IPPD package vignette

Martin Slawski [1] ms@cs.uni-sb.de
Rene Hussong [2] rene@bioinf.uni-sb.de
Matthias Hein [1] hein@cs.uni-sb.de

[1] Department of Computer Science, Machine Learning Group
[2] Center for Bioinformatics
Junior Research Group for Protein-Protein Interactions and Computational Proteomics

Saarland University, Saarbrücken, Germany

Abstract

This is the vignette of the Bioconductor add-on package IPPD which implements automatic isotopic pattern extraction from a raw protein mass spectrum. Basically, the user only has to provide mass/charge channels and corresponding intensities, which are automatically decomposed into a list of monoisotopic peaks. IPPD can handle several charge states as well as overlaps of peak patterns.

1 Aims and scope of IPPD

A crucial challenge in the analysis of protein mass spectrometry data is to automatically process the raw spectrum to a list of peptide masses. IPPD is tailored to spectra where peptides emerge in the form of isotope patterns, i.e. one observes several peaks for each peptide mass at a given charge state due to the natural abundance of heavy isotopes. Datasets with a size of up to 100,000 mass/charge channels and the presence of isotope patterns at multiple charge states frequently exhibiting overlap make the manual annotation of a raw spectrum a tedious task. IPPD provides functionality to perform this task in a fully automatic, transparent and user-customizable way. Basically, one feeds the raw spectrum into one single function to obtain a list of monoisotopic peaks described by a mass/charge channel, a charge and an intensity. What makes our approach particularly user-friendly is its dependence on only a small set of easily interpretable parameters. We also offer a method to display the decomposition of the spectrum graphically, thereby facilitating a manual validation of the output.

2 Methodology

2.1 Template model

In the context of this package, a protein mass spectrum is understood as a sequence of pairs $\{x_i, y_i\}_{i=1}^n$, where $x_i = m_i/z_i$ is a mass (m_i) per charge (z_i) value (measured in Thomson) and y_i is the intensity, i.e. the abundance of a particular mass (modulo charge state), observed at x_i, $i = 1, \dots, n$, which are assumed to be in an increasing order. The y_i are modeled as a linear combination of template functions representing prior knowledge about

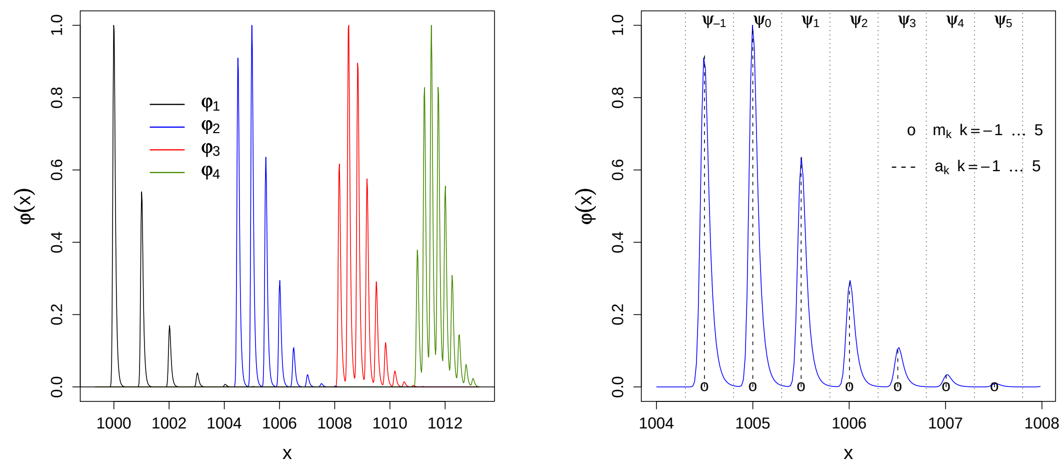


Figure 1: Illustration of the template construction as described in the text. The left panel depicts different templates of different charge states (1 to 4). The right panel zooms at the charge two template φ_2.

peak shapes and the composition of isotopic patterns. If our model were exact, we could write

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\beta}^*, \quad \boldsymbol{y} = (y_1, \ldots, y_n)^\top, \tag{1}$$

where $\boldsymbol{\Phi}$ is a matrix template functions and $\boldsymbol{\beta}^*$ a vector of weights for each template. Only a small fraction of all templates are needed to fit the signal, i.e. $\boldsymbol{\beta}^*$ is highly sparse. Since $\boldsymbol{y} \geq 0$, where '$\geq$' is understood componentwise, all template functions are nonnegative and accordingly $\boldsymbol{\beta}^* \geq 0$. Model (1) can equivalently be written as

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{\Phi}_1 & \ldots & \boldsymbol{\Phi}_C \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1^* \\ \vdots \\ \boldsymbol{\beta}_C^* \end{bmatrix} = \sum_{c=1}^{C} \boldsymbol{\Phi}_c \boldsymbol{\beta}_c^*, \tag{2}$$

where $\boldsymbol{\Phi}_c, \boldsymbol{\beta}_c^*$ denote the matrix of template functions and weight vector to fit isotopic patterns of a particular charge state c, $c = 1, \ldots, C$. Each submatrix $\boldsymbol{\Phi}_c$ can in turn be divided into columns $\boldsymbol{\varphi}_{c,1}, \ldots, \boldsymbol{\varphi}_{c,p_c}$, where the entries of each column vector store the evaluations of a template φ_{c_j}, $j = 1, \ldots, p_c$, at the x_i, $i = 1, \ldots, n$. Each template $\varphi_{c,j}$ depends on parameter $m_{c,j}$ describing the m/z position at which $\varphi_{c,j}$ is placed. A template $\varphi_{c,j}$ is used to fit an isotopic pattern of peaks composed of several single peaks, which is modeled as

$$\varphi_{c,j} = \sum_{k \in Z_{c,j}} a_{c,j,k} \, \psi_{c,j,k,\boldsymbol{\theta}_{c,j}}, \quad Z_{c,j} \subset \mathbb{Z} \tag{3}$$

where the $\psi_{c,j,k}$ are functions representing a peak of a single isotope within an isotopic pattern. They depend on $m_{c,j}$ and a parameter vector $\boldsymbol{\theta}_{c,j}$. The nonnegative weights $a_{c,j,k}$ reflect the relative abundance of the isotope indexed by k. The $a_{c,j,k}$ are computed according to the averagine model (Senko et al. [1995]) and hence are fixed in advance. Each $\psi_{c,j,k}$ is linked to a location $m_{c,j,k}$ at which it attains its maximum. The $m_{c,j,k}$ are calculated from $m_{c,j}$ as $m_{c,j,k} = m_{c,j} + \kappa \frac{k}{c}$, where κ equals 1 Dalton (≈ 1.003). The rationale behind Eq. (3) and the definitions that follow is the fact that the location of the most intense isotope is taken as characteristic location of the template, i.e. we set $m_{c,j,0} = m_{c,j}$ so that the remaining $m_{c,j,k},\ k \neq 0$, are computed by shifting $m_{c,j}$ in both directions on the m/z axis. By 'most intense isotope', we mean that $a_{c,j,0} = \max_k a_{c,j,k} = 1$. The set $Z_{c,j}$ is a subset of the integers which depends on the averagine model and a pre-specified tolerance, i.e. we truncate summation in Eq. (3) if the weights drop below that tolerance. Figure 1 illustrates the construction scheme and visualizes our notation.

2.2 Peak shape

In an idealized setting, the $\psi_{c,j,k}$ are delta functions at specific locations. In practice, however, the shape of a peak equals that of a bump which may exhibit some skewness. In the case of no to moderate skewness, we model peaks by Gaussian functions:

$$\psi_{c,j,k}(x) = \exp\left(-\frac{(x - m_{c,j,k})^2}{\sigma_{c,j}}\right). \tag{4}$$

The parameter to be determined is $\boldsymbol{\theta}_{c,j} = \sigma_{c,j} > 0$. In the case of considerable skewness, peaks are modeled by exponentially modified Gaussian (EMG) functions, see for instance Grushka [1972], Marco and Bombi [2001], and Schulz-Trieglaff et al. [2007] in the context of protein mass spectrometry:

$$\psi_{c,j,k}(x) = \frac{1}{\alpha_{c,j}} \exp\left(\frac{\sigma_{c,j}^2}{2\alpha_{c,j}^2} + \frac{\mu_{c,j} - (x - m_{c,j,k})}{\alpha_{c,j}}\right)\left(1 - F\left(\frac{\sigma_{c,j}}{\alpha_{c,j}} + \frac{\mu_{c,j} - (x - m_{c,j,k})}{\sigma_{c,j}}\right)\right),$$
$$F(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)\, du. \tag{5}$$

The EMG function involves a vector of three parameters $\boldsymbol{\theta}_{c,j} = (\alpha_{c,j}, \sigma_{c,j}, \mu_{c,j})^\top \in \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}$. The parameter $\alpha_{c,j}$ controls the additional length of the right tail as compared to a Gaussian. For $\alpha_{c,j} \downarrow 0$, the EMG function becomes a Gaussian. For our fitting approach as outlined in Section 2.3, it is crucial to estimate the $\boldsymbol{\theta}_{c,j}$, which are usually unknown, from the data as good as possible. To this end, we model each component θ_l of $\boldsymbol{\theta}$ as a linear combination of known functions $g_{l,m}$ of $x = m/z$ and an error component ε_l, i.e.

$$\theta_l(x) = \sum_{m=1}^{M_l} \nu_{l,m} g_{l,m}(x) + \varepsilon_l(x). \tag{6}$$

In the case of no prior knowledge about the $g_{l,m}$, we model θ_l as a constant independent of x. In most cases, it is sensible to assume a linear trend, i.e. $\theta_l(x) = \nu_{l,1} + \nu_{l,2} x$. In order to fit a model of the form (6), we have to collect information from the data $\{x_i, y_i\}_{i=1}^n$. To be precise, we proceed according to the following steps.

1. We apply a simple peak detection algorithm to the spectrum to identify disjoint regions $\mathcal{R}_r \subset \{1, \ldots, n\},\ \ r = 1, \ldots, R$, of well-resolved peaks.

2. For each region r, we fit the chosen peak shape to the data $\{x_i, y_i\}_{i \in \mathcal{R}_r}$ using nonlinear least squares:

$$\min_{\boldsymbol{\theta}} \sum_{i \in \mathcal{R}_r} (y_i - \psi_{\boldsymbol{\theta}}(x_i))^2, \tag{7}$$

yielding an estimate $\widehat{\boldsymbol{\theta}}_r(\widehat{x}_r)$, where $\widehat{x}_r$ denotes an estimation for the mode of the peak in region $\mathcal{R}_r$.

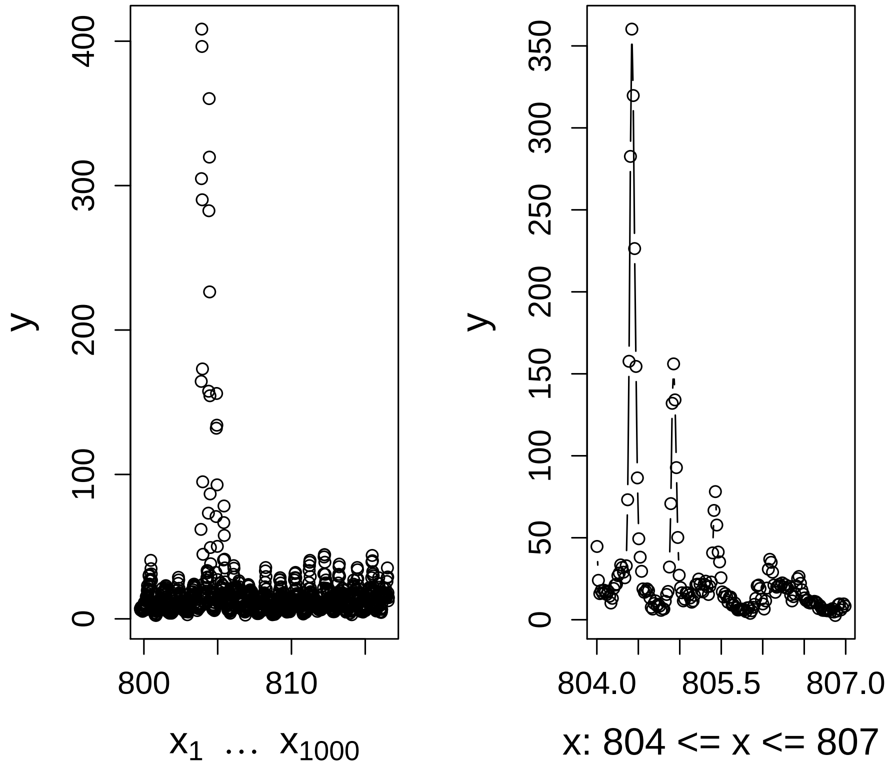
3. The sequence $\{\widehat{x}_r, \widehat{\boldsymbol{\theta}}_r\}_{r=1}^R$ is then used as input for the estimation of the parameters $\nu_{l,m}$ in model (6).

Step 2. is easily solved by the general purpose nonnegative least squares routine `nls` in `R:::stats` for a Gaussian peak shape. For the EMG, we have to perform a grid search over all three parameters to find a suitable starting value, which is then passed to the general purpose optimization routine `optim` in `R:::stats` with the option `method = "BFGS"` and a specification of a closed form expression of the gradient via the argument `gr`. For step 3., we use least absolute deviation regression because of the presence of outliers arising from less well-resolved, wiggly or overlapping peaks. The whole procedure is performed by the function `fitModelParameters` as demonstrated below. After loading the package, we access the real world dataset `myo500` and extract m/z channels (`x`) and the corresponding intensities (`y`). For computational convenience and since they contain very few relevant information, we discard all channels above 2500.

```
R> library(IPPD)
R> data(myo500)
R> x <- myo500[,"mz"]
R> y <- myo500[,"intensities"]
R> y <- y[x <= 2500]
R> x <- x[x <= 2500]
```

To have a look at the data, we plot the first 1000 (`x`,`y`) pairs:

```
R> layout(matrix(c(1,2), 1, 2))
R> plot(x[1:1000], y[1:1000], xlab = expression(x[1]~~ldots~~x[1000]),
     cex.lab = 1.5, cex.axis = 1.25, ylab = expression(y))
R> plot(x[x >= 804 & x <= 807], y[x >= 804 & x <= 807],
     xlab = "x: 804 <= x <= 807",
     cex.lab = 1.5, cex.axis = 1.25, ylab = expression(y), type = "b")
R> layout(matrix(1))
R>
```



In the plot, one identifies a prominent peak pattern beginning at about 804, which is zoomed at in the right panel.
We now apply `fitModelParameters` to fit model (6) for the width parameter σ of a Gaussian function (4). For simplicity, we take $g_1(x) = 1$, $g_2(x) = x$. The model is specified by using an `R` formula interface.

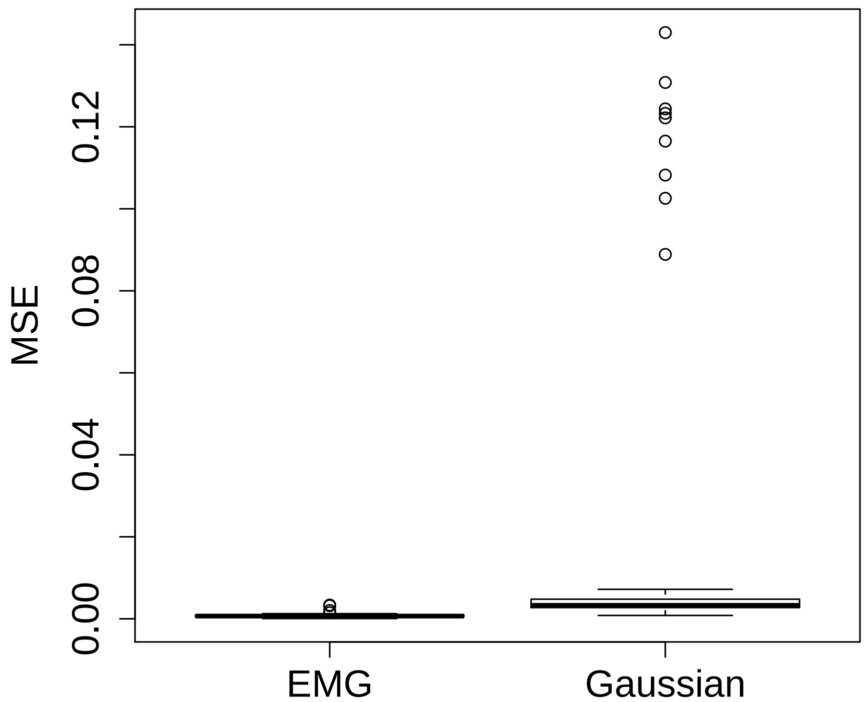
```
R> fitGauss <- fitModelParameters(mz = x, intensities = y,
                    model = "Gaussian", fitting = "model", formula.sigma = formula(~mz),
                    control = list(window = 6, threshold = 200))
```

An analogous command for the EMG (5) with the model formulae $\alpha(x) = \nu_{1,1} + \nu_{1,2}x$, $\sigma(x) = \nu_{2,1} + \nu_{2,2}x$, $\mu(x) = \nu_{3,1}$ is given by

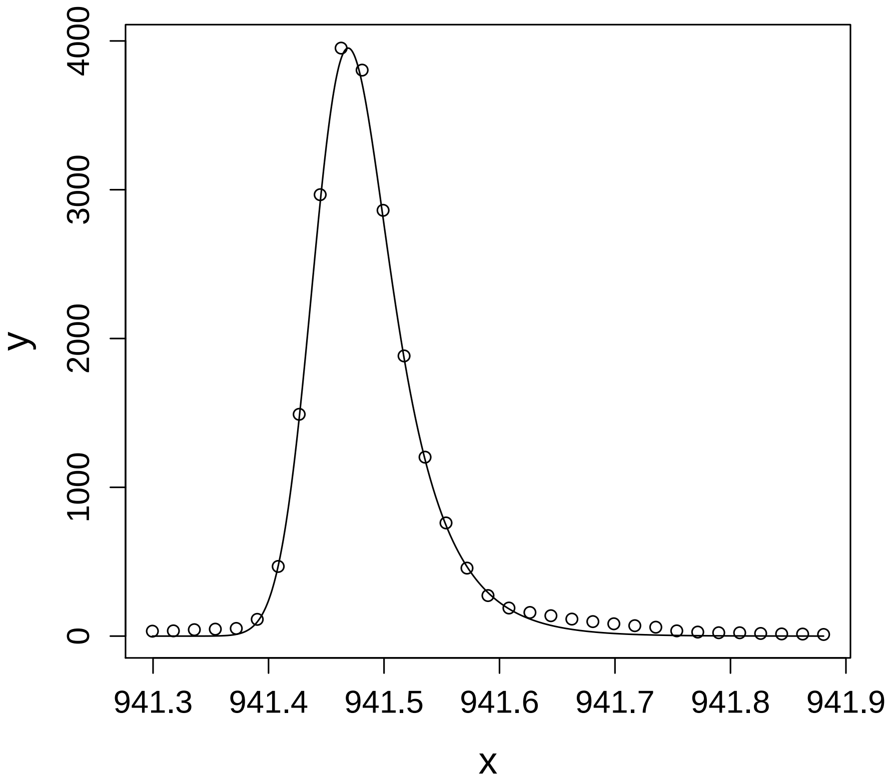
```
R> fitEMG <- fitModelParameters(mz = x, intensities = y,
                    model = "EMG", fitting = "model",
                            formula.alpha = formula(~mz),
                            formula.sigma = formula(~mz),
                            formula.mu = formula(~1),
                            control = list(window = 6, threshold = 200))
```

Inspecting the results, we find that $R = 55$ peak regions are used to fit an EMG parameter model. Moreover, it turns out that the EMG model is a more appropriate peak model for the data when visually comparing the list of mean residual sums of squares of the EMG fits and the Gauss fits extracted from `slot(fitEMG, "peakfitresults")` and slot(fitGauss, "peakfitresults"), respectively. The figure shows an example where the EMG shape comes relatively close to the observed data. A long right tail indicates that a Gaussian would yield a rather poor fit here.

```
R> show(fitEMG)

Peak model 'EMG' fitted as fuction of m/z
number of peaks used: 55

R> mse.EMG <- data.frame(mse = slot(fitEMG,"peakfitresults")[,"rss"]
                        / slot(fitEMG,"peakfitresults")[,"datapoints"],
                        peakshape = rep("EMG", nrow( slot(fitEMG,"peakfitresults"))))
R> mse.Gauss <- data.frame(mse = slot(fitGauss,"peakfitresults")[,"rss"]
                          / slot(fitGauss,"peakfitresults")[,"datapoints"],
                          peakshape = rep("Gaussian", nrow( slot(fitGauss,"peakfitresults"))))
R> mses <- rbind(mse.EMG, mse.Gauss)
R> with(mses, boxplot(mse ~ peakshape, cex.axis = 1.5, cex.lab = 1.5, ylab = "MSE"))
R>
```

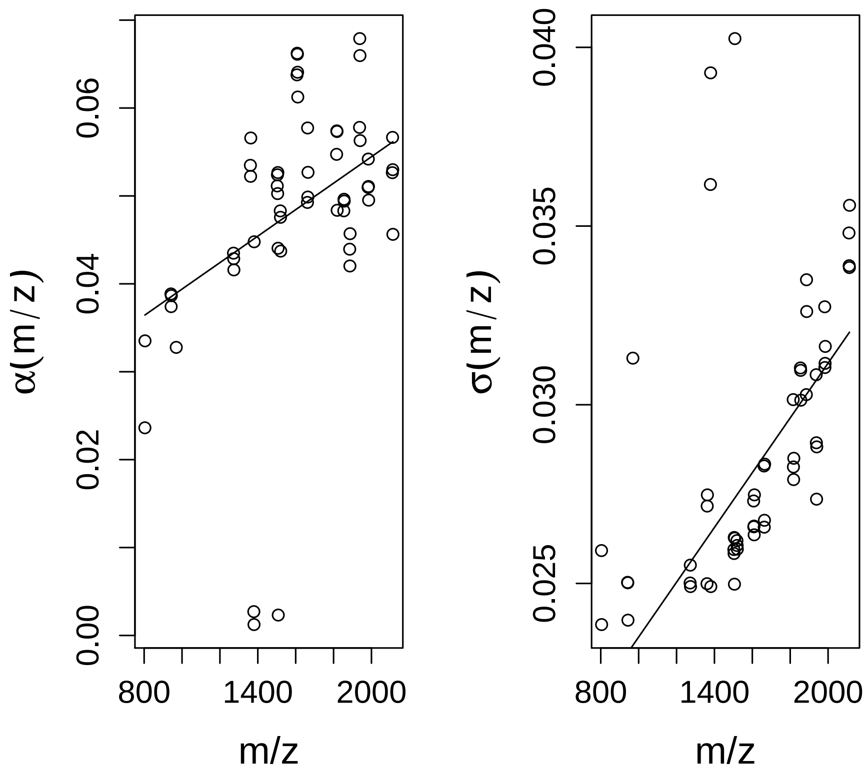



```
R> visualize(fitEMG, type = "peak", cex.lab = 1.5, cex.axis = 1.25)
```



To assess the fit of the two linear models for the EMG parameters α and σ, we use again the function `visualize` as follows:

```
R> visualize(fitEMG, type = "model", modelfit = TRUE,
           parameters = c("sigma", "alpha"),
           cex.lab = 1.5, cex.axis = 1.25)
R>
```



While the fit for σ seems to be reasonable except for some extreme outliers, the fit for α is not fully convincing. Nevertheless, in the absence of further knowledge, the fit produces good results in the template matching step detailed in the next section.

2.3 Template fitting

Once all necessary parameters have been determined, the positions at which the templates are placed have to be fixed. In general, one has to choose positions from the interval $[x_1, x_n]$. We instead restrict us to a suitable subset of the finite set $\{x_i\}_{i=1}^n$. The deviations from the true positions is then at least in the order of the sampling rate, but this can be improved by means of a postprocessing step described in 2.4. Using the whole set $\{x_i\}_{i=1}^n$ may be computationally infeasible if n is large. Such an approach would be at least computationally wasteful, since 'genuine' peaks patterns occur very sparsely in the spectrum. Therefore, we apply a pre-selection step on the basis of what we term 'local noise level' (LNL). The LNL is defined as a quantile (typically the median) of the intensities y_i falling into a sliding window of fixed width around a specific position. Given the LNL, we place templates on an x_i (one for each charge state) if and only if the corresponding y_i exceeds the LNL at x_i by a factor `factor.place`, which typically equals three or four and has to be specified by the user. Given the positions of the templates, we compute the matrix $\mathbf{\Phi}$ according to Eqs. (1) and (3). It then remains to estimate the coefficient vector $\boldsymbol{\beta}^*$ on the basis of two structural assumptions, sparsity and nonnegativity of all quantities involved. Related approaches in the literature (Du and Angeletti [2006], Renard et al. [2008]) account for sparsity of $\boldsymbol{\beta}^*$ by using ℓ_1-regularized regression (Tibshirani [1996]). We here argue empirically that ℓ_1 regularization is not the best to do, since it entails the selection of a tuning parameter which is difficult to choose in our setting, and secondly the structural constraints concerning nonnegativity turn out to be so strong that sparsity is more conveniently achieved by fitting followed by hard thresholding. We first determine

$$\begin{aligned} &\widehat{\boldsymbol{\beta}} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \, \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_q^q \,, \quad q = 1 \text{ or } q = 2, \\ &\text{subject to } \; \boldsymbol{\beta} \geq 0. \end{aligned} \tag{8}$$

The optimization problem (8) is a quadratic ($q = 2$) or linear ($q = 1$) program and is solved using standard techniques (Boyd and Vandenberghe [2004]); we omit further details here. We remark that in the presence of high noise, it is helpful to subtract the LNL from $\boldsymbol{y}$. Concerning the choice of q, we point out that $q = 1$ can cope better with deviations from model assumptions, i.e. deviations from the averagine model or from the peak model and thus may lead to a reduction of the number of false positives.

2.4 Postprocessing

Given an estimate $\widehat{\boldsymbol{\beta}}$, we define $\mathcal{M}_c = \{m_{c,j} : \; \widehat{\beta}_{c,j} > 0\} \subset \{x_i\}_{i=1}^n, \;\; c = 1, \dots, C$, as the set of all template locations where the corresponding coefficient exceeds 0, separately for each charge. Due to a limited sampling rate, different sources of noise and model misfit, the locations in the sets $\{\mathcal{M}_c\}_{c=1}^C$ may still deviate considerably from the set of true peak pattern locations. Specifically, the sets $\{\mathcal{M}_c\}_{c=1}^C$ tend to be too large, mainly caused by what we term 'peak splitting': for the reasons just mentioned, it frequently occurs that several templates are used to fit the same peak. This can at least partially be corrected by means of the following merging procedure.

1. Separately for each c, divide the sets $\mathcal{M}_c$ into groups $\mathcal{G}_{c,1}, \dots, \mathcal{G}_{c,G_c}$ of 'adjacent' positions. Positions are said to be adjacent if their distance on the m/z scale is below a certain tolerance as specified via a parts per million (ppm) value.

2. For each $c = 1, \dots, C$ and each group $g_c = 1, \dots, G_c$, we solve the following optimization problem.

$$(\widetilde{m}_{c,g}, \widetilde{\beta}_{c,g}) = \min_{m_{c,g}, \beta_{c,g}} \left\| \sum_{m_{c,j} \in \mathcal{G}_{c,g}} \widehat{\beta}_{c,j} \psi_{m_{c,j}} - \beta_{c,g} \psi_{m_{c,g}} \right\|_{L^2}^2 \tag{9}$$

   In plain words, we take the fitted function resulting from the functions $\{\psi_{m_{c,j}}\}$ representing the most intense peak of each peak pattern in the same group and then determine a function $\psi_{\widetilde{m}_{c,g}}$ placed at location $\widetilde{m}_{c,g}$ and weighted by $\widetilde{\beta}_{c,g}$ such that $\widetilde{\beta}_{c,g} \psi_{\widetilde{m}_{c,g}}$ approximates the fit of multiple functions $\{\psi_{m_{c,j}}\}$ best (in a least squares sense).

3. One ends up with sets $\widetilde{\mathcal{M}}_c = \{\widetilde{m}_{c,g}\}_{g=1}^{G_c}$ and coefficients $\{\widetilde{\beta}_{c,g}\}_{g=1}^{G_c}, \;\; c = 1, \dots, C$.

The additional benefit of step 2. as compared to the selection of the function with the largest coefficient as proposed in Renard et al. [2008] is that, in the optimal case, we are able to determine the peak pattern location even more accurate as predetermined by a limited sampling rate. The integral in (9) can be solved analytically for a Gaussian function, and we resort to numeric approximations for the EMG function.
The sets $\{\mathcal{M}_c\}$ tend to be too large in the sense that they still contain noise peak patterns.

Therefore, we apply hard thresholding to the $\{\widetilde{\beta}_{c,g}\}_{g=1}^{G_c}$, $c = 1, \ldots, C$, discarding all positions where the corresponding coefficients is less than a significance level times the LNL, where the signficance level has to be specified by the user.

3 Case study

We continue the data analysis starting in Section 2.2. The methodology of the Sections 2.3 and 2.4 is implemented in the function `getPeaklist`. For the computation of the template functions, we recycle the object `fitEMG` obtained in Section 2.2.

```
R> EMGlist <- getPeaklist(mz = x, intensities = y, model = "EMG",
 model.parameters = fitEMG,
 loss = "L2", trace = FALSE,
 control.localnoise = list(factor.place = 2),
 control.basis = list(charges = c(1, 2)),
 control.postprocessing = list(ppm = 200))
R> show(EMGlist)

An object of class 'peaklist'(with postprocessing)
Loss function used: L2
Peak model used: EMG
number of peaks: 1222
charge states used: 1,2

R>
```

The argument list can be summarized as follows: we compute EMG templates for charges 1 and 2; templates are placed on all m/z-positions in the spectrum where the intensity is at least two times the LNL; the fit is least squares (`loss = L2`); postprocessing is performed by merging peaks within a tolerance of 200 ppm. Subsequently, only the patterns with signal-to-noise ratio bigger than three are maintained. The result is of the following form.

```
R> threshold(EMGlist, threshold = 3, refit = TRUE, trace = FALSE)

      loc_init loc_most_intense charge      quant   amplitude localnoise      ratio
 [1,] 800.4642         800.4642      1   79.44145   50.273955   9.411770   5.341605
 [2,] 808.2414         808.2414      1   80.36507   50.663658  10.196100   4.968925
 [3,] 829.2292         829.2292      1   93.77381   58.509383   9.411770   6.216618
 [4,] 842.4935         842.4935      1   65.95672   40.883691   8.366010   4.886880
 [5,] 864.4065         864.4065      1  142.82358   87.563490  13.333300   6.567278
 [6,] 877.0373         877.0373      1  115.96709   70.647034   8.888890   7.947790
 [7,] 908.4247         908.4247      1   77.95258   46.743175   8.104580   5.767501
 [8,] 908.9339         908.9339      1   57.31024   34.357187   9.411770   3.650449
 [9,] 923.4380         923.4380      1   71.87806   42.784829   7.581700   5.643171
[10,] 924.4543         924.4543      1   72.74984   43.281494   8.366010   5.173493
[11,] 927.4746         927.4746      1  109.40194   64.990843   8.888890   7.311469
[12,] 927.9689         927.9689      1   60.24656   35.781209   9.934640   3.601661
[13,] 941.4672         941.4672      1 6891.18891 4065.615602  12.549000 323.979249
```

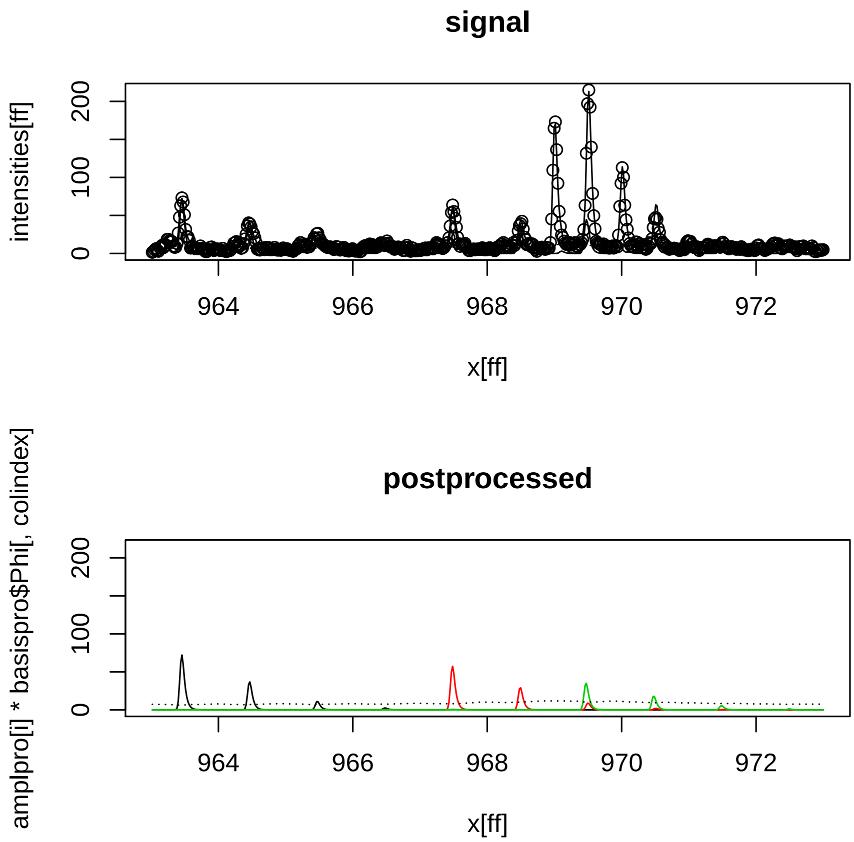
[14,]	941.6761	941.6761	1	135.99053	80.219510	13.594800	5.900750
[15,]	942.4249	942.4249	1	271.88236	160.316307	16.732000	9.581419
[16,]	949.4364	949.4364	1	71.81256	42.198106	7.320260	5.764564
[17,]	952.5318	952.5318	1	76.11884	44.659496	7.843140	5.694084
[18,]	954.4820	954.4820	1	106.27402	62.290498	8.104580	7.685839
[19,]	963.4553	963.4553	1	148.93764	86.903255	6.535950	13.296193
[20,]	967.4826	967.4826	1	145.35796	84.644336	8.104580	10.444013
[21,]	969.4694	969.4694	1	262.52317	152.715484	10.457500	14.603441
[22,]	985.4424	985.4424	1	125.56428	72.456118	7.320260	9.898025
[23,]	991.5025	991.5025	1	143.70662	82.668836	9.411770	8.783559
[24,]	992.0244	992.0244	1	126.06443	72.501354	10.196100	7.110695
[25,]	999.4665	999.4665	1	153.63582	88.022072	7.320260	12.024446
[26,]	1023.4509	1023.4509	1	72.97884	41.200637	6.535950	6.303695
[27,]	1057.4478	1057.4478	1	61.59659	34.040137	7.320260	4.650127
[28,]	1086.5573	1086.5573	1	163.01456	88.424740	6.013070	14.705423
[29,]	1125.5187	1125.5187	1	82.19542	43.609009	6.797390	6.415552
[30,]	1151.4789	1151.4789	1	97.86679	51.210334	5.751630	8.903621
[31,]	1168.6235	1168.6235	1	88.00277	45.625087	6.274510	7.271498
[32,]	1192.7011	1192.7011	1	73.30092	37.507669	6.535950	5.738671
[33,]	1271.6609	1271.6609	1	3373.47413	1646.576999	10.457500	157.454172
[34,]	1297.6800	1297.6800	1	143.23324	68.796163	7.581700	9.073976
[35,]	1360.7611	1360.7611	1	3816.67186	1720.040463	13.071900	131.583049
[36,]	1361.7379	1361.7379	1	1393.29669	627.255229	20.653600	30.370261
[37,]	1378.8379	1378.8379	1	2998.21544	1325.426782	15.686300	84.495820
[38,]	1394.8413	1394.8413	1	133.38764	57.958356	9.150330	6.334018
[39,]	1474.6387	1474.6387	1	154.35911	63.760694	9.934640	6.418018
[40,]	1484.6606	1484.6606	1	554.63328	227.677756	11.764700	19.352619
[41,]	1500.6588	1500.6588	1	186.99548	76.008359	14.902000	5.100548
[42,]	1501.6659	1501.6659	1	237.89772	96.648852	20.130700	4.801068
[43,]	1502.6668	1502.6668	1	6512.20399	2644.083819	27.973900	94.519671
[44,]	1506.9383	1506.9383	1	1362.30919	551.937920	18.039200	30.596585
[45,]	1518.6637	1518.6637	1	3192.21932	1285.242860	14.902000	86.246333
[46,]	1519.6113	1519.6113	1	167.63225	67.460094	15.424800	4.373483
[47,]	1524.6527	1524.6527	1	142.44825	57.174412	9.411770	6.074778
[48,]	1534.6603	1534.6603	1	271.30974	108.319909	12.287600	8.815384
[49,]	1546.6550	1546.6550	1	180.73996	71.699011	10.196100	7.032003
[50,]	1588.8538	1588.8538	1	206.83078	80.197111	12.287600	6.526670
[51,]	1589.8317	1589.8317	1	205.68076	79.705151	14.379100	5.543125
[52,]	1606.8601	1606.8601	1	47656.89747	18288.544972	27.451000	666.225091
[53,]	1622.8482	1622.8482	1	282.47887	107.338422	8.104580	13.244168
[54,]	1628.8462	1628.8462	1	197.47210	74.757363	7.843140	9.531560
[55,]	1632.8754	1632.8754	1	231.61098	87.460710	9.673200	9.041549
[56,]	1643.8448	1643.8448	1	295.35561	110.769208	10.980400	10.087903
[57,]	1650.8348	1650.8348	1	219.29921	81.883060	9.411770	8.700070
[58,]	1660.8520	1660.8520	1	346.30881	128.493109	15.424800	8.330293
[59,]	1661.8523	1661.8523	1	7081.62590	2625.534997	20.392200	128.751925
[60,]	1675.8106	1675.8106	1	140.11467	51.492290	7.058820	7.294745
[61,]	1683.8293	1683.8293	1	115.18086	42.111284	6.535950	6.443024
[62,]	1687.8674	1687.8674	1	138.47195	50.494593	7.581700	6.660062
[63,]	1712.6676	1712.6676	1	116.17816	41.771601	6.274510	6.657349

[64,]	1717.8060	1717.8060	1	100.79866	36.152097	6.274510	5.761740
[65,]	1718.8742	1718.8742	1	78.34258	28.083333	6.535950	4.296748
[66,]	1753.7102	1753.7102	1	89.35124	31.479318	5.490200	5.733729
[67,]	1777.8665	1777.8665	1	106.10237	36.930658	6.013070	6.141731
[68,]	1798.8819	1798.8819	1	175.36134	60.386534	7.581700	7.964775
[69,]	1815.9052	1815.9052	1	9222.12252	3149.052692	24.052300	130.925221
[70,]	1831.9004	1831.9004	1	327.93016	111.105627	5.751630	19.317242
[71,]	1837.8935	1837.8935	1	368.32466	124.419634	6.274510	19.829379
[72,]	1847.8954	1848.9034	1	373.93783	125.768310	6.274510	20.044324
[73,]	1852.9575	1853.9655	1	162.82743	54.759110	10.980400	4.986987
[74,]	1853.9657	1854.9737	1	2843.31754	956.193683	12.549000	76.196803
[75,]	1869.9595	1870.9675	1	428.17363	143.949136	6.013070	23.939375
[76,]	1885.0257	1886.0337	1	2133.58694	717.090626	9.150330	78.367734
[77,]	1885.9545	1886.9625	1	157.60503	52.969463	9.673200	5.475899
[78,]	1897.9419	1898.9499	1	73.78262	24.791973	4.444440	5.578200
[79,]	1901.0127	1902.0207	1	88.40222	29.699601	4.705880	6.311168
[80,]	1919.0078	1920.0158	1	146.81349	49.262051	6.013070	8.192496
[81,]	1937.0230	1938.0310	1	10570.05666	3542.225521	27.712400	127.820958
[82,]	1953.0164	1954.0244	1	116.41572	38.970328	3.660130	10.647252
[83,]	1958.9972	1960.0052	1	146.47000	49.010723	3.921570	12.497730
[84,]	1963.0361	1964.0441	1	219.18803	73.322477	6.013070	12.193851
[85,]	1969.9485	1970.9565	1	300.77516	100.566547	4.444440	22.627496
[86,]	1981.0615	1982.0695	1	244.81985	81.793238	13.594800	6.016509
[87,]	1982.0622	1983.0702	1	5026.45793	1679.199820	15.686300	107.048815
[88,]	1994.0448	1995.0528	1	131.82473	44.002939	3.921570	11.220746
[89,]	1995.0238	1996.0318	1	80.08518	26.730494	3.660130	7.303154
[90,]	1998.0483	1999.0563	1	86.94326	29.013517	3.398690	8.536676
[91,]	2004.0431	2005.0511	1	76.49469	25.516107	3.660130	6.971366
[92,]	2008.0804	2009.0884	1	135.36525	45.140515	3.660130	12.333036
[93,]	2039.0826	2040.0906	1	56.21758	18.706444	2.875820	6.504734
[94,]	2052.9999	2054.0079	1	68.79131	22.867988	2.352940	9.718900
[95,]	2092.1287	2093.1367	1	81.25866	26.938525	3.660130	7.359991
[96,]	2098.0541	2099.0621	1	76.68584	25.411940	3.137260	8.100043
[97,]	2105.0082	2106.0162	1	172.44634	57.105965	3.398690	16.802346
[98,]	2109.1722	2110.1802	1	245.67708	81.321211	11.764700	6.912306
[99,]	2110.1592	2111.1672	1	5284.76546	1749.119243	14.117600	123.896359
[100,]	2126.1453	2127.1533	1	79.79170	26.366710	2.352940	11.205857
[101,]	2132.1398	2133.1478	1	59.99236	19.812067	2.352940	8.420133
[102,]	2136.1764	2137.1844	1	73.48754	24.258854	2.614380	9.279008
[103,]	2154.1284	2155.1364	1	41.38814	13.637610	2.091500	6.520493
[104,]	2157.9980	2159.0060	1	41.70488	13.736534	1.830070	7.506015
[105,]	2166.1065	2167.1145	1	73.99906	24.353297	2.352940	10.350156
[106,]	2172.1089	2173.1169	1	33.40578	10.987274	1.830070	6.003745
[107,]	2188.0666	2189.0746	1	36.97284	12.140568	1.633987	7.430025
[108,]	2211.1144	2212.1224	1	68.46275	22.421494	2.091500	10.720294
[109,]	2226.1425	2227.1505	1	102.27084	33.430381	1.633987	20.459386
[110,]	2233.0991	2234.1071	1	33.83567	11.050566	1.633987	6.762944
[111,]	2257.0072	2258.0152	1	73.28653	23.862764	1.633987	14.604006
[112,]	2283.2155	2284.2235	1	182.54499	59.241203	2.091500	28.324745
[113,]	2284.1744	2285.1824	1	24.07025	7.810563	2.091500	3.734431

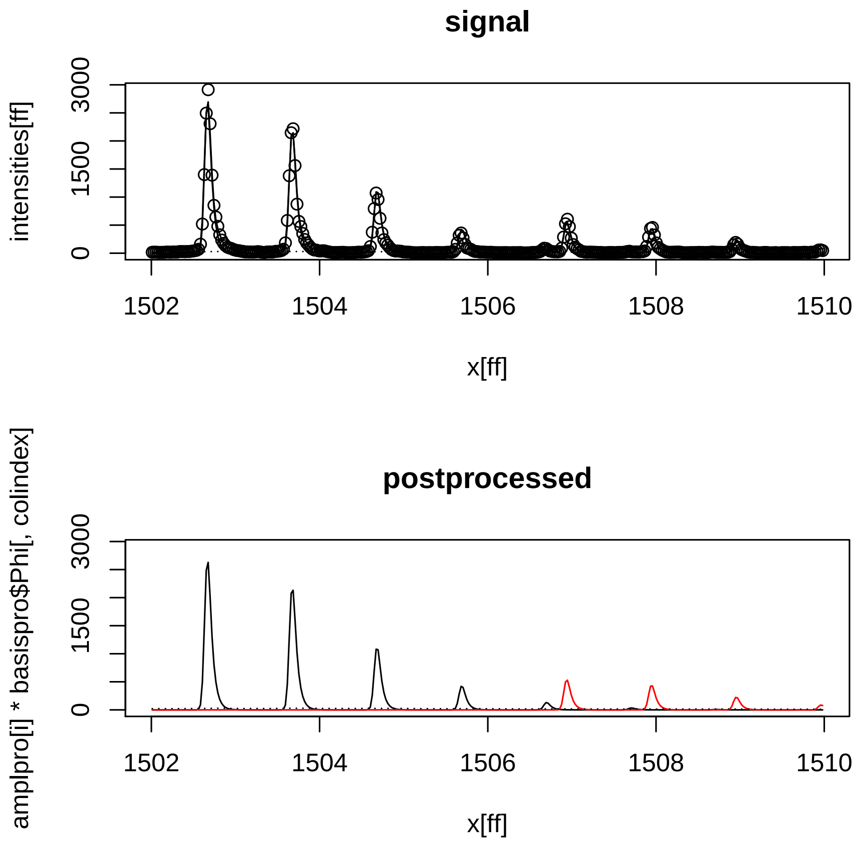
```
[114,] 2427.2620        2428.2700      1     50.99399     16.212221    1.633987    9.921876

R>
```

The results can be examined in detail graphically. We finally present some selected regions to demonstrate that our method performs well. The pre-defined method `visualize` can be used display the template fitting at several stages for regions within selected m/z intervals as specified by the arguments `lower` and `upper`.

```
R> visualize(EMGlist, x, y, lower= 963, upper = 973,
            fit = FALSE, fittedfunction = TRUE, fittedfunction.cut = TRUE,
            localnoise = TRUE, quantile = 0.5,
            cutoff.functions = 3)
R>
```

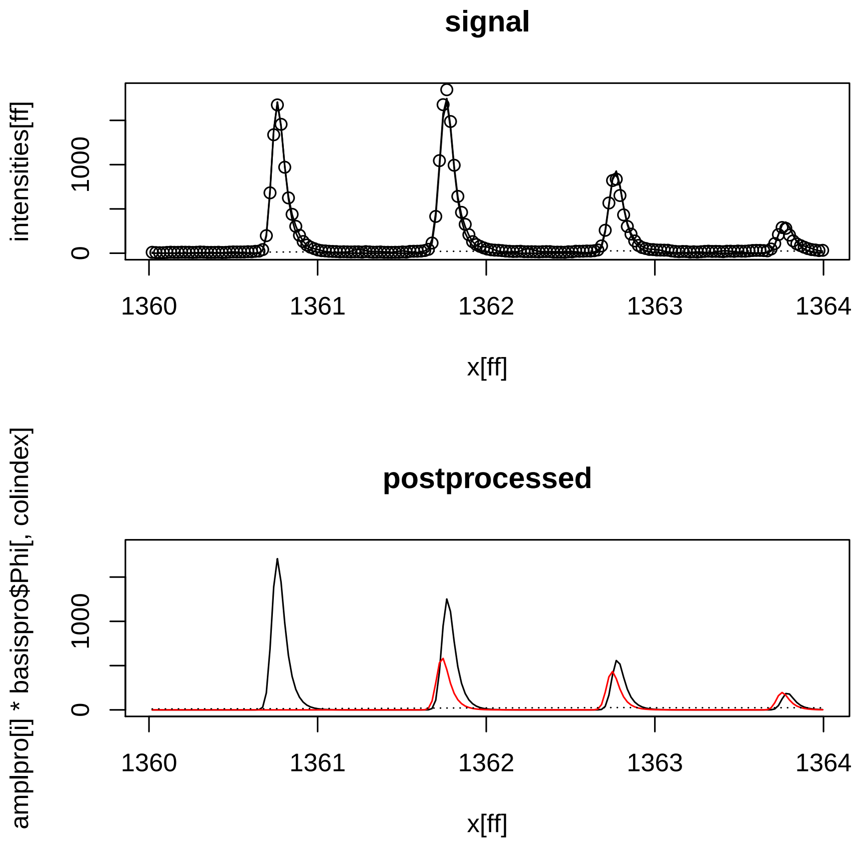



```
R> visualize(EMGlist, x, y, lower= 1502, upper = 1510,
            fit = FALSE, fittedfunction = TRUE, fittedfunction.cut = TRUE,
            localnoise = TRUE, quantile = 0.5,
            cutoff.functions = 2)
R>
R>
```



In the m/z range $[963, 973]$ a charge-1 peak overlaps with a more intense charge two peak. A further overlap occurs in the interval $[1502, 1510]$, and it is correctly resolved.
An even more challenging problem, in which it is already difficult to unravel the overlap by visual inspection, is displayed in the following plot.

```
R> visualize(EMGlist, x, y, lower= 1360, upper = 1364,
             fit = FALSE, fittedfunction = TRUE, fittedfunction.cut = TRUE,
             localnoise = TRUE, quantile = 0.5,
             cutoff.functions = 2)
R>
```



References

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

P. Du and R. Angeletti. Automatic Deconvolution of Isotope Resolved Mass Spectra Using Variable Selection and Quantized Peptide Mass Distribution. *Analytical Chemistry*, 78: 3385–3392, 2006.

E. Grushka. Characterization of Exponentially Modified Gaussian Peaks in Chromatography. *Analytical Chemistry*, 44:1733–1738, 1972.

V. Marco and G. Bombi. Mathematical functions for the representation of chromatographic peaks. *Journal of Chromatography A*, 931:1–30, 2001.

B. Renard, M. Kirchner, H. Steen, J. Steen, and F. Hamprecht. NITPICK: peak identification for mass spectrometry data. *BMC Bioinformatics*, 9:355, 2008.

O. Schulz-Trieglaff, R. Hussong, C. Gröpl, A. Hildebrandt, and K. Reinert. A Fast and Accurate Algorithm for the Quantification of Peptides from Mass Spectrometry Data. In *Proceedings of the Eleventh Annual International Conference on Research in Computational Molecular Biology (RECOMB 2007)*, pages 437–487, 2007.

M. Senko, S. Beu, and F. McLafferty. Determination of Monoisotopic Masses and Ion Populations for Large Biomolecules from Resolved Isotopic Distributions. *Journal of the American Society for Mass Spectrometry*, 6:229–233, 1995.

R. Tibshirani. Regression shrinkage and variable selection via the lasso. *Journal of the Royal Statistical Society Series B*, 58:671–686, 1996.