Supporting Information

A Detailed Investigation and Comparison of the XCMS and MZmine 2 Chromatogram Construction and Chromatographic Peak Detection Methods for Preprocessing Mass Spectrometry Metabolomics Data

Owen Myers,[†] Susan Sumner,[‡] Shuzhao Li,[¶] Stephen Barnes,[§] and Xiuxia Du[*,†]

†*University of North Carolina at Charlotte, Charlotte, NC 28223, USA*

‡*University of North Carolina at Chapel Hill, Chapel Hill, NC 27514, USA*

¶*Emory University, Atlanta, GA 30322, USA*

§*University of Alabama at Birmingham, Birmingham, AL 35294, USA*

E-mail: xiuxia.du@uncc.edu

Phone: (704) 687-7307

9201 University City Blvd., Charlotte, NC 28223, USA

Contents

1 Non-Comprehensive List of Software Packages Developed for Preprocessing Metabolomics Data

Commercial tools include Progenesis QI (Waters),[1] SIEVE (ThermoFisher),[2] MassHunter (Agilent),[3] MarkerView (Sciex),[4] and ChromaTOF (LECO)[5] which have been developed by MS vendors as well as AnalyzerPro (SpectralWorks)[6] and MS Resolver[7] among others. Freely available software tools include XCMS,[8–10] MZmine,[11,12] MetAlign,[13–15] apLCMS,[16–19] MAVEN,[20,21] MetSign,[22] OpenMS,[23–25] MathDAMP,.[26,27] There are many more not included in this list.

2 Experimental Procedures

2.1 DCSM

DCSM is a standard mixture file that was generated from a mixture of 21 standard compounds. Section "Compounds Manually Confirmed in the DCSM File" of the Supporting Information lists these 21 compounds.

The standard compounds were purchased from Sigma Aldrich (St. Louis, MO). HPLC grade water (Sigma Aldrich, St. Louis, MO), DMSO (Sigma Aldrich St. Louis, MO), and methanol (Sigma Aldrich St. Louis, MO) were used in dissolving standards. Mobile phase A: Water with 0.1% formic acid (Sigma Aldrich). Mobile phase B: Methanol with 0.1% formic acid (Sigma Aldrich, St. Louis, MO).

Stock solutions of the metabolite standards were prepared by dissolving each compound (1mg/mL) in DMSO. A working mixture of metabolites ($100\,\mu g/mL$ of each) was prepared in ethanol. The working mixture was further diluted in water to have a final concentration of $20\,\mu g/mL$.

A $10\,\mu L$ of standard mixture ($20\,\mu g/mL$ each standard concentration) was injected into an OrbiTrap Velos mass spectrometer (Thermo Scientific, CA) coupled to an Acquity UPLC (Waters Corporation, MA). The metabolites were separated on a Waters Acquity HSS T3 column ($2.1\,mm \times 100\,mm$, $1.8\,\mu m$ particle size) operating at $50\,^{\circ}C$ using a reversed-phase chromato-

graphic method. A gradient mobile phase consisting of water with 0.1% formic acid (A) and methanol with 0.1% formic acid (B) were used as previously described (Dunn, et al., 2011). All MS data were collected over $120 - 1000\ m/z$ in ESI positive ion mode.

2.2 YP01, YP02, and VT001

YP01, YP02, and VT001 were all generated from NIST Standard Reference Material (SRM) 1950, a representation of human plasma.[28] Metabolomics analysis was performed similarly as previously described in Jin et al.[29] and Li et al.[30] For each sample, $65\,\mu\mathrm{L}$ of plasma was used and acetonitrile containing a mixture of 14 stable isotope internal standards was added to the aliquot at 2:1 in order to precipitate proteins. The samples were kept on ice for $30\,\mathrm{min}$ and then centrifuged for $10\,\mathrm{min}$ at $13,400$rpm at $4\,^{\circ}\mathrm{C}$. The supernatant was then removed and placed into autosampler vials. Mass spectral data were collected with a $10\,\mathrm{min}$ gradient on a Dionex UltiMate 3000 rapid separation LC system coupled with: 1) Thermo Q Exactive HF Orbitrap system (Thermo Fisher Scientific, San Diego, California) that generated the VT001 data file. Ions were scanned in the m/z range $85-1275$ in the positive ionization mode with a resolution of $120,000$; or 2) a LTQ Orbitrap Velos system, with m/z scan range $85 - 2000$, resolution $30,000$ for generating the YP01, YP02 data files . The chromatography was performed using either a reverse phase C18 column (YP02) or an anion exchange column (YP01). Mass spectra were acquired in profile mode and converted to CDF format using the Xcalibur file converter software (ThermoFisher Scientific).

We manually checked for compounds certified by NIST as being in the samples. A certified compound is defined by NIST as "... a value for which NIST has the highest confidence of its accuracy in that all known or suspected sources of bias have been investigated or taken into account."[28] In Sections "Compounds Manually Confirmed in . . ." of the Supporting Information we list the compounds found by manual investigation in all data files.

2.3 MAR17

This data file is part of the study named ST000045 in the Metabolomics Workbench.[31] Both the raw data file and the experimental information are available on the website. Briefly, plasma quality-control samples used in the study were prepared from pooled plasma spiked with a selection of metabolites to mimic elevated levels of metabolites during insulin withdrawn condition. Plasma was spiked with a standard mixture (3:1 ratio of plasma to spiking solution) containing $100\,\mu\mathrm{g\,mL^{-1}}$ niacin, hypoxanthine, leucine, isoleucine, phenylalanine, tryptophan, citric acid, glucose, hippuric acid, and taurocholic acid dissolved in 1:1 acetonitrile/water. All plasma samples ($200\,\mu\mathrm{L}$) were thawed on ice at $4\,^{\circ}\mathrm{C}$ followed by deproteinization with methanol (1:4 ratio of plasma to methanol) and vortexed for $10\,\mathrm{s}$, followed by incubation at $-20\,^{\circ}\mathrm{C}$ for $2\,\mathrm{h}$. The samples were then centrifuged at $15,871\mathrm{g}$ for $30\,\mathrm{min}$ at $4\,^{\circ}\mathrm{C}$. The supernatants were lyophilized (Savant, Holbrook, NY) and stored at $-20\,^{\circ}\mathrm{C}$ prior to analysis. The samples were reconstituted in 50% H2O/acetonitrile and passed through a Microcon YM3 filter (Millipore Corporation). The supernatants were transferred to analytical vials, stored in the autosampler at $4\,^{\circ}\mathrm{C}$, and analyzed within $48\,\mathrm{h}$ of reconstitution in buffer.

The liquid chromatography platform consisted of an Acquity UPLC system (Waters, Milford, MA). Plasma metabolite separation was achieved using hydrophilic interaction chromatography (HILIC). The run time was $20\,\mathrm{min}$ at a flow rate of $400\,\mu\mathrm{L\,min^{-1}}$. The HILIC gradient was as follows: $0\,\mathrm{min}$, 100% B; $1\,\mathrm{min}$, 100% B; $5\,\mathrm{min}$, 90% B; $13.0\,\mathrm{min}$, 0% B; $16\,\mathrm{min}$, 0% B; $16.5\,\mathrm{min}$, 100% B; and $20\,\mathrm{min}$, 100% B. Other LC parameters were injection volume $5\,\mu\mathrm{L}$ and column temperature $50\,^{\circ}\mathrm{C}$.

A 6220 TOF MS (Agilent Technologies) was operated in negative electrospray ionization modes using a scan range of 50 to $1,200$ m/z. The mass accuracy and mass resolution were less than 5 (ppm) and $\sim 20,000$, respectively. The instrument settings were as follows: nebulizer gas temperature $325\,^{\circ}\mathrm{C}$, capillary voltage $3.5\,\mathrm{kV}$, capillary temperature $300\,^{\circ}\mathrm{C}$, fragmentor voltage $150\,\mathrm{V}$, skimmer voltage $58\,\mathrm{V}$, octapole voltage $250\,\mathrm{V}$, cycle time $0.5\,\mathrm{s}$, and run time $15.0\,\mathrm{min}$.

3 Preprocessing Parameters Used by XCMS and MZmine 2

Preprocessing parameters used by XCMS and MZmine 2 are shown in Table S1 and Table S2.

Table S1: XCMS parameters used for EIC building and peak picking

Parameters	**DCSM**	**YP01 and YP02**	**VT001**
method	centWave	centWave	centWave
mzTolerance	0.01	0.01	0.01
peakwidth	(1.2,36.0)	(1.2,60.0)	(0.6,30.0)
snthresh	10	10	10
prefilter	(1,5000)	(1,1000)	(1,1000)
mzCenterFun	wMean	wMean	wMean
integrate	2	2	2
mzdiff	-0.001	-0.001	-0.001
fitgauss	F	F	F
noise	100	100	100
sleep	0	0	0

Table S2: MZmine2 parameters used for EIC building and peak picking

Parameters	**DCSM**	**YP01 and YP02**	**VT001**
Chromatogram builder			
Min time span	0.04	0.05	0.01
Min height	5000	1000	1000
m/z *tolerance*	0.01	0.01	0.01
Chromatogram deconvolution			
S/N *threshold*	10	10	10
Wavelet scales	0.02-0.80	0.02-1.20	0.01-0.5
Peak duration range	0.02-0.60	0.05-1.00	0.01-0.5

4 EIC Construction by XCMS

Chromatogram building is done when the "find peaks" method is called. These EICs are referred to as the Regions of Interest (ROIs). The list of ROIs is a list of m/z and RT boundaries that contain the data comprising the EIC. Here we give a pedagogical pseudocode version of the algorithm used to build the ROIs, the purpose of which is to serve as an aid to a reader investigating the XCMS source code. We separate the important steps in a way that mirrors the actual source code. Variable

and function names are in italics and each piece of pseudocode has a corresponding itemized list of the variable descriptions. Each piece of pseudocode also has an associated table which equates the pseudocode variables and function names to the original XCMS when a one-to-one correspondence exists.

4.1 Most abstract overview of XCMS *findmzROI* method

4.1.1 Pseudocode

for curScan in allScans **do**
    sortedScan ← sortByInt(curScan)
    for dataPoint in sortedScan **do**
        if dataPoint.int $>$ noise **then**
            insertDataPoint(dataPoint)
        end if
    end forcleanUp()
end for

4.1.2 Description of Variables

- *curScan* → the current scan, data and scan number.
- *allScans* → a list of all of the scans in the data file.
- *sortedScan* → a scan sorted by the intensities.
- *sortByInt()* → function that sorts a scan by the intensities
- *dataPoint* → an object that contains the intensity and m/z value of the current point in the scan. Has attributes *dataPoint.int* → the intensity, and *dataPoint.mz* → the m/z value.
- *insertDataPoint* → See subsection 4.2.1.
- *cleanUp* → See subsection 4.3.

Table S3 shows the correspondence between the pseudocode variables and original variables in the script.

Table S3: Pseudocode variables to original variables correspondence.

Pseudocode	**Original**
curScan	*ctScan*
allScans	NA
sortedScan	*scanbuf*
dataPoint	*fMass & fInten*
cleanUp()	*cleanup(...)*

4.2 Insert Peak Function

4.2.1 Pseudocode

lpos $\leftarrow$ indexROIsAt(mzIn - mztol)

hpos $\leftarrow$ indexROIsAt(mzIn + mztol)

for i in range such that

(lpos $\geq$ rois[i].mzMean $<$ hpos) **do**

    diff $\leftarrow$ abs(mzIn - rois[i].mzMean)

    if diff $\leq$ mztol **then**

        updateMzMean(mzIn)

        if mzIn $<$ rois[i].mzmin **then**

            rois[i].min = mzIn

        end if

        if mzIn $>$ rois[i].mzmax **then**

            rois[i].max = mzIn

        end if

        rois[i].int += intensityOf(mzIn)

        wasFound = True

    end if

end for

if wasFound == False **then**

    if point in next scan within toleracne of curent point? **then**

```
        newROI = initilizeNewROI(mzIn)
        rois.add(newROI)
      end if
    end if
```

4.2.2 Description of Variables

- *mzIn* $\rightarrow$ the m/z value passed into the insert peak functions.
- *indexROIsAt*(m/z value) $\rightarrow$ the list of the ROIs index at the input m/z value.
- *lpos* and *hpos* $\rightarrow$ the min and max indices of ROIs that fall within the *mzIn* and the tolerance.
- *updateMzMean*(m/z value) $\rightarrow$ update the current ROIs mean m/z since a point is added.
- *mztol* $\rightarrow$ the m/z tolerance specified by the user.
- *intensityOf*(m/z value) $\rightarrow$ find the intensity corresponding to this data point.
- *rois* $\rightarrow$ a list/array of the regions of interests. Each object in the list/array has the attributes:
  - *mzMean* $\rightarrow$ Mean m/z of all points in ROI.
  - *mzmin* $\rightarrow$ minimum m/z of all points in ROI.
  - *mzmax* $\rightarrow$ maximum m/z of all points in ROI.
  - *int* $\rightarrow$ sum of individual intensities over all points in ROI.
  - *scmin* $\rightarrow$ minimum scan number.
  - *scmin* $\rightarrow$ maximum scan number.
- *initilizeNewROI*(m/z value) $\rightarrow$ make a new ROI with the m/z value
- *rois.add*(an ROI) $\rightarrow$ add the argument ROI to the list of ROIs.

Table S4 shows the correspondence between the pseudocode variables and original variables in the script.

4.3 Cleanup Function

After every scan, the function "cleanup" is performed. This function gets rid of EIC's which are too short (in intensity and RT) and have not had points added to them during the processing of the last scan.

Table S4: Pseudocode variables to original variables correspondence.

Pseudocode	Original
mzIn	*fMass*
tolerance	*ddev*
rois	*mzval*
lpos	*lpos*
hpos	*lpos*
diff	*ddiff*
wasFound	*wasfound*
mzMean	*mz*
mzmin	*mzmin*
mzmax	*mzmax*

4.3.1 Pseudocode

```
toKeepROIs ← empty list of ROIs.
for curROI in allROIs do
    if (length(curROI)>=minPts)
    or(curROI.lastScan==thisScan) then
        if  (length(curROI)>=minPts)
        and(curROI.lastScan<thisScan) then
            if curROI.numBigInt>=minBigInt then
                toKeepROIs.add(curROI)
            else
                Throw away curROI
            end if
        end if
    else
        Throw away curROI
    end if
end for
return toKeepROIs
```

4.3.2 Description of Variables

- *toKeepROIs* → a list to store only the ROIs that are going to be kept and returned at the end of the function.
- *toKeepROIs.add*(an ROI) → adds the argument ROI to the *toKeepROIs* list.
- *curROI* → the current ROI from the list *allROIs*. Has attribute *lastScan*.
- *curROI.lastScan* → the scan number of the points last added to the ROI. This can be compared to *thisScan*.
- *thisScan* → the clean up function is called after every scan has been processed. *thisScan* is the scan number that this function is currently being called.
- *allROIs* → a list of all the ROIs passed into the cleanup function.
- *length*(an ROI) → returns the number of scans in the argument ROI.
- *minPts* → a threshold number of points to compare the length (in RT) of a EIC to. Calculated from the minimum peak width specified by the user.
- *curROI.numBigInt* → the number of points in the *curROI* that have intensities over a specified value.
- *minBigInt* → a threshold number of large intensity points to compare against an ROI's *numBigInt* (above).

Table S5 shows the correspondence between the pseudocode variables and original variables in the script.

Table S5: Pseudocode variables to original variables correspondence.

Pseudocode	**Original**
toKeepROIs	*mzROI*
curROI	*mzval[i]*
curROI.lastScan	*lastscan (mzval[i].scmax)*
thisScan	*ctScan*
allROIs	*mzval*
length(curROI)	*entries (mzval[i].length)*
minPts	*pickOptions->minEntries*
curROI.numBigInt	*mzval[i].kI*
minBigInt	*pickOptions->minimumIntValues*

5 EIC Construction by MZmine 2

5.1 Most abstract view of building EICs

5.1.1 Pseudocode

finalChroms ← empty list of EIC

chromBuilder ← EIC building object

for scan in allScans **do**

    chromBuilder.addScan(scan)

end for

finalChroms ← chromBuilder.finishChroms()

5.1.2 Description of Variables

- *finalChroms* → initialize an empty list of EICs objects. This will become the final list of completed final EICs.
- *chromBuilder* → serves as an interface for the EIC building. Contains a list of EICs that is updated every time a scan is added. See below for description of *addScan*. Also see below for a description of *finishChroms()*

Table S6 shows the correspondence between the pseudocode variables and original variables in the script.

Table S6: Pseudocode variables to original variables correspondence.

Pseudocode	**Original**
finalChroms	*EICs*
chromBuilder	*massConnector*

Note: *massConnector* is of type *HighestDataPointConnector*.

5.2 Details of *addScan()*

5.2.1 Pseudocode

```
allChroms ← all EICs
sortedScan ← sorted current scans
connectedChroms ← empty EICs list
for mzVal in sortedScan do
    bestChrom ← empty EIC
    for testChrom in allChroms do
        if mzVal in
        testChrom.lastPntMz±mzTol then
            if bestChrom is empty or
            testChrom.lastPntInt>
            bestChrom.lastPntInt then
                bestChrom = testChrom
            end if
        end if
    end for
    if bestChrom == null then
        bestChrom = new EIC obj.
    else if bestChrom in
    connectedChroms then
        continue
    end if
    bestChrom.add(mzVal)
    connectedChroms.add(bestChrom)
end for
```

for testChrom in allChroms **do**

if testChrom in connectedChroms **then**

continue

end if

if length(testChrom) $>$ minPnts **then**

testChrom.numBuildingSegments $+ = 1$

connectedChroms.add(testChrom)

continue

end if

if testChrom.numBuildingSegments > 0 **then**

connectedChroms.add(testChrom)

end if

end for

buildingChromatograms $\leftarrow$ connectedChromatograms

Description of the last loop and why this processing step is done. The first **if** statement makes sure that only EICs for which NONE of the m/z values in the scan were added to are worked with. The second **if** statement adds the EIC to the list of connected ones only if it is long enough. To elaborate, if a EIC is begun at some point when addScan is called and a couple of m/z values are added to it for a couple of proceeding calls of addScan but then nothing is added to it, the length will be 2 (for example) and if no mass is added on this addScan call, if 2<minimumTimeSpan then this EIC will be thrown away. The third if statement adds it if it has a committed segment that was committed for some other reason than the prior if statement.

5.2.2 Description of the variables

- *allChroms* $\rightarrow$ a set of all the EICs current EICs.
- *sortedScan* $\rightarrow$ the current scan sorted by intensity.
- *connectedChroms* $\rightarrow$ initialized as an empty list of EICs this will store EICs that have had

points added to them during the processing.

- *mzVal* $\rightarrow$ the current m/z value from the sorted scan.
- *bestChrom* $\rightarrow$ Chromatograms who's last point falls within the tolerance range of *mzVal* will are considered. *bestChrom* is updated as these EICs are checked for the one containing the highest intensity last added point.
- *testChrom* $\rightarrow$ the current EIC in the enumeration of a EIC list.
- *length()* $\rightarrow$ function that gets the length in RT of a EIC.
- *add()* $\rightarrow$ function that either adds an m/z value to a EIC or adds a EIC to a list of EICs.

Table S7 shows the correspondence between the pseudocode variables and original variables in the script.

Table S7: Pseudocode variables to original variables correspondence.

Pseudocode	**Original**
allChroms	*buildingChromatograms* (in *HighestDataPointConnector*)
sortedScan	*mzValues* (in *HighestDataPointConnector addScan()*)
connectedChroms	*connectedChromatograms* (in *HighestDataPointConnector addScan*)
mzVal	*mzPeak*
bestChrom	*bestChromatogram* (in *HighestDataPointConnector addScan*)
testChrom	*testChrom* (in *HighestDataPointConnector addScan*)
length()	⟨Chromatogram object⟩ *.getBuildingSegmentLength()*
add()	*add()*

6 Detailed Description of XCMS EIC Peak Detection

This section is parallel to Section "Results and Discussion" subsection "XCMS Detection of EIC peaks" in the main text. This section describes the same material but in much more detail.

XCMS calls an EIC found through its EIC construction process a region of interest (ROI). In this section and the subsequent section "Detailed Explanation of Examples of False Positive Detection", we will refer to these built EICs as ROIs to distinguish them from temporary EICs that are created for estimating the baseline (to be explained) and signal-to-noise ratios of ROIs and peaks.

Not all ROIs are passed to the wavelet transform method for peak detection. Before the wavelet transform, XCMS discards ROIs in two different steps based on estimates of the ROI baseline and noise. After the wavelet transform, XCMS examines the wavelet coefficients and the intensities of the detected peaks and determines whether or not each peak is valid. A total of five sequential filters are used to filter out bad ROIs and peaks.

Three temporary EICs are used throughout the *centWave* filtering steps and all of them are made using the ROI's m/z range. The first temporary EIC, which we call *tEIC1*, is created by extending the ROI on both sides by either the *minimum peak width* times three, or the maximum wavelet scale times three, whichever is larger in number of scans. The second temporary EIC, *tEIC2*, is created by extending the ROI to span the entire RT domain. The third temporary EIC, *tEIC3*, is the ROI extended by the *minimum peak width* divided by two (rounded down to the closest integer in number of scans) on each side. Next we describe the five sequential filters.

6.1 Number of consecutive data points above an estimate of the ROI baseline

This is the first of the sequential filters. XCMS estimates the baseline of a given ROI by using either *tEIC1* or *tEIC2*. In the case of a particularly long ROI, which *centWave* defines to be an ROI with number of points greater than or equal to ten times the user defined *minimum peak width* in

number of scans, XCMS uses *tEIC2*. In the case of an ROI that is not long, XCMS uses *tEIC1*. The estimate of the ROI baseline based on either *tEIC1* or *tEIC2* is as follows. If the number of non-zero intensity points in the temporary EIC is less than three times the *minimum peak width* in number of scans, then the baseline is just the arithmetic mean of the intensities. Otherwise it is the arithmetic mean after 10% of the total number of points are removed. Of the removed points, half of them are the largest intensity points and the other half are the smallest intensity points.

With the ROI baseline estimated, XCMS counts the number of sequential points above the baseline in *tEIC3*. If the largest number of sequential points above the baseline is found to be greater than or equal to either four (hard coded), or the *minimum peak width* minus two, whichever is larger, the ROI continues to the next step, otherwise it is discarded. If the *minimum peak width* is set by the user to be less than six scans or is not set, then it is effectively set to six scans within XCMS.

6.2 Signal-to-Noise Ratios of ROIs

Additionally, an ROI must also pass a S/N (signal-to-noise ratio) check in order to be passed to the wavelet transform. XCMS estimates the noise using two values: $lnoise_{\mathrm{std}}$ and $lnoise_{\mathrm{mean}}$ which are found as follows. If *tEIC3* spans the entire RT range, XCMS removes 10% of the total number of data points with half of the removed points being the largest intensity points and the other half being the smallest intensity points. From these remaining points $lnoise_{\mathrm{std}}$ and $lnoise_{\mathrm{mean}}$ are the calculated standard deviation and mean respectively. If *tEIC3* does not span the entire RT range, $lnoise_{\mathrm{std}}$ and $lnoise_{\mathrm{mean}}$ are calculated from data points that fall immediately outside the retention time of the ROI. Specifically, they are found using two separate sets of points and the final $lnoise_{\mathrm{std}}$ and $lnoise_{\mathrm{mean}}$ are both taken to be the smaller value calculated from the two sets.

The first set of points, which we will call *set1*, are the data points in *tEIC1* which do not also belong to *tEIC3*. The second set of points, *set2*, are those beyond the boundaries of *tEIC3* by the *minimum peak width* times three scans. From both *set1* and *set2*, XCMS then removes groups of points with either four or *minimum peak width* minus two (whichever is greater) consecutive

neighbors which are above the baseline. Subsequently, the mean and standard deviation of the remaining points are calculated from *set1* and *set2*.

One problem exists in XCMS's determination of $lnoise_{\mathrm{std}}$ and $lnoise_{\mathrm{mean}}$. If no points or only a single point is found to be below the baseline in either *set1* or *set2* then the final $lnoise_{\mathrm{std}}$ and $lnoise_{\mathrm{mean}}$ will both be set to one. As we will show in the proceeding section, this can cause the identification of false positives because it results in a great under-estimate of the local noise.

Eventually, XCMS determines whether or not an ROI will be discarded based on two values: $lnoise_{\mathrm{mean}}$ and a value called *sdthr* (variable name in *centWave* code). The *sdthr* value is found by taking the S/N threshold set by the user and multiplying it by $lnoise_{\mathrm{std}}$. If the maximum intensity of *tEIC3* minus $lnoise_{\mathrm{mean}}$ is not greater than or equal to the *sdthr* value, the ROI is discarded.

6.3 Wavelet Transform and the First Wavelet Coefficient Check

For ROIs that have passed both of the two filters described so far, XCMS passes the corresponding *tEIC1*, not the ROIs themselves, to the wavelet transform method. As a result, it is possible to detect peaks that do not meet all of the criteria of the EIC construction algorithm. We do not claim that this is necessarily good or bad, but it is important to be aware of this property of the *centWave* algorithm.

After the CWT has been performed, another S/N check occurs using the wavelet transform coefficients. For each ROI, $lnoise_{\mathrm{mean}}$ is subtracted from the largest wavelet coefficient. Only if this value is greater than or equal to *sdthr* will this ROI be further considered.

We must note that this comparison of wavelet coefficients with $lnoise_{\mathrm{mean}}$ and *sdthr* is not valid as it mixes two entirely separate quantities. The coefficients from a CWT are found through an integral transformation which produces a quantity with different units from that of the intensities in an EIC. Even if this check correctly filters out a select few false positives, it is not a valid way of filtering and its results are unpredictable. In Section "Detailed Explanation of Examples of False Positive Detection" of the Supporting Information, we show an example of it producing a false negative.

6.4 Ridgeline Detection and the Second Wavelet Coefficient Check

True EIC peaks should produce local maxima in the wavelet coefficients at a continuum of wavelet scales and these maxima should be relatively close together in the RT domain. As a result, these maxima form the so-called ridgeline. Ridgeline detection is the process that connects local maximum wavelet coefficients across different wavelet scales that are close in RT. Requiring local maximum wavelet coefficients to form a ridgeline across a number of consecutive scales is a robust way of detecting EIC peaks. However, *centWave* does not impose a length requirement on the ridgelines. Instead, ridgelines of any length are accepted. Consequently, many false positive peaks could be produced.

After ridgelines are detected, *centWave* carries out the fourth filtering step. Specifically, a more exclusive version of the wavelet coefficient check from above is performed using a restricted set of coefficients. For all points in a ridgeline, the coefficients at the smallest scale corresponding to those points' RT are considered. For example, if the smallest scale of a CWT is 3 and a ridgeline contains 3 connected local maxima at 1.00, 1.01 and 1.00 minutes for scales 4, 5 and 6 respectively, the coefficients at scale 3 (smallest scale) for times 1.00, 1.01 and 1.00 are considered. Then $lnoise_{\mathrm{mean}}$ is subtracted from these values and the result is checked to see if it is greater than *sdthr*. Each ridgeline must have one point satisfying this condition to be further considered. We do not believe that this is a meaningful comparison and have no insight into the reasoning behind it.

6.5 ROI Intensity Check

After ridgeline detection where individual EIC peaks have been detected, a final S/N check is performed. For each ridgeline the maximum wavelet coefficient at different scales may be at different RT. *centWave* checks the intensity of the ROI at all of the corresponding RTs. At least one intensity minus $lnoise_{\mathrm{mean}}$ must be greater than *sdthr* for the EIC peak to be considered a peak.

7 Determination of Peak Boundaries by XCMS

In XCMS the initial estimates of the peak boundaries are the locations for which the largest wavelet coefficient is found over all the wavelet scales plus/minus the number of scans equal to the best scale of the wavelet.[9] This estimate is then refined to the location of the smallest intensity point between the left (right) boundary and the left (right) boundary minus (plus) half the minimum peak width specified by the user. If multiple zero intensity points lie on either side of the peak, and a boundary is beyond the first zero intensity point from the edge of the EIC peak, the boundary is shrunk to that first zero intensity point.

8 MZmine 2: Problem Using *centWave*

Figure S1 shows an example of an EIC (blue) and the result of the distortion (red) caused by an error in how MZmine 2 passes the information to the *centwave* algorithm. The peak shown in Figure S1 can be found in data file YP01 with a representative m/z value of 103.507.

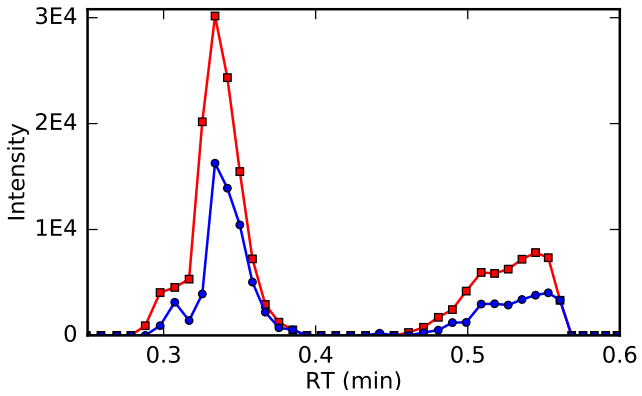

Figure S1: Original EIC (blue) to be passed into the *centWave* algorithm and distorted EIC (red) that is actually passed to *centWave* by MZmine 2.

The cause of this distortion lies in the way MZmine 2 incorrectly sets the *xcmsRaw* object environment variable that XCMS uses to store both the raw scan data as well as intermediate and final preprocessing results. MZmine 2 sets the XCMS environment variable *xraw@env$intensities*

to be the intensities of the EIC. However, XCMS expects *xraw@env$intensities* to be a one dimensional array containing all intensities in the file. This means that XCMS expects adjacent data points in *xraw@env$intensities* to be adjacent points in a *scan* (disregarding the very beginning or end of a scan). When XCMS calls the *rawEIC* function to get the EIC that will be passed into the wavelet transform, it tries to add adjacent points in the scan if they fall within the m/z range passed into the function. To find the indices associated with the upper and lower m/z bounds of the m/z range it calls two functions, *lowerBound* and *upperBound* which will return two different indices, the one above and below the m/z value specified. XCMS checks the m/z values between these indices and adds the intensities of the points that fall within the m/z range. When MZmine 2 data is passed in, the m/z range comprises an upper and lower bound that are exactly the same as one another. However, the *lowerBound* and *upperBound* function returns two different indices differing by one. Since MZmine 2 has set *xraw@env$intensities* to be the intensities of the EIC, but as we mentioned the adjacent indices refer to adjacent scans *not* to adjacent points in the same scan. Because the adjacent points in the scans all have the exact same m/z values (as MZmine 2 set them) they meet the m/z tolerance criteria in the loop and are summed. This is responsible for the distortion shown in Figure 5 in the main text.

We would like to note that this distorted EIC is used only inside *centWave* and therefore does not affect the relative quantitation (by using the area under the curve of an EIC peak) of the corresponding analyte by MZmine 2 since peak area is calculated outside of *centWave*.

9 Detailed Explanation of Examples of False Positive Detection

This section runs parallel to the same named section in the main text. This version goes in to further detail than the section in the main text.

Figure S2 explains how the part shaded in red in Figure 2(A) of the main text was incorrectly detected as an EIC peak. The black curve in Figure 2(A) is only a part of the full ROI. The full ROI is shown in Figure S2(A). The user-defined *minimum peak width* and S/N are 4 and 10,

respectively. The maximum wavelet scale is derived from the peak width parameter and the result is 57. The ROI in Figure S2(A) is considered long since its length is greater than 10 times the user-defined *minimum peak width* so XCMS uses the full RT range (*tEIC2*) to estimate the ROI baseline as the mean of all data points in *tEIC2*. *tEIC2* is shown in Figure S2(B) and the estimated baseline is shown as a red line. Figure S2(C) shows *tEIC1* which is the extension of the ROI on one side (because the other side is already at the maximum RT) by 171 scans. This number of scans, 171, is three times the maximum wavelet scale and is chosen since this value is larger than three times the *minimum peak width*. *tEIC3* is produced by extending the ROI on both sides by the *minimum peak width* divided by two. As a result, *tEIC3* has one single extra point on the left side than the ROI and ends at the same RT. We refer to panel (A) as the illustration of *tEIC3* as well as the full ROI because *tEIC3* is only one point longer on the left side than the ROI. With the estimated baseline, this ROI passes the check for the number of consecutive data points above the baseline because the largest number of sequential points above the baseline is found to be 154 which is greater than the threshold, 4 (the larger number between the hard coded four and *minimum peak width* minus two).

For the S/N check, $\mathit{lnoise}_{\mathrm{mean}}$ must be found. Figure S2(D) and (E) display the initial *set1* and *set2* points, respectively, in black. After groups of four (i.e., four or the *minimum peak width* minus two, whichever is greater) consecutive points are removed from both *set1* and *set2*, the remaining points are marked in red. These red points are used to calculate the final $\mathit{lnoise}_{\mathrm{mean}}$. The reader will notice some red points above the threshold (red line). This is because only groups of consecutive points (in this case length 4) above the threshold are removed from *set1* and *set2*. Since only one point remains after the filtering of *set2*, the mean and standard deviation values of *set2* are both set to be 1. This makes the final $\mathit{lnoise}_{\mathrm{mean}}$ of the ROI 1 since it is smaller than the $\mathit{lnoise}_{\mathrm{mean}}$ for *set1*. As a result, the final *sdthr* will be 10 ($\mathit{lnoise}_{\mathrm{mean}} \times S/N$). With such a small $\mathit{lnoise}_{\mathrm{mean}}$ and *sdthr*, clearly this peak will pass the S/N check.

Since this ROI passes both the S/N check and the check for the number of consecutive data points above the ROI baseline, it is passed to the subsequent wavelet transform which detects the area shaded in red in Figure S2(A)-(C) as a peak. Initially it was surprising that this portion of the

ROI should have been found as a peak by the CWT since it is unlikely that it could produce a large inner product with the wavelet at a variety of scales. But from our investigation of *centWave* we found that XCMS considers ridgelines with a minimum length of 1 as possible peaks. Figure S2(F) demonstrates this finding by showing the wavelet coefficients and the ridgelines detected from them. The area shaded in red in Figure S2(A)-(C) corresponds to the ridgeline pointed to by the black arrow. This ridgeline is of length of 1. We consider this requirement of the minimum length of ridgeline lines to be 1 as a mistake because the purpose of the CWT and ridgeline method is to detect peaks that produce maxima in the wavelet coefficients at a large enough number of consecutive wavelet scales that peaks like the one in Figure S2(A) can be filtered out.

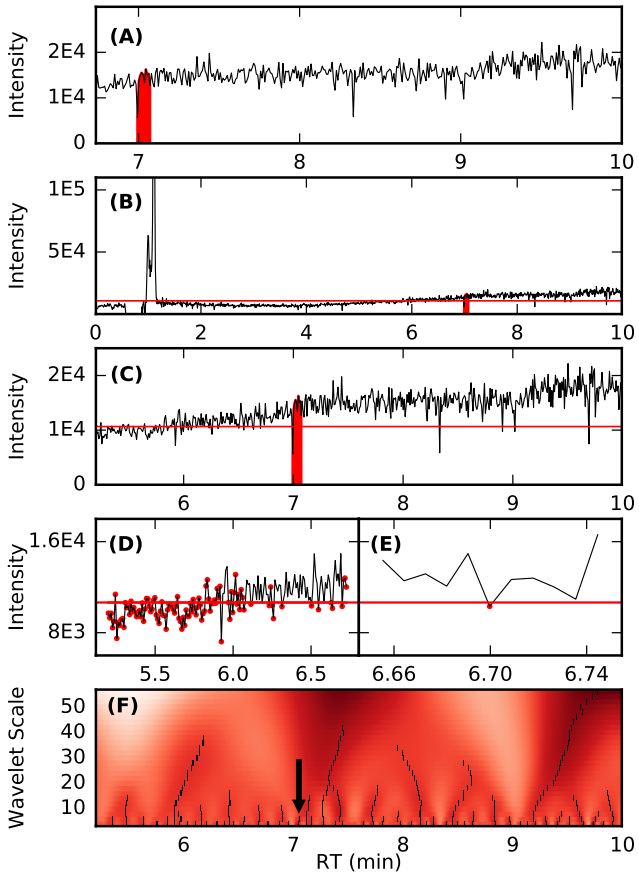

Figure S2: (A) The full ROI. *tEIC3* overlaps with the full ROI except that it has a single extra point on the left side. (B) Full retention time *tEIC2* used to estimate the baseline of the ROI. The estimated baseline is shown in red. (C) *tEIC1* (black) and the baseline (red). (D) and (E) show *set1* and *set2*, respectively. The red colored points in (D) and (E) are the remaining points in *set1* and *set2* after groups of consecutive points are removed. They are used for calculating the $lnoise_{\mathrm{mean}}$ and $lnoise_{\mathrm{std}}$. (F) Color map of the CWT coefficients found from *tEIC1*. Dark red corresponds to large coefficients and light red corresponds to small coefficients. The pixelated lines correspond to those coefficients which belong to the ridgelines constructed by XCMS.

Figure 2(B) in the main text shows a complete ROI (non-zero intensity points) and surrounding extensions that are used to estimate $lnoise_{\mathrm{mean}}$ and $lnoise_{\mathrm{std}}$. A portion, shaded in red, of the ROI's left half is found to be a peak by XCMS since this false peak passes all the filters. Firstly, the ROI passes the S/N check because the $lnoise_{\mathrm{mean}}$ is small ($= 1$) when there are primarily zero intensity values in both the left and right extensions of the ROI. Secondly, the wavelet coefficients of the shaded area passes the separate S/N check due to such a small $lnoise_{\mathrm{mean}}$. Thirdly, there is a ridgeline, pointed to by the black arrow in Figure S3 (Top) that is responsible for the detection

of this false EIC peak that is shown again in Figure S3 (Bottom) as the area shaded in red. This ridgeline comprises a single local maximum in the wavelet coefficients of the smallest scale only. Since *centWave* does not impose a large enough length of the ridgeline, this EIC peak is detected.

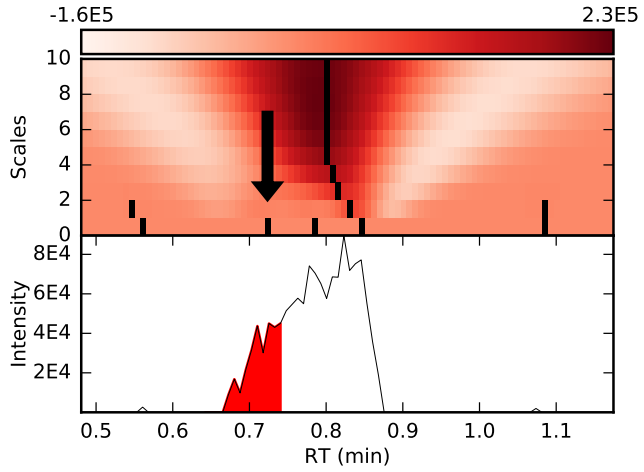

Figure S3: (Top) Wavelet coefficients at different scales are shown as a heat map for the EIC peak shown in Figure 2(B) of the main text. The color bar showing the scale of the heat map is directly above it. Ridgelines detected by XCMS are the black lines. The ridgeline pointed to by the arrow is responsible for the false positive peak shaded in red in the bottom panel. (Bottom) The EIC that generated the wavelet coefficients.

The peak shown in Figure 2(C) of the main text is found for the same reason as the peak in panel (A). Panel (C) shows a segment of a large EIC but one that is also surrounded by zero intensity data points. The surrounding points make the $lnoise_{\mathrm{mean}}$ small (~ 30). The *sdthr* value is about 1357 which is not large enough to prevent this peak from passing the S/N check.

In Figure 2(D) of the main text, $lnoise_{\mathrm{std}}$ is small (~ 70) so the *sdthr* value is small (~ 700). The $lnoise_{\mathrm{mean}}$ is also small because of the adjacent zero intensity values. As a result, the ROI passes the S/N check and eventually is detected as a peak.

10 The Effect of False Positive Peaks on Compound Identification

In Figure S4 we show a false positive peak and its isotopic distribution that leads to a false compound identification. This example's monoisotopic mass is ~ 520.337 and it was identified as $C_{27}H_{40}F_3N_7$ from the YP01 data file. It is clear that this peak detection error propagates through to compound identification. In table S8 we show the percentage of compounds identified from real EIC peaks. Though the majority of compounds detected are determined from real peaks, there still remain a troubling number that are determined from false peaks.

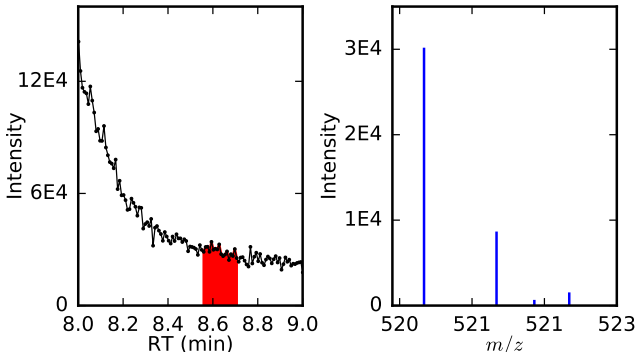

Figure S4: An example of a false positive peak (left) and its isotopic distribution (right) that results in the identification of a compound through the use of the CAMERA software package and a PubChem database search.

Table S8: Percentage of compounds identified from real peaks.

Data File	% Compounds from real peaks
DCSM	91.7
YP01	75.0
YP02	71.3
VT001	62.9

11 Methods and Parameters for Compound Identification

We use the software package CAMERA which can be found in the MZmine 2 framework for the clustering of peaks into isotopic distributions. The CAMERA parameters used for all data files are shown in Table S9. The database search was then performed using the PubChem database. The parameters used for the database search were *Mass Tolerance (m/z)* of 0.001, and *Similarity Tolerance* of 0.2.

Table S9: Parameters used with the CAMERA algorithm.

Parameter Name	**Value**
FWHM sigma	1
FWHM percentage	100%
Isotopes max. charge	1
Isotopes mass tolerance	4
Correlation threshold	0.001 m/z, 0.0 ppm
Correlation p-value	0.05
Group peaks by	Isotope ID

12 Compounds Manually Confirmed in the DCSM File

Table S10 lists the standard compounds that have been manually confirmed to be present in the DCSM file that is generated from a standard mixture sample. For easy reference, we give each compound a short reference name.

To determine if a compound is in the data file, we checked the mass error (i.e., the difference between the exact mass and the measured mass) and the similarity between the theoretical and experimental isotopic distribution. Table S11 lists these information. Of these compounds, XCMS was not able to detect 1,11-Undecanedicarboxylic acid. MZmine 2 was not able to detect Bufexamac. ADAP detected all.

Table S10: Compounds that have been manually confirmed to be present in the DCSM data file.

Compound	**Reference Name**
Bicine	c1
N-Acetylglutamine	c2
N-(Hydroxymethyl)nicotinamide	c3
Nicotinuric acid	c4
6-Hydroxynicotinic acid	c5
6-O-Methylguanine	c6
3-Chloro-L-tyrosine	c7
Acetazolamide	c8
Pantothenic acid	c9
L-Tryptophan	c10
4-Acetamidobenzaldehyde	c11
Glafenine	c12
3-(4-Fluorobenzoyl)propionic acid	c13
3-(2-Thienyl)acrylic acid (cis or trans)	c14
2,5-Dimethoxycinnamic acid (cis or trans)	c15
Bufexamac	c16
2-Amino-5-nitrobenzophenone	c17
Probenecid	c18
Dodecanedioic acid	c19
1,11-Undecanedicarboxylic acid	c20
DL-threo-1-Phenyl-2-palmitoylamino-3 -morpholino-1-propanol	c22

Table S11: Data file DCSM compound information. Parentheses next to a reference name distinguish possible isomers. Abbreviated column names: Man. $\rightarrow$ Manually, Theo. $\rightarrow$ Theoretical, Mass Err. $\rightarrow$ Mass Error, Iso. Dist. $\rightarrow$ Isotopic Distribution.

Compound	**Man. Detected**	**Theo.** m/z	**Data** m/z	**Mass Err. (ppm)**	**RT**	**Iso. Dist.**
c1	✓	164.0923	164.0909	5.3	0.65	yes
c2	✓	189.0875	189.0853	6.3	1.17	yes
c3	✓	153.0664	153.0649	6.4	1.94	yes
c4	✓	181.0613	181.0596	5.2	2.31	yes
c5	✓	140.0348	140.0332	7.8	2.65	yes
c6	✓	166.0728	166.0712	6.1	2.94	yes
c7	✓	216.0427	216.0411	3.6	3.25	yes
c8	✓	222.9960	222.9946	2.7	3.90	yes
c9	✓	220.1185	220.1168	3.5	4.00	yes
c10	✓	205.0977	205.0958	4.5	4.26	yes
c11	✓	164.0712	164.0698	4.9	6.08	yes
c12(1)	✓	373.0955	373.0937	1.3	6.90	yes
c12(2)	✓	373.0955	373.0937	1.3	7.15	yes
c13	✓	197.0614	197.0600	3.7	7.97	yes
c14	✓	155.0167	155.0153	5.6	8.40	yes
c15	✓	209.0814	209.0796	4.1	9.68	yes
c16	✓	224.1287	224.1270	3.4	9.88	yes
c17	✓	243.0770	243.0756	2.3	10.83	yes
c18	✓	286.1113	286.1095	2.2	11.18	yes
c19	✓	231.1596	231.1580	3.1	11.54	yes
c20(1)	✓	245.1753	245.1741	2.0	11.56	yes
c20(2)	✓	245.1753	245.1741	2.0	12.28	yes
c22	✓	475.3900	475.3881	0.8	14.25	yes

13 Compounds Manually Confirmed in the YP01, YP02, and VT001 Files

Data files YP01, YP02, and VT001 were generated from NIST Standard Reference Material 1950 (frozen human plasma). The compounds manually found in YP01, YP02, and VT001 are displayed in Table S12. All compounds which were not found in any of the data files manually are omitted from the table entirely.

Table S12: Compounds that have been manually confirmed to be present in the YP01, YP02, and VT001 data files.

Files	YP01	YP02	VT001
Alanine	✓	✓	✓
Histidine	✓		
Isoleucine	✓	✓	
Leucine	✓	✓	
Lysine	✓	✓	✓
Methionine	✓	✓	✓
Proline	✓	✓	✓
Serine	✓	✓	✓
Threonine	✓	✓	✓
Tyrosine	✓	✓	
Valine	✓	✓	
Cysteine	✓		✓
Phenylalanine	✓	✓	✓
Creatinine	✓	✓	
Glucose	✓	✓	
Uric Acid	✓	✓	
Homocysteine	✓	✓	
Testosterone		✓	
Arginine		✓	✓
(Z,Z)-9,12-Octadecadienoic Acid			✓
Lutein			✓

Manually detected compounds in each data file which are missed by either XCMS or MZmine 2 are shown in Table S13. A blank space in the table means that all manually detected compounds for that data file were detected by that software package.

Tables S14, S15, and S16 show detailed information about compounds that have been manually

Table S13: Compounds from the respective data file that were not detected by the respective software package.

	YP01	**YP02**	**VT001**
XCMS		Glucose Progesterone	Cysteine
MZmine 2		Progesterone	

confirmed to be present in the YP01, YP02, and VT001 files, respectively.

Table S14: Data file YP01 compound information.

Compound	**Theo.** m/z	**Data** m/z	**Mass Err. (ppm)**	**RT**	**Iso. Dist.**
Alanine	90.0558	90.0541	21.0	0.90	yes
Histidine	156.0768	156.0706	25.7	0.91	no
Isoleucine	132.1028	132.1007	12.0	0.89	yes
Leucine	132.1028	132.1007	12.0	0.89	yes
Lysine (1)	147.1138	147.1119	10.8	0.68	no
Lysine (2)	147.1138	147.1119	10.8	0.87	no
Methionine	150.0588	150.0572	7.1	0.91	yes
Proline	116.0708	116.0696	9.3	0.92	yes
Serine	106.0508	106.0489	17.2	0.92	yes
Threonine	120.0658	120.0646	8.3	0.92	maybe
Tyrosine	182.0818	182.0793	7.6	0.90	yes
Valine	118.0868	118.0851	12.1	0.89	yes
Cysteine	122.0278	122.0215	42.8	0.95	maybe
Phenylalanine	166.0868	166.0848	7.3	0.90	yes
Creatinine	114.0668	114.0651	13.1	0.86	yes
Glucose	181.0708	181.0707	0.5	1.00	yes
Uric Acid	169.0358	169.0342	5.5	1.03	yes
Homocysteine	136.0428	136.0474	24.8	0.86	yes

14 Results Produced by XCMS and MZmine 2 with Different Parameters

Here we show results where we have changed the upper bound of the *peakwidth* (*wavelet scales* in MZmine 2) parameter used in the paper (see section "Preprocessing Parameters used by XCMS and MZmine 2" for original parameters used to produce results presented in paper). The purpose

Table S15: Data file YP02 compound information.

Compound	**Theo.** m/z	**Data** m/z	**Mass Err. (ppm)**	**RT**	**Iso. Dist.**
Alanine	90.0558	90.0541	21.0	0.85	no
Histidine	156.0768	156.0757	4.6	0.68	yes
Isoleucine	132.1028	132.1007	12.0	0.87	yes
Leucine	132.1028	132.1007	12.0	0.87	yes
Lysine	147.1138	147.1119	8.8	0.67	no
Methionine	150.0588	150.0572	7.1	0.86	no
Proline	116.0708	116.0696	9.3	0.87	yes
Serine	106.0508	106.0489	17.2	0.87	no
Threonine	120.0658	120.0646	8.3	0.85	no
Tyrosine	182.0818	182.0802	4.9	0.88	no
Valine	118.0868	118.0851	12.1	0.87	yes
Phenylalanine	166.0868	166.0848	7.3	0.88	yes
Creatinine	114.0668	114.0651	13.1	0.86	yes
Glucose	181.0708	181.0707	0.5	1.00	yes
Uric Acid	169.0358	169.0342	5.5	0.81	no
Homocysteine	136.0428	136.0468	21.5	0.72	no
Testosterone	289.2168	289.2190	2.6	1.72	no
Arginine	175.1198	175.1177	7.0	0.67	yes

Table S16: Data file VT001 compound information.

Compound	**Theo.** m/z	**Data** m/z	**Mass Err. (ppm)**	**RT**	**Iso. Dist.**
Alanine	90.0558	90.0551	8.8	3.95	yes
Lysine	147.1138	147.1131	3.2	5.36	yes
Methionine	150.0588	150.0587	0.5	3.86	yes
Proline	116.0708	116.0707	0.7	4.31	yes
Serine	106.0508	106.0501	6.8	4.05	no
Threonine	120.0658	120.0658	0.3	3.98	yes
Cysteine	122.0278	122.0227	34.2	5.30	no
Phenylalanine (1)	166.0868	166.0864	1.5	3.08	yes
Phenylalanine (2)	166.0868	166.0864	1.5	3.37	yes
Phenylalanine (3)	166.0868	166.0864	1.5	3.56	yes
Arginine	175.1198	175.1194	1.4	5.26	yes
(Z,Z)-9,12-Octadecadienoic Acid	281.2478	147.1131	0.2	0.75	no
Lutein	569.4358	569.4307	1.6	0.80	yes

of this section is to show that false positives are a problem for different parameters as well. We changed only the one parameter because it is one of the most important parameters in *centWave* and also one of the most obfuscated, used in many of the *centWave* peak filtering steps. Here we have changed it significantly from what is used in the paper, but to a value that is reasonable based on the peak widths of the known compounds in the data files. The new *peakwidth* and *Wavelet scales* are shown in Table S17. We remind the reader the *Wavelet scales* is converted to seconds and then used as the *peakwidth* in *centWave*. The Venn diagrams created by the results found with the new parameter are shown in Fig. S5 and the proportions of false positive peaks for the three sections of each Venn diagram are shown in Table S18.

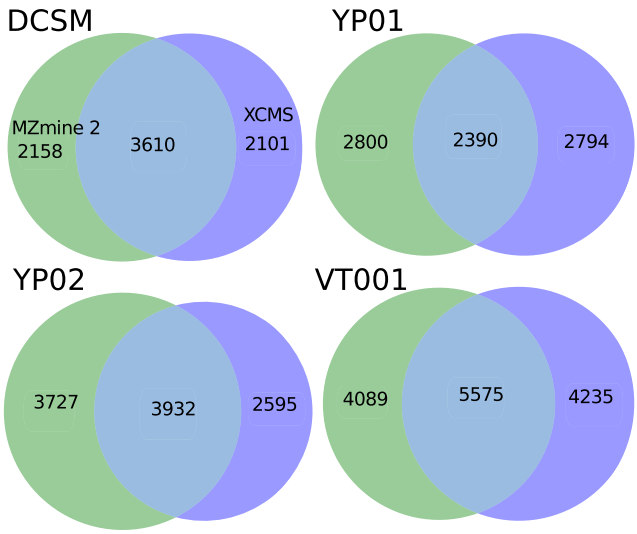

Figure S5: Venn diagrams showing the number of peaks found using XCMS and MZmine 2. Determination of overlap in the Venn diagram was done by checking the proximity of peaks in retention time (within 1.5 seconds) and m/z (within 0.01). These Venn diagrams we made using different (from those used to produce the results presented in the main text) *peakwidth* and *Wavelets scales* parameters in XCMS and MZmine 2 respectively. The new values used are shown in Table S17.

Table S17: XCMS and MZmine 2 modified parameters used for EIC building and peak picking.

	DCSM	**YP01 and YP02**	**VT001**
peakwidth (XCMS) (sec.)	(1.2, 20)	(3.0, 30)	(2.0,20)
Wavelet scales (MZmine 2) (min.)	0.02-0.33	0.10-0.50	0.33-0.33

Table S18: (*top number*) Proportion, shown in red, of peaks in the XCMS-only lobe, MZmine 2-only lobe, and overlapping sections of Figure S5 that are considered good peaks. (*bottom range*) 95% confidence interval, shown in black, of the proportion. These results were produced using the modified parameters shown in Table S17.

Data File	XCMS only	MZmine 2 only	Overlap
DCSM	0.25 0.21-0.29	0.46 0.41-0.51	0.70 0.65-0.74
YP01	0.18 0.14-0.22	0.10 0.07-0.13	0.62 0.57-0.67
YP02	0.38 0.33-0.43	0.11 0.08-0.15	0.56 0.51-0.61
VT001	0.37 0.32-0.41	0.06 0.04-0.09	0.67 0.62-0.71

15 XCMS *centWave* variables

Here we list important variable names in a processing step that XCMS preforms during the find peaks algorithm. For each variable we give a brief explanation that we hope can help the reader in reading and understanding this section of the XCMS code.

- *noiserange*: a ranged derived from the user defined min peak width and the max wavelet scale. It is $\{$min peak width times three, max scales times 3$\}$. It is used to expand the scaled range of the EIC being processed unless it is the entire total scan range. This is done so that a reasonable number of points can be used in an estimate of the noise.
- *sr*: the new scan range you get when using the *noiserange*. It could also just be the same as the actual scan range if it is
- *scRangeTol*: is just the *minPeakWidth*$/2$ rounded down to the closest integer
- *minPtsAboveBaseLine*: is either 4 or $minPeakWidth - 2$, whichever is greater.
- *scanrange*: scan range of all the scans
- *scrange*: scan range of the feature (EIC)
- *sr*: scan range expanded (if possible) using *noiserange*
- *eic*: EIC made with expanded scan range
- *d*: the intensity array from the eic with the expanded scan range ($eic\$intensity$)

- *td*: scan number array from expanded scan range min to max ($sr[1] : sr[2]$)
- *ftd*: similar to td but only expanded by the scan range tolerance (*scRangeTol*)
- *fd*: basically just the intensities of the feature (*d*) but expanded by the scan range tolerance (*scRangeTol*).
- *noised*: if it is a long (N $\geq 10\times$*minPeakWidth* where N is the number of scans in the current feature plus 1) EIC this will be the intensity values of an EIC made with the same m/z range as the current feature but across all scans. If it is not long then it will just be *d* (see description of *d*)
- *noise*: depending on how many zeros are in the *noised* array, this will either be the mean of the noised array (intensities) or a trimmed mean where 0.05 percent of the array is removed on either side before calculating he average.
- *lnoise*: or "local noise" is the noise calculated from the EIC excluding the feature itself. This is because if the feature really is a peak then a pure mean will result in an over estimate of the noise if the peak is included. This is really a list, the first value is the noise, the second value is the standard deviation.
- *baseline*: is either 1 or the smaller value between the "local noise" (*lnoise*$[1]$) or the estimated noise (*noise*), choose the highest.
- *sdnoise*: either 1, or the standard deviation of the local noise ($lnoise[2]$), whichever is larger
- *sdthr*: *sdnoise* multiplied by the signal to noise ratio set by the user.

References

(1) `http://www.nonlinear.com/progenesis/qi/`.

(2) `https://www.thermofisher.com/order/catalog/product/IQLAAEGABSFAHSMAPV`.

(3) `http://www.agilent.com/en-us/products/software-informatics/masshunter-suite/masshunter/masshunter-software`.

(4) http://sciex.com/products/software/markerview-software.

(5) http://www.leco.com/products/separation-science/software-accessories/chromatof-software.

(6) http://www.spectralworks.com/products/analyzerpro/.

(7) http://www.prs.no/MS%20Resolver/MS%20Resolver.html.

(8) Smith, C. A.; Want, E. J.; O'Maille, G.; Abagyan, R.; Siuzdak, G. *Anal Chem* **2006**, *78*, 779–87.

(9) Tautenhahn, R.; Bottcher, C.; Neumann, S. *BMC Bioinformatics* **2008**, *9*, 504.

(10) Tautenhahn, R.; Patti, G. J.; Rinehart, D.; Siuzdak, G. *Anal Chem* **2012**, *84*, 5035–9.

(11) Katajamaa, M.; Miettinen, J.; Oresic, M. *Bioinformatics* **2006**, *22*, 634–6.

(12) Pluskal, T.; Castillo, S.; Villar-Briones, A.; Oresic, M. *BMC Bioinformatics* **2010**, *11*, 395.

(13) Lommen, A. *Anal Chem* **2009**, *81*, 3079–86.

(14) Lommen, A. *Methods Mol Biol* **2012**, *860*, 229–53.

(15) Lommen, A.; Kools, H. J. *Metabolomics* **2012**, *8*, 719–726.

(16) Yu, T.; Park, Y.; Johnson, J. M.; Jones, D. P. *Bioinformatics* **2009**, *25*, 1930–6.

(17) Yu, T.; Peng, H. *BMC Bioinformatics* **2010**, *11*, 559.

(18) Yu, T.; Park, Y.; Li, S.; Jones, D. P. *J Proteome Res* **2013**, *12*, 1419–27.

(19) Yu, T.; Jones, D. P. *Bioinformatics* **2014**, *30*, 2941–8.

(20) Clasquin, M. F.; Melamud, E.; Rabinowitz, J. D. *Curr Protoc Bioinformatics* **2012**, *Chapter 14*, Unit14 11.

(21) http://genomics-pubs.princeton.edu/mzroll/index.php.

(22) Wei, X.; Sun, W.; Shi, X.; Koo, I.; Wang, B.; Zhang, J.; Yin, X.; Tang, Y.; Bogdanov, B.; Kim, S.; Zhou, Z.; McClain, C.; Zhang, X. *Anal Chem* **2011**, *83*, 7668–75.

(23) Rost, H. L. et al. *Nat Methods* **2016**, *13*, 741–8.

(24) Sturm, M.; Bertsch, A.; Gropl, C.; Hildebrandt, A.; Hussong, R.; Lange, E.; Pfeifer, N.; Schulz-Trieglaff, O.; Zerck, A.; Reinert, K.; Kohlbacher, O. *BMC Bioinformatics* **2008**, *9*, 163.

(25) http://www.openms.de/.

(26) Baran, R.; Kochi, H.; Saito, N.; Suematsu, M.; Soga, T.; Nishioka, T.; Robert, M.; Tomita, M. *BMC Bioinformatics* **2006**, *7*, 530.

(27) http://mathdamp.iab.keio.ac.jp/.

(28) NIST SRM 1950. https://www-s.nist.gov/srmors/certificates/1950.pdf, [Accessed January 30, 2017].

(29) Jin, R.; Banton, S.; Tran, V. T.; Konomi, J. V.; Li, S.; Jones, D. P.; Vos, M. B. *The Journal of Pediatrics* **2016**, *172*, 14 – 19.e5.

(30) Li, S.; Park, Y.; Duraisingham, S.; Strobel, F. H.; Khan, N.; Soltow, Q. A.; Jones, D. P.; Pulendran, B. *PLOS Computational Biology* **2013**, *9*, 1–11.

(31) http://www.metabolomicsworkbench.org/.