Received: 1 January 2016 ■ *Revised: 18 April 2016* ■ *Accepted: 18 April 2016* ■ *Publication: 22 April 2016*

JNIRS JOURNAL OF NEAR INFRARED SPECTROSCOPY

openaccess

Tutorial: Time series hyperspectral image analysis

Ronan Dorrepaal,[a] Cristina Malegori[b] and Aoife Gowen[a]*

[a]Biosystems and Food Engineering, School of Agriculture, Food Science and Veterinary Medicine, University College Dublin, Dublin, Ireland. E-mail: aoife.gowen@ucd.ie

[b]DeFENS Department of Food, Environmental and Nutritional Sciences, Università degli Studi di Milano, Milan, Italy

A hyperspectral image is a large dataset in which each pixel corresponds to a spectrum, thus providing high-quality detail of a sample surface. Hyperspectral images are thus characterised by dual information, spectral and spatial, which allows for the acquisition of both qualitative and quantitative information from a sample. A hyperspectral image, commonly known as a "hypercube", comprises two spatial dimensions and one spectral dimension. The data of such a file contain both chemical and physical information. Such files need to be analysed with a computational "chemometric" approach in order to reduce the dimensionality of the data, while retaining the most useful spectral information. Time series hyperspectral imaging data comprise multiple hypercubes, each presenting the sample at a different time point, requiring additional considerations in the data analysis. This paper provides a step-by-step tutorial for time series hyperspectral data analysis, with detailed command line scripts in the Matlab and R computing languages presented in the supplementary data. The example time series data, available for download, are a set of time series hyperspectral images following the setting of a cement-based biomaterial. Starting from spectral pre-processing (image acquisition, background removal, dead pixels and spikes, masking) and pre-treatments, the typical steps encountered in time series hyperspectral image processing are then presented, including unsupervised and supervised chemometric methods. At the end of the tutorial paper, some general guidelines on hyperspectral image processing are proposed.

Keywords: hyperspectral, chemometrics, Matlab, R, spike-detection, pre-treatment, masking, training, calibration, validation

Introduction

Near infrared (NIR) hyperspectral imaging is a powerful tool for non-destructive analysis, enabling real-time monitoring of spatially resolved spectral information of materials.[1] Hyperspectral images are data rich as each pixel of an acquired image corresponds to a spectrum and can therefore provide high-quality detail of a sample surface.

NIR hyperspectral images have two spatial dimensions in the form of a matrix, where each element of the matrix can be considered to be a pixel of an image. Further to the spatial dimensions of a hyperspectral image is a spectral dimension. A hyperspectral image file is therefore known as a hyperspectral cube, or a "hypercube", as it is three dimensional. The data of such a file contain both chemical and physical information.[2] An example representation of a three-dimensional $3 \times 3 \times 3$ hypercube is shown in Figure 1 (hypercubes typically represent >100 wavelength dimensions, but for clarity only three are shown).

Each matrix at a particular wavelength of the hypercube can be conceptualised as a two-dimensional slice of the hypercube. Each of these two-dimensional slices can be visualised as an image showing the relative spatial absorbance/reflectance intensities at that wavelength. This is illustrated in Figure 2.

Unfolding

As stated, hyperspectral data files are usually called hypercubes due to the three-dimensional structure of their data. However, many of the useful and established chemometric

ISSN: 0967-0335
doi: 10.1255/jnirs.1208

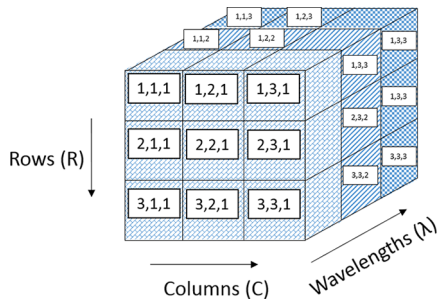

Figure 1. A three-dimensional hypercube with (3 rows) × (3 columns) × (3 wavelengths). The coordinates are shown on each element of the hypercube with the first index representing the rows, the second representing the columns and the third representing the wavelengths.

algorithms are designed to be performed on two-dimensional matrices rather than three-dimensional hypercubes. It is therefore common practice to "unfold" hypercubes such that the three-dimensional information is presented in two dimensions. The unfolding technique is shown schematically in Figure 3. It requires rearranging spectra from a hypercube with three dimensions [(1) rows, (2) columns and (3) wavelengths] to a matrix with two dimensions [(1) rows × columns against (2) wavelengths]. The spectral intensity values contained within the hypercube do not change nor does the number of elements of the dataset. The data are simply reorganised into two dimensions to aid in analysis.

Overview of typical steps in hyperspectral image processing

Typical steps followed in hyperspectral image processing are summarised in Figure 4. The left-hand side of this schematic describes hyperspectral pre-processing, i.e. operations which are carried out on hyperspectral images prior to modelling. The right-hand side summarises the general data processing techniques which may be applied to obtain useful information from hyperspectral images. Although this workflow is not prescriptive (e.g. the order and combination of steps can be changed), it does offer a logical step-by-step flow which will be useful for those learning how to tackle the large volumes of data routinely encountered in hyperspectral imaging. With this in mind, the study considers the steps of Figure 4, using the example dataset described in the "Dataset" section.

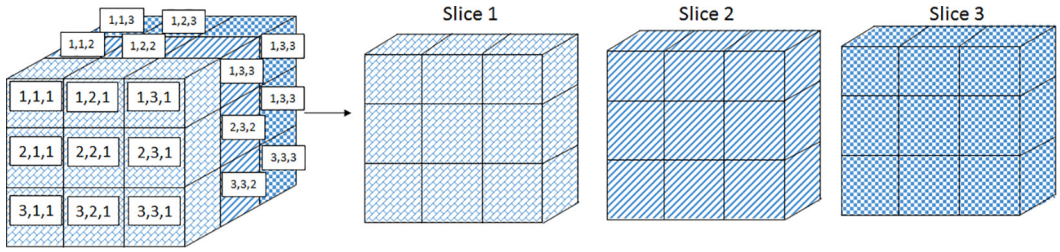

Figure 2. Wavelength slices of a hypercube.

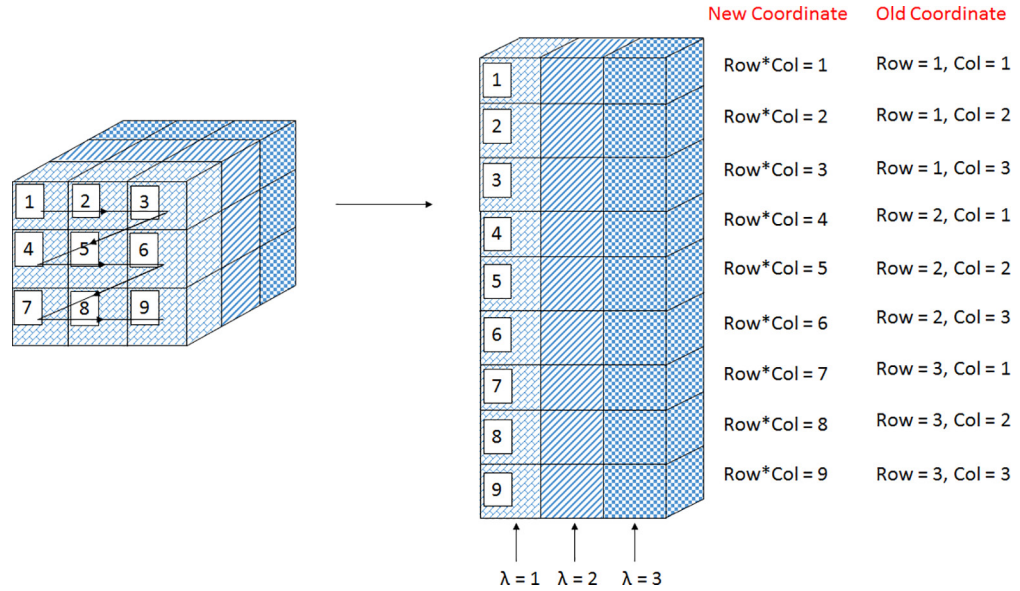

Figure 3. Unfolding a hypercube.

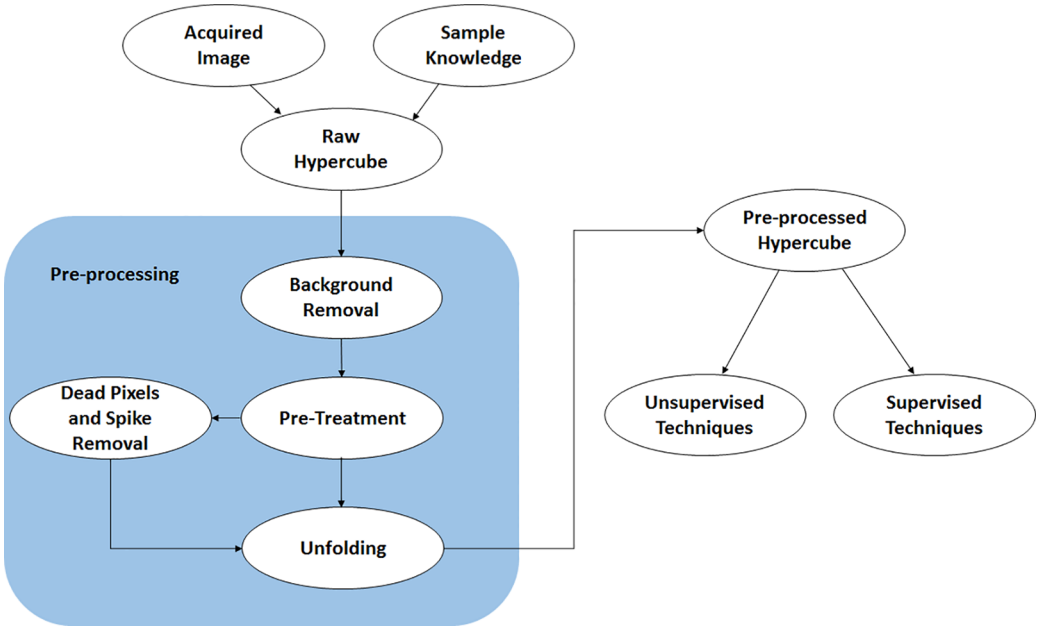

Figure 4. Typical steps involved in hyperspectral image analysis.

Image acquisition

A hyperspectral image may include pixels which relate to the sample as well as pixels which are only background to the sample. The background may be a stage or a prop used to keep the sample in position during imaging. When acquiring a hyperspectral image of a sample, one should strive to optimise conditions such that errors are not introduced before subsequent image analysis.

In order to achieve this goal, it is recommended that the background used be spectrally different from that of the sample. Ideally, the background chosen will be spatially homogenous, making it easier to distinguish pixels pertaining to the sample from those pertaining to the background.

The background should also ideally be homogenous spectrally, meaning that the background should have a flat spectrum. For example, a true-white background will have a flat spectrum as it will reflect/absorb light equally at all wavelengths. A flat background spectrum makes it easier to distinguish the background pixels from the sample pixels. For example, when imaging in the NIR range, as was the case for the present study, a white paper background is not recommended as paper absorbs light in regions of the NIR spectrum.

The chosen background should also be homogeneous over the time required to conduct the experiment. By this it is meant that the background should be stable over time. A background material which decomposes is not suitable.

Further, to increase acquisition quality, the direction of the light should be controlled to avoid saturation of pixels leading to the loss of information. Direct reflection of light could hide important details of the sample and mislead during data processing. For best results, diffuse light is preferred. However, if only direct illumination is possible, halogen lamps angled at 45° to the sample are recommended.

When deciding which acquisition parameters are to be used, it is recommended that some test images are obtained before carrying out a full experimental design. This can make acquisition errors evident, saving time.

Dataset

In this study, images of a magnesium oxychloride cement-based biomaterial[3] were chosen as an example with which to perform the time series data analysis. The cements were prepared as outlined by Li and Chau.[4] Time series hyperspectral imaging data comprise multiple hypercubes, each presenting the sample at a different time point, requiring additional considerations in data analysis.[5] The images were recorded using a line-mapping NIR hyperspectral imaging system, working in the range 880–1720 nm, where the NIR absorption was measured every 7 nm (further details of the instrument can be found in Gowen *et al.*[6]). The hyperspectral acquisitions were made at different cement setting times. The Matlab script used to carry out the analysis is provided in supplementary data 1 and the equivalent R script is provided in supplementary data 2. It is recommended that R is used in conjunction with RStudio, which provides a more user-friendly environment. The dataset can be downloaded from www.ucd.ie/biowater/datasets. The data file M7H16 should be downloaded, and the corresponding Matlab (ML) or R scripts and functions should then be downloaded. The commands for applying all steps in the analysis are given in the scripts [also shown in supplementary data 1 (Matlab) and

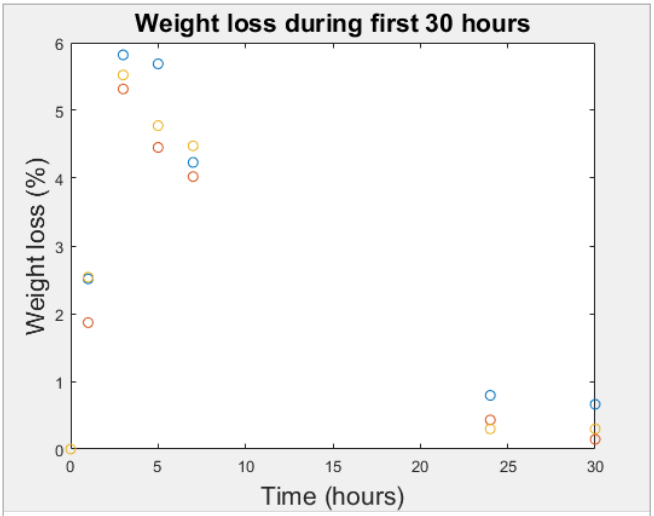

Figure 5. Plot of percentage weight loss over time for three replicate cement samples.

supplementary data 2 (R)]. For ease of use, the dataset, script and functions should be saved in the same folder. RStudio version 0.99.473 and Matlab version R2014b (including the Statistics and Machine Learning Toolboxes) were used to write all scripts and functions. It is important to note that all scripts are written sequentially (following the structure of this tutorial) so that each line depends on previous lines being executed. All scripts were run on a 3.30GHz Intel Core i5-4590 CPU with 8GB of RAM. The operating system used was 64 bit Windows 8.1 Pro.

The study was conducted using three replicate cement samples. Each sample was imaged at nine different time points and the hyperspectral acquisitions acquired over a period of 18 days. Sample weight loss (WL) was monitored by measuring sample mass at each time point. WL was found to be non-linear, likely due to changes in humidity and temperature. This is shown in Figure 5, where it can be seen that there was original WL followed by a subsequent weight gain over the first 30 hours for all three replicate samples.

The resulting dataset therefore contains 27 hypercubes in total. As each hypercube is a separate file, file management can quickly become unwieldy, possibly leading to delays due to mix-ups or even errors in analysis. It is therefore recommended that data structures (in the case of Matlab) or lists (in the case of R) are used to manage the data. Structures and lists are equivalent data types which allow saving of multiple variables within a single variable. In the present case, all 27 hypercubes were saved as a single variable, with a complementary character string reciting the file name. Further, the use of structures or lists also means that if new variables specific to each hypercube are created during the course of the analysis, they can be stored in fields within the structure or list visually aligned with the specific hypercube. This again reduces the likelihood of errors.

Finally, loops and vectorised functions can be performed on files within a structure or list in a sequential fashion, meaning that instead of needing to write a specific command for each hypercube, general commands can be written which will be applied to all hypercubes in the same manner. This leads to cleaner and more concise scripts and is convenient in time series analysis. Excerpts of the relevant script are provided in tables in some instances to explain new concepts in terms of programming syntax. However, where syntax is not seen as an issue, the concepts are simply discussed with reference to relevant lines of the script in the supplementary data.

Table 1 shows equivalent commands in both Matlab and R (providing relevant lines of supplementary data 1 and 2, respectively). In sequential order, the commands of Table 1:

- Set the current directory/working directory to the folder containing the dataset M7H16 and all relevant functions. Notice that R requires the file path to be entered with forward slashes which breaks standard file path convention.
- Load a ".mat" data file comprising the hypercubes. In the case of R, the data file is assigned to a variable called M7H16, whereas Matlab uses the title already saved within the loaded file, this is M7H16.
- Load an additional text file with data relating to the WL of the cement during setting. When loaded in this way, R requires a conversion of the data to a matrix data type.
- Save two one-dimensional arrays, known as vectors. The first vector is named "wvgood" and is a series of values from 880 to 1720, including every seventh unit, representing the wavelengths measured, i.e. 880 nm, 887 nm, … , 1720 nm. The second vector saves the time point.

Table 1. Loading initial time series hyperspectral data in Matlab and R.

Matlab	R
(3) cd("C:\...")	(2) setwd("C:/...")
(4) load M7H16	(12) M7H16=readMat("M7H16.mat")
(5) Wts=load("WL.txt");	(13) Wts=read.table("WL.txt", header=F, sep="\t")
	(14) Wts=as.matrix(Wts)
(8) wvgood=(880:7:1720);	(17) wvgood=seq(from=880, to=1720, by=7)
(9) Time=[0,0,0,1,1,1,3,3,3,5,5,5,7,7,7,24,24,24,30,30,30,17*24, 17*24,17*24,18*24,18*24,18*24];	(18) Time=c(0,0,0,1,1,1,3,3,3,5,5,5,7,7,7,24,24,24,30,30,30,17*24, 17*24,17*24, 18*24,18*24,18*24)

Table 2. Installation of R.matlab package in R.

R
(5) install.packages("R.matlab"); require(R.matlab)

In supplementary data 2, further libraries are loaded into R (pls, signal and chemometrics) which include additional functions to be used later in the program.

In both cases the same large ".mat" data file is loaded; however, a specific package must be installed in R ("R.matlab") in order to allow the loading of this format. This is shown in Table 2. The first function installs the package and the second includes it in the R library, meaning that the function which loads the ".mat" data file will be recognised by R.

The structure variable loaded in Matlab comprises two fields, one for the name of each hypercube and one for the hypercube itself. The field comprising the 27 hypercube names is called "M7H16" and the field comprising the 27 hypercube files is called "file". This is also the ideal layout to be used in R. The names and files can be accessed as shown in Table 3, where the examples show how to access the fifth name of the structure/list and how to access data from the fifth hypercube (rows 10 to 50, all columns and the first wavelength).

The ".mat" format is proprietary to Matlab and is therefore loaded in this suitable manner within Matlab. However, when the ".mat" file is initially loaded in R, the list comprises 54 variables in a single field with the file names (27 elements) and hypercube files (27 elements) alternating. We have written a small routine (Table 4) to reorganise the list so that it corresponds to the same data structure in Matlab. The commands of Table 4 are therefore run to create a new field (lines 21 to 23) within the list called "file" and delete the files from the "name" list (lines 25 to 27). These functions are performed using "for" loops, which allow repetitive functions to be performed without the requirement of explicitly typing the necessary function for each file. The first loop is designed in such a manner that a variable "*i*" is initiated to be equal to 1. The commands within the chain brackets will be run repeatedly and each time the loop restarts, the value of the variable "*i*" will increase by 1. The loop (1) is designed to end when "$i = 27$". In the first loop (1), the name field of the M7H16 list will be accessed at the $2 \times i$ position. During the first loop, $i = 1$ and $2 \times i = 2$. In the second loop $i = 2$, and therefore $2 \times i = 4$ etc. The for loop accesses every second entry of the field "M7H16" (the hypercube files) and assigns it to position "*i*" of a new field named "file" within the M7H16 list. The second loop (2) deletes the hypercube files of the "M7H16" field of the M7H16 list. The second loop (2) counts in reverse order, from 27 to 1, because if the loop counted normally, the positions of the latter entries of the list would have changed position by the time they were reached by the loop. For example, if entry two of the field were deleted, entry three would become the new entry two, entry four would be the new entry three etc. If done in reverse order, the position being deleted is more easily defined.

Table 4. Routine for formatting list of hypercube data in R.

R
(21) for (i in 1:27) {
(22) M7H16$file[i]=M7H16$M7H16[2*i];
(23) }
(25) for (i in 27:1) {
(26) M7H16$name[2*i]=NULL;
(27) }

Removing unwanted data from hypercubes

Removing noisy spectral data

While the NIR hyperspectral imaging system used can obtain spectra over an NIR range of 880 nm to 1720 nm, the readings at both extremes of this range are very noisy due to the decreased sensitivity of the InGaAs detector in these regions. This is a common issue in NIR hyperspectral imaging systems. Deletion of the noisy ranges is required in order to overcome the errors that this noise would introduce to the data analysis.

In order to proceed, it is useful to first view some images from the time series dataset. This is done with the "imshow" function in Matlab (line 31, supplementary data 1), and the "image" function in R (line 50, supplementary data 2). These functions require an input of a two-dimensional matrix to show an image, where the relative value of each element is represented by a shade between black (low value of reflectance) and white (high value of reflectance).

While it is not possible to display all of the spectral information of the hypercube in a single image, a two-dimensional matrix within a hypercube comprising NIR reflectance intensity values at a particular wavelength is viewable; alternatively up to three wavelengths can be combined in a false-colour RGB image.

For illustrative purposes, only 5 images of the 27 hypercubes were generated using a loop, and are shown in Figure 6 (lines 30 to 33 of supplementary data 1 and lines 46 to 52 of supplementary data 2). The images show the spatial arrange-

Table 3. Structure and list reference in Matlab and R, respectively, showing how to access filename and data fields.

	Matlab	R
Filename	M7H16(5).M7H16	M7H16$M7H16[[5]]
Data	M7H16(5).file(10:50,:,1)	M7H16$file[[5]] [10:50,,1]

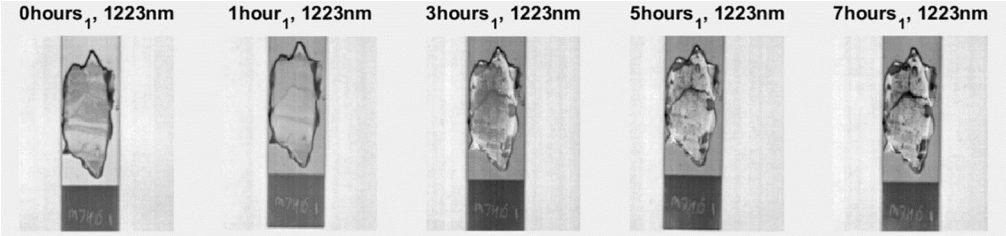

Figure 6. Images of samples at 1223 nm wavelength slice of hypercube (subscript indicates sample number).

ment of the relative intensities reflected at 1223 nm. It can be seen that changes are occurring in the cement over time. Most notably the surface becomes less homogeneous, as a crack is very clearly formed after five hours.

Using the information in the images of Figure 6, one small sample area (foreground) and three small background areas were selected as shown in Figure 7. The background regions covered (1) the glass slide upon which the cements were set, (2) the frosted region of a glass slide and (3) a region of a white tile upon which the glass slides were placed. The sample and background coordinates chosen were applied to identical row–column indices of all images. The spectra for these regions were then viewed to find an initial insight into the noisy spectral ranges (lines 35 to 47 of supplementary data 1 and lines 54 to 74 of supplementary data 2). Figure 8 shows the corresponding spectra for all the pixels of the regions of Figure 7. The sample spectra are presented in red while all background spectra are presented in blue.

Visually, it can be seen in Figure 8(a) that the noisy region is outside the 960 nm to 1680 nm range. These extremes were therefore cut from the spectra of all hypercubes and also from the saved wavelength range (lines 49 to 52 of supplementary data 1 and lines 76 to 79 of supplementary data 2).

Image background

As discussed above, most hypercubes contain a lot of redundant data in the form of image background. Since these data are usually irrelevant to a study, one of the first steps in preprocessing is their removal from the dataset. As previously mentioned, foresight during sample presentation makes subsequent tasks, such as separating the foreground from the background, more straightforward.

However, despite the recommendations provided in the introduction, it is often possible, due to sample related limitations, that the background in a set of images is not homogeneous or spectrally flat. This tutorial provides such a scenario in the example dataset as a challenge case.

The question of how to remove the background can initially be addressed by inspection of the background spectra as compared to that of the sample. This will give an indication of which wavelengths provide the greatest contrast between sample and background. A simple method for distinguishing between background and sample pixels would be to look for an intensity threshold which effectively separates sample data from the background. For example, it appears from the spectra of Figure 8 that finding a threshold value at around 1450 nm could be a good approach as the sample and background spectra separate well at these points. This is a simple, and often effective, method. However, initial attempts using this method ran into problems, with some foreground pixels being misclassified as background pixels. Therefore a more robust method was developed, as described below.

The developed method searches for an intensity ratio between two wavelengths of the sample spectra that effectively separates the foreground from the background. This method does not rely on the absolute measured value of a particular wavelength intensity, which can change dramatically at different sample heights (and time points in a time series), but rather looks at the relationship between two wavelengths. While changes in sample height have a great effect on the absolute intensity values, it has a far smaller effect on the overall shape of a spectrum.

In order to carry out this technique, the foreground and background spatial regions shown in Figure 7 were again utilised.

The selected foreground and background regions of each hypercube were first unfolded, resulting in two-dimensional matrices as described in the introduction. The background

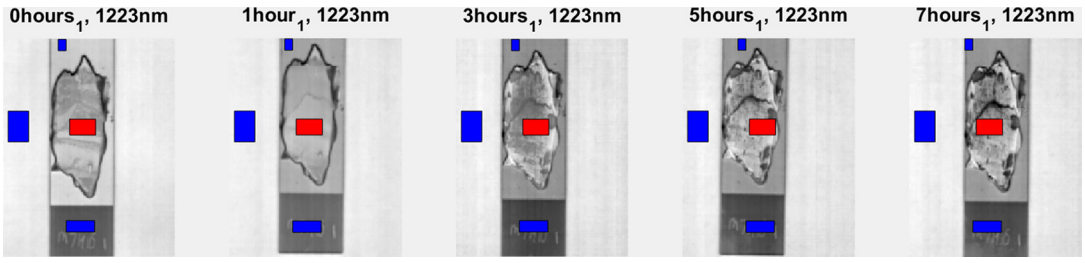

Figure 7. Images of Figure 6 showing sample region and three background regions selected.

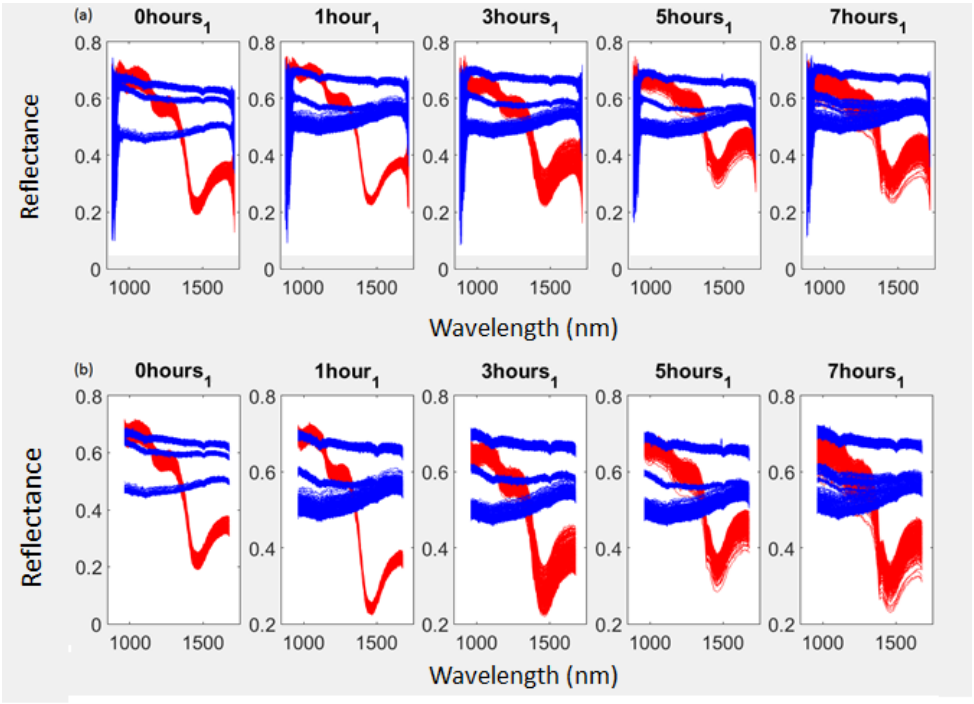

Figure 8. (a) Spectra of the regions selected in Figure 7 with sample spectra in red and background spectra in blue. (b) Spectra of (a) after the noisy extreme regions have been cut.

spectra from each region were then concatenated (i.e. stacked, one on top of the next) resulting in a single matrix for all background spectra (unfolding and concatenation: lines 70 to 80 in supplementary data 1 and lines 107 to 141 in supplementary data 2). Concatenation is the binding of multiple matrices resulting in one single matrix. In this instance the matrices are bound together by stacking one on top of the other. The number of columns must therefore be the same for all matrices which are to be bound. This is true in the present case as the columns refer to the wavelengths. A schematic depicting unfolding of foreground and background and subsequent concatenation of the background regions' unfolded matrices is shown in Figure 9.

Further to the concatenation of spectra of the different background regions for each individual hypercube as shown in Figure 9, all hypercube foreground spectra were concatenated together in a similar fashion.

Once unfolded and concatenated appropriately, the following steps where completed (lines 82 to 106 in supplementary data 1 and lines 143 to 169 in supplementary data 2). For each spectrum of the sample, the ratios of reflectance values for all wavelengths to all other wavelengths were calculated. In the present case, 103 wavelengths were considered. Therefore, for each spectrum, a matrix of size 103 × 103 was formed to store all ratios.

As thousands of spectra were considered (191 × 151 = 28,841), thousands of 103 × 103 matrices were thus created. In this instance, in order to separate the sample pixels from the background pixels, the general wavelength intensity ratios of the samples are important, but each small variation of wavelength ratios between sample pixels is not (for a homogeneous sample). For this reason, the mean value (averaged across spectra) of each element was calculated within the newly formed matrices. This resulted in a single average ratio matrix.

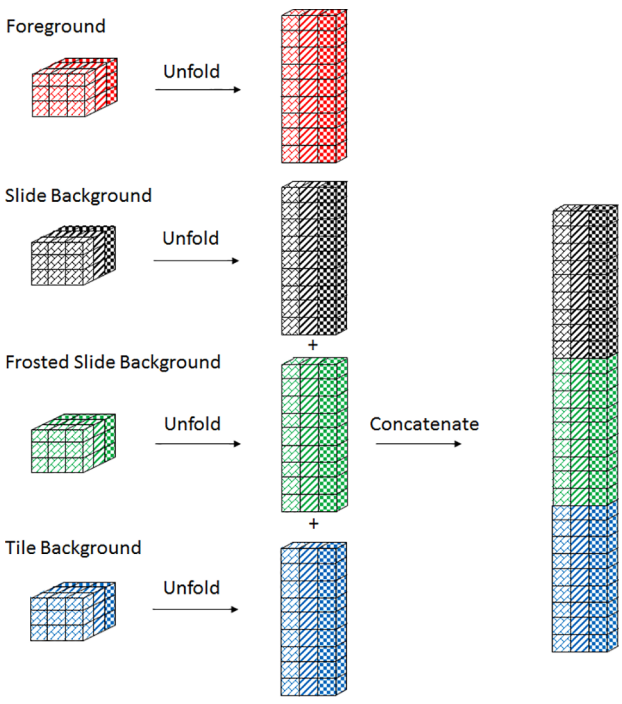

Figure 9. Schematic of unfolding and concatenation.

While the mean is a good representation of the relative ratios of intensities at different wavelengths, it is highly improbable that, for any given spectrum, the actual ratio will fall precisely on the mean. A range must be defined instead. The range used in the present example is ±2 × standard deviation from the mean, which provides a 95.44% confidence for a normally distributed sample. The present example does not follow a normal distribution; however, the final stage of the separation of sample pixels from background pixels, which is the visual assessment of the model, yielded positive results using this metric.

The above method was also completed for the background spectra.

For all wavelength ratios, the difference between the closest extremes of the "mean ± 2 × standard deviation" ranges of the foreground and background was calculated. This resulted in a 103 × 103 matrix, where the indices of the element with the greatest value indicate wavelengths with maximum difference.

The matrix produced is shown as an image in Figure 10 (lines 109 to 112 of supplementary data 1 and lines 171 to 174 of supplementary data 2).

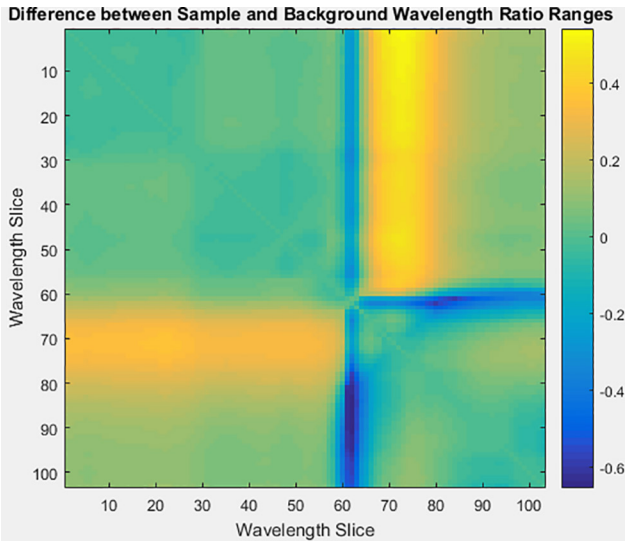

Figure 10. Matrix image showing difference between sample and background ratio ranges. Brighter colours indicate greater differences.

The indices of the brightest pixel of Figure 10 indicate the wavelength ratios with the greatest difference between sample and background. Attempts can be made to solve this visually; however, as this may not lead to an optimal result, a function to find the maximum value was used (lines 114 to 116 of supplementary data 1 and line 176 of supplementary data 2).

When the wavelengths whose average ratio resulted in the greatest separation between the foreground and background were found, a threshold ratio value was sought. The threshold was chosen to be the halfway point between the wavelength intensity ratio ranges.

This is illustrated in Figure 11 with a histogram of ratio values for background and foreground pixels combined from each hypercube (lines 118 to 130 of supplementary data 1 and lines 178 to 191 of supplementary data 2). Figure 11 clearly shows two distinct regions, one relating to the background and one relating to the foreground. Lines have been included to show the mean values of both the foreground and background and further lines have been included to show the relevant 2 × standard deviation range limits. Finally, a line is included to show the chosen threshold value, the halfway point between the relevant 2 × standard deviation range limits. Clearly the data are not normally distributed; however, this does not affect separation of the foreground from the background using the proposed method, indicating its robustness.

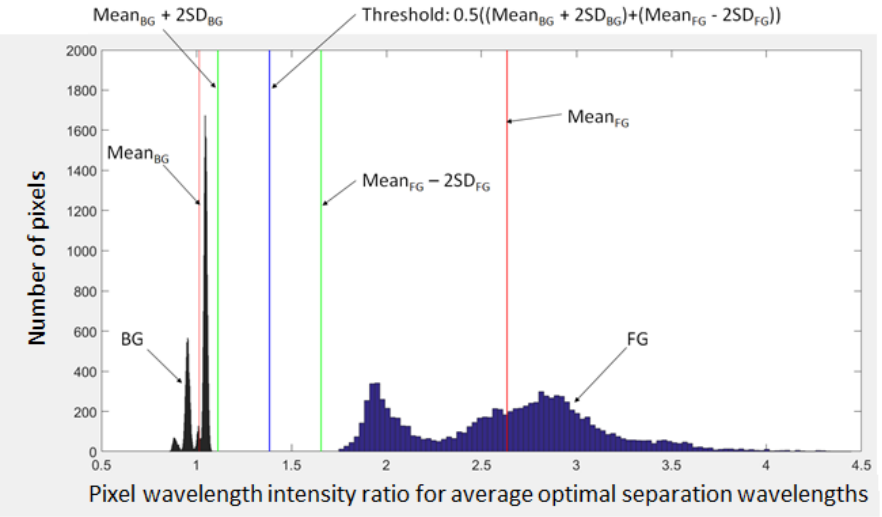

Figure 11. Histogram of foreground and background values, including bars showing foreground and background means, 2 × standard deviation markers and the central position between the 2 × standard deviation markers (BG, background; FG, foreground; SD, standard deviation).

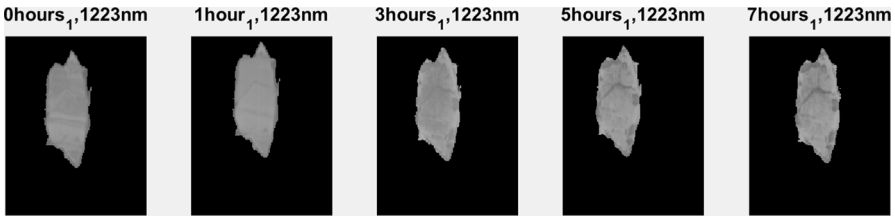

Figure 12. Images of Figure 6 after masking.

Ultimately the purpose of separation between the sample and the background is to conduct analysis on the sample pixels only. As a robust method to distinguish the pixels believed to relate to the sample and those believed to relate to the background has now been developed, the information can be used to create a "mask" (lines 132 to 143 of supplementary data 1 and lines 193 to 211 of supplementary data 2). The mask will be a matrix of the same size as the rows and columns of each hypercube and will comprise logical values of ones and zeros only. The hypercubes are then "masked", such that where an element of the mask has a value of 1, the equivalent element of the hypercube is a foreground sample pixel and the readings spectra are maintained. Conversely, where an element of the mask has a value of 0, the equivalent element of the hypercube is considered a background pixel and all spectra information is set to a value of zero.

The masked images at 1223 nm are shown in Figure 12 (lines 145 to 148 of supplementary data 1 and lines 213 to 217 of supplementary data 2).

Removing dead pixels and spikes

Dead pixels and spikes (also known more generically as "bad pixels") are a common occurrence in NIR hyperspectral imaging,[7,8] where typically up to 1% of pixels will be erroneous. Single isolated defective pixels located in random spatial positions can generate black pixels, called "dead pixels", and false intensity peaks known as "spikes".

Many NIR cameras come with pre-programmed functions that identify dead pixels and compensate for them by interpolation. However, dead pixels tend to increase in number over time, so it is important to check for them within each hypercube prior to data processing. A normal spectrum of a cement sample is shown in Figure 13(a) (lines 154 to 184 of supplementary data 1 and lines 224 to 246 of supplementary data 2). For demonstration purposes, Figure 13(a) also shows artificially altered spectra representing what a dead pixel or spike might look like. Detection of such pixels is not trivial. However, if a difference spectrum is generated (lines 186 to 200 of supplementary data 1 and lines 247 to 256 of supplementary data 2), obtained by subtracting the intensity at each wavelength from that of the preceding wavelength, as shown in Figure 13(b), it can be seen that for normal spectra, the difference spectrum is quite flat due to the smoothly varying nature of NIR spectra in general. Conversely, the difference spectrum of spectra with the spike and dead pixel presents a sharp discontinuity. This information is captured neatly by the standard deviation of each difference spectrum, with the standard deviation of the dead or spiked pixels being far larger than that of those which are functional.

These observations can be used to make a simple routine for identifying bad pixels in a hypercube. Difference matrices and their standard deviations were calculated (lines 202 to 212 of supplementary data 1 and lines 258 to 268 of supplementary data 2). Pixels with high standard deviation values are good candidates for dead pixels or spikes and thresholding can be used to identify these pixels. Histograms were produced to decide where the threshold for the standard deviation should be drawn (lines 214 to 227 of supplementary data 1 and lines 270 to 277 of supplementary data 2). This is shown in Figure 14, and the value of 0.0135 was chosen visually. The standard deviation image for a selection of hypercubes, as shown in Figure 15(a), gives the spatial distribution of the standard deviation of the difference spectrum of each pixel (lines 234 to 240 of supplementary data 1 and lines 279 to 291 of supplementary data 2).

Applying this simple thresholding method to a selection of hypercubes from our example time series dataset results in Figure 15(b), where outlier pixels are evident as bright points in the image (lines 242 to 248 of supplementary data 1 and lines 294 to 300 of supplementary data 2). Thresholding the standard deviation image results in Figure 15(b), which shows the location of the dead or spiked pixels. When the spectra corresponding to these pixels are examined as shown in Figure 15(c), it is clear that bad pixels are present (lines 250 to 262 of supplementary data 1 and lines 302 to 314 of supplementary data 2). It is possible to replace pixels like these by interpolating from surrounding pixels. However, in this case they were not included in the analysis, since there were relatively few in number.

An inverse of the spike mask (the non-spike pixels) was applied to the background-masked hypercubes using element-by-element matrix multiplication. As the spike mask is composed of zeros and ones, any element with a spike will be multiplied by zero, masking that pixel as shown in Figure 15(d) (lines 267 to 277 of supplementary data 1 and lines 316 to 327 of supplementary data 2).

Spectral pre-treatments

Spectral pre-treatments are commonly employed in NIR chemical imaging and spectroscopy to overcome variations in spectra caused by physical effects (e.g. light scattering,

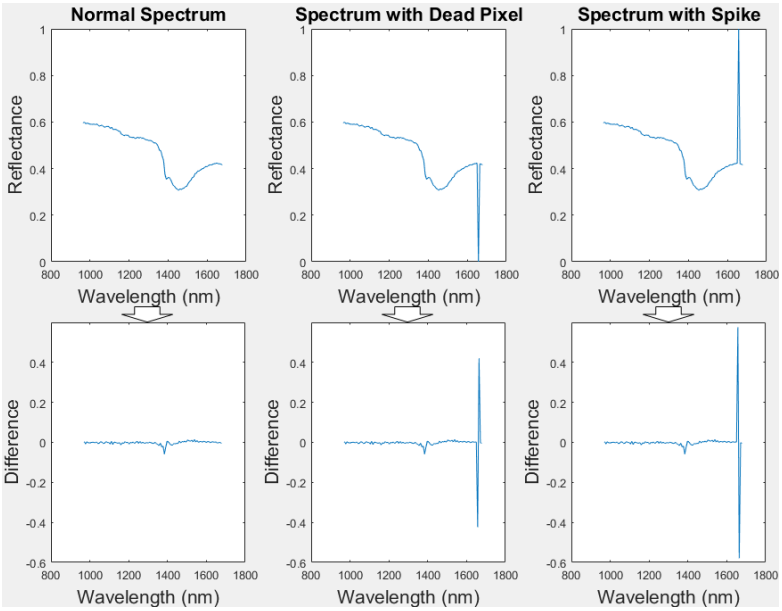

Figure 13. (a) Normal, dead pixel and spike reflectance spectra and (b) their difference spectra.

variable focal length due to non-flat samples). A commonly used spectral pre-treatment is multiplicative scatter correction (MSC). It is not always straightforward to measure the effectiveness of a pre-treatment (although efforts have been made to this effect[9]), and for this reason a common practice is to try a variety of pre-treatments and compare them in terms of model performance, the pre-treatment resulting in the lowest error being optimal. Application of the MSC pre-treatment results in a clear decrease in the spectral variability as shown in Figure 16(b) (lines 279 to 310 of supplementary data 1 and lines 332 to 362 of supplementary data 2). Though not shown in the Figures, sample 2 at the 17 day time point was not spike masked correctly. This illustrates the difficulties that can often occur when spike masking and further that a one-method-fits-all approach is not always viable.

Multivariate analysis of time series data

Unsupervised analysis

Principal component analysis on individual hypercubes

Principal component analysis (PCA) is a versatile tool which can be used to summarise and explore the information contained in a hypercube. PCA has several uses. For example, it can be used to find variation and trends within a dataset which might not otherwise be obvious. It does this by comparing the spectra against each other and finding peaks which are more likely to be signals than noise. This even applies to small features, which appear in many spectra. PCA is performed on two-dimensional matrices and in the present case is therefore performed on

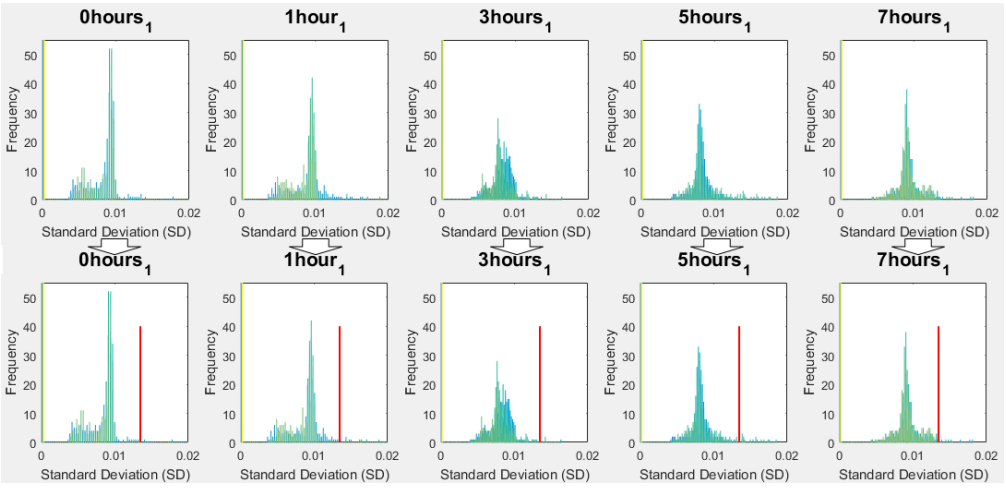

Figure 14. Histograms of spectral standard deviations (a) excluding and (b) including a threshold bar.

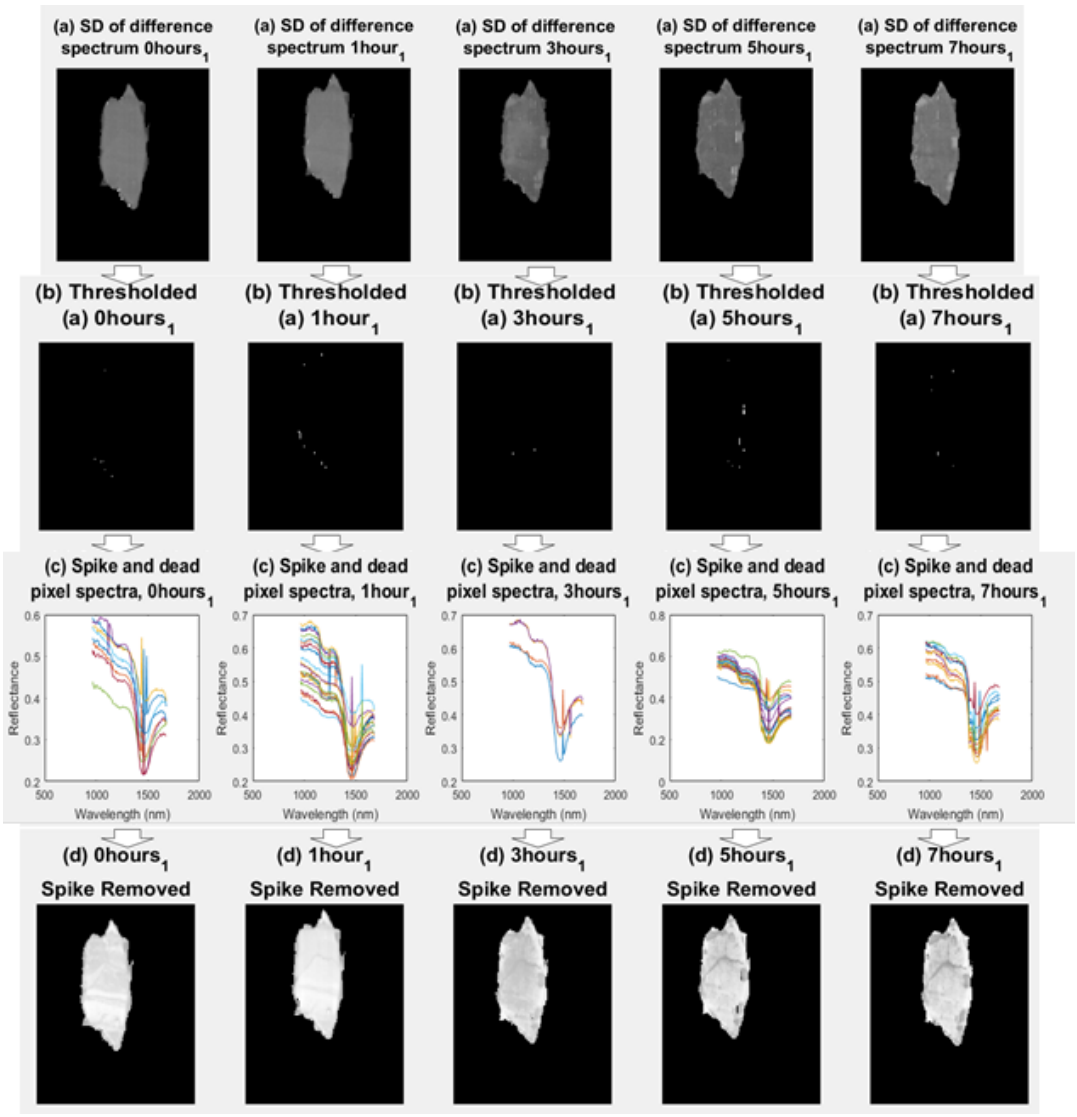

Figure 15. Identification of dead pixels. (a) Standard deviation (SD) of difference hypercube; (b) thresholding locates the dead pixels; (c) spectra corresponding to dead pixels; (d) images with spike removed.

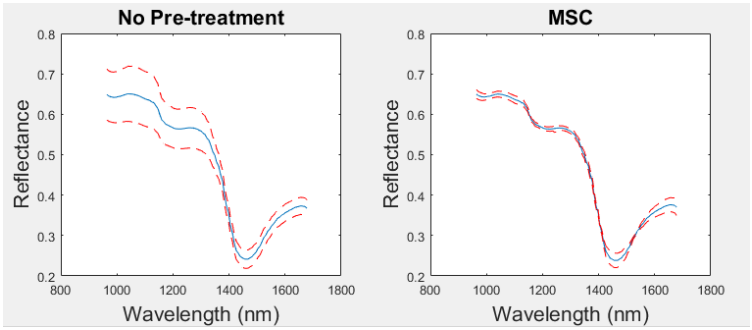

Figure 16. Effect of pre-treatments on mean (solid curves) and mean ± standard deviation (dashed curves) spectra: (a) no pre-treatment; (b) MSC.

unfolded hypercubes. PCA reconstructs the information of the unfolded hypercube into components known as the principal components (PCs) and loadings, which, when multiplied together, reproduce the original dataset. Each loading is a vector which provides information on the relative importance, or the weighting, of specific wavelengths relative to each other.

The ordering of the loadings also provides information. The first loading describes the largest variance in the dataset and each following loading describes progressively less of the variance of the dataset. Therefore, instead of using all loading vectors, we can opt to just use some of the earlier loading vectors to reproduce the original dataset. When multiplied by the mean-centred dataset, the majority of the variance is still described; however, the size of the data is greatly reduced. The data removed are called the residual, which, if done correctly, will result in the removal of noise from the dataset.

Figure 17 shows a schematic of the PCA process, where the non-zero rows of the unfolded, masked hypercube are first found (lines 312 to 337 of supplementary data 1 and lines 368 to 376 of supplementary data 2). Subsequently, the masked rows are removed. The remaining rows are mean-centred and PCA is performed producing PC scores and loadings. However, these PC scores are not further used as they relate only to the non-zero matrix. Some of the first PC loadings are chosen, in this instance the first 10, which are the 10 loadings which represent the greatest variance in the dataset. As seen in Figure 17(b), the unfolded, mean-centred hypercube is then projected along the PC loadings by matrix multiplication producing a PC scores matrix which is subsequently refolded. PCA is a commonly used statistical method and pre-programmed functions are available in both Matlab and R (the function is called 'princomp' in both). The steps involved in PCA are well documented in the literature.[10]

PC score images were obtained by applying PCA to the non-background pixel spectra of the individual hypercubes in the dataset (lines 322 to 337 of supplementary data 1 and lines 376 to 382 of supplementary data 2). When looking at different time points for each PC, it is important that the images are equally scaled. To find the ranges to use in the PC images, histograms of the PC data were plotted with bars marking the upper and lower limits. The approximate upper and lower limits were then found visually from the histograms of Figure 18 (lines 339 to 352 of supplementary data 1 and lines 411 to 428 of supplementary data 2). Figure 19 shows the first seven PC images (lines 354 to 401 of supplementary data 1 and lines 410 to 458 of supplementary data 2).

In the PC1 images of Figure 19, it can be seen that there are different amounts of variation across the surfaces of each sample, becoming more pronounced at later timepoints. PC4 appears to reveal variation caused by changing surface morphology. PCs 6 and 7 appear to be mostly noise with perhaps some variation introduced from different pixels of the detectors.

After the PC scores were produced, the mean score of each hypercube along each PC was calculated. These average score values were checked for correlation with the WL variable (lines 403 to 417 of supplementary data 1 and lines 398 to 407 of supplementary data 2).

Concatenated PCA

After removing the background, spikes and dead pixels, the resultant set of cleaned hypercubes can be analysed together as a block of time series data. This is a useful approach to evaluate trends in the entire dataset. A straightforward method of unsupervised analysis is to apply PCA to the entire set of hypercubes. This can be carried out in at least two different ways, as shown schematically in Figure 20. The first variation of PCA takes all of the sample spectra from each hypercube (omitting the background, dead pixels etc.), unfolds and then concatenates them to make a two-dimensional matrix (**X**), on which PCA is applied. The second variation of concatenated PCA calculates the mean spectrum of the sample spectra from each hypercube and compiles them into a matrix on which PCA is applied. This second variation is faster, as fewer spectra are analysed. Regardless of the methods used, after obtaining the PCA loadings, PC score images can be obtained in the same way as described in Figure 17. However, it is important to remember that in this case PCA is applied to the concatenated non-background spectra from each image. Therefore, when calculating scores for each separate hypercube, it is necessary to mean-centre the separate hypercubes according to the mean of the concatenated matrix. Similarly, when using a pre-treatment such as MSC (which requires a target spectrum), the target spectrum should be the mean of the concatenated matrix. As discussed above, after calculating the PC scores it is possible to calculate an average score value for each sample, and this can be compared to any measured variables that are available. This is useful for investigating whether the PC scores are correlated to measurements of interest (lines 419 to 473 of supplementary data 1 and lines 465 to 529 of supplementary data 2).

For comparison, Table 5 shows the correlation between mean PC score values and measured WL for the three variants of PCA applied as in Figure 20: applying PCA separately to each hypercube, to the concatenated pixel data and to the concatenated mean spectra. Straight away it is clear that the three variants of PCA are not equivalent. When PCA is applied to each image separately there is no correlation between WL and PC score values. When applied to concatenated data, there is a stronger correlation between WL and PC1. Interestingly, the application of MSC pre-treatment decreased the correlation between WL and PC1, indicating that scattering information is related to WL. The resultant PC1 score images for raw (non-pre-treated) and MSC pre-treated data are shown in Figure 21 (lines 518 to 532 of supplementary data 1 and lines 501 to 514 of supplementary data 2). It is clear that some variations due to morphology at the sample edges are reduced after applying MSC, but the trend with WL is also diminished.

Supervised analysis: regression using partial least squares (PLS) regression

Calibration methods developed for standard two-dimensional matrices, where each row represents a sample, can be applied

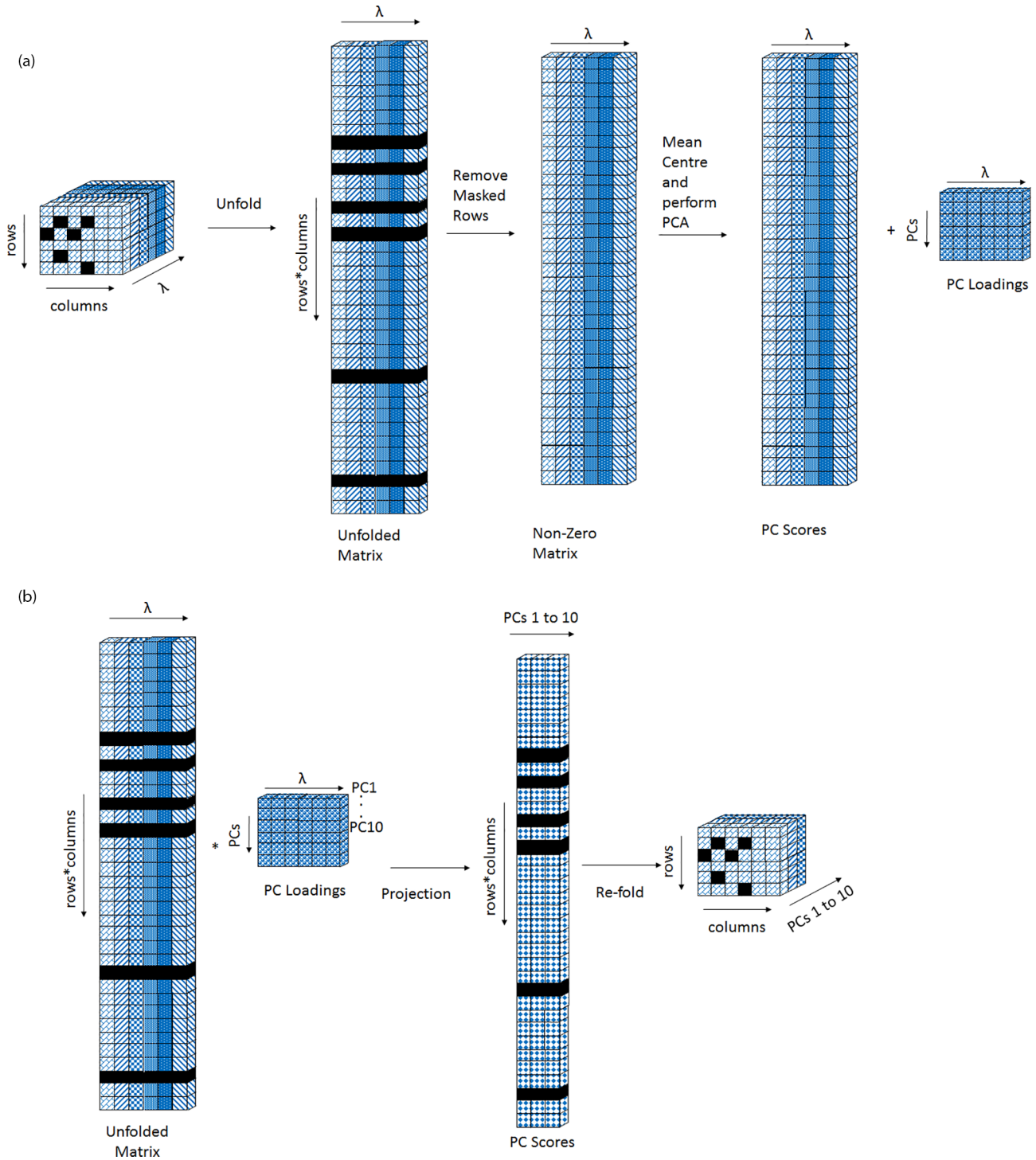

Figure 17. Application of PCA to a single hypercube. (a) Unfolding, then removal of background, extracting sample spectra from hypercube, followed by mean centring and matrix decomposition by PCA application. (b) Projection of the first 10 PC loadings along the original masked hypercube to produce PC scores matrix. The PC scores matrix is refolded to make a PC scores hypercube.

to hyperspectral data in a straightforward manner by obtaining the mean spectrum of each sample, as shown schematically in Figure 22(ii). An important point to note is that all calibration models should be validated with independent datasets. In the present case, of the three samples analysed, the first sample at each of the nine time points was used for model calibration, the second sample at each time point was used for model training and the third was used for model validation (lines 542 to 548 of supplementary data 1 and line 628 and 634 of supplementary data 2). PLS was then performed (lines 550 to 551 of supplementary data 1 and lines 642 and 643 of supplementary data 2).

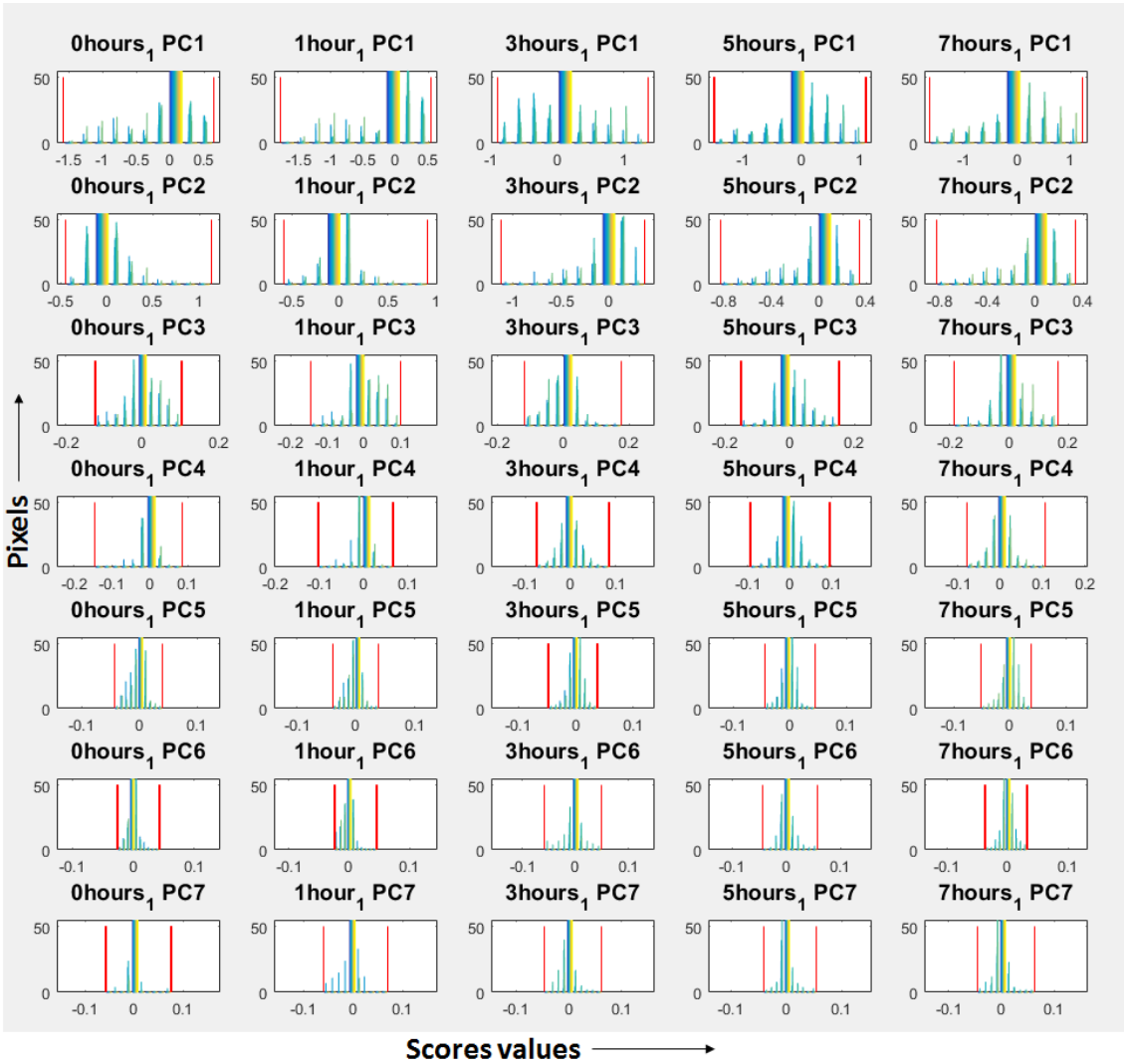

Figure 18. Histograms of PC1 to PC7.

The number of latent variables (LVs) to include in the PLS models developed was initially decided by selecting the minimum root mean squared error (*RMSE*) of the calibration model applied to the training set. Using this method, four LVs were selected for the raw data and three for the MSC pre-treated data. Model performance metrics (shown in Table 6) can be used to compare the effectiveness of the pre-treatments. Functions (PLS_output.R and PLS_output.m) were written to calculate key model performance metrics, inspired by Dardenne.[11]

In order to produce PLS prediction maps, the first seven LVs produced by PLS were multiplied by the unfolded pre-processed hypercubes. For comparison, prediction maps for the three LVs raw and MSC pre-treated data models are shown in Figure 23 (lines 581 to 591 of supplementary data 1 and lines 702 to 718 of supplementary data 2). The intensity values in the prediction maps are scaled according to the minimum and maximum values of the measured variable, to enable fair comparison.

Leave-one-out cross validation is a method commonly used to select the optimal number of LVs in a regression model. However, this method is prone to overfitting—when using leave-one-out cross validation and basing selection of LVs on the minimum root mean square error of cross validation, higher numbers of LVs (>7) were selected for this dataset, yet predictive performance was poorer, indicating overfitting. Another way to identify overfitting in PLS regression models applied to hyperspectral data is by inspection of prediction maps. Overfitted models tend to produce prediction maps with the appearance of noise. This can be quantified by calculating a root mean square error of prediction (*RMSEP*) for each hypercube ("RMSEPIM"), a method recently developed by Gowen *et al.*[6] (lines 555 to 562 of supplementary data 1 and lines 721 to 737 of supplementary data 2). For visual comparison, the prediction maps for 1 to 7 LV models for the raw data are shown in Figure 24 (lines 565 to 572 of supplementary data 1 and lines 679 to 688 of supplementary data 2). It is clear that the model is underfitted at 1 and 2 LVs, since a large number

Figure 19. PC1 to PC7 score images for sample 1 at 0, 1, 3, 5 and 7 hours, where PCA was performed on individual hypercubes.

of pixels are predicted outside of the range of WL. Evidence of overfitting starts to occur after 4 or 5 LVs. The pooled *RMSEP* calculated from the images (RMSEPIM) is plotted in Figure 25 (lines 593 to 602 of supplementary data 1 and lines 740 to 742 of supplementary data 2) and a local minimum at 3 LVs indicates this is the optimal number of LVs to include in the models. In previous applications of this method a global minimum was found in RMSEPIM—clearly, in our data, this minimum occurs at 1 LV, which is an underfitted model according to the prediction maps. The *RMSEP*, R^2P, *BIAS*, *SEP* and *RPDP* were also calculated for 1 LV and these results were evaluated against the 3 LV result for comparison. For 1 LV: *RMSEP* = 2.58, R^2P = 0.12, *BIAS* = 0.48, *SEP* = 2.53, *RPDP* = 1.09. It can be seen that the *RMSEP* and SEP errors are both considerably higher

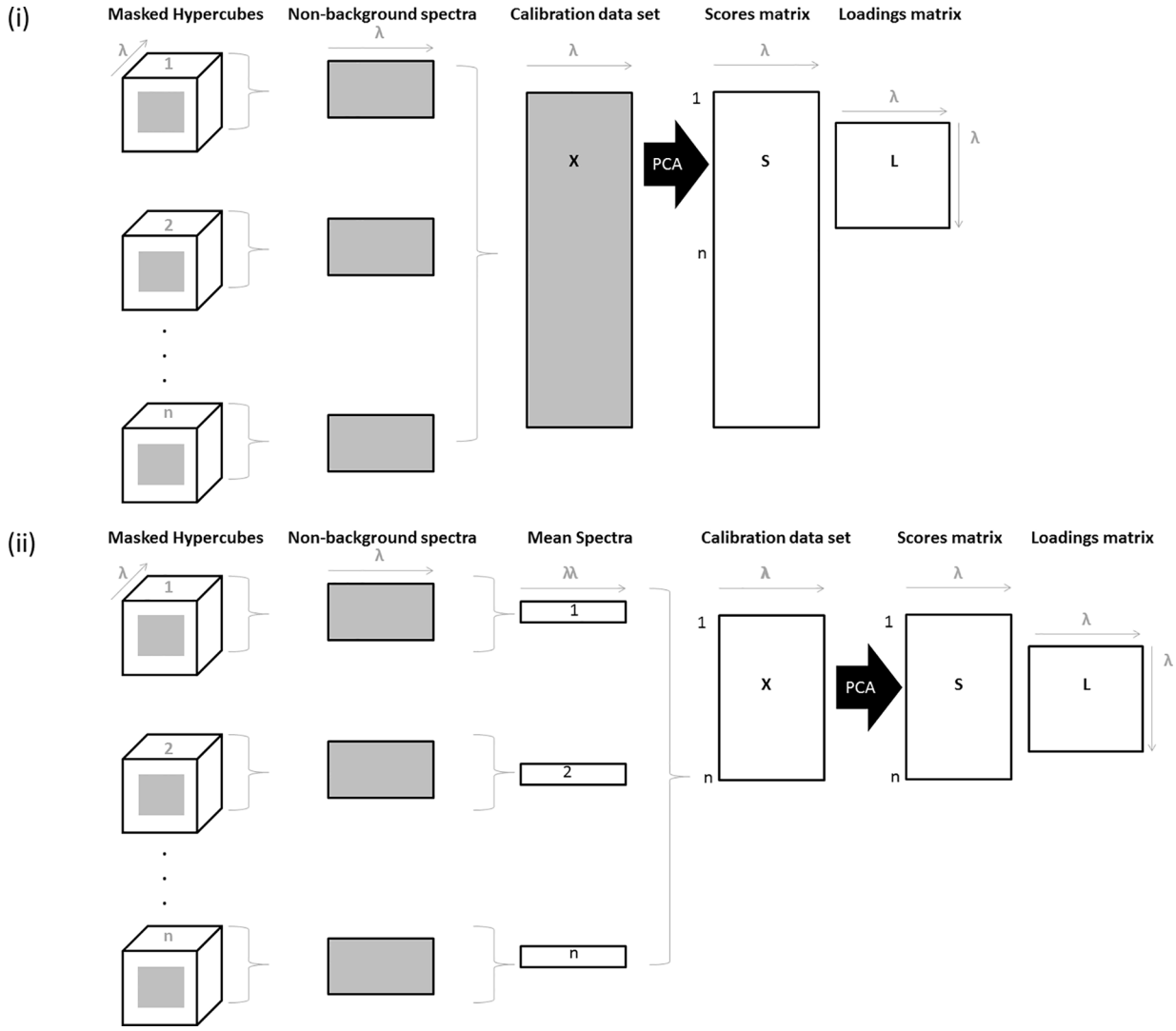

Figure 20. Application of PCA to a group of time series hypercubes: (i) the application of PCA to pixel spectra extracted from each hypercube and then concatenated; (ii) the application of PCA to mean spectra extracted from each hypercube and then concatenated.

Table 5. Correlation between mean PC score values and measured weight loss, using two variants of PCA [PCA 1 and 2 as described in Figure 20(i) and (ii), respectively], where MSC = multiplicative scatter correction.

Pre-treatment	PCA on each hypercube separately	PCA on concatenated pixel spectra		PCA on concatenated mean spectra	
	Raw	Raw	MSC	Raw	MSC
PC1	−0.09	0.84	0.64	0.83	0.64
PC2	−0.21	0.10	0.52	0.39	0.27
PC3	−0.04	0.56	0.03	0.21	−0.14
PC4	−0.05	0.11	−0.36	0.08	−0.57
PC5	−0.16	−0.28	−0.51	−0.03	0.10
PC6	0.1	−0.30	−0.03	−0.08	−0.27
PC7	0.28	−0.02	0.46	−0.13	−0.17
PC8	−0.02	−0.07	−0.32	−0.22	−0.15
PC9	0.13	0.60	−0.62	−0.07	0.05
PC10	0.1	0.51	−0.60	0.13	−0.11

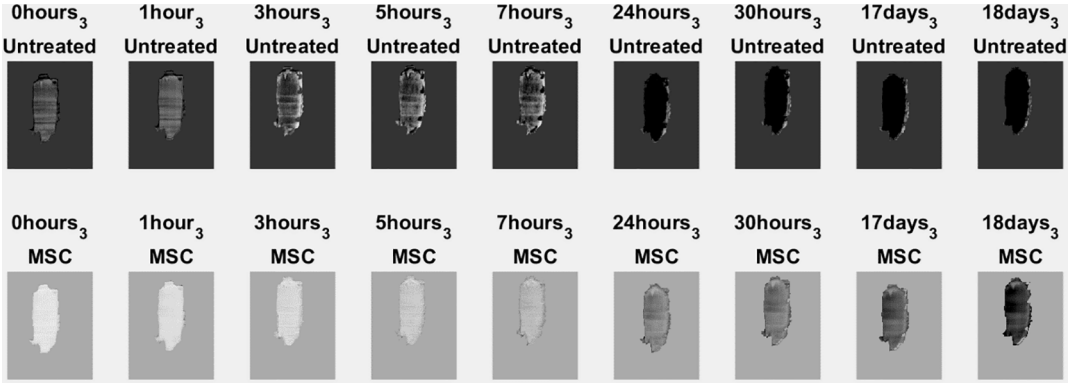

Figure 21. PC1 score images corresponding to concatenated PCA applied to raw and MSC pre-treated data.

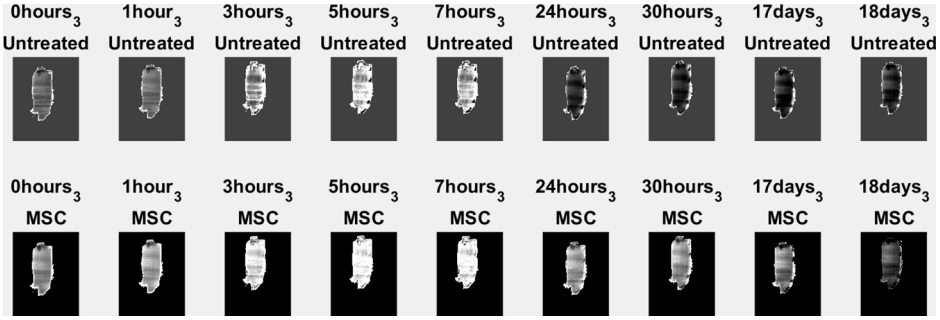

Figure 22. Prediction maps for raw 3 LV (top) and MSC 3 LV (bottom) models.

Table 6. PLS model performance metrics for prediction of weight loss from mean hypercube spectra[a].

Pre-treatment	*nLV*	*RMSECV*	*R^2CV*	*BIASCV*	*SECV*	*RPDCV*	*RPDCV*	*R^2P*	*BIASP*	*SEP*	*RPDP*
None	4	1.18	0.81	0.54	1.04	2.59	1.86	0.54	1.33	1.22	2.26
MSC	3	1.42	0.72	0.33	1.38	1.95	2.44	0.21	2.00	1.21	2.28

[a]MSC, multiplicative scatter correction; *nLV*, number of latent variables; *RMSE*, root mean squared error; *SEP*, standard error; *RPD*, residual predictive deviation; CV, cross validation; P, prediction.

than the 3 LV result while the *R^2P*, *BIAS* and *RPDP* are lower, indicating a less effective prediction and a non-optimal model. Therefore visual observation of prediction maps in conjunction with the RMSEPIM metric is a useful method for evaluating model performance.

Summary and suggestions for hyperspectral image processing

This tutorial provides a framework for the manipulation of multiset time series NIR hyperspectral imaging data using the command line languages R and Matlab. Analysis of the example dataset presented should benefit readers by revealing the data manipulations commonly required to deal with large time series hyperspectral imaging datasets. Due to the flexibility afforded by this approach, once the reader is proficient in these basic methods, the complexity of analysis (e.g. application of specific data pre-tretments, non-linear regression models) can easily be extended. For example, alternative spike masking techniques can be developed to better mask all hypercubes in the sample data set or other data sets. We finish with some suggestions pertinent to hyperspectral image analysis in general:

1. Prior to commencing modelling, carefully examine the raw data and reduce the image size (if necessary) by removing noisy wavelength and background regions.
2. Pay attention to the identification and removal of spikes and dead pixels.
3. Evaluate different spectral pre-processing methods, choosing the one that is most suited to the objective of the study.
4. Be aware of the issue of overfitting. It is important to validate predictive models on new datasets rather than the same dataset. Efforts should be made to avoid unintentional

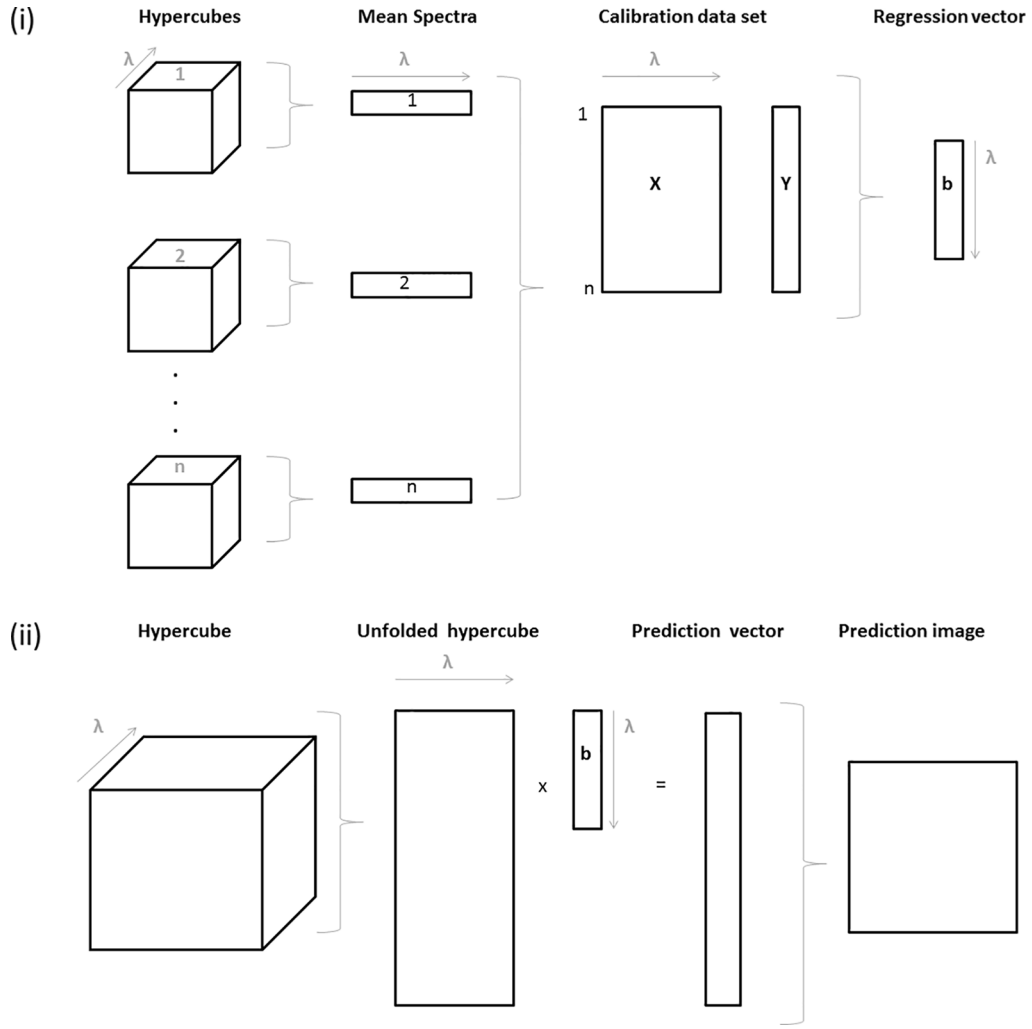

Figure 23. Application of regression modelling to hypercubes: (i) calculation of mean spectra from sample images and construction of regression vector b; (ii) application of regression model to hypercube, resulting in prediction image.

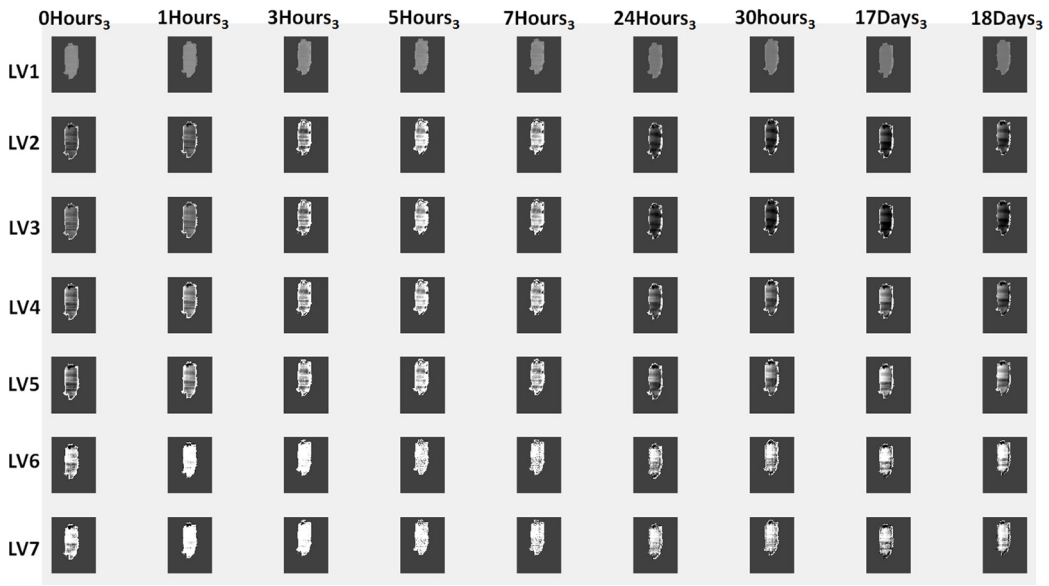

Figure 24. PLS prediction maps for 1 to 7 LV models for all measured time points of sample replicate 3.

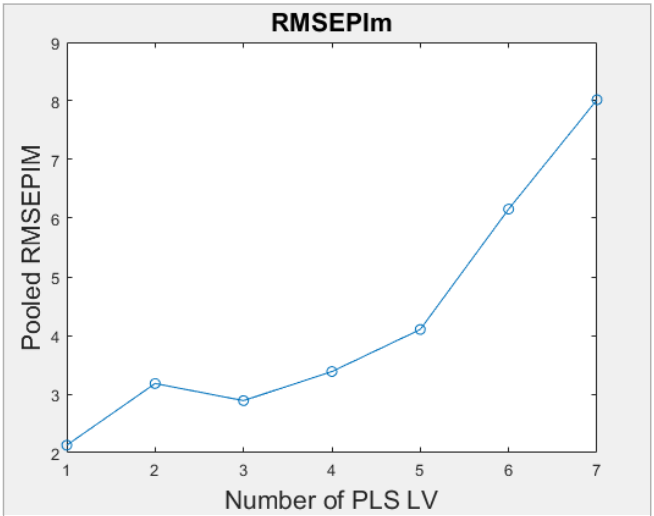

Figure 25. Pooled RMSEPIM for prediction of weight loss.

dependencies between the datasets used. For example, ideally the samples that are used to produce the data of the training set and the calibration set should be produced in different batches using the same raw materials from different sources.

5. When dealing with time series data, aim to examine the data from the entire dataset simultaneously (rather than individual images) for tasks such as threshold setting and trend analysis, using the methods provided in this paper.

Acknowledgement

The first and last authors acknowledge funding from the EU FP7 under the European Research Council Starting Grant programme.

Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1255/jnirs.1208.

References

1. J.M. Amigo, H. Babamoradi and S. Elcoroaristizabal, "Hyperspectral image analysis. A tutorial", *Anal. Chim. Acta* **896,** 34 (2015). doi: http://dx.doi.org/10.1016/j.aca.2015.09.030

2. H. Huang, L. Liu and M.O. Ngadi, "Recent developments in hyperspectral imaging for assessment of food quality and safety", *Sensors* **14,** 7248 (2014). doi: http://dx.doi.org/10.3390/s140407248

3. Y. Tan, Y. Liu, Z. Zhao, J.Z. Paxton and L.M. Grover, "Synthesis and *in vitro* degradation of a novel magnesium oxychloride cement", *J. Biomed. Mater. Res. A* **103,** 194 (2014). doi: http://dx.doi.org/10.1002/jbm.a.35166

4. Z. Li and C.K. Chau, "Influence of molar ratios on properties of magnesium oxychloride cement", *Cement Concr. Res.* **37,** 866 (2007). doi: http://dx.doi.org/10.1016/j.cemconres.2007.03.015

5. A. Gowen, F. Marini, C. Esquerre, C.P. O'Donnell, G. Downey and J. Burger, "Time series hyperspectral chemical imaging data: challenges, solutions and applications", *Anal. Chim. Acta* **705,** 272 (2011). doi: http://dx.doi.org/10.1016/j.aca.2011.06.031

6. A. Gowen, J. Burger, C. Esquerre, G. Downey and C.P. O'Donnell, "Near infrared hyperspectral image regression: on the use of prediction maps as a tool for detecting model overfitting", *J. Near Infrared Spectrosc.* **22,** 261 (2014). doi: http://dx.doi.org/10.1255/jnirs.1114

7. F. Firtha, A. Fekete, T. Kaszab, B. Gillay, M. Nogula-Nagy, Z. Kovács and D.B. Kantor, "Methods for improving image quality and reducing data load of NIR hyperspectral images", *Sensors* **8,** 3287 (2008). doi: http://dx.doi.org/10.3390/s8053287

8. L. Zhang and M.J. Henson, "A practical algorithm to remove cosmic spikes in Raman imaging data for pharmaceutical applications", *Appl. Spectrosc.* **61,** 1015 (2007). doi: http://dx.doi.org/10.1366/000370207781745847

9. C. Esquerre, A. Gowen, C.P. O'Donnell and G. Downey, "Suppressing sample morphology in near infrared spectral imaging of agriculture products by chemometric pre-treatments", *Chemometr. Intell. Lab. Syst.* **117,** 129 (2012). doi: http://dx.doi.org/10.1016/j.chemolab.2012.02.006

10. J. Burger and A. Gowen, "Data handling in hyperspectral image analysis", *Chemometr. Intell. Lab. Syst.* **108,** 13 (2011). doi: http://dx.doi.org/10.1016/j.chemolab.2011.04.001

11. P. Dardenne, "Some considerations about NIR spectroscopy: closing speech at NIR-2009", https://www.impublications.com/content/some-considerations-about-nir-spectroscopy (2009).