

# IPPD package vignette

Martin Slawski <sup>1</sup> [ms@cs.uni-sb.de](mailto:ms@cs.uni-sb.de)  
Rene Hussong <sup>2</sup> [rene@bioinf.uni-sb.de](mailto:rene@bioinf.uni-sb.de)  
Matthias Hein <sup>1</sup> [hein@cs.uni-sb.de](mailto:hein@cs.uni-sb.de)

<sup>1</sup> Department of Computer Science, Machine Learning Group

<sup>2</sup> 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

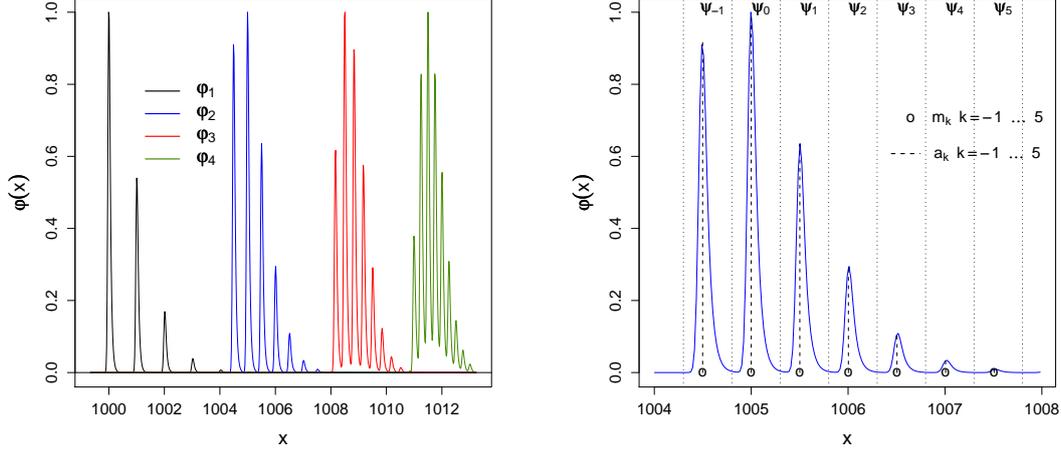


Figure 1: Illustration of the template construction as described in the text. The left panel depicts different templates of different charge states (1 to 4). The right panel zooms at the charge two template  $\varphi_2$ .

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

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

where  $\mathbf{\Phi}$  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  $\mathbf{y} \geq 0$ , where ' $\geq$ ' is understood componentwise, all template functions are nonnegative and accordingly  $\boldsymbol{\beta}^* \geq 0$ . Model (1) can equivalently be written as

$$\mathbf{y} = \begin{bmatrix} \mathbf{\Phi}_1 & \dots & \mathbf{\Phi}_C \end{bmatrix} \begin{bmatrix} \beta_1^* \\ \vdots \\ \beta_C^* \end{bmatrix} = \sum_{c=1}^C \mathbf{\Phi}_c \beta_c^*, \quad (2)$$

where  $\mathbf{\Phi}_c, \beta_c^*$  denote the matrix of template functions and weight vector to fit isotopic patterns of a particular charge state  $c$ ,  $c = 1, \dots, C$ . Each submatrix  $\mathbf{\Phi}_c$  can in turn be divided into columns  $\boldsymbol{\varphi}_{c,1}, \dots, \boldsymbol{\varphi}_{c,p_c}$ , where the entries of each column vector store the evaluations of a template  $\varphi_{c,j}$ ,  $j = 1, \dots, p_c$ , at the  $x_i$ ,  $i = 1, \dots, n$ . 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

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

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 average 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  $\kappa$  equals 1 Dalton ( $\approx 1.003$ ). 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 average 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:

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

The parameter to be determined is  $\theta_{c,j} = \sigma_{c,j} > 0$ . 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:

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

The EMG function involves a vector of three parameters  $\theta_{c,j} = (\alpha_{c,j}, \sigma_{c,j}, \mu_{c,j})^\top \in \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}$ . 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  $\theta_{c,j}$ , which are usually unknown, from the data as good as possible. To this end, we model each component  $\theta_l$  of  $\theta$  as a linear combination of known functions  $g_{l,m}$  of  $x = m/z$  and an error component  $\varepsilon_l$ , i.e.

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

In the case of no prior knowledge about the  $g_{l,m}$ , we model  $\theta_l$  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  $\mathcal{R}_r \subset \{1, \dots, n\}$ ,  $r = 1, \dots, R$ , 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:

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

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

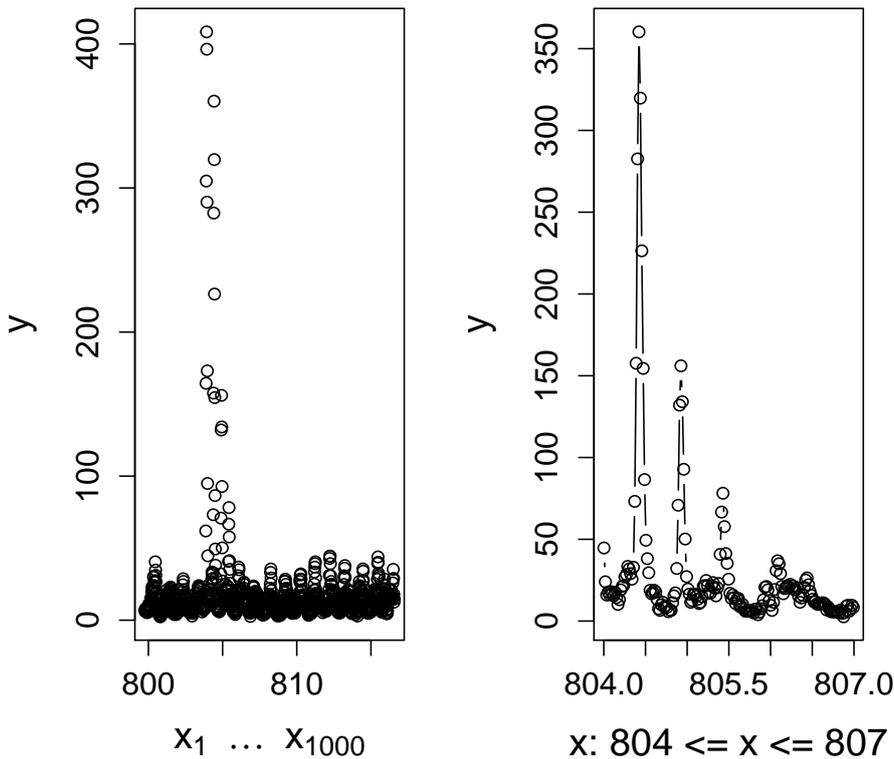
3. The sequence  $\{\hat{x}_r, \hat{\theta}_r\}_{r=1}^R$  is then used as input for the estimation of the parameters  $\nu_{l,m}$  in model (6).

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]~~1dots~~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  $\sigma$  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))
```

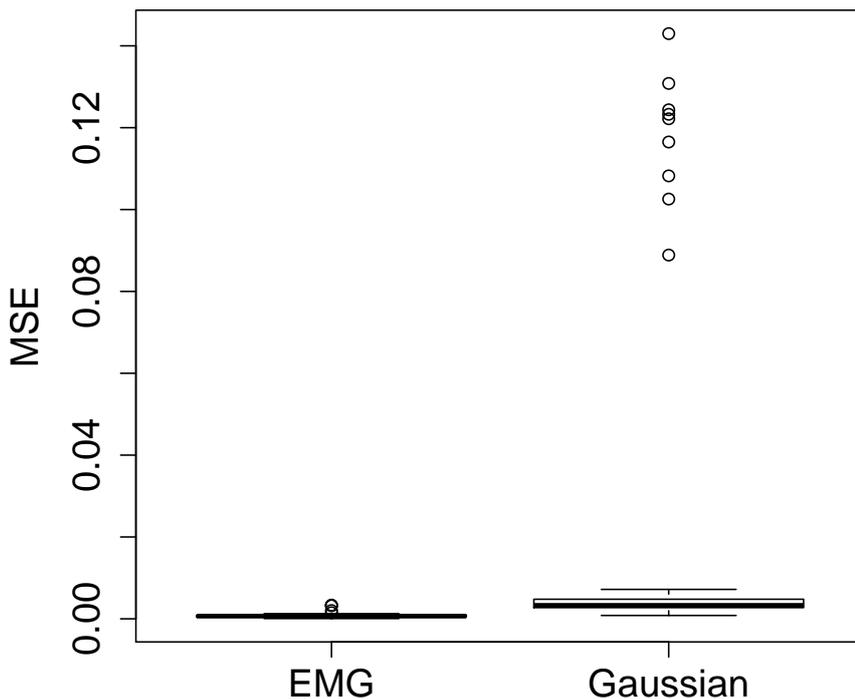
Inspecting the results, we find that  $R = 55$  peak regions are used to fit an EMG parameter model. Moreover, it turns out that the EMG model is a more appropriate peak model for the data when visually comparing the list of mean residual sums of squares of the EMG fits and the Gauss fits extracted from `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.

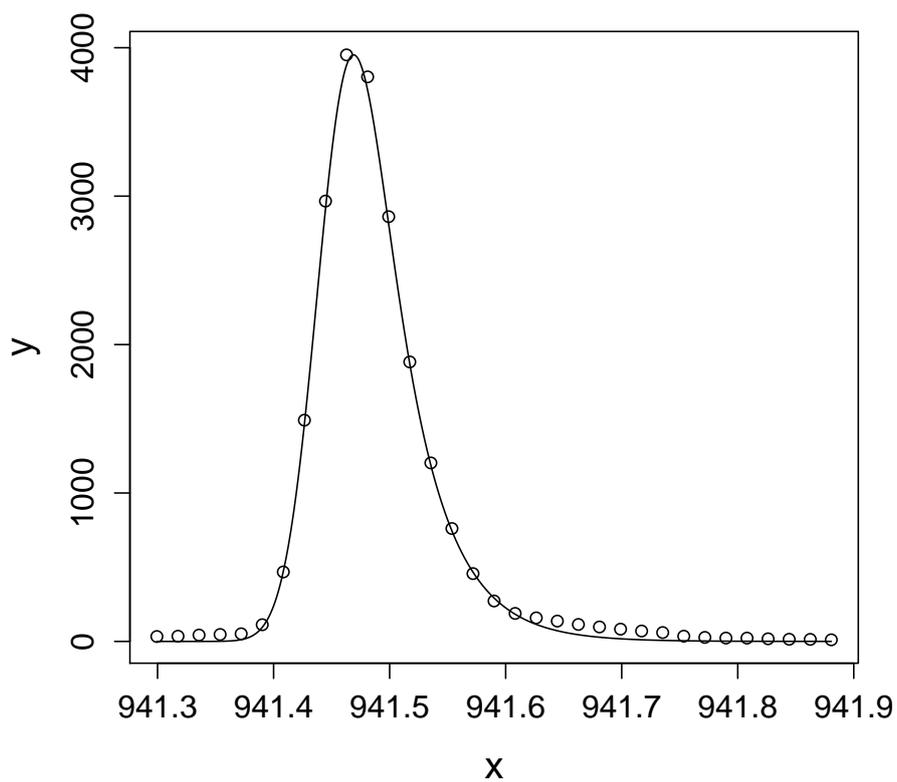
```
R> show(fitEMG)
```

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>
```



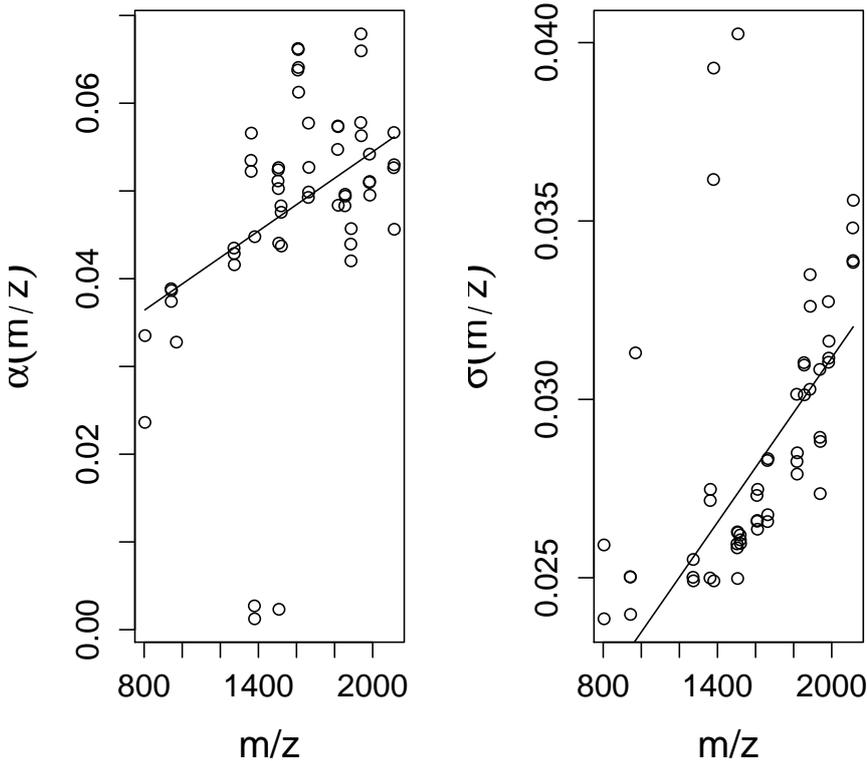
```
R> visualize(fitEMG, type = "peak", cex.lab = 1.5, cex.axis = 1.25)
```



To assess the fit of the two linear models for the EMG parameters  $\alpha$  and  $\sigma$ , 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  $\sigma$  seems to be reasonable except for some extreme outliers, the fit for  $\alpha$  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  $\Phi$  according to Eqs. (1) and (3). It then remains to estimate the coefficient vector  $\beta^*$  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  $\beta^*$  by using  $\ell_1$ -regularized regression (Tibshirani [1996]). We here argue empirically that  $\ell_1$  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

$$\hat{\beta} \in \underset{\beta}{\operatorname{argmin}} \|\mathbf{y} - \Phi\beta\|_q^q, \quad q = 1 \text{ or } q = 2, \quad (8)$$

subject to  $\beta \geq 0$ .

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  $\mathbf{y}$ . Concerning the choice of  $q$ , we point out that  $q = 1$  can cope better with deviations from model assumptions, i.e. deviations from the average 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  $\hat{\beta}$ , we define  $\mathcal{M}_c = \{m_{c,j} : \hat{\beta}_{c,j} > 0\} \subset \{x_i\}_{i=1}^n$ ,  $c = 1, \dots, C$ , 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.

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

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  $\psi_{\tilde{m}_{c,g}}$  placed at location  $\tilde{m}_{c,g}$  and weighted by  $\tilde{\beta}_{c,g}$  such that  $\tilde{\beta}_{c,g} \psi_{\tilde{m}_{c,g}}$  approximates the fit of multiple functions  $\{\psi_{m_{c,j}}\}$  best (in a least squares sense).

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

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  $\{\tilde{\beta}_{c,g}\}_{g=1}^{G_c}$ ,  $c = 1, \dots, C$ , discarding all positions where the corresponding coefficients is less than a significance level times the LNL, where the significance 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	79.45890	50.284996	9.411770	5.342778
[2,]	808.2414	808.2414	1	80.38142	50.673969	10.196100	4.969936
[3,]	829.2292	829.2292	1	93.79356	58.521705	9.411770	6.217928
[4,]	842.4935	842.4935	1	65.97511	40.895095	8.366010	4.888244
[5,]	864.4065	864.4065	1	142.85909	87.585262	13.333300	6.568911
[6,]	877.0373	877.0373	1	116.00313	70.668992	8.888890	7.950261
[7,]	908.4247	908.4247	1	77.97007	46.753667	8.104580	5.768796
[8,]	908.9339	908.9339	1	57.32894	34.368394	9.411770	3.651640
[9,]	923.4380	923.4380	1	71.89262	42.793492	7.581700	5.644314
[10,]	924.4543	924.4543	1	72.76684	43.291608	8.366010	5.174702
[11,]	927.4746	927.4746	1	109.43360	65.009654	8.888890	7.313585
[12,]	927.9689	927.9689	1	60.26696	35.793323	9.934640	3.602881
[13,]	941.4672	941.4672	1	6893.65089	4067.068104	12.549000	324.094996

[14,]	941.6761	941.6761	1	136.03924	80.248247	13.594800	5.902863
[15,]	942.4249	942.4249	1	272.67694	160.784835	16.732000	9.609421
[16,]	949.4363	949.4363	1	71.82727	42.206746	7.320260	5.765744
[17,]	952.5318	952.5318	1	76.13360	44.668155	7.843140	5.695188
[18,]	954.4819	954.4819	1	106.31324	62.313488	8.104580	7.688676
[19,]	963.4553	963.4553	1	148.97453	86.924780	6.535950	13.299487
[20,]	967.4826	967.4826	1	145.37196	84.652489	8.104580	10.445019
[21,]	969.4695	969.4695	1	262.77851	152.864019	10.457500	14.617645
[22,]	985.4424	985.4424	1	125.59410	72.473325	7.320260	9.900376
[23,]	991.5025	991.5025	1	143.74405	82.690367	9.411770	8.785847
[24,]	992.0244	992.0244	1	126.08678	72.514205	10.196100	7.111955
[25,]	999.4665	999.4665	1	153.67307	88.043413	7.320260	12.027361
[26,]	1023.4509	1023.4509	1	72.99416	41.209283	6.535950	6.305018
[27,]	1057.4478	1057.4478	1	61.60149	34.042842	7.320260	4.650496
[28,]	1086.5573	1086.5573	1	163.04078	88.438962	6.013070	14.707789
[29,]	1125.5187	1125.5187	1	82.20314	43.613106	6.797390	6.416155
[30,]	1151.4789	1151.4789	1	97.87723	51.215800	5.751630	8.904571
[31,]	1168.6235	1168.6235	1	88.01819	45.633080	6.274510	7.272772
[32,]	1192.7011	1192.7011	1	73.30971	37.512167	6.535950	5.739360
[33,]	1271.6609	1271.6609	1	3374.20339	1646.932946	10.457500	157.488209
[34,]	1297.6800	1297.6800	1	143.25269	68.805509	7.581700	9.075209
[35,]	1360.7611	1360.7611	1	3817.39884	1720.368091	13.071900	131.608113
[36,]	1361.7379	1361.7379	1	1393.45113	627.324757	20.653600	30.373628
[37,]	1378.8378	1378.8378	1	2998.73102	1325.654706	15.686300	84.510350
[38,]	1394.8413	1394.8413	1	133.40450	57.965682	9.150330	6.334819
[39,]	1474.6387	1474.6387	1	154.37951	63.769121	9.934640	6.418866
[40,]	1484.6606	1484.6606	1	554.65061	227.684870	11.764700	19.353224
[41,]	1500.6588	1500.6588	1	187.03620	76.024913	14.902000	5.101658
[42,]	1501.6659	1501.6659	1	237.91680	96.656604	20.130700	4.801453
[43,]	1502.6668	1502.6668	1	6512.70620	2644.287728	27.973900	94.526960
[44,]	1506.9383	1506.9383	1	1362.42146	551.983402	18.039200	30.599107
[45,]	1518.6637	1518.6637	1	3192.52250	1285.364924	14.902000	86.254524
[46,]	1519.6113	1519.6113	1	167.79384	67.525123	15.424800	4.377698
[47,]	1524.6527	1524.6527	1	142.46347	57.180521	9.411770	6.075427
[48,]	1534.6603	1534.6603	1	271.32355	108.325420	12.287600	8.815832
[49,]	1546.6550	1546.6550	1	180.74875	71.702500	10.196100	7.032346
[50,]	1588.8538	1588.8538	1	206.85345	80.205902	12.287600	6.527386
[51,]	1589.8317	1589.8317	1	205.68882	79.708272	14.379100	5.543342
[52,]	1606.8601	1606.8601	1	47661.12697	18290.168062	27.451000	666.284218
[53,]	1622.8482	1622.8482	1	282.49724	107.345400	8.104580	13.245029
[54,]	1628.8462	1628.8462	1	197.48103	74.760743	7.843140	9.531991
[55,]	1632.8754	1632.8754	1	231.62661	87.466614	9.673200	9.042159
[56,]	1643.8448	1643.8448	1	295.36931	110.774344	10.980400	10.088371
[57,]	1650.8347	1650.8347	1	219.31214	81.887890	9.411770	8.700583
[58,]	1660.8520	1660.8520	1	346.29800	128.489101	15.424800	8.330034
[59,]	1661.8523	1661.8523	1	7082.14080	2625.725897	20.392200	128.761286
[60,]	1675.8106	1675.8106	1	140.11671	51.493039	7.058820	7.294851
[61,]	1683.8293	1683.8293	1	115.18657	42.113371	6.535950	6.443344
[62,]	1687.8674	1687.8674	1	138.47946	50.497329	7.581700	6.660423
[63,]	1712.6676	1712.6676	1	116.18455	41.773898	6.274510	6.657715

[64,]	1717.8060	1717.8060	1	100.80335	36.153780	6.274510	5.762008
[65,]	1718.8742	1718.8742	1	78.34833	28.085396	6.535950	4.297064
[66,]	1753.7102	1753.7102	1	89.35317	31.479997	5.490200	5.733853
[67,]	1777.8665	1777.8665	1	106.10423	36.931307	6.013070	6.141839
[68,]	1798.8819	1798.8819	1	175.36497	60.387786	7.581700	7.964940
[69,]	1815.9052	1815.9052	1	9222.27876	3149.106044	24.052300	130.927439
[70,]	1831.9004	1831.9004	1	327.93340	111.106727	5.751630	19.317433
[71,]	1837.8935	1837.8935	1	368.32923	124.421177	6.274510	19.829624
[72,]	1847.8954	1848.9034	1	373.94181	125.769651	6.274510	20.044538
[73,]	1852.9575	1853.9655	1	162.82782	54.759239	10.980400	4.986999
[74,]	1853.9657	1854.9737	1	2843.34104	956.201588	12.549000	76.197433
[75,]	1869.9595	1870.9675	1	428.17533	143.949708	6.013070	23.939470
[76,]	1885.0257	1886.0337	1	2133.58715	717.090696	9.150330	78.367742
[77,]	1885.9545	1886.9625	1	157.60487	52.969410	9.673200	5.475893
[78,]	1897.9419	1898.9499	1	73.78238	24.791890	4.444440	5.578181
[79,]	1901.0127	1902.0207	1	88.40188	29.699486	4.705880	6.311144
[80,]	1919.0078	1920.0158	1	146.81333	49.261997	6.013070	8.192487
[81,]	1937.0230	1938.0310	1	10569.98535	3542.201623	27.712400	127.820096
[82,]	1953.0164	1954.0244	1	116.41395	38.969734	3.660130	10.647090
[83,]	1958.9972	1960.0052	1	146.46775	49.009972	3.921570	12.497539
[84,]	1963.0361	1964.0441	1	219.18407	73.321150	6.013070	12.193630
[85,]	1969.9485	1970.9565	1	300.77164	100.565370	4.444440	22.627231
[86,]	1981.0615	1982.0695	1	244.80035	81.786724	13.594800	6.016030
[87,]	1982.0622	1983.0702	1	5026.34105	1679.160771	15.686300	107.046325
[88,]	1994.0448	1995.0528	1	131.82206	44.002047	3.921570	11.220518
[89,]	1995.0238	1996.0318	1	80.08301	26.729770	3.660130	7.302957
[90,]	1998.0483	1999.0563	1	86.94175	29.013012	3.398690	8.536528
[91,]	2004.0431	2005.0511	1	76.49362	25.515750	3.660130	6.971269
[92,]	2008.0804	2009.0884	1	135.36218	45.139491	3.660130	12.332756
[93,]	2039.0826	2040.0906	1	56.21675	18.706168	2.875820	6.504638
[94,]	2052.9999	2054.0079	1	68.78863	22.867100	2.352940	9.718522
[95,]	2092.1287	2093.1367	1	81.25596	26.937633	3.660130	7.359747
[96,]	2098.0541	2099.0621	1	76.68373	25.411241	3.137260	8.099820
[97,]	2105.0082	2106.0162	1	172.43331	57.101648	3.398690	16.801076
[98,]	2109.1722	2110.1802	1	245.65255	81.313092	11.764700	6.911616
[99,]	2110.1592	2111.1672	1	5284.45733	1749.017259	14.117600	123.889135
[100,]	2126.1453	2127.1533	1	79.78798	26.365482	2.352940	11.205335
[101,]	2132.1398	2133.1478	1	59.98861	19.810828	2.352940	8.419606
[102,]	2136.1764	2137.1844	1	73.48334	24.257467	2.614380	9.278478
[103,]	2154.1284	2155.1364	1	41.38708	13.637264	2.091500	6.520327
[104,]	2157.9980	2159.0060	1	41.70229	13.735678	1.830070	7.505548
[105,]	2166.1065	2167.1145	1	73.99924	24.353354	2.352940	10.350181
[106,]	2172.1089	2173.1169	1	33.40541	10.987153	1.830070	6.003679
[107,]	2188.0666	2189.0746	1	36.97194	12.140274	1.633987	7.429845
[108,]	2211.1144	2212.1224	1	68.45944	22.420409	2.091500	10.719775
[109,]	2226.1425	2227.1505	1	102.26572	33.428708	1.633987	20.458362
[110,]	2233.0991	2234.1071	1	33.83169	11.049269	1.633987	6.762150
[111,]	2257.0072	2258.0152	1	73.28274	23.861532	1.633987	14.603252
[112,]	2283.2155	2284.2235	1	182.53420	59.237702	2.091500	28.323071
[113,]	2284.1745	2285.1825	1	24.06399	7.808532	2.091500	3.733460

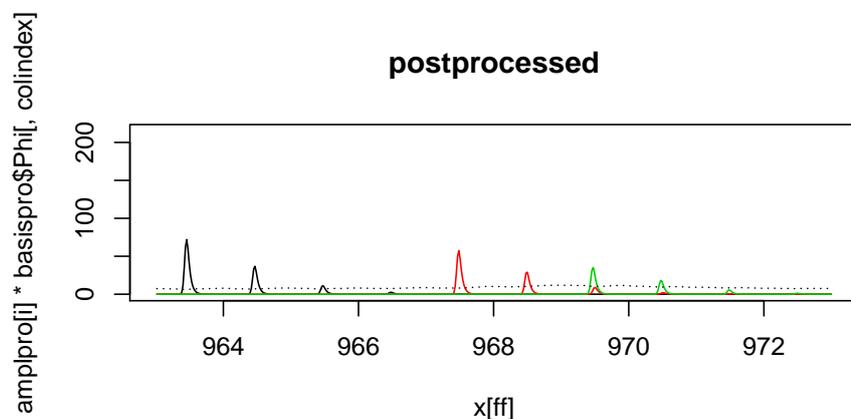
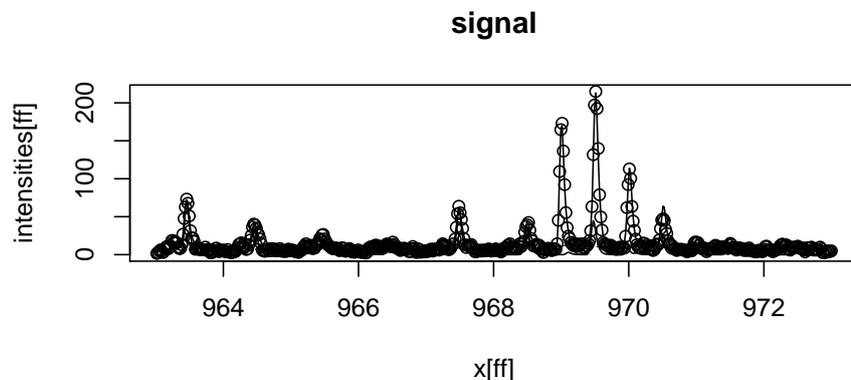
```
[114,] 2427.2620      2428.2700      1  50.98864  16.210521  1.633987  9.920835
```

```
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)
```

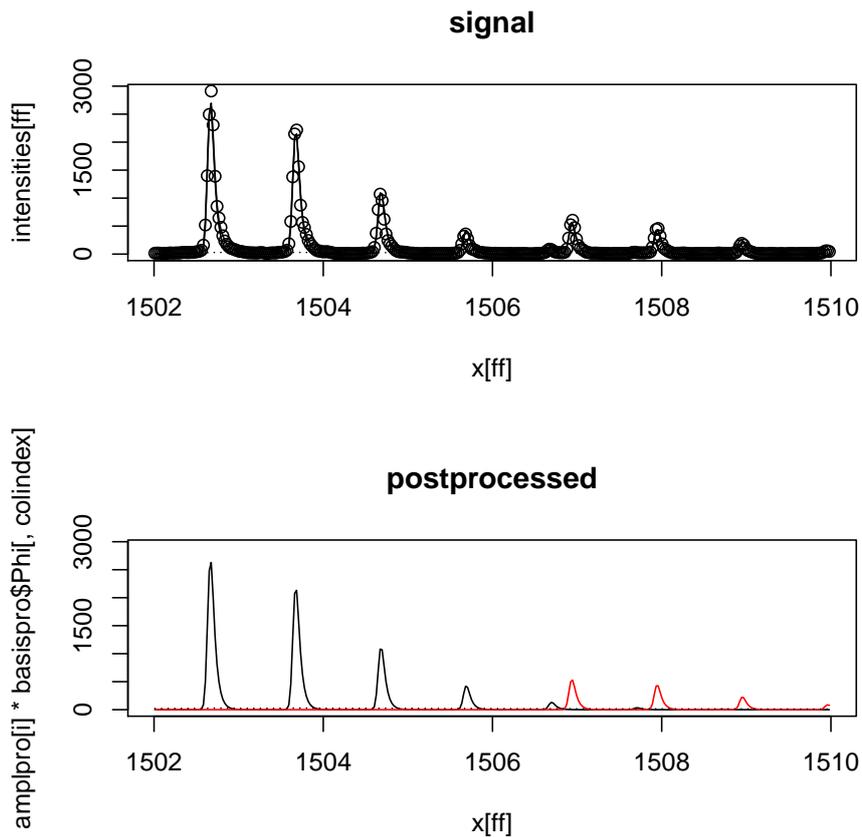
```
R>
```



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

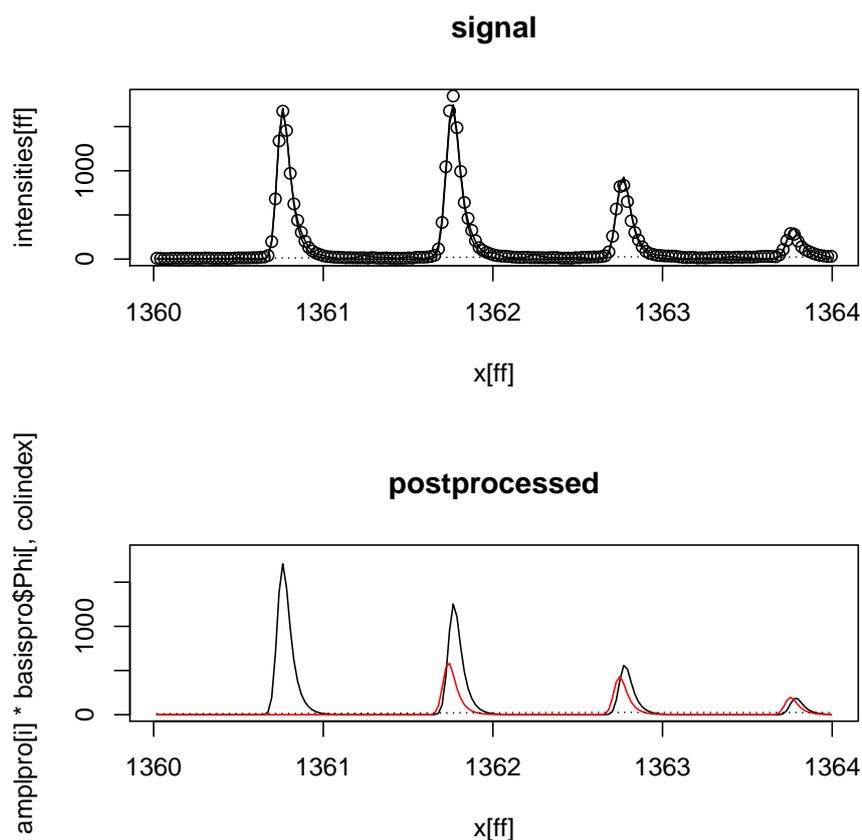
```
R>
```

```
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.