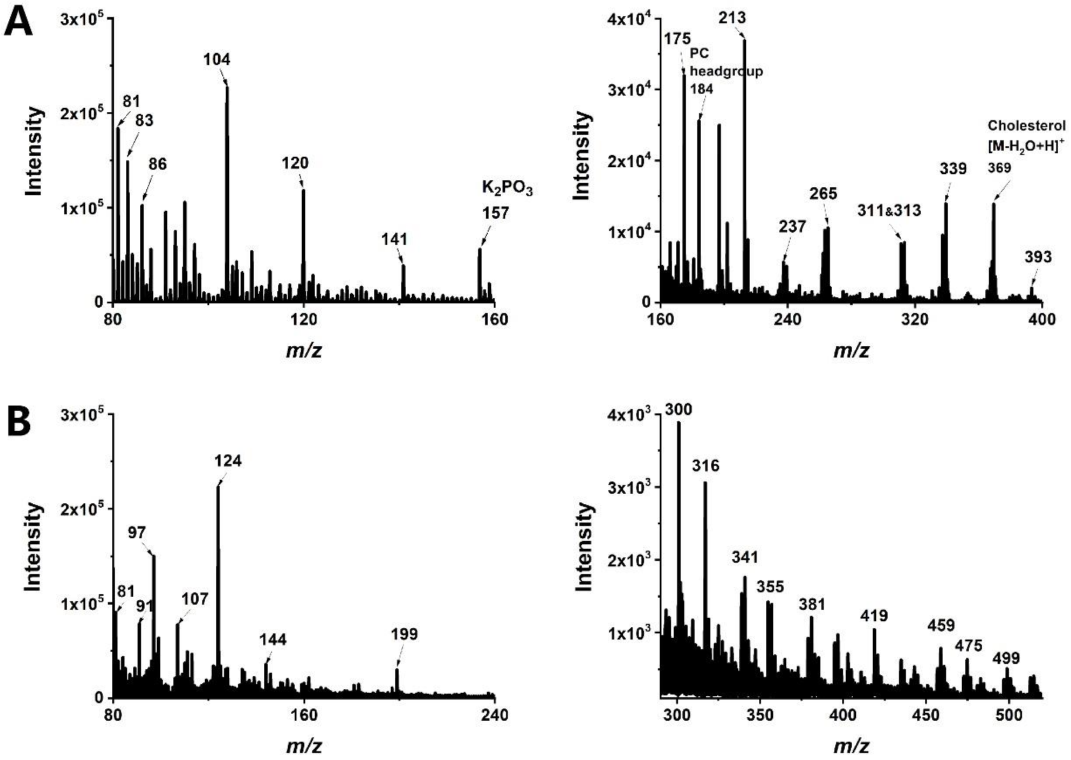
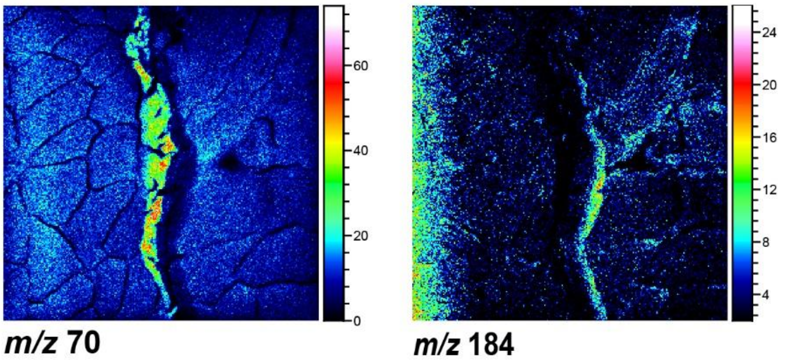
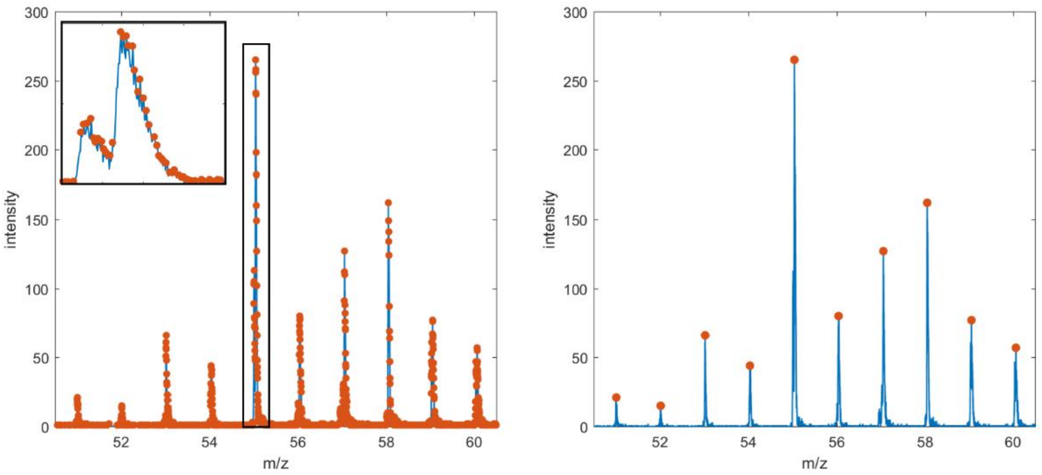


Supplemental Fig. 1 Mass spectra of other intense peaks in positive (A) and negative (B) ion modes.



Supplemental Fig. 2 MSIs of other intense peaks in different muscle regions. Peaks including *m/z* 70 and *m/z* 184 were imaged in positive ion mode. Area of 300 × 300 um^2 on the mouse skeletal muscle section.



Supplemental Fig. 3 Peak detection comparison between MATLAB *findpeaks* function and the optimized peak searching algorithm. Left panel: the library function *findpeaks* of MATLAB generated over-extracted results due to the noisy background of the MS data. The inserted figure with bold lines showed the magnified observation of local signals. Right panel: optimized peak searching algorithm searched for the local maxima within a defined *m/z* range, generating more accurate peak location results than the MATLAB *findpeaks* function.

Source code of the peak searching algorithm

```
function [ pkLocs,pksIntens,I ] = RoughFindPeaks( locs,profiles,halfAcc,minTor,isShow )
    if ~exist('isShow','var')
        isShow = 1;
    end
    if ~exist('minTor','var')
        minTor = 0.01;
    end
    pkLocs = matList();
    pksIntens = matList();
    I = matList();
    torIntens = max(profiles) * minTor;
    L = length(locs);
    indices = 1:L;
    m = 1;
    while(m<=L)
        dist = abs(locs - locs(m));
        bInRange = dist <= halfAcc;
        [intens,locId] = max(profiles(bInRange));
        idsInRange = indices(bInRange);
        if intens > torIntens
            maxId = idsInRange(locId);
            if maxId < m
                m = max([m+1,idsInRange(end)]);
            elseif maxId > m
                m = maxId;
            else
                I.addOne(m);
                pkLocs.addOne(locs(m));
                pksIntens.addOne(profiles(m));
                m = max([m+1,idsInRange(end)]);
            end
        else
            m = max([m+1,idsInRange(end)]);
        end
    end
    pkLocs = cell2mat(pkLocs.data);
    pksIntens = cell2mat(pksIntens.data);
    I = cell2mat(I.data);
    if isShow
        figure;
        plot(locs,profiles);
        hold on;
        scatter(pkLocs,pksIntens,'filled');
    end
end


classdef matList < handle

    properties(SetAccess=private)
        capacity;
        len;
        data;
    end

    methods
        function obj = matList(initCap)
            if ~exist('initCap','var')
                initCap = 100;
            end
            obj.capacity = initCap;
            obj.data = cell(obj.capacity,1);
            obj.len = 0;
        end
```

```
function addOne(obj,mat)
    if obj.len == obj.capacity
        obj.expandCap();
    end
    obj.len = obj.len + 1;
    obj.data{obj.len} = mat;
end

function b = isExist(obj,mat)
    b = any(cellfun(@(x)isequal(x,mat),obj.data(1:obj.len)));
end

function [b,m] = isExist2(obj,mat,acc)
    b = 0;
    if ~exist('acc','var')
        acc = 1e-6;
    end
    for m = 1:obj.len
        b = all(abs(obj.data{m}-mat)<acc);
        if b
            return;
        end
    end
    m = -1;
end

function replaceEle(obj,index,newEle)
    obj.data{index} = newEle;
end

function saveAsTable(obj,fileName,varNames)
    L = length(obj.data{1}(:));
    if L ~= length(strsplit(varNames,','))
        error('matList: varNames length %d is inconsist to the data length %d!',...
            length(varNames),L);
    end
    tmp = cellfun(@(x)length(x(:)),obj.data(1:obj.len));
    if any(tmp~=L)
        error('matList: there exists inconsist data length among data');
    end
    t = cell2mat(cellfun(@(x)x(:)',obj.data(1:obj.len),'UniformOutput',0));
    HScsvwrite(fileName,t,[],varNames);
end
```

```
        function saveAsMAT(obj,fileName)
            data = obj.data(1:obj.len);
            save(fileName,'data');
        end

        function r = toMat(obj)
            r = cell2mat(obj.data(1:obj.len));
        end

        function r = toCell(obj)
            r = obj.data(1:obj.len);
        end
    end

    methods(Access=private)
        function expandCap(obj)
            tmp = cell(2*obj.capacity,1);
            tmp(1:obj.len,:) = obj.data;
            obj.data = tmp;
            obj.capacity = 2 * obj.capacity;
        end
    end

end

function [] = HScsvwrite(fileName,M,strId,varargin)
    fid = fopen(fileName,'a');
    if isempty(strId)
        isStr = false;
    else
        isStr = true;
    end
    [R,L] = size(M);
    if ~isempty(varargin)
        header = varargin{1};
        fprintf(fid,[header,'\n']);
    end
    strLine = repStrline('%f,',L);
    if isStr
        idNum = size(strId,2);
        strFormat = repmat('%s,',idNum);
    end
    for m = 1:1:R
        if isStr
```

```
            fprintf(fid,strFormat,strId{m,:});
        end
        fprintf(fid,strLine,M(m,:));
    end
    fclose(fid);
end

function strline = repStrline(str,num)
    tmp = repmat(str,1,num);
    strline = [tmp(1:end-1),'\n'];
end
```