Calibration of Quinine Fluorescence Emission Vignette for the Data Set `flu` of the R package `hyperSpec`

C. Beleites, DMRN, Università degli Studi di Trieste, Trieste/I

July 2, 2009

This vignette gives an example how to

- write an import function for a spectrometer manufacturer's proprietary ASCII files,
- add further data columns to the spectra, and
- set up a linear calibration (inverse least squares).

The data set `flu` in `hyperSpec` consists of 6 fluorescence emission spectra of quinine solutions. They were acquired during an student practicum and were kindly provided by M. Kammer.

The concentrations of the solutions range from 0.05 mg/l to 0.30 mg/l. Spectra were acquired with a Perkin Elmer LS50-B fluorescence spectrometer at 350 nm excitation.

In order to work with `hyperSpec`, the library needs to be loaded in `R`:

```
> library (graphics)
> library (hyperSpec)
Package hyperSpec, version 0.5

Citation:
Claudia Beleites and Valter Sergo: Hyperspectral Data Sets in R: hyperSpec, JStatSoft,(2009); V.  0.5.

'citation("hyperSpec")' will give you also an BibTeX entry.

To get started, try
   help ("hyperSpec")
   help (package = "hyperSpec")
   vignette (package = "hyperSpec")

The project is hosted on http://r-forge.r-project.org/projects/hyperspec/
and the article mentioned above is openly accessible at http://
```

1 Writing an Import Function

Now we need to import the spectra. They are in Perkin Elmer's ASCII file format, one spectrum per file. The files are completely ASCII text, with the actual spectra starting at line 55.

The function should automatically read in all files specified by a pattern, such as `*.txt`. In order to gain some speed, the spectra matrix is preallocated after the first file is read. Also, the number of header lines are predefined instead of searching for the line after `#DATA`.

Note, that labels giving the correct units (e.g. for axis labels) are set. The label with the special name `.wavelength` corresponds to the wavelength axis, all data columns should have a label with the same name. The spectra are always in a data column called `spc`.

```
> read.PE <- function (files = "*.txt", skip = 54) {
+         files <- Sys.glob (files)
+
+         buffer <- matrix (scan (files [1], skip = skip), ncol = 2, byrow = TRUE)
+
+         wavelength <- buffer [, 1]
+         spc <- matrix (ncol = nrow (buffer), nrow = length (files))
+
+         spc [1, ] <-  buffer [, 2]
+
+         for (f in seq (along = files)[-1]) {
+
+                 buffer <- matrix (scan (files [f], skip = skip), ncol = 2, byrow = TRUE)
+
+                 if (! all.equal (buffer [, 1], wavelength))
+                         stop (paste(files [f], "has different wavelengh axis."))
+
+                 spc [f, ] <- buffer[, 2]
+         }
+
+         new ("hyperSpec", wavelength = wavelength, spc = spc,
+                         label = list (.wavelength = expression (lambda[fl] / nm),
+                                         spc = "I / a.u."))
+ }
```

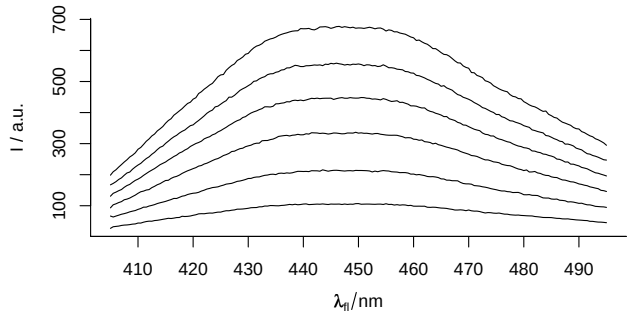
From now on, the function can be used:

```
> flu <- read.PE ("*.txt")
```

Now the spectra are in a hyperSpec object and can be examined e.g. by

```
> flu
hyperSpec object
   6 spectra
   1 data columns
   181 data points / spectrum
wavelength: lambda[fl]/nm [numeric 181]  405.0 405.5 406.0 ... 494.0 494.5 495.0
data:  (6 rows x 1 columns)
   (1) spc: I / a.u. [AsIs matrix 6 x 181] range 27.15000 32.34467 33.37867 ... 676.6457 676.7353 677.4947
```

```
> plot (flu)
```



2 Adding further Data Coumns

The calibration model needs the quinine concentrations for the spectra. This information can be stored together with the specta, and also gets an appropriate label:

```
> flu$c <- seq (from = 0.05, to = 0.30, by = 0.05)
> labels (flu, "c") <- "c / (mg / l)"
> flu
```

```
hyperSpec object
   6 spectra
   2 data columns
   181 data points / spectrum
wavelength: lambda[fl]/nm [numeric 181]  405.0 405.5 406.0 ... 494.0 494.5 495.0
data:  (6 rows x 2 columns)
   (1) spc: I / a.u. [AsIs matrix 6 x 181] range 27.15000 32.34467 33.37867 ... 676.6457 676.7353 677.4947
   (2) c: c / (mg / l) [numeric 6] range 0.05 0.10 0.15 0.20 0.25 0.30
```

Now the `hyperSpec` object `flu` contains two data columns, holding the actual spectra and the respective concentrations. The dollar operator returns such a data column:

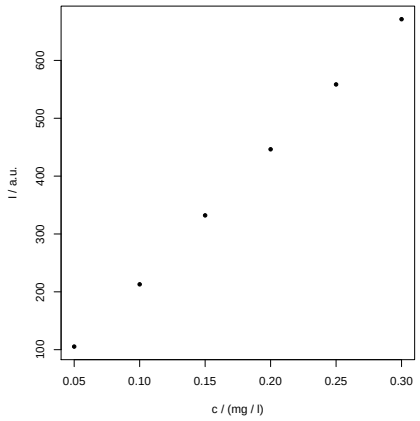
```
> flu$c
[1] 0.05 0.10 0.15 0.20 0.25 0.30
```

3 Linear Calibration

As `R` is developed for the purpose of statistical analysis, tools for a least squares calibration model are readily availabe.

The original spectra range from 405 to 495 nm. However, the intensities at 445 nm are perfect for a univariate calibration. Plotting them over the concentration is done by:

```
> plotc (flu[,,445])
```



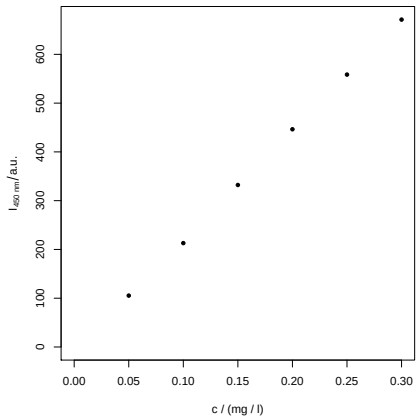
The square bracket operator extracts parts of a `hyperSpec` object. The first coordinate defines which spectra are to be used, the second which data columns, and the third gives the spectral range.

We discard all the wavelengths but 445 nm:

```
> flu <- flu [,,445]
```

The plot could be enhanced by annotating the ordinate with the emission wavelength. Also the axes should start at the origin, so that it is easier to see whether the calibration function will go through the origin:

```
> plotc (flu, zlab = expression (I ["450 nm"] / a.u.),
+            plot.dots = list (xlim = range (0, flu$c), ylim = range (0, flu$spc)))
```



The actual calibration is a linear model, which can be fitted by dhe `R` function `lm`. `lm` needs a *formula* that specifies which data columns are dependent and independent variables.

The normal calibration plot gives the emission intensity as a function of the concentration, and the calibration function thus models $I = f(c)$, i. e. $I = mc + b$ for a linear calibration. This is then solved for c when the calibration is used.

However, `R`'s linear model is a quite strict in predicting: a model set up as $I = f(c)$ will predict the intensity as a function of the concentration but not the other way round. Thus we set up an inverse calibration model[1]: $c = f(I)$. The corresponding `R` formula is `c ~ I`, or in our case `c ~ spc`, as the intensities are stored in the data column `spc`:

In addition, `lm` (like most `R` model building functions) expects the data to be a `data.frame`.

There are three abbrevations that help to get the parts of the `hyperSpec` object that are frequently needed:

`flu[[]]` returns the spectra matrix. It takes the same indices as `[]`.

`flu$` returns the data as a `data.frame`

`flu$..` returns a `data.frame` that has all data columns but the spectra

```
> flu[[]]
          445
[1,] 105.3617
[2,] 212.9753
[3,] 332.1467
[4,] 446.3653
[5,] 558.4950
[6,] 671.2343
> flu$.
         445    c
1 105.361666 0.05
2 212.975333 0.10
3 332.146666 0.15
4 446.365333 0.20
5    558.495 0.25
6 671.234333 0.30
> flu$..
[1] 0.05 0.10 0.15 0.20 0.25 0.30
```

Putting this together, the calibration model is calculated:

```
> calibration <- lm (c ~ spc, data = flu$.)
```

The `summary` gives a good overview of our model:

```
> summary (calibration)
Call:
lm(formula = c ~ spc, data = flu$.)

Residuals:
         1          2          3          4          5          6
-0.0008475  0.0018421 -0.0005493 -0.0007634 -0.0000591  0.0003772

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.527e-03  1.034e-03    4.38   0.0119 *
spc         4.396e-04  2.383e-06  184.47 5.18e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.001134 on 4 degrees of freedom
Multiple R-squared: 0.9999,        Adjusted R-squared: 0.9999
F-statistic: 3.403e+04 on 1 and 4 DF,  p-value: 5.18e-09
```

[1] As we can safely assume that the error on the concentrations is far larger than the error on the instrument signal, it is actually the correct type of model from the least squares fit point of view.

In order to get predictions for new measurements, a new `data.frame` with the same independent variables (in columns with the same names) as in the calibration data are needed. Then the function `predict` can be used. It can also calculate the prediction interval. If we observe e.g. an intensity of 125 units, the corresponding concentration and its 99 % prediction interval are:

```
> I <- 125
> conc <- predict (calibration, newdata = list (spc = as.matrix(I)), interval = "prediction", level = .99)
> conc
        fit        lwr        upr
1 0.05948115 0.05314837 0.06581394
```

Finally, we can draw the calibration function and its 99 % confidence interval (also via `predict`) together with the prediction example:

```
> plotc (flu, zlab = expression (I ["450 nm"] / a.u.),
+        plot.dots = list (xlim = range (0, flu$c), ylim = range (0, flu$spc)))
> int <- list (spc = as.matrix(min (flu) : max(flu)))
> ci <- predict (calibration, newdata = int, interval = "confidence", level = 0.99)
> matlines (ci, int$spc, col = c ("red","#606060","#606060"), lty = 1)
> # extrapolate to lower intensities
> int <- list (spc = as.matrix(0 : min (flu)))
> ci <- predict (calibration, newdata = int, interval = "confidence", level = 0.99)
> matlines (ci, int$spc, col = c ("red","#606060","#606060"), lty = 3)
> # our example
> lines (conc[-1], rep(I, 2), col = "blue")
> points (conc[1], I, col = "blue", pch = 4, cex = 0.5)
```