle8 = 0
                                    countEle6 = countEle6 + 1
                                    countEle7 = 0
                                countEle5 = countEle5 + 1
                                countEle6 = 0
                            countEle4 = countEle4 + 1
                            countEle5 = 0
                        countEle3 = countEle3 + 1
                        countEle4 = 0
                    countEle = countEle +1
                    countEle3 = 0
                else:
                    while countEle <= x:
```

```
                            relAbndList.append(((abund1**(x-
countEle))*(abund2**countEle))*nCr(x,countEle))
                            mzList.append((mass1*(x-countEle)) + (mass2*(countEle)))
                            countEle = countEle + 1
                else:
                    if (atomList[y] == "O") or (atomList[y] == "U") or (atomList[y] == "K")
or (atomList[y] == "Mg") or \
                       (atomList[y] == "Si") or (atomList[y] == "Ne") or (atomList[y] ==
"Ar"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                currentAbndList.append(((abund1**(x-countEle-
countEle3))*(abund2**countEle)*(abund3**countEle3))*(nCrThree(x, countEle, countEle3)))
                                currentMzList.append((mass1*(x-countEle-countEle3)) +
(mass2*(countEle)) + (mass3*(countEle3)))
                                countEle3 = countEle3 + 1
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Fe") or (atomList[y] == "S") or (atomList[y] ==
"Cr") or (atomList[y] == "Sr") or (atomList[y] == "Pb") or \
                        (atomList[y] == "Ce"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    currentAbndList.append(((abund1**(x-countEle-countEle3-
countEle4))*(abund2**countEle)*(abund3**countEle3)*(abund4**countEle4))* \
                                                           (nCrFour(x, countEle, countEle3,
countEle4)))
                                    currentMzList.append((mass1*(x-countEle-countEle3-
countEle4)) + (mass2*(countEle)) + (mass3*(countEle3)) + (mass4*(countEle4)))
                                    countEle4 = countEle4 + 1
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Ti") or (atomList[y] == "Ni") or (atomList[y] ==
"Zn") or (atomList[y] == "Ge") or (atomList[y] == "Zr") or \
                        (atomList[y] == "W"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        currentAbndList.append(((abund1**(x-countEle-
countEle3-countEle4-countEle5))*(abund2**countEle)*(abund3**countEle3)* \

(abund4**countEle4)*(abund5**countEle5))*(nCrFive(x, countEle, countEle3, countEle4,
countEle5)))
                                        currentMzList.append((mass1*(x-countEle-countEle3-
countEle4-countEle5)) + (mass2*(countEle)) + (mass3*(countEle3)) + \

(mass4*(countEle4))+(mass5*(countEle5)))
                                        countEle5 = countEle5 + 1
                                    countEle4 = countEle4 + 1
                                    countEle5 = 0
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle + 1
                            countEle3 = 0
                    elif (atomList[y] == "Ca") or (atomList[y] == "Se") or (atomList[y] ==
"Pd") or (atomList[y] == "Er") or (atomList[y] == "Pt") or \
                        (atomList[y] == "Kr") or (atomList[y] == "Hf"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
```

```
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                            currentAbndList.append(((abund1**(x-countEle-
countEle3-countEle4-countEle5-countEle6))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6))*\
                                                                    (nCrSix(x, countEle,
countEle3, countEle4, countEle5, countEle6)))
                                            currentMzList.append((mass1*(x-countEle-
countEle3-countEle4-countEle5-countEle6)) + (mass2*(countEle)) + \
                                                                (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6)))
                                            countEle6 = countEle6 + 1
                                        countEle5 = countEle5 + 1
                                        countEle6 = 0
                                    countEle4 = countEle4 + 1
                                    countEle5 = 0
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Mo") or (atomList[y] == "Ru") or (atomList[y] ==
"Ba") or (atomList[y] == "Nd") or (atomList[y] == "Sm") or \
                         (atomList[y] == "Gd") or (atomList[y] == "Dy") or (atomList[y] ==
"Yb") or (atomList[y] == "Os") or (atomList[y] == "Hg"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                            while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                                currentAbndList.append(((abund1**(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*\
                                                                    (abund7**countEle7))*\
                                                                    (nCrSeven(x, countEle,
countEle3, countEle4, countEle5, countEle6, countEle7)))
                                                currentMzList.append((mass1*(x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7)) + (mass2*(countEle)) + \
                                                                    (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7)))
                                                countEle7 = countEle7 + 1
                                            countEle6 = countEle6 + 1
                                            countEle7 = 0
                                        countEle5 = countEle5 + 1
                                        countEle6 = 0
                                    countEle4 = countEle4 + 1
                                    countEle5 = 0
                                countEle3 = countEle3 + 1
                                countEle4 = 0
                            countEle = countEle +1
                            countEle3 = 0
                    elif (atomList[y] == "Te") or (atomList[y] == "Cd"):
                        while countEle <= x:
                            while countEle3 <= (x-countEle):
                                while countEle4 <= (x-countEle-countEle3):
                                    while countEle5 <= (x-countEle-countEle3-countEle4):
                                        while countEle6 <= (x-countEle-countEle3-countEle4-
```

```
countEle5):
                                            while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                                while countEle8 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7):
                                                    currentAbndList.append(((abund1**(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8))*\
                                                                    (nCrEight(x, countEle,
countEle3, countEle4, countEle5, countEle6, countEle7, countEle8)))
                                                    currentMzList.append((mass1*(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8)) + (mass2*(countEle))
+ \
                                                                    (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8)))
                                                        countEle8 = countEle8 + 1
                                                    countEle7 = countEle7 + 1
                                                    countEle8 = 0
                                                countEle6 = countEle6 + 1
                                                countEle7 = 0
                                            countEle5 = countEle5 + 1
                                            countEle6 = 0
                                        countEle4 = countEle4 + 1
                                        countEle5 = 0
                                    countEle3 = countEle3 + 1
                                    countEle4 = 0
                                countEle = countEle +1
                                countEle3 = 0
                        elif (atomList[y] == "Xe"):
                            while countEle <= x:
                                while countEle3 <= (x-countEle):
                                    while countEle4 <= (x-countEle-countEle3):
                                        while countEle5 <= (x-countEle-countEle3-countEle4):
                                            while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                                while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                                    while countEle8 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7):
                                                        while countEle9 <= (x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8):

currentAbndList.append(((abund1**(x-countEle-countEle3-countEle4-countEle5-countEle6-
countEle7-countEle8-countEle9))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9))*\
                                                                    (nCrNine(x, countEle,
countEle3, countEle4, countEle5, countEle6, countEle7, countEle8, countEle9)))
                                                                currentMzList.append((mass1*(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9)) +
(mass2*(countEle)) + \
                                                                    (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9)))
                                                                countEle9 = countEle9 + 1
                                                            countEle8 = countEle8 + 1
                                                            countEle9 = 0
                                                        countEle7 = countEle7 + 1
                                                        countEle8 = 0
```

```
                                        countEle6 = countEle6 + 1
                                        countEle7 = 0
                                    countEle5 = countEle5 + 1
                                    countEle6 = 0
                                countEle4 = countEle4 + 1
                                countEle5 = 0
                            countEle3 = countEle3 + 1
                            countEle4 = 0
                        countEle = countEle +1
                        countEle3 = 0
                elif (atomList[y] == "Sn"):
                    while countEle <= x:
                        while countEle3 <= (x-countEle):
                            while countEle4 <= (x-countEle-countEle3):
                                while countEle5 <= (x-countEle-countEle3-countEle4):
                                    while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                        while countEle7 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6):
                                            while countEle8 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7):
                                                while countEle9 <= (x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8):
                                                    while countEle10 <= (x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9):

currentAbndList.append(((abund1**(x-countEle-countEle3-countEle4-countEle5-countEle6-
countEle7-countEle8-countEle9-countEle10))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9)*\

(abund10**countEle10))*(nCrTen(x, countEle, countEle3, countEle4, countEle5, countEle6,
countEle7, countEle8, countEle9, countEle10)))
                                                    currentMzList.append((mass1*(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9-countEle10))
+ (mass2*(countEle)) + \
                                                        (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9))+\
                                                        (mass10*(countEle10)))
                                                    countEle10 = countEle10 + 1
                                                countEle9 = countEle9 + 1
                                                countEle10 = 0
                                            countEle8 = countEle8 + 1
                                            countEle9 = 0
                                        countEle7 = countEle7 + 1
                                        countEle8 = 0
                                    countEle6 = countEle6 + 1
                                    countEle7 = 0
                                countEle5 = countEle5 + 1
                                countEle6 = 0
                            countEle4 = countEle4 + 1
                            countEle5 = 0
                        countEle3 = countEle3 + 1
                        countEle4 = 0
                    countEle = countEle +1
                    countEle3 = 0
                else:
                    while countEle <= x:
                        currentAbndList.append(((abund1**(x-
countEle))*(abund2**countEle))*nCr(x,countEle))
                        currentMzList.append((mass1*(x-countEle)) + (mass2*(countEle)))
```

```
                            countEle = countEle + 1
                    while t < len(currentAbndList):
                        while t1 < len(relAbndList):
                            if relAbndList[t1] < 0.000001:
                                if currentAbndList[t] < 0.000001:
                                    relAbndList[t1] = 0
                                    mzList[t1] = 0
                                    currentMzList[t] = 0
                                    currentAbndList[t] = 0
                            else:
                                tempAbndList.append(float(relAbndList[t1]) *
float(currentAbndList[t]))

tempMzList.append((float(currentMzList[t])+float(mzList[t1])))
                            t1 = t1 + 1
                        t = t + 1
                        t1 = 0
                    relAbndList[:] = tempAbndList[:]
                    mzList[:] = tempMzList[:]
                    del tempAbndList[:]
                    del tempMzList[:]
                y = y + 1
                t = 0
                t1 = 0
                countEle = 0
                countEle3 = 0
                countEle4 = 0
                countEle5 = 0
                countEle6 = 0
                countEle7 = 0
                countEle8 = 0
                countEle9 = 0
                countEle10 = 0
                abund1 = 0
                abund2 = 0
                abund3 = 0
                abund4 = 0
                abund5 = 0
                abund6 = 0
                abund7 = 0
                abund8 = 0
                abund9 = 0
                abund10 = 0
                mass1 = 0
                mass2 = 0
                mass3 = 0
                mass4 = 0
                mass5 = 0
                mass6 = 0
                mass7 = 0
                mass8 = 0
                mass9 = 0
                mass10 = 0
                del currentAbndList[:]
                del currentMzList[:]
            ## Converts all isotopomers which have an extremly small abundance to zero
            for item in relAbndList:
                if relAbndList[t1] < 0.00001:
                    relAbndList[t1] = 0
                    mzList[t1] = 0
                t1 = t1 + 1
            x = 0
            y = 0
```

```
            ## Removes all masses and abundances which are zero
            mzList = filter(lambda a: a != 0, mzList)
            relAbndList = filter(lambda a: a != 0, relAbndList)
            x = 0
            y = 0
            ## If statement to include molecular formulas with elements with no isotopes
with [M+] giving an isotpoe abundance of 100 %
            if len(relAbndList) == 0:
                relAbndList.append(1)
            ## If statement to include molecular formulas with elements with no isotopes
with [M+]
            if len(mzList) == 0:
                    mzList.append(0.0000000001) #adding a very small value so the entry
will be picked up by the next if statement
            ## Loop to add the masses of the elements which do not have natural isotopes
            for item in mzList:
                if mzList[x] > 0:
                    mzList[x] = mzList[x] + (numOfNa * 22.9897692809) + (numOfP *
30.97376163) + (numOfF * 18.99840322) + \
                                (numOfI * 126.904473) + (numOfAs * 74.9215965) + (numOfSc *
44.9559119) + (numOfAl * 26.98153863) + (numOfMn * 54.9380451) + \
                                (numOfCo * 58.9331950) + (numOfBe * 9.0121822) + (numOfAu *
196.9665687) + \
                                (numOfHe * 4.00260325415) + (numOfY * 88.9058483) + (numOfD
* 2.01410178) + \
                                (numOfNb * 92.9063781) + (numOfRh * 102.905504) + (numOfCs
* 132.905451933) + (numOfBi * 208.9803987)  + (numOfTh * 232.0380553) + \
                                (numOfPa * 231.0358840) + (numOfAc * 227.0277521) +
(numOfTm * 168.9342133) + (numOfHo * 164.9303221) + (numOfTb * 158.9253468) + \
                                (numOfPr * 140.9076528)
                x = x + 1
            x = 0
            ## Substracts the mass of an electron to give the masses a positive charge
            for item in mzList:
                mzList[x] = mzList[x] - (0.000548999)
                x = x + 1
            x = 0
            y = 1
            ## Loops through all masses in the mass list, looking for isotopomers to
combine
            for item in mzList:
                currentMZ = mzList[x]
                while y < len(mzList):
                    ## RESOLUTION(currentMZ/msResolution) ##
                    if msMode == "Gaussian":
                        if mzList[y] > (currentMZ - (currentMZ/msResolution)):
                            if mzList[y] < (currentMZ + (currentMZ/msResolution)):
                                if currentMZ != 0:
                                    currentMZ =
mzList[x]*(relAbndList[x]/(relAbndList[x]+relAbndList[y]))+mzList[y]* \

(relAbndList[y]/(relAbndList[x]+relAbndList[y]))
                                    mzList[y] = 0
                                    mzList[x] = currentMZ
                                    relAbndList[x] = relAbndList[x] + relAbndList[y]
                                    relAbndList[y] = 0
                    elif msMode == "Centroid":
                        if mzList[y] > (currentMZ - 0.05):
                            if mzList[y] < (currentMZ + 0.05):
                                if currentMZ != 0:
                                    currentMZ =
mzList[x]*(relAbndList[x]/(relAbndList[x]+relAbndList[y]))+mzList[y]* \
```

```
(relAbndList[y]/(relAbndList[x]+relAbndList[y]))
                                    mzList[y] = 0
                                    mzList[x] = currentMZ
                                    relAbndList[x] = relAbndList[x] + relAbndList[y]
                                    relAbndList[y] = 0
                    y = y + 1
                x = x + 1
                y = x + 1
            x = 0
            ## Removes all masses and abundances which are zero
            mzList = filter(lambda a: a != 0, mzList)
            relAbndList = filter(lambda a: a != 0, relAbndList)
            ## Sorts the two lists by mass
            mzList, relAbndList = zip(*sorted(zip(mzList, relAbndList)))
            x = 0
            y = 0
            elementList = []
            numElement = []
            atomNumList = []
            atomList = []
            resultList.append(compound)
            resultList.append(mzList[0])
            isotope1A = 0
            isotope2A = 0
            isotope1M = 0
            isotope2M = 0
            ## Loops through all the masses in the centroided isotope pattern and
            ## finds which one is at the second isotopomeric interval
            while y < len(mzList):
                if mzList[y] > mzList[0] + 0.8:
                    if  mzList[y] < mzList[0] + 1.2:
                        if relAbndList[y] > isotope1A:
                            isotope1A = relAbndList[y]
                            isotope1M = mzList[y]
                y = y + 1
            y = 0
            ## Loops through all the masses in the centroided isotope pattern and
            ## finds which one is at the thrid isotopomeric interval
            while y < len(mzList):
                if mzList[y] > mzList[0] + 1.8:
                    if  mzList[y] < mzList[0] + 2.2:
                        if relAbndList[y] > isotope2A:
                            isotope2A = relAbndList[y]
                            isotope2M = mzList[y]
                y = y + 1
            y = 0
            resultList.append(isotope1M)
            resultList.append(isotope2M)
            resultList.append(relAbndList[0])
            resultList.append(isotope1A)
            resultList.append(isotope2A)
            resultList.append(broNum)
            resultList.append(carbNum)
            resultList.append(oxyNum)
            resultList.append(nitNum)
            resultList.append(phosNum)
            resultList.append(hydroNum)
            resultList.append(ioNum)
            resultList.append(chloNum)
            resultList.append(sulphNum)
            resultList.append(borNum)
            resultList.append(ironNum+copNum+nikNum+magNum+zincNum+selNum+telNum)
            mzList = []
```

```
            relAbndList = []
            primary.txt.configure(state="normal")
            primary.txt.insert('1.0', str(resultList[0]) + "_" + str(resultList[1]) + \
                               "_" + str(resultList[2]) + "_" + str(resultList[3]) +  \
                               "_" + str(resultList[4])+ \
                               "_" + str(resultList[5]) + "_" + str(resultList[6]) + \
                               "_" + str(resultList[7]) + "_" + str(resultList[8]) + \
                               "_" + str(resultList[9]) + \
                               "_" + str(resultList[10]) + "_" + str(resultList[11]) + \
                               "_" + str(resultList[12]) + "_" + str(resultList[13]) + \
                               "_" + str(resultList[14]) + \
                               "_" + str(resultList[15]) + "_" + str(resultList[16]) + \
                               "_" + str(resultList[17]) + '\n')
            primary.txt.configure(state="disabled")
            x = 0
            nitNum = 0
            oxyNum = 0
            carbNum = 0
            broNum = 0
            chloNum = 0
            sulphNum = 0
            borNum = 0
            ironNum = 0
            phosNum = 0
            fluNum = 0
            ioNum = 0
            copNum = 0
            nikNum = 0
            magNum = 0
            zincNum = 0
            selNum = 0
            telNum = 0
            hydroNum = 0
            isotope1A = 0
            isotope2A = 0
            isotope1M = 1000
            isotope2M = 1000
            resultList = []
            n = n + 1
        return
if __name__ == "__main__":
    app = simpleapp_tk(None)
    app.title('FormExtract 1.01')
    img = PhotoImage(data=iconData)
    app.tk.call('wm', 'iconphoto', app._w, img)
    app.mainloop()
```

Code S-2: *Source code for FormExtract v1.01*

This programme processes a CSV file of molecular formulas and generates isotope pattern and molecular formula information for the [M+H]+ ions. The resulting information includes extract mass and relative abundances of the first three istope clusters.

```
from Tkinter import *
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from pylab import figure, show
import matplotlib.pyplot as plt
import Tkinter
import numpy
import math
import ttk
import tkFileDialog
import csv
import copy
import string
iconData = '''R0lGODlhEAAQAPcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArMwArZgArmQArzAAr/wBVAABVMwBV
        ZgBVmQBVzABV/wCAAACAMwCAZgCAmQCAzACA/wCqAACqMwCqZgCqmQCqzACq/wDVAADVMwDVZgDV
        mQDVzADV/wD/AAD/MwD/ZgD/mQD/zAD//zMAADMAMzMAZjMAmTMAzDMA/zMrADMrMzMrZjMrmTMr
        zDMr/zNVADNVMzNVZjNVmTNVzDNV/zOAADOAMzOAZjOAmTOAzDOA/zOqADOqMzOqZjOqmTOqzDOq
        /zPVADPVMzPVZjPVmTPVzDPV/zP/ADP/MzP/ZjP/mTP/zDP//2YAAGYAM2YAZmYAmWYAzGYA/2Yr
        AGYrM2YrZmYrmWYrzGYr/2ZVAGZVM2ZVZmZVmWZVzGZV/2aAAGaAM2aAZmaAmWaAzGaA/2aqAGaq
        M2aqZmaqmWaqzGaq/2bVAGbVM2bVZmbVmWbVzGbV/2b/AGb/M2b/Zmb/mWb/zGb//5kAAJkAM5kA
        ZpkAmZkAzJkA/5krAJkrM5krZpkrmZkrzJkr/5lVAJlVM5lVZplVmZlVzJlV/5mAAJmAM5mAZpmA
        mZmAzJmA/5mqAJmqM5mqZpmqmZmqzJmq/5nVAJnVM5nVZpnVmZnVzJnV/5n/AJn/M5n/Zpn/mZn/
        zJn//8wAAMwAM8wAZswAmcwAzMwA/8wrAMwrM8wrZswrmcwrzMwr/8xVAMxVM8xVZsxVmcxVzMxV
        /8yAAMyAM8yAZsyAmcyAzMyA/8yqAMyqM8yqZsyqmcyqzMyq/8zVAMzVM8zVZszVmczVzMzV/8z/
        AMz/M8z/Zsz/mcz/zMz///8AAP8AM/8AZv8Amf8AzP8A//8rAP8rM/8rZv8rmf8rzP8r//9VAP9V
        M/9VZv9Vmf9VzP9V//+AAP+AM/+AZv+Amf+AzP+A//+qAP+qM/+qZv+qmf+qzP+q///VAP/VM//V
        Zv/Vmf/VzP/V////AP//M///Zv//mf//zP///wAAAAAAAAAAAAAAACH5BAEAAPwALAAAAAAQABAA
        AAiKADt04CCQ4MCCCAlaOLiQwwWCDiEyHMihYoeHBzs0pFiRA4CCDy0OvFAQgEmKAjUe9Ghyn0mL
        EgWaBLCv5kuFHlm2rOkSAByCM2n25NnzQsudQ3kGrRlDaFKbAAwEnUq1alUiGjRkyDBEA5GvQ4Zk
        wOr169YPRJSYFUsEBIivH5KETfI1idywXwMCADs=
        '''
modeChoices1 = [
    'Centroid'#,
    #'Gaussian'
]
class simpleapp_tk(Tkinter.Tk):
    def __init__(primary,parent):
        Tkinter.Tk.__init__(primary,parent)
        primary.parent = parent
        primary.initialize()
        primary.update()
        primary.minsize(primary.winfo_width(), primary.winfo_height())
    def initialize(primary):
        primary.grid()
        primary.grid_rowconfigure(10,weight=1)
        primary.grid_columnconfigure(3,weight=1)
        primary.massList = []
        primary.abundanceList = []
        primary.rawMasterList = []
        primary.resultsMasterList = []
        primary.rSel = StringVar()
        primary.resVar = StringVar()
        primary.selectedMode = StringVar()
        color = '0.9'
        plotFrame = Frame(primary, width=300, height=300)
        plotFrame.pack(fill="both", expand=True)
        plotFrame.grid_propagate(False)
        plotFrame.grid_rowconfigure(10, weight=1)
        plotFrame.grid_columnconfigure(3, weight=1)
        plotFrame.grid(row=6, column=1, columnspan=2, rowspan=4, sticky="neswn")
        fig = Figure()
        rect = fig.patch
```

```
        rect.set_facecolor(color)
        primary.CSVBtn = Button(primary, text='Import CSV', command=primary.getFileName)
        primary.CSVBtn.grid(row=1, column=1, columnspan=2, pady=(5,5))
        primary.resVarLabel = Label(primary, text="Resolution:")
        primary.resVarLabel.grid(row=3, column=1, sticky=E, pady=(0,5))
        primary.resVarVal = Entry(primary, textvariable=primary.resVar, width=15)
        primary.resVarVal.grid(row=3, column=2, sticky=W)
        primary.resVarVal.insert(0, 50000)
        primary.resVarVal.configure(state="disabled")
        primary.modeLabel = Label(primary, text="Mode:")
        primary.modeLabel.grid(row=2, column=1, sticky=E)
        primary.modeOptions_value = StringVar()
        primary.modeOptions_value.trace('w', primary.ModeRefresh)
        primary.modeOptions = ttk.Combobox(primary, state='readonly', width=15,
textvariable=primary.modeOptions_value)
        primary.modeOptions["values"] = ('Centroid')#, 'Gaussian')
        primary.modeOptions.set("Centroid")
        primary.modeOptions.grid(row=2, column=2, columnspan=1, pady=(0,0), sticky=SW)
        primary.findBtn = Button(primary, text='Extract Formula List',
command=primary.findIsoAndComps)
        primary.findBtn.grid(row=5, column=1, columnspan=2, pady=(0,0))
        primary.txt = Text(primary, borderwidth=3, relief="sunken")
        primary.txt.config(font=("consolas", 9), undo=True, wrap='word', width=20,
height=20)
        primary.txt.grid(row=6, column=1, columnspan=3, rowspan=5, sticky="neswn", padx=2,
pady=5)
        primary.txt.configure(state="disabled")
        scrollb = Scrollbar(primary, command=primary.txt.yview)
        scrollb.grid(row=6, column=4, rowspan=5, padx=(0,10), pady=0, sticky="neswn")
        primary.txt['yscrollcommand'] = scrollb.set
    def ModeRefresh(primary, *args):
        msMode = primary.modeOptions_value.get()
        if msMode == "Centroid":
            result = primary.resVarVal.configure(state="disabled")
        if msMode == "Gaussian":
            result = primary.resVarVal.configure(state="normal")
        return
    def getFileName(primary):
        fileName = tkFileDialog.askopenfilename()
        justTheFile = StringVar
        if fileName != "":
            justTheFile = fileName.split('/')
            justTheFile = justTheFile[len(justTheFile)-1]
            app.title('PatExtract 1.01 - ' + justTheFile)
            csvRaw = csv.reader(open(fileName, 'U'), dialect='excel')
            formList = []
            completeList = []
            listLen = 0
            maxRT = 0
            for row in csvRaw:
                formList.append(row[1])
            listLen = len(formList)
            formList = formList[1:listLen]
            n = 0
            primary.txt.configure(state="normal")
            primary.txt.delete(1.0, END)
            for each in formList:
                primary.txt.insert('1.0', str(each) + '\n')
            primary.txt.configure(state="disabled")
            primary.rawMasterList = formList
        return
    def findIsoAndComps(primary):
        rawList = []
```

```
        rawList = copy.deepcopy(primary.rawMasterList)
        msResolution = float(primary.resVar.get())
        msMode = primary.modeOptions_value.get()
        editedList = []
        elementList = []
        numElement = []
        atomNumList = []
        atomList = []
        currentAbndList = []
        currentMzList = []
        tempAbndList = []
        tempMzList = []
        mzList = []
        relAbndList = []
        resultList = []
        pattern = '([A-Z])([a-z]*)(\d*)'
        pattern2 = '(\d*)'
        n = 0
        x = 0
        y = 0
        uniCount = 1
        formCount = 1
        primary.txt.configure(state="normal")
        primary.txt.delete(1.0, END)
        primary.txt.configure(state="disabled")
        n1 = 1
        def nCr(n,r):
            f = math.factorial
            return f(n) / f(r) / f(n-r)
        def nCrThree(n,b,c):
            f = math.factorial
            return f(n) / (f(n-(b+c)) * f(b) * f(c))
        def nCrFour(n,b,c,d):
            f = math.factorial
            return f(n) / (f(n-(b+c+d)) * f(b) * f(c) * f(d))
        def nCrFive(n,b,c,d,e):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e)) * f(b) * f(c) * f(d) * f(e))
        def nCrSix(n,b,c,d,e,g):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g)) * f(b) * f(c) * f(d) * f(e) * f(g))
        def nCrSeven(n,b,c,d,e,g,h):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h))
        def nCrEight(n,b,c,d,e,g,h,i):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h) *
f(i))
        def nCrNine(n,b,c,d,e,g,h,i,j):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i+j)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h)
* f(i) * f(j))
        def nCrTen(n,b,c,d,e,g,h,i,j,k):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i+j+k)) * f(b) * f(c) * f(d) * f(e) * f(g) *
f(h) * f(i) * f(j) * f(k))
        while n < len(rawList):
            compound = rawList[n]
            compound = compound + "H"
            numOfHe = 0
            numOfBe = 0
            numOfF = 0
            numOfNa = 0
```

```
            numOfAl = 0
            numOfP = 0
            numOfSc = 0
            numOfMn = 0
            numOfCo = 0
            numOfAs = 0
            numOfY = 0
            numOfNb = 0
            numOfRh = 0
            numOfI = 0
            numOfCs = 0
            numOfPr = 0
            numOfTb = 0
            numOfHo = 0
            numOfTm = 0
            numOfAu = 0
            numOfBi = 0
            numOfAc = 0
            numOfTh = 0
            numOfPa = 0
            oxyNum = 0
            nitNum = 0
            carbNum = 0
            broNum = 0
            chloNum = 0
            sulphNum = 0
            borNum = 0
            ironNum = 0
            phosNum = 0
            ioNum = 0
            fluNum = 0
            nikNum = 0
            magNum = 0
            copNum = 0
            zincNum = 0
            selNum = 0
            telNum = 0
            t = 0
            t1 = 0
            countEle = 0
            countEle3 = 0
            countEle4 = 0
            countEle5 = 0
            countEle6 = 0
            countEle7 = 0
            countEle8 = 0
            countEle9 = 0
            countEle10 = 0
            abund1 = 1
            abund2 = 0
            abund3 = 0
            abund4 = 0
            abund5 = 0
            abund6 = 0
            abund7 = 0
            abund8 = 0
            abund9 = 0
            abund10 = 0
            mass1 = 0
            mass2 = 0
            mass3 = 0
            mass4 = 0
            mass5 = 0
```

```
            mass6 = 0
            mass7 = 0
            mass8 = 0
            mass9 = 0
            mass10 = 0
            for match in re.finditer(pattern, compound):
                s = match.start()
                e = match.end()
                elementComp = compound[s:e]
                elementList.append(compound[s:e])
            for each in elementList:
                element = ''.join([i for i in elementList[x] if not i.isdigit()])
                numElement = ''.join([i for i in elementList[x] if i.isdigit()])
                if numElement == "":
                    numElement = "1"
                elementList[x] = element + "." + numElement
                x = x + 1
            for each in elementList:
                sepEleList = elementList[y].split(".")
                atomNumList.append(int(sepEleList[1]))
                atomList.append(sepEleList[0])
                y = y + 1
            x = 0
            y = 1
            for each in atomList:
                while y < len(atomList):
                    if atomNumList[x] == 0:
                        atomList[x] = ""
                    if atomList[x] == atomList[y]:
                        atomList[y] = ""
                        atomNumList[x] = atomNumList[x] + atomNumList[y]
                        atomNumList[y] = 0
                    y = y + 1
                x = x + 1
                y = x + 1
            x = 0
            for each in atomList:
                if atomList[x] == "He": #4.00260325415
                    numOfHe = atomNumList[x]
                    atomNumList[x] = 0
                    atomList[x] = ""
                elif atomList[x] == "Be": #9.0121822
                    numOfBe = atomNumList[x]
                    atomNumList[x] = 0
                    atomList[x] = ""
                elif atomList[x] == "F": #18.99840
                    numOfF = atomNumList[x]
                    atomNumList[x] = 0
                    atomList[x] = ""
                    fluNum = numOfF
                elif atomList[x] == "Na": #22.98977
                    numOfNa = atomNumList[x]
                    atomNumList[x] = 0
                    atomList[x] = ""
                elif atomList[x] == "Al": #26.98153863
                    numOfAl = atomNumList[x]
                    atomNumList[x] = 0
                    atomList[x] = ""
                elif atomList[x] == "P": #30.97376
                    numOfP = atomNumList[x]
                    atomNumList[x] = 0
                    atomList[x] = ""
                    phosNum = numOfP
```

```
            elif atomList[x] == "Sc": #44.9559119
                numOfSc = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Mn": #54.9380451
                numOfMn = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Co": #58.9331950
                numOfCo = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "As": #74.92160
                numOfAs = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Y": #88.9058483
                numOfY = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Nb": #92.9063781
                numOfNb = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Rh": #102.905504
                numOfRh = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "I": #126.90447
                numOfI = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
                ioNum = numOfI
            elif atomList[x] == "Cs": #132.905451933
                numOfCs = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Pr": #140.9076528
                numOfPr = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Tb": #158.9253468
                numOfTb = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Ho": #164.9303221
                numOfHo = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Tm": #168.9342133
                numOfTm = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Au": #196.9665687
                numOfAu = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Bi": #208.9803987
                numOfBi = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Ac": #227.0277521
                numOfAc = atomNumList[x]
```

```
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Th": #232.0380553
                numOfTh = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            elif atomList[x] == "Pa": #231.0358840
                numOfPa = atomNumList[x]
                atomNumList[x] = 0
                atomList[x] = ""
            x = x + 1
        while 0 in atomNumList: atomNumList.remove(0)
        while "" in atomList: atomList.remove("")
        x = 0
        y = 0
        numOfIsos = 0
        for each in atomList:
            x = atomNumList[y]
            if atomList[y] == "C":
                abund1 = 0.9893
                abund2 = 0.0107
                mass1 = 12
                mass2 = 13.0033548378
                carbNum = x
                numOfIsos = 2
            elif atomList[y] == "H":
                abund1 = 0.999885
                abund2 = 0.000115
                mass1 = 1.00782503207
                mass2 = 2.0141017778
                numOfIsos = 2
            elif atomList[y] == "Li":
                abund1 = 0.0759
                abund2 = 0.9241
                mass1 = 6.015122795
                mass2 = 7.01600455
                numOfIsos = 2
            elif atomList[y] == "B":
                abund1 = 0.199
                abund2 = 0.801
                mass1 = 10.0129370
                mass2 = 11.0093054
                borNum = x
                numOfIsos = 2
            elif atomList[y] == "N":
                abund1 = 0.99636
                abund2 = 0.00364
                mass1 = 14.0030740048
                mass2 = 15.0001088982
                nitNum = x
                numOfIsos = 2
            elif atomList[y] == "Cl":
                abund1 = 0.7576
                abund2 = 0.2424
                mass1 = 34.96885268
                mass2 = 36.96590259
                chloNum = x
                numOfIsos = 2
            elif atomList[y] == "V":
                abund1 = 0.00250
                abund2 = 0.99750
                mass1 = 49.9471585
                mass2 = 50.9439595
```

```
                numOfIsos = 2
            elif atomList[y] == "Cu":
                abund1 = 0.6915
                abund2 = 0.3085
                mass1 = 62.9295975
                mass2 = 64.9277895
                copNum = x
                numOfIsos = 2
            elif atomList[y] == "Ga":
                abund1 = 0.60108
                abund2 = 0.39892
                mass1 = 68.9255736
                mass2 = 70.9247013
                numOfIsos = 2
            elif atomList[y] == "Br":
                abund1 = 0.5069
                abund2 = 0.4931
                mass1 = 78.9183371
                mass2 = 80.9162906
                broNum = x
                numOfIsos = 2
            elif atomList[y] == "Rb":
                abund1 = 0.7217
                abund2 = 0.2783
                mass1 = 84.911789738
                mass2 = 86.909180527
                numOfIsos = 2
            elif atomList[y] == "Ag":
                abund1 = 0.51839
                abund2 = 0.48161
                mass1 = 106.905097
                mass2 = 108.904752
                numOfIsos = 2
            elif atomList[y] == "In":
                abund1 = 0.0429
                abund2 = 0.9571
                mass1 = 112.904058
                mass2 = 114.903878
                numOfIsos = 2
            elif atomList[y] == "Sb":
                abund1 = 0.5721
                abund2 = 0.4279
                mass1 = 120.903815
                mass2 = 122.9042140
                numOfIsos = 2
            elif atomList[y] == "La":
                abund1 = 0.0009
                abund2 = 0.99910
                mass1 = 137.907112
                mass2 = 138.9063533
                numOfIsos = 2
            elif atomList[y] == "Eu":
                abund1 = 0.4781
                abund2 = 0.5219
                mass1 = 150.9198502
                mass2 = 152.9212303
                numOfIsos = 2
            elif atomList[y] == "Lu":
                abund1 = 0.9741
                abund2 = 0.0259
                mass1 = 174.9407718
                mass2 = 175.9426863
                numOfIsos = 2
```

```
            elif atomList[y] == "Ta":
                abund1 = 0.00012
                abund2 = 0.99988
                mass1 = 179.9474648
                mass2 = 180.9479958
                numOfIsos = 2
            elif atomList[y] == "Re":
                abund1 = 0.3740
                abund2 = 0.6260
                mass1 = 184.9529550
                mass2 = 186.9557531
                numOfIsos = 2
            elif atomList[y] == "Ir":
                abund1 = 0.373
                abund2 = 0.627
                mass1 = 190.9605940
                mass2 = 192.9629264
                numOfIsos = 2
            elif atomList[y] == "Tl":
                abund1 = 0.2952
                abund2 = 0.7048
                mass1 = 202.9723442
                mass2 = 204.9744275
                numOfIsos = 2
            #########THREE#########
            elif atomList[y] == "O":
                abund1 = 0.99757
                abund2 = 0.00205
                abund3 = 0.00038
                mass1 = 15.99491461956
                mass2 = 17.9991610
                mass3 = 16.99913170
                oxyNum = x
                numOfIsos = 3
            elif atomList[y] == "Ne":
                abund1 = 0.9048
                abund2 = 0.0027
                abund3 = 0.0925
                mass1 = 19.9924401754
                mass2 = 20.99384668
                mass3 = 21.991385114
                numOfIsos = 3
            elif atomList[y] == "Mg":
                abund1 = 0.7899
                abund2 = 0.1000
                abund3 = 0.1101
                mass1 = 23.985041700
                mass2 = 24.98583692
                mass3 = 25.982592929
                magNum = x
                numOfIsos = 3
            elif atomList[y] == "Si":
                abund1 = 0.92223
                abund2 = 0.04685
                abund3 = 0.03092
                mass1 = 27.9769265325
                mass2 = 28.97649470
                mass3 = 29.97377017
                numOfIsos = 3
            elif atomList[y] == "Ar":
                abund1 = 0.003336
                abund2 = 0.000629
                abund3 = 0.996035
```

```
                mass1 = 35.967545106
                mass2 = 37.9627324
                mass3 = 39.9623831225
                numOfIsos = 3
            elif atomList[y] == "K":
                abund1 = 0.932581
                abund2 = 0.067302
                abund3 = 0.000117
                mass1 = 38.96370668
                mass2 = 40.96182576
                mass3 = 39.96399848
                numOfIsos = 3
            elif atomList[y] == "U":
                abund1 = 0.000054
                abund2 = 0.007204
                abund3 = 0.992742
                mass1 = 234.0409521
                mass2 = 235.0439299
                mass3 = 238.0507882
                numOfIsos = 3
            #########FOUR#########
            elif atomList[y] == "S":
                abund1 = 0.9493
                abund2 = 0.0076
                abund3 = 0.0429
                abund4 = 0.0002
                mass1 = 31.97207100
                mass2 = 32.97145876
                mass3 = 33.96786690
                mass4 = 35.96708076
                sulphNum = x
                numOfIsos = 4
            elif atomList[y] == "Cr":
                abund1 = 0.04345
                abund2 = 0.83789
                abund3 = 0.09501
                abund4 = 0.02365
                mass1 = 49.9460442
                mass2 = 51.9405075
                mass3 = 52.9406494
                mass4 = 53.9388804
                numOfIsos = 4
            elif atomList[y] == "Fe":
                abund1 = 0.05845
                abund2 = 0.91754
                abund3 = 0.02119
                abund4 = 0.00282
                mass1 = 53.9396105
                mass2 = 55.9349375
                mass3 = 56.9353940
                mass4 = 57.9332756
                ironNum = x
                numOfIsos = 4
            elif atomList[y] == "Sr":
                abund1 = 0.0056
                abund2 = 0.0986
                abund3 = 0.0700
                abund4 = 0.8258
                mass1 = 83.913425
                mass2 = 85.9092607309
                mass3 = 86.9088774970
                mass4 = 87.905612257
                numOfIsos = 4
```

```
            elif atomList[y] == "Ce":
                abund1 = 0.00185
                abund2 = 0.00251
                abund3 = 0.88450
                abund4 = 0.11114
                mass1 = 135.907172
                mass2 = 137.905991
                mass3 = 139.9054387
                mass4 = 141.909244
                numOfIsos = 4
            elif atomList[y] == "Pb":
                abund1 = 0.014
                abund2 = 0.241
                abund3 = 0.221
                abund4 = 0.524
                mass1 = 203.9730436
                mass2 = 205.9744653
                mass3 = 206.9758969
                mass4 = 207.9766521
                numOfIsos = 4
            #########FIVE#########
            elif atomList[y] == "Ti":
                abund1 = 0.0825
                abund2 = 0.0744
                abund3 = 0.7372
                abund4 = 0.0541
                abund5 = 0.0518
                mass1 = 45.9526316
                mass2 = 46.9517631
                mass3 = 47.9479463
                mass4 = 48.9478700
                mass5 = 49.9447912
                numOfIsos = 5
            elif atomList[y] == "Ni":
                abund1 = 0.680769
                abund2 = 0.262231
                abund3 = 0.011399
                abund4 = 0.036345
                abund5 = 0.009256
                mass1 = 57.9353429
                mass2 = 59.9307864
                mass3 = 60.9310560
                mass4 = 61.9283451
                mass5 = 63.9279660
                nikNum = x
                numOfIsos = 5
            elif atomList[y] == "Zn":
                abund1 = 0.4917
                abund2 = 0.2773
                abund3 = 0.0404
                abund4 = 0.1845
                abund5 = 0.0061
                mass1 = 63.9291422
                mass2 = 65.9260334
                mass3 = 66.9271273
                mass4 = 67.9248442
                mass5 = 69.9253193
                zincNum = x
                numOfIsos = 5
            elif atomList[y] == "Ge":
                abund1 = 0.2038
                abund2 = 0.2731
                abund3 = 0.0776
```

```
                abund4 = 0.3672
                abund5 = 0.0783
                mass1 = 69.9242474
                mass2 = 71.9220758
                mass3 = 72.9234589
                mass4 = 73.9211778
                mass5 = 75.9214026
                numOfIsos = 5
            elif atomList[y] == "Zr":
                abund1 = 0.5145
                abund2 = 0.1122
                abund3 = 0.1715
                abund4 = 0.1738
                abund5 = 0.0280
                mass1 = 89.9047044
                mass2 = 90.9056458
                mass3 = 91.9050408
                mass4 = 93.9063152
                mass5 = 95.9082734
                numOfIsos = 5
            elif atomList[y] == "W":
                abund1 = 0.0012
                abund2 = 0.2650
                abund3 = 0.1431
                abund4 = 0.3064
                abund5 = 0.2843
                mass1 = 179.946704
                mass2 = 181.9482042
                mass3 = 182.9502230
                mass4 = 183.9509312
                mass5 = 185.9543641
                numOfIsos = 5
            #########SIX#########
            elif atomList[y] == "Ca":
                abund1 = 0.96941
                abund2 = 0.00647
                abund3 = 0.00135
                abund4 = 0.02086
                abund5 = 0.00004
                abund6 = 0.00187
                mass1 = 39.96259098
                mass2 = 41.95861801
                mass3 = 42.9587666
                mass4 = 43.9554818
                mass5 = 45.9536926
                mass6 = 47.952534
                numOfIsos = 6
            elif atomList[y] == "Se":
                abund1 = 0.0089
                abund2 = 0.0937
                abund3 = 0.0763
                abund4 = 0.2377
                abund5 = 0.4961
                abund6 = 0.0873
                mass1 = 73.9224764
                mass2 = 75.9192136
                mass3 = 76.9199140
                mass4 = 77.9173091
                mass5 = 79.9165213
                mass6 = 81.9166994
                selNum = x
                numOfIsos = 6
            elif atomList[y] == "Kr":
```

```
                abund1 = 0.00355
                abund2 = 0.02286
                abund3 = 0.11593
                abund4 = 0.11500
                abund5 = 0.56987
                abund6 = 0.17279
                mass1 = 77.9203648
                mass2 = 79.9163790
                mass3 = 81.9134836
                mass4 = 82.914136
                mass5 = 83.91150
                mass6 = 85.91061073
                numOfIsos = 6
            elif atomList[y] == "Pd":
                abund1 = 0.0102
                abund2 = 0.1114
                abund3 = 0.2233
                abund4 = 0.2733
                abund5 = 0.2646
                abund6 = 0.1172
                mass1 = 101.905609
                mass2 = 103.904036
                mass3 = 104.905085
                mass4 = 105.903486
                mass5 = 107.903892
                mass6 = 109.905153
                numOfIsos = 6
            elif atomList[y] == "Er":
                abund1 = 0.00139
                abund2 = 0.01601
                abund3 = 0.33503
                abund4 = 0.22869
                abund5 = 0.26978
                abund6 = 0.14910
                mass1 = 161.928778
                mass2 = 163.929200
                mass3 = 165.9302931
                mass4 = 166.9320482
                mass5 = 167.932370
                mass6 = 169.9354643
                numOfIsos = 6
            elif atomList[y] == "Hf":
                abund1 = 0.0016
                abund2 = 0.0526
                abund3 = 0.1860
                abund4 = 0.2728
                abund5 = 0.1362
                abund6 = 0.3508
                mass1 = 173.940046
                mass2 = 175.9414086
                mass3 = 176.9432207
                mass4 = 177.9436988
                mass5 = 178.9458161
                mass6 = 179.9465500
                numOfIsos = 6
            elif atomList[y] == "Pt":
                abund1 = 0.00014
                abund2 = 0.00782
                abund3 = 0.32967
                abund4 = 0.33832
                abund5 = 0.25242
                abund6 = 0.07163
                mass1 = 189.959932
```

```
                mass2 = 191.9610380
                mass3 = 193.9626803
                mass4 = 194.9647911
                mass5 = 195.9649515
                mass6 = 197.967893
                numOfIsos = 6
            #########SEVEN#########
            elif atomList[y] == "Mo":
                abund1 = 0.14649
                abund2 = 0.09187
                abund3 = 0.15873
                abund4 = 0.16673
                abund5 = 0.09582
                abund6 = 0.24292
                abund7 = 0.09744
                mass1 = 91.906811
                mass2 = 93.9050883
                mass3 = 94.9058421
                mass4 = 95.9046795
                mass5 = 96.9060215
                mass6 = 97.9054082
                mass7 = 99.907477
                numOfIsos = 7
            elif atomList[y] == "Ru":
                abund1 = 0.0554
                abund2 = 0.0187
                abund3 = 0.1276
                abund4 = 0.1260
                abund5 = 0.1706
                abund6 = 0.3155
                abund7 = 0.1862
                mass1 = 95.907598
                mass2 = 97.905287
                mass3 = 98.9059393
                mass4 = 99.904219
                mass5 = 100.9055821
                mass6 = 101.9043493
                mass7 = 103.905433
                numOfIsos = 7
            elif atomList[y] == "Ba":
                abund1 = 0.00106
                abund2 = 0.00101
                abund3 = 0.02417
                abund4 = 0.06592
                abund5 = 0.07854
                abund6 = 0.11232
                abund7 = 0.71698
                mass1 = 129.9063208
                mass2 = 131.9050613
                mass3 = 133.9045084
                mass4 = 134.9056886
                mass5 = 135.9045759
                mass6 = 136.9058274
                mass7 = 137.9052472
                numOfIsos = 7
            elif atomList[y] == "Nd":
                abund1 = 0.272
                abund2 = 0.122
                abund3 = 0.238
                abund4 = 0.083
                abund5 = 0.172
                abund6 = 0.057
                abund7 = 0.056
```

```
                    mass1 = 141.9077233
                    mass2 = 142.9098143
                    mass3 = 143.9100873
                    mass4 = 144.9125736
                    mass5 = 145.9131169
                    mass6 = 147.916893
                    mass7 = 149.920891
                    numOfIsos = 7
                elif atomList[y] == "Sm":
                    abund1 = 0.0307
                    abund2 = 0.1499
                    abund3 = 0.1124
                    abund4 = 0.1382
                    abund5 = 0.0738
                    abund6 = 0.2675
                    abund7 = 0.2275
                    mass1 = 143.911999
                    mass2 = 146.9148979
                    mass3 = 147.9148227
                    mass4 = 148.9171847
                    mass5 = 149.9172755
                    mass6 = 151.9197324
                    mass7 = 153.9222093
                    numOfIsos = 7
                elif atomList[y] == "Gd":
                    abund1 = 0.0020
                    abund2 = 0.0218
                    abund3 = 0.1480
                    abund4 = 0.2047
                    abund5 = 0.1565
                    abund6 = 0.2484
                    abund7 = 0.2186
                    mass1 = 151.9197910
                    mass2 = 153.9208656
                    mass3 = 154.9226220
                    mass4 = 155.9221227
                    mass5 = 156.9239601
                    mass6 = 157.9241039
                    mass7 = 159.9270541
                    numOfIsos = 7
                elif atomList[y] == "Dy":
                    abund1 = 0.00056
                    abund2 = 0.00095
                    abund3 = 0.02329
                    abund4 = 0.18889
                    abund5 = 0.25475
                    abund6 = 0.24896
                    abund7 = 0.28260
                    mass1 = 155.924283
                    mass2 = 157.924409
                    mass3 = 159.9251975
                    mass4 = 160.9269334
                    mass5 = 161.9267984
                    mass6 = 162.9287312
                    mass7 = 163.9291748
                    numOfIsos = 7
                elif atomList[y] == "Yb":
                    abund1 = 0.0013
                    abund2 = 0.0304
                    abund3 = 0.1428
                    abund4 = 0.2183
                    abund5 = 0.1613
                    abund6 = 0.3183
```

```
                    abund7 = 0.1276
                    mass1 = 167.933897
                    mass2 = 169.9347618
                    mass3 = 170.9363258
                    mass4 = 171.9363815
                    mass5 = 172.9382108
                    mass6 = 173.9388621
                    mass7 = 175.9425717
                    numOfIsos = 7
                elif atomList[y] == "Os":
                    abund1 = 0.0002
                    abund2 = 0.0159
                    abund3 = 0.0196
                    abund4 = 0.1324
                    abund5 = 0.1615
                    abund6 = 0.2626
                    abund7 = 0.4078
                    mass1 = 183.9524891
                    mass2 = 185.9538382
                    mass3 = 186.9557505
                    mass4 = 187.9558382
                    mass5 = 188.9581475
                    mass6 = 189.9584470
                    mass7 = 191.9614807
                    numOfIsos = 7
                elif atomList[y] == "Hg":
                    abund1 = 0.0015
                    abund2 = 0.0997
                    abund3 = 0.1687
                    abund4 = 0.2310
                    abund5 = 0.1318
                    abund6 = 0.2986
                    abund7 = 0.0687
                    mass1 = 195.965833
                    mass2 = 197.9667690
                    mass3 = 198.9682799
                    mass4 = 199.9683260
                    mass5 = 200.9703023
                    mass6 = 201.9706430
                    mass7 = 203.9734939
                    numOfIsos = 7
                #########EIGHT#########
                elif atomList[y] == "Cd":
                    abund1 = 0.0125
                    abund2 = 0.0089
                    abund3 = 0.1249
                    abund4 = 0.1280
                    abund5 = 0.2413
                    abund6 = 0.1222
                    abund7 = 0.2873
                    abund8 = 0.0749
                    mass1 = 105.906459
                    mass2 = 107.904184
                    mass3 = 109.9030021
                    mass4 = 110.9041781
                    mass5 = 111.9027578
                    mass6 = 112.9044017
                    mass7 = 113.9033585
                    mass8 = 115.904756
                    numOfIsos = 8
                elif atomList[y] == "Te":
                    abund1 = 0.0009
                    abund2 = 0.0255
```

```
                    abund3 = 0.0089
                    abund4 = 0.0474
                    abund5 = 0.0707
                    abund6 = 0.1884
                    abund7 = 0.3174
                    abund8 = 0.3408
                    mass1 = 119.90402
                    mass2 = 121.903043
                    mass3 = 122.9042700
                    mass4 = 123.9028179
                    mass5 = 124.9044307
                    mass6 = 125.9033117
                    mass7 = 127.9044631
                    mass8 = 129.9062244
                    telNum = x
                    numOfIsos = 8
                #########NINE#########
                elif atomList[y] == "Xe":
                    abund1 = 0.000952
                    abund2 = 0.000890
                    abund3 = 0.019102
                    abund4 = 0.264006
                    abund5 = 0.040710
                    abund6 = 0.212324
                    abund7 = 0.269086
                    abund8 = 0.104357
                    abund9 = 0.088573
                    mass1 = 123.905893
                    mass2 = 125.904274
                    mass3 = 127.9035313
                    mass4 = 128.9047794
                    mass5 = 129.9035080
                    mass6 = 130.9050824
                    mass7 = 131.9041535
                    mass8 = 133.9053945
                    mass9 = 135.907219
                    numOfIsos = 9
                #########TEN#########
                elif atomList[y] == "Sn":
                    abund1 = 0.0097
                    abund2 = 0.0066
                    abund3 = 0.0034
                    abund4 = 0.1454
                    abund5 = 0.0768
                    abund6 = 0.2422
                    abund7 = 0.0859
                    abund8 = 0.3258
                    abund9 = 0.0463
                    abund10 = 0.0579
                    mass1 = 111.904818
                    mass2 = 113.902779
                    mass3 = 114.903342
                    mass4 = 115.901741
                    mass5 = 116.902952
                    mass6 = 117.901603
                    mass7 = 118.903308
                    mass8 = 119.9021947
                    mass9 = 121.9034390
                    mass10 = 123.9052739
                    numOfIsos = 10
                else:
                    errorBool = 1
                    errorEle = atomList[y]
```

```
                while countEle <= x:
                    if numOfIsos == 2:
#"C,H,Li,B,N,Cl,V,Cu,Ga,Br,Rb,Ag,In,Sb,La,Eu,Lu,Ta,Re,Ir,Tl"
                        currentAbndList.append(((abund1**(x-
countEle))*(abund2**countEle))*nCr(x,countEle))
                        currentMzList.append((mass1*(x-countEle)) + (mass2*(countEle)))
                        countEle3 = 100000
                    while countEle3 <= (x-countEle):
                        if numOfIsos == 3: #"O,U,K,Mg,Si,Ne,Ar"
                            currentAbndList.append(((abund1**(x-countEle-
countEle3))*(abund2**countEle)*(abund3**countEle3))*(nCrThree(x, countEle, countEle3)))
                            currentMzList.append((mass1*(x-countEle-countEle3)) +
(mass2*(countEle)) + (mass3*(countEle3)))
                            countEle4 = 100000
                        while countEle4 <= (x-countEle-countEle3):
                            if numOfIsos == 4: #"Fe,S,Cr,Sr,Pb,Ce"
                                currentAbndList.append(((abund1**(x-countEle-countEle3-
countEle4))*(abund2**countEle)*(abund3**countEle3)*(abund4**countEle4))* \
                                    (nCrFour(x, countEle, countEle3,countEle4)))
                                currentMzList.append((mass1*(x-countEle-countEle3-
countEle4)) + (mass2*(countEle)) + (mass3*(countEle3)) + (mass4*(countEle4)))
                                countEle5 = 100000
                            while countEle5 <= (x-countEle-countEle3-countEle4):
                                if numOfIsos == 5: #"Ti,Ni,Zn,Ge,Zr,W"
                                    currentAbndList.append(((abund1**(x-countEle-countEle3-
countEle4-countEle5))*(abund2**countEle)*(abund3**countEle3)* \

(abund4**countEle4)*(abund5**countEle5))*(nCrFive(x, countEle, countEle3, countEle4,
countEle5)))
                                    currentMzList.append((mass1*(x-countEle-countEle3-
countEle4-countEle5)) + (mass2*(countEle)) + (mass3*(countEle3)) + \
                                        (mass4*(countEle4))+(mass5*(countEle5)))
                                    countEle6 = 100000
                                while countEle6 <= (x-countEle-countEle3-countEle4-
countEle5):
                                    if numOfIsos == 6: #"Ca,Se,Pd,Er,Pt,Kr,Hf"
                                        currentAbndList.append(((abund1**(x-countEle-
countEle3-countEle4-countEle5-countEle6))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6))*\
                                            (nCrSix(x, countEle, countEle3, countEle4,
countEle5, countEle6)))
                                        currentMzList.append((mass1*(x-countEle-countEle3-
countEle4-countEle5-countEle6)) + (mass2*(countEle)) + \
                                            (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6)))
                                        countEle7 = 100000
                                    while countEle7 <= (x-countEle-countEle3-countEle4-
countEle5-countEle6):
                                        if numOfIsos == 7: #"Mo,Ru,Ba,Nd,Sm,Gd,Dy,Yb,Os,Hg"
                                            currentAbndList.append(((abund1**(x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*\
                                                (abund7**countEle7))*\
                                                (nCrSeven(x, countEle, countEle3,
countEle4, countEle5, countEle6, countEle7)))
                                            currentMzList.append((mass1*(x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7)) + (mass2*(countEle)) + \
                                                (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7)))
                                            countEle8 = 100000
                                        while countEle8 <= (x-countEle-countEle3-countEle4-
```

```
countEle5-countEle6-countEle7):
                                            if numOfIsos == 8: #"Te,Cd"
                                                currentAbndList.append(((abund1**(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*\

(abund7**countEle7)*(abund8**countEle8))*\
                                                                (nCrEight(x, countEle, countEle3,
countEle4, countEle5, countEle6, countEle7, countEle8)))
                                                currentMzList.append((mass1*(x-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8)) + (mass2*(countEle)) + \
                                                                (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8)))
                                                countEle9 = 100000
                                            while countEle9 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7-countEle8):
                                                if numOfIsos == 9: #"Xe"
                                                    currentAbndList.append(((abund1**(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-
countEle9))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9))*\
                                                                    (nCrNine(x, countEle, countEle3,
countEle4, countEle5, countEle6, countEle7, countEle8, countEle9)))
                                                    currentMzList.append((mass1*(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9)) +
(mass2*(countEle)) + \
                                                                    (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9)))
                                                    countEle10 = 100000
                                                while countEle10 <= (x-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7-countEle8-countEle9):
                                                    if numOfIsos == 10: #"Sn"

currentAbndList.append(((abund1**(x-countEle-countEle3-countEle4-countEle5-countEle6-
countEle7-countEle8-countEle9-countEle10))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9)*\

(abund10**countEle10))*(nCrTen(x, countEle, countEle3, countEle4, countEle5, countEle6,
countEle7, countEle8, countEle9, countEle10)))
                                                                    currentMzList.append((mass1*(x-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9-countEle10))
+ (mass2*(countEle)) + \
                                                                        (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9))+\
                                                                        (mass10*(countEle10)))
                                                    countEle10 = countEle10 + 1
                                                countEle9 = countEle9 + 1
                                                countEle10 = 0
                                            countEle8 = countEle8 + 1
                                            countEle9 = 0
                                        countEle7 = countEle7 + 1
                                        countEle8 = 0
                                    countEle6 = countEle6 + 1
                                    countEle7 = 0
                                countEle5 = countEle5 + 1
```

```
                                countEle6 = 0
                            countEle4 = countEle4 + 1
                            countEle5 = 0
                        countEle3 = countEle3 + 1
                        countEle4 = 0
                    countEle = countEle +1
                    countEle3 = 0
                if len(relAbndList) == 0:
                    relAbndList[:] = currentAbndList[:]
                    mzList[:] = currentMzList[:]
                else:
                    while t < len(currentAbndList):
                        while t1 < len(relAbndList):
                            if relAbndList[t1] < 0.000001:
                                if currentAbndList[t] < 0.000001:
                                    relAbndList[t1] = 0
                                    mzList[t1] = 0
                                    currentMzList[t] = 0
                                    currentAbndList[t] = 0
                            else:
                                tempAbndList.append(float(relAbndList[t1]) *
float(currentAbndList[t]))

tempMzList.append((float(currentMzList[t])+float(mzList[t1])))
                            t1 = t1 + 1
                        t = t + 1
                        t1 = 0
                    relAbndList[:] = tempAbndList[:]
                    mzList[:] = tempMzList[:]
                    del tempAbndList[:]
                    del tempMzList[:]
                y = y + 1
                t = 0
                t1 = 0
                countEle = 0
                countEle3 = 0
                countEle4 = 0
                countEle5 = 0
                countEle6 = 0
                countEle7 = 0
                countEle8 = 0
                countEle9 = 0
                countEle10 = 0
                abund1 = 0
                abund2 = 0
                abund3 = 0
                abund4 = 0
                abund5 = 0
                abund6 = 0
                abund7 = 0
                abund8 = 0
                abund9 = 0
                abund10 = 0
                mass1 = 0
                mass2 = 0
                mass3 = 0
                mass4 = 0
                mass5 = 0
                mass6 = 0
                mass7 = 0
                mass8 = 0
                mass9 = 0
                mass10 = 0
```

```
                del currentAbndList[:]
                del currentMzList[:]
            ## Converts all isotopomers which have an extremly small abundance to zero
            for item in relAbndList:
                if relAbndList[t1] < 0.00001:
                    relAbndList[t1] = 0
                    mzList[t1] = 0
                t1 = t1 + 1
            ## Removes all masses and abundances which are zero
            mzList = filter(lambda a: a != 0, mzList)
            relAbndList = filter(lambda a: a != 0, relAbndList)
            ## If statement to include molecular formulas with elements with no isotopes
with [M+] giving an isotpoe abundance of 100 %
            if len(relAbndList) == 0:
                relAbndList.append(1)
            # If statement to include molecular formulas with elements with no isotopes
with [M+]
            if len(mzList) == 0:
                    mzList.append(0.0000000001) #adding a very small value so the entry
will be picked up by the next if statement
            x = 0
            y = 0
            ## Loop to add the masses of the elements which do not have natural isotopes
            for item in mzList:
                if mzList[x] > 0:
                    mzList[x] = mzList[x] + (numOfNa * 22.9897692809) + (numOfP *
30.97376163) + (numOfF * 18.99840322) + \
                                (numOfI * 126.904473) + (numOfAs * 74.9215965) + (numOfSc *
44.9559119) + (numOfAl * 26.98153863) + (numOfMn * 54.9380451) + \
                                (numOfCo * 58.9331950) + (numOfBe * 9.0121822) + (numOfAu *
196.9665687) + \
                                (numOfHe * 4.00260325415) + (numOfY * 88.9058483) + \
                                (numOfNb * 92.9063781) + (numOfRh * 102.905504) + (numOfCs
* 132.905451933) + (numOfBi * 208.9803987)  + (numOfTh * 232.0380553) + \
                                (numOfPa * 231.0358840) + (numOfAc * 227.0277521) +
(numOfTm * 168.9342133) + (numOfHo * 164.9303221) + (numOfTb * 158.9253468) + \
                                (numOfPr * 140.9076528)
                x = x + 1
            x = 0
            ## Substracts the mass of an electron to give the masses a positive charge
            for item in mzList:
                mzList[x] = mzList[x] - (0.000548999)
                x = x + 1
            x = 0
            y = 1
            ## Loops through all masses in the mass list, looking for isotopomers to
combine
            for item in mzList:
                currentMZ = mzList[x]
                while y < len(mzList):
                    ##  RESOLUTION(currentMZ/msResolution)  ##
                    if msMode == "Gaussian":
                        if mzList[y] > (currentMZ - (currentMZ/msResolution)):
                            if mzList[y] < (currentMZ + (currentMZ/msResolution)):
                                if currentMZ != 0:
                                    currentMZ =
mzList[x]*(relAbndList[x]/(relAbndList[x]+relAbndList[y]))+mzList[y]* \

(relAbndList[y]/(relAbndList[x]+relAbndList[y]))
                                    mzList[y] = 0
                                    mzList[x] = currentMZ
                                    relAbndList[x] = relAbndList[x] + relAbndList[y]
                                    relAbndList[y] = 0
```

```
                    elif msMode == "Centroid":
                        if mzList[y] > (currentMZ - 0.05):
                            if mzList[y] < (currentMZ + 0.05):
                                if currentMZ != 0:
                                    currentMZ =
mzList[x]*(relAbndList[x]/(relAbndList[x]+relAbndList[y]))+mzList[y]* \

(relAbndList[y]/(relAbndList[x]+relAbndList[y]))
                                mzList[y] = 0
                                mzList[x] = currentMZ
                                relAbndList[x] = relAbndList[x] + relAbndList[y]
                                relAbndList[y] = 0
                    y = y + 1
                x = x + 1
                y = x + 1
            ## Removes all masses and abundances which are zero
            mzList = filter(lambda a: a != 0, mzList)
            relAbndList = filter(lambda a: a != 0, relAbndList)
            ## Sorts the two lists by mass
            mzList, relAbndList = zip(*sorted(zip(mzList, relAbndList)))
            x = 0
            y = 0
            elementList = []
            numElement = []
            atomNumList = []
            atomList = []
            resultList.append(compound)
            resultList.append(mzList[0])
            y = 0
            while y < len(mzList) and y < 15:
                primary.txt.configure(state="normal")
                primary.txt.insert('1.0', str(uniCount) + "_" + str(formCount*100) + "_" +
str(float(relAbndList[y])*1000000) + "_0_0_0_" + str(mzList[y]) + '\n')
                primary.txt.configure(state="disabled")
                uniCount = uniCount + 1
                y = y + 1
            y = 0
            x = 0
            nitNum = 0
            oxyNum = 0
            carbNum = 0
            broNum = 0
            chloNum = 0
            sulphNum = 0
            borNum = 0
            ironNum = 0
            phosNum = 0
            fluNum = 0
            ioNum = 0
            copNum = 0
            nikNum = 0
            magNum = 0
            zincNum = 0
            selNum = 0
            telNum = 0
            resultList = []
            mzList = []
            relAbndList = []
            n = n + 1
            formCount = formCount + 1
        return
if __name__ == "__main__":
    app = simpleapp_tk(None)
```

```
    app.title('PatExtract 1.01')
    img = PhotoImage(data=iconData)
    app.tk.call('wm', 'iconphoto', app._w, img)
    app.mainloop()
```

Code S-3: *Source code for PatExtract v1.01*

This programme processes a CSV file of molecular formulas and generates isotope pattern information for each isotope cluster. The simulated isotope clusters are produced in a formate similar to that which is produced by XCMS.

```
from Tkinter import *
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from pylab import figure, show
import matplotlib.pyplot as plt
import Tkinter
import numpy as np
import math
import ttk
iconData = '''R0lGODlhEAAQAPcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArMwArZgArmQArzAAr/wBVAABVMwBV
        ZgBVmQBVzABV/wCAAACAMwCAZgCAmQCAzACA/wCqAACqMwCqZgCqmQCqzACq/wDVAADVMwDVZgDV
        mQDVzADV/wD/AAD/MwD/ZgD/mQD/zAD//zMAADMAMzMAZjMAmTMAzDMA/zMrADMrMzMrZjMrmTMr
        zDMr/zNVADNVMzNVZjNVmTNVzDNV/zOAADOAMzOAZjOAmTOAzDOA/zOqADOqMzOqZjOqmTOqzDOq
        /zPVADPVMzPVZjPVmTPVzDPV/zP/ADP/MzP/ZjP/mTP/zDP//2YAAGYAM2YAZmYAmWYAzGYA/2Yr
        AGYrM2YrZmYrmWYrzGYr/2ZVAGZVM2ZVZmZVmWZVzGZV/2aAAGaAM2aAZmaAmWaAzGaA/2aqAGaq
        M2aqZmaqmWaqzGaq/2bVAGbVM2bVZmbVmWbVzGbV/2b/AGb/M2b/Zmb/mWb/zGb//5kAAJkAM5kA
        ZpkAmZkAzJkA/5krAJkrM5krZpkrmZkrzJkr/5lVAJlVM5lVZplVmZlVzJlV/5mAAJmAM5mAZpmA
        mZmAzJmA/5mqAJmqM5mqZpmqmZmqzJmq/5nVAJnVM5nVZpnVmZnVzJnV/5n/AJn/M5n/Zpn/mZn/
        zJn//8wAAMwAM8wAZswAmcwAzMwA/8wrAMwrM8wrZswrmcwrzMwr/8xVAMxVM8xVZsxVmcxVzMxV
        /8yAAMyAM8yAZsyAmcyAzMyA/8yqAMyqM8yqZsyqmcyqzMyq/8zVAMzVM8zVZszVmczVzMzV/8z/
        AMz/M8z/Zsz/mcz/zMz///8AAP8AM/8AZv8Amf8AzP8A//8rAP8rM/8rZv8rmf8rzP8r//9VAP9V
        M/9VZv9Vmf9VzP9V//+AAP+AM/+AZv+Amf+AzP+A//+qAP+qM/+qZv+qmf+qzP+q///VAP/VM//V
        Zv/Vmf/VzP/V////AP//M///Zv//mf//zP///wAAAAAAAAAAAAAAACH5BAEAAPwALAAAAAAQABAA
        AAiKADt04CCQ4MCCCAlaOLiQwwWCDiEyHMihYoeHBzs0pFiRA4CCDy0OvFAQgEmKAjUe9Ghyn0mL
        EgWaBLCv5kuFHlm2rOkSAByCM2n25NnzQsudQ3kGrRlDaFKbAAwEnUq1alUiGjRkyDBEA5GvQ4Zk
        wOr169YPRJSYFUsEBIivH5KETfI1idywXwMCADs=
        '''
ionChoices0 = []
ionChoices1 = []
ionChoices2 = []
ionChoices3 = []
ionChoices4 = []
ionChoices0 = [
    '[M]'
]
ionChoices1 = [
    '[M+H]',
    '[M+]',
    '[M-H2O+H]',
    '[M-2(H2O)+H]',
    '[M-3(H2O)+H]',
    '[M-H2O+Na]',
    '[M-H2O+NH4]',
    '[M+NH4]',
    '[M+Na]',
    '[M+K]',
    '[M+Li]',
    '[M+Ag]',
    '[M-HBr+H]',
    '[M-SO3+H]',
    '[M-H+Fe2+]',
    '[M-2H+Fe3+]',
    '[2M+H]',
    '[2M+NH4]',
    '[2M+Na]',
    '[2M+K]',
]
ionChoices2 = [
    '[M+2H]',
    '[M-H2O+2H]',
    '[M-2(H2O)+2H]',
    '[M-3(H2O)+2H]',
```

```
    '[M+NH4+H]',
    '[M+Na+H]',
    '[M+K+H]',
    '[M+Li+H]',
    '[M+2(NH4)]',
    '[M+2Na]',
    '[M+2K]',
    '[M-HBr+2H]',
    '[M-SO3+2H]',
    '[M+Fe2+]',
    '[M-H+Fe3+]',
    '[2M+2H]',
    '[2M+NH4+H]',
    '[2M+Na+H]',
    '[2M+K+H]',
]
ionChoices3 = [
    '[M+3H]',
    '[M-H2O+3H]',
    '[M-2(H2O)+3H]',
    '[M-3(H2O)+3H]',
    '[M+NH4+2H]',
    '[M+Na+2H]',
    '[M+K+2H]',
    '[M+Li+2H]',
    '[M+2(NH4)+H]',
    '[M+2Na+H]',
    '[M+2K+H]',
    '[M+3(NH4)]',
    '[M+3Na]',
    '[M+3K]',
    '[M+NH4+Na+H]',
    '[M+NH4+2Na]',
    '[M+2(NH4)+Na]',
    '[M-HBr+3H]',
    '[M-SO3+3H]',
    '[M+H+Fe2+]',
    '[M+Fe3+]',
    '[M+Na+Fe2+]',
    '[M+NH4+Fe2+]',
]
ionChoices4 = [
    '[M+4H]',
    '[M-H2O+4H]',
    '[M-2(H2O)+4H]',
    '[M-3(H2O)+4H]',
    '[M+NH4+3H]',
    '[M+Na+3H]',
    '[M+K+3H]',
    '[M+Li+3H]',
    '[M+2(NH4)+2H]',
    '[M+2Na+2H]',
    '[M+2K+2H]',
    '[M+3(NH4)+H]',
    '[M+3Na+H]',
    '[M+3K+H]',
    '[M+4(NH4)]',
    '[M+4Na]',
    '[M+4K]',
    '[M+NH4+Na+2H]',
    '[M+NH4+2Na+H]',
    '[M+2(NH4)+Na+H]',
    '[M-HBr+4H]',
```

```
    '[M-SO3+4H]',
    '[M+2H+Fe2+]',
    '[M+Na+H+Fe2+]',
    '[M+NH4+H+Fe2+]',
    '[M+H+Fe3+]',
    '[M+Na+Fe3+]',
    '[M+NH4+Fe3+]',
]
ionChoices5 = []
ionChoices6 = []
ionChoices7 = []
ionChoices8 = []
ionChoices5 = [
    '[M-H]',
    '[M-]',
    '[M+HCOOH-H]',
    '[M+CH3COOH-H]',
    '[M+Na+HCOOH-2H]',
    '[M+Na+CH3COOH-2H]',
    '[M-H2O-H]',
    '[M-2(H2O)-H]',
    '[M-3(H2O)-H]',
    '[M+NH4-2H]',
    '[M+Na-2H]',
    '[M+K-2H]',
    '[M+Cl]',
    '[M+Br]',
    '[M+NaCl+Cl]',
    '[M+NaCl+HCOOH-H]',
    '[M+Na+2(HCOOH)-2H]',
    '[2M-H]',
    '[2M+NH4-2H]',
    '[2M+Na-2H]',
    '[2M+K-2H]',
]
ionChoices6 = [
    '[M-2H]',
    '[M+HCOOH-2H]',
    '[M+CH3COOH-2H]',
    '[M+Na+HCOOH-3H]',
    '[M+Na+CH3COOH-3H]',
    '[M-H2O-2H]',
    '[M-2(H2O)-2H]',
    '[M-3(H2O)-2H]',
    '[M+NH4-3H]',
    '[M+Na-3H]',
    '[M+K-3H]',
    '[M+Cl-H]',
    '[M+2Cl]',
    '[M+Br-H]',
    '[M+2Br]',
]
ionChoices7 = [
    '[M-3H]',
    '[M+HCOOH-3H]',
    '[M+CH3COOH-3H]',
    '[M+Na+HCOOH-4H]',
    '[M+Na+CH3COOH-4H]',
    '[M-H2O-3H]',
    '[M-2(H2O)-3H]',
    '[M-3(H2O)-3H]',
    '[M+NH4-4H]',
    '[M+Na-4H]',
```

```
    '[M+K-4H]',
    '[M+Cl-2H]',
    '[M+3Cl]',
    '[M+Br-2H]',
    '[M+3Br]',
]
ionChoices8 = [
    '[M-4H]',
    '[M+HCOOH-4H]',
    '[M+CH3COOH-4H]',
    '[M+Na+HCOOH-5H]',
    '[M+Na+CH3COOH-5H]',
    '[M-H2O-4H]',
    '[M-2(H2O)-4H]',
    '[M-3(H2O)-4H]',
    '[M+NH4-5H]',
    '[M+Na-5H]',
    '[M+K-5H]',
    '[M+Cl-3H]',
    '[M+4Cl]',
    '[M+Br-3H]',
    '[M+4Br]',
]
class simpleapp_tk(Tkinter.Tk):
    def __init__(primary,parent):
        Tkinter.Tk.__init__(primary,parent)
        primary.parent = parent
        primary.initialize()
        primary.update()
        primary.minsize(primary.winfo_width(), primary.winfo_height())
    def initialize(primary):
        primary.grid()
        primary.grid_rowconfigure(25,weight=1)
        primary.grid_columnconfigure(1,weight=1)
        primary.grid_columnconfigure(2,weight=1)
        primary.userForm = StringVar()
        primary.chargeState = StringVar()
        primary.numOfAdducts = StringVar()
        primary.userMZ = StringVar()
        primary.carbonAbund = StringVar()
        primary.resolution = StringVar()
        primary.errorMessage = Tkinter.StringVar()
        primary.adductIonStr = StringVar()
        primary.coAdductIonStr = StringVar()
        primary.massList = []
        primary.abundanceList = []
        primary.A2IonStr = StringVar()
        primary.A3IonStr = StringVar()
        primary.A4IonStr = StringVar()
        primary.A2abnd = StringVar()
        primary.A3abnd = StringVar()
        primary.A4abnd = StringVar()
        primary.A2cState = StringVar()
        primary.A3cState = StringVar()
        primary.A4cState = StringVar()
        primary.numOfC13 = StringVar()
        primary.numOfN15 = StringVar()
        primary.numOfH2 = StringVar()
        primary.coNumOfC13 = StringVar()
        primary.coNumOfN15 = StringVar()
        primary.coNumOfH2 = StringVar()
        primary.coCarbonAbund = StringVar()
        primary.userCoForm = StringVar()
```

```
        primary.coChargeState = StringVar()
        primary.coCompoundNum = StringVar()
        primary.coComAbnd = StringVar()
        primary.labelCutOffPC = StringVar()
        primary.chargeMode = StringVar()
        color = '0.9'
        plotFrame = Frame(primary, width=654, height=500)
        plotFrame.pack(fill="both", expand=True)
        plotFrame.grid_propagate(False)
        plotFrame.grid_rowconfigure(25, weight=1)
        plotFrame.grid_columnconfigure(1, weight=1)
        plotFrame.grid(row=1, column=1, columnspan=2, rowspan=25, sticky="neswn")
        fig = Figure()
        rect = fig.patch
        rect.set_facecolor(color)
        primary.canvas = FigureCanvasTkAgg(fig, plotFrame)
        primary.canvas.get_tk_widget().grid(row=1, column=1, columnspan=2, rowspan=25,
sticky="neswn")
        toolbar = NavigationToolbar2TkAgg(primary.canvas,primary)
        toolbar.grid(row=0, column=1, columnspan=5, sticky=W)
        primary.ax2 = fig.add_subplot(1,1,1)
        primary.ax2.set_xlabel('Mass to charge (m/z)')
        primary.ax2.set_title('Isotope Pattern')
        primary.ax2.set_ylabel('Relative Abundance')
        primary.canvas.draw()
        primary.compoundLabel = Label(primary, text="Molecular formula:")
        primary.compoundLabel.grid(row=3, column=3, sticky=E)
        primary.mzEntry = Entry(primary, width=22, textvariable=primary.userForm)
        primary.mzEntry.bind('<Return>', primary.grph)# Added to allow enter pressing to
generate
        primary.mzEntry.grid(row=3, column=4, sticky=W)
        primary.adNumLabel = Label(primary, text="# of adducts:")
        primary.adNumLabel.grid(row=5, column=3, sticky=E)
        primary.adNumVal = Spinbox(primary, from_=1, to=4,
textvariable=primary.numOfAdducts, width=2, command=primary.AdductRefresh)
        primary.adNumVal.grid(row=5, column=4, sticky=W)
        primary.extraC13Label = Label(primary, text="Incorporated 13C:")
        primary.extraC13Label.grid(row=3, column=5, sticky=E)
        primary.extraC13Val = Spinbox(primary, from_=0, to=20,
textvariable=primary.numOfC13, width=2)
        primary.extraC13Val.grid(row=3, column=6, sticky=W)
        primary.carbonLabel = Label(primary, text="13C abundance:")
        primary.carbonLabel.grid(row=3, column=7, sticky=E)
        primary.carbonVal = Entry(primary, textvariable=primary.carbonAbund, width=5)
        primary.carbonVal.grid(row=3, column=8, sticky=W)
        primary.carbonVal.insert(0, 1.07)
        primary.carbonVal.bind('<Return>', primary.grph)
        primary.carbonValPcLabel = Label(primary, text="%")
        primary.carbonValPcLabel.grid(row=3, column=9, sticky=E)
        primary.extraC13Label = Label(primary, text="15N:")
        primary.extraC13Label.grid(row=4, column=5, sticky=E)
        primary.extraC13Val = Spinbox(primary, from_=0, to=20,
textvariable=primary.numOfN15, width=2)
        primary.extraC13Val.grid(row=4, column=6, sticky=W)
        primary.extraC13Label = Label(primary, text="2H:")
        primary.extraC13Label.grid(row=5, column=5, sticky=E)
        primary.extraC13Val = Spinbox(primary, from_=0, to=20,
textvariable=primary.numOfH2, width=2)
        primary.extraC13Val.grid(row=5, column=6, sticky=W)
        primary.ionAdLabel = Label(primary, text="Ion adduct:")
        primary.ionAdLabel.grid(row=7, column=3, sticky=E)
        primary.adductOptions = ttk.Combobox(primary, state='readonly', width=22,
textvariable=primary.adductIonStr)
```

```
        primary.adductOptions["values"] = ionChoices1
        primary.adductOptions.set("[M+H]")
        primary.adductOptions.grid(row=7, column=4, sticky=W)
        primary.adductOptions.bind('<Return>', primary.grph)
        primary.chargeLabel = Label(primary, text="Ion charge:")
        primary.chargeLabel.grid(row=7, column=5, sticky=E)
        primary.chargeState.set("1")
        primary.chargeVal = Spinbox(primary, from_=0, to=4,
textvariable=primary.chargeState, width=2, command=primary.ionAdList)
        primary.chargeVal.grid(row=7, column=6, sticky=W)
        primary.ionAbundLabel = Label(primary, text="Abundance:")
        primary.ionAbundLabel.grid(row=7, column=7, sticky=E)
        primary.ionAbundVal = Entry(primary, width=5)
        primary.ionAbundVal.grid(row=7, column=8, sticky=W)
        primary.ionAbundVal.insert(0, 100)
        primary.ionAbundVal.configure(state="disabled")
        primary.ionAbundPcLabel = Label(primary, text="%")
        primary.ionAbundPcLabel.grid(row=7, column=9, sticky=E)
        #####Adduct2
        primary.ionAdLabelA2 = Label(primary, text="Adduct #2:")
        primary.ionAdLabelA2.grid(row=8, column=3, sticky=E)
        primary.adductOptionsA2 = ttk.Combobox(primary, state='readonly', width=22,
textvariable=primary.A2IonStr)
        primary.adductOptionsA2["values"] = ionChoices1 #startIons
        primary.adductOptionsA2.set("[M+H]")
        primary.adductOptionsA2.grid(row=8, column=4, sticky=W)
        primary.adductOptionsA2.configure(state="disabled")
        primary.adductOptionsA2.bind('<Return>', primary.grph)
        primary.A2cState.set("1")
        primary.A2chargeVal = Spinbox(primary, from_=0, to=4,
textvariable=primary.A2cState, width=2, command=primary.ionAdListA2)
        primary.A2chargeVal.grid(row=8, column=6, sticky=W)
        primary.A2chargeVal.configure(state="disabled")
        primary.A2AbundVal = Entry(primary, textvariable=primary.A2abnd, width=5)
        primary.A2AbundVal.grid(row=8, column=8, sticky=W)
        primary.A2AbundVal.insert(0, 100)
        primary.A2AbundVal.bind('<Return>', primary.grph)
        primary.A2AbundVal.configure(state="disabled")
        primary.A2AbundPcLabel = Label(primary, text="%")
        primary.A2AbundPcLabel.grid(row=8, column=9, sticky=E)
        #####Adduct3
        primary.ionAdLabelA3 = Label(primary, text="Adduct #3:")
        primary.ionAdLabelA3.grid(row=9, column=3, sticky=E)
        primary.adductOptionsA3 = ttk.Combobox(primary, state='readonly', width=22,
textvariable=primary.A3IonStr)
        primary.adductOptionsA3["values"] = ionChoices1 #startIons
        primary.adductOptionsA3.set("[M+H]")
        primary.adductOptionsA3.grid(row=9, column=4, sticky=W)
        primary.adductOptionsA3.configure(state="disabled")
        primary.adductOptionsA3.bind('<Return>', primary.grph)
        primary.A3cState.set("1")
        primary.A3chargeVal = Spinbox(primary, from_=0, to=4,
textvariable=primary.A3cState, width=2, command=primary.ionAdListA3)
        primary.A3chargeVal.grid(row=9, column=6, sticky=W)
        primary.A3chargeVal.configure(state="disabled")
        primary.A3AbundVal = Entry(primary, textvariable=primary.A3abnd, width=5)
        primary.A3AbundVal.grid(row=9, column=8, sticky=W)
        primary.A3AbundVal.insert(0, 100)
        primary.A3AbundVal.bind('<Return>', primary.grph)
        primary.A3AbundVal.configure(state="disabled")
        primary.A3AbundPcLabel = Label(primary, text="%")
        primary.A3AbundPcLabel.grid(row=9, column=9, sticky=E)
        #####Adduct4
```

```
        primary.ionAdLabelA4 = Label(primary, text="Adduct #4:")
        primary.ionAdLabelA4.grid(row=10, column=3, sticky=E)
        primary.adductOptionsA4 = ttk.Combobox(primary, state='readonly', width=22,
textvariable=primary.A4IonStr)
        primary.adductOptionsA4["values"] = ionChoices1 #startIons
        primary.adductOptionsA4.set("[M+H]")
        primary.adductOptionsA4.grid(row=10, column=4, sticky=W)
        primary.adductOptionsA4.configure(state="disabled")
        primary.adductOptionsA4.bind('<Return>', primary.grph)
        primary.A4cState.set("1")
        primary.A4chargeVal = Spinbox(primary, from_=0, to=4,
textvariable=primary.A4cState, width=2, command=primary.ionAdListA4)
        primary.A4chargeVal.grid(row=10, column=6, sticky=W)
        primary.A4chargeVal.configure(state="disabled")
        primary.A4AbundVal = Entry(primary, textvariable=primary.A4abnd, width=5)
        primary.A4AbundVal.grid(row=10, column=8, sticky=W)
        primary.A4AbundVal.insert(0, 100)
        primary.A4AbundVal.bind('<Return>', primary.grph)
        primary.A4AbundVal.configure(state="disabled")
        primary.A4AbundPcLabel = Label(primary, text="%")
        primary.A4AbundPcLabel.grid(row=10, column=9, sticky=E)
        #####Co-Compound
        primary.coCompoundLabel = Label(primary, text="Coeluting compounds:")
        primary.coCompoundLabel.grid(row=11, column=3, sticky=E+S, pady=10)
        primary.coCompoundVal = Spinbox(primary, from_=0, to=1,
textvariable=primary.coCompoundNum, width=2, command=primary.AdductRefresh)
        primary.coCompoundVal.grid(row=11, column=4, sticky=W)
        primary.coCompoundLabel = Label(primary, text="Molecular formula:")
        primary.coCompoundLabel.grid(row=12, column=3, sticky=E)
        primary.coMZEntry = Entry(primary, width=22, textvariable=primary.userCoForm)
        primary.coMZEntry.grid(row=12, column=4, sticky=W)
        primary.coMZEntry.configure(state="disabled")
        primary.coMZEntry.bind('<Return>', primary.grph)
        primary.coExtraC13Label = Label(primary, text="Incorporated 13C:")
        primary.coExtraC13Label.grid(row=12, column=5, sticky=E)
        primary.coExtraC13Val = Spinbox(primary, from_=0, to=20,
textvariable=primary.coNumOfC13, width=2)
        primary.coExtraC13Val.grid(row=12, column=6, sticky=W)
        primary.coExtraC13Val.configure(state="disabled")
        primary.coExtraN15Label = Label(primary, text="15N:")
        primary.coExtraN15Label.grid(row=13, column=5, sticky=E)
        primary.coExtraN15Val = Spinbox(primary, from_=0, to=20,
textvariable=primary.coNumOfN15, width=2)
        primary.coExtraN15Val.grid(row=13, column=6, sticky=W)
        primary.coExtraN15Val.configure(state="disabled")
        primary.coExtraH2Label = Label(primary, text="2H")
        primary.coExtraH2Label.grid(row=14, column=5, sticky=E)
        primary.coExtraH2Val = Spinbox(primary, from_=0, to=20,
textvariable=primary.coNumOfH2, width=2)
        primary.coExtraH2Val.grid(row=14, column=6, sticky=W)
        primary.coExtraH2Val.configure(state="disabled")
        primary.coCarbonLabel = Label(primary, text="13C abundance:")
        primary.coCarbonLabel.grid(row=12, column=7, sticky=E)
        primary.coCarbonVal = Entry(primary, textvariable=primary.coCarbonAbund, width=5)
        primary.coCarbonVal.grid(row=12, column=8, sticky=W)
        primary.coCarbonVal.insert(0, 1.07)
        primary.coCarbonVal.configure(state="disabled")
        primary.coCarbonVal.bind('<Return>', primary.grph)
        primary.coCarbonValPcLabel = Label(primary, text="%")
        primary.coCarbonValPcLabel.grid(row=12, column=9, sticky=E)
        primary.coIonAdLabel = Label(primary, text="Co-Ion adduct:")
        primary.coIonAdLabel.grid(row=15, column=3, sticky=E)
        primary.coAdductOptions = ttk.Combobox(primary, state='readonly', width=22,
```

```
textvariable=primary.coAdductIonStr)
        primary.coAdductOptions["values"] = ionChoices1 #startIons
        primary.coAdductOptions.set("[M+H]")
        primary.coAdductOptions.grid(row=15, column=4, sticky=W)
        primary.coAdductOptions.configure(state="disabled")
        primary.coAdductOptions.bind('<Return>', primary.grph)
        primary.coChargeLabel = Label(primary, text="Ion charge:")
        primary.coChargeLabel.grid(row=15, column=5, sticky=E)
        primary.coChargeState.set("1")
        primary.coChargeVal = Spinbox(primary, from_=0, to=4,
textvariable=primary.coChargeState, width=2, command=primary.ionAdListCoCom)
        primary.coChargeVal.grid(row=15, column=6, sticky=W)
        primary.coChargeVal.configure(state="disabled")
        primary.coIonAbundLabel = Label(primary, text="Abundance:")
        primary.coIonAbundLabel.grid(row=15, column=7, sticky=E)
        primary.coIonAbundVal = Entry(primary, textvariable=primary.coComAbnd, width=5)
        primary.coIonAbundVal.grid(row=15, column=8, sticky=W)
        primary.coIonAbundVal.insert(0, 100)
        primary.coIonAbundVal.bind('<Return>', primary.grph)
        primary.coIonAbundVal.configure(state="disabled")
        primary.coIonAbundPcLabel = Label(primary, text="%")
        primary.coIonAbundPcLabel.grid(row=15, column=9, sticky=E)
        ######Mass and Abundance Box
        primary.txt = Text(primary, borderwidth=3, relief="sunken")
        primary.txt.config(font=("consolas", 11), undo=True, wrap='word', width=50,
height=15)
        primary.txt.grid(row=20, column=4, columnspan=4, rowspan=4, sticky=E, padx=2,
pady=5)
        primary.txt.configure(state="disabled")
        scrollb = Scrollbar(primary, command=primary.txt.yview)
        scrollb.grid(row=20, column=8, rowspan=4, sticky=W+N+S, pady=5)
        primary.txt['yscrollcommand'] = scrollb.set
        ######Below Graph
        primary.resLabel = Label(primary, text="Resolution:")
        primary.resLabel.grid(row=26, column=1, sticky=E)
        primary.resVal = Entry(primary, textvariable=primary.resolution, width=10)
        primary.resVal.grid(row=26, column=2, sticky=W)
        primary.resVal.insert(0, 45000)
        primary.resVal.bind('<Return>', primary.grph)
        primary.cutOffLabel = Label(primary, text="Label cut off %:")
        primary.cutOffLabel.grid(row=28, column=1, sticky=E)
        primary.cutOfVal = Entry(primary, textvariable=primary.labelCutOffPC, width=10)
        primary.cutOfVal.grid(row=28, column=2, sticky=W)
        primary.cutOfVal.insert(0, 99.0)
        primary.cutOfVal.bind('<Return>', primary.grph)
        primary.chargeModeLabel = Label(primary, text="Ion mode:")
        primary.chargeModeLabel.grid(row=29, column=1, sticky=E)
        primary.chargeMode_value = StringVar()
        primary.chargeMode_value.trace('w', primary.AdductChargeRefresh)
        primary.chargeMode = ttk.Combobox(primary, state='readonly',
textvariable=primary.chargeMode_value, width=10)
        primary.chargeMode['values'] = ('Positive', 'Negative')
        primary.chargeMode.set("Positive")
        primary.chargeMode.grid(row=29, column=2, sticky=W)
        primary.chargeMode.bind('<Return>', primary.grph)
        errorLabel = Tkinter.Label(primary, textvariable=primary.errorMessage, fg = "red")
        errorLabel.grid(row=25, column=3, columnspan=4)
        primary.errorMessage.set("")
    def massCalc(primary, eleForm, charge, carbon13, molecularAdduct, numOfC13s, numOfN15s,
numOfH2s):
        compound = eleForm
        removeEleC = numOfC13s
        removeEleN = numOfN15s
```

```
        extraRemoveEleH = numOfH2s
        adductForm = molecularAdduct
        resNum = primary.resolution.get()
        removeEleO = 0
        removeEleH = 0
        removeEleBr = 0
        removeEleS = 0
        electronChange = 0.000548999
        mNum = 1
        seqN = ""
        atomCombo = ""
        atomList = []
        elementList = []
        atomNumList = []
        comboList = []
        sequenceMass = 0
        sequenceAbundance = 0
        mzList = []
        relAbndList = []
        n = 0
        pattern = '([A-Z])([a-z]*)(\d*)'
        pattern2 = '(\d*)'
        errorBool = 0
        errorEle = ""
        if float(carbon13) > 100:
            carbon13 = 100
        elif float(carbon13) < 0:
            carbon13 = 0
        if adductForm == "[M+H]" or adductForm == "[M+2H]" or adductForm == "[M+3H]" or
adductForm == "[M+4H]":
            adductForm = "H" + charge
        elif adductForm == "[M+]":
            adductForm = ""
        elif adductForm == "[M-HBr+H]" or adductForm == "[M-HBr+2H]" or adductForm == "[M-
HBr+3H]" or adductForm == "[M-HBr+4H]":
            adductForm = "H" + charge
            removeEleBr = 1
            removeEleH = 1
        elif adductForm == "[M-H2O+H]" or adductForm == "[M-H2O+2H]" or adductForm == "[M-
H2O+3H]" or adductForm == "[M-H2O+4H]":
            adductForm = "H" + charge
            removeEleO = 1
            removeEleH = 2
        elif adductForm == "[M-H2O+Na]" or adductForm == "[M-H2O+2Na]" or adductForm ==
"[M-H2O+3Na]" or adductForm == "[M-H2O+4Na]":
            adductForm = "Na" + charge
            removeEleO = 1
            removeEleH = 2
        elif adductForm == "[M-H2O+NH4]":
            adductForm = "NH4"
            removeEleO = 1
            removeEleH = 2
        elif adductForm == "[M-2(H2O)+H]" or adductForm == "[M-2(H2O)+2H]" or adductForm ==
"[M-2(H2O)+3H]" or adductForm == "[M-2(H2O)+4H]":
            adductForm = "H" + charge
            removeEleO = 2
            removeEleH = 4
        elif adductForm == "[M-3(H2O)+H]" or adductForm == "[M-3(H2O)+2H]" or adductForm ==
"[M-3(H2O)+3H]" or adductForm == "[M-3(H2O)+4H]":
            adductForm = "H" + charge
            removeEleO = 3
            removeEleH = 6
        elif adductForm == "[M+NH4]" or adductForm == "[M+NH4+H]" or adductForm ==
```

```
"[M+NH4+2H]" or adductForm == "[M+NH4+3H]":
            adductForm = "NH"  + str((int(charge)+3))
        elif adductForm == "[M+Na]" or adductForm == "[M+2Na]" or adductForm == "[M+3Na]"
or adductForm == "[M+4Na]":
            adductForm = "Na" + charge
        elif adductForm == "[M+K]" or adductForm == "[M+2K]" or adductForm == "[M+3K]" or
adductForm == "[M+4K]":
            adductForm = "K" + charge
        elif adductForm == "[M+Li]" or adductForm == "[M+2Li]" or adductForm == "[M+3Li]"
or adductForm == "[M+4Li]":
            adductForm = "Li" + charge
        elif adductForm == "[M+Ag]" or adductForm == "[M+2Ag]" or adductForm == "[M+3Ag]"
or adductForm == "[M+4Ag]":
            adductForm = "Ag" + charge
        elif adductForm == "[2M+H]":
            adductForm = "H"
            mNum = 2
        elif adductForm == "[2M+NH4]":
            adductForm = "NH4"
            mNum = 2
        elif adductForm == "[2M+Na]":
            adductForm = "Na"
            mNum = 2
        elif adductForm == "[2M+K]":
            adductForm = "K"
            mNum = 2
        elif adductForm == "[M-H+Fe2+]":
            adductForm = "Fe"
            removeEleH = 1
        elif adductForm == "[M-2H+Fe3+]":
            adductForm = "Fe"
            removeEleH = 2
        elif adductForm == "[M-SO3+H]" or adductForm == "[M-SO3+2H]" or adductForm == "[M-
SO3+3H]" or adductForm == "[M-SO3+4H]":
            adductForm = "H" + charge
            removeEleO = 3
            removeEleS = 1
        #Charge2
        elif adductForm == "[M+Na+H]" or adductForm == "[M+Na+2H]" or adductForm ==
"[M+Na+3H]":
            adductForm = "NaH"  + str((int(charge)-1))
        elif adductForm == "[M+K+H]" or adductForm == "[M+K+2H]" or adductForm ==
"[M+K+3H]":
            adductForm = "KH"  + str((int(charge)-1))
        elif adductForm == "[M+Li+H]" or adductForm == "[M+Li+2H]" or adductForm ==
"[M+Li+3H]":
            adductForm = "LiH"  + str((int(charge)-1))
        elif adductForm == "[M+Ag+H]" or adductForm == "[M+Ag+2H]" or adductForm ==
"[M+Ag+3H]":
            adductForm = "AgH"  + str((int(charge)-1))
        elif adductForm == "[M+2(NH4)]" or adductForm == "[M+2(NH4)+H]" or adductForm ==
"[M+2(NH4)+2H]":
            adductForm = "N2H" + str((int(charge)+6))
        elif adductForm == "[M+Na+NH4]" or adductForm == "[M+NH4+Na+H]" or adductForm ==
"[M+NH4+Na+2H]":
            adductForm = "NaNH" + str((int(charge)+2))
        elif adductForm == "[M+Fe2+]":
            adductForm = "Fe"
        elif adductForm == "[M-H+Fe3+]":
            adductForm = "Fe"
            removeEleH = 1
        elif adductForm == "[2M+2H]":
            adductForm = "H2"
```

```
            mNum = 2
        elif adductForm == "[2M+NH4+H]":
            adductForm = "NH5"
            mNum = 2
        elif adductForm == "[2M+Na+H]":
            adductForm = "NaH"
            mNum = 2
        elif adductForm == "[2M+K+H]":
            adductForm = "KH"
            mNum = 2
        #Charge3
        elif adductForm == "[M+2Na+H]" or adductForm == "[M+2Na+2H]":
            adductForm = "Na2H" + str((int(charge)-2))
        elif adductForm == "[M+2K+H]" or adductForm == "[M+2K+2H]":
            adductForm = "K2H" + str((int(charge)-2))
        elif adductForm == "[M+3(NH4)]":
            adductForm = "N3H12"
        elif adductForm == "[M+NH4+2Na]":
            adductForm = "NH4Na2"
        elif adductForm == "[M+2(NH4)+Na]":
            adductForm = "N2H8Na"
        elif adductForm == "[M+H+Fe2+]":
            adductForm = "HFe"
        elif adductForm == "[M+Fe3+]":
            adductForm = "Fe"
        elif adductForm == "[M+Na+Fe2+]":
            adductForm = "NaFe"
        elif adductForm == "[M+NH4+Fe2+]":
            adductForm = "NH4Fe"
        #Charge4
        elif adductForm == "[M+3K+H]":
            adductForm = "K3H"
        elif adductForm == "[M+3(NH4)+H]":
            adductForm = "N3H13"
        elif adductForm == "[M+4(NH4)]":
            adductForm = "N4H16"
        elif adductForm == "[M+3Na+H]":
            adductForm = "Na3H"
        elif adductForm == "[M+NH4+2Na+H]":
            adductForm = "Na2NH5"
        elif adductForm == "[M+2(NH4)+Na+H]":
            adductForm = "NaN2H9"
        elif adductForm == "[M+2H+Fe2+]":
            adductForm = "H2Fe"
        elif adductForm == "[M+Na+H+Fe2+]":
            adductForm = "NaHFe"
        elif adductForm == "[M+NH4+H+Fe2+]":
            adductForm = "NH5Fe"
        elif adductForm == "[M+H+Fe3+]":
            adductForm = "HFe"
        elif adductForm == "[M+Na+Fe3+]":
            adductForm = "NaFe"
        elif adductForm == "[M+NH4+Fe3+]":
            adductForm = "NH4Fe"
        #Charge5 (1 neg)
        elif adductForm == "[M-H]" or adductForm == "[M-2H]" or adductForm == "[M-3H]" or
adductForm == "[M-4H]":
            removeEleH = int(charge)
            adductForm = ""
            electronChange = -0.000548999
        elif adductForm == "[M-]":
            adductForm = ""
            electronChange = -0.000548999
```

```
        elif adductForm == "[M-H2O-H]" or adductForm == "[M-H2O-2H]" or adductForm == "[M-
H2O-3H]" or adductForm == "[M-H2O-4H]":
            removeEleO = 1
            removeEleH = 2 + int(charge)
            adductForm = ""
            electronChange = -0.000548999
        elif adductForm == "[M-2(H2O)-H]" or adductForm == "[M-2(H2O)-2H]" or adductForm ==
"[M-2(H2O)-3H]" or adductForm == "[M-2(H2O)-4H]":
            removeEleO = 2
            removeEleH = 4 + int(charge)
            adductForm = ""
            electronChange = -0.000548999
        elif adductForm == "[M-3(H2O)-H]" or adductForm == "[M-3(H2O)-2H]" or adductForm ==
"[M-3(H2O)-3H]" or adductForm == "[M-3(H2O)-4H]":
            removeEleO = 3
            removeEleH = 6 + int(charge)
            adductForm = ""
            electronChange = -0.000548999
        elif adductForm == "[M+Cl]" or adductForm == "[M+2Cl]" or adductForm == "[M+3Cl]"
or adductForm == "[M+4Cl]":
            adductForm = "Cl" + charge
            electronChange = -0.000548999
        elif adductForm == "[M+Br]" or adductForm == "[M+2Br]" or adductForm == "[M+3Br]"
or adductForm == "[M+4Br]":
            adductForm = "Br" + charge
        elif adductForm == "[M+NH4-2H]" or adductForm == "[M+NH4-3H]" or adductForm ==
"[M+NH4-4H]" or adductForm == "[M+NH4-5H]":
            adductForm = "NH4"
            removeEleH = 1 + int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+Na-2H]" or adductForm == "[M+Na-3H]" or adductForm ==
"[M+Na-4H]" or adductForm == "[M+Na-5H]":
            adductForm = "Na"
            removeEleH = 1 + int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+K-2H]" or adductForm == "[M+K-3H]" or adductForm == "[M+K-
4H]" or adductForm == "[M+K-5H]":
            adductForm = "K"
            removeEleH = 1 + int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+Cl-H]" or adductForm == "[M+Cl-2H]" or adductForm == "[M+Cl-
3H]":
            adductForm = "Cl"
            removeEleH = 1 + int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+Br-H]" or adductForm == "[M+Br-2H]" or adductForm == "[M+Br-
3H]":
            adductForm = "Br"
            removeEleH = 1 + int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+HCOOH-H]" or adductForm == "[M+HCOOH-2H]" or adductForm ==
"[M+HCOOH-3H]" or adductForm == "[M+HCOOH-4H]":
            adductForm = "HCO2"
            removeEleH = -1 + int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+CH3COOH-H]" or adductForm == "[M+CH3COOH-2H]" or adductForm
== "[M+CH3COOH-3H]" or adductForm == "[M+CH3COOH-4H]":
            adductForm = "C2H3O2"
            removeEleH = -1 + int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+Na+HCOOH-2H]" or adductForm == "[M+Na+HCOOH-3H]" or
adductForm == "[M+Na+HCOOH-4H]" or adductForm == "[M+Na+HCOOH-5H]":
            adductForm = "NaHCO2"
```

```
            removeEleH = int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+Na+CH3COOH-2H]" or adductForm == "[M+Na+CH3COOH-3H]" or
adductForm == "[M+Na+CH3COOH-4H]" or adductForm == "[M+Na+CH3COOH-5H]":
            adductForm = "NaC2H3O2"
            removeEleH = int(charge)
            electronChange = -0.000548999
        elif adductForm == "[M+NaCl+Cl]":
            adductForm = "NaCl2"
            electronChange = -0.000548999
        elif adductForm == "[M+NaCl+HCOOH-H]":
            adductForm = "NaClCHO2"
            electronChange = -0.000548999
        elif adductForm == "[M+Na+2(HCOOH)-2H]":
            adductForm = "NaC2H2O4"
            electronChange = -0.000548999
        elif adductForm == "[2M-H]":
            adductForm = "H"
            removeEleH = 2
            mNum = 2
            electronChange = -0.000548999
        elif adductForm == "[2M+NH4-2H]":
            adductForm = "NH4"
            removeEleH = 2
            mNum = 2
            electronChange = -0.000548999
        elif adductForm == "[2M+Na-2H]":
            adductForm = "Na"
            removeEleH = 2
            mNum = 2
            electronChange = -0.000548999
        elif adductForm == "[2M+K-2H]":
            adductForm = "K"
            removeEleH = 2
            mNum = 2
            electronChange = -0.000548999
        elif adductForm == "[M]":
            adductForm = ""
            electronChange = 0
            charge = 1
        else:
            adductForm = "FeClBr4Ag" #crazy adduct form to know something has gone wrong
        compound = compound * mNum + adductForm
        def nCr(n,r):
            f = math.factorial
            return f(n) / f(r) / f(n-r)
        def nCrThree(n,b,c):
            f = math.factorial
            return f(n) / (f(n-(b+c)) * f(b) * f(c))
        def nCrFour(n,b,c,d):
            f = math.factorial
            return f(n) / (f(n-(b+c+d)) * f(b) * f(c) * f(d))
        def nCrFive(n,b,c,d,e):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e)) * f(b) * f(c) * f(d) * f(e))
        def nCrSix(n,b,c,d,e,g):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g)) * f(b) * f(c) * f(d) * f(e) * f(g))
        def nCrSeven(n,b,c,d,e,g,h):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h))
        def nCrEight(n,b,c,d,e,g,h,i):
            f = math.factorial
```

```
            return f(n) / (f(n-(b+c+d+e+g+h+i)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h) *
f(i))
        def nCrNine(n,b,c,d,e,g,h,i,j):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i+j)) * f(b) * f(c) * f(d) * f(e) * f(g) * f(h)
* f(i) * f(j))
        def nCrTen(n,b,c,d,e,g,h,i,j,k):
            f = math.factorial
            return f(n) / (f(n-(b+c+d+e+g+h+i+j+k)) * f(b) * f(c) * f(d) * f(e) * f(g) *
f(h) * f(i) * f(j) * f(k))
        for match in re.finditer(pattern, compound):
            s = match.start()
            e = match.end()
            elementComp = compound[s:e]
            elementList.append(compound[s:e])
        for item in elementList:
            element = ''.join([i for i in elementList[n] if not i.isdigit()])
            numElement = ''.join([i for i in elementList[n] if i.isdigit()])
            if numElement == "":
                numElement = "1"
            elementList[n] = element + "." + numElement
            n = n + 1
        n = 0
        for item in elementList:
            sepEleList = elementList[n].split(".")
            atomNumList.append(int(sepEleList[1]))
            atomList.append(sepEleList[0])
            n = n + 1
        n = 0
        n1 = 1
        t = 0
        t1 = 0
        countEle = 0
        countEle3 = 0
        countEle4 = 0
        countEle5 = 0
        countEle6 = 0
        countEle7 = 0
        countEle8 = 0
        countEle9 = 0
        countEle10 = 0
        abund1 = 1
        abund2 = 0
        abund3 = 0
        abund4 = 0
        abund5 = 0
        abund6 = 0
        abund7 = 0
        abund8 = 0
        abund9 = 0
        abund10 = 0
        mass1 = 0
        mass2 = 0
        mass3 = 0
        mass4 = 0
        mass5 = 0
        mass6 = 0
        mass7 = 0
        mass8 = 0
        mass9 = 0
        mass10 = 0
        currentAbndList = []
        currentMzList = []
```

```
        tempAbndList = []
        tempMzList = []
        numOfHe = 0
        numOfBe = 0
        numOfF = 0
        numOfNa = 0
        numOfAl = 0
        numOfP = 0
        numOfSc = 0
        numOfMn = 0
        numOfCo = 0
        numOfAs = 0
        numOfY = 0
        numOfNb = 0
        numOfRh = 0
        numOfI = 0
        numOfCs = 0
        numOfPr = 0
        numOfTb = 0
        numOfHo = 0
        numOfTm = 0
        numOfAu = 0
        numOfBi = 0
        numOfAc = 0
        numOfTh = 0
        numOfPa = 0
        numOfIsos = 0
        ## Loop to combine multiple entries of the same element
        for each in atomList:
            while n1 < len(atomList):
                if atomNumList[n] == 0:
                    atomList[n] = ""
                if atomList[n] == atomList[n1]:
                    atomList[n1] = ""
                    atomNumList[n] = atomNumList[n] + atomNumList[n1]
                    atomNumList[n1] = 0
                n1 = n1 +1
            n = n +1
            n1 = n +1
        n = 0
        n1 = 0
        tempList = []
        ## Loop to rearange the atomList and atomNumList to put oxygens first to make the
code run faster
        for each in atomList:
            if atomList[n] == "O":
                tempList.append(atomList[n])
                atomList[n] = ""
                for everyEntry in atomList:
                    tempList.append(everyEntry)
                atomList[:] = tempList[:]
                del tempList[:]
                tempList.append(atomNumList[n])
                atomNumList[n] = 0
                for everyEntry in atomNumList:
                    tempList.append(everyEntry)
                atomNumList[:] = tempList[:]
                del tempList[:]
            n = n +1
        n = 0
        while 0 in atomNumList: atomNumList.remove(0)
        while "" in atomList: atomList.remove("")
        ## Loop to remove elements from the atom list which do not have natural isotopes so
```

```
they are not
        ## included in relative abundance calculations
        for each in atomList:
            if atomList[n] == "He": #4.00260325415
                numOfHe = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Be": #9.0121822
                numOfBe = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "C": #12
                if atomNumList[n] < int(removeEleC):
                    removeEleC = str(atomNumList[n])
            elif atomList[n] == "N": #14.00307
                if atomNumList[n] < int(removeEleN):
                    removeEleN = str(atomNumList[n])
            elif atomList[n] == "F": #18.99840
                numOfF = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Na": #22.98977
                numOfNa = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Al": #26.98153863
                numOfAl = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "P": #30.97376
                numOfP = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Sc": #44.9559119
                numOfSc = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Mn": #54.9380451
                numOfMn = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Co": #58.9331950
                numOfCo = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "As": #74.92160
                numOfAs = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Y": #88.9058483
                numOfY = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Nb": #92.9063781
                numOfNb = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "Rh": #102.905504
                numOfRh = atomNumList[n]
                atomNumList[n] = 0
                atomList[n] = ""
            elif atomList[n] == "I": #126.90447
                numOfI = atomNumList[n]
```

```
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Cs": #132.905451933
            numOfCs = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Pr": #140.9076528
            numOfPr = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Tb": #158.9253468
            numOfTb = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Ho": #164.9303221
            numOfHo = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Tm": #168.9342133
            numOfTm = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Au": #196.9665687
            numOfAu = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Bi": #208.9803987
            numOfBi = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Ac": #227.0277521
            numOfAc = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Th": #232.0380553
            numOfTh = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        elif atomList[n] == "Pa": #231.0358840
            numOfPa = atomNumList[n]
            atomNumList[n] = 0
            atomList[n] = ""
        n = n + 1
    while 0 in atomNumList: atomNumList.remove(0)
    while "" in atomList: atomList.remove("")
    n = 0
    for each in atomList:
        n = atomNumList[n1]
        if atomList[n1] == "C":
            abund1 = (100 - float(carbon13))/100
            abund2 = (float(carbon13))/100
            mass1 = 12
            mass2 = 13.0033548378
            n = n - int(removeEleC)
            numOfIsos = 2
        elif atomList[n1] == "H":
            abund1 = 0.999885
            abund2 = 0.000115
            mass1 = 1.00782503207
            mass2 = 2.0141017778
            n = n - int(removeEleH) - int(extraRemoveEleH)
            numOfIsos = 2
        elif atomList[n1] == "Li":
```

```
                abund1 = 0.0759
                abund2 = 0.9241
                mass1 = 6.015122795
                mass2 = 7.01600455
                numOfIsos = 2
            elif atomList[n1] == "B":
                abund1 = 0.199
                abund2 = 0.801
                mass1 = 10.0129370
                mass2 = 11.0093054
                numOfIsos = 2
            elif atomList[n1] == "N":
                abund1 = 0.99636
                abund2 = 0.00364
                mass1 = 14.0030740048
                mass2 = 15.0001088982
                n = n - int(removeEleN)
                numOfIsos = 2
            elif atomList[n1] == "Cl":
                abund1 = 0.7576
                abund2 = 0.2424
                mass1 = 34.96885268
                mass2 = 36.96590259
                numOfIsos = 2
            elif atomList[n1] == "V":
                abund1 = 0.00250
                abund2 = 0.99750
                mass1 = 49.9471585
                mass2 = 50.9439595
                numOfIsos = 2
            elif atomList[n1] == "Cu":
                abund1 = 0.6915
                abund2 = 0.3085
                mass1 = 62.9295975
                mass2 = 64.9277895
                numOfIsos = 2
            elif atomList[n1] == "Ga":
                abund1 = 0.60108
                abund2 = 0.39892
                mass1 = 68.9255736
                mass2 = 70.9247013
                numOfIsos = 2
            elif atomList[n1] == "Br":
                abund1 = 0.5069
                abund2 = 0.4931
                mass1 = 78.9183371
                mass2 = 80.9162906
                n = n - removeEleBr
                numOfIsos = 2
            elif atomList[n1] == "Rb":
                abund1 = 0.7217
                abund2 = 0.2783
                mass1 = 84.911789738
                mass2 = 86.909180527
                numOfIsos = 2
            elif atomList[n1] == "Ag":
                abund1 = 0.51839
                abund2 = 0.48161
                mass1 = 106.905097
                mass2 = 108.904752
                numOfIsos = 2
            elif atomList[n1] == "In":
                abund1 = 0.0429
```

```
            abund2 = 0.9571
            mass1 = 112.904058
            mass2 = 114.903878
            numOfIsos = 2
        elif atomList[n1] == "Sb":
            abund1 = 0.5721
            abund2 = 0.4279
            mass1 = 120.903815
            mass2 = 122.9042140
            numOfIsos = 2
        elif atomList[n1] == "La":
            abund1 = 0.0009
            abund2 = 0.99910
            mass1 = 137.907112
            mass2 = 138.9063533
            numOfIsos = 2
        elif atomList[n1] == "Eu":
            abund1 = 0.4781
            abund2 = 0.5219
            mass1 = 150.9198502
            mass2 = 152.9212303
            numOfIsos = 2
        elif atomList[n1] == "Lu":
            abund1 = 0.9741
            abund2 = 0.0259
            mass1 = 174.9407718
            mass2 = 175.9426863
            numOfIsos = 2
        elif atomList[n1] == "Ta":
            abund1 = 0.00012
            abund2 = 0.99988
            mass1 = 179.9474648
            mass2 = 180.9479958
            numOfIsos = 2
        elif atomList[n1] == "Re":
            abund1 = 0.3740
            abund2 = 0.6260
            mass1 = 184.9529550
            mass2 = 186.9557531
            numOfIsos = 2
        elif atomList[n1] == "Ir":
            abund1 = 0.373
            abund2 = 0.627
            mass1 = 190.9605940
            mass2 = 192.9629264
            numOfIsos = 2
        elif atomList[n1] == "Tl":
            abund1 = 0.2952
            abund2 = 0.7048
            mass1 = 202.9723442
            mass2 = 204.9744275
            numOfIsos = 2
        #########THREE#########
        elif atomList[n1] == "O":
            abund1 = 0.99757
            abund2 = 0.00205
            abund3 = 0.00038
            mass1 = 15.99491461956
            mass2 = 17.9991610
            mass3 = 16.99913170
            n = n - removeEleO
            numOfIsos = 3
        elif atomList[n1] == "Ne":
```

```
                abund1 = 0.9048
                abund2 = 0.0027
                abund3 = 0.0925
                mass1 = 19.9924401754
                mass2 = 20.99384668
                mass3 = 21.991385114
                numOfIsos = 3
            elif atomList[n1] == "Mg":
                abund1 = 0.7899
                abund2 = 0.1000
                abund3 = 0.1101
                mass1 = 23.985041700
                mass2 = 24.98583692
                mass3 = 25.982592929
                numOfIsos = 3
            elif atomList[n1] == "Si":
                abund1 = 0.92223
                abund2 = 0.04685
                abund3 = 0.03092
                mass1 = 27.9769265325
                mass2 = 28.97649470
                mass3 = 29.97377017
                numOfIsos = 3
            elif atomList[n1] == "Ar":
                abund1 = 0.003336
                abund2 = 0.000629
                abund3 = 0.996035
                mass1 = 35.967545106
                mass2 = 37.9627324
                mass3 = 39.9623831225
                numOfIsos = 3
            elif atomList[n1] == "K":
                abund1 = 0.932581
                abund2 = 0.067302
                abund3 = 0.000117
                mass1 = 38.96370668
                mass2 = 40.96182576
                mass3 = 39.96399848
                numOfIsos = 3
            elif atomList[n1] == "U":
                abund1 = 0.000054
                abund2 = 0.007204
                abund3 = 0.992742
                mass1 = 234.0409521
                mass2 = 235.0439299
                mass3 = 238.0507882
                numOfIsos = 3
            #########FOUR#########
            elif atomList[n1] == "S":
                abund1 = 0.9493
                abund2 = 0.0076
                abund3 = 0.0429
                abund4 = 0.0002
                mass1 = 31.97207100
                mass2 = 32.97145876
                mass3 = 33.96786690
                mass4 = 35.96708076
                n = n - removeEleS
                numOfIsos = 4
            elif atomList[n1] == "Cr":
                abund1 = 0.04345
                abund2 = 0.83789
                abund3 = 0.09501
```

```
            abund4 = 0.02365
            mass1 = 49.9460442
            mass2 = 51.9405075
            mass3 = 52.9406494
            mass4 = 53.9388804
            numOfIsos = 4
        elif atomList[n1] == "Fe":
            abund1 = 0.05845
            abund2 = 0.91754
            abund3 = 0.02119
            abund4 = 0.00282
            mass1 = 53.9396105
            mass2 = 55.9349375
            mass3 = 56.9353940
            mass4 = 57.9332756
            numOfIsos = 4
        elif atomList[n1] == "Sr":
            abund1 = 0.0056
            abund2 = 0.0986
            abund3 = 0.0700
            abund4 = 0.8258
            mass1 = 83.913425
            mass2 = 85.9092607309
            mass3 = 86.9088774970
            mass4 = 87.905612257
            numOfIsos = 4
        elif atomList[n1] == "Ce":
            abund1 = 0.00185
            abund2 = 0.00251
            abund3 = 0.88450
            abund4 = 0.11114
            mass1 = 135.907172
            mass2 = 137.905991
            mass3 = 139.9054387
            mass4 = 141.909244
            numOfIsos = 4
        elif atomList[n1] == "Pb":
            abund1 = 0.014
            abund2 = 0.241
            abund3 = 0.221
            abund4 = 0.524
            mass1 = 203.9730436
            mass2 = 205.9744653
            mass3 = 206.9758969
            mass4 = 207.9766521
            numOfIsos = 4
        #########FIVE#########
        elif atomList[n1] == "Ti":
            abund1 = 0.0825
            abund2 = 0.0744
            abund3 = 0.7372
            abund4 = 0.0541
            abund5 = 0.0518
            mass1 = 45.9526316
            mass2 = 46.9517631
            mass3 = 47.9479463
            mass4 = 48.9478700
            mass5 = 49.9447912
            numOfIsos = 5
        elif atomList[n1] == "Ni":
            abund1 = 0.680769
            abund2 = 0.262231
            abund3 = 0.011399
```

```
            abund4 = 0.036345
            abund5 = 0.009256
            mass1 = 57.9353429
            mass2 = 59.9307864
            mass3 = 60.9310560
            mass4 = 61.9283451
            mass5 = 63.9279660
            numOfIsos = 5
        elif atomList[n1] == "Zn":
            abund1 = 0.4917
            abund2 = 0.2773
            abund3 = 0.0404
            abund4 = 0.1845
            abund5 = 0.0061
            mass1 = 63.9291422
            mass2 = 65.9260334
            mass3 = 66.9271273
            mass4 = 67.9248442
            mass5 = 69.9253193
            numOfIsos = 5
        elif atomList[n1] == "Ge":
            abund1 = 0.2038
            abund2 = 0.2731
            abund3 = 0.0776
            abund4 = 0.3672
            abund5 = 0.0783
            mass1 = 69.9242474
            mass2 = 71.9220758
            mass3 = 72.9234589
            mass4 = 73.9211778
            mass5 = 75.9214026
            numOfIsos = 5
        elif atomList[n1] == "Zr":
            abund1 = 0.5145
            abund2 = 0.1122
            abund3 = 0.1715
            abund4 = 0.1738
            abund5 = 0.0280
            mass1 = 89.9047044
            mass2 = 90.9056458
            mass3 = 91.9050408
            mass4 = 93.9063152
            mass5 = 95.9082734
            numOfIsos = 5
        elif atomList[n1] == "W":
            abund1 = 0.0012
            abund2 = 0.2650
            abund3 = 0.1431
            abund4 = 0.3064
            abund5 = 0.2843
            mass1 = 179.946704
            mass2 = 181.9482042
            mass3 = 182.9502230
            mass4 = 183.9509312
            mass5 = 185.9543641
            numOfIsos = 5
        #########SIX#########
        elif atomList[n1] == "Ca":
            abund1 = 0.96941
            abund2 = 0.00647
            abund3 = 0.00135
            abund4 = 0.02086
            abund5 = 0.00004
```

```
            abund6 = 0.00187
            mass1 = 39.96259098
            mass2 = 41.95861801
            mass3 = 42.9587666
            mass4 = 43.9554818
            mass5 = 45.9536926
            mass6 = 47.952534
            numOfIsos = 6
        elif atomList[n1] == "Se":
            abund1 = 0.0089
            abund2 = 0.0937
            abund3 = 0.0763
            abund4 = 0.2377
            abund5 = 0.4961
            abund6 = 0.0873
            mass1 = 73.9224764
            mass2 = 75.9192136
            mass3 = 76.9199140
            mass4 = 77.9173091
            mass5 = 79.9165213
            mass6 = 81.9166994
            numOfIsos = 6
        elif atomList[n1] == "Kr":
            abund1 = 0.00355
            abund2 = 0.02286
            abund3 = 0.11593
            abund4 = 0.11500
            abund5 = 0.56987
            abund6 = 0.17279
            mass1 = 77.9203648
            mass2 = 79.9163790
            mass3 = 81.9134836
            mass4 = 82.914136
            mass5 = 83.91150
            mass6 = 85.91061073
            numOfIsos = 6
        elif atomList[n1] == "Pd":
            abund1 = 0.0102
            abund2 = 0.1114
            abund3 = 0.2233
            abund4 = 0.2733
            abund5 = 0.2646
            abund6 = 0.1172
            mass1 = 101.905609
            mass2 = 103.904036
            mass3 = 104.905085
            mass4 = 105.903486
            mass5 = 107.903892
            mass6 = 109.905153
            numOfIsos = 6
        elif atomList[n1] == "Er":
            abund1 = 0.00139
            abund2 = 0.01601
            abund3 = 0.33503
            abund4 = 0.22869
            abund5 = 0.26978
            abund6 = 0.14910
            mass1 = 161.928778
            mass2 = 163.929200
            mass3 = 165.9302931
            mass4 = 166.9320482
            mass5 = 167.932370
            mass6 = 169.9354643
```

```
            numOfIsos = 6
        elif atomList[n1] == "Hf":
            abund1 = 0.0016
            abund2 = 0.0526
            abund3 = 0.1860
            abund4 = 0.2728
            abund5 = 0.1362
            abund6 = 0.3508
            mass1 = 173.940046
            mass2 = 175.9414086
            mass3 = 176.9432207
            mass4 = 177.9436988
            mass5 = 178.9458161
            mass6 = 179.9465500
            numOfIsos = 6
        elif atomList[n1] == "Pt":
            abund1 = 0.00014
            abund2 = 0.00782
            abund3 = 0.32967
            abund4 = 0.33832
            abund5 = 0.25242
            abund6 = 0.07163
            mass1 = 189.959932
            mass2 = 191.9610380
            mass3 = 193.9626803
            mass4 = 194.9647911
            mass5 = 195.9649515
            mass6 = 197.967893
            numOfIsos = 6
        #########SEVEN#########
        elif atomList[n1] == "Mo":
            abund1 = 0.14649
            abund2 = 0.09187
            abund3 = 0.15873
            abund4 = 0.16673
            abund5 = 0.09582
            abund6 = 0.24292
            abund7 = 0.09744
            mass1 = 91.906811
            mass2 = 93.9050883
            mass3 = 94.9058421
            mass4 = 95.9046795
            mass5 = 96.9060215
            mass6 = 97.9054082
            mass7 = 99.907477
            numOfIsos = 7
        elif atomList[n1] == "Ru":
            abund1 = 0.0554
            abund2 = 0.0187
            abund3 = 0.1276
            abund4 = 0.1260
            abund5 = 0.1706
            abund6 = 0.3155
            abund7 = 0.1862
            mass1 = 95.907598
            mass2 = 97.905287
            mass3 = 98.9059393
            mass4 = 99.904219
            mass5 = 100.9055821
            mass6 = 101.9043493
            mass7 = 103.905433
            numOfIsos = 7
        elif atomList[n1] == "Ba":
```

```
            abund1 = 0.00106
            abund2 = 0.00101
            abund3 = 0.02417
            abund4 = 0.06592
            abund5 = 0.07854
            abund6 = 0.11232
            abund7 = 0.71698
            mass1 = 129.9063208
            mass2 = 131.9050613
            mass3 = 133.9045084
            mass4 = 134.9056886
            mass5 = 135.9045759
            mass6 = 136.9058274
            mass7 = 137.9052472
            numOfIsos = 7
        elif atomList[n1] == "Nd":
            abund1 = 0.272
            abund2 = 0.122
            abund3 = 0.238
            abund4 = 0.083
            abund5 = 0.172
            abund6 = 0.057
            abund7 = 0.056
            mass1 = 141.9077233
            mass2 = 142.9098143
            mass3 = 143.9100873
            mass4 = 144.9125736
            mass5 = 145.9131169
            mass6 = 147.916893
            mass7 = 149.920891
            numOfIsos = 7
        elif atomList[n1] == "Sm":
            abund1 = 0.0307
            abund2 = 0.1499
            abund3 = 0.1124
            abund4 = 0.1382
            abund5 = 0.0738
            abund6 = 0.2675
            abund7 = 0.2275
            mass1 = 143.911999
            mass2 = 146.9148979
            mass3 = 147.9148227
            mass4 = 148.9171847
            mass5 = 149.9172755
            mass6 = 151.9197324
            mass7 = 153.9222093
            numOfIsos = 7
        elif atomList[n1] == "Gd":
            abund1 = 0.0020
            abund2 = 0.0218
            abund3 = 0.1480
            abund4 = 0.2047
            abund5 = 0.1565
            abund6 = 0.2484
            abund7 = 0.2186
            mass1 = 151.9197910
            mass2 = 153.9208656
            mass3 = 154.9226220
            mass4 = 155.9221227
            mass5 = 156.9239601
            mass6 = 157.9241039
            mass7 = 159.9270541
            numOfIsos = 7
```

```
            elif atomList[n1] == "Dy":
                abund1 = 0.00056
                abund2 = 0.00095
                abund3 = 0.02329
                abund4 = 0.18889
                abund5 = 0.25475
                abund6 = 0.24896
                abund7 = 0.28260
                mass1 = 155.924283
                mass2 = 157.924409
                mass3 = 159.9251975
                mass4 = 160.9269334
                mass5 = 161.9267984
                mass6 = 162.9287312
                mass7 = 163.9291748
                numOfIsos = 7
            elif atomList[n1] == "Yb":
                abund1 = 0.0013
                abund2 = 0.0304
                abund3 = 0.1428
                abund4 = 0.2183
                abund5 = 0.1613
                abund6 = 0.3183
                abund7 = 0.1276
                mass1 = 167.933897
                mass2 = 169.9347618
                mass3 = 170.9363258
                mass4 = 171.9363815
                mass5 = 172.9382108
                mass6 = 173.9388621
                mass7 = 175.9425717
                numOfIsos = 7
            elif atomList[n1] == "Os":
                abund1 = 0.0002
                abund2 = 0.0159
                abund3 = 0.0196
                abund4 = 0.1324
                abund5 = 0.1615
                abund6 = 0.2626
                abund7 = 0.4078
                mass1 = 183.9524891
                mass2 = 185.9538382
                mass3 = 186.9557505
                mass4 = 187.9558382
                mass5 = 188.9581475
                mass6 = 189.9584470
                mass7 = 191.9614807
                numOfIsos = 7
            elif atomList[n1] == "Hg":
                abund1 = 0.0015
                abund2 = 0.0997
                abund3 = 0.1687
                abund4 = 0.2310
                abund5 = 0.1318
                abund6 = 0.2986
                abund7 = 0.0687
                mass1 = 195.965833
                mass2 = 197.9667690
                mass3 = 198.9682799
                mass4 = 199.9683260
                mass5 = 200.9703023
                mass6 = 201.9706430
                mass7 = 203.9734939
```

```
            numOfIsos = 7
        #########EIGHT#########
        elif atomList[n1] == "Cd":
            abund1 = 0.0125
            abund2 = 0.0089
            abund3 = 0.1249
            abund4 = 0.1280
            abund5 = 0.2413
            abund6 = 0.1222
            abund7 = 0.2873
            abund8 = 0.0749
            mass1 = 105.906459
            mass2 = 107.904184
            mass3 = 109.9030021
            mass4 = 110.9041781
            mass5 = 111.9027578
            mass6 = 112.9044017
            mass7 = 113.9033585
            mass8 = 115.904756
            numOfIsos = 8
        elif atomList[n1] == "Te":
            abund1 = 0.0009
            abund2 = 0.0255
            abund3 = 0.0089
            abund4 = 0.0474
            abund5 = 0.0707
            abund6 = 0.1884
            abund7 = 0.3174
            abund8 = 0.3408
            mass1 = 119.90402
            mass2 = 121.903043
            mass3 = 122.9042700
            mass4 = 123.9028179
            mass5 = 124.9044307
            mass6 = 125.9033117
            mass7 = 127.9044631
            mass8 = 129.9062244
            numOfIsos = 8
        #########NINE#########
        elif atomList[n1] == "Xe":
            abund1 = 0.000952
            abund2 = 0.000890
            abund3 = 0.019102
            abund4 = 0.264006
            abund5 = 0.040710
            abund6 = 0.212324
            abund7 = 0.269086
            abund8 = 0.104357
            abund9 = 0.088573
            mass1 = 123.905893
            mass2 = 125.904274
            mass3 = 127.9035313
            mass4 = 128.9047794
            mass5 = 129.9035080
            mass6 = 130.9050824
            mass7 = 131.9041535
            mass8 = 133.9053945
            mass9 = 135.907219
            numOfIsos = 9
        #########TEN#########
        elif atomList[n1] == "Sn":
            abund1 = 0.0097
            abund2 = 0.0066
```

```
                abund3 = 0.0034
                abund4 = 0.1454
                abund5 = 0.0768
                abund6 = 0.2422
                abund7 = 0.0859
                abund8 = 0.3258
                abund9 = 0.0463
                abund10 = 0.0579
                mass1 = 111.904818
                mass2 = 113.902779
                mass3 = 114.903342
                mass4 = 115.901741
                mass5 = 116.902952
                mass6 = 117.901603
                mass7 = 118.903308
                mass8 = 119.9021947
                mass9 = 121.9034390
                mass10 = 123.9052739
                numOfIsos = 10
            else:
                errorBool = 1
                errorEle = atomList[n1]
                numOfIsos = 0
            while countEle <= n:
                if numOfIsos == 2:
#"C,H,Li,B,N,Cl,V,Cu,Ga,Br,Rb,Ag,In,Sb,La,Eu,Lu,Ta,Re,Ir,Tl"
                    currentAbndList.append(((abund1**(n-
countEle))*(abund2**countEle))*nCr(n,countEle))
                    currentMzList.append((mass1*(n-countEle)) + (mass2*(countEle)))
                    countEle3 = 100000
                while countEle3 <= (n-countEle):
                    if numOfIsos == 3: #"O,U,K,Mg,Si,Ne,Ar"
                        currentAbndList.append(((abund1**(n-countEle-
countEle3))*(abund2**countEle)*(abund3**countEle3))*(nCrThree(n, countEle, countEle3)))
                        currentMzList.append((mass1*(n-countEle-countEle3)) +
(mass2*(countEle)) + (mass3*(countEle3)))
                        countEle4 = 100000
                    while countEle4 <= (n-countEle-countEle3):
                        if numOfIsos == 4: #"Fe,S,Cr,Sr,Pb,Ce"
                            currentAbndList.append(((abund1**(n-countEle-countEle3-
countEle4))*(abund2**countEle)*(abund3**countEle3)*(abund4**countEle4))* \
                                (nCrFour(n, countEle, countEle3,countEle4)))
                            currentMzList.append((mass1*(n-countEle-countEle3-countEle4)) +
(mass2*(countEle)) + (mass3*(countEle3)) + (mass4*(countEle4)))
                            countEle5 = 100000
                        while countEle5 <= (n-countEle-countEle3-countEle4):
                            if numOfIsos == 5: #"Ti,Ni,Zn,Ge,Zr,W"
                                currentAbndList.append(((abund1**(n-countEle-countEle3-
countEle4-countEle5))*(abund2**countEle)*(abund3**countEle3)* \
                                    (abund4**countEle4)*(abund5**countEle5))*(nCrFive(n,
countEle, countEle3, countEle4, countEle5)))
                                currentMzList.append((mass1*(n-countEle-countEle3-
countEle4-countEle5)) + (mass2*(countEle)) + (mass3*(countEle3)) + \
                                    (mass4*(countEle4))+(mass5*(countEle5)))
                                countEle6 = 100000
                            while countEle6 <= (n-countEle-countEle3-countEle4-countEle5):
                                if numOfIsos == 6: #"Ca,Se,Pd,Er,Pt,Kr,Hf"
                                    currentAbndList.append(((abund1**(n-countEle-countEle3-
countEle4-countEle5-countEle6))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6))*\
                                        (nCrSix(n, countEle, countEle3, countEle4,
countEle5, countEle6)))
```

```
                                    currentMzList.append((mass1*(n-countEle-countEle3-
countEle4-countEle5-countEle6)) + (mass2*(countEle)) + \
                                        (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6)))
                                    countEle7 = 100000
                                while countEle7 <= (n-countEle-countEle3-countEle4-
countEle5-countEle6):
                                    if numOfIsos == 7: #"Mo,Ru,Ba,Nd,Sm,Gd,Dy,Yb,Os,Hg"
                                        currentAbndList.append(((abund1**(n-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*\
                                            (abund7**countEle7))*\
                                            (nCrSeven(n, countEle, countEle3, countEle4,
countEle5, countEle6, countEle7)))
                                        currentMzList.append((mass1*(n-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7)) + (mass2*(countEle)) + \
                                            (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7)))
                                        countEle8 = 100000
                                    while countEle8 <= (n-countEle-countEle3-countEle4-
countEle5-countEle6-countEle7):
                                        if numOfIsos == 8: #"Te,Cd"
                                            currentAbndList.append(((abund1**(n-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*\
                                                (abund7**countEle7)*(abund8**countEle8))*\
                                                (nCrEight(n, countEle, countEle3,
countEle4, countEle5, countEle6, countEle7, countEle8)))
                                            currentMzList.append((mass1*(n-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8)) + (mass2*(countEle)) + \
                                                (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8)))
                                            countEle9 = 100000
                                        while countEle9 <= (n-countEle-countEle3-countEle4-
countEle5-countEle6-countEle7-countEle8):
                                            if numOfIsos == 9: #"Xe"
                                                currentAbndList.append(((abund1**(n-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-
countEle9))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9))*\
                                                    (nCrNine(n, countEle, countEle3,
countEle4, countEle5, countEle6, countEle7, countEle8, countEle9)))
                                                currentMzList.append((mass1*(n-countEle-
countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9)) +
(mass2*(countEle)) + \
                                                    (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9)))
                                                countEle10 = 100000
                                            while countEle10 <= (n-countEle-countEle3-
countEle4-countEle5-countEle6-countEle7-countEle8-countEle9):
                                                if numOfIsos == 10: #"Sn"
                                                    currentAbndList.append(((abund1**(n-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9-
countEle10))*(abund2**countEle)*\

(abund3**countEle3)*(abund4**countEle4)*(abund5**countEle5)*(abund6**countEle6)*(abund7**co
untEle7)*(abund8**countEle8)*(abund9**countEle9)*\
```

```
                                                        (abund10**countEle10))*(nCrTen(n,
countEle, countEle3, countEle4, countEle5, countEle6, countEle7, countEle8, countEle9,
countEle10)))
                                                    currentMzList.append((mass1*(n-
countEle-countEle3-countEle4-countEle5-countEle6-countEle7-countEle8-countEle9-countEle10))
+ (mass2*(countEle)) + \
                                                        (mass3*(countEle3)) +
(mass4*(countEle4))+(mass5*(countEle5))+(mass6*(countEle6))+(mass7*(countEle7))+(mass8*(cou
ntEle8))+(mass9*(countEle9))+\
                                                        (mass10*(countEle10)))
                                                countEle10 = countEle10 + 1
                                            countEle9 = countEle9 + 1
                                            countEle10 = 0
                                        countEle8 = countEle8 + 1
                                        countEle9 = 0
                                    countEle7 = countEle7 + 1
                                    countEle8 = 0
                                countEle6 = countEle6 + 1
                                countEle7 = 0
                            countEle5 = countEle5 + 1
                            countEle6 = 0
                        countEle4 = countEle4 + 1
                        countEle5 = 0
                    countEle3 = countEle3 + 1
                    countEle4 = 0
                countEle = countEle +1
                countEle3 = 0
            if len(relAbndList) == 0:
                relAbndList[:] = currentAbndList[:]
                mzList[:] = currentMzList[:]
            else:
                while t < len(currentAbndList):
                    while t1 < len(relAbndList):

                        if relAbndList[t1] < 0.000001:
                            if currentAbndList[t] < 0.000001:
                                relAbndList[t1] = 0
                                mzList[t1] = 0
                                currentMzList[t] = 0
                                currentAbndList[t] = 0
                        else:
                            tempAbndList.append(float(relAbndList[t1]) *
float(currentAbndList[t]))
                            tempMzList.append((float(currentMzList[t])+float(mzList[t1])))
                        t1 = t1 + 1
                    t = t + 1
                    t1 = 0
                relAbndList[:] = tempAbndList[:]
                mzList[:] = tempMzList[:]
                del tempAbndList[:]
                del tempMzList[:]
            n1 = n1 + 1
            t = 0
            t1 = 0
            countEle = 0
            countEle3 = 0
            countEle4 = 0
            countEle5 = 0
            countEle6 = 0
            countEle7 = 0
            countEle8 = 0
            countEle9 = 0
            countEle10 = 0
```

```
            abund1 = 0
            abund2 = 0
            abund3 = 0
            abund4 = 0
            abund5 = 0
            abund6 = 0
            abund7 = 0
            abund8 = 0
            abund9 = 0
            abund10 = 0
            mass1 = 0
            mass2 = 0
            mass3 = 0
            mass4 = 0
            mass5 = 0
            mass6 = 0
            mass7 = 0
            mass8 = 0
            mass9 = 0
            mass10 = 0
            numOfIsos = 0
            del currentAbndList[:]
            del currentMzList[:]
        n = 0
        n1 = 1
        currentMZ = 0
        for item in relAbndList:
            if relAbndList[t1] < 0.00001:
                relAbndList[t1] = 0
                mzList[t1] = 0
            t1 = t1 + 1
        n = 0
        mzList = filter(lambda a: a != 0, mzList)
        relAbndList = filter(lambda a: a != 0, relAbndList)
        for item in mzList:
            currentMZ = mzList[n]
            while n1 < len(mzList):
                if mzList[n1] > (currentMZ - (currentMZ/float(resNum))):#float(resNum)):
                    if mzList[n1] < (currentMZ +
(currentMZ/float(resNum))):#float(resNum)):
                        if currentMZ != 0:
                            currentMZ =
mzList[n]*(relAbndList[n]/(relAbndList[n]+relAbndList[n1]))+mzList[n1]*(relAbndList[n1]/(re
lAbndList[n]+relAbndList[n1]))
                            mzList[n1] = 0
                            mzList[n] = currentMZ
                            relAbndList[n] = relAbndList[n] + relAbndList[n1]
                            relAbndList[n1] = 0
                n1 = n1 + 1
            n = n + 1
            n1 = n + 1
        n = 0
        # If statement to include molecular formulas with elements with no isotopes with
[M+]
        if len(relAbndList) == 0:
                relAbndList.append(1)
        # If statement to include molecular formulas with elements with no isotopes with
[M+]
        if len(mzList) == 0:
                mzList.append(0.0000000001) #adding a very small value so the entry will be
picked up by the next if statement
        ## Loop to add the masses of the elements which do not have natural isotopes
        for item in mzList:
```

```
            if mzList[n] > 0:
                mzList[n] = mzList[n] + (numOfNa * 22.9897692809) + (numOfP * 30.97376163)
+ (numOfF * 18.99840322) + \
                            (numOfI * 126.904473) + (numOfAs * 74.9215965) + (numOfSc *
44.9559119) + (numOfAl * 26.98153863) + (numOfMn * 54.9380451) + \
                            (numOfCo * 58.9331950) + (numOfBe * 9.0121822) +
(int(removeEleC) * 13.0033548378) + (numOfAu * 196.9665687) + \
                            (numOfHe * 4.00260325415) + (int(removeEleN) * 15.0001088982) +
(int(extraRemoveEleH) * 2.0141017778) + (numOfY * 88.9058483) + \
                            (numOfNb * 92.9063781) + (numOfRh * 102.905504) + (numOfCs *
132.905451933) + (numOfBi * 208.9803987)  + (numOfTh * 232.0380553) + \
                            (numOfPa * 231.0358840) + (numOfAc * 227.0277521) + (numOfTm *
168.9342133) + (numOfHo * 164.9303221) + (numOfTb * 158.9253468) + \
                            (numOfPr * 140.9076528)
            n = n + 1
        n = 0
        mzList = filter(lambda a: a != 0, mzList)
        relAbndList = filter(lambda a: a != 0, relAbndList)
        ## Loop to remove the weight of an electron from each mass to give a positive
charge and to divide by the charge state
        for item in mzList:
            mzList[n] = mzList[n] - (electronChange * int(charge))
            mzList[n] = mzList[n] / int(charge)
            n = n + 1
        if errorBool == 1:
            del mzList[:]
            mzList.append(1)
            del relAbndList[:]
            relAbndList.append(100)
        return (mzList, relAbndList, errorBool, errorEle)
    def ionAdList(primary):
        stateOfCharge = primary.chargeState.get()
        detectorCharge = primary.chargeMode.get()
        if int(stateOfCharge) == 2:
            if detectorCharge == "Positive":
                primary.adductOptions.set("[M+2H]")
                primary.adductOptions["values"] = ionChoices2
            else:
                primary.adductOptions.set("[M-2H]")
                primary.adductOptions["values"] = ionChoices6
        elif int(stateOfCharge) == 3:
            if detectorCharge == "Positive":
                primary.adductOptions.set("[M+3H]")
                primary.adductOptions["values"] = ionChoices3
            else:
                primary.adductOptions.set("[M-3H]")
                primary.adductOptions["values"] = ionChoices7
        elif int(stateOfCharge) == 4:
            if detectorCharge == "Positive":
                primary.adductOptions.set("[M+4H]")
                primary.adductOptions["values"] = ionChoices4
            else:
                primary.adductOptions.set("[M-4H]")
                primary.adductOptions["values"] = ionChoices8
        elif int(stateOfCharge) == 0:
            primary.adductOptions.set("[M]")
            primary.adductOptions["values"] = ionChoices0
        else:
            if detectorCharge == "Positive":
                primary.adductOptions.set("[M+H]")
                primary.adductOptions["values"] = ionChoices1
            else:
                primary.adductOptions.set("[M-H]")
```

```
                primary.adductOptions["values"] = ionChoices5
        return
    def ionAdListA2(primary):
        stateOfCharge2 = primary.A2cState.get()
        detectorCharge = primary.chargeMode.get()
        if int(stateOfCharge2) == 2:
            if detectorCharge == "Positive":
                primary.adductOptionsA2.set("[M+2H]")
                primary.adductOptionsA2["values"] = ionChoices2
            else:
                primary.adductOptionsA2.set("[M-2H]")
                primary.adductOptionsA2["values"] = ionChoices6
        elif int(stateOfCharge2) == 3:
            if detectorCharge == "Positive":
                primary.adductOptionsA2.set("[M+3H]")
                primary.adductOptionsA2["values"] = ionChoices3
            else:
                primary.adductOptionsA2.set("[M-3H]")
                primary.adductOptionsA2["values"] = ionChoices7
        elif int(stateOfCharge2) == 4:
            if detectorCharge == "Positive":
                primary.adductOptionsA2.set("[M+4H]")
                primary.adductOptionsA2["values"] = ionChoices4
            else:
                primary.adductOptionsA2.set("[M-4H]")
                primary.adductOptionsA2["values"] = ionChoices8
        elif int(stateOfCharge2) == 0:
            primary.adductOptionsA2.set("[M]")
            primary.adductOptionsA2["values"] = ionChoices0
        else:
            if detectorCharge == "Positive":
                primary.adductOptionsA2.set("[M+H]")
                primary.adductOptionsA2["values"] = ionChoices1
            else:
                primary.adductOptionsA2.set("[M-H]")
                primary.adductOptionsA2["values"] = ionChoices5
        return
    def ionAdListA3(primary):
        stateOfCharge3 = primary.A3cState.get()
        detectorCharge = primary.chargeMode.get()
        if int(stateOfCharge3) == 2:
            if detectorCharge == "Positive":
                primary.adductOptionsA3.set("[M+2H]")
                primary.adductOptionsA3["values"] = ionChoices2
            else:
                primary.adductOptionsA3.set("[M-2H]")
                primary.adductOptionsA3["values"] = ionChoices6
        elif int(stateOfCharge3) == 3:
            if detectorCharge == "Positive":
                primary.adductOptionsA3.set("[M+3H]")
                primary.adductOptionsA3["values"] = ionChoices3
            else:
                primary.adductOptionsA3.set("[M-3H]")
                primary.adductOptionsA3["values"] = ionChoices7
        elif int(stateOfCharge3) == 4:
            if detectorCharge == "Positive":
                primary.adductOptionsA3.set("[M+4H]")
                primary.adductOptionsA3["values"] = ionChoices4
            else:
                primary.adductOptionsA3.set("[M-4H]")
                primary.adductOptionsA3["values"] = ionChoices8
        elif int(stateOfCharge3) == 0:
            primary.adductOptionsA3.set("[M]")
```

```
            primary.adductOptionsA3["values"] = ionChoices0
        else:
            if detectorCharge == "Positive":
                primary.adductOptionsA3.set("[M+H]")
                primary.adductOptionsA3["values"] = ionChoices1
            else:
                primary.adductOptionsA3.set("[M-H]")
                primary.adductOptionsA3["values"] = ionChoices5
        return
    def ionAdListA4(primary):
        stateOfCharge4 = primary.A4cState.get()
        detectorCharge = primary.chargeMode.get()
        if int(stateOfCharge4) == 2:
            if detectorCharge == "Positive":
                primary.adductOptionsA4.set("[M+2H]")
                primary.adductOptionsA4["values"] = ionChoices2
            else:
                primary.adductOptionsA4.set("[M-2H]")
                primary.adductOptionsA4["values"] = ionChoices6
        elif int(stateOfCharge4) == 3:
            if detectorCharge == "Positive":
                primary.adductOptionsA4.set("[M+3H]")
                primary.adductOptionsA4["values"] = ionChoices3
            else:
                primary.adductOptionsA4.set("[M-3H]")
                primary.adductOptionsA4["values"] = ionChoices7
        elif int(stateOfCharge4) == 4:
            if detectorCharge == "Positive":
                primary.adductOptionsA4.set("[M+4H]")
                primary.adductOptionsA4["values"] = ionChoices4
            else:
                primary.adductOptionsA4.set("[M-4H]")
                primary.adductOptionsA4["values"] = ionChoices8
        elif int(stateOfCharge4) == 0:
            primary.adductOptionsA4.set("[M]")
            primary.adductOptionsA4["values"] = ionChoices0
        else:
            if detectorCharge == "Positive":
                primary.adductOptionsA4.set("[M+H]")
                primary.adductOptionsA4["values"] = ionChoices1
            else:
                primary.adductOptionsA4.set("[M-H]")
                primary.adductOptionsA4["values"] = ionChoices5
        return
    def ionAdListCoCom(primary):
        coStateOfCharge = primary.coChargeState.get()
        detectorCharge = primary.chargeMode.get()
        if int(coStateOfCharge) == 2:
            if detectorCharge == "Positive":
                primary.coAdductOptions.set("[M+2H]")
                primary.coAdductOptions["values"] = ionChoices2
            else:
                primary.coAdductOptions.set("[M-2H]")
                primary.coAdductOptions["values"] = ionChoices6
        elif int(coStateOfCharge) == 3:
            if detectorCharge == "Positive":
                primary.coAdductOptions.set("[M+3H]")
                primary.coAdductOptions["values"] = ionChoices3
            else:
                primary.coAdductOptions.set("[M-3H]")
                primary.coAdductOptions["values"] = ionChoices7
        elif int(coStateOfCharge) == 4:
            if detectorCharge == "Positive":
```

```
                primary.coAdductOptions.set("[M+4H]")
                primary.coAdductOptions["values"] = ionChoices4
            else:
                primary.coAdductOptions.set("[M-4H]")
                primary.coAdductOptions["values"] = ionChoices8
        elif int(coStateOfCharge) == 0:
            primary.coAdductOptions.set("[M]")
            primary.coAdductOptions["values"] = ionChoices0
        else:
            if detectorCharge == "Positive":
                primary.coAdductOptions.set("[M+H]")
                primary.coAdductOptions["values"] = ionChoices1
            else:
                primary.coAdductOptions.set("[M-H]")
                primary.coAdductOptions["values"] = ionChoices5
        return
    def grph(primary, event):
        tempMassList = []
        isoMassList = []
        tempAbundanceList = []
        currentMass = 0
        adductCount = primary.numOfAdducts.get()
        resNum = primary.resolution.get()
        coCompoundCount = primary.coCompoundNum.get()
        labelMax = primary.labelCutOffPC.get()
        maxCounts = 0
        if labelMax != "NA":
            labelMax = float(labelMax)
            if labelMax < 0:
                labelMax = 0
            labelMax = labelMax/100
        intNum = 0
        n = 0
        testList = []
        while intNum < int(adductCount):
            if intNum == 0:
                massList, abundanceList, boolErr, eleSybError =
primary.massCalc(primary.userForm.get(), primary.chargeState.get(),
primary.carbonAbund.get(),

primary.adductIonStr.get(), primary.numOfC13.get(), primary.numOfN15.get(),

primary.numOfH2.get())
                maxCounts = max(abundanceList)*100
                for each in abundanceList:
                    abundanceList[n] = each / (maxCounts/100)
                    n = n + 1
            elif intNum == 1:
                massList, abundanceList, boolErr, eleSybError =
primary.massCalc(primary.userForm.get(), primary.A2cState.get(), primary.carbonAbund.get(),

primary.A2IonStr.get(), primary.numOfC13.get(), primary.numOfN15.get(),

primary.numOfH2.get())
                maxCounts = max(abundanceList)*100
                for each in abundanceList:
                    abundanceList[n] = each / (maxCounts/100)
                    n = n + 1
                abundanceList = [x*((float(primary.A2abnd.get()))/100) for x in
abundanceList]
            elif intNum == 2:
                massList, abundanceList, boolErr, eleSybError =
primary.massCalc(primary.userForm.get(), primary.A3cState.get(), primary.carbonAbund.get(),
```

```
primary.A3IonStr.get(), primary.numOfC13.get(), primary.numOfN15.get(),

primary.numOfH2.get())
                maxCounts = max(abundanceList)*100
                for each in abundanceList:
                    abundanceList[n] = each / (maxCounts/100)
                    n = n + 1
                abundanceList = [x*((float(primary.A3abnd.get()))/100) for x in
abundanceList]
            elif intNum == 3:
                massList, abundanceList, boolErr, eleSybError =
primary.massCalc(primary.userForm.get(), primary.A4cState.get(), primary.carbonAbund.get(),

primary.A4IonStr.get(), primary.numOfC13.get(), primary.numOfN15.get(),

primary.numOfH2.get())
                maxCounts = max(abundanceList)*100
                for each in abundanceList:
                    abundanceList[n] = each / (maxCounts/100)
                    n = n + 1
                abundanceList = [x*((float(primary.A4abnd.get()))/100) for x in
abundanceList]
            tempMassList = tempMassList + massList
            tempAbundanceList = tempAbundanceList + abundanceList
            intNum = intNum + 1
            n = 0
        if int(coCompoundCount) == 1:
            massList, abundanceList, boolErr, eleSybError =
primary.massCalc(primary.userCoForm.get(), primary.coChargeState.get(),
primary.coCarbonAbund.get(),

primary.coAdductIonStr.get(), primary.coNumOfC13.get(), primary.coNumOfN15.get(),

primary.coNumOfH2.get())
            maxCounts = max(abundanceList)*100
            for each in abundanceList:
                abundanceList[n] = each / (maxCounts/100)
                n = n + 1
            abundanceList = [x*((float(primary.coComAbnd.get()))/100) for x in
abundanceList]
            tempMassList = tempMassList + massList
            tempAbundanceList = tempAbundanceList + abundanceList
        intNum = 0
        abundanceList = tempAbundanceList
        massList = tempMassList
        n = 0
        n1 = 1
        massList, abundanceList = zip(*sorted(zip(massList, abundanceList)))
        massList, abundanceList = (list(t) for t in zip(*sorted(zip(massList,
abundanceList))))
        for item in massList:
            currentMass = massList[n]
            while n1 < len(massList):
                if massList[n1] > (currentMass - (currentMass/float(resNum))):
                    if massList[n1] < (currentMass + (currentMass/float(resNum))):
                        if currentMass != 0:
                            currentMass =
massList[n]*(abundanceList[n]/(abundanceList[n]+abundanceList[n1]))+massList[n1]*(abundance
List[n1]/(abundanceList[n]+abundanceList[n1]))
                            massList[n1] = 0
                            massList[n] = currentMass
                            abundanceList[n] = abundanceList[n] + abundanceList[n1]
```

```
                            abundanceList[n1] = 0
                n1 = n1 + 1
            n = n + 1
            n1 = n + 1
        massList = filter(lambda a: a != 0, massList)
        abundanceList = filter(lambda a: a != 0, abundanceList)
        intNum = 0
        counts = 1
        maxCounts = counts
        maxMass = 0
        minMass = 1000000000
        resVal = StringVar()
        resVal = primary.resolution.get()
        labelHeight = 0
        primary.ax2.clear()
        colorNum = 0.2
        colorChange = 0
        if boolErr == 1:
            primary.errorMessage.set("Error: Symbol " + eleSybError + " not available.")
        else:
            primary.errorMessage.set("")
        maxCounts = max(abundanceList)*100
        intNum = 0
        for each in massList:
            if massList[intNum] > maxMass:
                maxMass = massList[intNum]  + 0.5
            if massList[intNum] < minMass:
                minMass = massList[intNum]  - 0.5
            intNum = intNum + 1
        labelHeight = (maxCounts+100/4)*0.14
        intNum = len(massList)-1
        primary.txt.configure(state="normal")
        primary.txt.delete(1.0, END)
        massText = StringVar()
        abundText = StringVar()
        for each in massList:
            massText = "%.5f" % float(massList[intNum])
            abundText = "%.4f" % float((abundanceList[intNum])*100)
            primary.txt.insert('1.0', massText + ': ' + abundText + '\n')
            intNum = intNum - 1
        primary.txt.configure(state="disabled")
        intNum = 0
        colorsMap = [(0.976,0.333,0.518), (0.976,0.333,0.518), (0.976,0.333,0.518)]
        while intNum < len(abundanceList):
            x = np.linspace(massList[intNum] - (massList[intNum]/float(resVal)),
massList[intNum] + (massList[intNum]/float(resVal)), 200)
            y =
(abundanceList[intNum]*50)*np.sin((np.pi*(1/(massList[intNum]/float(resVal)*2)*2))*x-
((massList[intNum]-
((massList[intNum]/float(resVal)*0.5)))*(np.pi*(1/(massList[intNum]/float(resVal)*2)*2))))
+ (abundanceList[intNum]*50)
            primary.ax2.plot(x, y, color = (colorNum,1-colorNum,0.952))
            primary.ax2.plot([0,30000], [0,0], "b", linewidth=0.5)
            if labelMax != "NA":
                if abundanceList[intNum] > float(labelMax):
                    primary.ax2.text(massList[intNum],
(abundanceList[intNum]*100)+labelHeight, str(format(massList[intNum],'.4f')), fontsize=9,
rotation=80)
                if abundanceList[intNum] <= float(labelMax):
                    if (intNum > 3) and (intNum+4 < len(abundanceList)):
                        if (abundanceList[intNum] > abundanceList[intNum+1]) and
(abundanceList[intNum] > abundanceList[intNum+2]) \
                            and (abundanceList[intNum] > abundanceList[intNum+3]) and
```

```
(abundanceList[intNum] > abundanceList[intNum+4]):
                            if (abundanceList[intNum] > abundanceList[intNum-1]) and
(abundanceList[intNum] > abundanceList[intNum-2]) \
                                and (abundanceList[intNum] > abundanceList[intNum-3]) and
(abundanceList[intNum] > abundanceList[intNum-4]):
                                 primary.ax2.text(massList[intNum],
(abundanceList[intNum]*100)+labelHeight, str(format(massList[intNum],'.4f')), fontsize=9,
rotation=80)
                    elif (intNum == 3) and (intNum+1 < len(abundanceList)) and
(abundanceList[intNum] > abundanceList[intNum+1]) \
                          and (abundanceList[intNum] > abundanceList[intNum-1]) and
(abundanceList[intNum] > abundanceList[intNum-2]):
                         primary.ax2.text(massList[intNum],
(abundanceList[intNum]*100)+labelHeight, str(format(massList[intNum],'.4f')), fontsize=9,
rotation=80)
                    elif (intNum == 2) and (intNum+1 < len(abundanceList)) and
(abundanceList[intNum] > abundanceList[intNum+1]) \
                          and (abundanceList[intNum] > abundanceList[intNum-1]) and
(abundanceList[intNum] > abundanceList[intNum-2]):
                         primary.ax2.text(massList[intNum],
(abundanceList[intNum]*100)+labelHeight, str(format(massList[intNum],'.4f')), fontsize=9,
rotation=80)
                    elif (intNum == 1) and (intNum+1 < len(abundanceList)) and
(abundanceList[intNum] > abundanceList[intNum+1]) and (abundanceList[intNum] >
abundanceList[intNum-1]):
                         primary.ax2.text(massList[intNum],
(abundanceList[intNum]*100)+labelHeight, str(format(massList[intNum],'.4f')), fontsize=9,
rotation=80)
                    elif (intNum == 0) and (intNum+1 < len(abundanceList)) and
(abundanceList[intNum] > abundanceList[intNum+1]) and (abundanceList[intNum] >
abundanceList[intNum+2]):
                         primary.ax2.text(massList[intNum],
(abundanceList[intNum]*100)+labelHeight, str(format(massList[intNum],'.4f')), fontsize=9,
rotation=80)
            primary.ax2.set_xlabel('Mass to charge (m/z)')
            primary.ax2.set_title('Isotope Pattern')
            primary.ax2.set_ylabel('Relative Abundance')
            if colorChange == 0:
                colorNum = colorNum + 0.02
            if colorChange == 1:
                colorNum = colorNum - 0.02
            if colorNum > 1:
                colorNum = colorNum - 0.04
                colorChange = 1
            if colorNum < 0.2:
                colorNum = colorNum + 0.04
                colorChange = 0
            intNum = intNum + 1
        primary.ax2.set_ylim( ((-(100+100/4))/50,maxCounts+maxCounts/4) )
        primary.ax2.set_xlim( (minMass - ((maxMass-minMass)*0.1), maxMass + ((maxMass-
minMass)*0.1)) )
        primary.canvas.draw()
        return
    def AdductChargeRefresh(primary, *args):
        result = getattr(primary, 'ionAdList')()
        result = getattr(primary, 'ionAdListA2')()
        result = getattr(primary, 'ionAdListA3')()
        result = getattr(primary, 'ionAdListA4')()
        result = getattr(primary, 'ionAdListCoCom')()
        return
    def AdductRefresh(primary):
        numOfAds1 = primary.numOfAdducts.get()
        numOfMols = primary.coCompoundNum.get()
```

```
        if numOfAds1 == "1":
            primary.adductOptionsA2.configure(state="disabled")
            primary.adductOptionsA3.configure(state="disabled")
            primary.adductOptionsA4.configure(state="disabled")
            primary.A2AbundVal.configure(state="disabled")
            primary.A3AbundVal.configure(state="disabled")
            primary.A4AbundVal.configure(state="disabled")
            primary.A2chargeVal.configure(state="disabled")
            primary.A3chargeVal.configure(state="disabled")
            primary.A4chargeVal.configure(state="disabled")
        if numOfAds1 == "2":
            primary.adductOptionsA2.configure(state="enabled")
            primary.adductOptionsA3.configure(state="disabled")
            primary.adductOptionsA4.configure(state="disabled")
            primary.A2AbundVal.configure(state="normal")
            primary.A3AbundVal.configure(state="disabled")
            primary.A4AbundVal.configure(state="disabled")
            primary.A2chargeVal.configure(state="normal")
            primary.A3chargeVal.configure(state="disabled")
            primary.A4chargeVal.configure(state="disabled")
        if numOfAds1 == "3":
            primary.adductOptionsA2.configure(state="enabled")
            primary.adductOptionsA3.configure(state="enabled")
            primary.adductOptionsA4.configure(state="disabled")
            primary.A2AbundVal.configure(state="normal")
            primary.A3AbundVal.configure(state="normal")
            primary.A4AbundVal.configure(state="disabled")
            primary.A2chargeVal.configure(state="normal")
            primary.A3chargeVal.configure(state="normal")
            primary.A4chargeVal.configure(state="disabled")
        if numOfAds1 == "4":
            primary.adductOptionsA2.configure(state="enabled")
            primary.adductOptionsA3.configure(state="enabled")
            primary.adductOptionsA4.configure(state="enabled")
            primary.A2AbundVal.configure(state="normal")
            primary.A3AbundVal.configure(state="normal")
            primary.A4AbundVal.configure(state="normal")
            primary.A2chargeVal.configure(state="normal")
            primary.A3chargeVal.configure(state="normal")
            primary.A4chargeVal.configure(state="normal")
        if numOfMols != "0":
            primary.coAdductOptions.configure(state="normal")
            primary.coIonAbundVal.configure(state="normal")
            primary.coMZEntry.configure(state="normal")
            primary.coChargeVal.configure(state="normal")
            primary.coExtraC13Val.configure(state="normal")
            primary.coCarbonVal.configure(state="normal")
            primary.coExtraN15Val.configure(state="normal")
            primary.coExtraH2Val.configure(state="normal")
        else:
            primary.coAdductOptions.configure(state="disabled")
            primary.coIonAbundVal.configure(state="disabled")
            primary.coMZEntry.configure(state="disabled")
            primary.coChargeVal.configure(state="disabled")
            primary.coExtraC13Val.configure(state="disabled")
            primary.coCarbonVal.configure(state="disabled")
            primary.coExtraN15Val.configure(state="disabled")
            primary.coExtraH2Val.configure(state="disabled")
        return
if __name__ == "__main__":
    app = simpleapp_tk(None)
    app.title('IsoPat 1.09')
    img = PhotoImage(data=iconData)
```

```
    app.tk.call('wm', 'iconphoto', app._w, img)
    app.mainloop()
```

Code S-4: *Source code for IsoPat v1.09*

This programme simulates isotope patterns for given molecular formulas for specific ions, adducts, fragmentation ions, and coeluting compounds.

References

(1) Rasmussen, S. A.; Gedsted, S. M.; Andersen, N.; Blossom, H. E.; Duus, J. Ø.; Nielsen, K. F.; Hansen, P. J.; Larsen, T. O. *J. Nat. Prod.* **2016**

(2) Igarashi, T.; Satake, M.; Yasumoto, T. *J. Am. Chem. Soc.* **1996**, *118* (2), 479–480.

(3) Murata, M.; Yasumoto, T. *Nat. Prod. Rep.* **2000**, *17* (3), 293–314.

(4) Kind, T.; Fiehn, O. *BMC Bioinformatics* **2007**, *8*, 105.