Natzaret Gálvez Rísquez

Avaluació de diferents algorismes de detecció de pics de cromatografia de gasos amb senyals GC-MS simulades d'orina mitjançant R

Treball Fi de Grau

dirigit pel Dr. Jesús Brezmes

Grau en Enginyeria Biomèdica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2023

Agraïments

En primer lloc, donar-li la meva sincera gratitud al director del treball de fi de grau, Jesús, per haver-me donat l'oportunitat de fer aquest treball i aprendre d'aquest, juntament amb la seva ajuda i guia al llarg d'aquest projecte.

I agrair als meus pares i familiars que m'han recolzat en el transcurs del treball.

Resum

Avui en dia hi ha molts algoritmes diferents que analitzen les dades de diferents mètodes de mesura de cromatografia de gasos amb espectrometria de masses. A causa de la gran diversitat de solucions que podem emprar per a la seva anàlisi trobem que aquests estan subjectes a diverses variacions en els resultats que poden afectar el seu senyal de sortida o espectre. És per aquestes raons que s'ha creat un algoritme capaç de simular dades de sortida d'un GC/MS el qual es pot ajustar a les necessitats de l'usuari per cada cas. Per provar l'eficàcia d'aquesta simulació, s'ha sotmès a tres tipus de mètodes de detecció de pics, XCMS, GCalignR i un algoritme creat basat en els mètodes ja existents i comparat amb els resultats d'una mostra real. Es pot observar que amb XCMS i l'algoritme creat es poden detectar els pics del cromatograma i acabar visualitzant i manipulant les dades per trobar senyals específics de forma gairebé idèntica. Ambdós mètodes presenten gràfics en comú com el TIC i també presenten característics seus com la correspondència o els senyals dels ions dels metabòlits en quadres, lo qual valida la utilitat de l'eina.

Paraules clau: GC/MS, pics, soroll, *baseline drift*, cromatograma, m/z, temps de retenció.

Índex de continguts

Índex de taules

Índex d'imatges

Glossari

- Adducte: "complex que es forma quan una substància química s'uneix a una molècula biològica" [1]
- Dextrogir i levogir: propietat d'una substància per desviar la llum polaritzada [2]. Trobem que en cercar els metabòlits en *databases* metabolòmiques o d'espectres de m/z, com [3, 4 i 5], se'ns presenten els metabòlits amb aquesta variació.
- *Features*: pics cromatogràfics agrupats pels arxius.
- FP: *false positive*, error de classificació en què s'indica un resultat dins la classe positiva, però que no pertany a aquesta.
- Metabòlit: molècula emprada o produïda durant el metabolisme [6].
- m/z: relació massa-càrrega. Massa dels ions dels metabòlits relacionats amb la càrrega elèctrica d'aquests.
- PCA: *principal component analysis*, pot emprar-se tant per la reducció de dimensionalitat com per la visualització de dades [7].
- *Ridgeline* (línia de cresta): la seva longitud "pot reflectir l'estabilitat de l'ajust de la forma d'un pic candidat a l'onada a través d'una sèrie d'escales CWT consecutives" [8].
- rt: *retention time*, temps de retenció. Temps que passa un compost a la columna cromatogràfica.
- SNR: *signal to noise ratio*, nivell de senyal i soroll.
- TP: *true positive*, es classifica el resultat dins la classe que li correspon.
- *Volcano plot*: "és un diagrama de dispersió que mostra la significació estadística (*p-value*) en funció de la magnitud del canvi (*fold change*)" [9].

1 Introducció

1.1 Context

A mitjans del segle XX es va desenvolupar la tècnica d'anàlisi químic basada en la combinació de la cromatografia de gasos (GC) amb l'espectrometria de masses (MS). Gràcies a la utilitat d'aquesta tecnologia combinada la seva utilització va a realitzar-se de forma ubiqua i relativament rutinària. Amb aquesta unió de mètodes analítics es podia obtenir l'habilitat per separar mescles químiques amb la part cromatogràfica (en el cas que ens ocupa de gasos, "GC") i l'habilitat d'identificació dels components mitjançant el detector de Masses i el seu espectre de fragmentació mitjançant el detector("MS") donant pas a què avui s'empri com a mesura d'estudi de mostres per identificar i/o quantificar substàncies per anàlisis ràpides, com per exemple per l'ús d'anàlisi d'aigües, terres, compostos de síntesi, orines, puresa de determinats productes, detecció de químics... [10].

L'estructura d'aquesta aproximació consisteix en la injecció de la mostra a l'aparell on es transferirà a la columna analítica d'aquest. En cas que la mostra sigui líquida es vaporitzarà a un escalfador dins el GC (l'anomenat "injector"), en cas contrari s'injecta o bé l'espai de cap o una fibra SPME. A la columna es separaran els components en analits a causa de "les seves diferències en la partició en la fase mòbil gas portadora i la fase estacionaria líquida mantinguda dins la columna, o per gasos més volàtils, la seva adsorció per una fase estacionaria sòlida" [11]. Després "les molècules neutres eluiran a través d'una línia de transferència escalfada cap a l'espectròmetre de masses" [11]. On a l'analitzador de masses del MS les molècules seran ionitzades per ser separades en un patró de fragments iònics que es distingeixen en funció de la seva relació massa-càrrega (m/z) i la seva intensitat relativa. Darrer a aquest pas trobem el detector de ions, on s'amplificarà el senyal i serà registrat per produir el cromatograma i l'espectre de masses [11].

Les dades de sortida del GC/MS són tridimensionals a causa que la informació obtinguda és: el temps de retenció, normalment a l'eix x, que és el temps des de la injecció de la mostra fins al final del GC; la intensitat mesurada, normalment a l'eix z; i la m/z dels ions en el seu rang de m/zs en el temps adquirit, normalment a l'eix y [11].

A l'analitzador el hardware requereix un software per poder llegir la informació de sortida del MS i posteriorment visualitzar el cromatograma o manipular aquelles dades per poder veure els compostos i la seva abundància. Abans, els resultats del MS eren impresos, lo qual portava a errors analítics com en el càlcul de les intensitats o les àrees [12], i avui en dia aquests errors són minimitzats en ser manipulats directament per un algoritme, però alhora trobem que es continuen obtenint dades cromatogràfiques que poden tenir errors a causa del soroll o l'error de línia de base ("*baseline drift")*, entre d'altres, on la gran majoria d'algorismes d'anàlisis ja incorporen un treball de flux per poder minimitzar els efectes que puguin alterar els resultats i alhora refinar les dades.

A causa de la varietat de aparells i configuracions GC/MS al mercat, s'ha de tindre especial cura pels seus components a l'hora de manipular les dades, ja que normalment els efectes que alteren els resultats no solen provocar gran diferència a causa de la qualitat dels aparells, però al mateix temps podem trobar variabilitats dependents de la quantitat de mostra, la pressió d'injecció, flux de gas transportador, els components emprats, etc.

Així doncs, es busca amb aquest projecte la incorporació d'un algorisme capaç d'adaptar-se a les circumstàncies de la maquinària emprada per poder generar dades simulades i poder estudiar-les junt amb dades reals i comparar-les conjuntament amb els diferents mètodes d'anàlisi de dades cromatogràfiques.

1.2 Objectius

Els principals objectius d'aquest treball de fi de grau són:

1. Fer una cerca bibliogràfica per veure quins factors cal tenir en compte per una simulació "in-silico", de forma que els cromatogrames sintètics s'assemblin el més possible als reals
2. Programar un paquet d'R per tal de generar, artificialment, cromatogrames amb soroll, interferències, imperfeccions i altres dificultats junt amb pics de diferents formes i amplituds per servir de base per l'avaluació posterior de diferents algorismes de detecció de pics
3. Fer una cerca bibliogràfica dels diferents mètodes proposats per la detecció de pics de cromatografia
4. Implementar en R algun algorisme no existent i fer servir els que ja existeixen per detecció de pics cromatogràfics en un conjunt de cromatogrames artificials per determinar l'eficàcia de la simulació i de les diferents solucions d'anàlisi de dades

1.3 Creació de l'algorisme per la generació de cromatogrames "in-silico"

Dins d'aquest projecte crearem des de zero un codi capaç de recrear les dades de sortida d'un GC/MS, elaborant així cromatogrames simulats ("in-silico"). La construcció d'aquest algorisme constarà dels següents passos:

- Creació de soroll i de deriva de línia base generat aleatòriament per una determinada potència d'aquests valors
- Lectura de dades GC/MS reals per poder extraure el temps de retenció (rt) de les m/z de les mostres
- Creació d'una llista de metabòlits de referència amb totes les propietats característiques d'aquests. El valor del rt per cada m/z extret anteriorment és donat als metabòlits corresponents de la llista
- Creació de matrius amb informació modificada en intensitat dels pics dels ions i/o en nombre de metabòlits
- Creació pertinent de funcions específiques per la creació de les dades cromatogràfiques, com: "mzNominal", "createMzAndIntensityLargeMatrix" i "treatedMetabolite()", entre d'altres

Al llarg d'aquest procés també s'ha creat una funció específica per poder obtenir diferents formes d'ona pels senyals donats.

1.4 Estructura de la memòria

Una vegada hem vist el context i els objectius, el treball s'orientarà d'acord amb l'acompliment dels objectius plantejats. Primerament, veurem quina informació es fonamental obtenir d'una cromatografia de gasos/espectrometria de masses, per seguidament analitzar el pre-processament i processament de les dades per la detecció de pics de cromatografia i mètodes emprats. Posteriorment, desenvoluparem un algoritme per crear artificialment unes dades cromatogràfiques controlades afegint les variacions característiques a les ones com el soroll o la deriva de la línia base (*baseline drift)*. Finalment, avaluarem l'eficàcia de diferents algorismes per la detecció de pics a partir dels cromatogrames reals i artificials.

1.5 Tecnologies emprades

1.5.1 R

R és un llenguatge de programació i un entorn de desenvolupament de programari per informàtica estadística i gràfics [13].

Aquesta plataforma proporciona una gran flexibilitat per l'anàlisi de dades a banda dels paquets ("*packages*") que es poden fer ús i es troben pujats a la xarxa sobre diferents tipus d'algorismes tractant diversos camps, com bé és la lectura de dades o la manipulació i l'anàlisi d'aquestes, com per exemple amb mètodes estadístics. Existeixen força paquets per la lectura i manipulació de dades de sortida de mesures fetes amb GC/MS.

Emprarem al llarg de tot el treball de fi de grau RStudio, que és un entorn pel desenvolupament de R ("IDE") i es troba disponible per Windows, MAC i Linux.

1.6 Fases del projecte

Les diferents parts en què es dividirà el treball de fi de grau són:

- Obtenció de l'algoritme capaç de recrear les dades artificials ("in silico") de sortida GC/MS amb format matricial
- Creació de diferents funcions capaces de visualitzar els cromatogrames de diferents formes
- Lectura i manipulació de dades reals de sortida del GC/MS per usar-les com a model per acabar d'ajustar la simulació del cromatograma
- Ús de diferents fluxos de treball per l'obtenció de cromatogrames per la detecció de pics
- Crear un algorisme propi de detecció de pics
- Comparar els resultats amb diferents algorismes (com XCMS) amb mesures reals i simulades
- Elaboració de la memòria de Treball de Fi de Grau

2 Cromatografia de gasos/espectrometria de masses (GC/MS)

La GC/MS és una tècnica analítica que consisteix en la separació de compostos volàtils i semi volàtils [14], on es combina la capacitat de separació de les tècniques cromatogràfiques de gasos amb la capacitat de sensibilitat i selecció del detector de masses [15]. A causa de les seves característiques, separa els compostos per la seva relació massa-càrrega (m/z) amb una durada d'un cicle de 30 minuts a 1 hora aproximadament. Les dades reals proporcionades per fer aquest treball proporcionen la informació dels alquens, blancs, compostos i estàndards de qualitat ("std") obtinguts per l'espectrometria de gasos amb una durada de 2 a 60 minuts. Aquesta informació és desada en format .mzML que pot ser llegit per la llibreria XCMS a R.

Més específicament, amb la informació de sortida de l'espectròmetre, trobarem que cada m/z conté la informació de la intensitat assolida en el temps de retenció en què va sorgir aquest pic. Gràcies a això es pot representar un gràfic per veure l'ona d'aquella m/z en el temps (cada m/z, doncs, es podria considerar "un canal de detecció diferent"), on podem generar un gràfic on es mostraran tots els espectres de cada grup llegit, on en cada ona de cada grup representarà la suma de totes les ones de les mostres d'aquell grup. De fet, la suma en cada instant de la senyal de tots els canals es el que s'anomena TIC ("Total Ion Chromatogram").

El resultant individual tant de les mostres com el grupal es veuran subjectes a una sèrie de factors que alteraran la corba original. A més a més, també tenim que les corbes poden variar d'acord amb els compostos recollits en la mostra original.

A continuació farem un recorregut per les variacions a les quals estan sotmeses aquestes ones.

2.1 Forma de pics cromatogràfics

2.1.1 Components en l'estructura de l'aparell emprats per obtenir els resultats

Tenim que a causa d'emprar uns aparells o altres, els components poden afectar a la sortida. Un estudi ho exemplifica mitjançant l'anàlisi en les formes de pic variada per la modulació de flux polsat per cromatografia de gasos bidimensional (PFM)-GCxGC. Més concretament, queda exposat que un *ratio* de flux baix pot portar a esbiaixar els pics i, al contrari, si s'incrementa aquest flux es reduirà aquest fenomen [16]. Tenim per esbiaixar a la resultant de la distorsió dels valors dels senyals, així com la modificació de l'amplada i/o de la forma del pic de l'ona o fins i tot, podem trobar superposicions o apropaments dels senyals en l'espectre. Per altra banda, també trobem mètodes com la modulació termal o la basada en vàlvules que requereix uns components específics, on a causa de les característiques operacionals necessàries per aquests el flux pot ser diferent per cada detector on es pot variar la concentració d'analits en el temps donant lloc a una cua més o menys pronunciada del senyal resultant [17].

2.1.2 Propietats de les mostres

> Els perfils d'elució resulten de la interacció entre diversos fenòmens. La difusió axial, la resistència a la transferència de massa en la fase mòbil i en la fase estacionària i la irregularitat del patró de flux són responsables de l'augment de l'amplada del pic amb l'augment del temps de retenció. [P. Moretti, et al., 2004, p.171, 18]

2.1.3 Forma de la corba

Trobem que normalment ni són corbes gaussianes ni simètriques a causa de les interaccions a la columna, sinó asimètriques, amb una corba més pronunciada fins a arribar al màxim d'intensitat llegida i més relaxada a partir d'aquest, depenent del solvent i la quantitat de solut [18]. En S. Vezzani esmenta que en condicions sense variació de temperatura però sí de pressió d'entrada per components no-polars i hidrocarburs, l'àrea de pic i la seva simetria serà a causa de la partició i la seva "cua a causa del mecanisme d'adsorció-desorció" [S. Vezzani, et al., 2004, p. 233, 19]. També podem trobar que l'asimetria pot ser a causa de solapament de pics o contaminació a banda de les condicions de la columna o el solut mateix [20].

2.1.4 Temperatura, pressió i quantitat de substància injectada

En augmentar la temperatura, la pressió d'entrada a la columna i/o el flux del gas portador trobem que la cua del senyal disminueix, el soroll és baix i els valors residuals es mantenen petits pel cas de la partició no linear [18].

En un altre estudi, variant el nombre d'àtoms de carboni de compostos a la mateixa temperatura però a pressió diferent, es va observar que els compostos més lleugers "es van eluir a la cua del pic de dissolvent, i els pics dels més pesats eren massa grans i plans i afectats pel soroll de la línia de base" [S. Vezzani, et al., 2004, p. 233, 19].

Així mateix, trobem que la *baseline* disminuirà amb l'increment de pressió d'entrada i la intensitat dels pics es veurà reduïda a major increment de substància injectada en columnes no polars pel cas d'una partició no linear isotèrmica [18] i pel cas de partició linear isotèrmica trobem que a major quantitat injectada l'àrea d'adsorció disminueix [19].

2.1.5 Soroll i *baseline drift*

Normalment sempre trobarem soroll en les dades de sortida a causa de les fluctuacions de mesura i pel propi soroll tèrmic de l'aparell de detecció. Avui en dia, sols ser petit per la qualitat dels espectròmetres de masses i a banda, s'empren mètodes com el límit de detecció d'instruments (IDL) i el límit de detecció del mètode (MDL) per minimitzar l'error, així com l'ús de la mitja i la desviació estàndard extreta [21]. La *baseline drift* característica que podem trobar a les cromatografies de gasos és a causa de molts factors, entre ells la contaminació del sistema de gasos, canvi en els paràmetres de temperatura i que no s'ha calibrat la columna, i alguns dels factors abans mencionats entre d'altres [22].

2.1.6 Interferències, imperfeccions i errors

Podem trobar que instruments a prop de la maquinària poden aportar interferències i quedar exemplificats com a un *offset* a la *baseline* a l'hora de representar les corbes. També trobem altres imperfeccions com pics o polsos a causa de senyals elèctrics externs. A més, hi ha uns altres factors com el *wander* a causa de la contaminació a l'injector, columna o el gas portador, i es presenta com una *baseline* que pot variar en alçada [22]. Finalment, sempre podrem trobar diferents tipus d'errors, ja sigui de mesura, contaminació, calibració, etc., on podem indagar més en com acaba afectant més els errors en els resultats de l'espectrometria, tal i com s'explica a l'estudi d'Edwards Alexis N. et al. (2021) on es parla de la propagació d'error pel cas d'espectrometria de masses [23].

Com bé es troba desglossat anteriorment, tots poden ser causes de l'alteració de la senyal final, a més a més, podem observar que hi ha moltes variables a tindre en compte per cada solut o mecanisme a emprar. On en cas que es vulgui fer una indagació més profunda en les interaccions del solut i/o propietats i mecanismes usats, es troben explicats amb profunditat en els papers [16-20].

3 Paquet d'R per la generació "in-silico" de cromatogrames GC/MS

Dissenyarem un algoritme capaç de generar artificialment la informació d'un cromatograma obtingut per GC/MS per després exportar-lo en un paquet. És lo que es sol anomenar "dades in silico".

Primerament, crearem una matriu amb el soroll i la *baseline drift*, aquests dos es generaran aleatòriament (veure el codi a l'apartat d'annexa).

Posteriorment, procedim a poder visualitzar les "intensitats" obtingudes al llarg del temps de retenció per cada m/z per una matriu sense informació de metabòlits en 2D. I també procedim a recrear la visualització 3D.

Totes les m/z i temps de retenció estaran en format nominal (a on cada m/z és un número sencer i , per tant, "un canal" de informació). En cas contrari, ho manipularem perquè es trobin en aquesta estructura.

Ara procedirem a la creació d'una llista de metabòlits. Per això, a causa de totes les variàncies possibles pel que fa al temps de retenció de cada ió de cada metabòlit degut a les seves característiques, (com si són dextrogirs, levogirs[1], abundància, un compost amb aquell metabòlit, etc.) hem decidit agafar el temps de retenció d'aquelles masses-càrrega que trobem a les dades d'un GC/MS real. Les relacions massa-càrrega (m/z) i les intensitats les agafarem d'Envipat [24] d'acord amb la seva fórmula química, on seran exportades en format Excel.

Envipat és una "calculadora d'isòtops de massa exacta, perfil, centroide o intensoide" per espectrometria de masses [24]. Podem extreure el perfil d'un metabòlit a partir de la seva fórmula química, així podent observar les intensitats per cada m/z. Es pot ajustar la resolució i els adductes.

[1] Descripció al glossari.

Aquestes dades seran subjectes a les següents funcions:

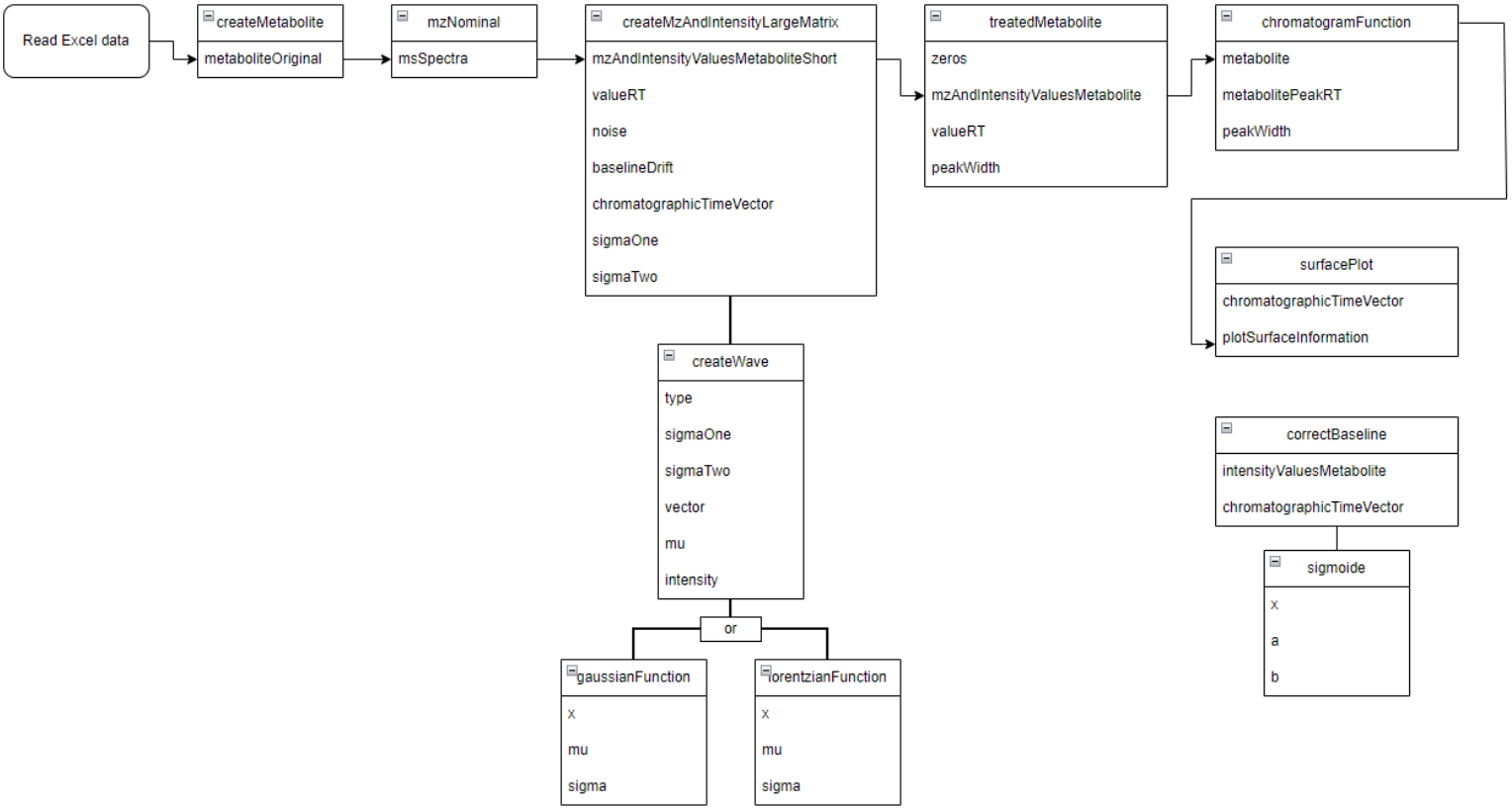

Figura 1. Diagrama de les funcions a emprar per al flux de treball per la creació del cromatograma simulat.

La creació de la llista mencionada que contindrà la informació de metabòlits seguirà el flux del diagrama de la figura 1, on trobem que a la tercera funció ("createMzAndIntensityLargeMatrix()") li passarem dades buides de soroll i *baseline drift*, ja que el que voldrem serà fer una llista dels metabòlits i les seves característiques per poder cercar-los posteriorment als cromatogrames. Els resultats donats per aquesta funció per un metabòlit es repetirà per tots els volguts i aquesta sortida s'organitzarà per cada analit amb la seva informació dins una llista, així seria com fer una llibreria de dades amb els seus atributs.

Una vegada ja tenim la llista, ara voldrem crear la matriu amb soroll, *baseline drift*, i tota la informació dels senyals dels metabòlits. Per la qual cosa agafarem la sortida de la funció "createMzAndIntensityLargeMatrix()" desada per cada metabòlit a la llista i els afegirem a la matriu creada en primera instància amb el soroll i la *baseline drift*.

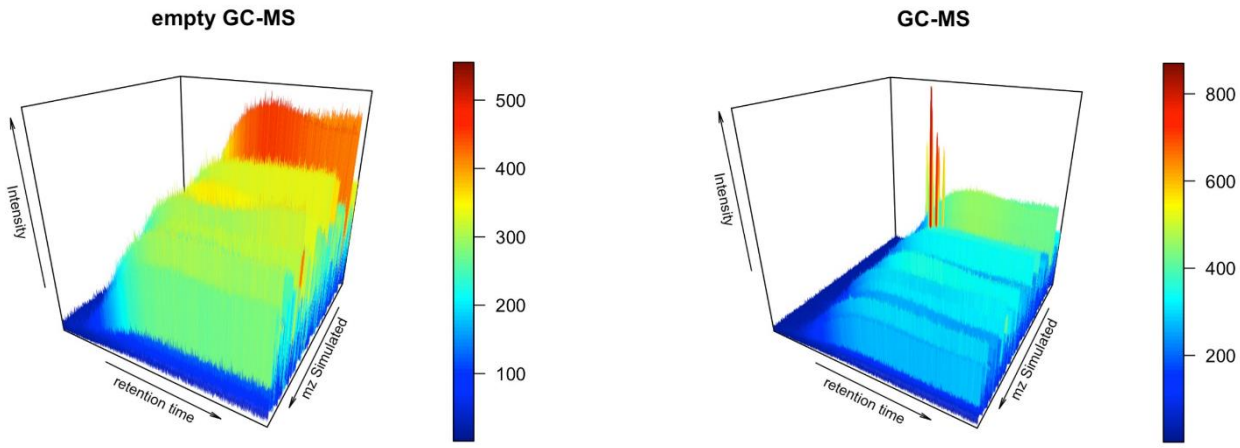

Figura 2. Cromatograma amb soroll i *baseline drift* sense i amb ions de metabòlits.

Podem observar l'entrada dels ions obtinguts per cada metabòlit al llarg del cromatograma simulat.

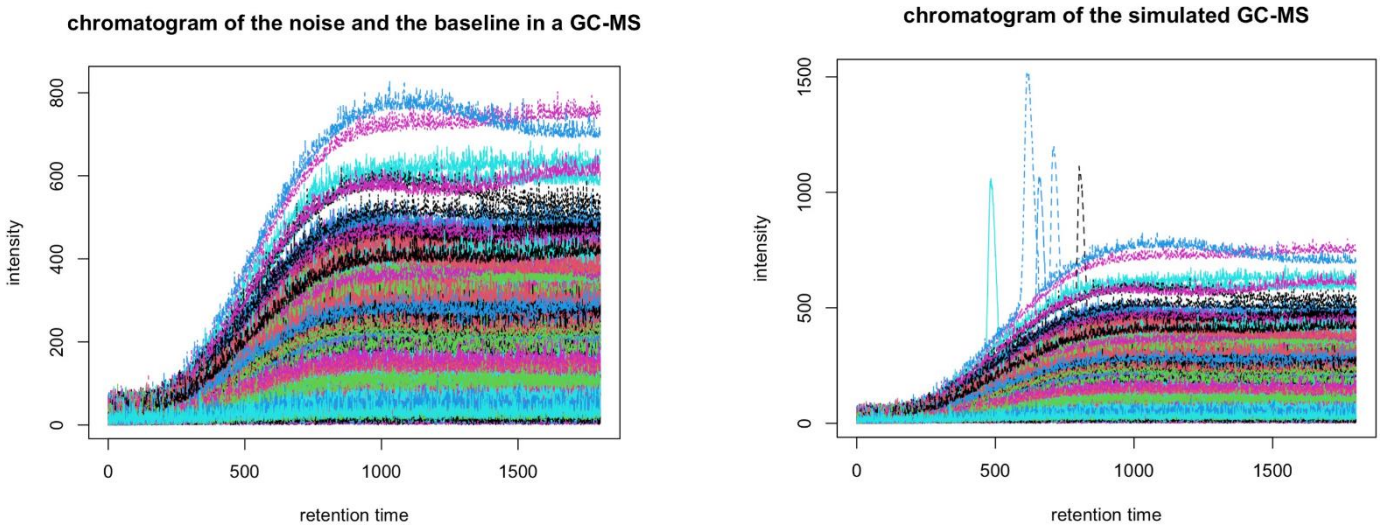

Figura 3. "Total Ion Chromatogram" amb soroll i *baseline drift* sense i amb ions de metabòlits. És representat per color cada m/z al llarg del temps de retenció (s).

Hem afegit un total de cinc metabòlits, els quals ja es trobaven ionitzats, es pot apreciar que sols un ió de cada metabòlit presenta un pic màxim i els altres es troben perduts en la informació del soroll i la deriva de línia base (*baseline drift)*. A més a més, hem volgut representar un cromatograma on es puguin apreciar la *baseline* i el soroll, també, les ones no presenten simetria per poder simular la sortida obtinguda a causa de diferents efectes de la maquinària emprada per obtenir la informació com bé es troba esmentat a l'apartat 2. La forma d'ona pot ser canviada entre simètrica i asimètrica de les següents funcions: gaussiana,

lorentziana i gaussiana amb lorentziana. Així, amb la variació en l'ona, l'aleatorietat en el soroll i la *baseline drift* simulem el cromatograma amb imperfeccions i pertorbacions pel flux de detecció de pics.

Ens basem en la informació d'anàlisis d'orina de Biosfer Teslab sobre la presència de metabòlits a l'orina [25] per cercar les fórmules químiques dels metabòlits presents a una mostra d'aquest tipus.

Creem també dues matrius més per poder fer una llista de matrius amb informació amb metabòlits. Seran diferents entre elles, tenint alguns metabòlits en comú.

Les funcions fetes servir són les següents:

- createMetabolite: aquesta funció s'encarregarà de crear un *dataframe* amb les m/z i les seves intensitats de l'Excel extret d'Envipat [24].
- mzNominal: s'encarregarà de passar les m/z amb decimals al format nominal.
- gaussianFunction: s'encarregarà de retornar una funció gaussiana.
- lorentzianFunction: retornarà una funció lorentziana.
- createWave: crearà una ona, té diferents opcions de creació, aquestes són: gaussiana simètrica i/o asimètrica, lorentziana simètrica i asimètrica i gaussiana combinada amb lorentziana simètrica i asimètrica. A la funció es demanen sigmes, les quals corresponen a l'amplada de les funcions a emprar, per exemple, pel cas de les ones no simètriques s'utilitzaran les dues sigmes, la primera per definir l'amplada del costat esquerre i, la segona, del dret. En cas d'emprar ones simètriques es farà servir la primera sigma i es passarà per alt la segona. Mu farà referència al centre del pic de l'ona.
- createMzAndIntensityLargeMatrix: omplim el *dataframe* que conté sols les m/z i les intensitats llegides amb els vectors de soroll i de *baseline*, per així tindre l'espectre al llarg del temps de retenció, a més a més, es donen als senyals forma de corba amb createWave().
- treatedMetabolite: creem una llista per a cada ió del metabòlit amb la informació de les intensitats i la seva corba al llarg del temps.
- chromatogramFunction: creem una llista que contingui diversa informació relacionada amb l'espectre dels ions del metabòlit. En aquesta llista trobarem: la informació de sortida de createMzAndIntensityLargeMatrix(); un *dataframe* on es desi en arxius les intensitats, les m/z i el temps de retenció del cromatograma, aquest format és específic per la representació de dades amb geom_desntiy_ridges(); un altre *dataframe* amb les dades especificades a una finestra en el temps per veure el cromatograma en 2D; una llista que conté tres tipus de gràfics diferents, aquests són: el primer "intensitiesMzs" per visualitzar les intensitats de cada ió en el temps per separat en el mateix gràfic en fila; "intensitiesStacked" on visualitzarem les intensitats al mateix gràfic en una finestra de temps; "plotMzsSeparately" per veure els senyals amb la finestra de temps en un quadre per files i columnes; en últim lloc, a la llista de sortida de la funció tenim la informació necessària per poder visualitzar les dades en 3D, on per poder visualitzar-les necessitarem trucar una altra funció, "surfacePlot". A més a més, aquesta funció també negligeix el senyal fora de la finestra d'amplada en el temps de retenció dels ions.
- sigmoide: funció per crear una corba sigmoidal.
- correctBaseline: funció per corregir la *baseline drift* d'un vector.
- surfacePlot: funció per poder visualitzar les dades en 3D del cromatograma creat.

En cas que es volgués indagar més en el significat de cada atribut de les funcions emprades, podem referir-nos al codi de l'apartat annexa.

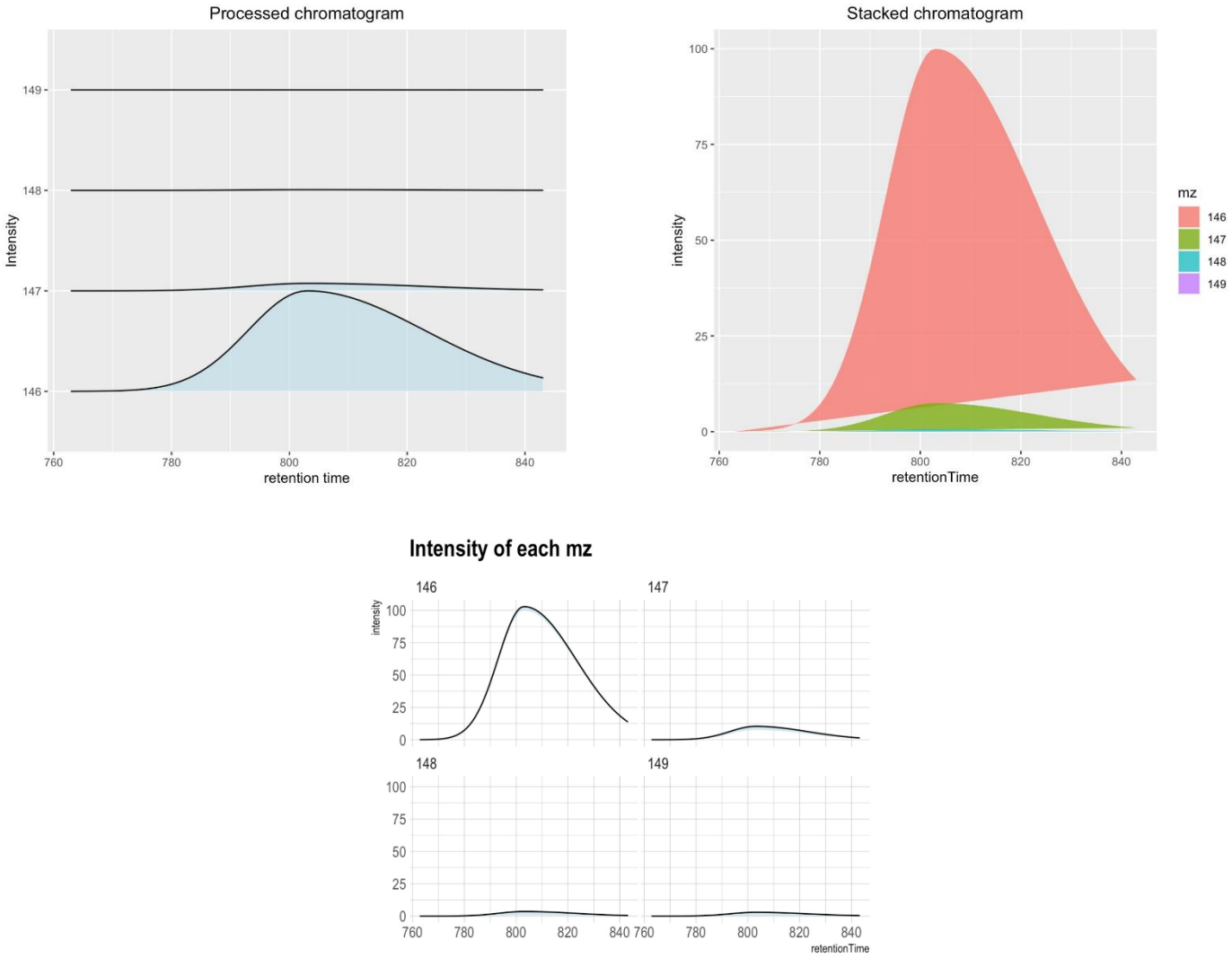

Figura 4. Gràfics desats en la sortida de la funció "chromatogramFunction()" com a una llista, els gràfics són "intensitiesMzs", "intensitiesStacked" i "plotMzsSeparately", respectivament.

Per aquests tres tipus de gràfics trobem la representació dels senyals obtinguts per la cerca en una finestra de m/z i temps de retenció en segons. Representem les dades de tres maneres diferents per poder apreciar millor les dades aconseguides i també fer la seva comparació. Pel primer cas, podem comparar els senyals assolits al llarg del temps, on han sigut representats en paral·lel per poder apreciar el senyal de cada ió ("canal" m/z) sense perdre informació i així també observar independentment el seu recorregut. Al segon, ho comparem amb una superposició de tots els senyals, veient així millor la diferència entre les intensitats i les seves àrees. Finalment, al tercer, els representem separats, ja que en ser gràfics independents entre ells, podem apreciar millor les intensitats d'aquells pics amb menor intensitat. Els nombres mostrats en l'eix vertical del primer gràfic fan referència a la relació massa-càrrega (m/z) igual que els valors a sobre cada representació del tercer gràfic.

Hem donat a les corbes una forma asimètrica gaussiana per a la base de dades per les variàncies que generalment podem trobar als resultats dels cromatogrames per GC/MS esmentades a l'apartat 2. Es per això que podem veure que s'ha generat "una petita cua" al pic asimètric.

També podem observar que aquest metabòlit se separa en quatre ions de m/z nominals: 146, 147, 148, 149, on alhora tenim que en fer la funció "mzNominal()" en cas que hi hagi m/z que siguin les mateixes s'arrodoniran a un nombre sencer i les intensitats d'aquestes seran sumades, per la qual cosa sols podem veure un rang de m/z igual a la finestra cercada.

A l'últim gràfic podem apreciar millor que totes quatre m/z presenten senyal, observant quatre ones en total, però que pel cas de les m/z 148 i 149 el pic d'intensitat és molt més petit en comparació amb la m/z 146. Però gràcies a aquest gràfic es pot apreciar millor el senyal per ones amb pics menys pronunciats, ja que les quatre ones estan representades independentment de les altres, fent que s'ajusti l'eix per cada ona. D'altra banda, tenim que en afegir soroll i la deriva de línia base les ones amb senyals més baixos es veuran encoberts a l'hora de visualitzar el cromatograma per aquestes variacions.

4 Processat per la detecció de pics de cromatografia

4.1 Pre-processament

Encapçalarem en aquest apartat la lectura de dades de sortida del GC/MS fins a la preparació de les funcions prèvies per a la detecció de pics de cromatografia.

Normalment trobem que el procediment a seguir és:

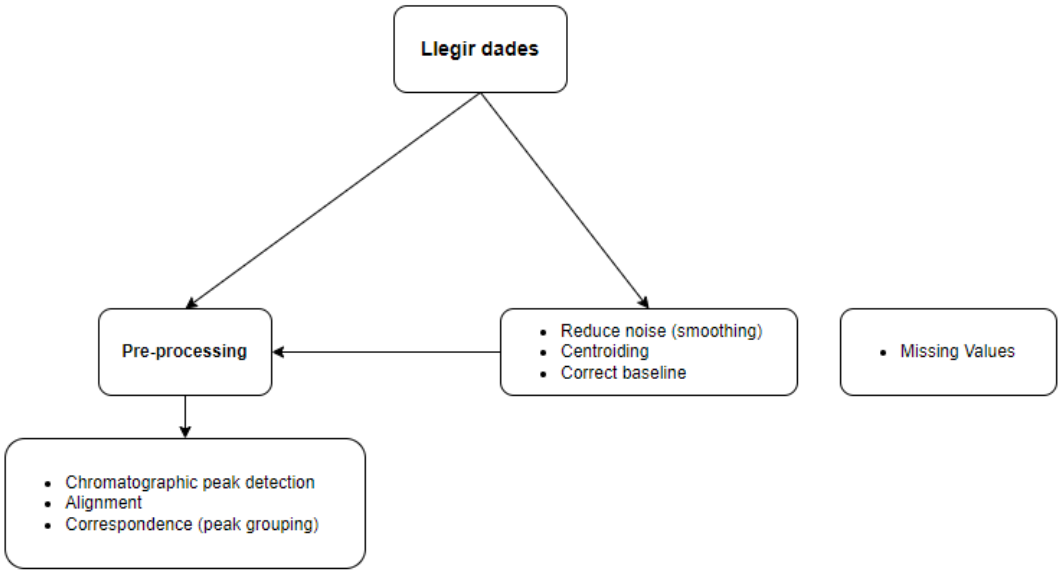

Figura 5. Esquema del pre-processament de dades [26-32].

Primerament, necessitarem llegir el format amb què l'espectròmetre desa les dades. Les més usuals solen ser els formats mzXML, mzData i mzML, els quals tots tres poden ser llegits amb el paquet d'R XCMS. També hem de veure com està la informació classificada, per si les mostres estan classificades en arxius amb el seu grup, o no.

Seguidament, trobem que alguns algoritmes van al pas de pre-processat i d'altres a l'altre quadre, això és a causa que al segon quadre ja es troben implementades les funcions del pre-processat, però d'altres han de fer aquest pas per separat. Així i tot, la majoria d'algoritmes implementa la majoria de les funcions mostrades, però no es té perquè emprar-les totes.

Finalment, tenim el tractament dels valors incomplerts ("*missing vàlues")*, que es sols fer al processament una vegada ja s'ha cercat la m/z i el temps de retenció desitjat. Perquè amb el tipus de dades que tenim es sols manipular molta informació i d'aquesta manera es poden obtenir els mateixos resultats en la cerca de pics cromatogràfics estalviant memòria i temps si es fa al final del pre-processat o al processament a unes dades específiques.

Les funcions emprades són les següents:

- *Reduce noise* (*smoothing*): reduir el soroll de cada espectre emprant un filtre, com pot ser el filtratge Savitzky-Golay [26].
- *Centroiding*: reduir diverses dades relacionades a un sols punt per a un ió per cada espectre. Trobant així el "centre de massa" de cada pic [26, 33].
- *Correct baseline*: descartar o corregir els valors que produeixin *baseline* o deriva de línia base.
- *Chromatographic peak detection*: trobar la regió dels pics al llarg del temps de retenció que representaran el senyal dels ions de compostos. Trobem que en aquesta mateixa funció es pot realitzar la reducció de soroll i de *baseline*. També hi ha diferents mètodes per la detecció de pics com distribucions probabilístiques o diverses transformades de Wavelet [26, 28, 32] entre d'altres.
- *Alignment*: la cromatografia es pot veure afectada per variàncies que poden portar a terme canvis en el temps de retenció entre mostres del mateix experiment [26], aquesta funció ajuda a ajustar aquests temps.
- *Correspondence* (*peak grouping*): agrupar pics cromatogràfics per les diferents mostres, que generalment seran pel mateix ió [26].
- *Missing values*: primerament determinar on són els valors perduts per després omplir els buits amb algun criteri determinat.

4.2 Processament

Una vegada ja s'han emprat les funcions mencionades a l'anterior apartat ja podem observar la resta de funcions per finalment visualitzar els pics o informació relacionada.

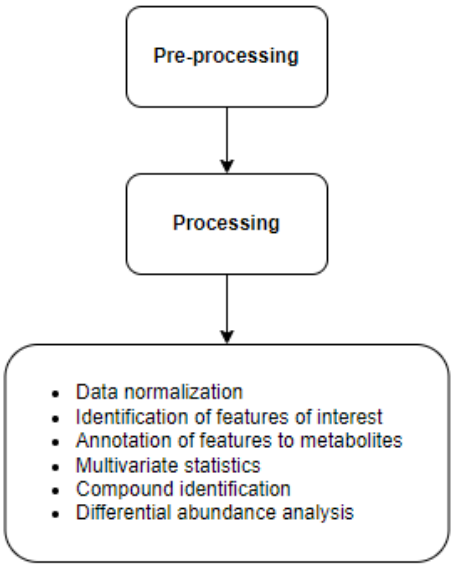

Figura 6. Esquema del processament de dades [26-32].

Per seguir explicant, introduïm dos conceptes importants:

- *Feature*: pics cromatogràfics agrupats per arxius [26].
- *Multivariate statistics*: es poden fer diferents anàlisis com PCA (*principal component analysis*), *volcano plot*, dendrograma...

Una vegada ja hem tractat les dades podem procedir a buscar informació més concretament allà on volem. Podem realitzar la detecció de pics per una m/z i temps de retenció determinat o obtenir els *features* de les dades per poder analitzar els pics obtinguts i visualitzar-los. Ho podem fer tot més orientat en organitzar les dades en *features* per poder veure els ions i metabòlits que representen o anar directament a les dades que volem per veure els pics que coincideixin amb els paràmetres de cerca que hàgim donat, com per exemple la m/z o el rt.

L'idea tant del pre-processat com del processament és organitzar les dades amb aquest fi, poder analitzar diversos aspectes com bé podem veure a la figura 6. Alhora, una vegada fet el pre-processament podem visualitzar diversos cromatogrames o portar a cap diversos passos del processament i visualitzar diferents pics cromatogràfics relacionats.

Més centrats en els mètodes de detecció de pics trobem que es poden emprar tècniques estadístiques com fluxos de treball per reduir el soroll i *baseline* juntament amb transformades de Wavelet.

A continuació observem diferents fluxos de treball per la detecció i anàlisis de dades per obtenir els pics i la seva informació.

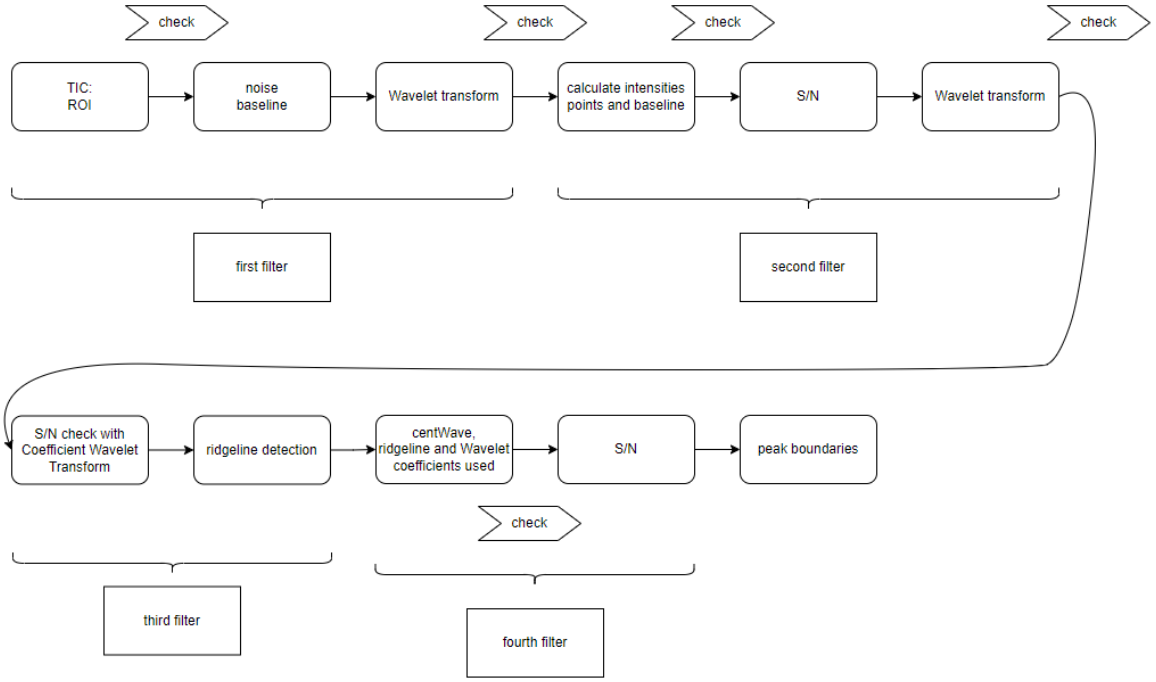

Figura 7. Esquema dels mètodes emprats per XCMS per la detecció de pics [30].

En graficar "*check*" entre quadres fem referència al pas de comprovació per les dades mitjançant els mètodes del quadre anterior, i així poder passar al següent quadre i continuar el flux.

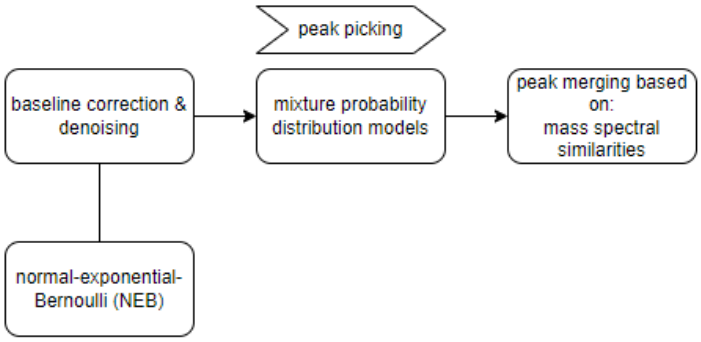

Figura 8. Esquema dels mètodes emprats per un algorisme de detecció de pics per a l'anàlisi de dades integrals de cromatografia de gasos d'espectrometria de masses [29].

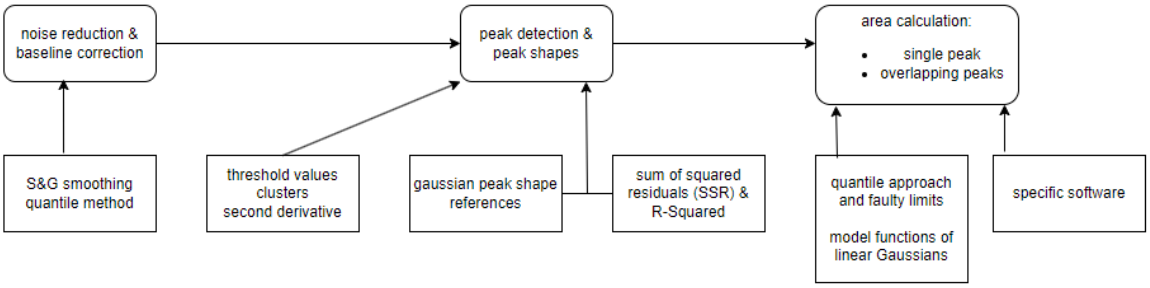

Figura 9. Esquema dels mètodes emprats per un algorisme totalment automàtic per a l'anàlisi de dades de cromatografia de gasos [32].

Assenyalem les funcions emprades en els mètodes dels esquemes anteriors:

- ROI: *region of interest*, XCMS crea regions d'interès basades en les m/z i rt, on aquestes regions seran desades en una llista. A més a més, es passarà una funció de neteja a la llista per eliminar regions amb intensitats i rts petits.
- *Baseline and noise*: abans del mètode per detecció de pics amb la transformada de Wavelet, les dades són classificades en tres, en *baseline* ROIs, soroll ROIs i ROIs amb la informació a analitzar, els dos primers són descartats.
- Transformada de Wavelet: una vegada s'han obtingut els coeficients de la transformada i les intensitats dels pics detectats s'examina si aquests són vàlids o no.
- *Intensities and baseline*: en aquest pas s'empra un filtratge de dades de les intensitats i també de la *baseline* [30].
- S/N: s'estima el soroll de les mostres i s'elimina un 10% dels punts d'unes dades estudiades específicament [p.18, 30]. El número de percentatge tracta del valor per defecte dins un mètode dins la funció "chromatogram()" [30].
- *Wavelet Transform*: *Wavelet Coefficient Check*, amb la transformada es torna a estudiar les dades per descartar els ROIs que no compleixin els requisits per ser pics.
- S/N: es continua emprant els valors dels coeficients de la transformada per continuar classificant les dades per descartar el soroll.
- *Ridgeline detection*: els ROIs dels pics generen màxims en els coeficients en el domini rt formant la *ridgeline*, la qual connecta els màxims dels coeficients per les diferents escales que estan properes en rt. S'empra aquest *ridgeline* per detectar pics dels ROIs [30].
- *centWave*: es du a terme un pas de filtratge dels valors obtinguts per la detecció per la *ridgeline*. També es duen a terme altres passos de filtratge dins aquest pas [30].
- S/N: una altra comprovació i filtratge del *ratio* de soroll i senyal.
- *Peak boundaries*: s'estudien els coeficients de wavelet i les escales de wavelet per totes les mostres i es refina per acabar aconseguint els límits dels pics [30].
- *Probability Distribution models*: el model exacte emprat no és de coneixement, però les distribucions probabilístiques més comunes són la Bernoulli, uniforme, binomial, normal, Poisson i l'exponencial.

- *S&G smoothing* (Savitzky-Golay filter): "per a cada punt de dades de l'espectre, l'algorisme SG seleccionarà una finestra (per exemple, cinc punts) al voltant d'aquest punt; ajustar un polinomi als punts de la finestra seleccionada; i substituïu el punt de dades en qüestió pel valor corresponent del polinomi ajustat" [34].
- *Quantile method*: "classifica les dades en un nombre determinat de categories amb el mateix nombre d'unitats en cada categoria" [35].
- *Clusters*: classificació d'objectes similars en diferents grups.
- *Sum of squared residuals* (SSR): "tècnica utilitzada per mesurar la quantitat de variància que queda explicada amb el model en un conjunt de dades" [36].
- *R-squared*: "mesura estadística en un model de regressió que determina la proporció de variància en la variable dependent que es pot explicar per la variable independent" [37].
- *Quantile approach*: "modela la relació entre un conjunt de variables predictores (independents) i percentils específics (o "quantils") d'una variable objectiu (dependent), sovint la mediana" [38].
- *Faulty limits*: juntament amb els quantils pot produir regions de pics pobres determinades. Per resoldre el problema es va millorar l'algoritme per les regions de pics.

Els fluxos en profunditat els podem seguir en els *papers* [30, 29 i 32] per saber més informació en el tractament d'informació amb aquests passos.

Trobem passos en comú com és el tractament del soroll i de la *baseline*, i passos més concrets a cada algoritme per la detecció de pics cromatogràfics. A més a més, se solen seguir aquests passos o l'ús de funcions que duguin a terme aquestes tasques, ja que tots tenen la mateixa finalitat, la detecció de pics i, s'estudien diferents mètodes per poder aconseguir aquest objectiu per veure quin és més eficient o més flexible (s'adapta millor a les dades). Per la qual cosa, trobem que els passos com la detecció de màxims d'intensitat i àrees es troba en comú en tots els mètodes, però els mètodes emprats són diferents, fent així que els fluxos de treball per la detecció de pics varïi entre algorismes.

5 Avaluació de diferents algoritmes de detecció de pics

5.1 XCMS

5.1.1 Flux de treball d'XCMS per dades de GC/MS

Els passos a seguir pel pre-processat i anàlisi de les dades per la detecció de pics per aquesta eina són els següents:

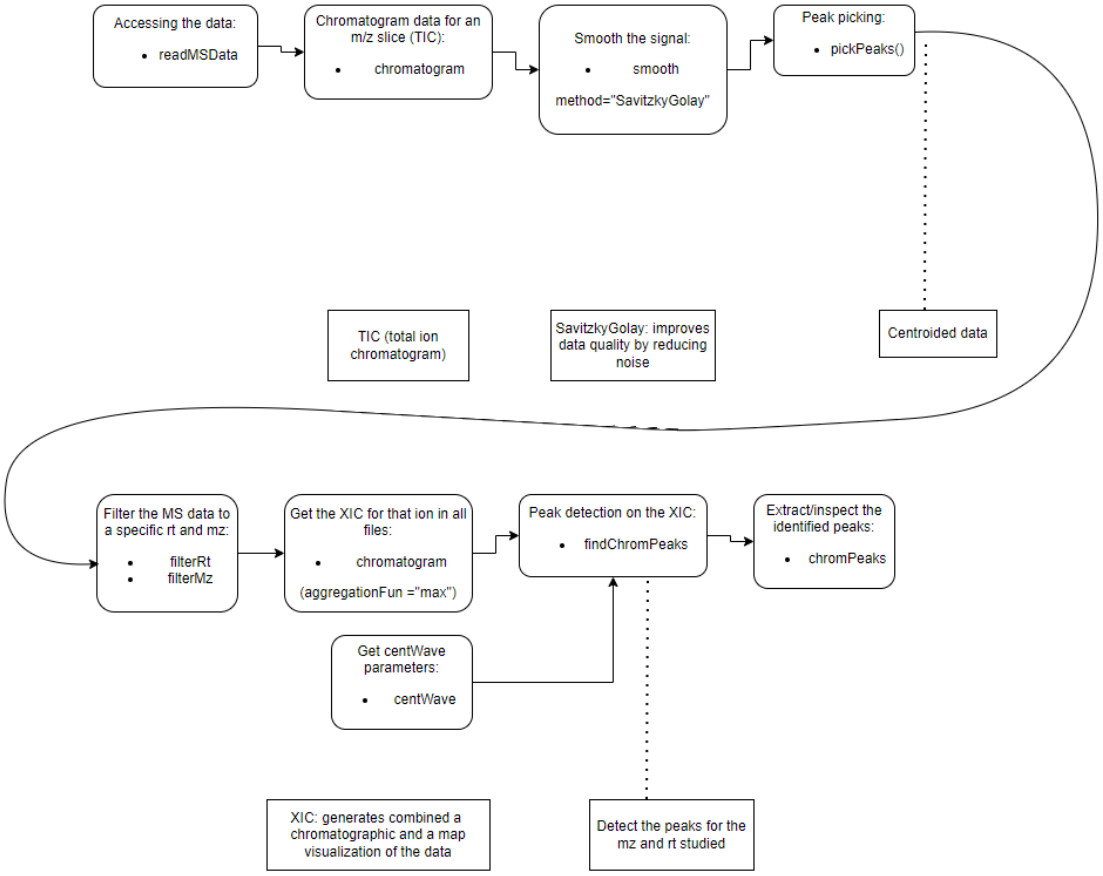

Figura 10. Diagrama de les funcions a emprar per al flux de treball per l'eina XCMS per dades d'un GC/MS [26].

- *Tuples*: "col·lecció de dades l'ordre de les quals és inalterable, o sigui, són elements ordenats en una seqüència específica i que posseeix importància" [39].

- XIC (*extracted ion chromatograms*): d'acord amb J. Rainer trobem l'explicació del que és un XIC, el qual "genera una visualització cromatogràfica i de mapa combinada de les dades (és a dir, una gràfica de les *tuples* de dades individuals m/z, rt i intensitat amb punts de dades acolorits per la seva intensitat a l'espai de temps de retenció m/z)" [26].

A més a més, podem continuar amb les següents funcions com a part de post-processament:

- refineChromPeaks: fusionar pics d'acord amb les característiques dels paràmetres.
- mergeNeighboringPeaksParam: fusionar pics que se superposin a la finestra de temps donada (*expandRt*) per arxiu si el senyal entre pics és més petita de 75% de la intensitat màxima del pic més petit [26].

En aquest punt com no tenim tantes dades a causa que ja hem escollit el rang m/z i el rt i que hem emprat diverses funcions, podem realitzar l'alineació:

- groupChromPeaks: li passem per paràmetre la funció peakDensityParam per definir els paràmetres per l'agrupació de pics.
- adjustRtime: a banda de les dades a alinear li passem per paràmetre la funció peakGroupsParam per definir la proporció de mostres per detectar pics i el grau de suavizatge (*smooth*) per permetre a diferents regions ser ajustades [26].

Els senyals d'estudi per la detecció de pics seran els senyals per les dades reals i per dades simulades, matrius ja creades a l'algoritme de l'apartat 3, o pel cas de les dades reals, la lectura ja es troba feta.

5.1.2 Flux de treball emprat d'XCMS per dades reals

Seguim el flux del diagrama de la figura número 10 per les dades reals i procedim a visualitzar el TIC (*total ion chromatogram*) per les dades:

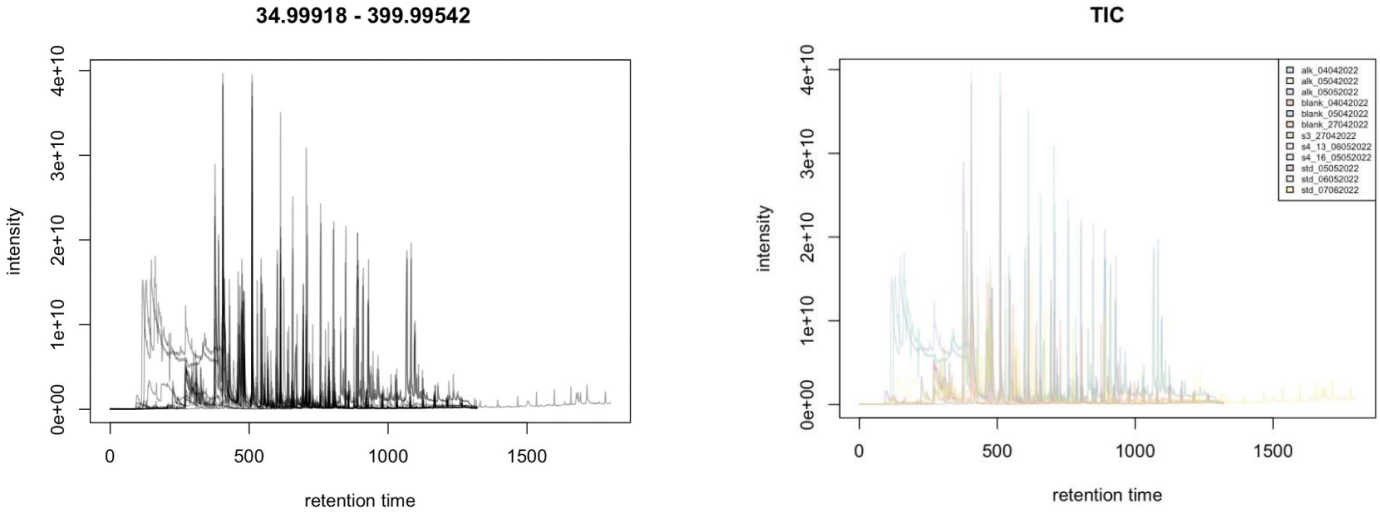

Figura 11. Cromatograma d'ions totals per dades d'un GC/MS.

Podem veure les m/z que s'han enregistrat al títol del gràfic de l'esquerra, i al de la dreta podem identificar el cromatograma de cada mostra que hem fet servir per representar aquest en projecte.

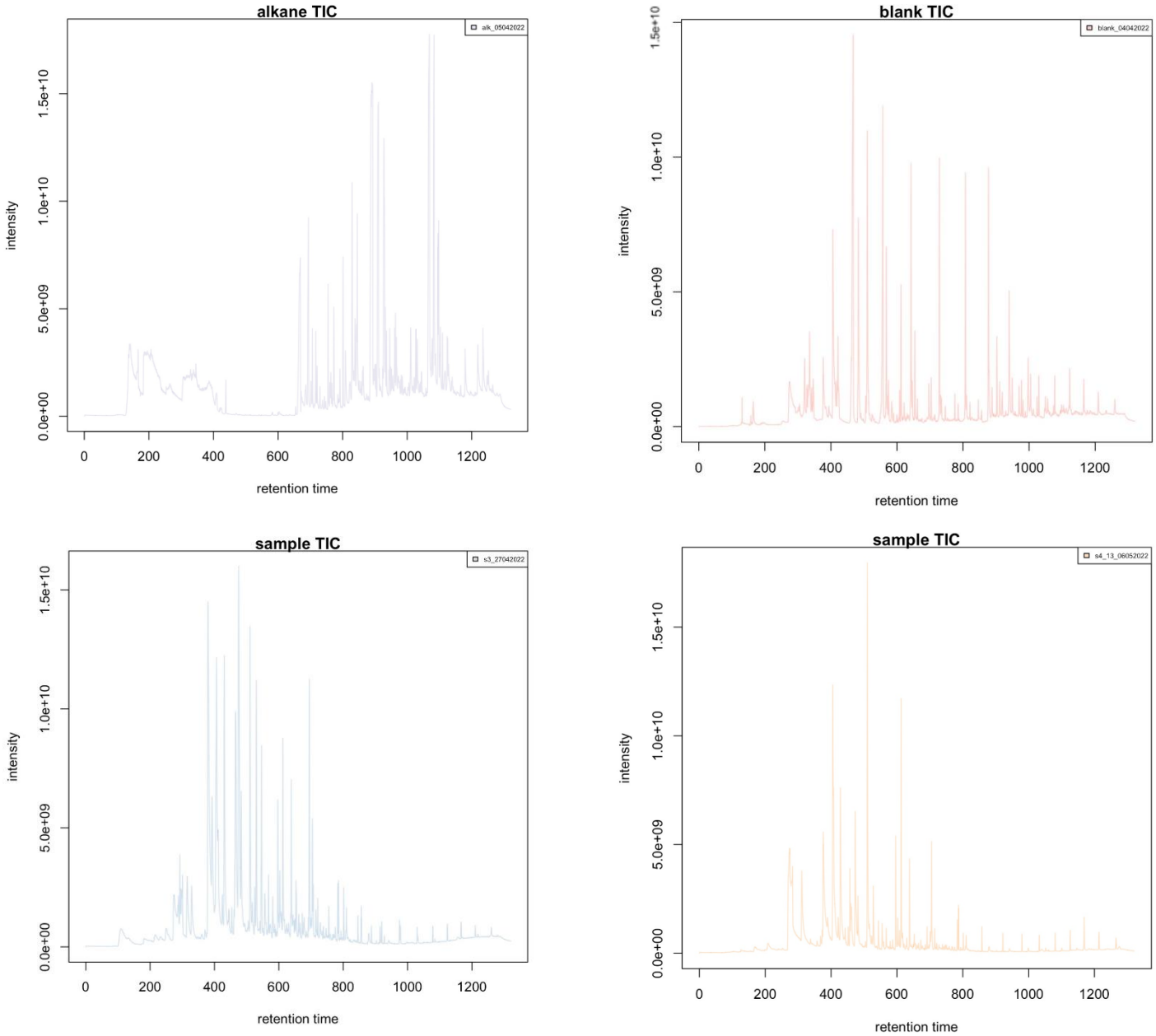

Figura 12. TIC per les mesures alk_05042022, blank_04042022, s3_27042022 i s4_13_06052022, respectivament.

Tenim diferents mostres, les mostres ja classificades com alquens i blancs i les mostres corresponents als senyals cromatogràfics dels ions obtinguts.

Respecte a la figura 12, a la gràfica del blanc, no s'introdueix una mostra al GC/MS, trobem diversos pics i també que tots els pics estan sota el valor 1.5e10 d'intensitat d'ions, però que encara així, són pronunciats al cromatograma. Si mirem les gràfiques inferiors on sí que s'ha injectat mostra es presenten els majors pics al voltant d'aquest valor, on ho podem relacionar que aquesta mostra de blanc és el residual d'anteriors execucions amb mostres al GC/MS, per la qual cosa quedarà netejar l'entrada o canviar algunes de les parts que pugui estar contaminada amb compostos anteriors. A més a més, també es pot condicionar la columna, amb ajustaments de fluxos o temperatura, entre altres [40].

A l'anterior figura, la 12, podem observar les diferències entre els cromatogrames (TIC) generats per diferents mostres dels senyals de sortida obtinguts pel GC/MS. Pel primer cas parlem d'una sortida dels alquens obtinguts per la mostra injectada, juntament amb uns dels blancs d'aquesta. Finalment, a la segona fila trobem dues de les mostres cromatogràfiques generades pels ions detectats, on en fer un estudi sobre aquests podrem relacionar-ho amb possibles metabòlits. Alhora podem analitzar els diferents tipus de gràfics que genera cada dada, com els seus màxims, àrees, o fins i tot una mica de deriva de línia base.

Abans de realitzar la reducció de soroll mitjançant el mètode SavitzkyGolay, veurem el cromatograma específic per la creatina:

- rt: 708 (segons)
- peakWidth: 5, 40
- m/z:

m/z	*Intensity*
131	100.00000
132	5.6019778
133	0.5286024
134	0.0177810

Taula 1. Intensitats de les m/z pel cas de la creatina.

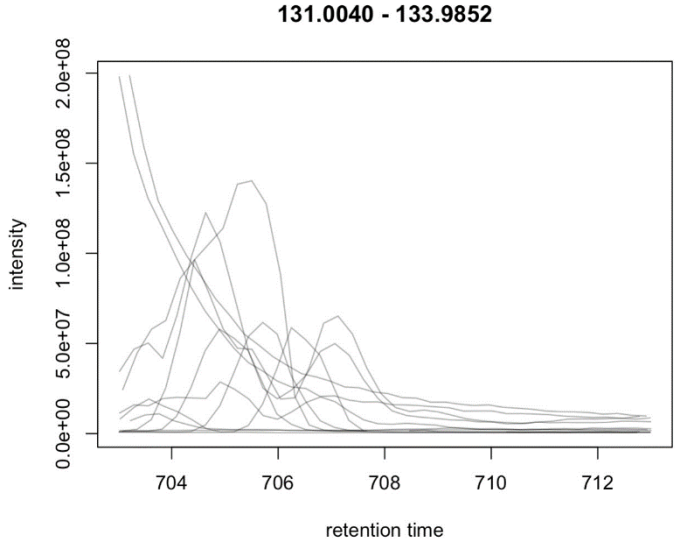

Figura 13. Cromatograma d'ions als rangs de la creatina (m/z i rt).

Podem veure, que anem des dels rangs donats en m/z i en temps de retenció nominal. Pel que fa als pics, podem veure que si presenta la figura pics propers al temps 708 segons que és on apareix la creatina. Es pot veure un total de tres pics situats propers a aquest temps.

Visualitzem el XIC de les dades en aquesta finestra, on inspeccionem les dades del perfil per l'adducte d'ions [M+H]+ pel rang on apareix la creatina.

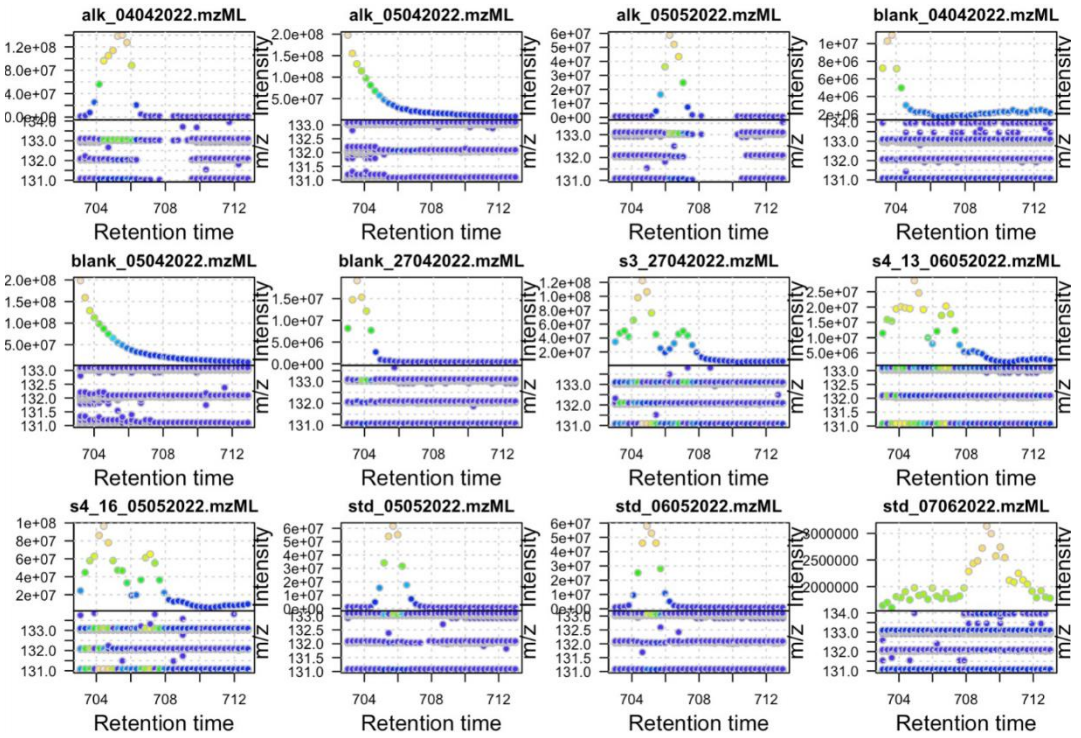

Figura 14. Visualització tipus XIC pels rangs de la creatina (m/z i rt) abans de la reducció de soroll.

Ara, realitzem la reducció de soroll i la funció de trobar pics, i tornem a visualitzar el perfil de les dades:

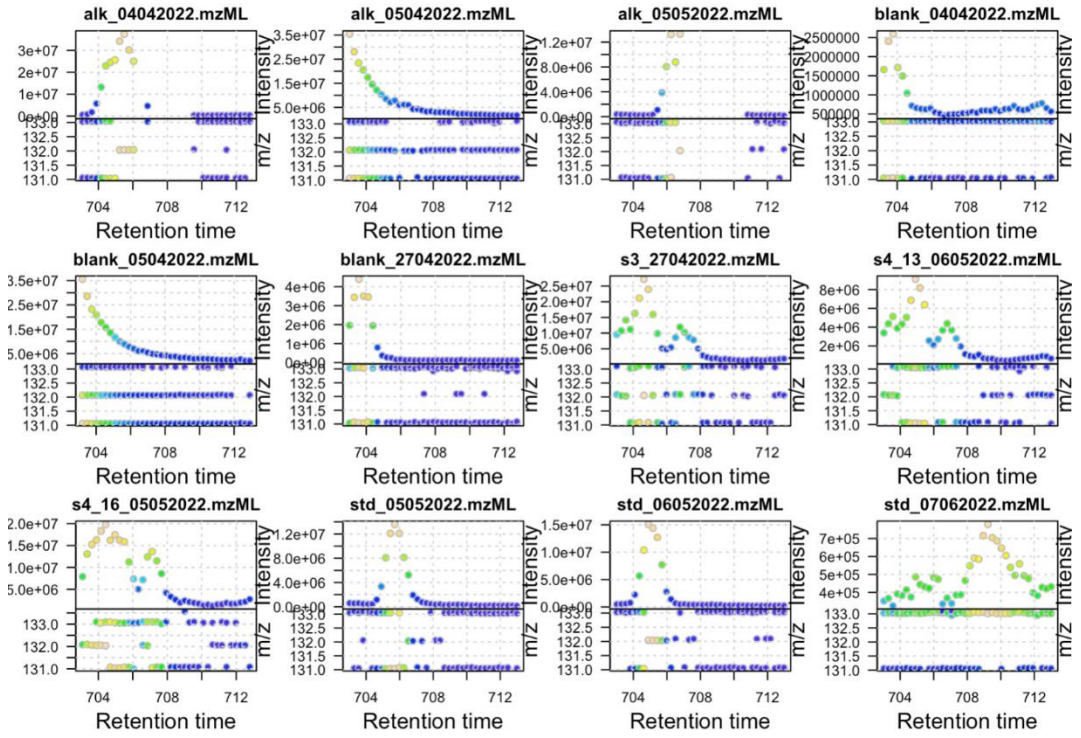

Figura 15. Visualització tipus XIC pels rangs de la creatina (m/z i rt) després de la reducció de soroll.

Podem apreciar com s'ha reduït el soroll, a més a més, podem visualitzar en quines mostres es presenta la creatina. Mirarem pics al voltant del rt 708, on totes tres mostres, s3_27042022, s4_13_06052022 i s4_16_05052022, presenten pics una mica desplaçats, també tenim que algunes funcions desplacen pics, i per això sols ser important en anàlisis d'XCMS l'ús de l'alineació. Per tant, podem dir que aquesta mostra d'orina presenta nivells que podrien ser identificats o relacionats amb la creatina.

Una vegada ja hem realitzat les visualitzacions per poder veure les dades i confirmar l'aparició del senyal volgut, filtrem en aquestes condicions les dades tractades per la reducció de soroll amb la funció chromatogram() i ho visualitzem.

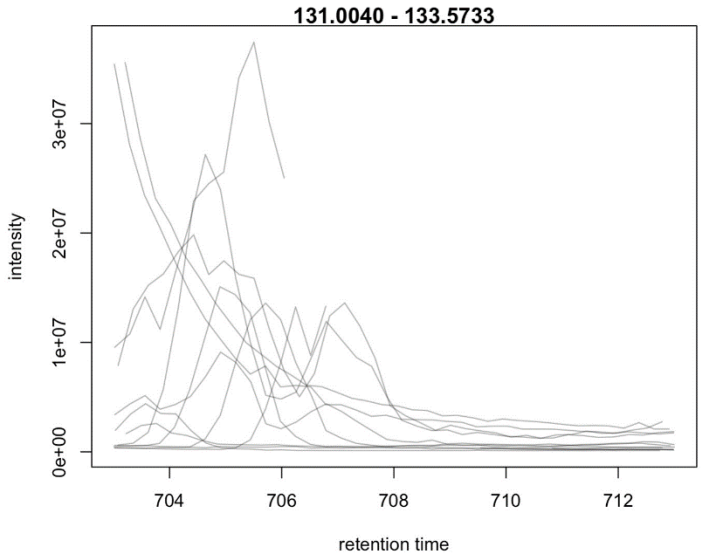

Figura 16. Cromatograma obtingut a la finestra de m/z i rt per la creatina després de reduir el soroll.

A la figura 16 representem els senyals que continguin solament dades dins la finestra de temps i la m/z, 703-713 segons i 131-134 m/z, amb els marges inclosos. Cada ona representa la suma de les intensitats obtingudes al llarg d'aquesta finestra de temps pel rang de m/z, cada senyal és el resultat obtingut per totes les dades cromatogràfiques (alquens, blancs i mostres). Més específicament, a la figura 15, als títols del "XIC" veiem de quines mostres surten els senyals observats a la figura 16, com per exemple els pics tallats a l'esquerra els veiem a les mostres alk_05042022 i blank_05042022. Aquesta visualització dels senyals és una vegada ja hem realitzat la correcció de soroll.

Podem veure que els pics en comparació amb els vists a la figura 13 es troben una mica més agrupats, a més a més, de la correcció dels senyals en les seves intensitats i àrees.

Finalment, realitzem una detecció de pics amb findChromPeaks() i chromPeaks(), on veiem que encara que puguem veure senyals als gràfics aquests no són detectats per les funcions.

El flux de treball total d'XCMS és:

- *Data import*
- *Initial raw data inspection*
- *XCMS chromatographic peak detection*
- *XCMS alignment*
- *XCMS correspondence*
- *Handling missing values*
- *Evaluating the process history*

Ens queda fer del quart pas endavant, com en aquest cas no hem trobat informació, agafem les dades tractades amb la reducció de soroll i realitzem una detecció de pics amb aquestes dades per continuar amb el flux de treball.

En realitzar l'alineació i veure-la podem veure els canvis ajustats en el temps de retenció:

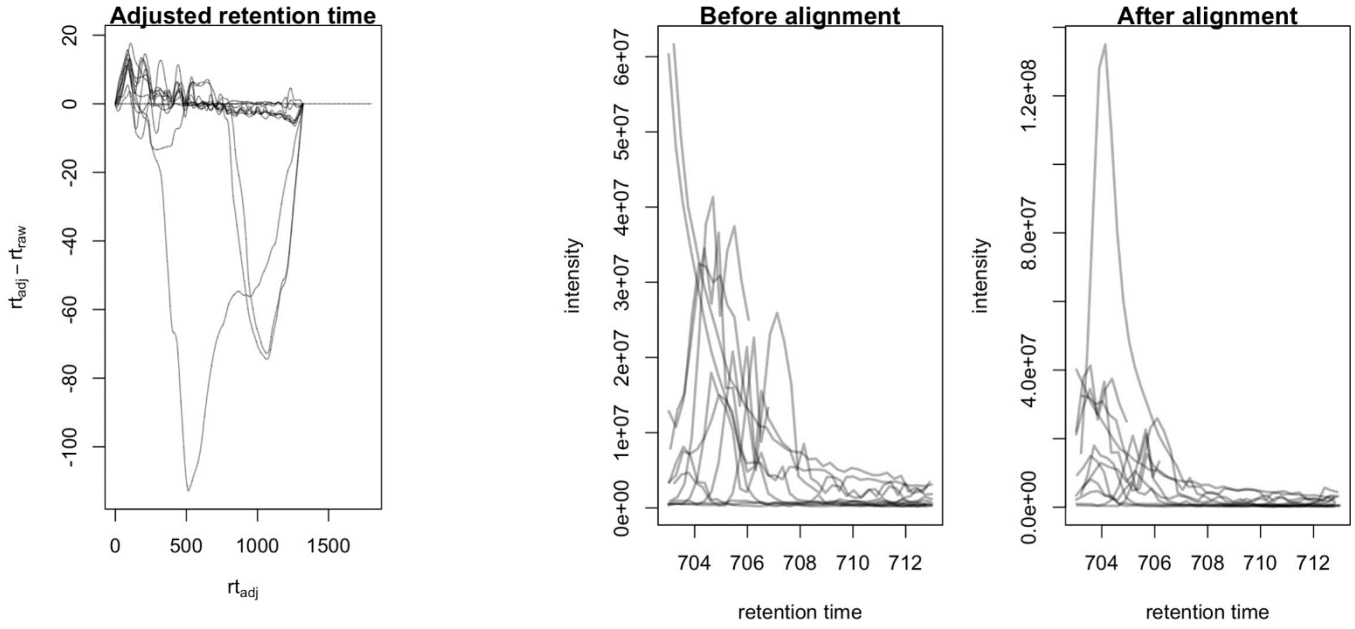

Figura 17. Temps ajustat a l'esquerra. Cromatograma sense ajustar i ajustat a la dreta.

Podem veure com s'han agrupat els pics al llarg del cromatograma, més específicament per la finestra de dades de la creatina.

Al primer gràfic podem observar la correcció de temps que s'ha fet al llarg del temps per cada senyal (rt ajustat -rt inicial) de cada m/z, on a la dreta comparem l'abans i després de l'alineació dels pics al llarg del temps pels càlculs de temps de correcció.

Seguidament acabem el flux amb la correspondència:

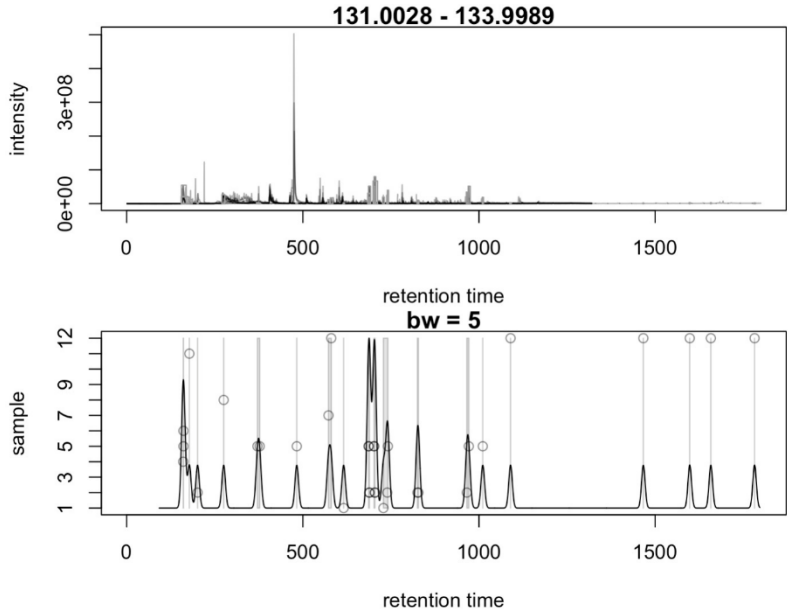

Figura 18. Cada punt representat són els pics per mostra, la línia negra és la distribució de la densitat del pic i els rectangles grisos són els pics agrupats (*features*).

A la figura 18 representem la correspondència, la qual junta pics cromatogràfics entre les mostres, suposadament pels mateixos ions.

Al gràfic superior observem el cromatograma extret per les m/z definides (131 a 134) per totes les dades cromatogràfiques (alquens, blancs i mostres). Així mateix, al gràfic de sota podem observar una agrupació de pics amb un rectangle proper al valor de 708 segons, amb la qual cosa, com bé es troba esmentat, hem aconseguit extraure com a mínim una agrupació pel valor del temps de retenció proper a 708 amb la presència d'alguna o algunes m/z de 131 a 134, informació lligada pels ions de la creatina, com bé estàvem buscant. A més a més, podem observar que els cercles són els pics dels senyals obtinguts per les mostres, on no podem extraure del gràfic a quina mostra correspon cada cercle, però podem observar el resultant de tots els pics de les mostres per aquest rang de m/z.

No realitzem el tracte de valors perduts ni avaluem l'historial, ja que ara sols ens volem centrar en els pics detectats i no totalment en la manipulació de dades per fer anàlisis completes d'aquestes, on en cas afirmatiu, es podria continuar amb l'estudi amb aquests passos i amb anàlisis estadístiques amb els *features* trobats com PCA (*Principal Component Analysis*) o *Volcano Plot*.

5.1.3 Flux de treball emprat d'XCMS per la simulació

Com que XCMS treballa amb l'objecte onDisk el que farem per poder fer el flux de treball per aquest mètode serà afegir les dades de la simulació a l'objecte ja creat.

Els passos que consta per fer-ho seran els del següent diagrama:

Figura 19. Diagrama de punts a canviar a l'objecte MSnbase per contenir les dades de la simulació a analitzar.

A phenoData@data canviem tant el *sample_name* de les dades com les *row_names*.

Canviant això obtenim l'objecte MSnbase "OnDiskMSnExp". Ara procedim a realitzar els passos del diagrama 10.

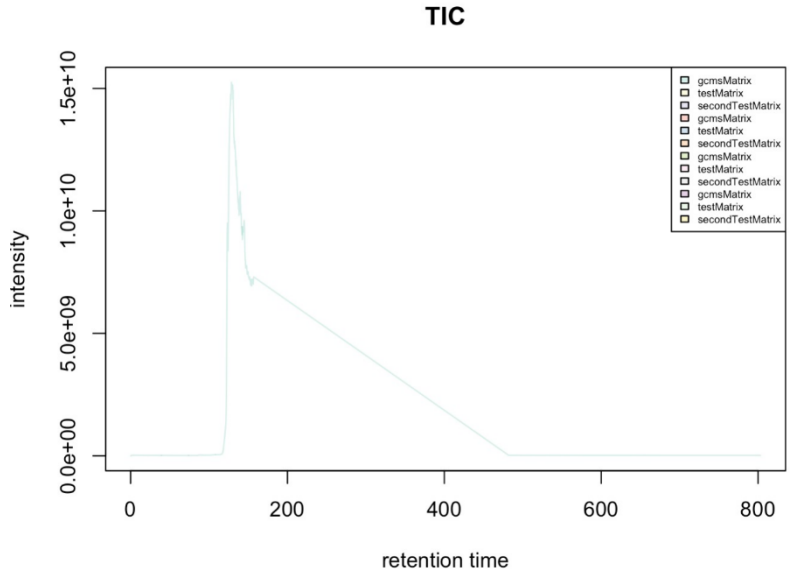

Figura 20. Cromatograma total d'ions de la simulació pel mètode del flux de treball XCMS.

Canviant l'objecte de la manera esmentada a la figura 20 trobem que sols hem aconseguit omplir la primera classe de l'objecte amb el codi emprat, per això sols es representa el senyal en un color. També, en afegir dades sols hem omplert pel cas de les mostres d'una matriu amb la informació del cromatograma, es podria continuar omplint amb més informació. A més a més, hem omplert les dades per la informació de cada ió dels metabòlits i no per les

seves *features*. També observem que, tot i que hàgim afegit dades per dades al llarg d'un temps de retenció de 0 a 1800 segons, sols es representa el senyal fins a 800 segons.

Filtrem les dades pel cas de la m/z i temps de retenció amb una finestra de marge per ambdós casos pel cas de la creatina.

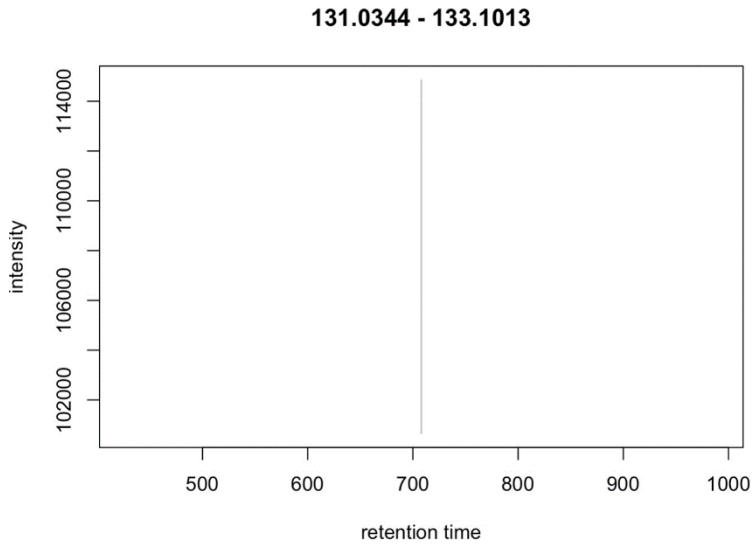

Figura 21. Finestra en el temps de retenció i m/z en el rang de la creatina pel cas de la simulació amb l'ús de funcions d'XCMS.

Com podem veure hem trobat un pic de senyal que podria fer referència a un ió del metabòlit de la creatina. En haver ficat nosaltres les dades sabem que justament aquest pic sí que pertany al metabòlit, també trobem que dins aquest marge podríem trobar altres pics fent referència als altres ions de la creatina, però com la seva intensitat és molt més baixa no es poden visualitzar al gràfic. A més, trobem que de la manera en què hem afegit les dades no deixa espai perquè es puguin veure efectes adversos com el soroll o la *baseline*, encara que afegim que hi apareixen més pics per senyal o es canviï el rang de m/z (*lowMZ* i *highMZ*) continuarem observant els mateixos resultats. A més a més, trobem que obtenim un pic i no una corba del senyal, això ho relacionem amb la manipulació de dades a l'objecte, on hem de fer un estudi més exhaustiu en com les dades de la matriu principal (*featureData*) s'empren amb les funcions d'XCMS, ja que encara que hàgim afegit les dades necessàries per visualitzar ones obtenim aquesta resposta.

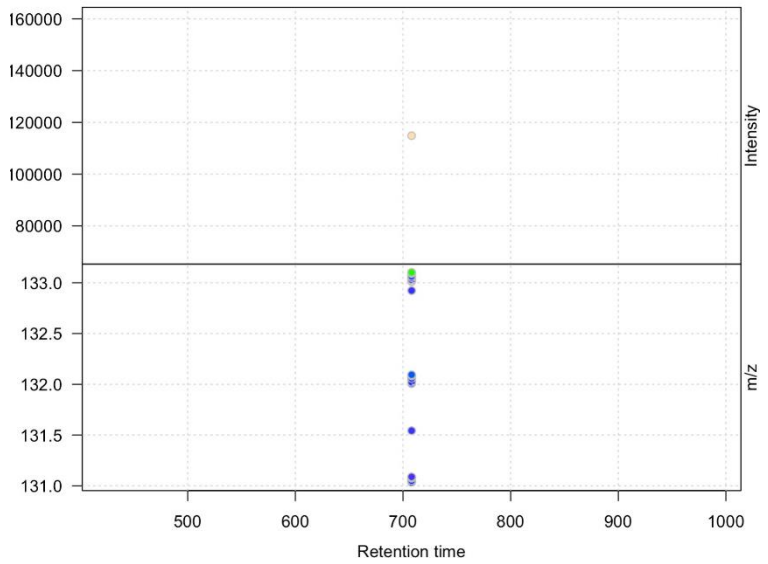

Figura 22. Rang de m/z i temps que presenten dades pel cas de la simulació amb l'ús de XCMS.

Podem observar les diferents m/z que presenten un pic per aquest rang de temps. Els senyals que presentin dades per una finestra de temps per 708 segons i per m/z entre 131 i 134, inclosos, es veuran representats a la figura anterior. Les m/z obtingudes es troben representades al gràfic de sota, on cada nombre de l'esquerra és la m/z pel temps de retenció en l'eix x, observem que totes les m/z coincideixen en tindre senyal, o un pic, al temps 708 i que aquesta intensitat llegida és mostrada al gràfic de dalt de la figura 22. Al gràfic de la intensitat veiem sols una esfera, on més endavant veurem que és causa que els altres valors són molt petits, i per aquest cas el gràfic representa dades des de 80000 fins a 16000 d'intensitat, per la qual cosa no deixa marge per intensitats amb pics més baixos.

Realitzem la reducció de soroll a les dades i ho visualitzem. La funció de trobar pics ("pickPeaks()") pel cas de les dades simulades no es pot emprar per a aquesta informació. Tot i que hem omplert totes les dades necessàries de la matriu "featureData@data" amb les dades in silico, no s'aconsegueix que aquesta funció doni resultats. Cal fer un estudi més exhaustiu per aquesta funció i el perquè no es pot obtindre una sortida amb nombres per aquest cas, ja que podria ser a causa que no conté milers de pics l'objecte creat, ja que pel cas de les dades reals trobem un total de 60564 observacions en total, i pel nostre cas hi ha 581 amb dades.

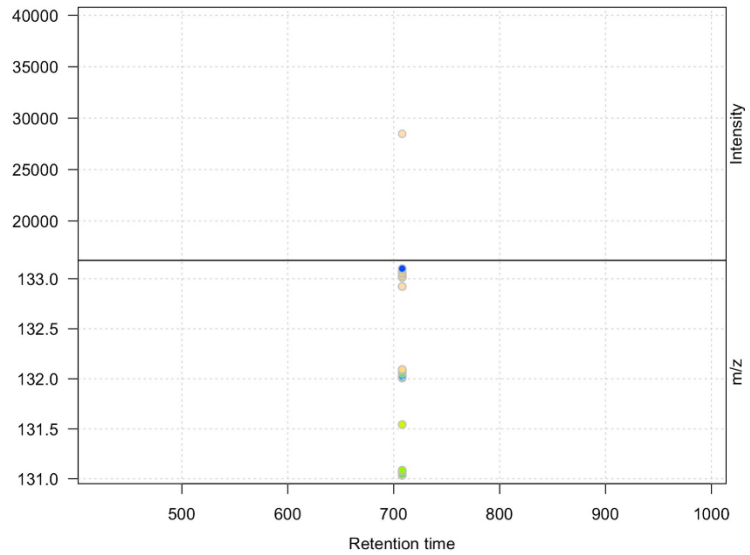

Figura 23. Rang de m/z i temps que presenten dades pel cas de la simulació amb l'ús de XCMS després de la reducció de soroll.

Cal mencionar que XCMS treballa amb m/z fraccionades com bé podem observar als gràfics obtinguts, però pel nostre cas, quan simulem i treballem amb funcions específiques, convertirem les dades a format nominal.

Com és una simulació i sols hem afegit les dades d'una matriu, en què volguéssim observar molts més senyals en fer les cerques específiques, hauríem d'afegir moltes més dades a l'objecte emprat. A més a més, el problema de la visualització del soroll i la *baseline* hauria d'estudiar-se a part.

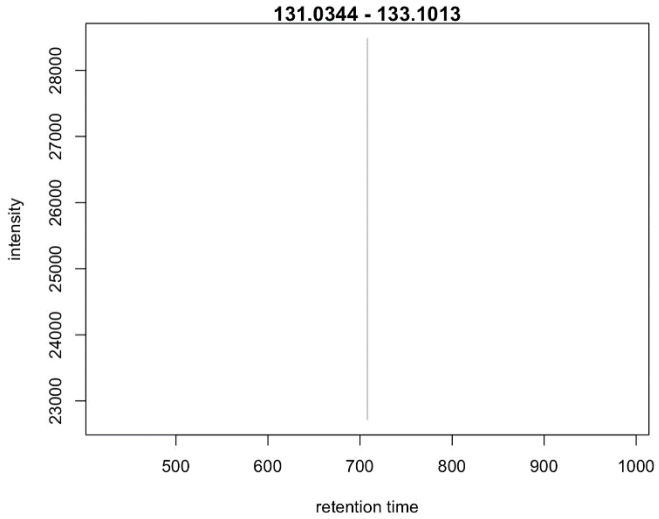

Figura 24. Finestra en el temps de retenció i m/z en el rang de la creatina pel cas de la simulació amb l'ús de funcions d'XCMS després de la reducció de soroll.

En filtrar en aquest rang després dels passos previs es pot veure a la figura 24 que obtenim el mateix gràfic que al de la figura 21, ja que les dades obtingudes abans ja es trobaven molt nítides, per la qual cosa en comparar les figures no trobem diferències. En cas que s'aconseguís representar l'ona del senyal, segurament podríem apreciar aquesta correcció del soroll en les dades, aconseguint així una imatge més neta.

D'ara endavant no es poden seguir els passos del flux de treball, podem pensar que és justament per l'abans mencionat, per la manca d'observacions, ja que d'ara endavant les funcions del flux de treball d'XCMS no donen sortides vàlides.

5.1.4 Discussió

Primerament, cal esmentar que pel cas de la simulació trobem que a causa d'aquest tipus de manipulació de dades d'un objecte MSnbase "OnDiskMSnExp" hem hagut de parar el flux de treball a causa que les funcions no funcionen encara que hàgim reemplaçat dades i no hàgim modificat el nombre de files, columnes o nombre d'espais als vectors. Podem pensar que la causa principal que no es pugui continuar amb el flux és la manca d'informació, ja que les observacions pel cas real són 60 mil aproximadament i les nostres són 500 aproximadament, i a causa del mencionat, que no hem esborrat dades sols canviades el seu valor, no es prediu que l'aturada del flux sigui a causa d'altres variables. Parlem que no hem esborrat dades i alhora que tenim 500 observacions, això és perquè la resta d'observacions han sigut tractades a valors 0, ja que no disposàvem de més informació per omplir la matriu de "featureData@data".

D'altra banda, podem comparar els gràfics obtinguts pels dos casos, on el cromatograma total d'ions de totes les mostres és molt més net i notable els pics específics corresponents als ions dels metabòlits pel cas de la simulació que pel cas real, però que en cas que s'afegissin moltes més dades es podria acabar obtenint uns resultats similars. També pel cas de la simulació tenim la manca de diferents grups que presentin dades, com alquens, blancs o més mostres amb senyals dels ions dels metabòlits.

En visualitzar la finestra pel cas dels valors de temps de retenció i m/z desitjats el cas real presenta molta més informació que el simulat, ja que aquest és un entorn molt més controlat, però de seguir l'esmentat abans podríem aconseguir uns resultats similars. Passa el mateix amb la visualització de les mostres al llarg de les dades de l'objecte, el cas simulat és el més ideal per les dades que es poden apreciar a la figura 22 en relació amb la taula 1, i el cas de les dades reals a la figura 13 podem apreciar millor les m/z i senyals pels valors cercats, on alhora en ser mostres reals costarà més trobar que els pics estiguin centrats al valor exacte de temps de retenció a causa de diverses variàncies com podem veure algunes exemplificades al punt dos d'aquest treball. Pel cas de la reducció de soroll sempre es notarà millor com més soroll hi hagi en una mostra i, encara que hàgim intentat afegir a la taula de l'objecte soroll no s'ha assolit generar cap punt de conflicte, però pel cas real si es nota més la correcció de valors i neteja. On ho podem acabar de veure en visualitzar un altre cop el cromatograma. A més a més, esmentar que no hem pogut afegir suficient soroll pel cas de la simulació o bé que, els senyals de les mostres presentaven uns valors d'ona molt més alts que el soroll, fent que la relació senyal-soroll sigui molt més alt "in silico" que per les mostres reals.

D'altra banda, en cas que per la simulació pugem més valors pel soroll o la deriva de línia de base hauríem d'inserir a la matriu "featureData@data" si aquests senyals produiran pics al llarg del temps de retenció, i en cas afirmatiu quan i quant (en quin temps i quina

intensitat) i també ajustar a quina m/z pot aportar dades significatives. Per la qual cosa, si ho ajustem per tal que sí que presenti corbes al cromatograma podríem observar al TIC aquests efectes. Alhora, trobem que si ho modifiquem d'aquesta manera el soroll i la deriva pot ser confós com a senyal d'un ió (depenent de la intensitat per aquestes pertorbacions), però com bé hem vist anteriorment, hi ha ions que no presenten gran intensitat, per la qual cosa, aquestes pertorbacions poden emmascarar valors d'ions i alhora es poden identificar aquestes ones com possibles TP (*true positives*). Per aquest tipus de casos, XCMS té un flux específic per intentar descartar valors que puguin identificar-se com pics de senyals d'ions i no imperfeccions, com bé podem veure al flux de la figura 7, com el tractament dels ROIs per veure si poden pertànyer al senyal de l'ió o és soroll o deriva. Aquest tractament es troba intern en les funcions del flux de treball, com per exemple en emprar chromatogram() o findChromPeaks().

Com bé hem esmentat, el flux de treball pel cas de la simulació acaba aquí, on es pot apreciar que pel cas de dades reals, en haver intentat realitzar la funció findChromPeaks() ja en la finestra de temps i m/z tampoc aconseguim valors com el cas de la simulació, ho podem relacionar aquests resultats per la manca d'informació a la matriu "featureData@data", ja que ambdós no presenten la mateixa quantitat que pel cas en el qual sí dona resultats aquesta funció.

Seguint el flux de treball pel cas de les dades reals podem veure com afecta l'alineació, ajudant a rectificar alguns temps de retenció, deixant un cromatograma amb unes corbes més relacionades entre elles i omplint buits de dades. Finalment, tenim la correspondència, on, com bé hem esmentat, podem observar els pics i distribució d'aquests juntament amb l'agrupació de pics.

Per acabar, pel que fa a la detecció de pics, pels dos casos hem pogut trobar la informació d'algun ió de la creatina al cromatograma, pel que fa al cas real hem pogut observar amb més nitidesa els diversos pics amb les m/z amb el format de visualització tipus XIC a la figura 15 i com es veu també aquests pics en el cromatograma normal amb la figura 16, i pel cas de la simulació a la figura 23 i 24. Gràcies a l'alineació i la correspondència hem pogut classificar millors els pics per diferenciar-los d'altres. I pel cas de la simulació, ben aviat hem pogut trobar i detectar la informació desitjada i amb molta més nitidesa.

5.2 GCalignR

5.2.1 Flux de treball de GCalignR per dades de GC/MS

És una eina creada per l'alineació per tipus GC/FID (el FID, "*Flame Ionization Detector*" és d'un sol canal, suma totes les m/z en una sola senyal). És per això que la configuració GC/FID dona sempre directament el TIC.

Per tant, aquesta eina es basa en temps de retenció, però que també pot manipular dades que continguin informació de temps de retenció per pics com el GC/MS [27].

El format d'entrada per aquesta eina serà una llista de pics, amb el seu temps de retenció per cada àrea de pic. A causa que es basa en les dades de temps de retenció la qualitat de les dades que li donem afectaran significativament en els resultats [27].

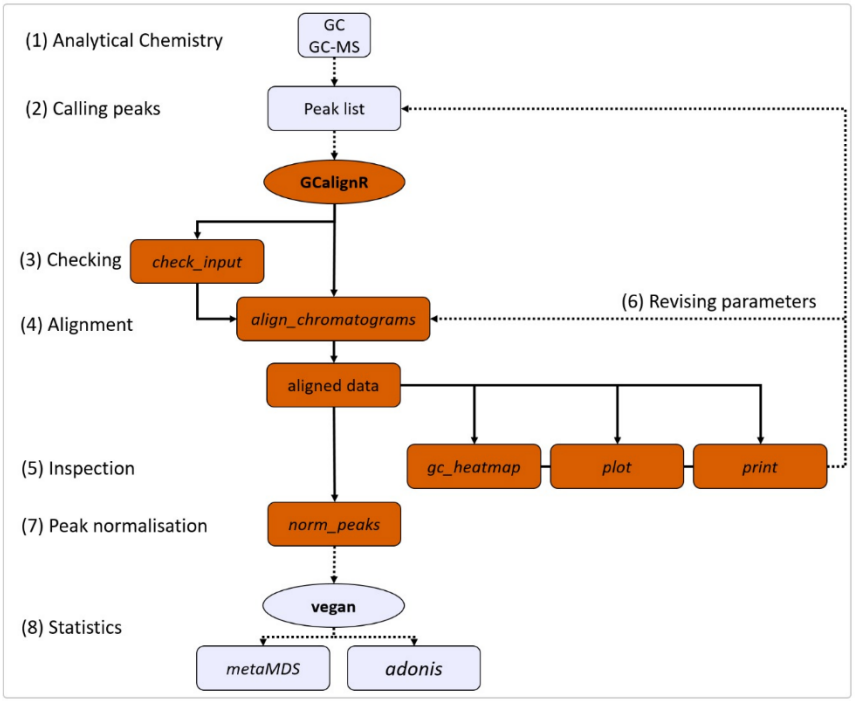

Figura 25. Diagrama de les funcions a emprar per al flux de treball per l'eina GCalignR per dades d'un GC/MS extret de [27].

Funcions:

- `align_chromatograms`: The entire alignment procedure is implemented within this function based on three parameter values described below. Additional parameters allow optional processing steps.
- `plot.GCalign`: Diagnostic plots summarise the aligned dataset.
- `print.GCalign`: Summarises the alignment process and lists all arguments to the function call.
- `gc_heatmap`: Visualises alignment results and enables to quickly inspect the distribution of substances (i.e aligned peaks) within the dataset. Furthermore, the deviation from mean retention times of a given substance can be used to detect potential issues with the alignment.
- `check_input`: Checks the format of the input data for conformity with the requirements and highlights violations.
- `norm_peaks`: Normalises the abundance measure of peaks by calculating relative intensities within samples.
- `as.data.frame.GCalign`: Outputs aligned datasets within a list of data frames for each variable in the input data.

Figura 26. Imatge esmentant les funcions de GCalignR principals i el seu procediment, extret de [27].

Opcionals:

- delete_single_peak: si al seu atribut li fiquem *TRUE* esborrarem pics que sols passen a una sola mostra [27].
- align_chromatogram: es poden esborrar mostres especificant els seus noms en l'argument "blanks".

5.2.2 Flux de treball emprat de GCalignR per dades reals

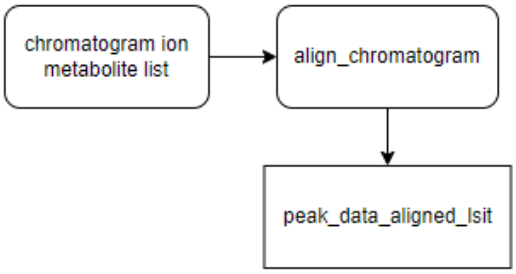

Figura 27. Flux de treball de GCalignR en R per dades reals.

Primerament, creem una llista amb les dades extretes del cromatograma real en format matricial i després la donem a la funció "align_chromatogram()" on els passos 3 al 7 de la figura 25 estan encapsulats en aquesta funció. Posteriorment, podem visualitzar les dades tractades amb la sortida d'aquesta funció.

De temps li donem de 2 a 30 minuts, negligint així els valors dels primers minuts. El gràfic dels pics alineats serà amb temps en minuts en comptes de segons.

Tenim que pel cas de les dades reals es troben un total de 27 substàncies i que no es tenen cap en comú entre mostres. També que en cada mostra s'ha alineat el mateix nombre de pics.

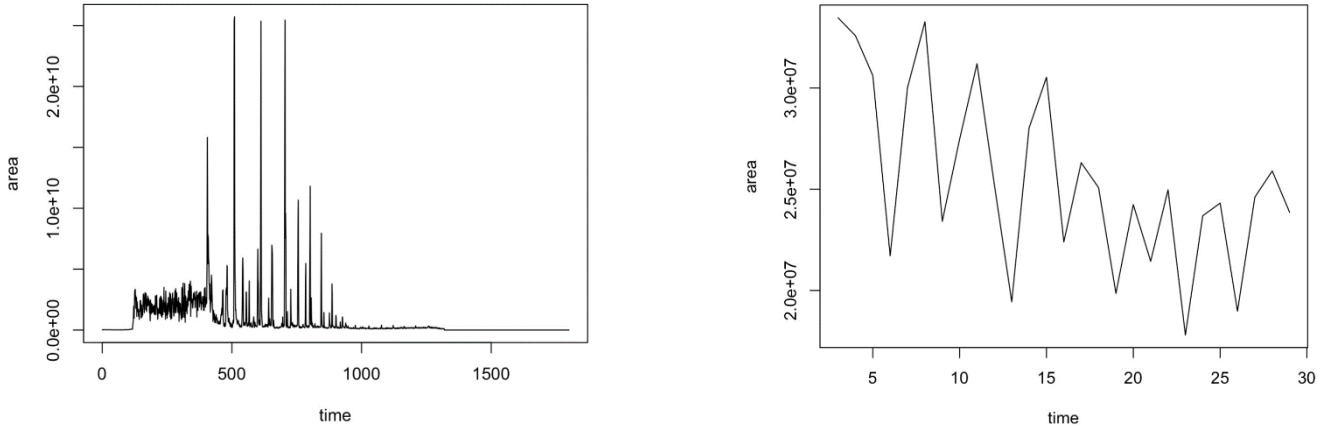

Figura 28. Gràfic de les intensitats llegides al llarg del temps per una mostra a l'esquerra. Gràfic dels pics obtinguts amb la funció de GCalignR de les dades reals a la dreta.

El gràfic de la dreta en comptes d'emprar el temps de retenció en segons, s'utilitza en minuts.

Les mostres reals en format matricial donades per trobar pics (s3_27042022, s4_13_06052022 i s4_16_05052022) presenten un gràfic similar al de la figura 28. On ho relacionem que totes tres són mostres d'orina on tenen molts o tots els metabòlits en comú, variant l'abundància d'aquests i possibles pertorbacions o imperfeccions, per la qual cosa, han de presentar un gràfic semblant.

Alhora podem observar que el gràfic de la figura 28 a la dreta en relació amb el de l'esquerra no té massa sentit, ja que els pics identificats i la seva àrea pel cas de la dreta no guarda relació en molts dels pics observats al cromatograma de l'esquerra.

5.2.3 Flux de treball emprat de GCalignR per la simulació

Primerament realitzem una correcció de la *baseline* perquè els pics es puguin veure millor. No duem a terme més passos de correcció, ja que volem comparar els resultats obtinguts per GCalignR per llistes de mostres amb la informació de les intensitats al llarg del temps amb les funcions que donen cada algoritme de detecció de pics.

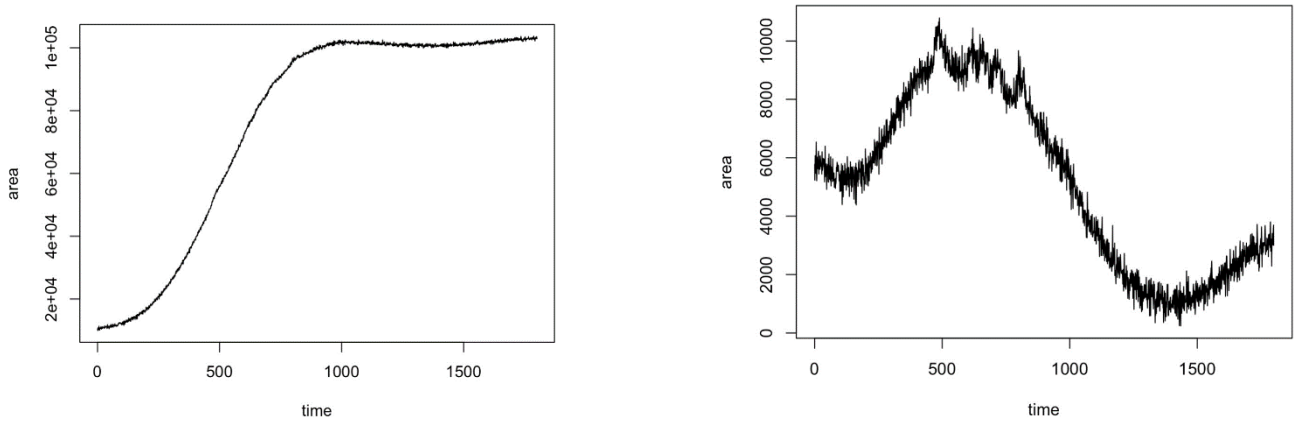

Figura 29. El cromatograma total dels ions (TIC) per la primera mostra de les dades simulades a l'esquerra, i la correcció del *baseline* a la dreta.

Podem observar a la figura anterior que en corregir la *baseline* alhora també s'ha modificat els valors de la intensitat, obtenint un ordre de magnitud inferior que a l'esquerra pel que fa a la intensitat (eix vertical).

Repetim els passos del punt 5.2.3 amb la llista creada en el punt quatre per les dades simulades.

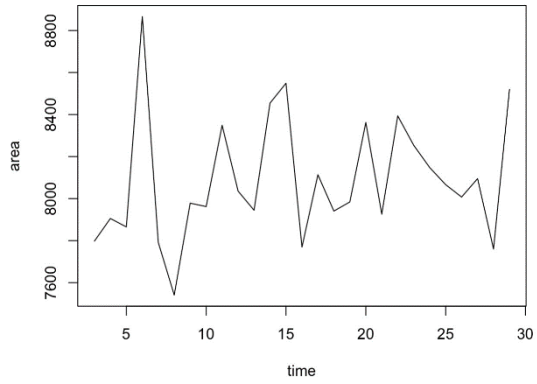

Figura 30. Gràfic dels pics obtinguts amb la funció de GCalignR de les dades simulades.

Aconseguim un total de 27 substàncies identificades en les dades. També no s'han trobat substàncies compartides entre les mostres i s'ha realitzat el mateix nombre d'alienacions en cada mostra.

Amb el gràfic de la figura 30 ens trobem un altre cop amb la problemàtica de la relació amb el cromatograma i els pics, ja que els pics identificats per la figura 30 no es poden relacionar majoritàriament amb els que podem observar a la figura 29 a la dreta.

5.2.4 Discussió

Podem veure que en els dos casos s'han identificat el mateix nombre de substàncies, entenem per substàncies compostos que continguin aquests ions dels metabòlits donats. Pel cas de les dades aportades per la llibreria original de l'algoritme s'obtenen altres valors, però per aquests dona el mateix encara que les dades reals tenen molts més metabòlits analitzats, ja que les dades reals venen d'una mostra real d'orina analitzada en un GC/MS, per tant, d'acord amb els resultats de l'anàlisi d'orina hem de trobar més de cinc metabòlits, i pel nostre cas hem emprat cinc per a la simulació i encara així hem obtingut els mateixos resultats.

A banda d'això, les dades reals presenten un TIC molt similar entre mostres (s3_27042022, s4_13_06052022 i s4_16_05052022) i les dades simulades tenen en comú diversos metabòlits (normalSimulated, alteratedSimulated i missingMetabolite) però la funció no ha pogut identificar similituds entre les mostres, en canvi, per dades web que aquest mètode de treball ofereix per l'estudi de dades cromatogràfiques [27] si es troba tota aqueta informació i es pot estudiar, com el nombre de components compartits entre mostres o l'alineació feta.

Finalment, tenim la detecció de pics, les figures 28 i 30, on si ho comparem amb els TICs veurem que tampoc s'identifiquen els pics més pronunciats i amb amplada en cada mostra.

5.3 Algoritme no existent basat en el flux de treball per la detecció de pics

Per la creació de l'algoritme seguirem el flux de treball de les figures 5 i 6 i, ens basarem una mica en els mètodes emprats per la detecció de pics de les figures 7, 8 i 9.

Les dades que emprarem serà en format matricial amb les intensitats obtingudes al llarg del temps de retenció per cada m/z. Aquesta matriu és equivalent a la sortida de la funció creada amb "createMzAndIntensityLargeMatrix()".

Tenim que el flux a seguir serà:

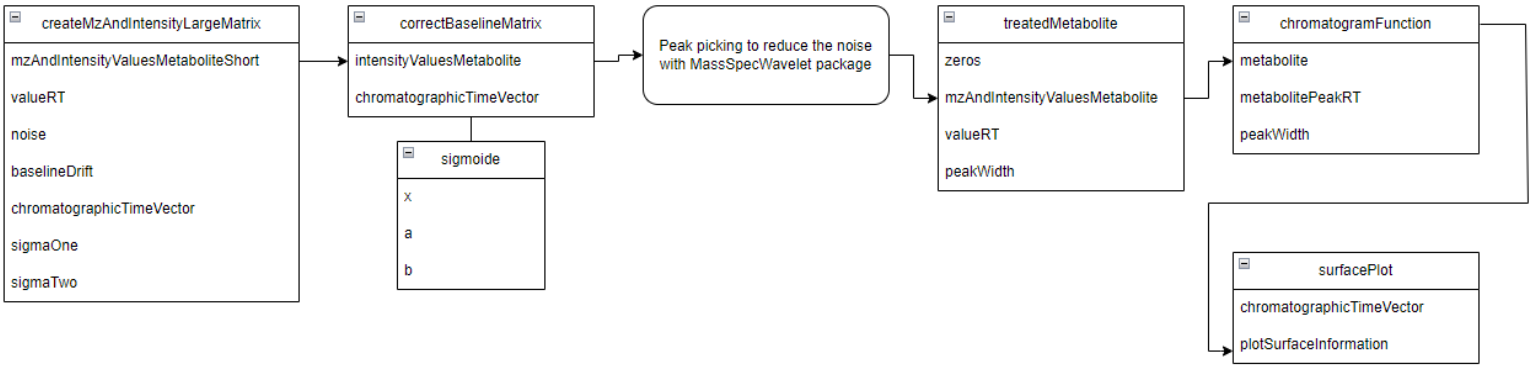

Figura 31. Diagrama de flux de l'algoritme creat per la detecció de pics.

Les funcions a fer servir es troben explicades a l'apartat 3, cal afegir que s'ha modificat el codi de la funció correctBaseline() per poder corregir una matriu sencera i no sols un vector.

Hem emprat el paquet de MassSpecWavelet perquè amb la transformada de Wavelet podem obtenir "una representació del senyal tant en temps com en el domini freqüencial" [41] i podem fer una identificació dels pics d'una manera ràpida i eficient, ja que ja es troben treballades les funcions i atributs necessaris per a la identificació de pics amb aquest mètode.

- paquet MassSpecWavelet: fem servir el codi per obtenir els coeficients de Wavelet i emprar-los per extraure els índexs dels pics al llarg del cromatograma, una vegada obtinguts, disminuïm els valors de la resta de dades que no siguin aquests valors ni els seus voltants.

El paquet identifica els índexs m/z dels pics identificats, però pel nostre cas, en comptes de passar-li un llistat dels m/zs li passem del temps de retenció.

5.3.1 Flux de treball emprat de l'algoritme creat per dades reals

La matriu nominal extreta al punt quatre de la informació real serà utilitzada, on el següent pas seria la correcció de la *baseline drift*, però per aquest cas no cal fer-la, ja que no presenta cap desplaçament del senyal propi d'aquest efecte. Anem directament amb el tractament amb la transformada de Wavelet i el seu procediment per trobar els pics:

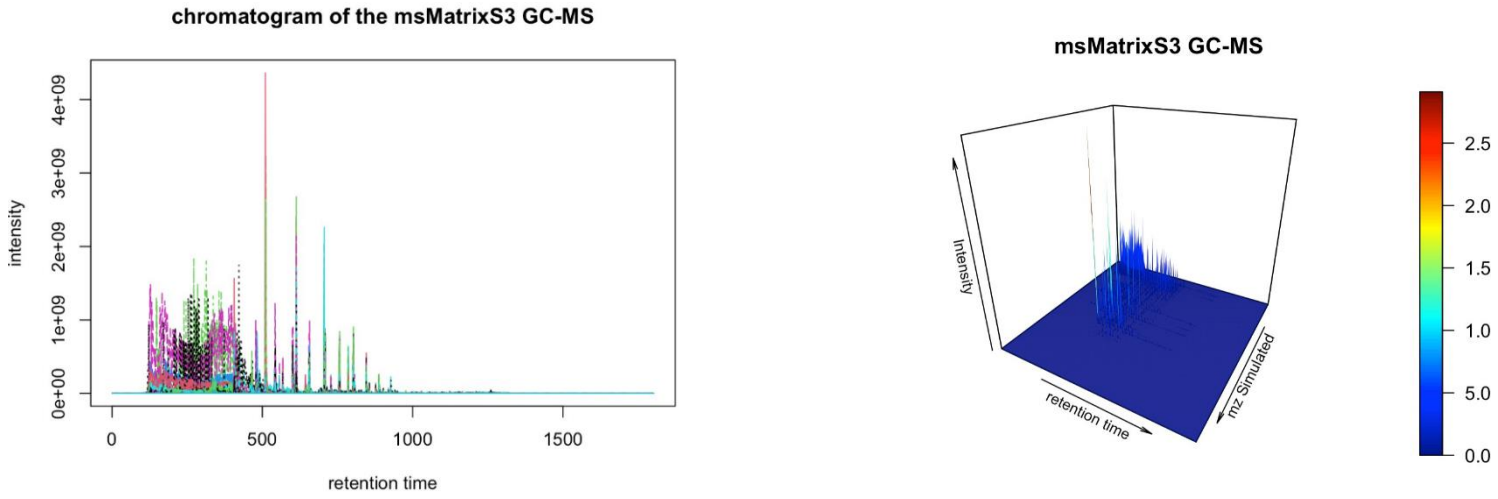

Figura 32. Cromatograma en 2D i 3D de les dades reals.

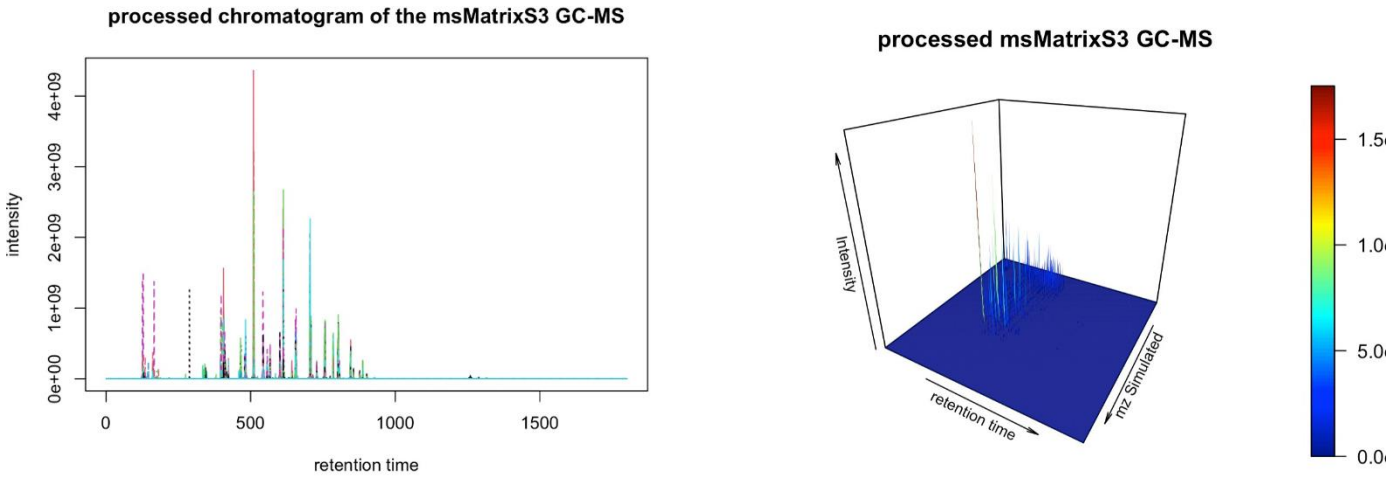

Figura 33. Cromatograma en 2D i 3D de les dades reals després de reduir el soroll.

Amb l'ús de la transformada de Wavelet i els seus coeficients i escales podem identificar pics al llarg del cromatograma, on agafem els índexs juntament amb una petita finestra de cinc valors a cada costat, i es redueix el valor de les intensitats que no corresponen a les posicions dels pics identificats ni els seus voltants.

A les figures 32 i 33, els colors observants pertanyen a cada m/z de la matriu emprada, les m/z nominals per aquest cas són 581, per la qual cosa, encara que emprem el paquet de colors més gran, no aconseguim acolorir tots els senyals de diferents colors, a més a més, a causa del gran nombre de m/z tampoc hem afegit una llegenda. En cas que vulguem saber les ones observades als gràfics anteriors, haurem de fer una cerca de m/z.

A la dreta trobem la representació de les dades en 3D, on alhora veiem la barra de colors per veure la variabilitat d'intensitat dels pics.

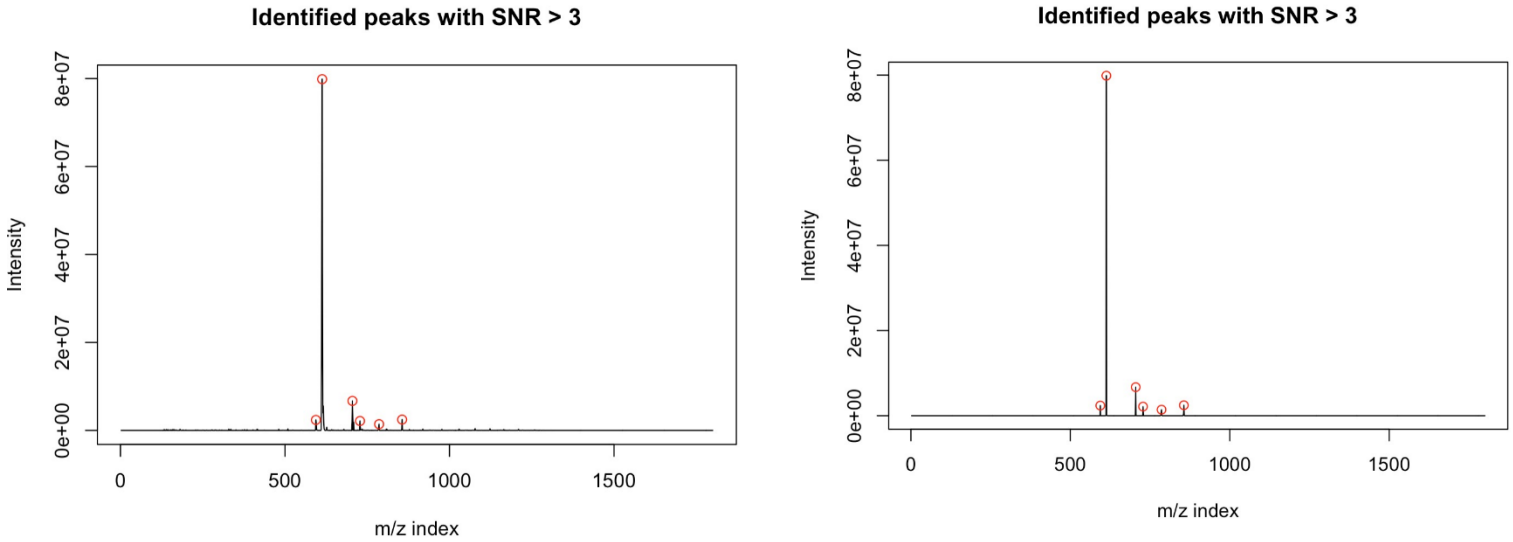

Figura 34. Identificació dels pics amb l'ajuda de les funcions del paquet MassSpecWavelet a l'esquerra. Cromatograma amb soroll reduït a la dreta. En comptes d'índex m/z vam emprar temps de retenció.

Podem observar que l'etiqueta de l'eix x continua sent el mateix amb el qual la funció funciona, però pel nostre cas en comptes d'índex m/z és temps de retenció.

Procedim a la detecció de pics pels ions de la creatina:

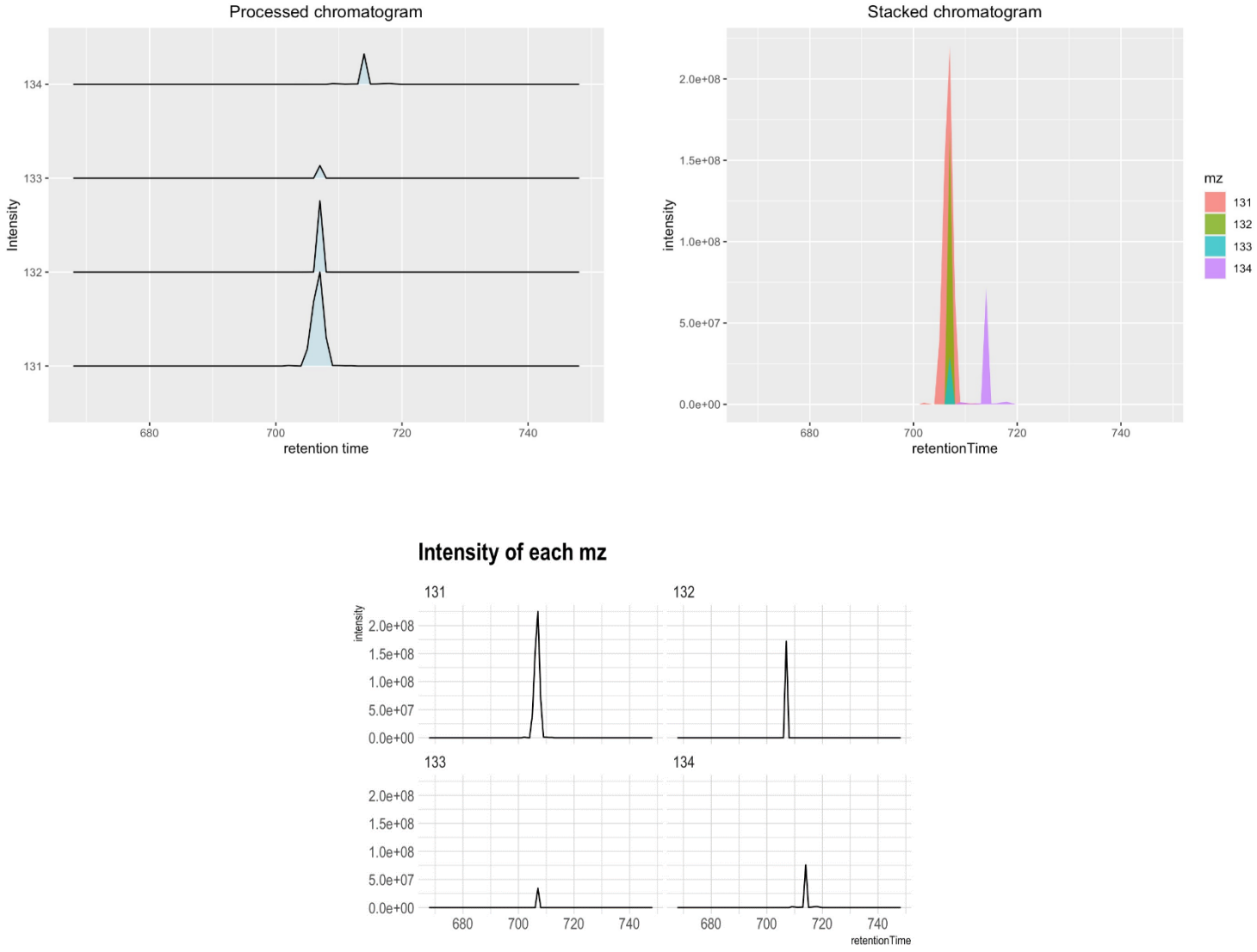

Figura 35. Gràfics desats en la sortida com a una llista amb la funció "chromatogramFunction", els quals són "intensitiesMzs", "intensitiesStacked" i "plotMzsSeparately", respectivament pel cas de dades reals.

Podem observar pics identificats dins el rang de retenció de temps i m/z corresponents a la creatina dins les dades reals. On el primer ió (de m/z 131) presenta una corba amb major intensitat i, pel cas de l'últim, el pic es troba desplaçat en el temps en cas que correspongui realment a un ió de la creatina, en cas contrari seria un *false positive* (FP).

Els valors dels gràfics 131 a 134 fan referència a les m/z amb senyal per aquell temps de retenció (703-713) del cromatograma.

5.3.2 Flux de treball emprat de l'algoritme creat per la simulació

Per aquest cas sí que necessitarem l'ús de la funció de correcció de la *baseline* i la resta del flux de treball a seguir serà el mateix que per les dades reals.

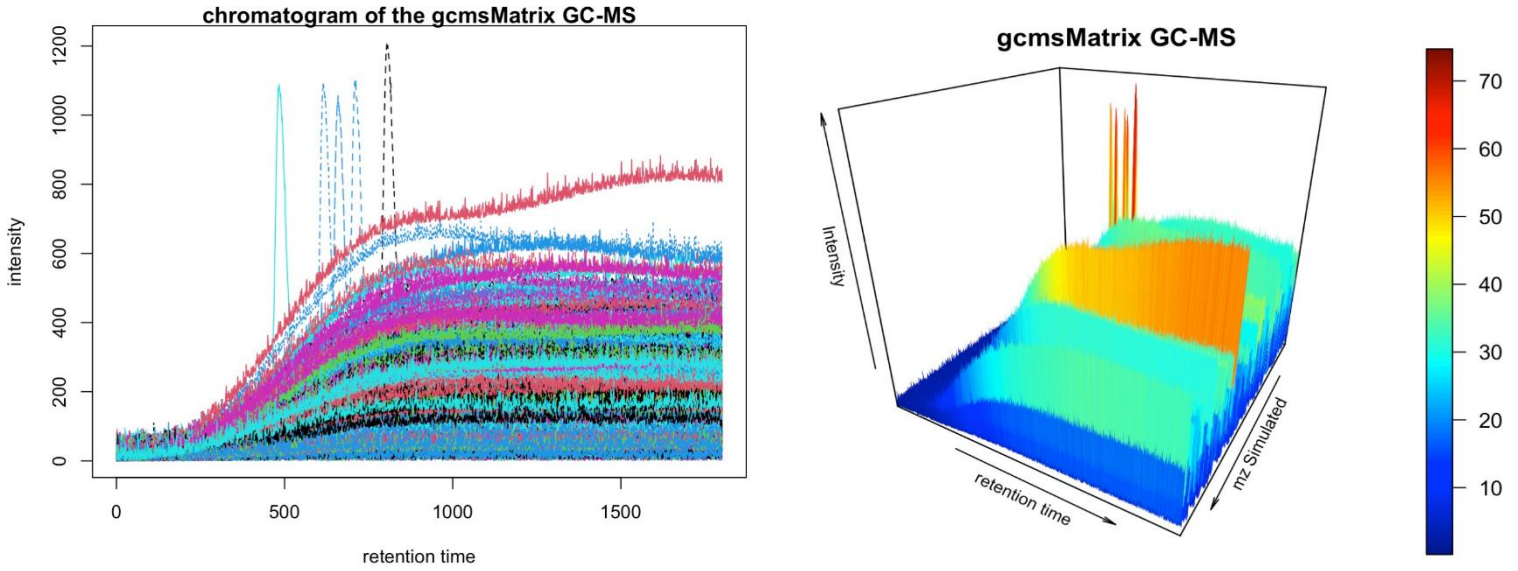

Figura 36. Cromatograma en 2D i 3D de les dades simulades.

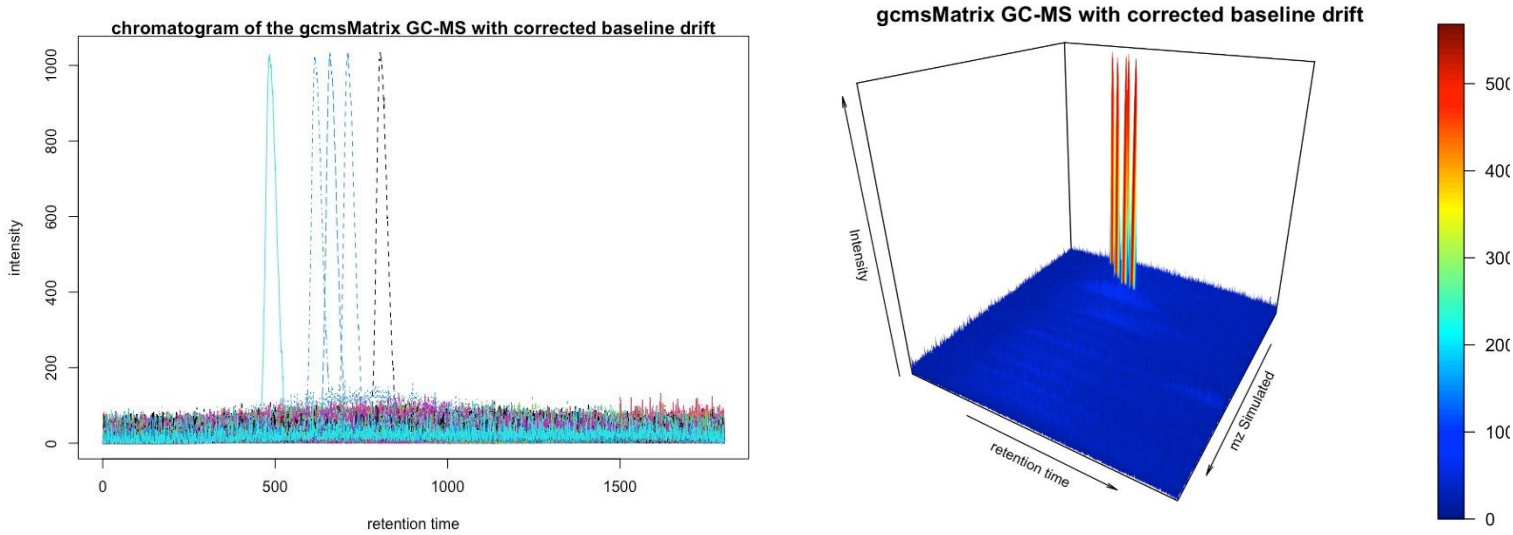

Figura 37. Cromatograma en 2D i 3D de les dades simulades després de la correcció de la *baseline drift*.

Podem observar el senyal de les ones amb pics d'intensitats més alts pels ions dels metabòlits ficats.

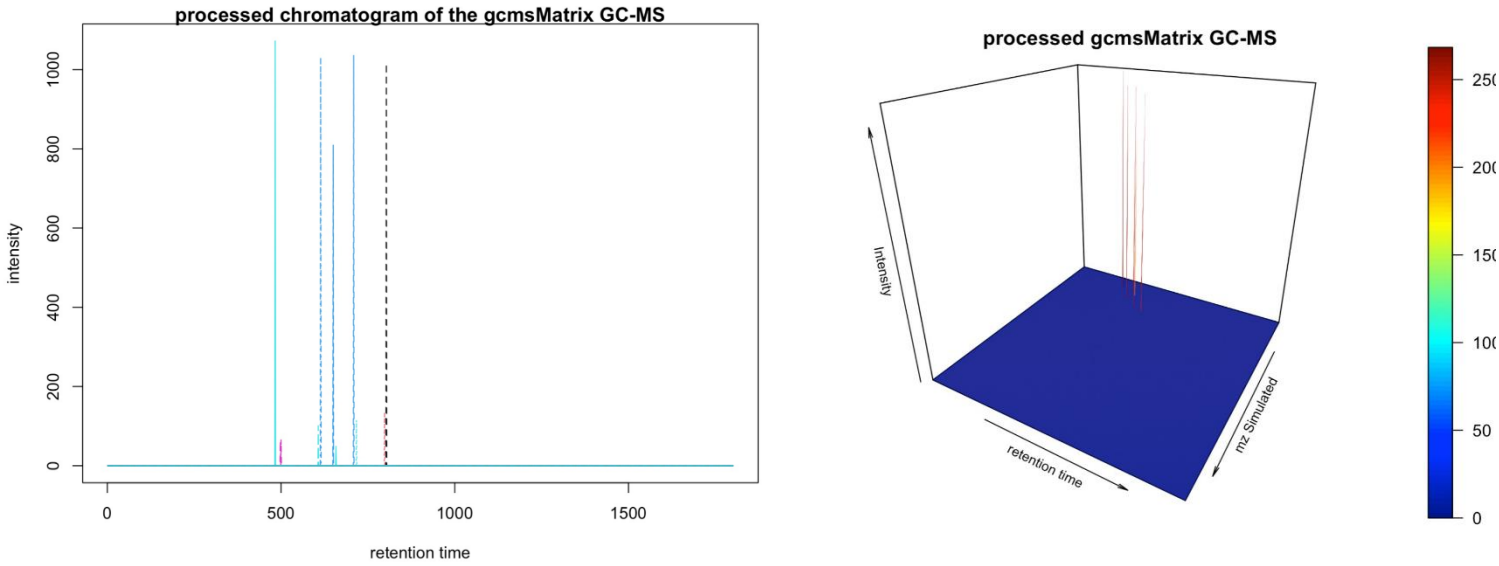

Figura 38. Cromatograma en 2D i 3D de les dades simulades després de reduir el soroll.

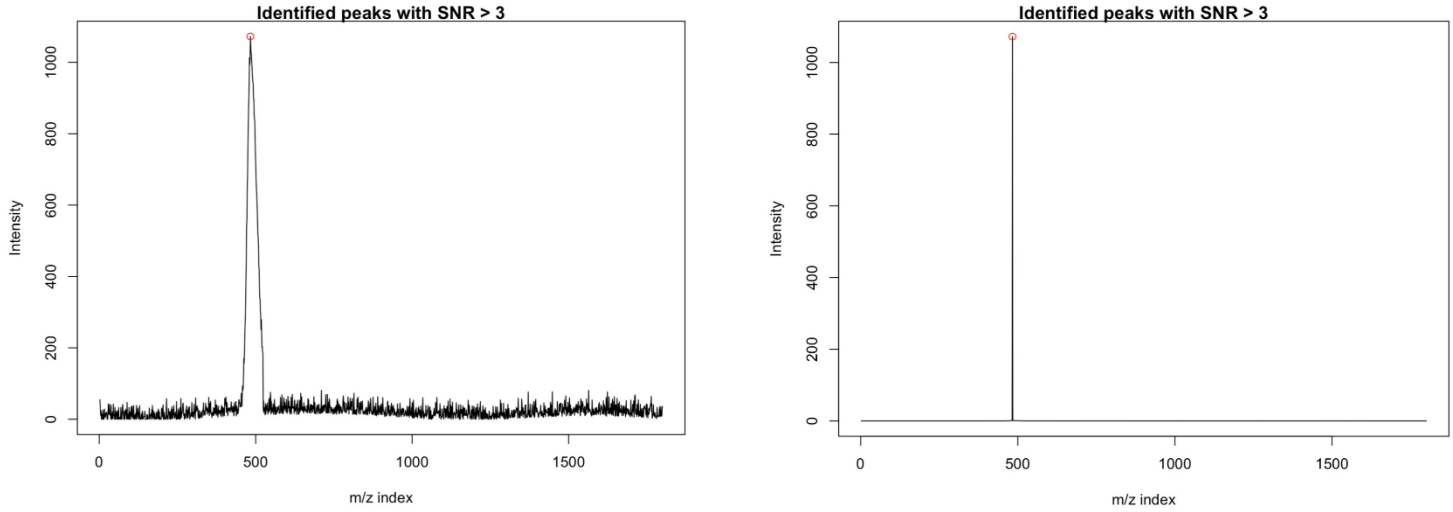

Figura 39. Identificació dels pics amb l'ajuda de les funcions del paquet MassSpecWavelet a l'esquerra. Cromatograma amb soroll reduït a la dreta. En comptes d'índex m/z vam emprar temps de retenció.

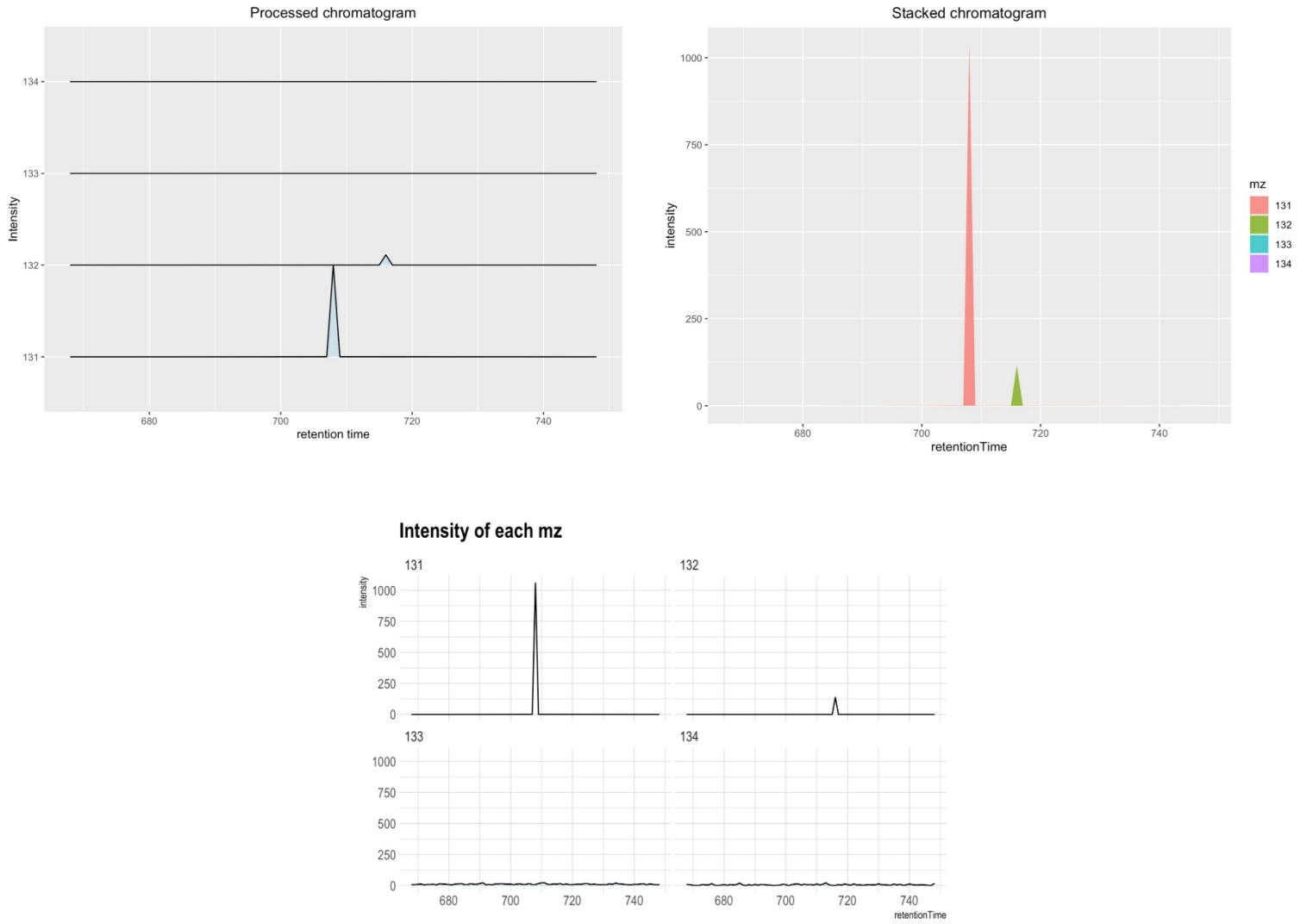

Figura 40. Gràfics desats en la sortida com a una llista amb la funció "chromatogramFunction", els quals són "intensitiesMzs", "intensitiesStacked" i "plotMzsSeparately", respectivament pel cas de la simulació.

Podem observar que pel cas de les m/z 133 i 134 a la tercera imatge veiem un espectre molt baix en el qual no es poden distingir cap pic, i als dos de sobre no es distingeix cap dada.

Pel cas de l'ió amb m/z 131, observem el pic de major intensitat i, per la posició als gràfics, correspon a un rang de temps pròxim al 708, propi de la creatina.

5.3.3 Discussió

Respecte al cromatograma total d'ions observem que la simulació dona un recorregut molt més net i que sols s'observa una mostra, i pel cas real observen diverses mostres i més dades, com els alquens, estàndards de qualitat ("std") o blancs. Podem afegir més dades a la simulació per acabar obtenint un gràfic semblant.

Tant al cas real com a la simulació veiem un gran canvi en la visualització del TIC en 2D i 3D amb el tractament del soroll, observant amb més claredat els pics escollits com a possibles ions de metabòlits.

Concretament pel cas de la simulació, podem observar la diferència amb la correcció de la *baseline*, on els pics més abundants són més notoris una vegada s'ha esbiaixat la resta de dades, el cas real no presenta aquesta desviació, per tant, no cal emprar aquesta correcció.

Finalment, a l'hora de cercar més concretament les dades que volem, amb una finestra de temps i m/z en l'espectre, podem veure diferències i semblances en els resultats obtinguts amb les dades reals i simulades.

La primera gran diferència és que pel cas de les dades simulades s'ha perdut informació entre el soroll i la *baseline* però pel cas de les reals no, com bé ho podem observar en comparar els gràfics de les figures 35 i 40. Això ho sabem ja que tota la informació que hi ha a les dades simulades ha de presentar un senyal a l'hora de graficar-ho, ja que per defecte tots els ions presenten corbes al cromatograma i, en observar la figura 40 hi ha m/z que no presenten senyal. A més a més, tenim que potser les dades amb una intensitat més petita no presentaran una corba prou notable als dos primers gràfics, de la figura 40, però al tercer sempre s'acaba podent visualitzar la corba pròpia dels ions, i pel cas de la simulació no s'observa senyal per les m/z 133 i 134.

A banda d'això, trobem que tant a la simulació com amb les dades reals el pic de major intensitat correspon al mateix ió, al de la m/z 131. També podem observar que tant als gràfics de les figures 34 com als de la 39 hi ha pics desplaçats, podem pensar que s'han desplaçat amb l'ús de funcions o pel cas de les dades reals a causa de factors de maquinària, però pel nostre cas, el desplaçament de pics a causa de funcions no pot ser, ja que no hi ha cap pas que pugui provocar aquest efecte, per la qual cosa, pel cas de la simulació, encara que hàgim emprat una finestra, el valor de pic de la m/z 132 podria ser soroll que no s'ha filtrat bé i presenta un FP (*false positive*), en canvi, pel cas de les dades reals no podem saber si és un TP (*true positive*) o un FP.

Cal afegir que no es pot fer una visualització final en 3D, ja que en fer-ho, encara que les dades s'hagin manipulat específicament per poder veure's bé, la funció crea unes corbes entre el temps de retenció en comptes de les m/z, així obtenint diferents corbes en comptes de només, per aquest cas, quatre corbes, una corba per cada m/z indicant les intensitats obtingudes al llarg del temps, el gràfic característic d'un cromatograma en 3D.

Per concloure podem dir que pel cas de les dades reals l'algoritme ha sigut capaç de detectar els pics cercats per la creatina, podent observar intensitats dins el marge de les m/z nominals, en canvi, pel cas de la simulació trobem que s'ha perdut la informació d'aquelles intensitats més petites, com bé podem apreciar a la taula 1 que els valors són molt petits per aquest cas.

6 Discussió

Al llarg del punt cinc hem discutit els resultats obtinguts en cada algoritme per la detecció de pics entre les dades reals i les dades simulades, ja que volíem apreciar la qualitat de detecció de dades en un entorn controlat (simulades) i en un no controlat (reals). On alhora en fer la comparativa podem també veure el recorregut de les dades simulades i cap a on pot arribar.

Més concretament volem estudiar globalment tots tres tipus d'algorismes de detecció de pics pels dos casos.

Podem visualitzar els passos en comú de tots tres algorismes:

Algorisme detecció de pics\funcions	TIC inicial (*)	Reducció de soroll (*) i de *baseline* (*)	Alineació (*) i correspondència (*)	Pics en finestra de temps i m/z (*)	Detecció de pics (*)
XCMS	*	**	**	*	*
GCalignR			*		*
Algoritme creat	*	**		*	*

Taula 2. Funcions realitzades pels algorismes compartides entre ells.

A banda, també trobem que pels dos primers algorismes podem trobar informació extra a la taula dos, per exemple, amb XCMS podem obtenir els *features* de les dades i continuar amb un estudi exhaustiu, com la seva classificació en grups i estudi del seu comportament amb diferents classificacions d'acord amb les seves dades estadístiques i, amb GCalignR podem saber el nombre de substàncies o *heatmap*, per exemple. Pel cas de l'algorisme creat, en haver sigut creat principalment i només per la detecció de pics, no va més enllà i no hi ha més funcions a emprar si es volgués continuar estudiant les dades amb funcions externes o ja implementades en R.

A banda, en tots tres trobem desavinences al llarg de l'execució del seu flux de treball, les quals són:

Algorisme detecció de pics\dades	Reals
XCMS	Flux de treball: no hi havia dades amb findChromPeaks() per la informació ja reduïda als factors de cerca (m/z i rt).
GCalignR	Flux de treball: pics trobats incoherents i nombre de substàncies en comú entre les mostres incoherent.
Algoritme creat	Flux de treball: gràfic 3D dels pics detectats amb surfacePlot() incorrecte. Alguns gràfics 3D els nombres del rang de colors a vegades no són visibles completament pel cas d'emprar R script.

Taula 3. Resultats desfavorables pels algorismes estudiats per la detecció de pics per cas real.

Algorisme detecció de pics\dades	Simulades
XCMS	Flux de treball: no es pot emprar la funció pickPeaks() i tampoc podem emprar findChromPeaks() ni continuar amb el flux de treball (alineació i correspondència). Dades molt nítides.
GCalignR	Flux de treball: és necessària una correcció de la *baseline* pel tipus de dades que empra aquest algoritme, en cas que les dades presentin aquest factor. Valors de pics detectats no desitjats. El tipus de dades a emprar és el resultant d'una visualització TIC, on les mostres es troben sumades al llarg del temps, dificultant la separació i individualitat de les mostres de problemes com el soroll o pics característics d'uns tipus de mostra i no d'altres.
Algoritme creat	Flux de treball: gràfic 3D per la finestra de temps i m/z incorrecte. Rang de colors no visible completament en alguns casos pel cas de visualitzacions 3D en R script.

Taula 4. Resultats desfavorables pels algorismes estudiats per la detecció de pics per cas simulat.

Podem apreciar les desavinences de totes tres, on el que més dificulta la detecció de pics és el problema amb GCalignR, ja que les dades obtingudes com pics detectats no és coherent amb el cromatograma inserit, ni tampoc les dades examinades amb aquest mètode, tant per les dades reals com per les simulades, perquè com bé hem esmentat al punt cinc, és incoherent que s'obtinguin els mateixos resultats per les dades reals com per les dades simulades, ja que per la simulació hi ha sols cinc metabòlits i per les reals ha d'haver-hi una gran varietat de metabòlits, com els que podem veure amb el document de mostra de metabòlits d'orina per Biosfer Teslab. Per altra banda, trobem que per les dades exposades al seu document sí que podem trobar i detectar pics i la seva informació, però en manipular-lo amb altres dades no aconseguim resultats congruents.

Aquest mètode presentava avantatges per la seva facilitat d'ús i pels seus resultats, com el nombre de substàncies registrades, l'alineació, la visualització d'un *heatmap* o la variació entre les mostres a banda dels pics detectats. Per la qual cosa, aquest mètode queda descartat per fer anàlisis per detecció de pics per les dades emprades per les problemàtiques trobades en el flux de treball.

Per altra banda, com bé hem pogut observar a la taula dos, podem detectar els pics d'un cromatograma amb tots tres, però d'ells, GCalignR ha presentat problemes amb la seva resolució final, donant-nos pics que no esperàvem, en canvi, amb XCMS i amb l'algoritme creat podem fer finestres en les dades de temps i m/z per acabar de visualitzar millor les dades que busquem, no sols això sinó que també podem refinar les dades amb la correcció de soroll i/o de *baseline* juntament, en cas que es pugui, amb l'alineació i correspondència, per acabar de trobar tota la informació pertinent als ions que estem cercant.

Per aquestes raons, els fluxos d'XCMS i l'algoritme creat es poden emprar per a la detecció de pics, tant per les dades simulades com per les reals, ja que amb ambdós mètodes

podem detectar i cercar els pics que vulguem per poder fer la seva anàlisi. On cadascun presenta les seves característiques de treball com bé podem observar tant al codi emprat, com als fluxos de treball com als gràfics obtinguts. Fent així que es pugui escollir un algoritme o un altre per executar-los per realitzar uns estudis o uns altres.

A continuació omplim unes taules per les característiques d'ambdós que no es tenen en comú (com la correcció de soroll i de *baseline)* pels resultats viables obtinguts al llarg dels fluxos de treball fets al llarg del document i codi.

funcions\algorisme detecció de pics	XCMS
XIC	Visualització de totes les mostres que continguin dades en el rang de m/z i temps. Es mostra un gràfic per la intensitat i un altre per les m/z al llarg del temps per cada mostra.
Alignment	Alineació de pics propers per les mostres.
Correspondència	Representar en el cromatograma la distribució de densitat de pics i els pics agrupats (*features*).

Taula 5. Funcions característiques del flux de treball per la detecció de pics d'XCMS.

funcions\algorisme detecció de pics	Algoritme creat
Cromatograma 3D	Visualització del cromatograma en 3D
chromatogramFunction()	Obtenció de les ones per les finestres de temps i m/z en gràfics específics per mostrar el seu comportament ("intensitiesMzs", "intensitiesStacked" i "plotMzsSeparately").
treatedMetabolite()	Es pot reduir les dades fora de la finestra de temps i m/z a 0 per ajudar en visualitzacions i tractaments específics posteriors.

Taula 6. Funcions característiques del flux de treball per la detecció de pics per l'algorisme creat.

Es va fer l'estudi amb dades reals i simulades per poder comparar els resultats i així poder ajustar millor la simulació d'acord amb el que s'espera obtenir per dades reals i fer més fidel la simulació. On un dels aspectes a més ressaltar va ser la manca d'informació per la simulació, ja que encara que tingués informació al llarg de tot el temps de retenció i m/z, aquestes eren majoritàriament omplerts amb dades que seran negligides, soroll i *baseline*, i no dades amb informació, com els metabòlits, per la qual cosa, s'espera afegir molta més informació significant a la simulació.

D'altra banda, la simulació creada encara necessita que se li afegeixi dades d'alquens, blancs i estàndards de qualitat ("std") per acabar tenint tota la informació relativa a la sortida d'un GC/MS. Es va intentar realitzar diverses prediccions, però a causa del gran nombre de dades el temps de processat i els resultats no van ser els esperats. S'hauria d'emprar una xarxa neuronal creada específicament per aquests tipus de dades per poder acabar predient les sortides de mostres com pels alquens o std per a un tipus de mostra de metabòlits específics o també, es pot acabar fent una altra llibreria per aquests casos, els quals es crearia una funció relacionada amb aquests tres tipus de dades juntament amb els metabòlits, si és necessari, per acabar obtenint les dades que estiguin relacionades si ho han d'estar.

Fora d'això, es pot modificar el tipus d'ona dels ions, on per defecte està l'asimètrica gaussiana. Però per cas d'intentar simular la sortida per uns aparells i mostres específiques, s'hauria d'estudiar les variants a les quals estan sotmeses les ones i ajustar la curvatura dels metabòlits d'acord amb aquestes propietats, així recreant una mostra més fidel a la que hauríem d'esperar per un cas real.

Finalment, la simulació hauria de ser prou funcional per poder acabar un flux de treball exhaustiu d'anàlisis de dades per acabar arribant a la seva major finalitat i ús. Ja que la finalitat de la simulació serà l'ús d'aquesta com mitjà intermediari per l'estudi de característiques de mostres i el seu comportament, tant entre els metabòlits com la maquinària emprada.

Podem observar que la simulació presenta problemes en l'ús d'XCMS i no es pot sotmetre a una alineació o correspondència. Aquest mètode de treball (XCMS) trobem que s'empra bastant pel nombre de funcions específiques per l'estudi i anàlisi de dades cromatogràfiques, per la qual cosa, aconseguir completar aquest treball de flux amb la simulació ajudaria a validar-la com a conjunt de dades que pot ser sotmès a un estudi exhaustiu d'anàlisis per la informació de senyals de sortida d'un GC/MS.

A la fi, volem que aquesta simulació sigui independent d'altres factors i que per ella sola sigui capaç d'assolir els mateixos resultats que si es tractés d'informació real, així poder manipular la informació de la simulació per poder veure les variants que poden aportar l'*input* donat, ja sigui un cert nivell d'ions de metabòlits amb els alquens, la interacció, l'abundància i les distorsions a les quals les dades reals es poden veure sotmeses sense necessitat de passar per un laboratori amb un GC/MS ni recollida de mostres. On per ara, encara necessita alguns ajustaments, com l'addició de més informació, ja sigui de metabòlits com d'alquens, blancs i std, juntament amb un estudi del perquè no s'acaba de poder acabar el flux de treball d'XCMS, juntament amb la recreació de més mètodes d'estudi de dades cromatogràfiques matricials.

7 Conclusions i línies futures

Al llarg d'aquest document hem pogut observar les diverses variacions que es poden obtenir en el senyal de sortida d'un cromatograma per GC/MS, com el soroll, la *baseline drift*, la pressió d'entrada, les interaccions a la columna, etc. Aquestes s'han tingut en compte al implementar un algoritme capaç d'adaptar-se a les demandes per modificar les distorsions i dades dels metabòlits i així ser capaç de recrear "in silico" el senyal de sortida d'un GC/MS de metabòlits reals.

Amb l'estudi d'algoritmes per la detecció de pics s'han recreat els fluxos de treballs adients als algoritmes estudiats (XCMS, GCalignR i algoritme creat), dels quals, en tractar les dades reals i simulades per aquests mètodes, hem aconseguit visualitzar i detectar els pics solament pels casos d'XCMS i l'algoritme creat.

També trobem que l'algoritme creat, en haver sigut especialment produït per la detecció de pics, és més senzill d'emprar que XCMS i podem obtenir diferents matrius i gràfics especialment elaborats per visualitzar o manipular les dades per obtenir la informació dels pics del cromatograma sigui més visual o més numèric. A més, les dades numèriques es poden emprar per posteriors estudis, ja que tant es troben les dades en la finestra de temps i m/z com pel llarg de totes les dades. En canvi, XCMS és molt més específic pel que fa a l'anàlisi de dades amb la finalitat de seguir un flux de treball propi d'estudis de perfils metabòlics. Com bé hem pogut visualitzar, tots tres programes presenten les seves característiques en anàlisi i detecció, per les quals, el tipus de dades emprades, podem optar entre una anàlisi per la detecció de pics amb XCMS o l'algoritme creat, depenent dels gràfics que es vulgui aconseguir

i/o dades. Respecte al moment, tant amb les simulacions com les dades reals es poden seguir aquests dos fluxos.

L'algoritme creat incorpora funcions independents dels mètodes XCMS i GCalignR, on es poden agafar dades matricials cromatogràfiques i sotmetre-les a l'estudi per la detecció del pic, fent que sigui independent del format i sols es necessiti que sigui una matriu amb les intensitats de cada m/z al llarg del temps de retenció.

Per futurs treballs proposem els següents aspectes per noves versions dels algoritmes implementats:

- Es pot acabar d'omplir la simulació amb les dades d'alquens, blancs i estàndards de qualitat ("std"), una bona manera seria trobar un tipus de predicció adient amb la quantitat d'informació que es gestiona, juntament amb la implementació de funcions que puguin relacionar la informació de llibreries que continguin aquests tres aspectes i les sàpiguen lligar a una matriu cromatogràfica, ja amb les pertorbacions i informació de metabòlits feta.
- Acabar de sotmetre a un estudi exhaustiu la simulació i veure que funciona per tots els passos del flux de treball per casos reals de GC/MS.
- Perfeccionar els problemes obtinguts vists a la taula tres i quatre.
- Incorporar cromatogrames reals amb *alkanes* al creat "in silico" i intentar incorporar metabòlits de llibreries amb *Retention Index* (RI) basat en *alkanes* o RT (*retention time*) sense basar-se en els *alkanes*.

Enllaços d'interès

Repositori del codi del projecte:

https://drive.google.com/drive/folders/1gkErQvJU_7DyH1O32qdIwfB6X_MoQg92?usp=drive_link

Tots els diagrames han sigut creats amb les eines de draw.io website:

https://app.diagrams.net/

I totes les altres figures que no han sigut citades han sigut obtingudes pel codi emprat.

Referències

[1] "adduct traduction", *diccionario.reverso*. Recuperat el 6 de septembre de 2023, website: https://diccionario.reverso.net/ingles-espanol/adduct

[2] "Biomolècules II", *studocu*. Recuperat el 5 de septembre de 2023, website: https://www.studocu.com/ca-es/document/universitat-pompeu-fabra/biologia-celular/biologia-pau-pau/39998275.

[3] "HMDB", *hmdb*. Recuperat el 5 de septembre de 2023, website: https://hmdb.ca/unearth/q?utf8=%E2%9C%93&query=alanine&searcher=metabolites&button=.

[4] "Golm metabolome database", *gmd*. Recuperat el 5 de septembre de 2023, website: http://gmd.mpimp-golm.mpg.de/search.aspx#&&query=alanine.

[5] "Mass Bank", *massbank.eu*. Recuperat el 5 de septembre de 2023, website: https://massbank.eu/MassBank/Result.jsp?compound=alanine&op1=and&mz=&tol=0.3&op2=and&formula=&type=quick&searchType=keyword&sortKey=not&sortAction=1&pageNo=1&exec=&inst_grp=ESI&inst=CE-ESI-TOF&inst=ESI-ITFT&inst=ESI-ITTOF&inst=ESI-QIT&inst=ESI-QQ&inst=ESI-QTOF&inst=ESI-TOF&inst=LC-ESI-FT&inst=LC-ESI-IT&inst=LC-ESI-ITFT&inst=LC-ESI-ITTOF&inst=LC-ESI-Q&inst=LC-ESI-QFT&inst=LC-ESI-QIT&inst=LC-ESI-QQ&inst=LC-ESI-QQQ&inst=LC-ESI-QTOF&inst=LC-ESI-TOF&ms=MS2&ion=0.

[6] "Metabolito", *Quimica.es*. Recuperat el 28 d'agost el 2023, website: https://www.quimica.es/enciclopedia/Metabolito.html.

[7] Amat Rodrigo, Joaquín, "Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE", *Ciencia de datos*. Recuperat el 28 d'agost de 2023, website: https://cienciadedatos.net/documentos/35_principal_component_analysis, 2017.

[8] Caihong Bai, Suyun Xu, Jingyi Tang, Yuxi Zhang, Jiahui Yang, Kaifeng Hu, "A 'shape-orientated' algorithm employing an adapted Marr wavelet and shape matching index improves the performance of continuous wavelet transform for chromatographic peak detection and quantification", *Journal of Chromatography A*, Volume 1673, 2022, 463086, ISSN 0021-9673, https://doi.org/10.1016/j.chroma.2022.463086.

[9] Biostatsquid, "Volcano plots in R: easy step-by-step tutorial", *biostatsquid*. Recuperat el 28 d'agost el 2023, website: https://biostatsquid.com/volcano-plots-r-tutorial/, 2023.

[10] M. Jones 2019, "Gas Chromatographi-Mass Spectrometry", *ACS Chemistry for Life*. Recuperat el 12 d'agost de 2023, website: https://www.acs.org/education/whatischemistry/landmarks/gas-chromatography-mass-spectrometry.html.

[11] D. Turner, "Principio, instrumento y análisis de GC-MS y GC-MS/MS", *News Courier*. Recuperat el 14 d'agost de 2023, website: https://www.news-courier.com/analysis/articles/gc-ms-principle-instrument-and-analyses-and-gc-msms-362513#D1.

[12] T. Lindquist, "Fully automatic algorithm for analysis of gas cromatography data", 2018. Recuperat el 14 d'agost de 2023, de website: https://www.diva-portal.org/smash/get/diva2:1236185/FULLTEXT01.pdf.

[13] "The R Project for statistical computing", R. Recuperat el 12 d'agost de 2023, website: https://www.r-project.org/.

[14] "Cromatografía de gases/masas (GC-MS)", UCM. Recuperat el 9 d'agost de 2023, website: https://www.ucm.es/data/cont/docs/650-2013-12-02-gases%20masas.pdf.

[15] "Cromatografía de gases/masas (GC-MS)", UCM. Recuperat el 9 d'agost de 2023, website: https://cai.ucm.es/ciencias-tierra-arqueometria/tecnicas-geologicas/tecnicas/cromatografia-de-gases-masas-gc-ms/38/.

[16] P. M. A. Harvey and R. A. Shellie, "Factors affecting peak shape in comprehensive two-dimensional gas chromatography with non-focusing modulation," *J Chromatogr A*, vol. 1218, no. 21, pp. 3153–3158, May 2011, doi: 10.1016/J.CHROMA.2010.08.029.

[17] C. E. Freye, H. D. Bahaghighat, and R. E. Synovec, "Comprehensive two-dimensional gas chromatography using partial modulation via a pulsed flow valve with a short modulation period," *Talanta*, vol. 177, pp. 142–149, Jan. 2018, doi: 10.1016/J.TALANTA.2017.08.095.

[18] P. Moretti, S. Vezzani, E. Garrone, and G. Castello, "Evaluation and prediction of the shape of gas chromatographic peaks," *J Chromatogr A*, vol. 1038, no. 1–2, 2004, doi: 10.1016/j.chroma.2004.03.039.

[19] S. Vezzani, P. Moretti, R. Peri, and G. Castello, "Effect of injected sample amount on the shape of chromatographic peaks under condition of linear partition isotherm," *J Chromatogr A*, vol. 1065, no. 2, pp. 231–237, Feb. 2005, doi: 10.1016/J.CHROMA.2004.12.048.

[20] Elimelech. Grushka, M. N. Myers, P. D. Schettler, and J. Calvin. Giddings, "Computer characterization of chromatographic peaks by plate height and higher central moments," *Anal Chem*, vol. 41, no. 7, pp. 889–892, May 2002, doi: 10.1021/ac60276a014.

[21] Wells, Gregory. and Harry Frank Prest. "Signal, Noise, and Detection Limits in Mass Spectrometry.", 2011.

[22] "5 Ways to Approeach Baseline Issues" GC, MDGC, chromatography Today, 2014.

[23] A. N. Edwards, H. M. Tran, and E. S. Gallagher, "Propagating Error through Traveling-Wave Ion Mobility Calibration," *J Am Soc Mass Spectrom*, vol. 32, no. 11, pp. 2621–2630, Oct. 2021, doi: 10.1021/jasms.1c00144.

[24] M. Loos, et al., "enviPatWeb". Recuperat el 15 d'agost de 2023, website: https://www.envipat.eawag.ch/, 2013.

[25] "BiosferTeslab". Recuperat el 15 d'agost de 2023, website: https://biosferteslab.com/wp-content/uploads/2020/05/ES_ListadoOrina.pdf.

[26] J. Rainer, "Metabolomics data pre-processing using xcms", Eurac Research, Bolzano, Italy, 2018.

[27] M. Ottensmann, M. A. Stoffel, H. J. Nichols and J. I. Hoffman, "GCalignR Step by Step".

[28] Borgsmüller, Nico, Yoann Gloaguen, Tobias Opialla, Eric Blanc, Emilie Sicard, Anne-Lise Royer, Bruno Le Bizec, Stéphanie Durand, Carole Migné, Mélanie Pétéra, and et al. 2019. "WiPP: Workflow for Improved Peak Picking for Gas Chromatography-Mass Spectrometry (GC-MS) Data" *Metabolites* 9, no. 9: 171. https://doi.org/10.3390/metabo9090171

[29] Kim, Seongho et al. "A NEW METHOD OF PEAK DETECTION FOR ANALYSIS OF COMPREHENSIVE TWO-DIMENSIONAL GAS CHROMATOGRAPHY MASS SPECTROMETRY DATA." *The annals of applied statistics* vol. 8,2 (2014): 1209-1231. doi:10.1214/14-aoas731

[30] O. D. Myers, S. J. Sumner, S. Li, S. Barnes, and X. Du, "Detailed Investigation and Comparison of the XCMS and MZmine 2 Chromatogram Construction and Chromatographic Peak Detection Methods for Preprocessing Mass Spectrometry Metabolomics Data," *Anal Chem*, vol. 89, no. 17, pp. 8689–8695, Aug. 2017, doi: 10.1021/acs.analchem.7b01069.

[31] Freire, Rafael, Luis Fernandez, Celia Mallafré-Muro, Andrés Martín-Gómez, Francisco Madrid-Gambin, Luciana Oliveira, Antonio Pardo, Lourdes Arce, and Santiago Marco. 2021. "Full Workflows for the Analysis of Gas Chromatography—Ion Mobility Spectrometry in Foodomics: Application to the Analysis of Iberian Ham Aroma" *Sensors* 21, no. 18: 6156. https://doi.org/10.3390/s21186156

[32] T. Lindquist, 'Fully automatic algorithm for analysis of gas chromatography data', Dissertation, 2018.

[33] "Spectral Centroiding and Calibration", Prappleizer, github.

[34] "Savitzky-Golay smoothing method", *NIRPY research*,. Recuperat el 25 d'agost de 2023, website: https://nirpyresearch.com/savitzky-golay-smoothing-method/, 2019.

[35] "Data classification: quantile and equal interval", *NCGIA*. Recuperat el 25 d'agost de 2023, website: http://www.ncgia.ucsb.edu/cctp/units/unit47/html/comp_class.html.

[36] A. Barone, "Residual Sum of Squares (RSS): What It Is, How to Calculate It", *Investopedia*. Recuperat el 25 d'agost de 2023, website: https://www.investopedia.com/terms/r/residual-sum-of-squares.asp, 2022.

[37] S. Taylor, "R-Squared". *CFI.* Recuperat el 25 d'agost de 2023, website: https://corporatefinanceinstitute.com/resources/data-science/r-squared/.

[38] "Quantile Regression", *IBM*. Recuperat el 25 d'agost de 2023, website: https://www.ibm.com/docs/en/spss-statistics/saas?topic=regression-quantile, 2023.

[39] "Tuples", *hetpro*. Recuperat el 6 de septembre de 2023, website: https://hetpro-store.com/TUTORIALES/tuples-en-python-6-colecciones/.

[40] revathy, "*Blank Run*", community.agilent. Recuperat el 6 de septembre de 2023, website: https://community.agilent.com/technical/gc/f/forum/311/blank-run.

[41] Wirsing, Kariton. "Time Frequency Analysis of Wavelet and Fourier Transform*"*, *intechopen*. Recuperat el 27 d'agost de 2023, website: https://www.intechopen.com/chapters/74096.

Annexa

Codi creat i utilitzat per aquest treball de fi de grau:

```
#####################################################################
#####################################################
#
#    Evaluation of different gas chromatography peak detection algorithms with
#    simulated urine GC-MS signals using R
#
#    Version 2.0.  2023/08/31
#
#    @Natzaret Gálvez Rísquez
#
#####################################################################
#####################################################




#####################################################################
#####################################################
# Set working directory
#####################################################################
#####################################################


setwd("~/Desktop")
```

```
#####################################################################
#########################################################
# Libraries
#####################################################################
#########################################################

library(signal)
require(plot3D)
library(xcms)
library(RColorBrewer)
library(magrittr)
library(DT)
library(GCalignR)
library(MassSpecWavelet)


#####################################################################
#########################################################
# Functions
#####################################################################
#########################################################
#####################################################################
#########################################################
#####################################################################
#########################################################


#####################################################################
#########################################################
# Function to create metabolites information in a dataframe
#####################################################################
#########################################################


createMetabolite <- function(metaboliteOriginal){
  #we erase the first row
  head(metaboliteOriginal, 1)
  metabolite <- metaboliteOriginal[2:nrow(metaboliteOriginal), ]


  #in order to see the wave form first we need to extract the m/z and the intensity values
  mzAndIntensityValuesMetabolite <- metabolite[2:nrow(metabolite), 1:2]
  #then, we can add the names
  #rownames(mzAndIntensityValuesMetabolite) <- t(metabolite[1, 3:ncol(metabolite)])
  colnames(mzAndIntensityValuesMetabolite) <- c("mz", "intensity")


  return(mzAndIntensityValuesMetabolite)
}
```

```
#####################################################################
#####################################################
# Function to do a nominal m/z matrix
#####################################################################
#####################################################


#we pass the m/zs to nominal, also, with the repeated nominal m/zs we sum their intensities
mzNominal <- function (msSpectra){
  nominalMz <- unique(round(as.numeric(msSpectra[, 1])))

  nominalMatrix <- matrix(data=0, nrow=length(nominalMz), ncol=2)
  colnames(nominalMatrix) <- c("mz", "intensity")
  nominalMatrix[, 1] <- nominalMz
  var <- 1

  for (variable in 1:nrow(msSpectra)) {
    if (round(as.numeric(msSpectra[variable, 1]))!=nominalMatrix[var, 1]){
      var <- var+1
    }
    nominalMatrix[var, 2] <- as.numeric(nominalMatrix[var, 2])+as.numeric(msSpectra[variable, 2])
  }
  return(nominalMatrix)
}


#####################################################################
#####################################################
# Waveform Definitions
#####################################################################
#####################################################


#mean mu rt
#sd sigma


#gaussian function
gaussianFunction <- function (x, mu, sigma){
  (1/(sigma*sqrt(2*pi)))*exp(-.5*(x-mu)^2/sigma^2)
}


#lorentzian function
lorentzianFunction <- function (x, mu, sigma){
  (1/pi)*(sigma/((x-mu)^2+sigma^2))
```

```
}


##################################################################################################################

# Creation of a Wave

##################################################################################################################


#type, 1 for normal gaussian distribution, 2 for asymmetrical gaussian distribution

#3 for symmetrical lorentzian and 4 for asymmetrical, 5 for symmetrical gaussian and lorentzian and other values for asymmetrical

#sigma one for the normal gaussian/lorentzian and/or the left side of the asymmetrical, and the second sigma for the right side of the gaussian/lorentzian

createWave <- function(type, sigmaOne, sigmaTwo, vector, mu, intensity){

  if (type==1){

    #normal gaussian

    results <- gaussianFunction(vector, mu, sigmaOne)

    #we scale the results in order to have the desired intensity, we grab the value in the retention time where the mz will be

    value <- max(results)

    results <- (intensity/value)*results

  } else if (type==2) {

    #asymmetric gaussian

    leftSide <- gaussianFunction(vector, mu, sigmaOne)

    rigthSide <- gaussianFunction(vector, mu, sigmaTwo)


    #we adjust the values for the left side

    value <- max(leftSide)

    leftSide <- (intensity/value)*leftSide

    #and for the right side

    value <- max(rigthSide)

    rigthSide <- (intensity/value)*rigthSide


    #we grab only certain values

    #for the left side all the first numbers until the rt

    #for the right side, from the rt+1 position to the end

    #then, we add them into a vector

    leftSide <- leftSide[1:mu]

    rigthSide <- rigthSide[(mu+1):length(rigthSide)]


    results <- c(leftSide, rigthSide)

  } else if (type==3){ #symmetrical lorentzian

    results <- lorentzianFunction(vector, mu, sigmaOne)
```

```
    value <- max(results)
    results <- (intensity/value)*results
  } else if (type==4){ #asymmetrical lorentzian
    leftSide <- lorentzianFunction(vector, mu, sigmaOne)
    rigthSide <- lorentzianFunction(vector, mu, sigmaTwo)
    value <- max(leftSide)
    leftSide <- (intensity/value)*leftSide
    value <- max(rigthSide)
    rigthSide <- (intensity/value)*rigthSide
    leftSide <- leftSide[1:mu]
    rigthSide <- rigthSide[(mu+1):length(rigthSide)]
    results <- c(leftSide, rigthSide)
  } else if (type==5){ #symmetrical gaussian and lorentzian
    results <- intensity*gaussianFunction(vector, mu, sigmaOne)+intensity*lorentzianFunction(vector, mu, sigmaOne)
    value <- max(results)
    results <- (intensity/value)*results
  } else { #type 6, asymmetrical gaussian and lorentzian
    leftSide <- intensity*gaussianFunction(vector, mu, sigmaOne)+intensity*lorentzianFunction(vector, mu, sigmaOne)
    rigthSide <- intensity*gaussianFunction(vector, mu, sigmaTwo)+intensity*lorentzianFunction(vector, mu, sigmaTwo)
    value <- max(leftSide)
    leftSide <- (intensity/value)*leftSide
    value <- max(rigthSide)
    rigthSide <- (intensity/value)*rigthSide
    leftSide <- leftSide[1:mu]
    rigthSide <- rigthSide[(mu+1):length(rigthSide)]
    results <- c(leftSide, rigthSide)
  }
  return(results)
}


##########################################################################
#########################################################

# Function to create a large matrix with the m/zs and intensity of a metabolite ions

##########################################################################
#########################################################


#we need to add the noise and the baseline drift

createMzAndIntensityLargeMatrix <- function(mzAndIntensityValuesMetaboliteShort, valueRT, noise, baselineDrift, chromatographicTimeVector, sigmaOne, sigmaTwo){

  #we create a matrix which will contain all the information
```

```
  matrixMzIntensity <- matrix(data=0, nrow=nrow(noise), ncol=nrow(mzAndIntensityValuesMetaboliteShort))
  dimnames(matrixMzIntensity) <- list(chromatographicTimeVector, mzAndIntensityValuesMetaboliteShort[, 1])


  #we give to each m/z the intensity according to their number and also we add the intensity given by
mzAndIntensityValuesMetaboliteShort
  for (variable in 1:nrow(mzAndIntensityValuesMetaboliteShort)) {
    #number representing the m/z
    number <- round(as.numeric(mzAndIntensityValuesMetaboliteShort[variable, 1]))


    intensityNumber <- as.numeric(mzAndIntensityValuesMetaboliteShort[variable, 2])
    intensityVector <- c(1:80, intensityNumber)


    #the peak is put in the middle, then we will position it where it corresponds
    vectorIntensity <- createWave(2, sigmaOne, sigmaTwo, intensityVector, 40, intensityNumber)


    #we create a vector for the intensity with the same length that the chromatographic time vector
    #we add zeros before and after the rt
    zerosBeginning <- valueRT-floor(length(vectorIntensity)/2)+1
    zerosAtTheEnd <- length(chromatographicTimeVector)-(zerosBeginning+length(vectorIntensity))


    zerosBeginning <- rep(0, zerosBeginning)
    zerosAtTheEnd <- rep(0, zerosAtTheEnd)


    intensity <- append(zerosBeginning, vectorIntensity)
    intensity <- append(intensity, zerosAtTheEnd)


    #then, we sum the three vectors, intensity with the noise and the baseline drift
    Intensity <- intensity+noise[, number]+baselineDrift[, number]


    matrixMzIntensity[, variable] <- Intensity
  }


  return(matrixMzIntensity)
}


##################################################################
#####################################################
# Treatment of the metabolite with their information
##################################################################
#####################################################


#for the function the metabolite needs to be in a dataframe where
```

```
#each column is the intensity values of the m/zs for the retention time of the chromatography


#we return a list of each m/z with the intensities down to 0 except the ones that fitted the window inn case that the attribute zeros is true,
#on the other way, return with all intensities values


#function
treatedMetabolite <- function (zeros, mzAndIntensityValuesMetabolite, valueRT, peakWidth) {
  #we create an empty list were we will fill with the treated information
  metabolite <- list()
  #mzs
  rowNames <- colnames(mzAndIntensityValuesMetabolite)


  #we obtain the window for the chromatography time
  rt.window <- peakWidth[1] #min rt width in seconds
  rtWindow <- c(valueRT-rt.window, valueRT+rt.window)


  #we make a dataframe for each ion with the information: rt and intensity
  #then, we add each ion in the list
  for (variable in 1:ncol(mzAndIntensityValuesMetabolite)) {
    if (zeros==TRUE){
      #we extract the intensity values from a specific m/z
      vectorIntensity  <-  as.numeric(unlist(mzAndIntensityValuesMetabolite[(rtWindow[1]+1):(rtWindow[2]+1), variable]))


      chromatographicTimeVector <- row.names(mzAndIntensityValuesMetabolite)
      #we add zeros before and after the rt
      zerosBeginning <- rtWindow[1]
      zerosAtTheEnd <- (length(chromatographicTimeVector)-(zerosBeginning+length(vectorIntensity)))


      zerosBeginning <- rep(0, zerosBeginning)
      zerosAtTheEnd <- rep(0, zerosAtTheEnd)


      intensity <- append(zerosBeginning, vectorIntensity)
      intensity <- append(intensity, zerosAtTheEnd)
    } else {
      intensity <- as.numeric(unlist(mzAndIntensityValuesMetabolite[, variable]))
    }
    #finally we add it to a dataframe
    df <- data.frame(chromatographicTimeVector, intensity)
```

```
    nameIsotop <- rowNames[variable]
    metabolite[[nameIsotop]] <- df
  }
  return(metabolite)
}

##################################################################
##################################################################
# Chromatogram function
##################################################################
##################################################################

#the metabolite information needs to be organized as the one returned by the function treatedMetabolite
#function
chromatogramFunction <- function(metabolite, metabolitePeakRT, peakWidth){

  rt.window <- peakWidth[2] #min rt width in seconds
  rt <- c(metabolitePeakRT-rt.window, metabolitePeakRT+rt.window)

  #mzs numbers
  mzNames <- names(metabolite)
  #time vector
  chromatographicTimeVector <- metabolite[[1]][["chromatographicTimeVector"]]

  #empty matrix which will contain the metabolite information
  matrix <- matrix(data=0, nrow=length(chromatographicTimeVector), ncol=length(mzNames))
  dimnames(matrix) <- list(chromatographicTimeVector, mzNames)

  #we fill the matrix by column with the intensities
  for (variable in 1:length(mzNames)) {
    #we extract the column and add it to the matrix
    matrix[, variable] <- metabolite[[variable]][["intensity"]]
  }

  #we make a matrix only with the values within the rt window
  frontRows <- rt[1]
  behindRows <- length(chromatographicTimeVector) - rt[2] -1

  matrixSmall <- head(matrix, -behindRows)
  matrixSmall <- tail(matrixSmall, -frontRows)
```

```
  #we plot the m/zs in the retention time window
  #library(RColorBrewer)
  #group_colors <- brewer.pal(length(mzNames), name = "Set3")
  #matplot(x=rownames(matrixSmall), matrixSmall, type="l", ylab="intensity", xlab="chromatographic Time Vector", main="metabolite chromatogram", col=group_colors)
  #legend("topleft", colnames(matrixSmall), fill=group_colors, cex=0.5, title="mz")


  #we plot all the spectrum
  # matplot(x=rownames(matrix), matrix, type="l", ylab="intensity", xlab = "chromatographic Time Vector", main="metabolite chromatogram", col=group_colors)
  # legend("topleft", colnames(matrix), fill = group_colors, cex = 0.5)


  #plot all the spectrum in 3D
  # plot3D::persp3D(y=lengthChromatographicVector, z=t(matrix), theta=120, phi=25, ylab="chromatographic Time Vector", xlab="mz Simulated", zlab="Intensity", cex.lab=0.8)
  # title(main="chromatogram")


  df <- matrix(data=0, nrow=length(chromatographicTimeVector), ncol=(length(mzNames)-1))
  plotSurfaceInformation <- cbind(matrix, df)
  number <- length(mzNames) + (length(mzNames)-1)
  #in order to visualize the m/zs in a clear rows, we add empty columns between each mz
  for (variable in length(mzNames):2) {
    plotSurfaceInformation[, number] <- plotSurfaceInformation[, variable]
    plotSurfaceInformation[, variable] <- 0
    colnames(plotSurfaceInformation)[number] <- mzNames[variable]
    #for the names we use the difference between m/z, divided by two, and then add it to the lowest value, to finally transform it into a character
    colnames(plotSurfaceInformation)[(number-1)] <- as.character(((as.numeric(mzNames[variable])-as.numeric(mzNames[(variable-1)]))/2)+as.numeric(mzNames[(variable-1)]))
    number <- number-2
  }


  # plot3D::persp3D(y=lengthChromatographicVector, z=t(plotSurfaceInformation), theta=120, phi=25, ylab="chromatographic Time Vector", xlab="mz Simulated", zlab="Intensity", cex.lab=0.8)
  # title(main="chromatogram")


  #we prepare the data in order to fit the conditions that the plot geom_density_ridges asks for
  library("ggridges")
  intensityVector <- NULL
  mzVector <- NULL
  newRetentionTimeVector <- NULL
  #we are grouping the values of intensity with their m/z, we will include all the values kept before
  for (variable in 1:length(mzNames)) {
```

```
  intensityVector <- append(intensityVector, matrix[, variable])
  mzVector <- append(mzVector, rep(mzNames[variable], each=length(matrix[, variable])))
  newRetentionTimeVector <- append(newRetentionTimeVector, chromatographicTimeVector)
}

#finally, we give the information to a dataframe to plot it
dataMetabolite <- data.frame(
  x <- newRetentionTimeVector,
  y <- mzVector,
  intensity <- intensityVector
)
colnames(dataMetabolite) <- c("retentionTime", "mz", "intensity")

library(ggplot2)
#plot within all the retention time
# ggplot(dataMetabolite, aes(retentionTime, mz, height=intensity, group=mz)) +
#   geom_density_ridges(scale=1, stat="identity", fill="lightblue", alpha=0.5) +
#   labs(x="retention time", y="Intensity") +
#   ggtitle("Processed chromatogram") +
#   theme(plot.title=element_text(hjust=0.5))

#in order to see it more clearly, we visualize only where the window is
intensityVector2 <- NULL
mzVector2 <- NULL
newRetentionTimeVector2 <- NULL
#dataframe on the rt window
for (variable in 1:length(mzNames)) {
  intensityVector2 <- append(intensityVector2, matrixSmall[, variable])
  mzVector2 <- append(mzVector2, rep(mzNames[variable], each=length(matrixSmall[, variable])))
  newRetentionTimeVector2 <- append(newRetentionTimeVector2, as.numeric(rownames(matrixSmall)))
}

#dataframe for the information in the window time
data2Metabolite <- data.frame(
  x2 <- newRetentionTimeVector2,
  y2 <- mzVector2,
  intensity2 <- intensityVector2
)
colnames(data2Metabolite) <- c("retentionTime", "mz", "intensity")

intensitiesMzs <- ggplot(data2Metabolite, aes(retentionTime, mz, height=intensity, group=mz)) +
```

```
    geom_density_ridges(scale=1, stat="identity", fill="lightblue", alpha=0.5) +
    labs(x="retention time", y="Intensity") +
    ggtitle("Processed chromatogram") +
    theme(plot.title=element_text(hjust=0.5))


  #plot all together
  intensitiesStacked <- ggplot(data2Metabolite, aes(x=retentionTime, y=intensity, fill=mz)) +
geom_polygon(alpha=0.8) +
    ggtitle("Stacked chromatogram") +
    theme(plot.title=element_text(hjust=0.5))
  # ggplot(data2Metabolite, aes(x=retentionTime)) + geom_density(position="stack", aes(fill=mz))


  #plot the intensities of all the m/zs side by side within the window retention time
  library(hrbrthemes)
  plotMzsSepparately <- ggplot(data2Metabolite, aes(retentionTime, intensity, height=intensity)) +
    geom_density_ridges(scale=7, stat="identity", fill="lightblue", alpha=0.5) +
    theme_ipsum() +
    facet_wrap(~mz) +
    theme(
      legend.position="none",
      panel.spacing=unit(0.2, "lines"),
      axis.ticks.x=element_blank()
    ) +
    ggtitle("Intensity of each mz")

  chromatogramData <- list()
  plotsList <- list()

  plotsList[["intensitiesMzs"]] <- intensitiesMzs
  plotsList[["intensitiesStacked"]] <- intensitiesStacked
  plotsList[["plotMzsSeparately"]] <- plotMzsSepparately

  chromatogramData[["treatedData"]] <- as.data.frame(matrix)
  chromatogramData[["ggplotDataOriginal"]] <- dataMetabolite
  chromatogramData[["plotDataWindow"]] <- matrixSmall
  chromatogramData[["plots"]] <- plotsList
  chromatogramData[["plotSurfaceInformation"]] <- plotSurfaceInformation

  return(chromatogramData)
}
```

```
#####################################################################
#########################################################

#####################################################################
#########################################################




#####################################################################
#########################################################

# Function to plot the surface

#####################################################################
#########################################################


#plot the information in 3D about the m/z intensities within the retention time

surfacePlot <- function(chromatographicTimeVector, plotSurfaceInformation){

  lengthChromatographicVector <- 1:length(chromatographicTimeVector)


  #  plot3D::persp3D(y=lengthChromatographicVector,  z=t(plotSurfaceInformation),  theta=120,  phi=25,
ylab="chromatographic Time Vector", xlab="mz Simulated", zlab="Intensity", cex.lab=0.8)

  # title(main="chromatogram")


  plot3D::persp3D(z=plotSurfaceInformation, theta=120, phi=25, xlab="mz Simulated", ylab="retention time",
zlab="Intensity", cex.lab=0.8)

  title(main="chromatogram")

}


#####################################################################
#########################################################

# Sigmoide function

#####################################################################
#########################################################


#we make this function in order to use it to reduce the baseline drift in the intensities values due to the fact
that all of them have a sigmoide expression with the baseline drift

sigmoide <- function(x, a, b){

  1/(1+exp(-a*(x-b)))

}


#####################################################################
#########################################################

# Function to correct the baseline

#####################################################################
#########################################################


#for a vector

correctBaseline <- function(intensityValuesMetabolite, chromatographicTimeVector){
```

```
  lastNumbers <- tail(intensityValuesMetabolite, -(length(chromatographicTimeVector)-500))

  A <- min(lastNumbers)
  #a curvature
  #b pos
  a <- 0.007
  b <- 530
  sigmoideFunction <- A*sigmoide(chromatographicTimeVector, a, b)

  intensityValuesMetabolite <- intensityValuesMetabolite-sigmoideFunction

  #we check that all values are between 0 and the maximum intensity in each row of that column
  for (rowNumber in 1:length(chromatographicTimeVector)) {
    if (intensityValuesMetabolite[rowNumber]<0){
      intensityValuesMetabolite[rowNumber] <- 0
    }
  }
  return(intensityValuesMetabolite)
}

#for a matrix
correctBaselineMatrix <- function(mzAndIntensityValuesMetabolite, chromatographicTimeVector){
  mzNames <- colnames(mzAndIntensityValuesMetabolite)
  matrixLastNumbers <- tail(mzAndIntensityValuesMetabolite, -(length(chromatographicTimeVector)-500))

  for (variable in 1:length(mzNames)) {
    A <- min(matrixLastNumbers[, variable])
    #a curvature
    #b pos
    a <- 0.007
    b <- 530
    sigmoideFunction <- A*sigmoide(chromatographicTimeVector, a, b)

    mzAndIntensityValuesMetabolite[, variable] <- mzAndIntensityValuesMetabolite[, variable]-sigmoideFunction

    for (rowNumber in 1:length(chromatographicTimeVector)) {
      if (mzAndIntensityValuesMetabolite[rowNumber, variable]<0){
        mzAndIntensityValuesMetabolite[rowNumber, variable] <- 0
      }
    }
  }
```

```
  return(mzAndIntensityValuesMetabolite)
}


##########################################################################
#########################################################
##########################################################################
#########################################################



##########################################################################
#########################################################
# Global Simulation Parameters
##########################################################################
#########################################################


sampleTime <-1 # in seconds
elutionDuration <- 1800 # in seconds
nominalMzSimulated <- 20:600  # integer m/zs


##########################################################################
#########################################################
# Original Empty GC-MS Matrix Generation
##########################################################################
#########################################################


chromatographicTimeVector <- seq(0, elutionDuration, by=sampleTime)


gcmsMatrixNumRow <- length(chromatographicTimeVector) # axis
gcmsMatrixNumColumns <- length(nominalMzSimulated)


gcmsMatrix <- matrix(data=0, nrow=gcmsMatrixNumRow, ncol=gcmsMatrixNumColumns)
dimnames(gcmsMatrix) <- list(chromatographicTimeVector, nominalMzSimulated)


##########################################################################
#########################################################
# Retention Index or Absolute Retention based on Alkanes
##########################################################################
#########################################################


alkanesPresent <- FALSE


##########################################################################
#########################################################
```

```
noisePower <- 8 # assuming an equal noise power for each nominal m/z channel for now
noiseVector <- abs(noisePower*rnorm(gcmsMatrixNumRow*gcmsMatrixNumColumns, mean=0, sd=sqrt(noisePower)))
noiseMatrix <- matrix(data=noiseVector, nrow=gcmsMatrixNumRow, ncol=gcmsMatrixNumColumns)


#################################################################################################
# Baseline drift Simulation Parameters and generation
#################################################################################################


# different "power levels" for each m/z drift. For now, random, but it could be selected manually or made constant
# baselinePower <- 0.682
baselinePower <- 0.200
baselinePower <- abs(100*rnorm(gcmsMatrixNumRow, mean = 0, sd = sqrt(baselinePower)))


# For now, constant CutOff Frequency and FIR filter length
lengthFIR <- 1000
cutOffFreq <- 0.0001


# low pass filter to simulate baseline drift, for now fixed, but could be personalized for each m/z manually or randomly in length (n), cutoff frequency (wc) and fir parameters (firCoefficientVector)
firCoefficientsVector <- NULL
n <- NULL
wc <- NULL


for (i in (1:length(nominalMzSimulated))) {
  # length n of the FIR filter
  n[i]=lengthFIR


  # discrete cut-off frequency of the filter, being 1 = NYQUIST frequency
  wc[i]=cutOffFreq


  # fir coefficients
  aux8 <- fir1(n[i], wc[i], type=c("low"), window=hamming(n[i]+1), scale = TRUE)
  firCoefficientsVector <- c(firCoefficientsVector, aux8)
}


firCoefficientsMatrix <- matrix(data=firCoefficientsVector, length(aux8), length(nominalMzSimulated))
noiseBaseLineDriftVector <- NULL


for (i in (1:length(nominalMzSimulated))) {
```

```
  aux1 <- abs(baselinePower[i+1]*rnorm(gcmsMatrixNumRow, mean=0, sd=sqrt(baselinePower)))
  filteredBaseLine <- fftfilt(firCoefficientsMatrix[, i], aux1, n=NULL)
  aux2 <- filteredBaseLine[1:1801]
  noiseBaseLineDriftVector <- c(noiseBaseLineDriftVector, aux2)
}

# Here, manually input baseLines can be introduced for each m/z if considered more realistic. Coefficients are columns
baseLineDriftMatrix <- matrix(data=noiseBaseLineDriftVector, nrow=gcmsMatrixNumRow, ncol=gcmsMatrixNumColumns)

##################################################################################################
# Compose the matrix with noise and baseline drift
##################################################################################################

gcmsMatrix <- gcmsMatrix+noiseMatrix+baseLineDriftMatrix

##################################################################################################
# 3D representation of GC-MS
##################################################################################################

lengthNominalMZSimulated=1:length(nominalMzSimulated)
lengthChromatographicVector=1:length(chromatographicTimeVector) # x axis

# 2D representation of the GC-MS
matplot(gcmsMatrix, type="l", ylab="intensity")
title(main="chromatogram of the noise and the baseline in a GC-MS", xlab="retention time")

# 3D representation of the GC-MS with the value for the noise and the baseline
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(gcmsMatrix), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="empty GC-MS")

##################################################################################################
# Reading mzml data
##################################################################################################

# first of all, we read the dataset
```

```
path.mzML <- "mzml/"
mzML.files <- list.files(path.mzML, pattern=".mzML",
                         recursive=T, full.names=T)


pheno <- phenoDataFromPaths(mzML.files)
pheno$sample_name <- rownames(pheno)
pheno$sample_group <- pheno$class # because we don't have the data separated by group, we can erase this line


raw_data <- readMSData(files=mzML.files,
                       pdata=new("NAnnotatedDataFrame", pheno),
                       mode="onDisk")


# summary(pheno)
# we can see the samples
# pheno$sample_name


# Get TICs
TICs <- chromatogram(raw_data, aggregationFun="sum")


# we can plot the TIC according to the experimental samples
group_colors <- paste0(brewer.pal(length(pData(raw_data)$sample_name), "Set3"), "60")
names(group_colors) <- levels(pData(raw_data)$sample_name)


plot(TICs, col=group_colors, main="TIC")
legend("topright", raw_data$sample_name, fill=group_colors, cex=0.5)


# alk_05042022
plot(TICs[[2]], col=group_colors[3], main="alkane TIC")
legend("topright", raw_data$sample_name[2], fill=group_colors[3], cex=0.5)


# blank_04042022
plot(TICs[[4]], col=group_colors[4], main="blank TIC")
legend("topright", raw_data$sample_name[4], fill=group_colors[4], cex=0.5)


# s3_27042022
plot(TICs[[7]], col=group_colors[5], main="sample TIC")
legend("topright", raw_data$sample_name[7], fill=group_colors[5], cex=0.5)


# s4_13_06052022
plot(TICs[[8]], col=group_colors[6], main="sample TIC")
```

```
legend("topright", raw_data$sample_name[8], fill=group_colors[6], cex=0.5)


# raw_data

spectraList <- raw_data %>% spectra


#####################################################################
#####################################################

# Definition of Metabolites

#####################################################################
#####################################################


# we will need the metabolite rt, and we will extract it from the samples of the GC-MS given, due to the fact that we know the m/z but not the rt

# we look for the names that contain the data, which will be the ones that contains F01 in their name

namesSpectra <- names(spectraList)

specificNames <- grep("F01", namesSpectra, value=TRUE)


# we observe that the retention time goes from 0:00 to 29:60 minutes, because we will work in nominal and seconds, the retention time will be the following:

retentionTimeFile <- c(0:1800) # seconds

nominalMzSimulated <- 20:600  # integer m/zs

msMatrixNumRow <- length(retentionTimeFile) # axis

msMatrixNumColumns <- length(nominalMzSimulated)


# first

msMatrixS3 <- matrix(data=0, nrow=msMatrixNumRow, ncol=msMatrixNumColumns)

dimnames(msMatrixS3) <- list(retentionTimeFile, nominalMzSimulated)

specificNames <- grep("F07", namesSpectra, value=TRUE)

for (variable in 1:length(specificNames)) {

  msMatrixS3[round(spectraList[[variable]]@rt), round(spectraList[[variable]]@mz)] <- msMatrixS3[round(spectraList[[variable]]@rt), round(spectraList[[variable]]@mz)] + spectraList[[variable]]@intensity

}

# second

msMatrixS4_13 <- matrix(data=0, nrow=msMatrixNumRow, ncol=msMatrixNumColumns)

dimnames(msMatrixS4_13) <- list(retentionTimeFile, nominalMzSimulated)

specificNames <- grep("F08", namesSpectra, value=TRUE)

for (variable in 1:length(specificNames)) {

  msMatrixS4_13[round(spectraList[[variable]]@rt), round(spectraList[[variable]]@mz)] <- msMatrixS4_13[round(spectraList[[variable]]@rt), round(spectraList[[variable]]@mz)] + spectraList[[variable]]@intensity

}

# third

msMatrixS4_16 <- matrix(data=0, nrow=msMatrixNumRow, ncol=msMatrixNumColumns)
```

```
dimnames(msMatrixS4_16) <- list(retentionTimeFile, nominalMzSimulated)

specificNames <- grep("F09", namesSpectra, value=TRUE)

for (variable in 1:length(specificNames)) {

  msMatrixS4_16[round(spectraList[[variable]]@rt), round(spectraList[[variable]]@mz)] <- msMatrixS4_16[round(spectraList[[variable]]@rt), round(spectraList[[variable]]@mz)] + spectraList[[variable]]@intensity

}


# FIRST METABOLITE

# We introduce the characteristics for creatine (C4H9N3O2)

# we upload the csv with the m/z and abundance of the metabolite, also its ions

name <- "creatine"

metaboliteOneOriginal <- read.csv(file="~/Desktop/metabolites/C4H9N3O2original.csv", header=F, sep=";", dec=",")

msSpectra <- createMetabolite(metaboliteOneOriginal)

msSpectra <- mzNominal(msSpectra)


# we look for the retention time for this mass spectra with the mzml information

# retention time for creatine

numberMz <- as.character(msSpectra[1, 1])

vectorOfRt <- msMatrixS4_16[, numberMz]

rtMetabolite <- as.numeric(which.max(vectorOfRt))


# plot of the m/z intensities in the first metabolite

matplot(x=msSpectra[1], y=msSpectra[2], type="h", xlab="mz", ylab="intensity")

title(main="Metabolite 1")


peakMax <- msMatrixS4_16[rtMetabolite, numberMz] # Intensity


# we need to obtain a matrix with the wave in each m/z of the metabolite

# we want to use this function to construct a wide matrix, but we want a perfect spectra, so we empty the noise and baseline matrices

noiseM <- matrix(data=0, nrow=gcmsMatrixNumRow, ncol=gcmsMatrixNumColumns)

dimnames(noiseM) <- list(chromatographicTimeVector, nominalMzSimulated)


baselineDriftM <- matrix(data=0, nrow=gcmsMatrixNumRow, ncol=gcmsMatrixNumColumns)

dimnames(baselineDriftM) <- list(chromatographicTimeVector, nominalMzSimulated)


# we give the sigmas for the gaussian function

sigmaOne <- 10

sigmaTwo <- 20

matrixMetabolite <- createMzAndIntensityLargeMatrix(msSpectra, rtMetabolite, noiseM, baselineDriftM, chromatographicTimeVector, sigmaOne, sigmaTwo)
```

```
peakWidth <- c(5, 40)
adduct <- 1.007276
monoisotopicMolecularWeight <- 131.0694766
MAdduct <- monoisotopicMolecularWeight + adduct

zeros <- FALSE
metabolite <- treatedMetabolite (zeros, matrixMetabolite, rtMetabolite, peakWidth)
metaboliteChromatogramInformation <- chromatogramFunction(metabolite, rtMetabolite, peakWidth)

metabolite001 <- list(name, peakMax, rtMetabolite, msSpectra, matrixMetabolite, metaboliteChromatogramInformation, peakWidth)
names(metabolite001) <- c("name", "peakMax", "rtMetabolite", "msSpectra", "matrixMetabolite", "metaboliteChromatogramInformation", "peakWidth")
############

# SECOND METABOLITE
# C3H7NO2, alanine
name <- "alanine"
metaboliteTwoOriginal <- read.csv(file="~/Desktop/metabolites/C3H7NO2original.csv", header=F, sep=";", dec=",")
msSpectra <- createMetabolite(metaboliteTwoOriginal)
msSpectra <- mzNominal(msSpectra)

numberMz <- as.character(msSpectra[1, 1])
vectorOfRt <- msMatrixS4_16[, numberMz]
rtMetabolite <- as.numeric(which.max(vectorOfRt))

peakMax <- msMatrixS4_16[rtMetabolite, numberMz] # Intensity

sigmaOne <- 10
sigmaTwo <- 20
matrixMetabolite <- createMzAndIntensityLargeMatrix(msSpectra, rtMetabolite, noiseM, baselineDriftM, chromatographicTimeVector, sigmaOne, sigmaTwo)

peakWidth <- c(5, 40)
adduct <- 1.007276
monoisotopicMolecularWeight <- 89.04767847
MAdduct <- monoisotopicMolecularWeight + adduct

zeros <- FALSE
metabolite <- treatedMetabolite (zeros, matrixMetabolite, rtMetabolite, peakWidth)
```

```
metaboliteChromatogramInformation <- chromatogramFunction(metabolite, rtMetabolite, peakWidth)


metabolite002 <- list(name, peakMax, rtMetabolite, msSpectra, matrixMetabolite, metaboliteChromatogramInformation, peakWidth)
names(metabolite002) <- c("name", "peakMax", "rtMetabolite", "msSpectra", "matrixMetabolite", "metaboliteChromatogramInformation", "peakWidth")


# THIRD METABOLITE
# C4H7N3O, creatinine
name <- "creatinine"
metaboliteOriginal <- read.csv(file="~/Desktop/metabolites/C4H7N3Ooriginal.csv", header=F, sep=";", dec=",")
msSpectra <- createMetabolite(metaboliteOriginal)
msSpectra <- mzNominal(msSpectra)


numberMz <- as.character(msSpectra[1, 1])
vectorOfRt <- msMatrixS4_16[, numberMz]
rtMetabolite <- as.numeric(which.max(vectorOfRt))


peakMax <- msMatrixS4_16[rtMetabolite, numberMz] # Intensity


sigmaOne <- 10
sigmaTwo <- 20
matrixMetabolite <- createMzAndIntensityLargeMatrix(msSpectra, rtMetabolite, noiseM, baselineDriftM, chromatographicTimeVector, sigmaOne, sigmaTwo)


peakWidth <- c(5, 40)
adduct <- 1.007276
monoisotopicMolecularWeight <- 113.0589119
MAdduct <- monoisotopicMolecularWeight + adduct


zeros <- FALSE
metabolite <- treatedMetabolite (zeros, matrixMetabolite, rtMetabolite, peakWidth)
metaboliteChromatogramInformation <- chromatogramFunction(metabolite, rtMetabolite, peakWidth)


metabolite003 <- list(name, peakMax, rtMetabolite, msSpectra, matrixMetabolite, metaboliteChromatogramInformation, peakWidth)
names(metabolite003) <- c("name", "peakMax", "rtMetabolite", "msSpectra", "matrixMetabolite", "metaboliteChromatogramInformation", "peakWidth")


# FOURTH METABOLITE
# C1H4N2O1, urea
name <- "urea"
```

```
metaboliteOriginal <- read.csv(file="~/Desktop/metabolites/CH4N2Ooriginal.csv", header=F, sep=";", dec=",")
msSpectra <- createMetabolite(metaboliteOriginal)
msSpectra <- mzNominal(msSpectra)

numberMz <- as.character(msSpectra[1, 1])
vectorOfRt <- msMatrixS4_16[, numberMz]
rtMetabolite <- as.numeric(which.max(vectorOfRt))

peakMax <- msMatrixS4_16[rtMetabolite, numberMz] # Intensity

sigmaOne <- 10
sigmaTwo <- 20
matrixMetabolite <- createMzAndIntensityLargeMatrix(msSpectra, rtMetabolite, noiseM, baselineDriftM, chromatographicTimeVector, sigmaOne, sigmaTwo)

peakWidth <- c(5, 40)
adduct <- 1.007276
monoisotopicMolecularWeight <- 60.03236277
MAdduct <- monoisotopicMolecularWeight + adduct

zeros <- FALSE
metabolite <- treatedMetabolite (zeros, matrixMetabolite, rtMetabolite, peakWidth)
metaboliteChromatogramInformation <- chromatogramFunction(metabolite, rtMetabolite, peakWidth)

metabolite004 <- list(name, peakMax, rtMetabolite, msSpectra, matrixMetabolite, metaboliteChromatogramInformation, peakWidth)
names(metabolite004) <- c("name", "peakMax", "rtMetabolite", "msSpectra", "matrixMetabolite", "metaboliteChromatogramInformation", "peakWidth")

# FIFTH METABOLITE
# C6H14N2O2, lysine
name <- "lysine"
metaboliteOriginal <- read.csv(file="~/Desktop/metabolites/C6H14N2O2original.csv", header=F, sep=";", dec=",")
msSpectra <- createMetabolite(metaboliteOriginal)
msSpectra <- mzNominal(msSpectra)

numberMz <- as.character(msSpectra[1, 1])
vectorOfRt <- msMatrixS4_16[, numberMz]
rtMetabolite <- as.numeric(which.max(vectorOfRt))

peakMax <- msMatrixS4_16[rtMetabolite, numberMz] # Intensity
```

```
sigmaOne <- 10
sigmaTwo <- 20
matrixMetabolite <- createMzAndIntensityLargeMatrix(msSpectra, rtMetabolite, noiseM, baselineDriftM, chromatographicTimeVector, sigmaOne, sigmaTwo)

peakWidth <- c(5, 40)
adduct <- 1.007276
monoisotopicMolecularWeight <- 146.1055277
MAdduct <- monoisotopicMolecularWeight + adduct

zeros <- FALSE
metabolite <- treatedMetabolite (zeros, matrixMetabolite, rtMetabolite, peakWidth)
metaboliteChromatogramInformation <- chromatogramFunction(metabolite, rtMetabolite, peakWidth)

metabolite005 <- list(name, peakMax, rtMetabolite, msSpectra, matrixMetabolite, metaboliteChromatogramInformation, peakWidth)
names(metabolite005) <- c("name", "peakMax", "rtMetabolite", "msSpectra", "matrixMetabolite", "metaboliteChromatogramInformation", "peakWidth")

#################################################################################################
# Compound library
#################################################################################################

listOfMetabolites <- list(metabolite001, metabolite002, metabolite003, metabolite004, metabolite005)

#################################################################################################
# Artificial chromatogram information
#################################################################################################
#now that we have a list with all the metabolite information regarding the GC-MS samples, now we simulate the final data

# because we already make the metabolite Matrix, we only have to sum all the information with all the metabolites data that we want with noise and with the baseline drift

# we fill the matrix that have the noise and the baseline with the spectra information of the metabolites
for (variable in 1:length(listOfMetabolites)) {
  vectorNames <- colnames(listOfMetabolites[[variable]][["matrixMetabolite"]])
  for (names in 1:length(vectorNames)) {
    # because we want to express a bigger quantity of each metabolite, we scale it
```

```
    gcmsMatrix[, vectorNames[names]] <- gcmsMatrix[, vectorNames[names]] + 10*(listOfMetabolites[[variable]][["matrixMetabolite"]][, vectorNames[names]])
  }
}

# 2D representation of the GC-MS
matplot(gcmsMatrix, type="l", ylab="intensity")
title(main="chromatogram of the simulated GC-MS", xlab="retention time")

# 3D representation of the GC-MS with the value for the noise and the baseline
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(gcmsMatrix), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="GC-MS")

# we create two more matrices to simulate a way of saving data with samples and their intensities along time
# we scale to obtain different information per matrix, also, in one of them we leave out one metabolite information
# in order to have a little more variability across the three matrices
testMatrix <- gcmsMatrix
secondTestMatrix <- gcmsMatrix
for (variable in 1:length(listOfMetabolites)) {
  vectorNames <- colnames(listOfMetabolites[[variable]][["matrixMetabolite"]])
  for (names in 1:length(vectorNames)) {
    # we scale it
    testMatrix[, vectorNames[names]] <- testMatrix[, vectorNames[names]] + 15*(listOfMetabolites[[variable]][["matrixMetabolite"]][, vectorNames[names]])
  }
}

for (variable in 2:length(listOfMetabolites)) {
  vectorNames <- colnames(listOfMetabolites[[variable]][["matrixMetabolite"]])
  for (names in 1:length(vectorNames)) {
    # we scale it
    secondTestMatrix[, vectorNames[names]] <- secondTestMatrix[, vectorNames[names]] + 5*(listOfMetabolites[[variable]][["matrixMetabolite"]][, vectorNames[names]])
  }
}

# we sum the rows to obtain the intensity for all the m/zs in that retention time
gcmsMatrix_2 <- data.frame(chromatographicTimeVector, rowSums(gcmsMatrix))
colnames(gcmsMatrix_2) <- c("time", "area")
testMatrix_2 <- data.frame(chromatographicTimeVector, rowSums(testMatrix))
colnames(testMatrix_2) <- c("time", "area")
```

```
secondTestMatrix_2 <- data.frame(chromatographicTimeVector, rowSums(secondTestMatrix))
colnames(secondTestMatrix_2) <- c("time", "area")


dataListSimulated <- list()
dataListSimulated[["normalSimulated"]] <- gcmsMatrix_2
dataListSimulated[["alteratedSimulated"]] <- testMatrix_2
dataListSimulated[["missingMetabolite"]] <- secondTestMatrix_2


##################################################################
##########################################################
##################################################################
##########################################################



#for all the cases we will focus on the first metabolite output, their ions and intensities
#listOfMetabolites[[1]][["name"]]
#creatine

rtMetabolite <- listOfMetabolites[[1]][["rtMetabolite"]]
msSpectraMetabolite <- listOfMetabolites[[1]][["msSpectra"]]
peakWidth <- listOfMetabolites[[1]][["peakWidth"]]


##################################################################
##########################################################
# XCMS study for the real data
##################################################################
##########################################################


#read the data is already done


# we visualize the TIC (chromatographic data) for an m/z slice
chr <- chromatogram(raw_data)
chr
plot(chr)


# we plot the TIC according to the experimental samples
# this plot is already done before, but we repeat this step to compare
group_colors <- paste0(brewer.pal(length(pData(raw_data)$sample_name), "Set3"), "60")
names(group_colors) <- levels(pData(raw_data)$sample_name)
plot(chr, col=group_colors, main="TIC")
legend("topright", chr$sample_name, fill=group_colors, cex=0.5)
```

```
# we filter the data around the ion metabolite desired to compare
# Extract and plot the XIC for creatine
raw_data %>%
  filterRt(rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])) %>%
  filterMz(mz=msSpectraMetabolite[, 1]) %>%
  chromatogram(aggregationFun="max") %>%
  plot()

# we filter the MS data to the signal from the creatine ions and plot it using type="XIC"
raw_data %>%
  filterRt(rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])) %>%
  filterMz(mz=msSpectraMetabolite[, 1]) %>%
  plot(type="XIC")

# next step is smoothing, and also we do a peak picking
data_cent <- raw_data %>%
  smooth(method="SavitzkyGolay", halfWindowSize=6) %>%
  pickPeaks()

# now, we can observe the differences between before and after the smoothing
data_cent %>%
  filterRt(rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])) %>%
  filterMz(mz=msSpectraMetabolite[, 1]) %>%
  plot(type="XIC")

# we filter around the values that we want and also we look in a specific region for the rt and m/z
crtn_chr <- chromatogram(data_cent, rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1]),
                         mz=msSpectraMetabolite[, 1],
                         aggregationFun="max")
# plot of the data
par(mfrow=c(1, 1), mar=c(4, 4.5, 1, 1))
plot(crtn_chr)

# centWave parameters
# cwp <- CentWaveParam() default parameters
cwp <- CentWaveParam(ppm=25, peakwidth=c(0.1, 5),
                     snthresh=10, prefilter=c(3, 100),
                     mzCenterFun="wMean", integrate=2,
                     mzdiff=-0.001, fitgauss=FALSE,
                     noise=0, verboseColumns=FALSE,
                     roiList=list(), firstBaselineCheck=TRUE,
```

```
                    roiScales=numeric())

# peak detection on the XIC
results <- findChromPeaks(crtn_chr, param=cwp)
# chromPeaks(results)
results <- chromPeaks(results)

# plot(results)
datatable(results) %>%
  formatRound(c("rt", "rtmin", "rtmax"), 0, mark="") %>%
  formatRound(c("into", "intb", "maxo"), 0, mark="")

# because we did not find any peak with the before data we perform chromatographic peak detection in the whole dataset
xdata <- findChromPeaks(data_cent, param=cwp)

# we define settings for the alignment
# we align the samples using obiwarp
xdata <- adjustRtime(xdata, param=ObiwarpParam(binSize=1))

plotAdjustedRtime(xdata, main="Adjusted retention time")

# we use adjustedRtime parameter to access raw/adjusted retention times
mzr <- c(min(msSpectraMetabolite[, 1]), max(msSpectraMetabolite[, 1]))
rtr <- c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])

par(mfrow = c(1, 2), mar = c(4, 4.5, 0.9, 0.5))
plot(chromatogram(xdata, mz=mzr, rt=rtr, adjustedRtime=FALSE), peakType="none", lwd=2,
    main="Before alignment")
plot(chromatogram(xdata, mz=mzr, rt=rtr), peakType="none", lwd=2,
    main="After alignment")

hasAdjustedRtime(xdata)

# now, we do the XCMS correspondence
# we will do group signal (peaks) from the same ion across samples
# Using the groupChromPeaks() function and the peak density and nearest methods

# The method peak density iterate through slices along m/z. So, we compare the retention times of the peaks
# within each slice and group the peaks if they are close to each other
## we set the XCMS parameters for peak density
```

```
onlinexcms_pdp <- PeakDensityParam(sampleGroups=pheno$sample_name,
                           bw=5,
                           minFraction=0.8,
                           binSize=0.015,
)

# Now, we adjust the peak density parameters. At the same time we will plot the data for the m/z slice
# containing the creatine peaks

# The plotChromPeakDensity() function-method plots the distribution of the identified peaks along rt for a
# given m/z segment, simulating correspondence analysis

# we extract a BPC for an m/z slice containing creatine
par(mfrow=c(2, 1), mar=c(4, 4.3, 1, 0.5))
plot(chromatogram(xdata, mz=mzr, aggregationFun="max"))
highlightChromPeaks(xdata, mz=mzr, whichPeaks="apex_within", main="Chromatogram peak density")

# dry-run correspondence and results.
plotChromPeakDensity(xdata, mz=mzr, type="apex_within",
                     param=onlinexcms_pdp, main="bw = 5")

###extra###
# merge peaks that are very close
# mpp <- MergeNeighboringPeaksParam(expandRt=4)
# data_cent_pp <- refineChromPeaks(data_cent, param=mpp)
# chromPeakData(data_cent_pp)

##########################################################################
#########################################################
# XCMS study for the simulation
##########################################################################
#########################################################

# we will grab the onDisk object and fill it with the information of the simulation
simulation_rawData <- raw_data

# simulation_rawData@phenoData@data <- raw_data@phenoData@data[1:3, ]
simulation_rawData@phenoData@data[["sample_name"]] <- c("gcmsMatrix", "testMatrix", "secondTestMatrix",
                                          "gcmsMatrix2", "testMatrix2", "secondTestMatrix2",
                                          "gcmsMatrix3", "testMatrix3", "secondTestMatrix3",
                                          "gcmsMatrix4", "testMatrix4", "secondTestMatrix4"
```

```
)
namesToUse <- simulation_rawData@phenoData@data[["sample_name"]]
length(namesToUse)
row.names(simulation_rawData@phenoData@data) <- c("gcmsMatrix", "testMatrix", "secondTestMatrix",
                                                  "gcmsMatrix2", "testMatrix2", "secondTestMatrix2",
                                                  "gcmsMatrix3", "testMatrix3", "secondTestMatrix3",
                                                  "gcmsMatrix4", "testMatrix4", "secondTestMatrix4"
)
# row.names(simulation_rawData@phenoData@data) <- c("gcmsMatrix", "testMatrix", "secondTestMatrix")

simulation_rawData@phenoData
simulation_rawData@phenoData@data[["sample_name"]]
row.names(simulation_rawData@phenoData@data)

# second part
# simulation_rawData@featureData@data <- data.frame()
# simulation_rawData@featureData@data <- raw_data@featureData@data

# variables that will be the same along the "features"
simulation_rawData@featureData@data[, "originalPeaksCount"] <- 0

# we will change the rt only in the "features" that contain any ion metabolite
nMetabolites <- 5 # length(listOfMetabolites)

# we add five metabolites, so we need to look for these to incorporate the rt
# also, because there are several m/zs with the same rt, we make an extra loop for that
for (time in 1:nMetabolites) {
  mzNumber <- listOfMetabolites[[time]][["msSpectra"]][ ,1]
  for (mzVariable in 1:length(mzNumber)) {
    # because the m/z starts at 20 and here at 1
    # we want to show only the ion metabolite peak, the others signals are noise and baseline drift
    simulation_rawData@featureData@data[mzNumber[mzVariable]-19, "originalPeaksCount"] <- 1
    simulation_rawData@featureData@data[mzNumber[mzVariable]-19, "retentionTime"] <-
listOfMetabolites[[time]][["rtMetabolite"]]
    # we give the intensity peak in each m/z that contains an ion metabolite
    simulation_rawData@featureData@data[mzNumber[mzVariable]-19, "basePeakIntensity"] <-
gcmsMatrix[listOfMetabolites[[time]][["rtMetabolite"]], mzNumber[mzVariable]]
  }
}

namesMatrix <- colnames(gcmsMatrix)
```

```
gcmsMatrix_3 <- colSums(gcmsMatrix)
testMatrix_3 <- colSums(testMatrix)
secondTestMatrix_3 <- colSums(secondTestMatrix)

vector <- ncol(gcmsMatrix)

for (variable in 1:vector) {
  # because there will be only one peak per "feature"
  simulation_rawData@featureData@data[variable, "totIonCurrent"] <- gcmsMatrix_3[variable] + testMatrix_3
[variable] + secondTestMatrix_3[variable]

  simulation_rawData@featureData@data[variable, "basePeakMZ"] <- as.numeric(namesMatrix[variable])
  # because it will only be one peak, the m/z values below can be the same as the main m/z
  simulation_rawData@featureData@data[variable, "lowMZ"] <- as.numeric(namesMatrix[variable])
  simulation_rawData@featureData@data[variable, "highMZ"] <- as.numeric(namesMatrix[variable])
  # we will leave the same injection time, otherwise, it can be changed by the following line
  # simulation_rawData@featureData@data[variable, "injectionTime"] <-
}

# we will give the information to each feature of the onDisk object the information of each m/z
# we want the same amount of features that the columns created for the simulation
simulation_rawData@featureData@data[(vector+1):nrow(simulation_rawData@featureData@data), ] <- 1

simulation_rawData@featureData
simulation_rawData@featureData@data

### now, we can start the XCMS workflow
# we visualize the TIC (chromatographic data) for an m/z slice
chr <- chromatogram(simulation_rawData)
chr
plot(chr)

# we plot the TIC according to the simulated samples
group_colors <- paste0(brewer.pal(length(pData(simulation_rawData)$sample_name), "Set3"), "60")
names(group_colors) <- levels(pData(simulation_rawData)$sample_name)
plot(chr, col=group_colors, main="TIC")
legend("topright", chr$sample_name, fill=group_colors, cex=0.5)

# we filter the data around the ion metabolite desired to compare
# Extract and plot the XIC for creatine
```

```
simulation_rawData %>%
  filterRt(rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])) %>%
  filterMz(mz=msSpectraMetabolite[, 1]) %>%
  chromatogram(aggregationFun="max") %>%
  plot()

# we filter the MS data to the signal from the creatine ion and plot it using type = "XIC"
simulation_rawData %>%
  filterRt(rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])) %>%
  filterMz(mz=msSpectraMetabolite[, 1]) %>%
  plot(type="XIC")

# next step is smoothing
data_cent <- simulation_rawData %>%
  smooth(method="SavitzkyGolay", halfWindowSize = 6) #%>%
#pickPeaks()

# now, we can observe the differences between before and after the smoothing
data_cent %>%
  filterRt(rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])) %>%
  filterMz(mz=msSpectraMetabolite[, 1]) %>%
  plot(type="XIC")

# we filter around the values that we want and also we look in a specific region for the rt and m/z
crtn_chr <- chromatogram(data_cent, rt=c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1]),
                         mz=msSpectraMetabolite[, 1],
                         aggregationFun="max")
# we plot the data
par(mfrow=c(1, 1), mar=c(4, 4.5, 1, 1))
plot(crtn_chr)

# centWave parameters
# cwp <- CentWaveParam() default parameters
cwp <- CentWaveParam(ppm=25, peakwidth=c(0.1, 5),
                     snthresh=10, prefilter=c(3, 100),
                     mzCenterFun="wMean", integrate=2,
                     mzdiff=-0.001, fitgauss=FALSE,
                     noise=0, verboseColumns=FALSE,
                     roiList=list(), firstBaselineCheck=TRUE,
                     roiScales=numeric())
```

```
# peak detection on the XIC
# results <- findChromPeaks(crtn_chr, param = cwp)
# chromPeaks(results)
# results <- chromPeaks(results)

# plot(results)
# datatable(results) %>%
#   formatRound(c("rt", "rtmin", "rtmax"), 0, mark="") %>%
#   formatRound(c("into", "intb", "maxo"), 0, mark="")

# xdata <- findChromPeaks(data_cent, param =  cwp)

# xdata <- adjustRtime(xdata, param=ObiwarpParam(binSize=1))

# plotAdjustedRtime(xdata, main="Adjusted retention time")

# mzr <- c(min(msSpectraMetabolite[, 1]), max(msSpectraMetabolite[, 1]))
# rtr <- c(rtMetabolite-peakWidth[1], rtMetabolite+peakWidth[1])

# par(mfrow=c(1, 2), mar=c(4, 4.5, 0.9, 0.5))
# plot(chromatogram(xdata, mz=mzr,rt=rtr, adjustedRtime=FALSE), peakType="none", lwd=2,
#      main="Before alignment")
# plot(chromatogram(xdata, mz=mzr, rt=rtr), peakType="none", lwd=2,
#      main="After alignment")

# hasAdjustedRtime(xdata)

# now, we do the XCMS correspondence

# onlinexcms_pdp <- PeakDensityParam(sampleGroups = pheno$sample_name,
#                                    bw=5,
#                                    minFraction=0.8,
#                                    binSize=0.015,
# )

# Now, we adjust the peak density parameters. At the same time we will plot the data for the m/z slice
# containing the creatine peaks.

#we extract a BPC for an m/z slice containing creatine
# par(mfrow=c(2, 1), mar=c(4, 4.3, 1, 0.5))
# plot(chromatogram(xdata, mz=mzr, aggregationFun="max"))
```

```
# highlightChromPeaks(xdata, mz=mzr, whichPeaks="apex_within", main="Chromatogram peak density")
#
# #dry-run correspondence and we show the results.
# plotChromPeakDensity(xdata, mz=mzr, type="apex_within",
#                      param=onlinexcms_pdp, main="bw = 5")


##############################################################################
#####################################################
# GCalignR study for the real data
##############################################################################
#####################################################


# we need the data organized in a list, where we will add it to a dataframe,
# with 2D chromatogram we will use the three read data before
# because the peaks will be represented as rectangles and the base will be 1, the enough time to get it,
# the area will be the same number as the intensities, due to that we will be multiplying the intensities per the base which will be one
msMatrixS3_2 <- data.frame(chromatographicTimeVector, rowSums(msMatrixS3))
colnames(msMatrixS3_2) <- c("time", "area")
msMatrixS4_13_2 <- data.frame(chromatographicTimeVector, rowSums(msMatrixS4_13))
colnames(msMatrixS4_13_2) <- c("time", "area")
msMatrixS4_16_2 <- data.frame(chromatographicTimeVector, rowSums(msMatrixS4_16))
colnames(msMatrixS4_16_2) <- c("time", "area")

dataList <- list()
dataList[["s3_27042022"]] <- msMatrixS3_2
dataList[["s4_13_27042022"]] <- msMatrixS4_13_2
dataList[["s4_16_06052022"]] <- msMatrixS4_16_2

peak_data_aligned <- align_chromatograms(data=dataList, # input data
                                         rt_col_name="time", # retention time variable name
                                         rt_cutoff_low=2, # remove peaks below 2 Minutes
                                         rt_cutoff_high=30, # remove peaks exceeding 30 Minutes
                                         reference=NULL, # choose automatically
                                         max_linear_shift=0.05, # max. shift for linear corrections
                                         max_diff_peak2mean=0.03, # max. distance of a peak to the mean across samples
                                         min_diff_peak2peak=0.03, # min. expected distance between peaks
                                         #blanks = "name", # negative control
                                         delete_single_peak=TRUE, # delete peaks that are present in just one sample
                                         write_output=NULL) # add variable names to write aligned data to text files

plot(peak_data_aligned$aligned_list$s3_27042022, type="l")
```

```
# visual diagnostics for the aligned data
# gc_heatmap(peak_data_aligned, threshold = 0.03)
plot(peak_data_aligned, which_plot = "all")

print(peak_data_aligned)

#################################################################
#####################################################
# GCalignR study for the simulation
#################################################################
#####################################################

# first of all, we want to get better the chromatograms, so we do a baseline correction
dataListSimulated$normalSimulated$area <- correctBaseline(dataListSimulated$normalSimulated$area, dataListSimulated$normalSimulated$time)
dataListSimulated$alteratedSimulated$area <- correctBaseline(dataListSimulated$alteratedSimulated$area, dataListSimulated$alteratedSimulated$time)
dataListSimulated$missingMetabolite$area <- correctBaseline(dataListSimulated$missingMetabolite$area, dataListSimulated$missingMetabolite$time)

# now, we will repeat the above steps but with the simulated data
# we use the list above created
peak_data_alignedSimulated <- align_chromatograms(data=dataListSimulated, # input data
                                  rt_col_name="time", # retention time variable name
                                  rt_cutoff_low=2, # remove peaks below 2 Minutes
                                  rt_cutoff_high=30, # remove peaks exceeding 30 Minutes
                                  reference=NULL, # choose automatically
                                  max_linear_shift=0.05, # max. shift for linear corrections
                                  max_diff_peak2mean=0.03, # max. distance of a peak to the mean across samples
                                  min_diff_peak2peak=0.03, # min. expected distance between peaks
                                  #blanks = "name", # negative control
                                  delete_single_peak=TRUE, # delete peaks that are present in just one sample
                                  write_output=NULL) # add variable names to write aligned data to text files

plot(peak_data_alignedSimulated$aligned_list$alteratedSimulated, type="l")

# visual diagnostics for the aligned data
plot(peak_data_alignedSimulated, which_plot = "all")

print(peak_data_alignedSimulated)
```

```
#############################################################################
#####################################################
# Algorithm created to study the real data
#############################################################################
#####################################################


# first step with the matrix information will be smoothing the data by reducing the noise and the baseline in case that it is there


# 1D representation of the GC-MS
matplot(msMatrixS3, type="l", ylab="intensity")
title(main="chromatogram of the msMatrixS3 GC-MS", xlab="retention time")


# 3D representation of the GC-MS
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(msMatrixS3), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="msMatrixS3 GC-MS")


# as we can see above, there is no need to do a baseline correction, but we can do a smooth
# for that we will identify the peaks and reduce the numbers that not correspond to any peak


# we will use the MassSpecWavelet package, but in order to use the functions we need to give one vector, so we will make
# a loop to read all the information and also reduce the numbers not chosen as peaks
scales <- seq(1, 64, 2)
SNR.Th <- 3 # signal to noise ratio
nearbyPeak <- TRUE


for (variable in 1:ncol(msMatrixS3)) {
  vectorMatrix <- as.vector(as.numeric(msMatrixS3[, variable]))
  wCoefs <- cwt(vectorMatrix, scales=scales, wavelet="mexh")
  # peak identification process
  # attach the raw spectrum as the first column
  wCoefs <- cbind(as.vector(vectorMatrix), wCoefs)
  colnames(wCoefs) <- c(0, scales)
  localMax <- getLocalMaximumCWT(wCoefs)


  ridgeList <- getRidge(localMax)


  # we check if it has been any ridge calculated
  if (ridgeList[1]!="NULL"){
    majorPeakInfo <- identifyMajorPeaks(
```

```
  vectorMatrix,
  ridgeList,
  wCoefs,
  SNR.Th = SNR.Th,
  nearbyPeak=nearbyPeak
)

peakIndex <- majorPeakInfo$peakIndex

# plotPeak(
#   vectorMatrix,
#   peakIndex,
#   range=c(1, length(vectorMatrix)),
#   main=paste('Identified peaks with SNR >', SNR.Th)
# )

# vectorMatrix[peakIndex] with this we can extract the intensities of the peaks
# peakIndex show us the position in the vector vectorMatrix of the peaks

# now, we reduce the others values
# first of all we create a logical vector the same length as vectorMatrix
logicalVector <- rep(TRUE, length(vectorMatrix))
peakIndexVector <- as.vector(peakIndex)

# now we change the index values in "logicalVector" to false
if (length(peakIndexVector)!=0){
 for (indexVariable in 1:length(peakIndexVector)) {
   if (indexVariable<5){
     logicalVector[peakIndexVector[indexVariable]] <- FALSE
   } else {
     logicalVector[(peakIndexVector[indexVariable]):(peakIndexVector[indexVariable]+5)] <- FALSE
     logicalVector[(peakIndexVector[indexVariable]-5):(peakIndexVector[indexVariable]-1)] <- FALSE
   }
 }
}
# and reduce the data not selected as peak and their nearest values
vectorMatrix[logicalVector] <- vectorMatrix[logicalVector]/1000

# plotPeak(
#   vectorMatrix,
#   peakIndex,
```

```
    #   range=c(0, length(vectorMatrix)),
    #   main=paste('Identified peaks with SNR >', SNR.Th)
    # )

    # now, we change the values in the matrix for the ones treated
    # only for the cases that the ridgeList is not null, in this last case, nothing will be done
    msMatrixS3[, variable] <- vectorMatrix
  }
}

# 1D representation of the GC-MS
matplot(msMatrixS3, type="l", ylab="intensity")
title(main="processed chromatogram of the msMatrixS3 GC-MS", xlab="retention time")

# 3D representation of the GC-MS
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(msMatrixS3), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="processed msMatrixS3 GC-MS")

# chromatographic peak detection
## before to continue, we will filter the data only with the mzs values that we are looking for
numbersIndex<- as.vector(msSpectraMetabolite[, 1])-20+1
msMatrixS3MZs <- msMatrixS3[, numbersIndex]

zeros <- FALSE
treatedRealData <- treatedMetabolite(zeros, msMatrixS3MZs, rtMetabolite, peakWidth)
chromatogramRealResults <- chromatogramFunction(treatedRealData, rtMetabolite, peakWidth)

# we plot the graphics
chromatogramRealResults$plots$intensitiesMzs
chromatogramRealResults$plots$intensitiesStacked
chromatogramRealResults$plots$plotMzsSeparately

# surfacePlot(chromatographicTimeVector, chromatogramRealResults$plotSurfaceInformation)



#first step with matrix information will be smoothing the data by reducing the noise and the baseline in case that it is there

# 2D representation of the GC-MS
matplot(msMatrixS3_2, type="l", ylab="intensity")
```

```
title(main="chromatogram of the msMatrixS3_2 GC-MS", xlab="retention time")


# 3D representation of the GC-MS
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(msMatrixS3_2), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="msMatrixS3_2 GC-MS")


# as we can see above, a baseline correction is not needed, and later a smooth is needed


# we will identify the peaks and reduce the numbers that not correspond to any peak
scales <- seq(1, 64, 2)
SNR.Th <- 3 # signal to noise ratio
nearbyPeak <- TRUE


for (variable in 1:ncol(msMatrixS3_2)) {
  vectorMatrix <- as.vector(as.numeric(msMatrixS3_2[, variable]))
  wCoefs <- cwt(vectorMatrix, scales=scales, wavelet="mexh")
  # peak identification process
  # attach the raw spectrum as the first column
  wCoefs <- cbind(as.vector(vectorMatrix), wCoefs)
  colnames(wCoefs) <- c(0, scales)
  localMax <- getLocalMaximumCWT(wCoefs)

  ridgeList <- getRidge(localMax)

  # we check if it has been any ridge calculated
  if (ridgeList[1]!="NULL"){
    majorPeakInfo <- identifyMajorPeaks(
      vectorMatrix,
      ridgeList,
      wCoefs,
      SNR.Th = SNR.Th,
      nearbyPeak=nearbyPeak
    )

    peakIndex <- majorPeakInfo$peakIndex

    # plotPeak(
    #   vectorMatrix,
    #   peakIndex,
    #   range=c(1, length(vectorMatrix)),
```

```
    #   main=paste('Identified peaks with SNR >', SNR.Th)
    # )

    # now, we reduce the others values
    # first of all we create a logical vector the same length as vectorMatrix
    logicalVector <- rep(TRUE, length(vectorMatrix))
    peakIndexVector <- as.vector(peakIndex)

    # now we change the index values in "logicalVector" to false
    if (length(peakIndexVector)!=0){
     for (indexVariable in 1:length(peakIndexVector)) {
       if (indexVariable<5){
         logicalVector[peakIndexVector[indexVariable]] <- FALSE
       } else {
         logicalVector[(peakIndexVector[indexVariable]):(peakIndexVector[indexVariable]+5)] <- FALSE
         logicalVector[(peakIndexVector[indexVariable]-5):(peakIndexVector[indexVariable]-1)] <- FALSE
       }
     }
    }

    # and reduce the data not selected as peak and their nearest values
    vectorMatrix[logicalVector] <- vectorMatrix[logicalVector]/1000

    # plotPeak(
    #   vectorMatrix,
    #   peakIndex,
    #   range=c(0, length(vectorMatrix)),
    #   main=paste('Identified peaks with SNR >', SNR.Th)
    # )

    # now, we change the values in the matrix for the ones treated
    # only for the cases that the ridgeList is not null
    msMatrixS3_2[, variable] <- vectorMatrix
  }
}

# 2D representation of the GC-MS
matplot(msMatrixS3_2, type="l", ylab="intensity")
title(main="processed chromatogram of the msMatrixS3_2 GC-MS", xlab="retention time")

# 3D representation of the GC-MS
```

```
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(msMatrixS3_2), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="processed msMatrixS3_2 GC-MS")


# chromatographic peak detection
## before to continue, we will filter the data only with the m/zs values that we are looking for
numbersIndex<- as.vector(msSpectraMetabolite[, 1])-20+1
msMatrixS3_2MZs <- msMatrixS3_2[, numbersIndex]


zeros <- FALSE
treatedRealData <- treatedMetabolite(zeros, msMatrixS3_2MZs, rtMetabolite, peakWidth)
chromatogramRealResults <- chromatogramFunction(treatedRealData, rtMetabolite, peakWidth)


# we plot the graphics
chromatogramRealResults$plots$intensitiesMzs
chromatogramRealResults$plots$intensitiesStacked
chromatogramRealResults$plots$plotMzsSeparately


# surfacePlot(chromatographicTimeVector, chromatogramRealResults$plotSurfaceInformation)


##################################################################################################
# Algorithm created to study the simulation
##################################################################################################


#first step with matrix information will be smoothing the data by reducing the noise and the baseline in case that it is there


# 2D representation of the GC-MS
matplot(gcmsMatrix, type="l", ylab="intensity")
title(main="chromatogram of the gcmsMatrix GC-MS", xlab="retention time")


# 3D representation of the GC-MS
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(gcmsMatrix), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="gcmsMatrix GC-MS")


# as we can see above, a baseline correction is needed, and later a smooth


# function to reduce the baseline drift
gcmsMatrix <- correctBaselineMatrix(gcmsMatrix, chromatographicTimeVector)
```

```
# 2D representation of the GC-MS
matplot(gcmsMatrix, type="l", ylab="intensity")
title(main="chromatogram of the gcmsMatrix GC-MS with corrected baseline drift", xlab="retention time")
# 3D representation of the GC-MS
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(gcmsMatrix), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="gcmsMatrix GC-MS with corrected baseline drift")

# we will identify the peaks and reduce the numbers that not correspond to any peak
scales <- seq(1, 64, 2)
SNR.Th <- 3 # signal to noise ratio
nearbyPeak <- TRUE

for (variable in 1:ncol(gcmsMatrix)) {
  vectorMatrix <- as.vector(as.numeric(gcmsMatrix[, variable]))
  wCoefs <- cwt(vectorMatrix, scales=scales, wavelet="mexh")
  # peak identification process
  # attach the raw spectrum as the first column
  wCoefs <- cbind(as.vector(vectorMatrix), wCoefs)
  colnames(wCoefs) <- c(0, scales)
  localMax <- getLocalMaximumCWT(wCoefs)

  ridgeList <- getRidge(localMax)

  # we check if it has been any ridge calculated
  if (ridgeList[1]!="NULL"){
    majorPeakInfo <- identifyMajorPeaks(
      vectorMatrix,
      ridgeList,
      wCoefs,
      SNR.Th = SNR.Th,
      nearbyPeak=nearbyPeak
    )

    peakIndex <- majorPeakInfo$peakIndex

    # plotPeak(
    #   vectorMatrix,
    #   peakIndex,
    #   range=c(1, length(vectorMatrix)),
    #   main=paste('Identified peaks with SNR >', SNR.Th)
```

```
    # )

    # now, we reduce the others values
    # first of all we create a logical vector the same length as vectorMatrix
    logicalVector <- rep(TRUE, length(vectorMatrix))
    peakIndexVector <- as.vector(peakIndex)

    # now we change the index values in "logicalVector" to false
    if (length(peakIndexVector)!=0){
      for (indexVariable in 1:length(peakIndexVector)) {
        if (indexVariable<5){
          logicalVector[peakIndexVector[indexVariable]] <- FALSE
        } else {
          logicalVector[(peakIndexVector[indexVariable]):(peakIndexVector[indexVariable]+5)] <- FALSE
          logicalVector[(peakIndexVector[indexVariable]-5):(peakIndexVector[indexVariable]-1)] <- FALSE
        }
      }
    }

    # and reduce the data not selected as peak and their nearest values
    vectorMatrix[logicalVector] <- vectorMatrix[logicalVector]/1000

    # plotPeak(
    #   vectorMatrix,
    #   peakIndex,
    #   range=c(0, length(vectorMatrix)),
    #   main=paste('Identified peaks with SNR >', SNR.Th)
    # )

    # now, we change the values in the matrix for the ones treated
    # only for the cases that the ridgeList is not null
    gcmsMatrix[, variable] <- vectorMatrix
  }
}

# 2D representation of the GC-MS
matplot(gcmsMatrix, type="l", ylab="intensity")
title(main="processed chromatogram of the gcmsMatrix GC-MS", xlab="retention time")

# 3D representation of the GC-MS
```

```
plot3D::persp3D(x=lengthNominalMZSimulated, y=lengthChromatographicVector, z=t(gcmsMatrix), theta=120, phi=25, xlab="mz Simulated", ylab="retention time", zlab="Intensity", cex.lab=0.8)
title(main="processed gcmsMatrix GC-MS")


# chromatographic peak detection
## before to continue, we will filter the data only with the m/zs values that we are looking for
numbersIndex<- as.vector(msSpectraMetabolite[, 1])-20+1
gcmsMatrixMZs <- gcmsMatrix[, numbersIndex]


zeros <- FALSE
treatedSimulatedData <- treatedMetabolite(zeros, gcmsMatrixMZs, rtMetabolite, peakWidth)
chromatogramSimulatedResults <- chromatogramFunction(treatedSimulatedData, rtMetabolite, peakWidth)


# we plot the graphics
chromatogramSimulatedResults$plots$intensitiesMzs
chromatogramSimulatedResults$plots$intensitiesStacked
chromatogramSimulatedResults$plots$plotMzsSeparately


# surfacePlot(chromatographicTimeVector, chromatogramSimulatedResults$plotSurfaceInformation)
```