Fitting Baselines to Spectra

Claudia Beleites <Claudia.Beleites@chemometrix.gmbh>
DIA Raman Spectroscopy Group, University of Trieste/Italy (2005 – 2008)
Spectroscopy · Imaging, IPHT, Jena/Germany (2008 – 2017)
ÖPV, JKI, Berlin/Germany (2017 – 2019)
Arbeitskreis Lebensmittelmikrobiologie und Biotechnologie, Hamburg University, Hamburg/Germa
Chemometric Consulting and Chemometrix GmbH,
Wölfersheim/Germany (since 201

November 27, 2020

Contents

1

1

1 Introduction

This document discusses baseline correction methods that can be used with *hyperSpec*. *hyperSpec* provides two fitting functions for polynomial baselines, `spc.fit.poly` and `spc.fit.poly.below`. Another possibility is `spc.rubberband`, a "rubberband" method that determines support points by finding the convex hull of each spectrum. The baselines are then piecewise linear or (smoothing) splines through the support points.

Please note that a specialized package for baseline fitting, *baseline*[1], exists that provides many more methods to fit baselines to spectroscopic data. Using *baseline* with *hyperSpec* objects is demonstrated in `vignette ("hyperspec")`.

2 Polynomial Baselines

In contrast to many other programs that provide baseline correction methods, *hyperSpec*'s polynomial baseline functions do least squares fits. However, the baselines can be forced through particular points, if this behaviour is needed.

The main difference between the two functions is that `spc.fit.poly` returns a least squares fit through the complete spectrum that is given in *fit.to* whereas `spc.fit.poly.below` tries to find appropriate spectral regions to fit the baseline to.

2.1 Syntax & parameters

```
spc.fit.poly (fit.to, apply.to = fit.to, poly.order = 1

spc.fit.poly.below (fit.to, apply.to = fit.to, poly.order = 1, npts.min = NULL,
                    noise = 0)
```

`fit.to`:	`hyperSpec` object with the spectra whose baselines are to be fitted.
`apply.to`:	`hyperSpec` object giving the spectral range, on which the baselines should be evaluated. If `apply` is `NULL`, a `hyperSpec` object with the polynomial coefficients is returned instead of evaluated baselines.
`poly.order`:	polynomial order of the baselines
`npts.min`:	minimal number of data points per spectrum to be used for the fit. `npts.min` defaults to the larger of 3 times (`poly.order` + 1) or $\frac{1}{20th}$ of the number of data points per spectrum. If `npts.min` ≤ poly.order, a warning is issued and `npts.min <- poly.order + 1` is used.
`noise`:	a vector giving the amount of noise, see below.

2.2 General Use

Both functions fit the polynomial to the spectral range given in *hyperSpec* object *fit.to*. If *apply.to* is not `NULL`, the polynomials are evaluated on the spectral range of *apply.to*. Otherwise, the polynomial coefficients are returned.

Subtracting the baseline is up to the user, it is easily done as *hyperSpec* provides the `-` (minus) operator.

2.3 Fitting polynomial baselines using least squares

Commonly, baselines are fit using (single) support points that are specified by the user. Also, usually $n+1$ support point is used for a polynomial of order n. This approach is appropriate for spectra with high signal to noise ratio.

Such a baseline can be obtained by restricting the spectra in *fit.to* to the respective points (see figure 1):

```
> bl <- spc.fit.poly (chondro [c (1, 101),, c (633, 1788)], chondro [c (1, 101)])
> plot (chondro [c (1, 101)], plot.args = list (ylim = c(200, 600)), col = 1 : 2)
> plot (chondro [c (1, 101),, c(633, 1788)], add = TRUE, col = 1:2,
+       lines.args = list (type = "p", pch = 20))
> plot (bl, add = TRUE, col = 1 : 2)
```

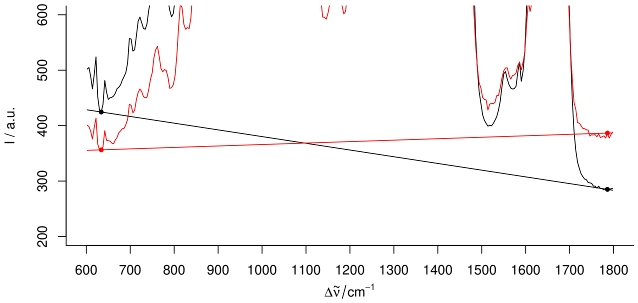


Figure 1 Fitting a linear baseline through two points. If the signal to noise ratio is not ideal, wavelengths that work fine for one spectrum (black) may not be appropriate for another (red).

However, if the signal to noise ratio is not ideal, a polynomial with $n+1$ supporting points (i.e. with zero degrees of freedom) is subject to a considerable amount of noise. If on the other hand, more data points consisting of baseline only are available, the uncertainty on the polynomial can be reduced by a least squares fit.

Both `spc.fit.poly` and `spc.fit.poly.below` therefore provide least squares fitting for the polynomial.

`spc.fit.poly` fits to the whole spectral region of *fit.to*. Thus, for baseline fitting the spectra need to be cut to appropriate wavelength ranges that do not contain any signal.

In order to speed up calculations, the least squares fit is done by using the Vandermonde matrix and solving the equation system by `qr.solve`.

This fit is not weighted. A spectral region with many data points therefore has greater influence on the resulting baseline than a region with just a few data points. It is up to the user to decide whether this should be corrected for by selecting appropriate numbers of data points (e.g. by using replicates of the shorter spectral region).

2.4 The mechanism of automatically fitting the baseline in `spc.fit.poly.below`

`spc.fit.poly.below` tries to automatically find appropriate spectral regions for baseline fitting. This is done by excluding spectral regions that contain signals from the baseline fitting. The idea is that all data points that lie above a fitted polynomial (initially through the whole spectrum, then through the remaining parts of the spectrum) will be treated as signal and thus be excluded from the baseline fitting.

The supporting points for the baseline polynomials are calculated iteratively:

1. A polynomial of the requested order is fit to the considered spectral range, initially to the whole spectrum given in *fit.to*

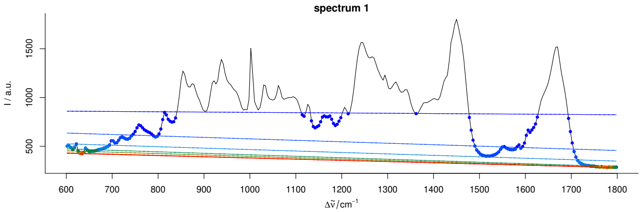


Figure 2 Iterative fitting of the baseline. The dots give the supporting points for the next iteration's baseline, color: iterations.

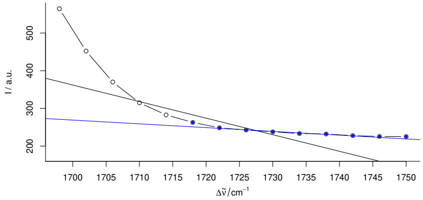


Figure 3 Influence of `fit.to` on the baseline polynomial. The black baseline is fit to the spectral range 1700 – 1750 cm^{-1}, the blue to 1720 – 1750 cm^{-1} only.

2. Only the parts of the spectrum that lie below this polynomial plus the `noise` are retained as supporting points for the next iteration.

These two steps are repeated until either

- no further points are excluded, or
- the next polynomial would have less than `npts.min` supporting points.

The baselines and respective supporting points for each iteration of `spc.fit.poly.below (chondro [1], poly.order = 1)` are shown in figure 2.

2.5 Specifying the spectral range

It is possible to exclude spectral regions that do not contribute to the baseline from the fitting, while the baseline is used for the whole spectrum. This selection of appropriate spectral regions is essential for `spc.fit.poly`. But also `spc.fit.poly.below` can benefit from narrower spectral ranges: the fitting gains speed. The default value for *npts.min* depends on the number of data points per spectrum. Thus one should also consider requiring more support points than the default value suggests.

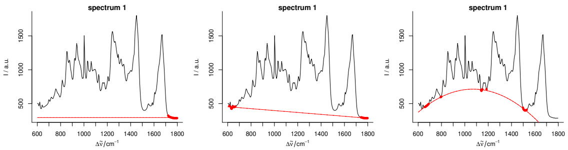


Figure 4 Baseline polynomial fit to the first spectrum of the chondro data set of order 0 – 2 (left to right). The dots indicate the points used for the fitting of the polynomial.

```
> system.time (spc.fit.poly.below (chondro, NULL, npts.min = 20))

   user  system elapsed
  0.372   0.021   0.393

> system.time (spc.fit.poly.below (chondro [,, c (min ~ 700, 1700 ~ max)], NULL, npts.min = 20))

   user  system elapsed
  0.201   0.000   0.201
```

The choice of the spectral range in `fit.to` influences the resulting baselines to a certain extent, as becomes clear from figure 3.

2.6 Fitting polynomials of different orders

Figure 4 shows the resulting baseline polynomial of `spc.fit.poly.below (chondro [1], poly.order = order)` with `order` = 0 to 3 for the first spectrum of the chondro data set.

2.7 The noise level

Besides defining a minimal number of supporting points, a "noise level" may be given. Consider a spectral range consisting only of noise. The upper part of figure 5 illustrates the problem. As the baseline fitting algorithm cannot distinguish between noise and real bands appearing above the fitted polynomial, the resulting baseline (black) is too low if the `noise` parameter is not given.

Setting the noise level to 4 (2 standard deviations), the fitting converges immediately with a much better result. The resulting baselines for `spc.fit.poly.below (chondro [1], poly.order = 1, noise = 12)` of the whole spectrum are shown in the middle and lower part of figure 5

`noise` may be a single value for all spectra, or a vector with the noise level for each of the spectra separately.

3 Rubberband Method

Particularly Raman spectra often show increasing background towards $\Delta\tilde{\nu} = 0$. In this case, polynomial baselines often either need high order or residual background is left in the spectra.

In that case, smoothing splines fitted through the supporting points are a good alternative.

For the `paracetamol` spectrum (fig. 6), a noise level of 300 counts and the equivalent of 20 degrees of freedom work well.

5

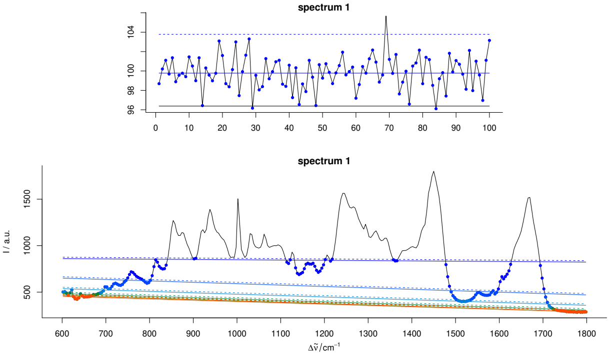


Figure 5 Iterative fitting of the baseline with noise level. Upper part: effects of the noise parameter on the baseline of a spectrum consisting only of noise and offset: without giving `noise` the resulting baseline (black) is clearly too low. A noise level of 4 results in the blue baseline. The lower part show the baseline fitting with noise level on the complete spectrum. Colour: iterations, dots: supporting points for the respectively next baseline. Dashed: baseline plus noise. All points above this line are excluded from the next iteration.

```
> bl <- spc.rubberband (paracetamol [,, 175 ~ 1800], noise = 300, df = 20)
```

However, there is possibly some background left between 1200 and 1750 cm^{-1} where the original spectrum is slightly concave. This can be corrected by bending the spectrum before applying the rubberband correction (fig. 7):

```
> bend <- 5e4 * wl.eval (paracetamol [,, 175 ~ 1800], function (x) x^2, normalize.wl=normalize01)
> bl <- spc.rubberband (paracetamol [,, 175 ~ 1800] + bend) - bend
```

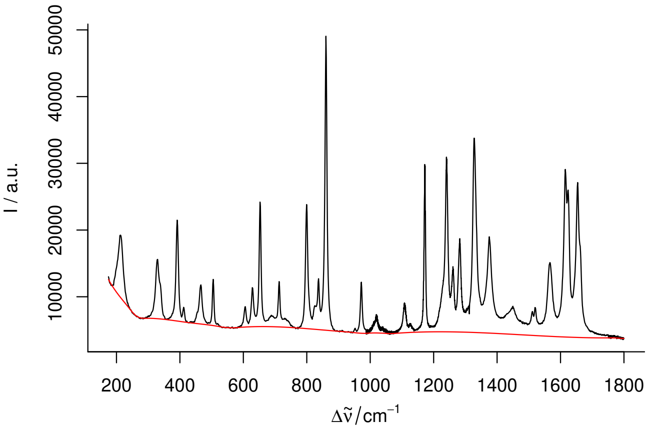
References

[1] Kristian Hovde Liland, Trygve Almøy, and Bjørn-Helge Mevik. Optimal choice of baseline correction for multivariate calibration of spectra. *Applied Spectroscopy*, 64:1007–1016, 2010.

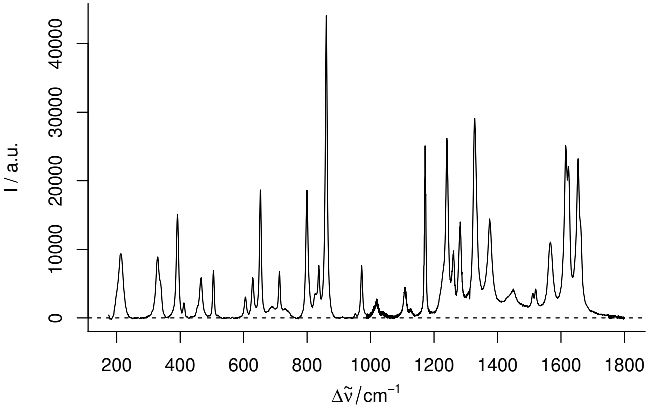
Session Info

```
R version 3.6.3 (2020-02-29)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.5 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/libopenblasp-r0.2.20.so
```

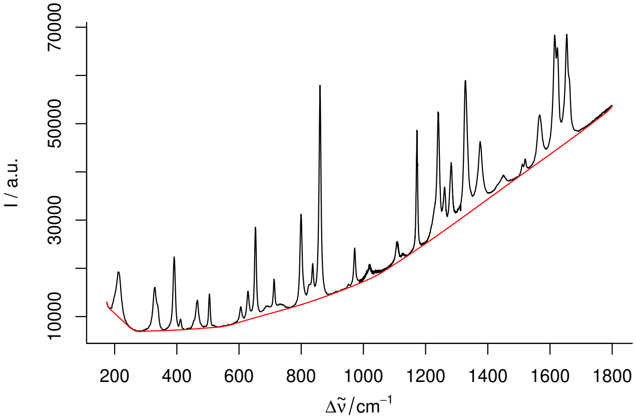


(a) `paracetamol` with the `rubberband` fitted baseline.

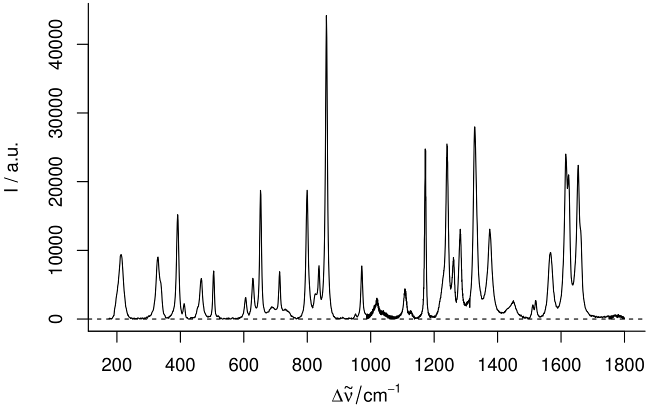


(b) Corrected spectrum.

Figure 6 Rubberband baselines for the paracetamol spectrum.



(a) Bent `paracetamol` spectrum and rubberband baseline.



(b) Corrected spectrum.

Figure 7 Rubberband baselines for the paracetamol spectrum after bending.

```
locale:
 [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C               LC_TIME=de_DE.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=de_DE.UTF-8
 [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] hyperSpec_0.99-20201127 xml2_1.3.2              ggplot2_3.3.2           lattice_0.20-41

loaded via a namespace (and not attached):
 [1] magrittr_2.0.1      tidyselect_1.1.0    munsell_0.5.0       colorspace_2.0-0
 [5] R.cache_0.14.0      R6_2.5.0            jpeg_0.1-8.1        rlang_0.4.8
 [9] dplyr_1.0.2         tools_3.6.3         gtable_0.3.0        png_0.1-7
[13] R.oo_1.24.0         latticeExtra_0.6-29 withr_2.3.0         ellipsis_0.3.1
[17] lazyeval_0.2.2      digest_0.6.27       tibble_3.0.4        lifecycle_0.2.0
[21] crayon_1.3.4        R.rsp_0.44.0        RColorBrewer_1.1-2  purrr_0.3.4
```

```
[25] vctrs_0.3.5         R.utils_2.10.1      testthat_3.0.0      glue_1.4.2
[29] compiler_3.6.3      pillar_1.4.7        generics_0.1.0      scales_1.1.1
[33] R.methodsS3_1.8.1   pkgconfig_2.0.3
```