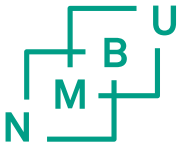

U
B
M
N

Use of regularizing methods in modeling and prediction of wood weathering as a response to UV radiation.

Knut Arne Smeland

December 2015

Abstract

Samples of norway spruce (*Picea abies*) were exposed to various amounts of natural or artificial UV radiation and recorded with a NIR hyperspectral camera. Multivariate statistical analysis using Tikhonov regularization, Principal Component Regression and Partial Least squares were applied to the resulting data. Results were mixed, but with consistently better models for the samples exposed to artificial radiation, able to predict exposure amounts with a reasonable degree of accuracy and generalizability. Tikhonov regularization performed better or on par with PCR or PLS in most cases, and showed a tendency to better predict new data. Samples exposed to natural weathering yielded significantly worse results, presumably related to the myriad of interfering, unquantified phenomena present in the environment. It was thus concluded that the weathering effect on wooden surfaces is well reflected in the NIR spectrum. NIR hyperspectral imaging holds promise as a useful tool in the further investigations into the subject, with the caveat that advancements need to be made in the statistical modeling of the data, to make it more robust and applicable outside of a purely research oriented environment.

Contents

1 Introduction **1**

2 Materials and equipment **2**

3 Methods **4**
3.1 Pre-processing 5
3.1.1 Segmentation 6
3.1.2 Smoothing 8
3.1.3 Scatter correction 9
3.2 Statistics 13
3.2.1 Least Squares Regression 14
3.2.2 Regularization 16
3.2.3 Partial Least Squares 22
3.2.4 Principal Component Regression 22

4 Results **22**
4.1 Ground wood samples 23
4.2 Weather exposed 23

4.3 UV chamber . . . 29
4.4 Lignin content . . . 31

5 Discussion **33**
5.1 Data preparation . . . 33
5.2 Analysis . . . 35

6 Conclusion **39**

Appendices **44**

A Results **44**
A.1 Weather-exposed samples . . . 44
A.1.1 Earlywood . . . 44
A.1.2 Earlywood, cleaned . . . 51
A.1.3 Latewood . . . 54
A.1.4 Latewood, cleaned . . . 57
A.1.5 Earlywood, validation . . . 60
A.1.6 Earlywood, validation, cleaned . . . 63
A.1.7 Latewood, validation . . . 65
A.1.8 Latewood, validation, cleaned . . . 68
A.2 Lab samples . . . 71
A.2.1 Earlywood . . . 71
A.2.2 Latewood . . . 74
A.2.3 Earlywood, validation . . . 77
A.2.4 Latewood, validation . . . 80
A.3 Ground wood samples . . . 83

B Preprocessing **85**

C Functions and scripts **88**
C.1 Import . . . 88
C.2 Masking . . . 92
C.3 Regression . . . 94
C.3.1 Ridge regression . . . 94
C.3.2 Tikhonov Regularization, derivatives operator . . . 96
C.3.3 PCR . . . 98
C.3.4 PLS . . . 98

List of Figures

1 Wood slice sample . . . 2
2 Illustration of the experimental setup. . . . 3
3 Camera setup . . . 3
4 Push-broom scanning of a surface. . . . 4
5 Image hypercube . . . 5
6 Reshaped hypercube . . . 7
7 Example of score values . . . 8
8 Early- and latewood masks for the samples treated in the UV chamber. . . . 9
9 Smoothing effects . . . 10
10 Mean spectra for weather-exposed samples, earlywood. . . . 11
11 Mean spectra for weather-exposed samples, latewood. . . . 11
12 EMSC effects . . . 13
13 Coefficient vectors, weather-exposed samples . . . 17
14 Coefficient vectors, UV-chamber samples . . . 18
15 Coefficient vector shrinkage . . . 19
16 Bias-variance trade off . . . 20
17 PRESS values example . . . 21
18 Lignin content in ground wood. . . . 24
19 Predictions of UV-exposure on the earlywood spectra from the samples left outside. All of the sample points were used to fit the model. . . . 25
20 Predictions of UV-exposure on the latewood spectra from the samples left outside. All samples used to make to fit the model. . . . 25
21 Ground wood predictions . . . 26
22 Predictions, latewood, cleaned . . . 26
23 Coefficient vectors for the model with all samples used for training, with the anomalous observations included. . . . 27
24 Coefficient vectors for the model with all samples used for training, anomalous observations excluded. . . . 27
25 Validation results, earlywood . . . 28
26 Validations results, earlywood, "cleaned" . . . 28
27 Predictions of earlywood UV exposure for samples in UV-chamber. All data points used for model fitting. . . . 29
28 Predictions of latewood UV exposure for samples in UV-chamber. All data points used for model fitting. . . . 30
29 Predictions of UV exposure on samples in UV-chamber using model fitted to weather exposed samples. . . . 30
30 Lignin vs UV, weather-exposed . . . 32
31 Lignin vs UV, UV chamber . . . 32
32 Predicted days of exposure for weather exposed samples, using ridge regression. . . 35
33 General work flow . . . 39

Acknowledgements

Several people have been of great importance during the writing of this thesis. I would like to thank my co-supervisor Kristian Hovde Liland at Nofima/IKBM, for insights into challenging and, for me, unfamiliar territory in hyperspectral imaging and statistics, and for excellent help in problem solving when encountering "unexplicable" bugs in the code. Lone Ross Gobakken and Janka Dibdiakova at Skog & Landskap for contributing expertise in the field of wooden materials, Anna and Jakub Sandak for supplying data for preliminary analysis and for enlightening discussion of error sources and how to address them, and Ulf Geir Indahl for an excellent introduction into the field of statistics that ended up becoming the focus of the thesis. Further, I would like to thank Ingvild Vadla Sørensen and May-Linn Sortland for proofreading, and friends and family for moral support.

Last, but certainly not least, I would like to thank my supervisor Ingunn Burud, who I feel has gone above and beyond in terms of availability, interest and enthusiasm, which has made the writing process a positive experience. I could not have hoped for better counsel.

1 Introduction

The use of wood as a building material has a long tradition in Norway, and is still widespread. In the recent years, there has been renewed interest in the use of wood as the main building material in large scale projects as well, as opposed to being reserved mainly for private and small scale projects (Aasheim, 2012; Treu et al., 2014). At the same time, a multitude of new treatment methods are being used to increase the longevity of the wood (Treu et al., 2014) by improving its resistance to the weathering effects of the typically harsh Norwegian climate. There is also a trend moving towards less surface treatment of the wood, and letting it weather naturally (Thiis et al., 2015; Gobakken, 2009). To develop the most successful procedures, it is necessary to have a firm understanding of the forces that lead to the deterioration of the building materials. While it is known that UV radiation is a major cause of decrease in the lignin content of wood (Müller et al., 2003) and its subsequent yellowing, there are several other factors involved, such as moisture, temperature, chemicals and mechanical abrasion (Gobakken, 2009). Since lignin constitutes about 30 % of the dry weight of softwood and 20 % of hardwood (Kansal et al., 2008), understanding the mechanics behind its degradation is an important goal, and it involves not only understanding the different influences themselves, but also their interactions.

The focus of this work is the influence of UV radiation on the weathering of wood, and its perceptibility in NIR-spectra images of wood samples. According to Grossman (1994), a widely used practice has been to use total Joules as the timing variable for the amount of radiation a sample has been exposed to. This is obtained by integrating over the entire spectral power curve of sunlight, and multiplying by exposure time (Williams, 2005). If the amount of UV radiation that a sample has been exposed to could account for most of the degradation of the lignin, then it should be possible to obtain good predictions of lignin degradation given that information about the total UV exposure is available. In turn, if changes in UV exposure is well reflected in the spectra, it should be possible to determine exposure times based on said spectra, as deviations from a certain baseline. Being able to use NIR spectroscopy to accurately determine exposure times and lignin content would be a useful tool, as it can be used quickly, cheaply and at scale. However, Grossman observes that using Joules as the timing variable gives the illusion of precision, while in reality the effect of a Joule is dependent on temperature and humidity of the sample, and the spectral power distribution of the light (that is, one Joule at 800 nm has a different effect that one at 1600 nm). Brennan & Fedor (1993) also noted that while using accelerated UV/condensation testers in a controlled laboratory setting may be useful for quality control and product development, it might not correlate well with outdoor exposure tests.

In addition to exploring how weathering changes the spectral profile of the wood, it would be of great interest to be able to determine the constituent chemical amounts based on this profile. In a previous study by Yeh et al. (2004), transmittance NIR spectroscopy was used to determine the lignin content of Loblolly Pine (*Pinus taeda*). In the study, PLS and multiple regression of NIR spectra were used to predict the lignin content of wood meal, obtaining good predictions for the training set, but yielding unsatisfactory results when predicting new samples. It was also noted that the extensive sample preparation involved makes the method unfeasible for rapid determination. In the same study, attempts were made at predictions based on images of thin wood wafers, which produced more generalizable results while avoiding the hurdles of a complex sample preparation procedure. Here, this approach is built upon by comparing results for PLS and PCR regression, with results obtained using Tikhonov regularization.

After accounting for the materials and equipment used, I will briefly cover the pre-processing

procedures used, and then go into some more detail on the statistical analysis. Here I will mainly focus on the Tikhonov regularization method, and cover PCR and PLS only in passing, since these were used mainly for comparison. The results will then be presented and discussed, along with a resumé of the work that was performed and the practical problems that were met.

Regarding the structuring of this document, I have opted to present the bulk of the results, in the form of various plots and figures, in the appendix, and a selection of illustrating results in the main document. This is because a large amount of analyses has been performed, with a proportionally large number of resulting figures and segments of code. To maintain the flow of the discourse and avoid too much clutter, a lot of the figures and documentation of the code has been placed in the appendix for review.

2 Materials and equipment

Materials
The material on which the study was conducted consists of 134 slices of Norway spruce (*Picea abies*) of $\sim 100\mu$m thickness (Fig. 1), collected from the same spot on the same tree at a height of two meters from the ground, and having an effective exposed surface of 30×30mm.

Figure 1: An example of the wood slices that was recorded with the camera.

The samples were split in two groups and exposed to two different treatments.

Group one consisted of 104 of the samples. These were recorded with a near-infrared hyperspectral camera, and subsequently placed outside in the same location in Ås, Norway, facing south at 45° for various lengths of time between 0 and 39 days, illustrated in figure 2. Note that the figure shows treatment times between 1 and 21 days, which was the case for most samples. A few of the samples was left out for significantly longer, up to 39 days. After treatment the samples were collected and recorded again.

Group two consisted of the remaining samples, which were recorded with the same camera and subsequently treated with a UV instrument for 1, 2, 3, 4, 5, 8, 9 or 10 cycles, each lasting 2.5 hours, with intermediary wetting for 0.5 hours. The samples were then recorded again.

A subset of the samples from each group were also analyzed with a Simultaneous Thermal Analyzer coupled with Fourier Transform InfraRed spectrometer (STA-FTIR) to accurately measure the lignin and holocellulose (cellulose + hemicellulose) content. The samples on which this additional measurement was performed had all been placed outside on the same day, and the samples were chosen such that the whole range from 1 to 21 days of exposure were covered, with equal intervals.

In addition, results from triplicate measurements of spectra of ground wood from the Trees4Future Project, along with corresponding klason lignin content (Alves et al., 2006), were used to per-

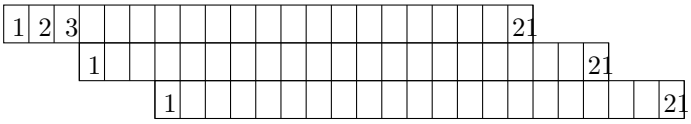


Figure 2: Illustration of the experimental setup.

form some preliminary tests and establish a baseline performance of the model. There were 597 observations from repeat measurements of 199 samples in total, taken using an instrument with a range of 833-2500 nm and 0.8 nm intervals, yielding 2075 variables for each observation. Each measurement were taken as the average of 32 successive scans. The results for each sample were averaged again before analysis.

Equipment

A SPECIM SWIR (Spectral Imaging Ltd., Oulu, Finland) near-infrared camera was used for recording the images, providing a spectral range and resolution of 1000-2500 nm and 12 nm, respectively. The camera is of the line-scan variety, and to record the whole sample a push-broom approach was used (Fig. 4), in which the sample or camera is kept still while the other is moving. In this case the sample was kept still while the camera was moved across the surface by a rail attached to and controlled by proprietary software bundled with the camera (Fig. 3). The laser shown in the image was not used for here. For lighting, a box containing six halogen lamps covered by a diffuse glass panel was placed underneath the sample. Thus the signal recorded by the camera represented the transparent to NIR wavelengths transmittance of the sample at each pixel. The sample was covered by a clear, NIR-transmitting glass panel to keep it flat. For the UV exposure samples an Atlas UVTest^{TM} UV instrument was used, in which was installed eight UVA-340 lamps with an output of 0.89 W/m^2/nm.

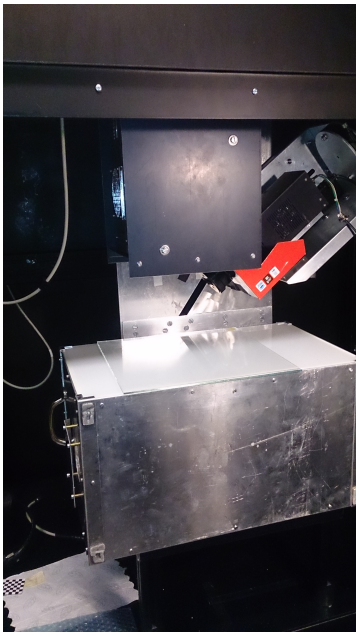


Figure 3: The camera setup for the recording of the images.

For all processing and analysis of the collected data, Matlab R2015b (The MathWorks, Inc.) was used, supplemented with in-house MatLab routines programmed by Kristian Hovde Liland, in-house routines by José M.A. Rubio[1], and PLS and regularized regression code implementation by Ulf G. Indahl[2].

[1]The Hypertools package, Department of Food, Quality and Technology, University of Copenhagen

[2]Presented in lectures on Data Mining and Pattern Recognition, Dept. of Mathematical Sciences and Technology, Norwegian University of Life Sciences, spring 2015

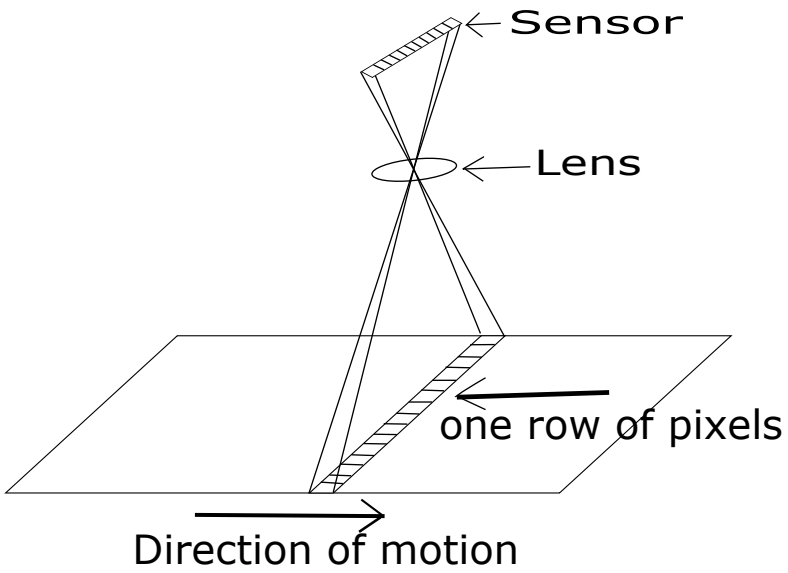


Figure 4: Push-broom scanning of a surface.

3 Methods

When working with a large number of samples, manually performing various operations by trial and error for each, is a time consuming process. It has therefore been a goal to automate these operations, the philosophy being that it is preferable to spend some more time initially to produce generalized code, which can then quickly and easily be applied to new material. As a result, substantial time and effort was spent to produce the code supplied in the appendices. While they can certainly be streamlined, for example by rewriting the scripts to functions, I believe they are relatively well adapted for use with new data, even if some parameters would likely have to be tweaked. Also note that many of the scripts contain several lines of code that may seem superfluous, but are included to provide insight into the performed operations and their results. In other words, for diagnostic purposes. Also, while most of the worker functions used throughout are not written by me, except for the automated masking function, substantial work has been put into making them work together, so that the whole process from importing the initial raw images to the resulting statistical analysis can be performed as quickly and with as few modifications as possible.

Work on the data collected can be divided into three steps. Initially, basic data preparation was performed. The camera outputs images in `.raw` format, which had to be converted to a three-dimensional hypercube (Fig. 5) that can be processed by MatLab. Following this conversion, each image was corrected for level and white balance.

In each image, the camera includes an area that is recorded with the shutter closed, thus this area represents for this camera intensity 0. To correct the rest of the image, the mean of each column of pixels in this area is subtracted from each pixel in the remaining image, in each waveband:

$$x_{bcorr}^{(i,j,k)} = x^{(i,j,k)} - \bar{x}^{(j,k)}$$

White balance correction is performed on the image to negate the effects of varying intensities

of the light hitting different parts of the sample. This lack of uniformity is a source of variation not related to the chemical buildup of the sample and has to be accounted for. This was done by dividing each value in the image cube by its corresponding value in an image taken with only the background and no sample, thus representing maximum whiteness in that pixel:

$$x_{wcorr}^{(i,j,k)} = \frac{x^{(i,j,k)}}{x_w^{(i,j,k)}}$$

where both x and x_w has already been corrected in accordance with the previous equation. After

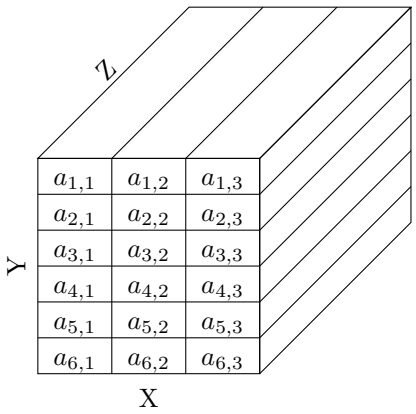


Figure 5: Hypercube representing one image. The X and Y dimensions are rows and columns of pixels, respectively, while wavebands are ordered along the Z dimension.

these corrections the hypercube was cropped spatially to include only the center of each sample. The same coordinates were used on each image to ensure identical dimensions. Finally the set of prepared images was stored as a four-dimensional hypercube, with dimensions $[height \times width \times wavebands \times samples]$. These operations were performed with the function `read_raws.m` (Appendix C.1).

Some of the samples that were exposed to treatment were missing when the imaging was performed, and the samples were sorted differently in the spectral hypercube and in the response vector. The missing samples had to be removed from the response vector, and the remaining samples had to be matched. This was automated with the implementation in `import_images.m` (Appendix C.1). This concludes basic data preparation.

3.1 Pre-processing

After initial preparation of the data, a set of algorithms were applied to facilitate the statistical analysis by enhancing the data and separating parts of the image that were assumed to be sufficiently dissimilar as to warrant a separate analysis. In this context, that involved separating areas of early- and latewood. In the spring and summer months, when there is ample supply of sunlight, the tree has a higher growth rate compared to the winter and late autumn months (Heide, 1974). This affects the density and chemical makeup of the parts of the wood that are from different months of the year to such a degree that it is likely they should be analyzed separately (Westermark et al., 1988). In a study by Heide (1974) it was shown that a transition to formation of tracheids with thick walls and narrow lumens took place in the late-wood when

exposed to the short days associated with the late growth period. This denser late-wood is less translucent, and thus intensities in pixels corresponding to these areas will be lower. Another factor is that the more porous summer wood are likely to react differently to desiccation, presumably in a stronger fashion since it initially contains more water (Fromm et al., 2001). Third, Bertaud & Holmbom (2004) found that lignin concentration is higher in earlywood than latewood. Hence we expect to see a more pronounced effect in the earlywood as the lignin degrades, a process that is the focus of this study.

In the interest of deriving a model that performs well in the sense that it successfully predicts outcomes with a high degree of reliability, the data set on which the model is based should contain as little noise as possible, as in any statistical analysis. In the case of hyperspectral images, where the variables measured are subject to certain physical characteristics, a special set of methods can be used separately or in combination to enhance the data and optimize it for statistical analysis (Kohler et al., 2009). It is a also used to decouple the physical and the chemical variation in the sample, so they can be studied individually. A summary of some of these methods and their performance can be found in (Esquerre et al., 2012; Kohler et al., 2009).

3.1.1 Segmentation

It is easy to visually identify which parts of a piece of wood is from which growth period (early or late). Getting a computer to do this automatically for a set of different images is a somewhat more involved process, and one that can likely be performed in many different ways. It has to take into account that the range of intensities in the images may vary, and that some images may contain pieces of background while others do not, and deal with this. In this case a relatively simple algorithm utilizing singular value decomposition (SVD) of the images were used to differentiate the parts that belong to the early and late growth periods.

Singular Value Decomposition
It can be shown that any real $m \times n$ matrix A can be factorized as $A = USV^t$, where $U \in \mathbb{R}^{m \times m}$ is an orthonormal matrix containing the eigenvectors of AA^t, $V \in \mathbb{R}^{n \times n}$ is an orthonormal matrix containing the eigenvectors of A^tA, and

$$S = \begin{pmatrix} \Sigma \\ \mathbf{0} \end{pmatrix}$$

where $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing the singular values of A (Lay, 2012; Krishnan et al., 2011; Eldén, 2007). $\mathbf{0}$ is an $(m-n) \times n$ zero matrix. Thus,

$$U^tU = V^tV = VV^t = I_n$$

and, in general

$$UU^t \neq I$$

is the projection matrix onto Col(A).

If A is an $m \times n$ matrix, then A^tA is symmetric and orthogonally diagonizable. Letting V be an orthonormal basis for $\mathbb{R}^n$ constructed from the eigenvectors of A^tA and letting $\lambda_1, ..., \lambda_n$ be the corresponding eigenvalues, for $1 \leq i \leq n$,

$$\begin{aligned} ||A\vec{v_i}||^2 &= (A\mathbf{v}_i)^t A\mathbf{v}_i = \mathbf{v}_i^t(A^tA)\mathbf{v}_i \\ &= \mathbf{v}_i^t(\lambda_i \mathbf{v}_i) \\ &= \lambda_i \end{aligned}$$

Taking the square root of these eigenvalues of A^tA, we get the singular values $\sigma_1, ..., \sigma_n$ of A. These are arranged in decreasing order of dominance; the degree to which they stretch or shrink their corresponding eigenvectors. That is, variation in the direction specified by one eigenvector is larger than in any of the following ones. If $rank(A) = r < n$, then $\sigma_{r+1}, ..., \sigma n = 0$, that is, all the variation is accounted for by the previous singular values. This is useful if we want to reduce the size of a data set or extract only the significant variation contained within, as we can construct a *Rank z approximation* by using only the z first singular values, and padding the S matrix with zeroes

$$A_z = US_zV^t \quad \text{where } S_z = \begin{bmatrix} \underset{z\times z}{\Sigma_z} & \underset{z\times(n-z)}{\mathbf{0}} \\ \underset{(n-z)\times z}{\mathbf{0}} & \underset{(n-z)\times(n-z)}{\mathbf{0}} \end{bmatrix}$$

The fraction of the total variability accounted for by the approximation is given by

$$\frac{trace(S_z)}{trace(S)}$$

There are several computer algorithms for performing SVD. The method described here initially is based on eigenvalues, but in reality calculating the eigenvalues of A^tA explicitly proves to be a numerically unstable solution (Golub & Reinsch, 1970). Matlab implements routines from the LAPACK software library (Moler, 2000), which utilizes bidiagonalization to improve numerical accuracy (Anderson et al., 1999). Further details about the algorithms is beyond the scope of this text, but the theoretical foundations can be found in Golub & Reinsch (1970).

Implementation

The goal with the segmentation was to automatically create two logical masks for each image that identified the parts belonging to early and late wood. The clear visual distinction between the early and the late wood parts of the sample was likely to be reflected in one or more of the first components of the SVD, since it is such a dominant factor. Initially, each image in the sample set was reshaped from a tensor (Fig. 5) to a matrix (Fig. 6), for which normal SVD is defined. While there are extensions to SVD that applies to higher order tensors (i.e. Tucker in chemometrics) (Eldén, 2007), their use is outside the scope of this text, and neither were they needed. To account for the fact that some of the images included parts of the white background (yielding intensities close to 1), or black foreground (yielding intensities close to 0), it was necessary to include cut-off values before performing the SVD. Looking at the images, these regions were most clearly visible in the middle wavebands, so the values to be compared to the cut-off value were from one of these, specifically band 125. Suitable values to use were .15 and .85 for earlywood and .65 and .85 for latewood, found by trial and error. Pixels with values in the 125th waveband below or above these, respectively, were assigned a value of 0 in the masking matrix and excluded from the SVD.

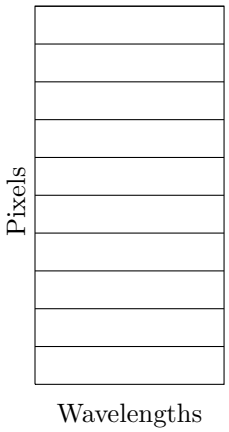


Figure 6: Reshaped hypercube

After performing SVD of several of the images and inspecting score values calculated with different components, it was found that the distinction between early and late wood was best represented in the second component (Figure 7). Thus score values for each image

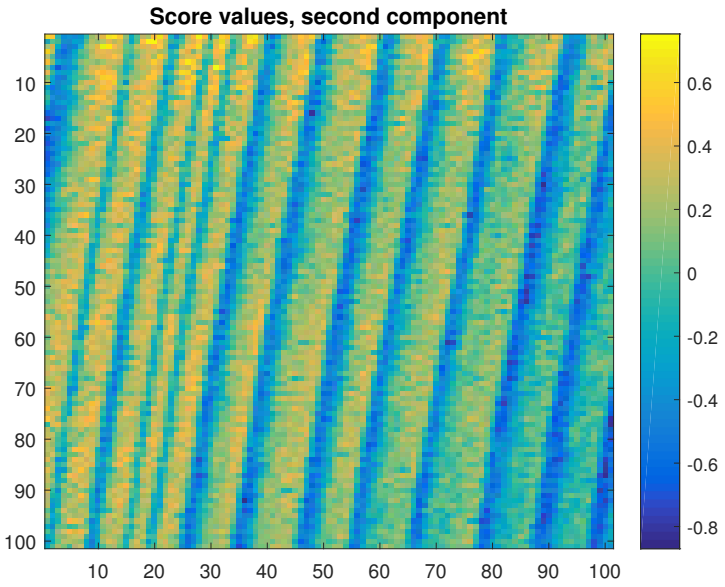


Figure 7: Example of score values

based on this component was used as the basis for thresholding,

$$\mathbf{x}_i = A_i \mathbf{v}_{i,2}$$

where A_i is the reshaped image i, $\mathbf{x}_i$ is the score values of image i, and $\mathbf{v}_{i,2}$ is the second right singular vector of A_i. Early wood areas of the image was characterized by low score values and late wood by high score values, but the ranges varied from image to image. To avoid this problem, quantiles were used instead of absolute values. For this particular dataset, the lower .25 and the upper .35 quantiles proved suitable. The early and late masks were then updated to include only pixels with scores in those quantiles, respectively. This is implemented in the function `make_mask.m` (Appendix C.2). Output for the UV-chamber samples is shown in Figure 8.

3.1.2 Smoothing

Since we know that the electromagnetic spectrum for all intents and purposes is continuous, and that the signal at one wavelength is strongly correlated with signals at proximate wavelengths, we can conclude that ideal spectra recorded by our instrument should have the same characteristics. This, however, is rarely the case. Various instrumental and atmospheric influences perturb the actual signal and lead to what is collectively termed 'noise' in the data. Whereas we cannot, in statistics in general, conclude on the validity of data points without further investigation, our *a priori* knowledge in this case permits the use of a set of tools to remove some of this noise before any further analysis is performed (Savitzky & Golay, 1964), improving the signal-to-noise ratio. The simplest of these is calculating a moving average over the the whole spectrum, by defining a window size and averaging the ordinate values within that window, and assign this value to the data point at the centre abscissa of the interval. The window is then moved one step and

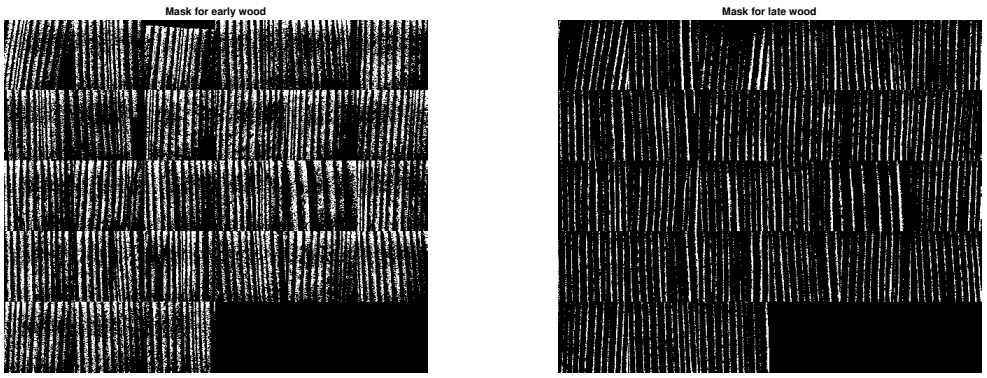


Figure 8: Early- and latewood masks for the samples treated in the UV chamber.

the process is repeated. Variations include various weighted averages, so that data points closer to the center abscissa are given a stronger influence on the final average. The various moving averages has the undesired effect of also smoothing over any peaks in the spectra and reducing their intensity, which may amount to removing important information (Savitzky & Golay, 1964).

As an improvement, Savitzky & Golay proposed an alternative solution based on the method of least squares. A model (usually a low-degree polynomial model) is fitted for the data points within a window of predetermined size. The value of the center abscissa of the window is then inserted in the model, and the ordinate value at this coordinate is replaced by the result.

$$y* = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_n x^n$$

The window is then moved one step, as when calculating the moving average, and the process is repeated. This method was built further upon by showing that when fitting the model solely as a way to estimate a single point, it is possible to substitute the model fitting by a set of integers, termed "convolution coefficients" that provide the weighting function, thus speeding the procedure greatly. When the observations are equally spaced, such sets of integers can also be found for derivatives of those functions (Savitzky & Golay, 1964). The convolution coefficients for the polynomials and their derivatives for various window sizes are supplied in the original article. Before the smoothing, the image was transformed from absorbance to transmission with a $-\log_{10}$ transform. A few of the images turned out to contain some dead wavelengths with intensities 0. Since $-\log_{10}(0) = \infty$, this caused problems further on. This was remedied by applying a spike removal procedure (`htdeadwavelengths` from Hypertools). The ten first and last bands were also excluded, since they were entirely dominated by noise. The code implementation used for the smoothing was from the Hypertools, specifically the function `filpoly.m`. The ideal window size depends on the wavelength resolution, and in this case a window of nine was found to produce the best results. The model used was a second-degree polynomial, without derivatives. Illustration of the effect is shown in Fig. 9.

3.1.3 Scatter correction

In order to study the chemical makeup of the samples most successfully, it seems advantageous to separate the physical variation from the chemical one. To accomplish this, a different set of

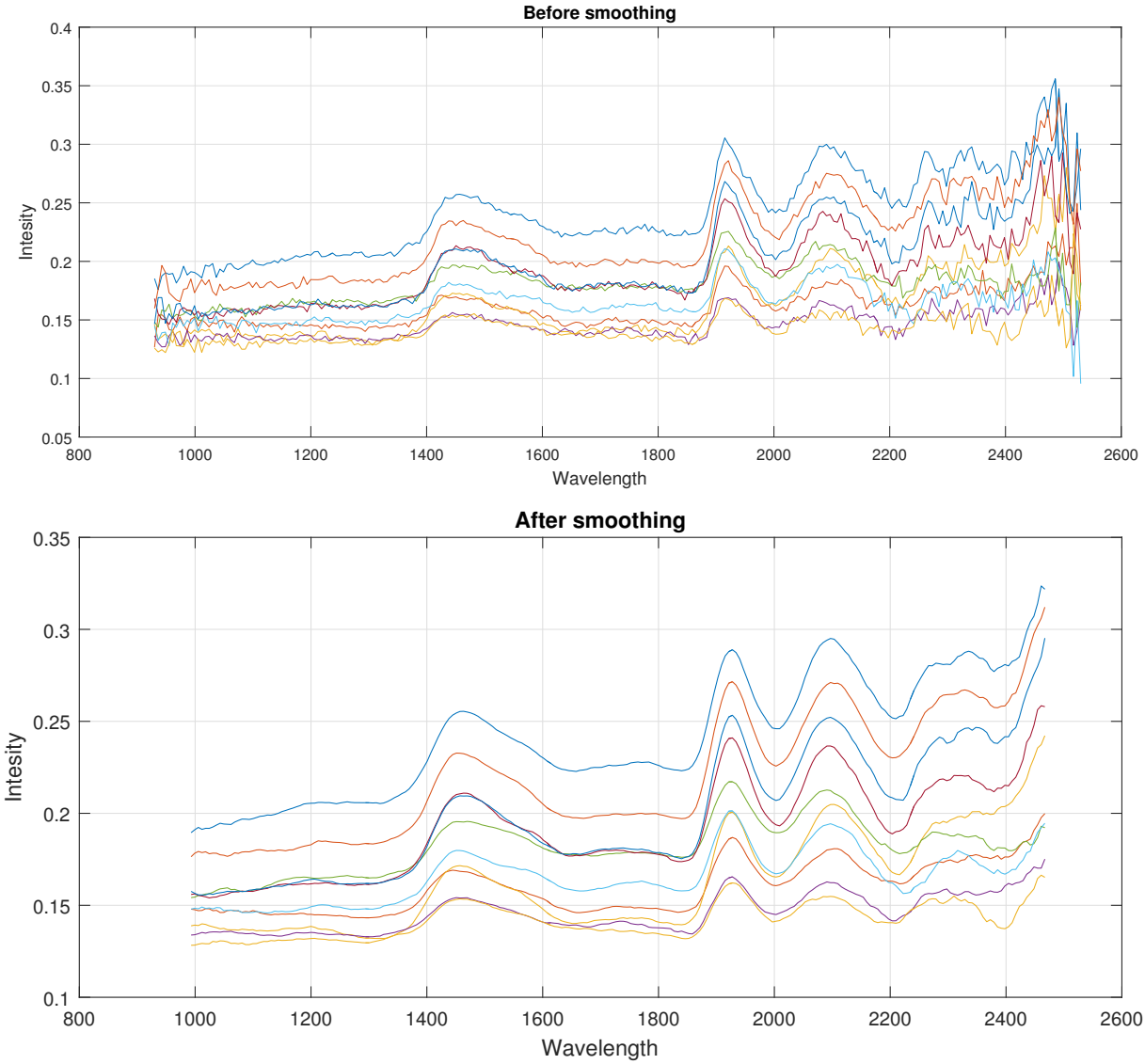


Figure 9: Ten random pixels from an arbitrarily selected image, before and after smoothing.

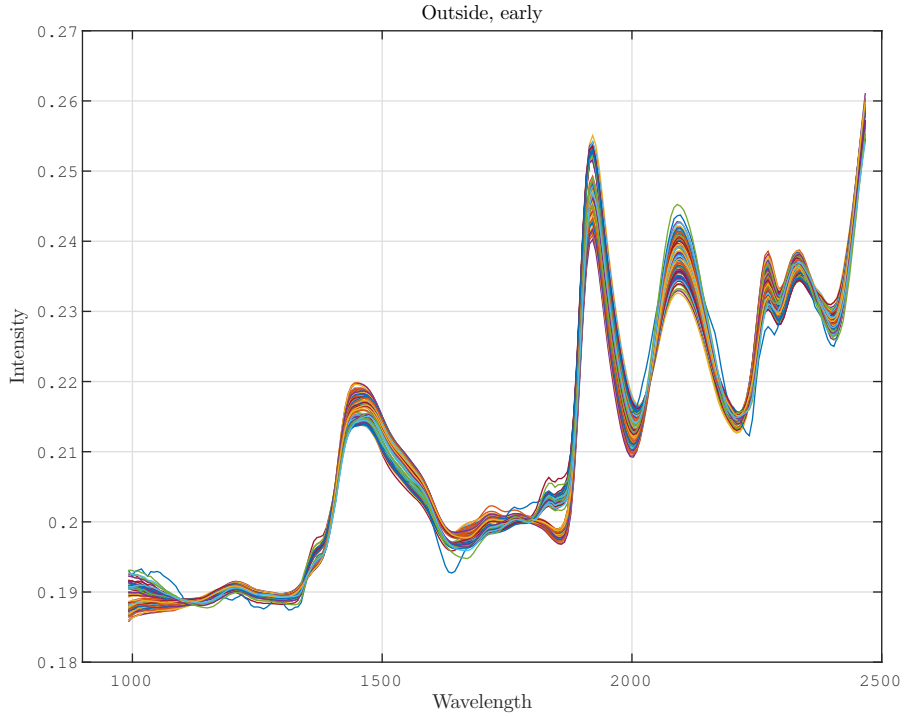


Figure 10: Mean spectra for weather-exposed samples, earlywood.

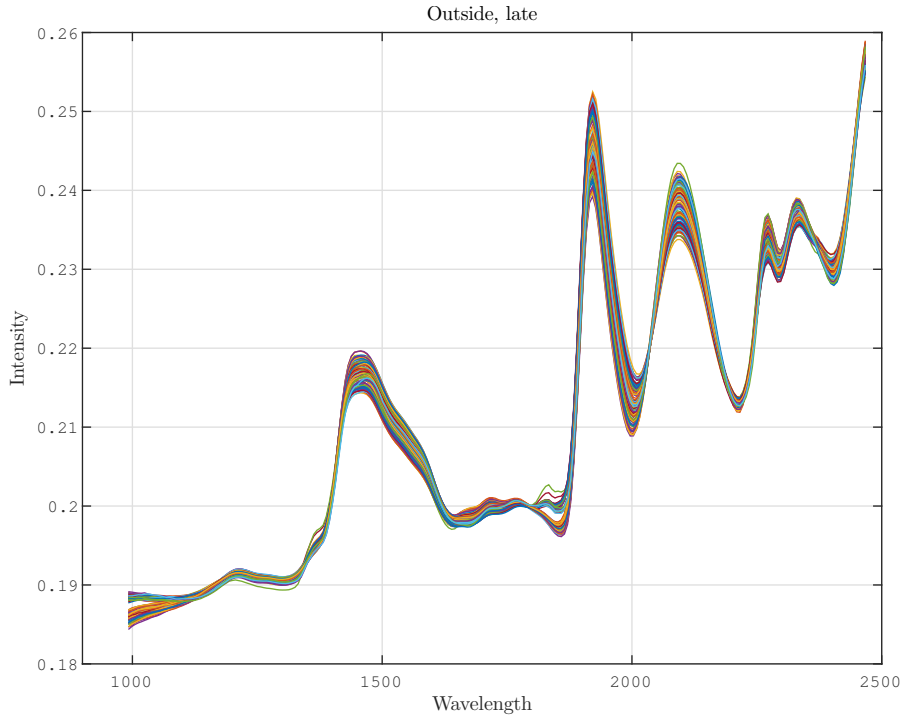


Figure 11: Mean spectra for weather-exposed samples, latewood.

tools than those used for smoothing is needed, and one of the most meritorious of the available methods is (Extended) Multiplicative Signal Correction, or (E)MSC (Esquerre et al., 2012). The basic idea of MSC is to fit a regression model of each spectra in an image on an ideal (reference) spectra, and then use the coefficients to perform the correction.

According to Lambert Beer's Law we can view the absorbance spectra as the sum of contributions from each constituent compound in the sample (Li et al., 2013) :

$$\mathbf{z}_{chemical} = c_1 k_1 + c_2 k_2 + ... + c_j k_j$$

where c_j is the concentration and k_j is the spectrum of compound j, respectively. In the absence of any interfering physical phenomena, we could study this as-is without further pre-processing. However, because interfering physical phenomena such as variation in baseline and sample thickness are present, we can use the following formula as a starting point (Kohler et al., 2009):

$$\mathbf{z}_{observed} = \mathbf{z}_{chemical} + \mathbf{z}_{physical} + \varepsilon$$

In MSC we want to estimate parameters that we can use to adjust for additive and multiplicative effects. As mentioned we do this by regressing $\mathbf{z}_{observed}$ on $\mathbf{z}_{chemical}$ to obtain parameters α and β so that

$$\mathbf{z}_{observed} = \beta \mathbf{z}_{chemical} + \alpha + \varepsilon$$

If the pure spectra of the sample is available it can be used for the reference, but often the mean spectra $\mathbf{m}$ of the sample is used instead (Kohler et al., 2009). The equation then becomes

$$\mathbf{z}_{observed} = \beta \mathbf{m} + \alpha + \varepsilon$$

where β is the scaling parameter and α is the baseline offset. The coefficients are then used to correct the observed spectra:

$$\mathbf{z}_{corrected} = \frac{\mathbf{z}_{observed} - \alpha}{\beta}$$

EMSC extends this by adding terms that account for variation that can interfere with the calculation of the mentioned parameters, such as differences in scattering over different wavelengths (Kohler et al., 2009; Li et al., 2013). In this case, a model with parameters for wavelength and squared wavelength was used, so the model becomes

$$\mathbf{z}_{observed} = \alpha \mathbf{1} + \beta_1 \mathbf{m} + \beta_2 \lambda + \beta_3 \lambda^2$$

where λ is the wavelength. The final correction then becomes

$$\mathbf{z}_{corrected} = \frac{\mathbf{z}_{observed} - (\alpha + \beta_2 \lambda + \beta_3 \lambda^2)}{\beta_1}$$

The effect of each step is shown in Fig. 12 This is performed for each spectrum in a sample, one for each pixel. The samples used each had $101 \times 101 = 10201$ pixels, amounting to 10201 sets of coefficients. The code implementation were from the Hyspec package [3].

The code for performing all these operations was used on each of the data sets, and are included in Appendix B.

[3] In-house routines by Kristian Hovde Liland, Nofima.

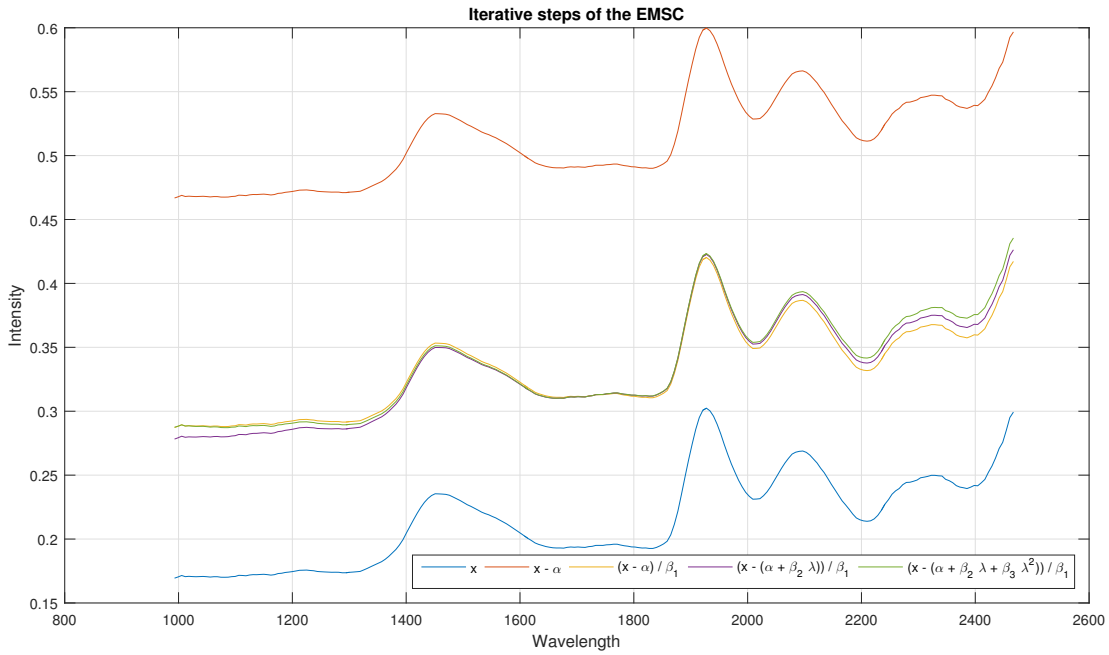


Figure 12: Original spectrum and effect of each step for a random pixel.

3.2 Statistics

After pre-processing was complete, the material ready for analysis consisted of four sets of data, which were the early- and latewood mean spectra for each sample in each group (outside exposure group and UV-chamber group), as well as the pre-processed original images themselves. Three separate approaches were used to analyze the data: (a) using the mean spectra as predictor variables and the number of days/cycles of exposure as the response variable. (b) Using the complete sets of spectra from the images subject to the least and most exposure as predictor variables and a dummy variable denoting the amount of degradation as the response, and (c) Using weather data from the period to calculate total solar radiation for each sample. The first two had some apparent flaws:

(a) While this was in itself a fairly straightforward analysis to do, it did not model what we wanted to model. Informed by the assumption that it is the exposure to UV radiation that is the main reason for the degradation in the samples, and the fact that the amount of radiation varies with the weather and time of year, it made sense to look for a better response variable to use.

(b) The idea behind this approach was to use a measure of "degradation" of the sample as the response variable, so that the least degraded (i.e. least exposed) sample got a value of 1, and the most degraded sample got a value of 100. The stacked complete spectra (i.e. not means) of these samples were used as predictor variables while a dummy variable with 1's and 100's were

used as the response:

$$X = \begin{bmatrix} Z_1 \\ {\scriptstyle m\times n} \\ Z_2 \\ {\scriptstyle p\times n} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} \mathbf{1}_m \\ 100 \cdot \mathbf{1}_p \end{bmatrix}$$

A model was then fitted to these data, and used to predict the degradation of the remaining samples based on their mean spectra. Since the data was arranged according to the amount of exposure, a good fit should translate into a smooth slope when plotted. This turned out not to be the case. The choice ultimately fell on using the weather data to calculate a new response variable denoting total radiation exposure, and fitting a model based on mean spectra.

3.2.1 Least Squares Regression

Here follows a short review of some fairly basic statistics, to provide a foundation to build upon and to shed some light on why it might be advantageous to employ the methods used in the study.

We are trying to predict total radiation exposure in the samples based on their spectra. The set of predictor variables and the response variable are all continuous, and so our point of departure is Ordinary Least Squares Regression, of the familiar form shown earlier: $y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \varepsilon$, where β_0 is the expected value when all $x = 0$, $\beta_1, ..., \beta_n$ are the coefficients associated with each variable, and ε is the error term representing variation not accounted for by the model, and assumed to have the distribution $\varepsilon \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$. In linear algebra notation the data can be arranged as (Lay, 2012):

$$X\boldsymbol{\beta} = \mathbf{y}$$

where the rows of X correspond to observations and the columns correspond to variables, with a unit vector added to account for the constant term. $\boldsymbol{\beta}$ is a vector containing the regression coefficients and $\mathbf{y}$ is the response vector.

$$\begin{bmatrix} \mathbf{1}_m & \underset{m\times n}{X} \end{bmatrix}$$

The normal equations are defined as

$$X^t X\boldsymbol{\beta} = X^t\mathbf{y} \tag{3.1}$$

which given that X^tX is invertible gives the least squares solution

$$\hat{\boldsymbol{\beta}} = (X^tX)^{-1}X^t\mathbf{y} \tag{3.2}$$

which are used as coefficients for the linear combination of the columns of X to obtain the estimate of y:

$$\hat{\mathbf{y}} = X\hat{\boldsymbol{\beta}} \tag{3.3}$$

This is in effect a projection of $\mathbf{y}$ on the space spanned by the columns of X (Lay, 2012):

$$\hat{\mathbf{y}} = \text{proj}_{ColA}(\mathbf{y})$$

The residual sum of squares, that is the variation not accounted for by the model, is given by $RSS(\beta) = (\mathbf{y} - X\beta)^t(\mathbf{y} - X\beta)$ for any coefficient vector β (Hastie et al., 2008). If β is the $\hat{\beta}$ from Eqn. (3.2), we obtain the minimum RSS:

$$\hat{\beta} = \min_{\beta} ||\mathbf{y} - X\boldsymbol{\beta}||^2$$

It was stated earlier that we want to derive a model that predicts new observations *accurately* and *reliably*. In the statistical literature these two properties of an estimator are referred to as its *accuracy* and its *precision*. Viewing the model as an estimator for y, the accuracy of the model describes its ability to make predictions of a parameter that are close to the true value, and a model who deviates from this is biased. The precision of the model is how much fluctuation there is in these predictions (Menditto et al., 2007), and is quantified in terms of its variance.

Suppose we have an observation y from a population with an unknown mean μ, an associated variable x, and an estimate of y given by some function $\hat{y} = f(x)$. Then the mean squared error (MSE) of $\hat{y}$ is given by

$$\begin{aligned} MSE(\hat{y}) &= E(y - f(x))^2 \\ &= E[(y - \mu) + (\mu - f(x))]^2 \\ &= E[(y - \mu)^2 + 2(y - \mu)(\mu - f(x)) + (\mu - f(x))^2] \\ &= E(y - \mu)^2 + 2\underbrace{E(y - \mu)}_{0}(\mu - f(x)) + (y - f(x))^2 \\ &= \underbrace{E(y - \mu)^2}_{Var(y)} + \underbrace{(\mu - f(x))^2}_{bias^2} \end{aligned} \tag{3.4}$$

This is known as the *bias-variance decomposition* (Hastie et al., 2008), and viewing the prediction error in this way will be useful later on.

Looking back on the least squares solution to the normal equations shown in Eqn. (3.2), it was stated that if X^tX is invertible, then a solution exists. This is true if, and only if, the columns of X are linearly independent, which is equivalent to X having full rank (Lay, 2012). It can be shown that this solution is unbiased and unique, hence it obviously has the least possible variance, as the name implies (Golub & Van Loan, 1980).

It is also possible to pose the problem in terms of the SVD of X. With the decomposition $X = USV^t$ as a starting point, define the Moore-Penrose inverse (pseudoinverse) as (Prasad & Bapat, 1992)

$$X^+ = VS^{-1}U^t \tag{3.5}$$

If X has full rank, then $X^+X = I$. Applying this to the linear system $X\beta = \mathbf{y}$, the solution becomes

$$X^+X\beta = X^+\mathbf{y} \Rightarrow \beta = X^+\mathbf{y} \tag{3.6}$$

which is unique given the full rank of X. However, for X to have full rank, it must have at least as many equations as unknowns (for an $m \times n$ matrix X, $\text{Rank}(X) \leq \min(m, n)$) (Lay, 2012). This is not the case here, since there are 236 variables (wavelengths) and only 104 observations at most. Since we have more unknowns than equations, the system is *under-determined*, and while it can still be solved as above, the solution is not unique (Eldén, 2007). Looking at the SVD of the matrix

$$X \in \mathbb{R}^{m\times n} = U\,[S\ 0]\begin{bmatrix} V_1^t \\ V_2^t \end{bmatrix}, \qquad V_1 \in \mathbb{R}^{m\times m} \qquad V_2 \in \mathbb{R}^{m\times(n-m)}$$

the columns of V_1 span the row space of X, while those of V_2 span the null space of X. The minimum norm solution is given by

$$\beta = V_1^tS^{-1}U^t\mathbf{y} \tag{3.7}$$

and the orthogonal projection is

$$\begin{aligned}\hat{\mathbf{y}} &= X\beta \\ &= XX^{+}\mathbf{y} = USV^{t}VS^{-1}U^{t}\mathbf{y} \\ &= UU^{t}\mathbf{y}\end{aligned} \tag{3.8}$$

Since V_2 span the null space of X, we can add any component here and still have a solution, but it will no longer have the smallest norm (Eldén, 2007).

3.2.2 Regularization

Often when performing regression analysis, the β coefficients themselves are of interest. This is typically the case in traditional LS problems where there are usually few variables, and the values of the coefficients can be interpreted meaningfully and provide insight into the problem being studied. However, in many modern applications where the number of variables can be several orders of magnitude larger, it is not the values of the regression coefficients themselves that are of interest. Rather, obtaining these values can be seen as an intermediary step, where establishing a stable and accurate model is the ultimate goal, with the characteristics outlined earlier. Specifically, the merit of the model is ultimately evaluated in terms of its error rate (MSE). The decomposition of this error rate into two components; bias and variance, was shown in Eqn. (3.4). This can be seen as a trade-off; in increasing the model complexity the bias can be decreased, but the variance will increase. Such a complex model may predict well for the set on which it was trained, but it might not generalize well to new data (Hastie et al., 2008). By relaxing the requirement that the model should be unbiased, it is possible to obtain a model that has stronger predictive power, by finding the optimal trade-off between bias and variance (Hoerl & Kennard, 1970). The method of doing this by adding a constant to the diagonal entries of X^tX in the least squares estimator is called Tikhonov regularization (Kalivas, 2012). It was stated earlier that the variables in a hyperspectral image are often highly correlated, because one wavelength will have values similar to its neighboring wavelengths. This can make the data matrix severely *ill-conditioned* (Hansen, 1992), having a high condition number $\mathcal{K}$, defined as

$$\mathcal{K}_X = \frac{\sigma_1}{\sigma_r},$$

where σ_r is the smallest singular value of the matrix. When this is the case, the least squares solution will be very sensitive to perturbations in the data, such as those stemming from measurement errors (Hoerl & Kennard, 1970) Since these are common in hyperspectral imaging, the model should be resistant to such perturbations. Fig. 14 shows and example of the coefficient vectors from a much more robust model, fitted to a different dataset than the one in Fig. 13. When a problem is ill-conditioned, the coefficient vector may fluctuate wildly and the values may be very large, as seen in Fig. 13. This means that each value in the data matrix will be multiplied by a large number, and thus even small variation in X will have large effects on the prediction outcome $\hat{y}$ (Eldén, 2007). The goal in regularizing the data is to shrink this vector of coefficients so it is more resilient to small variation in X. The effect of increasing the value of the regularization parameter is shown in Fig. 15. Evidently, as the value increases, the fluctuations in the vector are further reduced. It is also quite apparent that with further increase in the regularizing parameter value, the curve would end up as a straight line, in which case none of the variables would have any effect on the outcome.

So there are two objectives: Keep $O_1 = ||X\beta - \mathbf{y}||^2$ small, and keep $O_2 = ||F\beta - g||^2$ small. When $F = I$ and $g = 0$, this corresponds to keeping $O_2 = ||\beta||^2$ small, that is, putting more weight

on objective 2 emphasizes shrinking the coefficient vector. The expression of this weighted-sum objective is

$$\begin{aligned} ||X\beta - \mathbf{y}||^2 + \lambda||F\beta - g||^2 &= \left|\left|\begin{bmatrix} X \\ \sqrt{\lambda}F \end{bmatrix}\beta - \begin{bmatrix} \mathbf{y} \\ \sqrt{\lambda}g \end{bmatrix}\right|\right|^2 \\ &= ||\tilde{X}\beta - \tilde{\mathbf{y}}||^2 \end{aligned} \tag{3.9}$$

where

$$\tilde{X} = \begin{bmatrix} X \\ \sqrt{\lambda}F \end{bmatrix}, \qquad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \sqrt{\lambda}g \end{bmatrix}$$

and λ is the weight put on objective 2, called the regularization parameter. If $\lambda = 0$, the result is classic least squares. As λ approaches infinity, the coefficients get nulled out and only the intercept (mean value) remains (Kim et al., 2007). The solution of the system is

$$\begin{aligned} \beta &= (\tilde{X}^t\tilde{X})^{-1}\tilde{X}^t\tilde{\mathbf{y}} \\ &= (X^tX + \lambda F^tF)^{-1}(X^t\mathbf{y} + \lambda F^tg) \end{aligned} \tag{3.10}$$

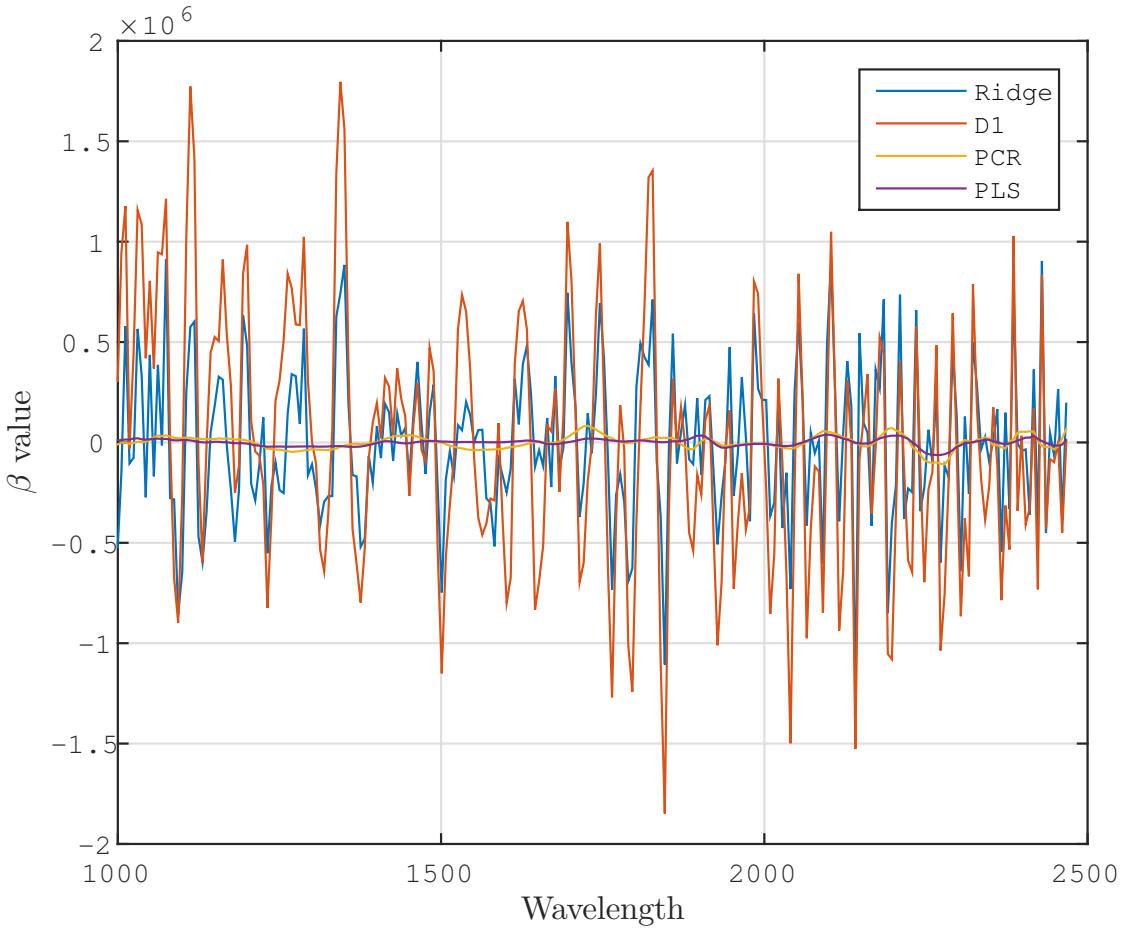


Figure 13: Coefficient vectors for the four different methods, used on the latewood spectra of the weather-exposed data.

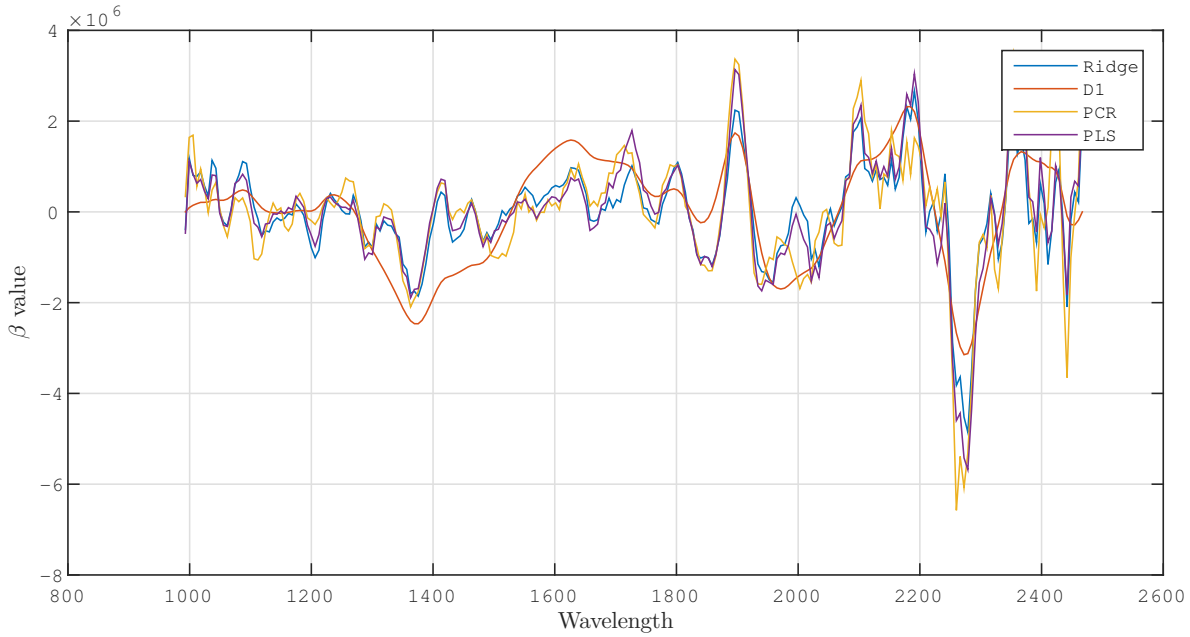


Figure 14: Coefficent vectors of the models, used on the earlywood spectra of the UV-chamber exposed samples.

Naturally, there will be one model for each value of λ, and the challenge is to find the best one in accordance with the chosen criteria. Choosing the Prediction Residual Error Sum of Squares (PRESS) as the loss function, the strategy is to iterate over a sequence of λ values, record the PRESS, and choose the model associated with the lowest of these.

The optimization are illustrated in Fig. 16. Any point on in the gray area is unobtainable. Any point on the white area is obtainable, but sub optimal. Any point on the curve is optimal and obtainable for some regularization parameter λ. Searching for the ideal choice of regularization parameter means searching for λ^* in the figure, whose line is tangential to the pareto-optimality curve.

Since the data matrix does not have full rank, the solution is found using the pseudoinverse from Eqn. (3.7). This requires performing SVD on the matrix, which can be a computationally costly operation, particularly for large matrices (Tzeng, 2013; Burger, 2006). While one instance of running the algorithm is not a problem even when the matrix is large, it becomes a problem if it has to be performed hundreds of times, one for each choice of λ. Fortunately, that is not the case in ridge regression (where $F = I$ and $g = 0$) (Indahl, 2015). Looking at the matrix

$$Z_\lambda = \begin{bmatrix} X_s \\ \sqrt{\lambda} I \end{bmatrix}$$

where X_s it the centered version of the matrix X from (eqn. 3.9) and applying what we know

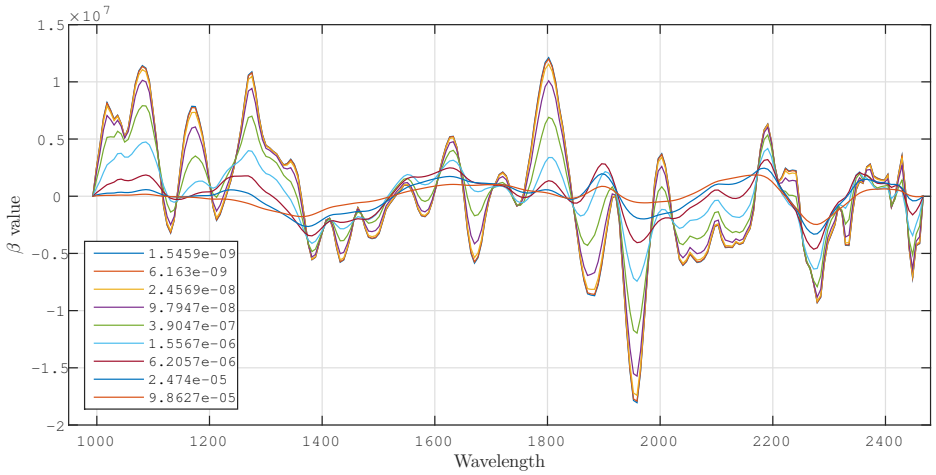


Figure 15: Shrinkage of coefficient vectors with increase in regularizing paramter value.

about the SVD, we get

$$\begin{aligned} Z_\lambda^t Z_\lambda &= (X_s^t X_s + \lambda I) \\ &= (VS^t U^t USV^t + V\lambda IV^t) \\ &= V(S^t S + \lambda I)V^t \end{aligned} \tag{3.11}$$

$$(Z_\lambda^t Z_\lambda)^{-1} = V(S^t S + \lambda I)^{-1} V^t$$

Hence the columns of V are eigenvectors of $Z^t Z$, and the corresponding eigenvalues are the square roots of the diagonal of S^2. Further, let $\mathbf{v}$ be a column in V, the left singular vectors of $Z^t Z$. Then

$$Z^t Z\mathbf{v} = (X_s^t X_s + \lambda I)\mathbf{v} = X_s^t X_s \mathbf{v} + \lambda I\mathbf{v} = X_s^t X_s \mathbf{v} + \lambda \mathbf{v} \tag{3.12}$$

Since $\mathbf{v}$ is a multiple of $X_s^t X_s^t$ and λ, it must be an eigenvector of both. This means that performing the SVD only once, we have all the information we need to test any number of λ values. The singular vectors does not change and can be reused throughout the iterations, and the diagonal matrix containing the singular values need only be updated by adding the regularization parameter to each entry, a computationally cheap operation (Rifkin & Lippert, 2007; Indahl, 2015).

To find the β-coefficients for a given choice of λ, following the reasoning in Eqn. (3.11) the solution is obtained as (Indahl, 2015)

$$\begin{aligned} \boldsymbol{\beta}_\lambda &= V(S^t S + \lambda I)^{-1} V^t VSU^t \mathbf{y} \\ &= V(S^t S + \lambda I)^{-1} SU^t \mathbf{y} \end{aligned} \tag{3.13}$$

Having established an efficient way to estimate the regression coefficients for each value of the regularization, a way to calculate the PRESS value for that model is needed. This value should indicate how well the model fits to new data, and a common approach to this is to use cross-validation. The basic idea is to fit a model to the data set with one observation left out, and then use the model to predict the value of the left-out observation. This is repeated for each

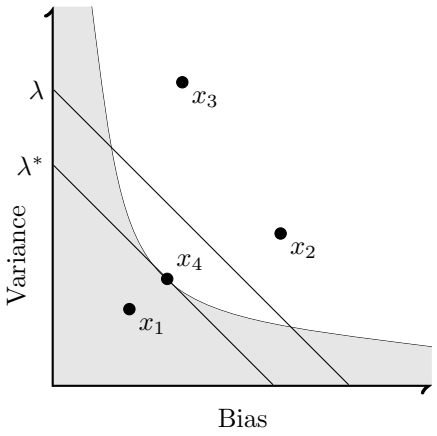


Figure 16: Pareto-optimal of Bias vs Variance. x_1 and any point in the shaded area are unobtainable, x_2 and x_3 and any point in the white area are sub-optimal but obtainable, any point on the curve is optimal for some choice of regularization parameter, and x_4 is the optimal for the best choice of regularization parameter, the one who will be chosen.

observation in the dataset, and the sum of the squared residuals is the PRESS value,

$$PRESS_\lambda = \sum_{i=1}^{n} (y_i - \mathbf{x}_i^t \beta_{\lambda,(i)})^2 \tag{3.14}$$

where $\mathbf{x}_i$ is the ith observation of X, and $\beta_{\lambda,(i)}$ is the coefficients for a given choice of regularization parameter with $\mathbf{x}_i$ left out of the dataset, or in matrix notation

$$PRESS_\lambda = ||\mathbf{y} - \widetilde{\mathbf{y}}_\lambda||^2 \tag{3.15}$$

where $\widetilde{\mathbf{y}}_\lambda$ is the cross-validated y values for regularization parameter value λ (Krishnan et al., 2011). It is also possible to use larger hold-out sets, but in this case the aforementioned Leave-one-out Cross Validation (LooCV) was used. Fig. 17 shows a plot of the PRESS values associated with each regularization parameter or number of components (for PCR and PLS). While the outlined approach is intuitive, it requires fitting exactly n models for a dataset with n observations. Given that this has to be performed separately for each choice of regularization parameter, the total tally can quickly get untenably high. Say we search for a suitable choice with a comb of 1000 candidate values. With 100 observations, 100000 models have to be fitted, each potentially containing tens or hundreds of variables. Fortunately this is easily avoided, by using the leverage values of the observations to modify the residual vectors. The leverage of an observation is a measure of the influence that it exerts on the fit of the model (Johnson & Wichern, 2007; Burger, 2006), and the leverages can be found on the diagonal of the matrix

$$H = X(X^t X)^{-1} X$$

so that $h_{j,j}$ is the leverage associated with the jth observation. This can be used to efficiently get the LooCV residuals. If the residual vector

$$\mathbf{r} = \mathbf{y} - X\boldsymbol{\beta}$$

then the LooCV residual for observation j is

$$r_j^* = \frac{r_j}{1 - h_j}$$

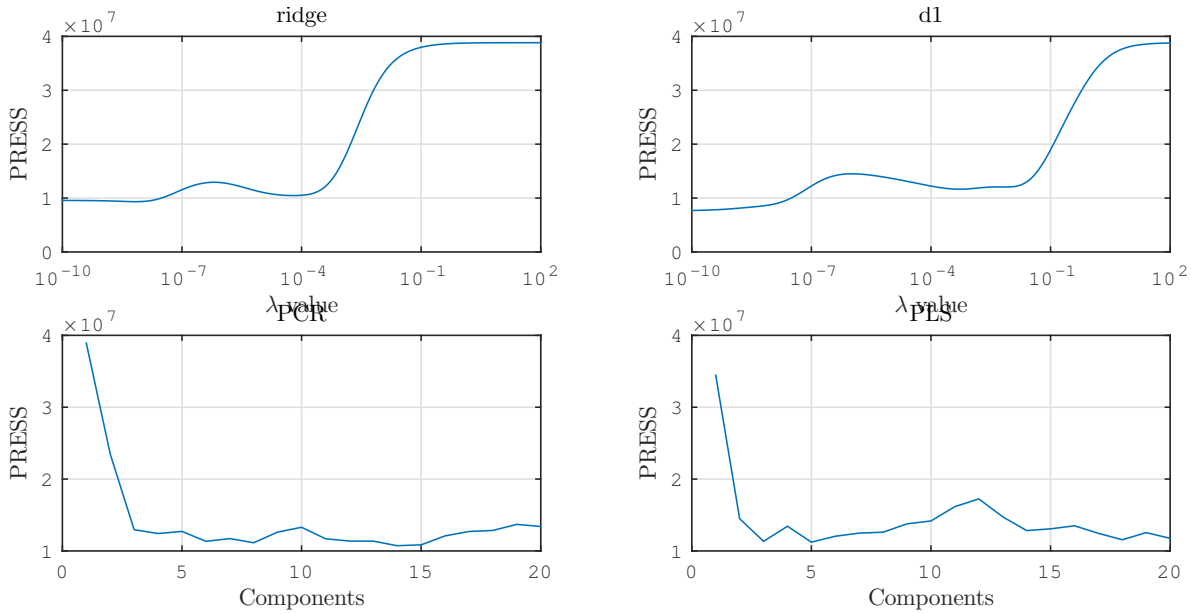


Figure 17: PRESS values for the four different methods used. The data is the latewood spectra of the weather-exposed samples.

all of which are obtained by element-wise division of the two vectors $\mathbf{r}$ and $\mathbf{1} - diag(H)$ (Indahl, 2015). For the ridge regression implementation used here, the leverage vector is obtained with a method developed by Indahl (2015) by squaring and summing row-wise the elements of the updated left singular vectors

$$U_{r,\lambda} = U_r S_r S_{r,\lambda}^{-1}$$

The elements of the vector must then be updated by adding $1/n$, where n is the number of observations, to work correctly for the uncentered version of the data. The implemented code used here can be found in Appendix C.3.1.

There are applications of Tikhonov regularization that use matrices other than $F = I$, as is the case in ridge regression, treated by Stout & Kalivas (2006). Two of these methods were applied to the data, and what they do is to replace the identity matrix with an operator similar to those mentioned in the section on Savitzky-Golay derivatives. The operators are

$$\begin{bmatrix} -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 1 \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}$$

for first derivatives and

$$\begin{bmatrix} 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{(n-2)\times n}$$

for the second derivatives version. This has the effect of optimizing with respect to the first and second derivatives of the coefficient vector, respectively, but otherwise they do the same thing as ridge regression. Appendix C.3.2 contains the code used throughout this work.

3.2.3 Partial Least Squares

Like PCR, PLS was used for comparison, and are treated here with equal brevity. A more thorough treatment can be found in Wold et al. (2001) and Krishnan et al. (2011), and only the basic principle is described here.

Like the regularization methods already described, one of the strengths of PLS lies in its ability to handle data that are noisy, underdetermined, and highly collinear (Wold et al., 2001). It has much in common with PCR, but it has the added feature of letting the right-hand side of the equation $A\mathbf{x} = \mathbf{y}$ influence the choice of basis vectors (Eldén, 2007). It is an iterative algorithm that models the relationship between the predictors and response variables while also modeling the structure in the predictor matrix, in the form of *latent variables* each describing a unique aspect of the variance of X. For each iteration the variance accounted for by the latent variables up till that point are subtracted from the matrix X and the response vector $\mathbf{y}$, and the process are repeated until the desired number of components are calculated or until X is completely decomposed (r components for a rank r matrix) (Krishnan et al., 2011). The weights, scores and loadings are stored in matrices W (weights), T (scores), P (projection loadings) and $\mathbf{q}$ (regression coefficients, a vector as long as only a single response variable are being modeled), where the columns of $W^tW = I$ spans the row space of X and the columns of $T^tT = I$ spans the column space of X (Indahl, 2015). P are the projection loadings of X on T, and $\mathbf{q}$ are the regression coefficients of $\mathbf{y}$ wrt. T. The regression coefficients of $\mathbf{y}$ wrt. X are obtained by (Indahl, 2015)

$$\boldsymbol{\beta} = W(P^tW)^{-1}\mathbf{q} \tag{3.16}$$

LooCV for PLS is limited to the explicit formulation of n models for n observations, as describe on page 19. An efficient implementation is described by McWilliams & Montana (2010), but it is not used here. Code implementation can be found in Appendix C.3.4

3.2.4 Principal Component Regression

Data analysis using PCR was used for comparison, and are only touched on briefly. Full description of the theory and use of the method can be found in Eldén (2007) and Hastie et al. (2008), but the equation used for finding the PCR β coefficients are given in Eqn. 3.7. For a rank k approximation, the matrices used are $V = [\mathbf{v}_1, ..., \mathbf{v}_k]$, $S = \sigma_1, ..., \sigma_k$ (in a $k \times k$ diagonal matrix) and $U = [\mathbf{u}_1, ..., \mathbf{u}_k]$. As with PLS, when doing LooCV for PCR, the leverages cannot be calculated that easily, and so the cross-validation is done using "brute force", as describe on page 19. Refer to Appendix C.3.3 to inspect the code used.

The scripts for performing all four statistical analyses, possibly with partitioning into training- and test sets, are supplied in Appendix A.1.1.

4 Results

Settling on a model using the mean spectra of the samples as predictors, and total measured UV radiation exposure as the response vector, the results were somewhat varied for the different datasets. A total of five data sets were analyzed:

- Ground wood sample spectra
- Weather exposed earlywood

- Weather exposed latewood
- UV-chamber exposed earlywood
- UV-chamber exposed latewood

For each data set, comparisons were made for four different methods:

- Ridge regression
- Tikhonov regularization wrt. 1st derivatives
- Principal Component Regression (PCR)
- Partial Least Squares (PLS)

Additionally, all analyses of the weather exposed samples were performed with and without the removal of "suspicious" the data. The reason for this is covered in the discussion section, but in summary, some of the observations had commonalities that justified performing analyses with those observations removed.

The prediction results for most sets is presented here, but additional results and diagnostic figures are included in Appendix A. Also note that results for regularized regression using the second-derivative operator are not included, the reason being that for this data set, it failed to produce even remotely interesting results (unless this failure in itself is interesting).

4.1 Ground wood samples

For the analysis of the ground wood samples, 150 data points were used to fit the model, which was then validated by predicting the remaining 49. The results are shown in Fig. 18. All four methods generate comparable results, with ridge regression faring slightly better at $\mathbf{R^2} = .85$, compared to $\mathbf{R^2} = .84$ for the first-derivatives version and PCR, with PLS showing the worst performance at $\mathbf{R^2} = .83$. The RMSEV (Root Mean Square Error of Validation) are .194, .200, .200 and .205, in the same order. Additional diagnostic plots are available in Appendix A.3

4.2 Weather exposed

All four methods achieved some fairly good results, but ridge- and first-derivatives Tikhonov regression generally performed better, if not by much. The results in terms of R^2 are summarized in Table 1.

Inspecting the coefficient vectors for the models with and without exclusion of the "outlier" samples, there is a dramatic and interesting difference. Fig. 19 shows the coefficients for the model with all samples included. It has large and frequent fluctuations, yielding vectors with large norms, which might imply an overfitted model. Compare this with Fig. 21, which is for models with the outlier samples removed. The curves are much smoother. Looking at Figs. (19, 21), it is evident that the overfitted model leads to better predictions for the calibration dataset, but comparing this to Figs. (23, 24) the overfitting is made apparent by the superior performance of the model with the "smooth" coefficient vector when it comes to predicting for new observations.

To check the validity of the model, the dataset was split in two, with 2/3 of the dataset assigned as training data en the remaining 1/3 as test data. The observations were assigned at random,

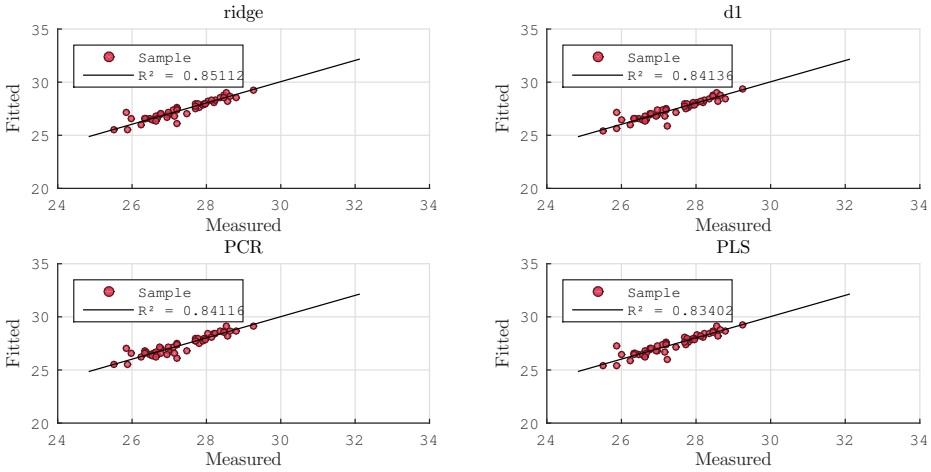


Figure 18: Predictions of lignin content in ground wood samples. The predicted datapoints were not used for fitting the model.

and the model fitted to the training set were used to predict for the test set. The results are shown in Fig. 25, with first derivatives regularization consistently offering the best performance.

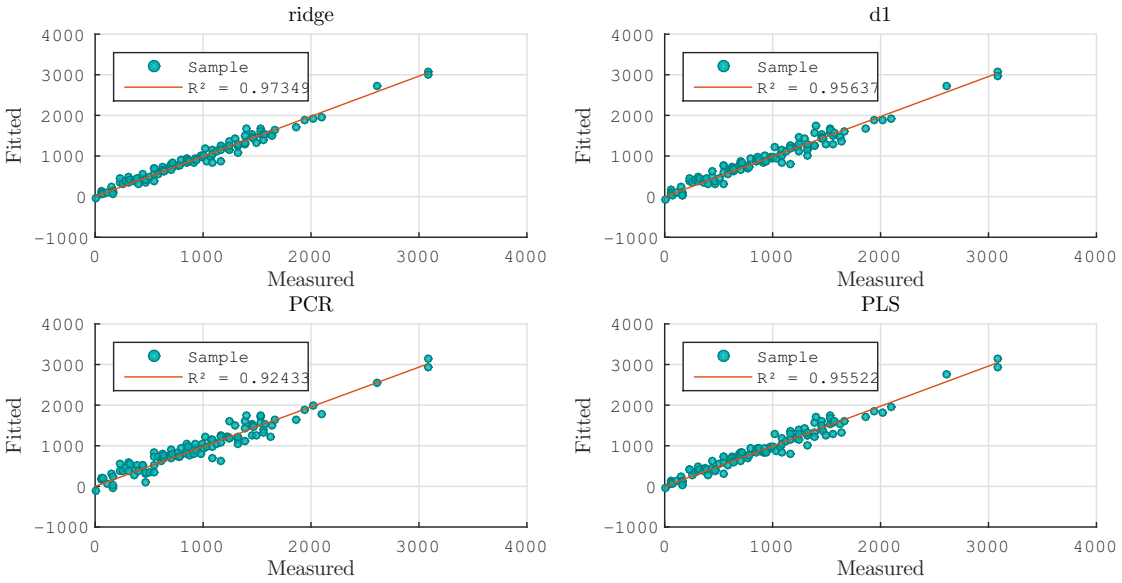


Figure 19: Predictions of UV-exposure on the earlywood spectra from the samples left outside. All of the sample points were used to fit the model.

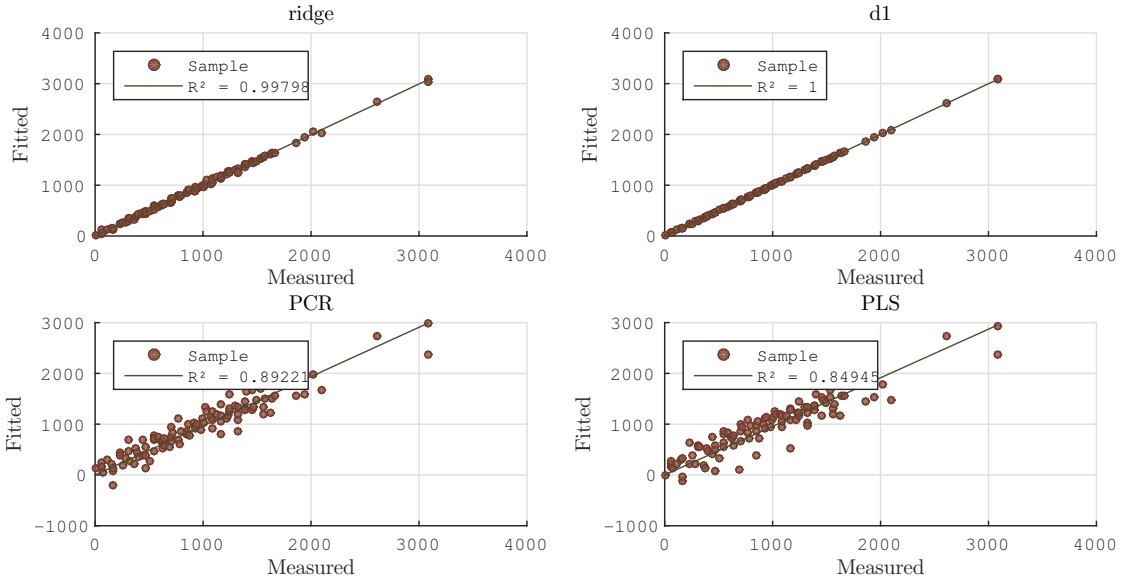


Figure 20: Predictions of UV-exposure on the latewood spectra from the samples left outside. All samples used to make to fit the model.

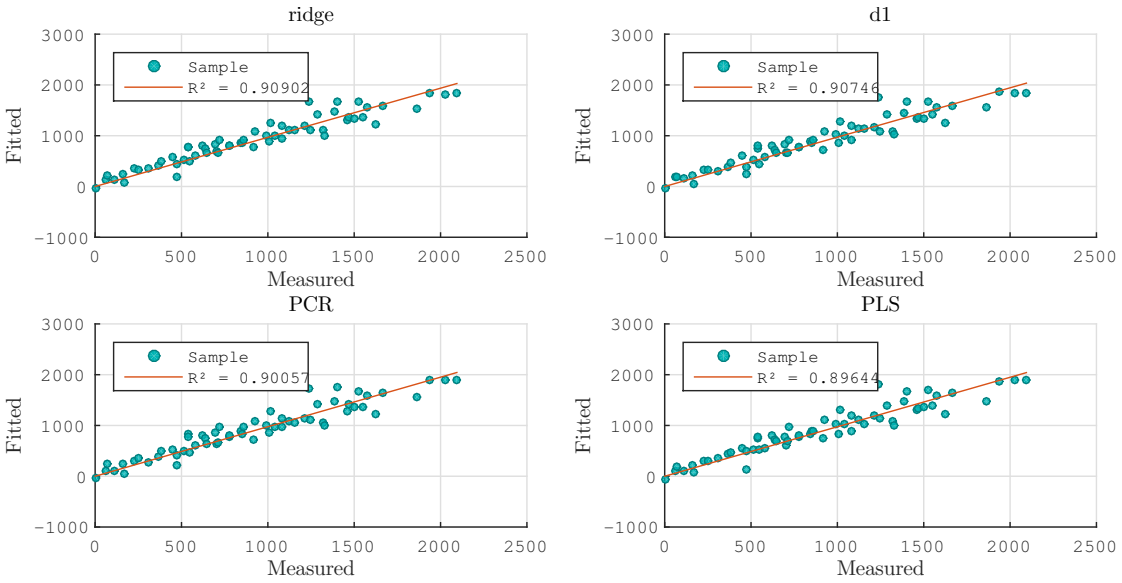


Figure 21: Predictions of UV-exposure on the earlywood spectra from samples left outside. Here the anomalous observations were removed before the model was fitted.

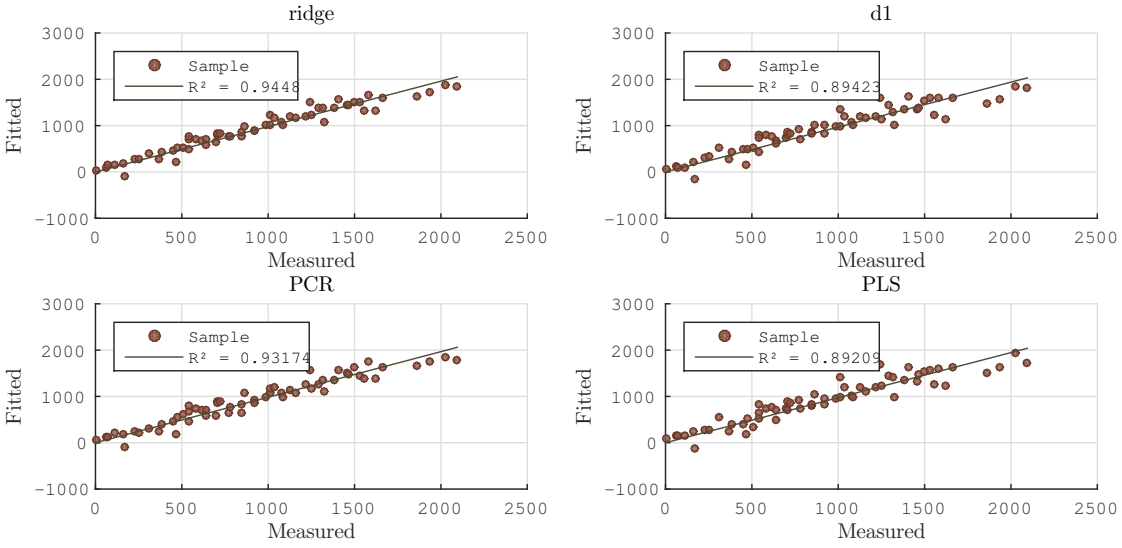


Figure 22: Predictions of UV-exposure on the latewood spectra from the samples left outside. All samples was used to fit the model, after exclusion of the "anomalous" samples.

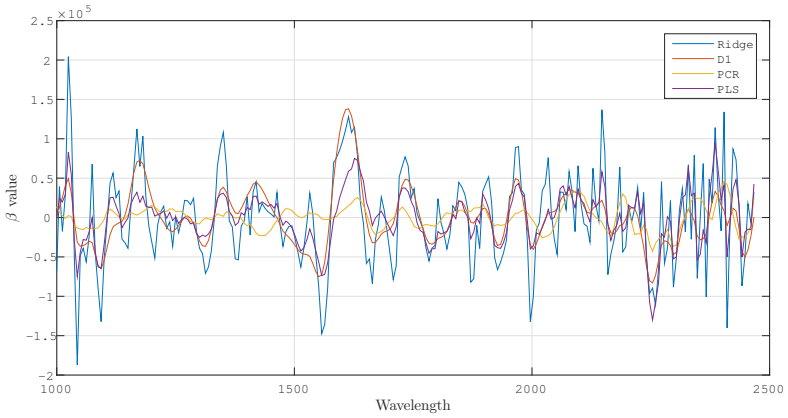


Figure 23: Coefficient vectors for the model with all samples used for training, with the anomalous observations included.

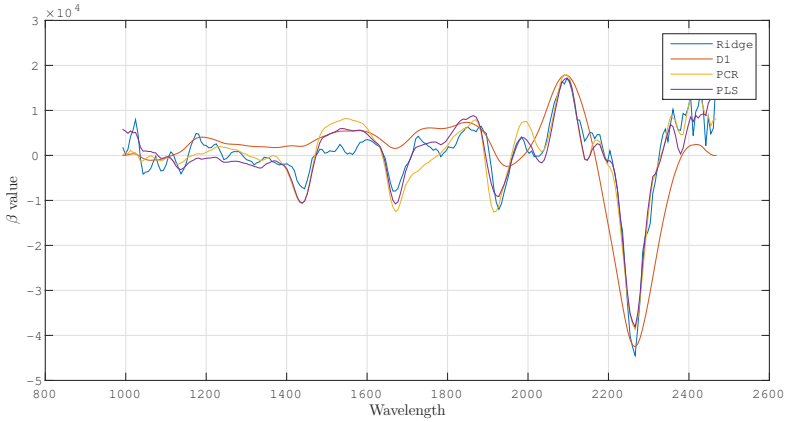


Figure 24: Coefficient vectors for the model with all samples used for training, anomalous observations excluded.

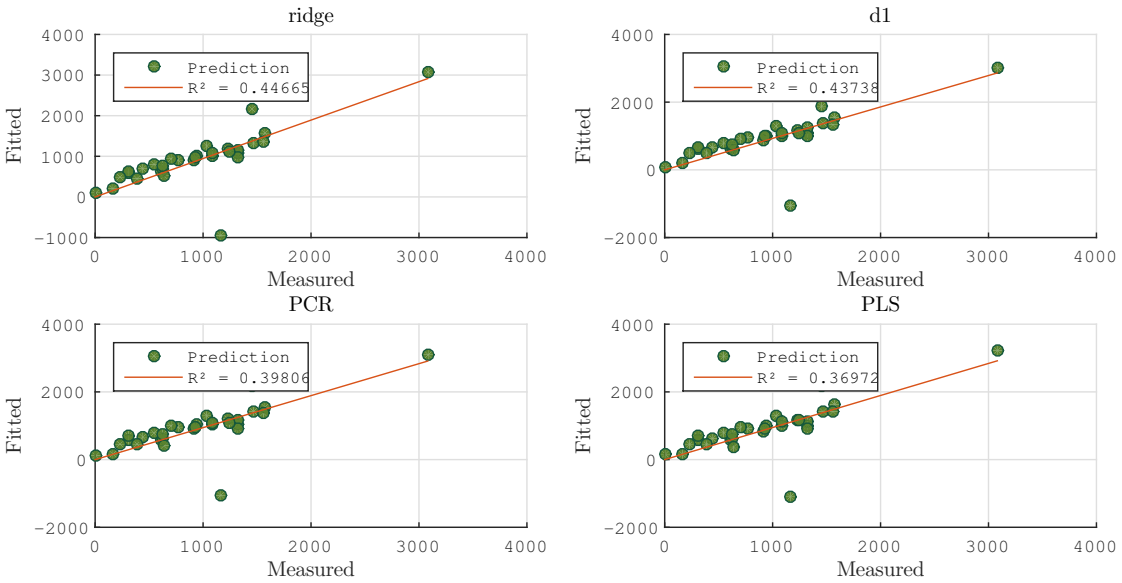


Figure 25: Predictions of UV-exposure on the earlywood spectra from the samples left outside. The data was partitioned in training and test sets, with 70% and 30% en each set, respectively.

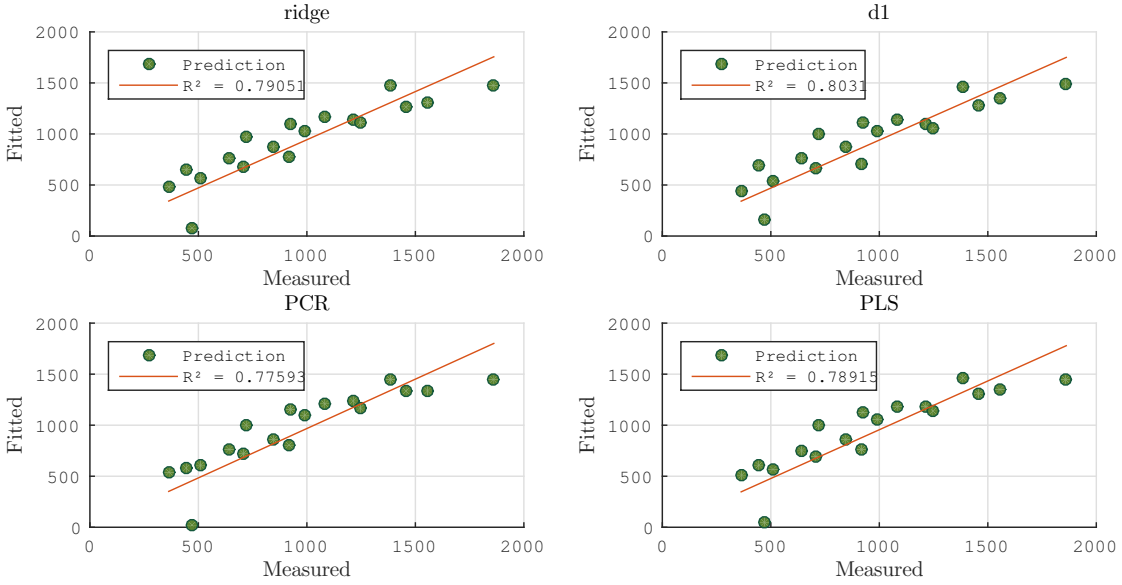


Figure 26: Predictions of UV-exposure on the earlywood spectra from the samples left outside. The data was partitioned in training and test sets, with 70% and 30% en each set, respectively, but after the anomalous observations were removed

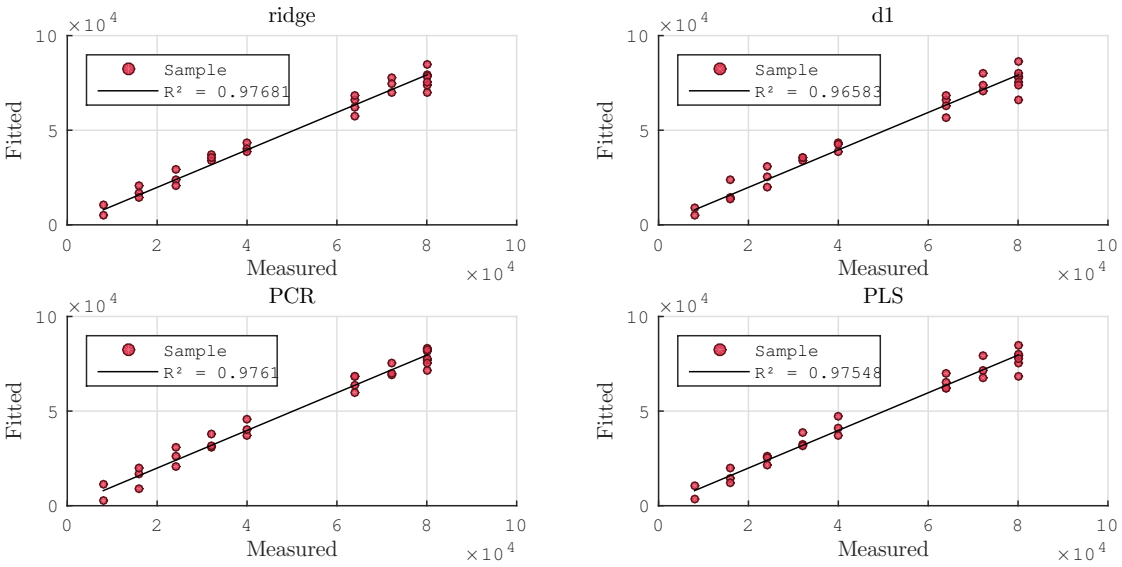


Figure 27: Predictions of earlywood UV exposure for samples in UV-chamber. All data points used for model fitting.

4.3 UV chamber

The laboratory samples did not contain any anomalous spectra, but it also contained fewer samples. Results for the different methods are shown in Figs. (27, 28) for early- and latewood, respectively. Once again ridge regression has the best performance ($R^2 = 0.97$ and $R^2 = 0.96$), but for this data it is only negligibly better than PCR or PLS. As expected, the results when using a model fitted to the weather exposed samples to predict for the laboratory samples did not produce very good results, as it may in fact be like the proverbial comparison of apples and oranges. That is, there might not be grounds for assuming that the samples exposed to the natural weather and those exposed to artificial UV radiation are directly comparable, since there are so many influencing factors, which has already been mentioned. The results are shown in Fig.29, where the ridge regression model has been used for prediction.

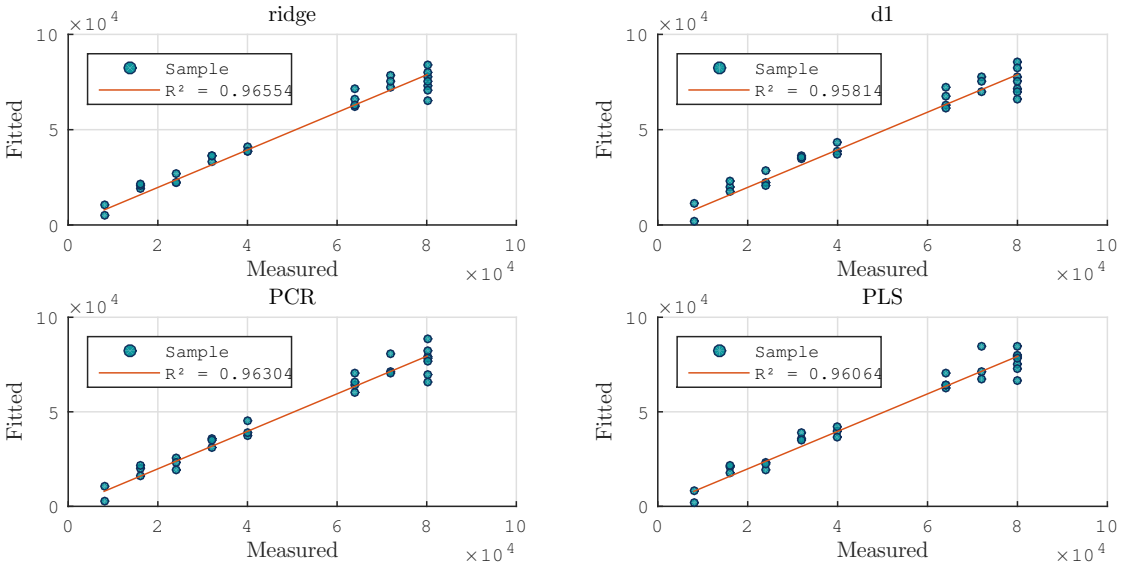


Figure 28: Predictions of latewood UV exposure for samples in UV-chamber. All data points used for model fitting.

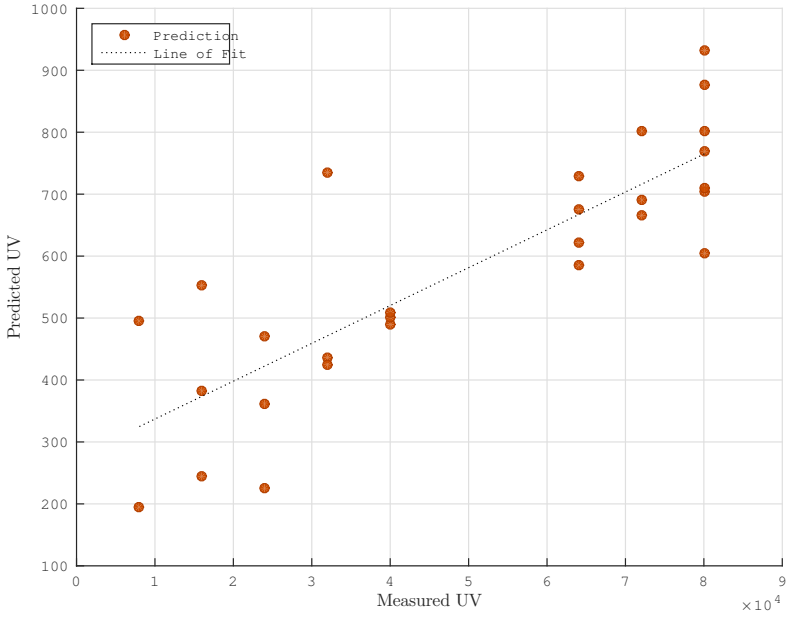


Figure 29: Predictions of UV exposure on samples in UV-chamber using model fitted to weather exposed samples.

Table 1: Coefficient of determination values for the models

Dataset	R^2			
	ridge	d1	PCR	PLS
Weathered, early	0.975	0.956	0.924	0.955
Weathered, early, cleaned	0.909	0.907	0.900	0.896
Weathered, late	0.997	$\sim$1	0.892	0.849
Weathered, late, cleaned	0.945	0.894	0.932	0.892
Weathered, early, validation	0.446	0.437	0.398	0.370
Weathered, early, validation, cleaned	0.790	0.803	0.776	0.789
Weathered, late, validation	0.447	0.437	0.398	0.370
Weathered, late, validation, cleaned	0.816	0.772	0.809	0.710
Lab, early	0.976	0.965	0.976	0.975
Lab, late	0.965	0.958	0.963	0.960
Lab, early, validation	0.855	0.853	0.807	0.857
Lab, late, validation	0.816	0.846	0.810	0.807
Ground samples	0.851	0.841	0.841	0.834

4.4 Lignin content

In addition to relating spectral data to UV-exposure, a simple linear regression was performed for total UV versus lignin and holocellulose. The results for lignin are shown in Figs. (30, 31). The correlation coefficients are .71 and .82 for the weather- and UV-chamber exposed samples, respectively.

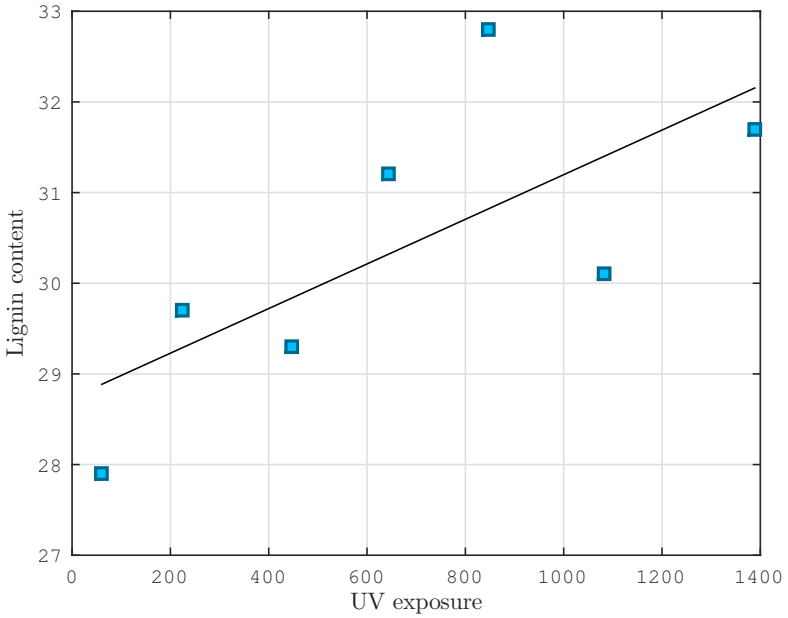


Figure 30: Measured lignin content versus measured UV exposure for weather-exposed samples (earlywood).

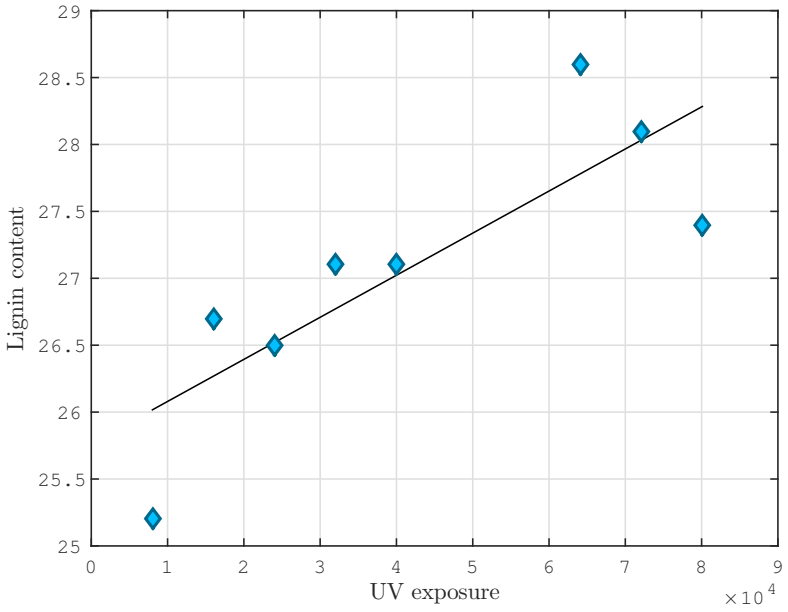


Figure 31: Measured lignin content versus measured UV exposure for UV-chamber exposed samples.

5 Discussion

5.1 Data preparation

Working with, and analyzing the material has in a very strong sense been an iterative process, where several different approaches have been tried and a quite a few hurdles has had to be overcome, some of which I have touched on already. I will try to present these approaches and their problems and solutions in the order in which they've been met and addressed, to hopefully lend meaning to the choices that have been made.

The main idea has been twofold: to explore the degree to which weathering in wood can be analyzed by the use of hyperspectral imaging, and to test a somewhat novel statistical procedure and its performance in terms of predictive power, and how it compares to other available, perhaps more widely used, methods.

The first, although minor, problem was the sorting of the images. The function that converts the raw files to MatLab matrices reads them in the order they are sorted by the operating system, which is ASCII. This means that files with numbers 1 - 30, as an example, will be sorted as $1, 10, 11, ...19, 2, 20, 21, ...29, 3...30$. The observations in the response vector were sorted numerically according to the sample number. While matching these proved in the end not to be complicated, it introduces uncertainty because it is hard to directly verify that the spectra you are looking at actually belong to the sample you think it does. Unless you can visually recognize each image, it can only be checked indirectly by observing how the sorting behaves on a dummy vector where you know what to expect. Even though the sorting proved correct on the first go, and that the problem might seem trivial, I was led to question it every time something unexpected cropped up somewhere in the process.

Fig. 33 represent the work flow from raw images to final results from the analyses, in its idealized form. In reality, there are a lot of invisible arrows in that chart, because noticing an error and finding its root cause is in large part guided by intuition, exploration and plain trial-and-error.

The basic steps in the pre-processing of the images were settled upon relatively early. It was clear that the samples should be separated in two segments, for which the masking function already described was implemented. The use of SG filtering and EMSC were also implemented from the start, as they are pretty standard methods. The spike removal was not included from the start, because the need to use it was not immediately noticeable. Since there are so many pixels from which each average spectrum is calculated, the effects of a few spikes did not make themselves apparent. However, the first few iterations of the whole process were performed on spectra that had not been transformed from absorbency to transmission. As soon as this was corrected, the problem with spikes made itself known: since some of the spikes were in the form of wavelengths with zero-values, performing a $-\log_{10}$ transform destroyed the data, because $-\log_{10}(0) = \infty$, as mentioned in the Smoothing section, along with the solution that was used. On a side note, before implementing the spike removal algorithm, the problem was "solved" by setting all infinite-valued elements of the data matrix to zero, which yielded acceptable results, presumably because only about $1/10^4$ of the values were modified in this way, thus exerting little influence of the final averaging.

Of central importance was the differentiation of late- and earlywood in their response to UV radiation. That is, in addition to benchmarking the efficacy of hyperspectral imaging as a quantifier for light-induced weathering in general, it was suspected that recognizable differences could be found between responses in the two tissue types. The reasoning behind this is described

in the pre-processing section on p. 5, and the ability to successfully investigate these differences rests on the ability of the segmentation algorithm to reliably separate the two. After programming the function used for the segmentation, it was run on the samples with several choices for cut-off values and score thresholds, to find an optimal choice. The function can output masks for both early- and latewood in a single run, and to save on processing time this was done, with seemingly good results.

While developing the complete procedure, all the tests were performed on earlywood data, for good effect. However, when extending the procedure to the late wood samples and inspecting the spectra, there were some obvious deviations. Some of the mean spectra stood out as wildly different from the others, which suggested either a difference in the sample itself, or an error in the segmentation. On closer scrutiny it was found that the approach of using the same cut-off and score threshold values for early- and latewood was not sound, and that the two masks had to be found separately. Specifically, the erroneous mean spectra was a result of some images which contained a lot of background, and some high intensity pixels that were not excluded with the original values. In addition to unwanted pixels being included in the first place, the effects were magnified when averaging the spectra for the image, because extreme values exerted a much stronger influence. This was due to the sample size being much smaller (in some of the images, fully half of it was background, and thus excluded from the calculation), and was solved by using a more aggressive cut-off for the latewood mask, and a slight adjustment to score thresholds.

After this weakness in the segmentation had been sorted out, there were still some "strangeness" in the spectra that were unaccounted for, and whose cause would prove much harder to identify. Looking at Fig. 10, there is a marked split of the spectra into two broad bands at around 1800 nm. This is true for both the early- and latewood spectra, although less pronounced in the latter. At first it was believed that this was due to further, more subtle limitations in the masking algorithm, which was ruled out by testing it on other datasets. The fact that the phenomenon is not evident in the samples from the UV-chamber left two possibilities: the split was an artifact of some part of the pre-processing, or it was a feature of the samples themselves. The former was quickly ruled out by examining the results when either SG filtering or EMSC were left out, and finding that the pattern persisted. Inferring from this that the explanation lay with the physical samples themselves, the problem became figuring out what set the particular samples apart from the rest. I began by compiling all the information I had on the samples; sample number, number of days exposure, lignin measurements, total UV radiation exposure, sample group (samples were placed outside in one of four groups), etc. I then extracted the numbers of the samples that lay in the upper and in the lower of the two bands, and used this as an index vector to sort the observations into two groups, hoping to find a common denominator for each. At first this yielded no useful results, as there were no recognizable pattern. There were representatives from all groups in both bands, spanning the whole spectrum of day/UV exposure.

The breakthrough came when adding a last piece of information. The samples had been kept in eight different envelopes during storage. When recording the images, I had taken out the envelope, recorded, and put the samples back in the envelope. When the images were saved to the computer, I put the files in named folders corresponding to the envelopes, thinking that even though unlikely, it might prove relevant at some point. Adding to the compiled data this last piece of information, the pattern became immediately apparent; all of the samples that stood out were from one of two envelopes. One contained almost one third of all the samples, and the other just a few. In total, 43 of the 104 samples were stored in these two envelopes, which explained why so many of the samples deviated from the norm.

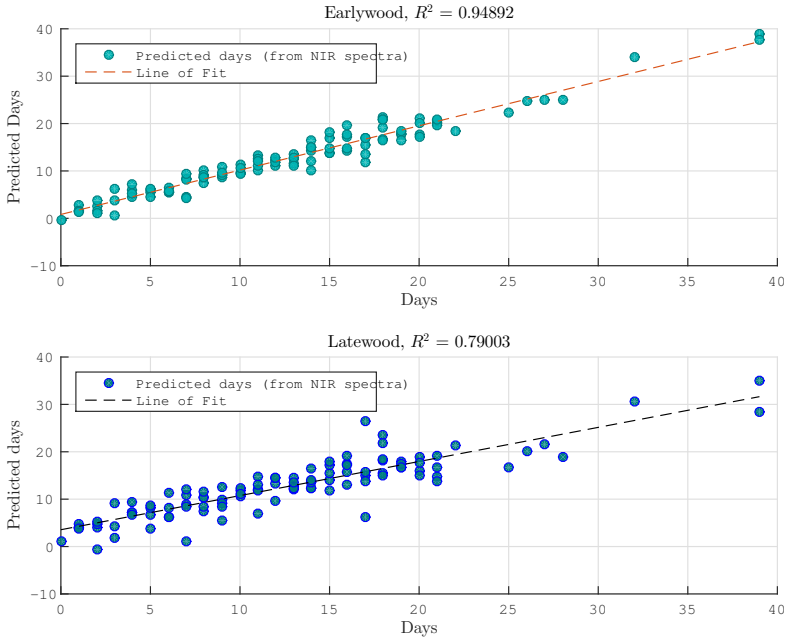


Figure 32: Predicted days of exposure for weather exposed samples, using ridge regression.

Although it was too late to correct the problem, and difficult to hypothesize what in particular had caused the changes in the samples, knowing that the observed anomalies were due to methodological error and not a part of the phenomenon being studied is important, because it allows us to focus on the data that can be compared on equal footing, and adjusts our expectations about how well the data can be fitted in a model. It also highlights the importance of careful handling of the samples; the two envelopes were not exposed to radically different treatments, so the fact that it produced such notable differences says something about the sensitivity of the samples and the importance of diligent methodology.

5.2 Analysis

I will now move on from the pre-processing of the data to the statistical treatment of it. As was stated earlier, choosing what to use as the actual response vector was not a given. My first approach was to use the number of days of exposure as the response, since at the time that was all the information I had. The lignin measurements and the weather data used to calculate UV radiation exposure were not available initially. While using days of exposure as the response variable did produce some acceptable results, they were not very accurate because there is a lot of variation between days when it comes to things like UV radiation, temperature and moisture, those being the main factors behind the weathering effect. Even though variation in temperature and moisture were not accounted for in the final model, at least a large part of the weathering effect was captured by regressing on radiation totals instead of days, as can be observed by comparing Fig. 19 with the upper plot in Fig. 32 . As an alternative to using days as the response variable, fitting a model to spectra from all the pixels in the least and most exposed sample was attempted, as described earlier, to see if it was possible to obtain good predictions of the intermediate samples with the fitted model. While this was more of an

exploratory exercise than a final goal, it was abandoned reasonably early because it failed to produce meaningful results. While this idea did not get any more attention in this case, I do think the approach of using more than just the mean spectra for each sample has some merit, as it would be interesting to see if it is possible to fit models that can predict exposures not just for an image, but for regions within an image. This in turn would rely on being able to obtain spatially accurate measurements of a response variable, be it lignin content, radiation exposure or some other meaningful, measurable quantity. Since it is pretty clear that it would be impossible to get measurements from a sample corresponding to each pixel, averaging measurements from within a region of a sample and matching it to the mean spectrum for the corresponding region seems like a better alternative. While averaging individual spectra within regions of an image is not fundamentally different from averaging all spectra from each image, it could perhaps be useful for exploring how applicable models based on mean spectra are on those from single pixels. For example it would be interesting to try to average temporally instead of spatially; recording several images of the same sample, and obtaining averages for each pixel over the repeat measurements. If much of the noise could be canceled out by a moderate number of such repetitions, then the only limitation would be the spatial accuracy of the measurements of the response variable. Of course, this is given a perfect alignment between pixels in each repeat, which becomes a problem or maybe an impossibility in settings other than the laboratory.

So the decision finally fell on using the radiation measurements of the area as the response variable. The data came in the form of a spreadsheet with different measurements from the period, recorded each hour. This was transformed to daily totals by reshaping the desired response vector (UV), so that each column corresponded to one day, and summing the columns of the resulting matrix. The next step was calculating the cumulative sums for each sample. For a full sequence of samples exposed for 1-21 days, this was calculated by the matrix multiplication $A\mathbf{d}$, where A is a 21×21 lower triangular matrix of ones, and $\mathbf{d}$ is a vector with the daily totals for the corresponding days. This then had to be modified to account for incomplete sequences and observations not fitting the basic pattern in general, like those samples exposed for 39 days.

When it comes to the actual predictions, referring to Table 1 it is gratifying to get such good results, but humility and caution is in order, as there are some caveats to keep in mind. First, the predictions in Figs. (19,27, 20) are predictions for the data that was used to fit the model, and as such has limited credibility as an indicator of the general applicability of the model in predicting future samples. There is also a big question mark attached to the first two plots in Fig. 20. These predictions were performed on the dataset including the anomalous spectra mentioned earlier. The effect of this seems to be that the algorithm converges to an overfitted model, actually one that exactly predicts the training data. When the same data was partitioned into training- and test samples, the performance was severely impacted, with most CoD about 0.5. When these observations were removed from the sample before partitioning, the results were the ones shown in Fig. 22.

A recurring pattern appears to be that the ridge regression performs better on the data on which it was trained, while the first derivatives regularization outshines ridge regression when predicting external (test) data. Looking at the curves of the coefficient vectors, this is perhaps not surprising, as the ridge regression coefficients are allowed to fluctuate more and thus better fit the data, while the first derivatives regularization coefficient vector is much smoother. This enforced smoothness inhibits its ability to fit the training data (Johansen, 1997), but this is made up for by its decreased sensitivity to small variation in the measurements, which is what allows it to predict so well for new observations.

Regarding this, it is important to note that since the assignment of a sample to either the

training or the test set is randomized, the fit will be different each time. During the work with the data, the scripts were run several times, and the $\mathbf{R^2}$ values varied greatly (in the range 0.6 to 0.95), indicating that the model is sensitive to outliers. This is not surprising, because after removing 43 of the 104 weather exposed samples, and then fitting the model with only one third of the remaining ones, only 40 observations are left in the training sample. It would appear that this is pushing it, and underscores that the methods being used are not magic wands that can produce meaningful results out of any data set, but are in fact subject to the same limitations as classical statistical methods, if to a lesser degree. To get a more representative description of the performance of the models, the partitioning of the data should be performed several times, with subsequent fitting of the models, and the $\mathbf{R^2}$ values recorded and averaged. With enough runs, the resulting averages would better reflect the validity of the model.

While the results from the methods used here are heartening, there are related methods that could potentially perform better, described by Kalivas (2012). Two of these are the LASSO and the elastic net. I will not go into mathematical detail, but hint at how they could be used.

The Least Absolute Shrinkage and Selection Operator (LASSO) is a form of TR that optimizes the bias/variance trade off by putting a penalty on the 1-norm (Manhattan distance) of the coefficient vector instead of the 2-norm (Euclidean distance) used in ridge regression

$$\min(||X\boldsymbol{\beta} - \mathbf{y}||_2^2 + \eta||\boldsymbol{\beta}||_1) \tag{5.1}$$

This has the effect of shrinking coefficients to zero or nearly zero much quicker, in effect performing variable selection. As such, the coefficient vector will be sparse, with mostly zero values and some non-zero values corresponding to significant wavelengths (Kalivas, 2012). In earlier studies by Ottaway et al. (2010) and Stout, Kalivas & Héberger (2007), this was found to produce lower prediction errors compared to models using all wavelengths. Relating this to the dataset at hand (weather-exposed samples), and remembering that the anomalous observations mainly deviated in a relatively small subset of the wavelengths, one could hypothesize that such a variable selection procedure would automatically reduce the associated model coefficients to (near) zero, thus negating their impact. Unfortunately, exploring this possibility will have to be left to further studies.

The elastic net puts a penalty on both the 1-norm and the 2-norm of the coefficient vector, using different weights for each. The equation then becomes

$$\min(||X\boldsymbol{\beta} - \mathbf{y}|| + \eta||\boldsymbol{\beta}||_1 + \lambda||\boldsymbol{\beta}||_2^2) \tag{5.2}$$

According to Kalivas (2012), this performs well in calibrating the model to new data. Say we have a model fitted to some data, under a certain set of conditions. Suppose we take new measurements, under new conditions, with a new instrument or perhaps both. In that case it would be useful to be able to update the previous model to take into account these new conditions, to maintain predictive power. The updating of models and the various methods to achieve this is beyond this study, but would have been interesting for further development. Various methods are treated by Feudale et al. (2002), and ways to use TR for model updating are discussed in Kalivas (2012), along with generalizations where alternative matrix operators are used instead of the identity matrix.

Referring to Figs. (30, 31), a trend where the lignin content is positively correlated with UV exposure is evident, while the opposite is true for holocellulose (not shown). At first glance this runs counter to what was anticipated, as it was already stated that UV radiation is known to degrade lignin. While it is difficult to ascertain the reason for this contradiction, it appears that

the precise mechanics are very complex (Williams, 2005), and the essentials of what is known is that UV radiation leads to the formation of free radicals on the surface. Williams points to unknowns relating to factors such as wavelength dependence, the interactions with water and oxygen, along with others. One possibility that was suggested is that the radiation causes the lignin to deposit on the surface of the wood before it begins to break down, while it is normally stored in the middle lamella of the wood (Williams, 2005). This might cause an apparent increase in lignin content, as more of it is picked up by the camera. I am somewhat skeptical of this. Since transmission spectroscopy of very thin samples was used, it follows that the light that reaches the sensor has been through the whole sample, and as such I suspect it would not matter how deep in the sample the lignin was stored. The other possible explanation is that the UV exposure, causing the lignin (or derivatives thereof) to deposit on the surface, makes it easier for the STA-FTIR instrument to pick up. Again, while I am not familiar with the details of the instrument, I would suspect that given the small thickness of the samples and the fact that they were ground, all of their content would be picked up by this instrument as well. However, I am not able to provide a better explanation, and so will have to stick with this for now. I will note however that while lignin content appears to be positively correlated with UV exposure when looking at the weather-exposed and UV-chamber samples separately, the opposite is true when they are compared. The samples from the UV chamber have been exposed to far more radiation, and also have a significantly lower lignin content, with $p < 0.001$ on a two-sample t-test. It also has much lower standard deviation, with 1.03 for the UV-chamber samples and 1.65 for the weather-exposed ones. This corroborates what has already been said about the large number of interfering factors in the outside environment.

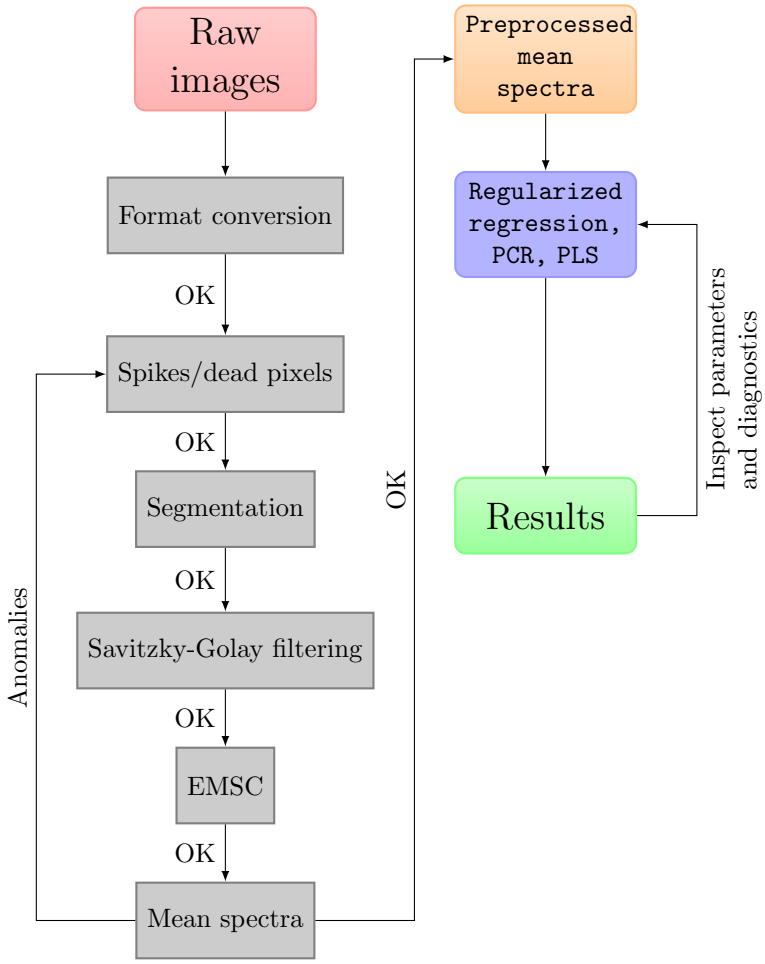


Figure 33: General work flow

6 Conclusion

The effects on wood of the weather in general, and solar radiation in particular, is complex and leaves much to be explored. Using hyperspectral imaging along with judicious use of preprocessing and statistical/data mining techniques that can deal with the challenges posed by such datasets, namely multicollinearity, underdeterminacy and noise, appears to be a worthwhile pursuit because it is quick, cheap and non-intrusive. Still, additional qualifiers are called for, because the amount of interfering phenomena and their varying intensities undermines the predictive potential of any fitted model to a significant degree. As such, it is not the performance of the model in handling the data that is the limiting factor, but the lack of a suitable response variable. Successfully decoupling the various factors that constitute the weathering effect, so that they could be studied separately, would perhaps be a step in the right direction.

It also appears that if the data used to train the ridge regression model has enough significant outliers, the algorithm can be forced to choose an overfitted model. While this problem could

be addressed and solved by removing the observations from the data set, this severely reduces the available data for fitting the model. Hence, if it is the case that only some wavelengths are affected by, say, variations in the humidity during storage while the other wavelengths maintain their predictive power, it might be advantageous to use methods that can perform variable selection and automatically choose the most reliable variables (wavelengths), perhaps one of the ones mentioned (LASSO or elastic net), so that data that has been subject to disturbances like this will not have to be thrown away.

Finally, the work has been highly multidisciplinary, and this has brought to light the need for a clearer, more concise language in this area. Having quantifiable measurements for use in mathematical modeling will be an advantage, as opposed to qualitative descriptions, perhaps encoded as numbers, that are subjective. This will make both the input and output of data between institutions and research groups more communicable, and will add mathematical rigor to a field that might profit from it.

References

Aasheim, Per A. (2012). *Bygg i tre*. Yearly report. Innovasjon Norge et al.

Alves, Ana, Helena Pereira, Mannfred Scwanninger & José Rodrigues (2006). "Analytical pyrolysis as a direct method to determine the lignin content in wood. Part 1: Comparison of pyrolysis lignin with Klason lignin". In: *Journal of Analytical and Applied Pyrolysis* (76), pp. 209–213.

Anderson, Edward, Zhaojun Bai, Christian Bischof, Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, S Hammerling, Alan McKenney, et al. (1999). *LAPACK Users' guide*. Vol. 9. Siam.

Bertaud, Frédérique & Bjarne Holmbom (2004). "Chemical composition of earlywood and latewood in Norway spruce heartwood, sapwood and transition zone wood". English. In: *Wood Science and Technology* 38.4, pp. 245–256. ISSN: 0043-7719.

Brennan, PJ & GR Fedor (1993). *High irradiance UV/condensation testers allow faster accelerated weathering test results*. Tech. rep. American Chemical Society, Washington, DC (United States).

Burger, James E. (2006). "Hyperspectral NIR Image Analysis". Doctoral Thesis. Swedish University of Agricultural Sciences.

Eldén, Lars (2007). *Matrix Methods in Data Mining and Pattern Recognition*. SIAM.

Esquerre, C., A.A. Gowen, J. Burger, G. Downey & C.P. O'Donnell (2012). "Suppressing sample morphology effects in near infrared spectral imaging using chemometric data pre-treatments". In: *Chemometrics and Intelligent Laboratory Systems* (177), pp. 129–137.

Feudale, Robert N, Nathaniel A Woody, Huwei Tan, Anthony J Myles, Steven D Brown & Joan Ferré (2002). "Transfer of multivariate calibration models: a review". In: *Chemometrics and Intelligent Laboratory Systems* 64.2, pp. 181–192.

Fromm, Jörg H, Irina Sautter, Dietmar Matthies, Johannes Kremer, Peter Schumacher & Carl Ganter (2001). "Xylem water content and wood density in spruce and oak trees detected by high-resolution computed tomography". In: *Plant Physiology* 127.2, pp. 416–425.

Gobakken, Lone Ross (2009). "Surface mould growth on painted and unpainted wood". Doctoral Thesis. Norwegien University of Life Sciences.

Golub, Gene H & Christian Reinsch (1970). "Singular value decomposition and least squares solutions". In: *Numerische mathematik* 14.5, pp. 403–420.

Golub, Gene H & Charles F Van Loan (1980). "An analysis of the total least squares problem". In: *SIAM Journal on Numerical Analysis* 17.6, pp. 883–893.

Grossman, Douglas M (1994). "Errors caused by using joules to time laboratory and outdoor exposure tests". In: *ASTM Special Technical Publication* 1202, pp. 68–68.

Hansen, Per Christian (1992). "Analysis of Discrete Ill-Posed Problems by Means of the L-Curve". English. In: *SIAM Review* 34.4, pp. 561–580.

Hastie, Trevor, Robert Tibsharani & Jerome Friedman (2008). *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Second edition. Springer.

Heide, Ola M. (1974). "Growth and Dormancy in Norway Spruce Ecotypes (Picea abies) I. Interaction of Photoperiod and Temperature". In: *Physiologia Plantarum* 30.1, pp. 1–12.

Hoerl, Arthur E & Robert W Kennard (1970). "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1, pp. 55–67.

Indahl, Ulf Geir (2015). "Efficient cross-validation for ridge regression. Lecture notes. Unpublished".

Johansen, Tor A (1997). "On Tikhonov regularization, bias and variance in nonlinear system identification". In: *Automatica* 33.3, pp. 441–446.

Johnson, Richard A. & Dean W. Wichern (2007). *Applied Multivariate Statistical Analysis*. Sixth Edition. Pearson Education.

Kalivas, John H. (2012). "Overview of two-norm (L2) and one-norm (L1) Tikhonov regularization variants for full wavelength or sparse spectral multivariate calibration models or maintenance". In: *Journal of Chemometrics* 26.6, pp. 218–230. ISSN: 1099-128X. DOI: 10.1002/cem.2429. URL: http://dx.doi.org/10.1002/cem.2429.

Kansal, SK, M Singh & D Sud (2008). "Studies on TiO 2/ZnO photocatalysed degradation of lignin". In: *Journal of Hazardous materials* 153.1, pp. 412–417.

Kim, Seung-Jean, Kwangmoo Koh, Michael Lustig, Stephen Boyd & Dimitry Gorinevsky (2007). "An interior-point method for large-scale l 1-regularized least squares". In: *Selected Topics in Signal Processing, IEEE Journal of* 1.4, pp. 606–617.

Kohler, Achim, M. Zimonja, V. Segtnan & H. Martens (2009). "Standard Normal Variate, Multiplicative Signal Correction and Extended Multiplicative Signal Correction Preprocessing in Biospectroscopy". In: *Comprehensive Chemometrics*. Ed. by Brown S, Tauler R & Walczak R. Vol. 2. Oxford: Elsevier, pp. 136–162.

Krishnan, Anjali, Lynne J Williams, Anthony Randal McIntosh & Hervé Abdi (2011). "Partial Least Squares (PLS) methods for neuroimaging: a tutorial and review". In: *Neuroimage* 56.2, pp. 455–475.

Lay, David C. (2012). *Linear Algebra and its Applications*. Fourth Edition. Pearson.

Li, Qingbo, Qishuo Gao & Guangjun Zhang (2013). "Improved Extended Multiplicative Scatter Correction Algorithm Applied in Blood Glucose Noninvasive Measurement with FT-IR Spectroscopy". In: *Journal of Spectroscopy* 2013.

McWilliams, Brian & Giovanni Montana (2010). "A PRESS statistic for two-block partial least squares regression". In: *Computational Intelligence (UKCI), 2010 UK Workshop on*. IEEE, pp. 1–6.

Menditto, Antonio, Marina Patriarca & Bertil Magnusson (2007). "Understanding the meaning of accuracy, trueness and precision". In: *Accreditation and quality assurance* 12.1, pp. 45–47.

Moler, Cleve (2000). *Matlab Incorporates LAPACK*. URL: http://se.mathworks.com/company/newsletters/articles/matlab-incorporates-lapack.html?refresh=true.

Müller, Uwe, Manfred Rätzsch, Manfred Schwanninger, Melanie Steiner & Harald Zöbl (2003). "Yellowing and IR-changes of spruce wood as result of UV-irradiation". In: *Journal of Photochemistry and Photobiology B: Biology* 69.2, pp. 97–105.

Ottaway, Joshua, John H Kalivas & Erik Andries (2010). "Spectral multivariate calibration with wavelength selection using variants of Tikhonov regularization". In: *Applied spectroscopy* 64.12, pp. 1388–1395.

Prasad, K Manjunatha & RB Bapat (1992). "The generalized Moore-Penrose inverse". In: *Linear Algebra and its Applications* 165, pp. 59–69.

Rifkin, Ryan M. & Ross A. Lippert (2007). *Notes on Regularized Least Squares*. Massachusetts Institute of Technology - Computer Science and Artificial Intelligence Laboratory.

Savitzky, Abraham & M. J. E. Golay (1964). "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." In: *Analytical Chemistry* 36.8, pp. 1627–1639.

Stout, Forrest & John H. Kalivas (2006). "Tikhonov regularization in standardized and general form for multivariate calibration with application towards removing unwanted spectral artifacts". In: *Journal of chemometrics* 20.1-2, pp. 22–33.

Stout, Forrest, John H. Kalivas & Károly Héberger (2007). "Wavelength selection for multivariate calibration using Tikhonov regularization". In: *Applied spectroscopy* 61.1, pp. 85–95.

Thiis, Thomas K., Ingunn Burud, Dimitrios Kraniotis & Lone R. Gobakken (2015). "The role of transient wetting on mould growth on wooden claddings". In: *6th International Building Physics Conference, Energy Procedia*.

Treu, Andreas, Lone Ross Gobakken, Erik Lamøy, Gry Alfredesen & Lars Sandved Dalen (2014). *Trebehandling - innovasjon, metoder og trender*. Ed. by Norsk institutt for skog og landskap.

Tzeng, Jengnan (2013). "Split-and-Combine Singular Value Decomposition for Large-Scale Matrix". In: *Journal of Applied Mathematics* 2013.

Westermark, U, O Lidbrandt & I Eriksson (1988). "Lignin distribution in spruce (Picea abies) determined by mercurization with SEM-EDXA technique". In: *Wood Science and Technology* 22.3, pp. 243–250.

Williams, R Sam (2005). "Weathering of Wood". In: *Handbook of wood chemistry and wood composites*, p. 139.

Wold, Svante, Michael Sjöström & Lennart Eriksson (2001). "PLS-regression: a basic tool of chemometrics". In: *Chemometrics and intelligent laboratory systems* 58.2, pp. 109–130.

Yeh, Ting-Feng, Hou-min Chang & John F Kadla (2004). "Rapid prediction of solid wood lignin content using transmittance near-infrared spectroscopy". In: *Journal of Agricultural and food chemistry* 52.6, pp. 1435–1439.

Appendices

A Results

All results and diagnostic plots are presented here. Results for the first set features captions, while the rest does not. The figures should still be recognizable, as they strongly resemble each other.

A note on the naming of the results are in order. Other than the obvious division into ground-, weather- and UV-chamber exposed samples, some are noted as validated, and some are also noted as "cleaned". The first denotes that the predicted data points were not used to fit the model. The second denotes that the observations identified as anomalous was excluded before the model was fitted. In the case of a validated model, these observations were removed before partitioning into training- and test sets.

A.1 Weather-exposed samples

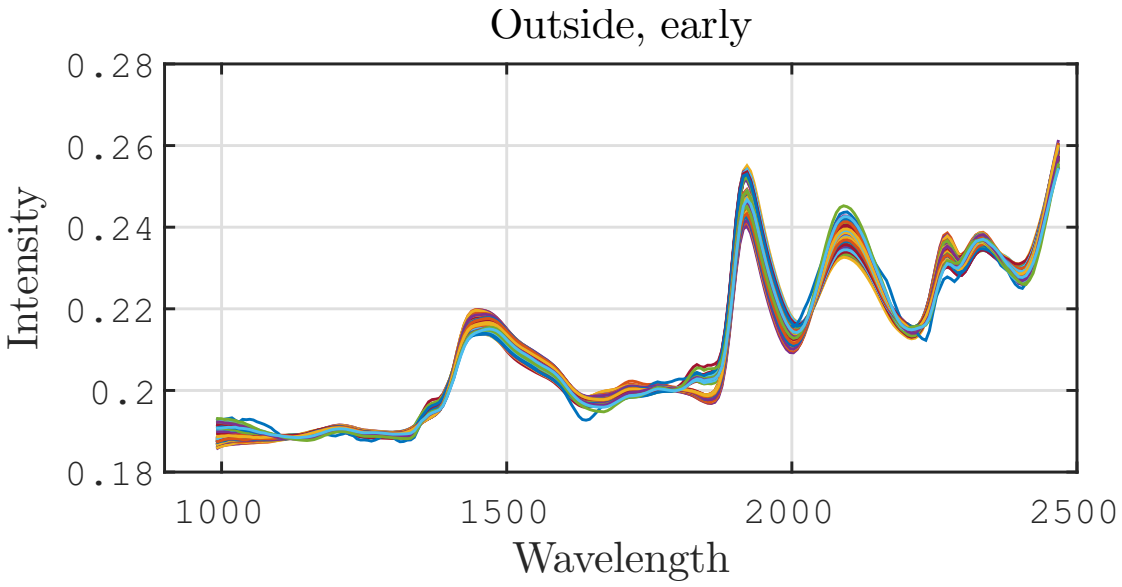
A.1.1 Earlywood

Import data

```
clear, clc, close all
set(0, 'defaultTextInterpreter', 'latex',...
        'defaultAxesFontName', 'AvantGarde')
load /home/smeland/Documents/Datasett_master/outside_v4

% Identify the rows containing "strange" observations.
lower = find(outside_means.early(:,135) < 0.201);

spectra = outside_means.early;
[n, ~] = size(spectra);

% Spectra including constant term
aug_spectra = [ones(n,1), spectra];

response = outside_response_sort(:,4);
```

Plot spectra

```
figure('Position', [100 100 400 200])
plot(wavelengths,spectra'), grid;
title('Outside, early')
xlabel('Wavelength');
ylabel('Intensity')
 xlim([900, 2500])
```



Mean spectra for the weather-exposed samples, earlywood

Regularization
Get PRESS values, regression coefficients and optimal regularization parameter for ridge regression and first derivatives regularization.

```
lambdas = logspace(-10,2, 1000);
[ridge.PRESS, ~, ridge.beta_coeffs, ridge.optimal] =
RregsLooCV(spectra,response,lambdas);
[d1.PRESS, ~, d1.beta_coeffs, d1.optimal] =
TregsLooCV(spectra,response,lambdas,1);
```

PCR
Specify the maximum number of components and perform brute force LOOCV for each number of components. Record PRESS values and obtain best choice of component number, translating to model complexity.

```
maxcomp = 20;
PCR.yhat_cv = cell(maxcomp, 1);
PCR.PRESS = zeros(maxcomp,1);

for k = 1:maxcomp
    PCR.yhat_cv{k} = zeros(n, 1);
    for sample = 1:n
        idx = setdiff(1:n, sample);
        XX = spectra(idx, :);
        yy = response(idx);
        en = ones(n-1,1);
        mX = mean(XX);
```

```
        % Mean centering the data
        X0 = XX - en*mX;

        [T, S, W] = svd(X0, 'econ');

        % Constant term and first k PCA-scores
        % The same as Tk =[en X0*W(:,1:k)];
        Tk = [en (T(:,1:k)*S(1:k,1:k))];

        % Finding coeffiecients in terms of the PCA scores
        qhat2 = Tk\yy;
        yhatT = Tk*qhat2;

        % The regression coeffs of the X-variables
        bb = W(:,1:k)*qhat2(2:end);

        % The constant term is: qhat2(1)-mX*bb
        betahat2 = [qhat2(1)-mX*bb; bb];

        % Uncentered version
        PCR.yhat_cv{k}(sample) = (
        [1, spectra(sample,:)]*betahat2)';
    end
    PCR.PRESS(k) = sum((response - PCR.yhat_cv{k}).^2);
    % Same as the squared norm of the residual vector
    [~, press_idx_pcr] = min(PCR.PRESS);
end
PCR.optimal = press_idx_pcr;
PCR.beta_coeffs = pcr(spectra, response, PCR.optimal);
```

PLS

Perform PLS with increasing number of components, perform LOOCV, record PRESS values and find the optimal number of components, as with PCR

```
PLS.yhat_cv = cell(maxcomp, 1);
PLS.PRESS = zeros(maxcomp,1);
PLS.beta_coeffs = zeros(size(spectra,2)+1, maxcomp);
for k = 1:maxcomp
    PLS.yhat_cv{k} = zeros(n, 1);
    for sample = 1:n
        idx = setdiff(1:n, sample);
        XX = spectra(idx, :);
        yy = response(idx);

        % Use the NIPALS algorithm to obtain coefficients,
        % weights and scores
        [PLS.beta_coeffs(:,k), W, T] = plsNIP(XX, yy, k);
        PLS.yhat_cv{k}(sample) = (
```

```
        [1 spectra(sample,:)]*PLS.beta_coeffs(:,k))';
    end
    PLS.PRESS(k) = sum((response - PLS.yhat_cv{k}).^2);
    [~, press_idx_pls] = min(PLS.PRESS);
end
PLS.optimal = press_idx_pls;
PLS.beta_coeffs = plsNIP(spectra, response, PLS.optimal);
```

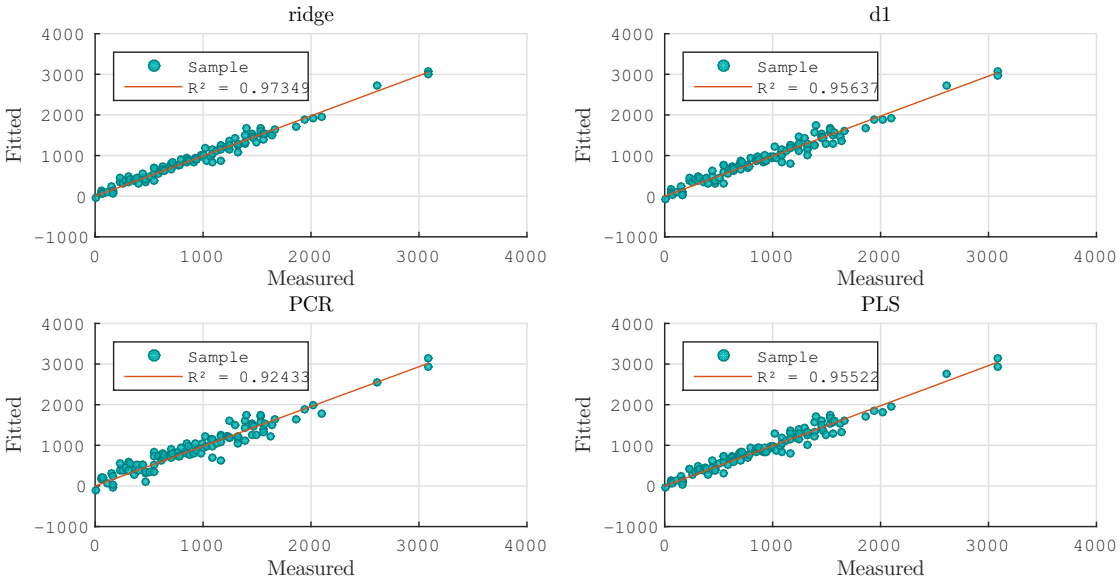
Get diagnostics

```
% Predict with "best" coefficient vectors
ridge.yhat = aug_spectra*ridge.beta_coeffs;
d1.yhat = aug_spectra*d1.beta_coeffs;
PCR.yhat = aug_spectra*PCR.beta_coeffs;
PLS.yhat = aug_spectra*PLS.beta_coeffs;

% Calculate prediction errors
ridge.res = response - ridge.yhat;
d1.res = response - d1.yhat;
PCR.res = response - PCR.yhat;
PLS.res = response - PLS.yhat;

% Calculate root mean squared errors of prediction (RMSEP)
ridge.RMSEC = sqrt(sum(ridge.res.^2)/n);
d1.RMSEC = sqrt(sum(d1.res.^2)/n);
PCR.RMSEC = sqrt(sum(PCR.res.^2)/n);
PLS.RMSEC = sqrt(sum(PLS.res.^2)/n);

% Coefficient of Determination
k = {ridge.yhat, d1.yhat, PCR.yhat, PLS.yhat};
for i = 1:4
    r2(i) = calc_r2(k{i}, response);
end
```

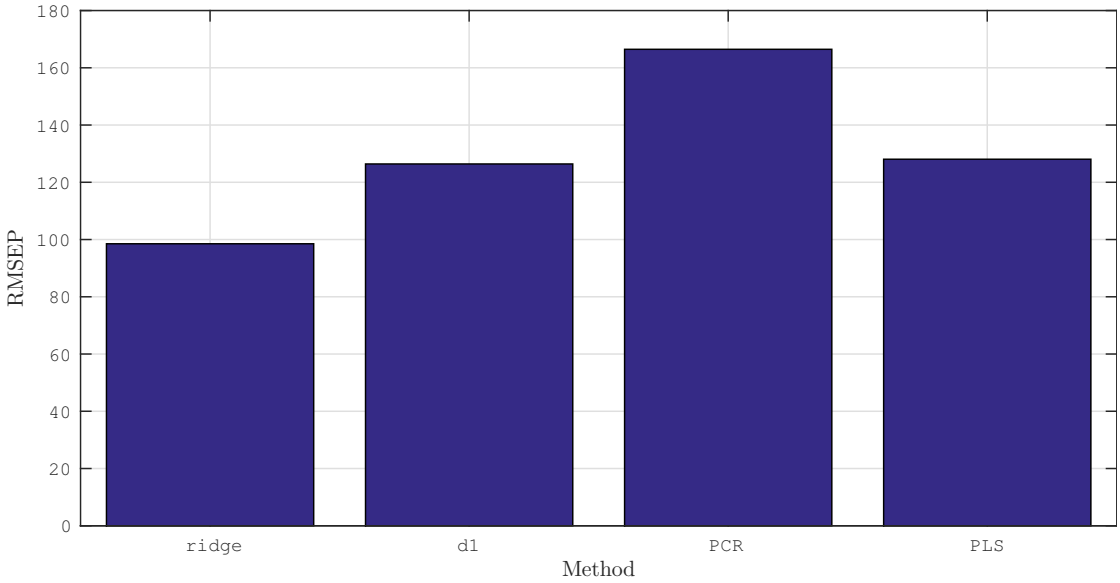
Plots

```
methods = {'ridge', 'd1', 'PCR', 'PLS'};
yhats = [ridge.yhat, d1.yhat, PCR.yhat, PLS.yhat];
p = size(response, 1);
dummy_x = linspace(min(response), max(response), p);

figure('Name', 'Predictions',...
'Position', [100 100 700 600]);
for method = 1:4
    subplot(2,2,method)
    betas = response\yhats(:,method);
    ys = dummy_x*betas;
    scatter(response, yhats(:,method), 15,...
        'MarkerEdgeColor',[0 .5 .5],...
        'MarkerFaceColor',[0 .7 .7]), grid, hold on
    xlabel('Actual'), ylabel('Fitted')
    plot(dummy_x, ys)
    xlabel('Actual'), ylabel('Fitted')
```

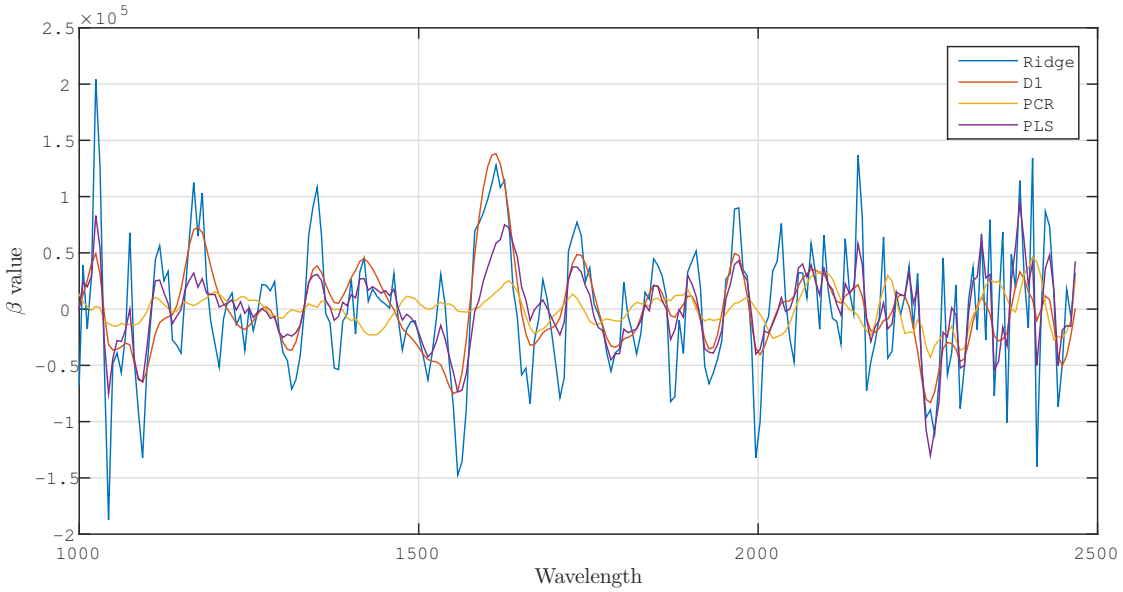
```
    legend('Sample', ['R2 = ' num2str(r2(method))],...
    'Location', 'NorthWest')
    title(methods(method))
end

figure('Name', 'Residual comparison');
bar([ridge.RMSEC, d1.RMSEC, PCR.RMSEC, PLS.RMSEC]), grid
set(gca, 'XTickLabel', methods)
xlabel('Method'); ylabel('RMSEP')

figure('Name', 'Coefficients')
plot(wavelengths, [ridge.beta_coeffs(2:end),...
d1.beta_coeffs(2:end),PCR.beta_coeffs(2:end),...
PLS.beta_coeffs(2:end)]),grid, hold on
xlabel('Wavelength'); ylabel('$\beta$ value')
legend('Ridge', 'D1', 'PCR', 'PLS')

figure('Name', 'PRESS values',...
'Position', [200 200 700 400])
for method = 1:4
    subplot(2,2,method)
    if method == 1 || method == 2
        plot(lambdas,eval([methods{method} '.PRESS']))
        grid,hold on
        set(gca, 'Xscale', 'log', 'XTick',..
         logspace(-10,2, 5) )
        xlim([10^-10 10^2])
        xlabel('$\lambda$ value'); ylabel('PRESS')
    else
        plot(eval([methods{method} '.PRESS']))
        grid, hold on
        xlabel('Components'); ylabel('PRESS')
    end
    title(methods(method))
end
```



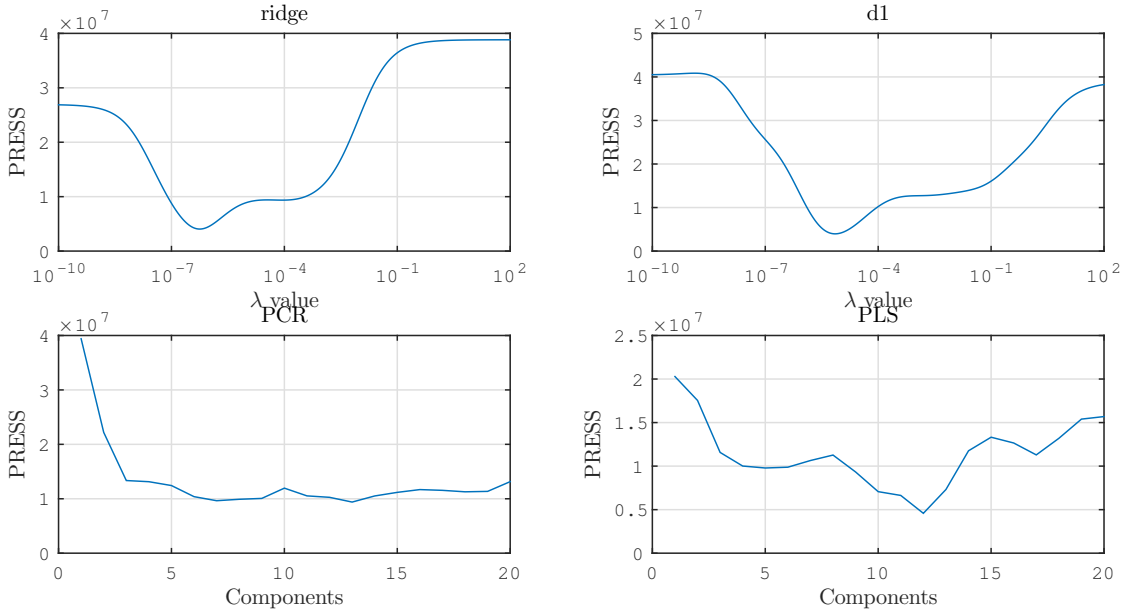
Prediction versus measured UV exposure.



RMSEP values for each method



Coefficient vectors for four methods.

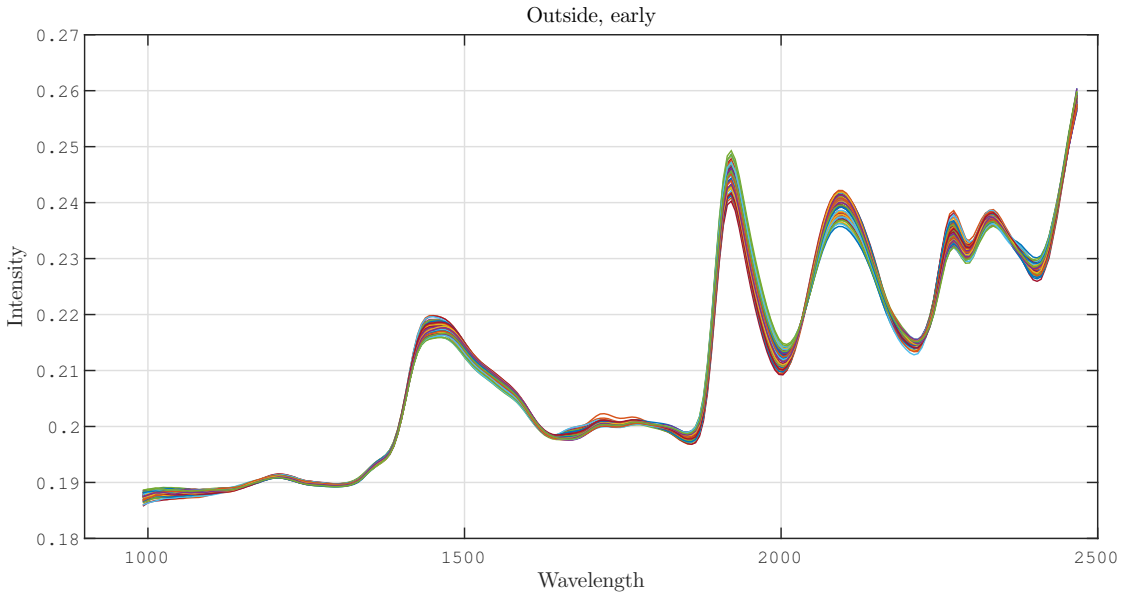


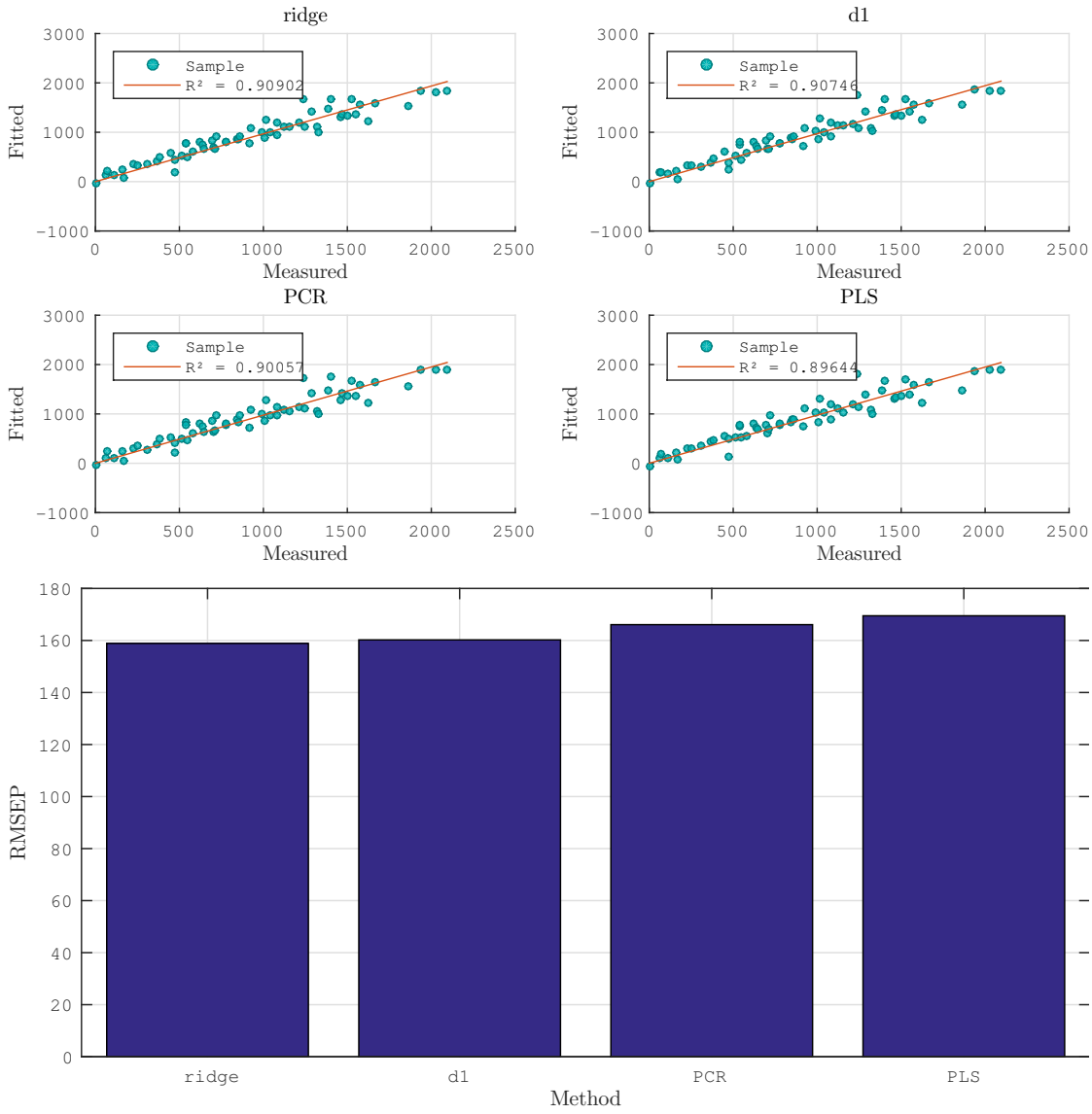
PRESS values for the methods.

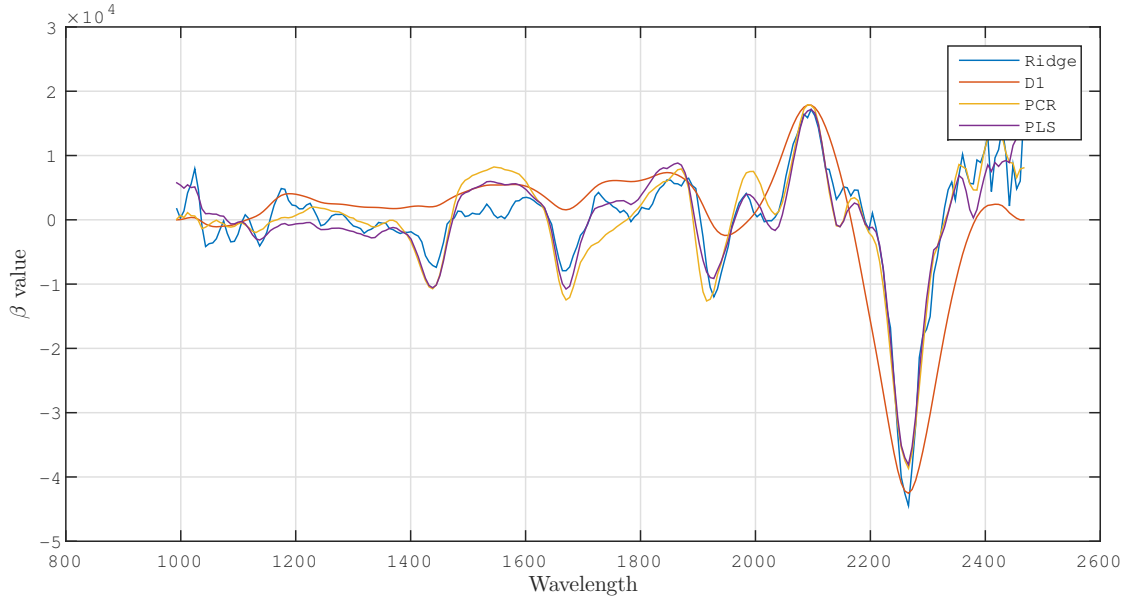
A.1.2 Earlywood, cleaned

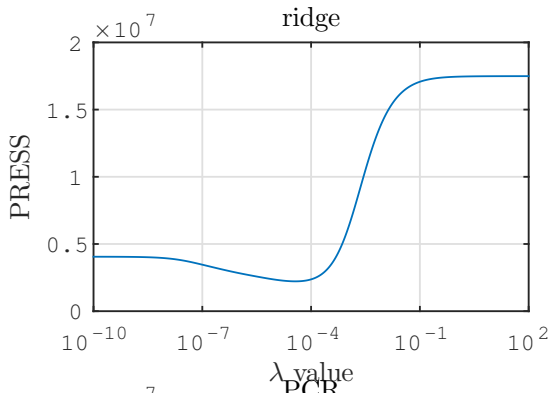
Same data, but with the anomalous observations removed.

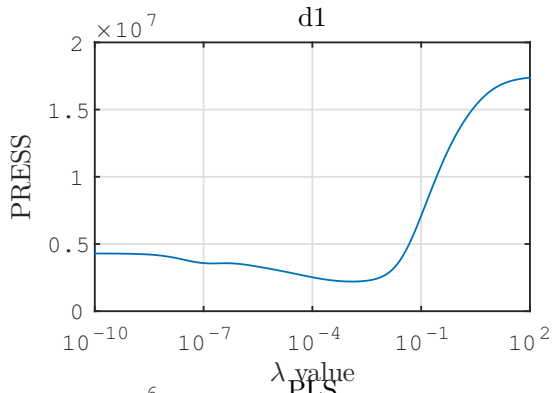
```
clear, clc, close all
set(0, 'defaultTextInterpreter', 'latex',...
        'defaultAxesFontName', 'AvantGarde')
load /home/smeland/Documents/Datasett_master/outside_v4

% Identify the rows containing "strange" observations.
lower = find(outside_means.early(:,135) < 0.201);

spectra = outside_means.early(lower,:);
[n, ~] = size(spectra);

% Spectra including constant term
aug_spectra = [ones(n,1), spectra];

response = outside_response_sort(lower,4);
```

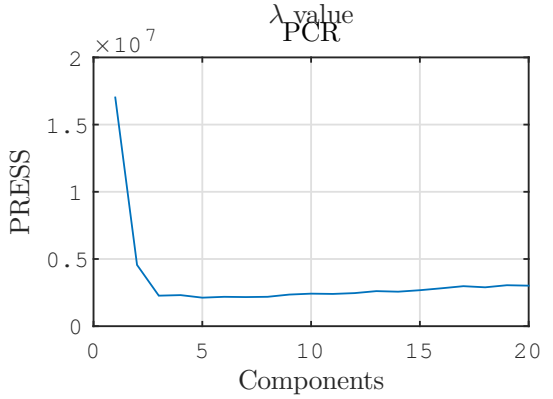


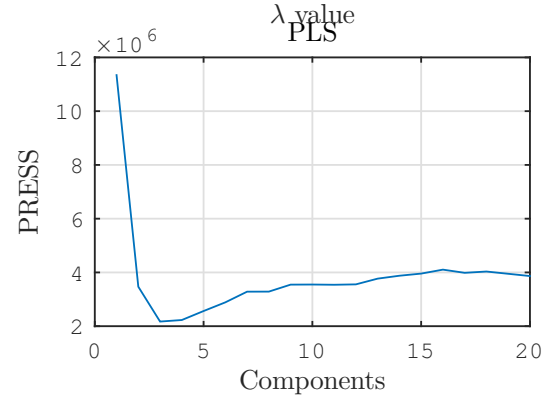

ridge
Sample
R² = 0.90902
Fitted
Measured
d1
Sample
R² = 0.90746
Fitted
Measured
PCR
Sample
R² = 0.90057
Fitted
Measured
PLS
Sample
R² = 0.89644
Fitted
Measured
RMSEP
ridge
d1
PCR
PLS
Method


×10^4
3
2
1
0
−1
−2
−3
−4
−5
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS


ridge
×10^7
2
1.5
1
0.5
0
PRESS
10^−10
10^−7
10^−4
10^−1
10^2
λ value

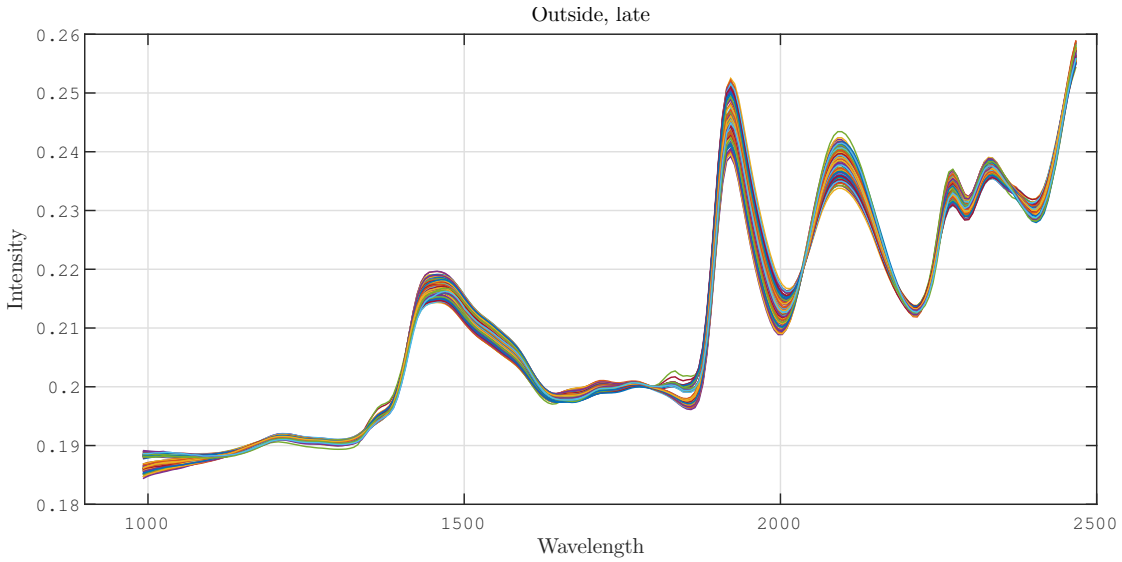

d1
×10^7
2
1.5
1
0.5
0
PRESS
10^−10
10^−7
10^−4
10^−1
10^2
λ value

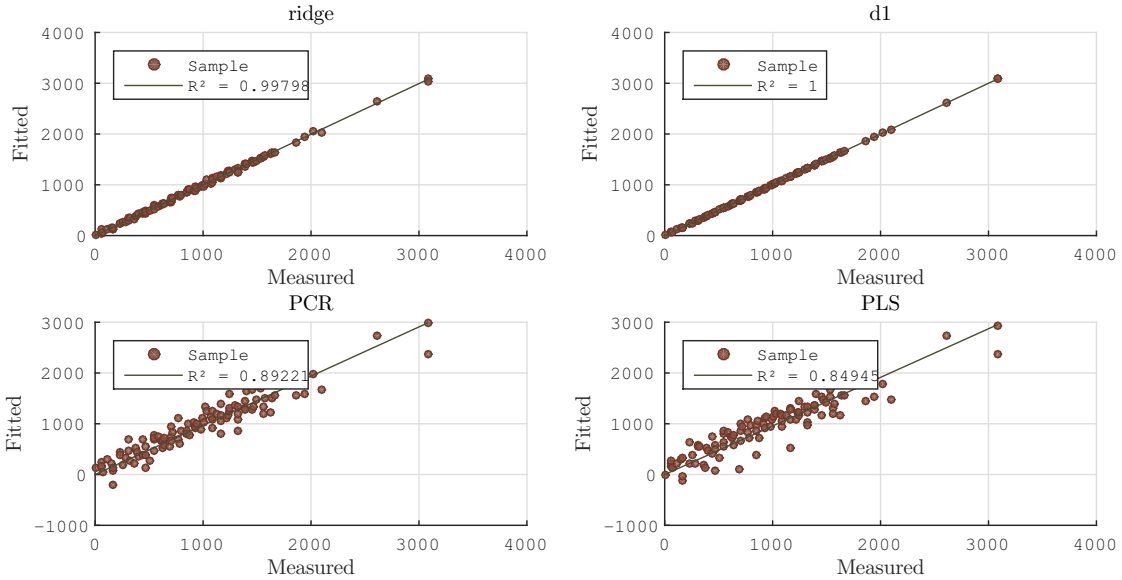

PCR
×10^7
2
1.5
1
0.5
0
PRESS
0
5
10
15
20
Components

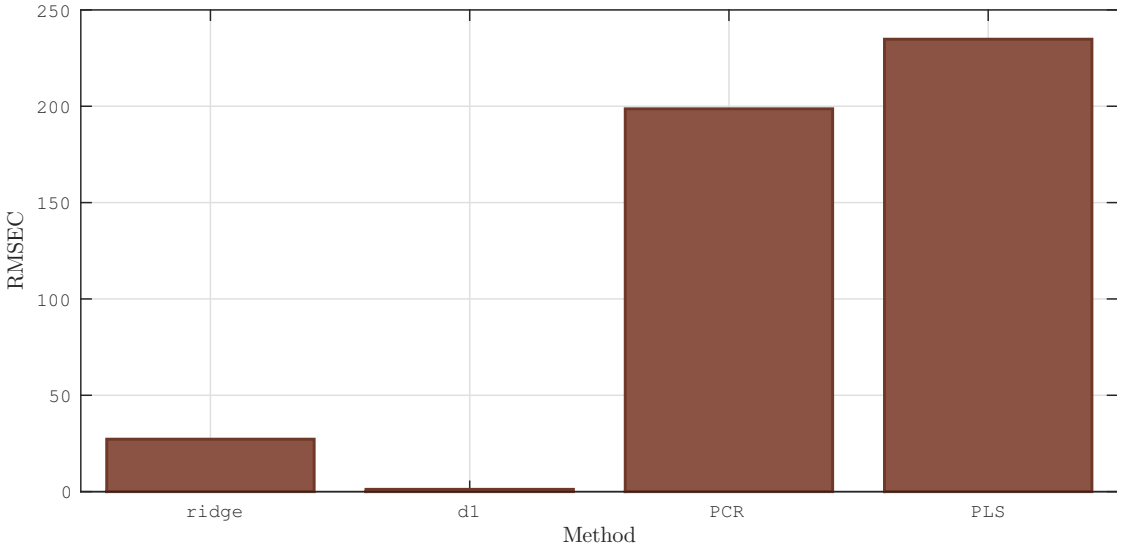

PLS
×10^6
12
10
8
6
4
2
PRESS
0
5
10
15
20
Components

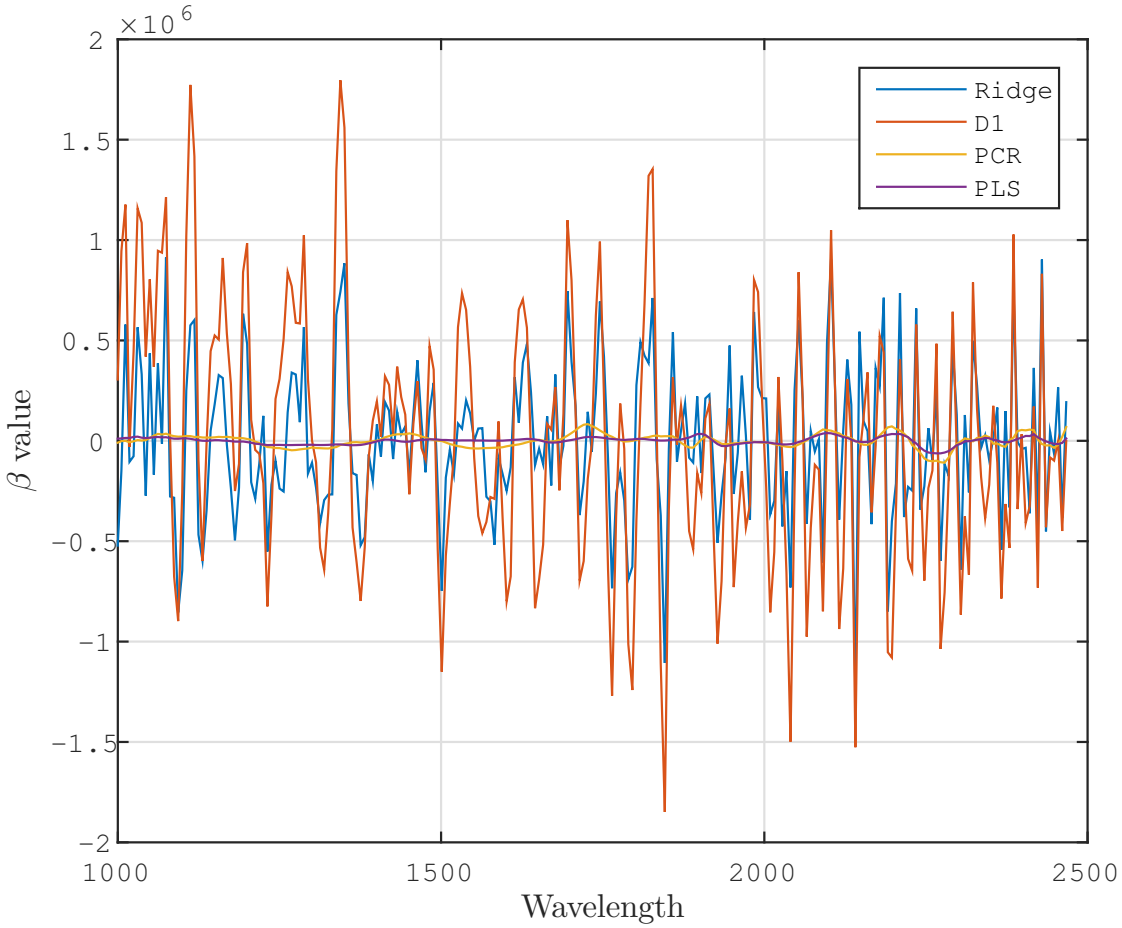
A.1.3 Latewood

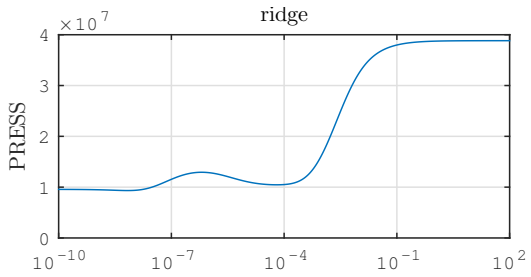
```
clear, clc, close all
load /home/smeland/Documents/Datasett_master/outside_v4
lower = find(outside_means.early(:,135) < 0.201);

spectra = outside_means.late;
[n, ~] = size(spectra);
aug_spectra = [ones(n,1), spectra];
response = outside_response_sort(:,4);
```

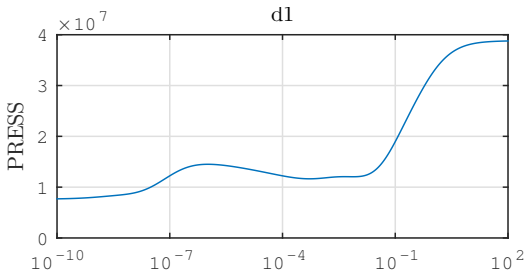


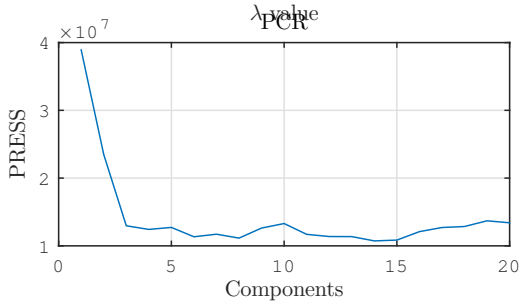


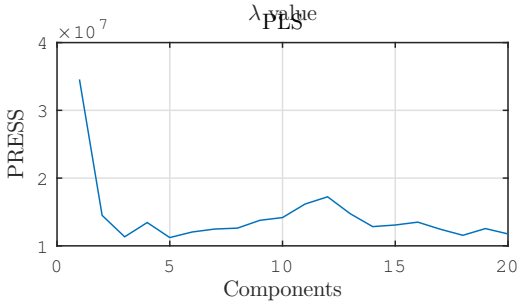

250
200
150
100
50
0
RMSEC
ridge
d1
PCR
PLS
Method


×10^6
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
β value
1000
1500
2000
2500
Wavelength
Ridge
D1
PCR
PLS


ridge
PRESS
×10^7
4
3
2
1
0
10^-10
10^-7
10^-4
10^-1
10^2
λ value

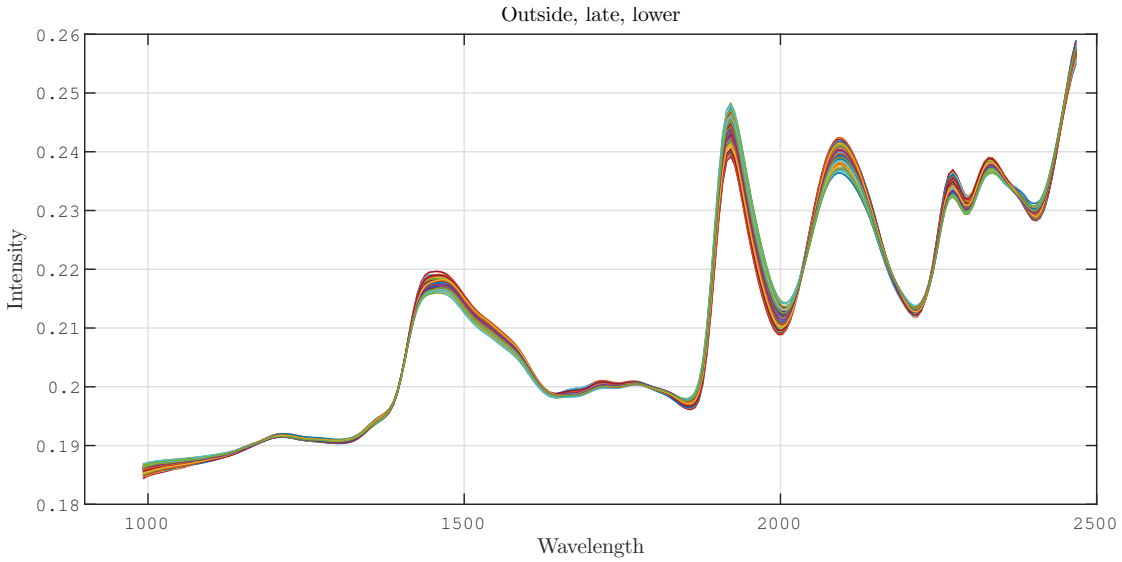

d1
PRESS
×10^7
4
3
2
1
0
10^-10
10^-7
10^-4
10^-1
10^2
λ value

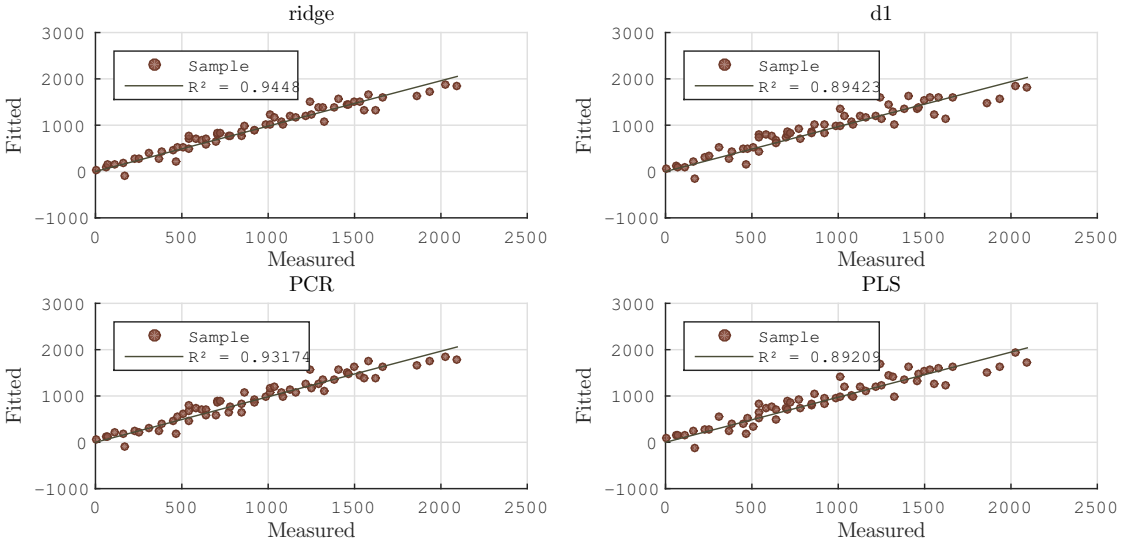

PCR
PRESS
×10^7
4
3
2
1
0
5
10
15
20
Components

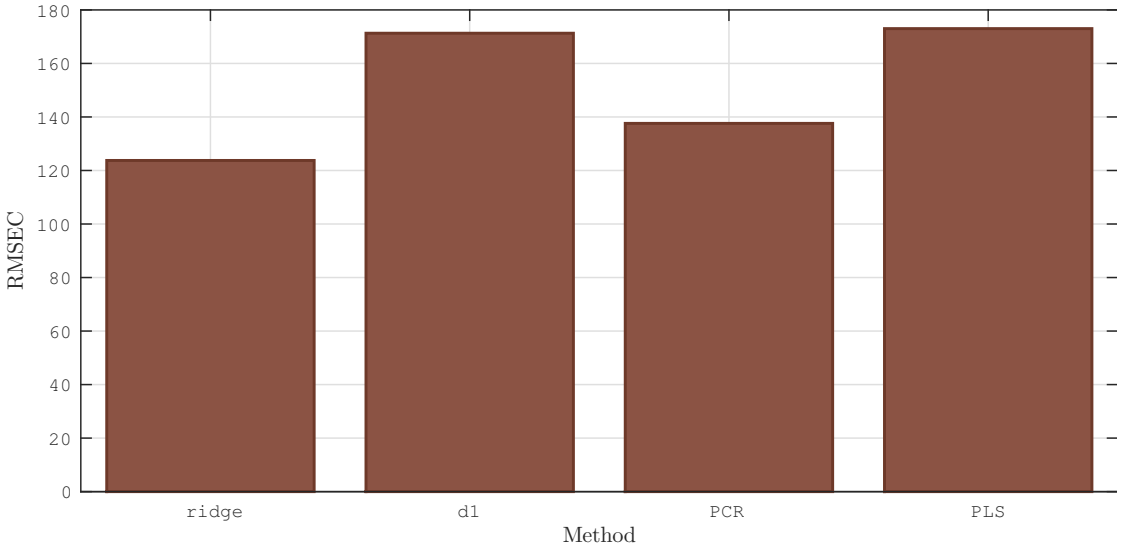

PLS
PRESS
×10^7
4
3
2
1
0
5
10
15
20
Components

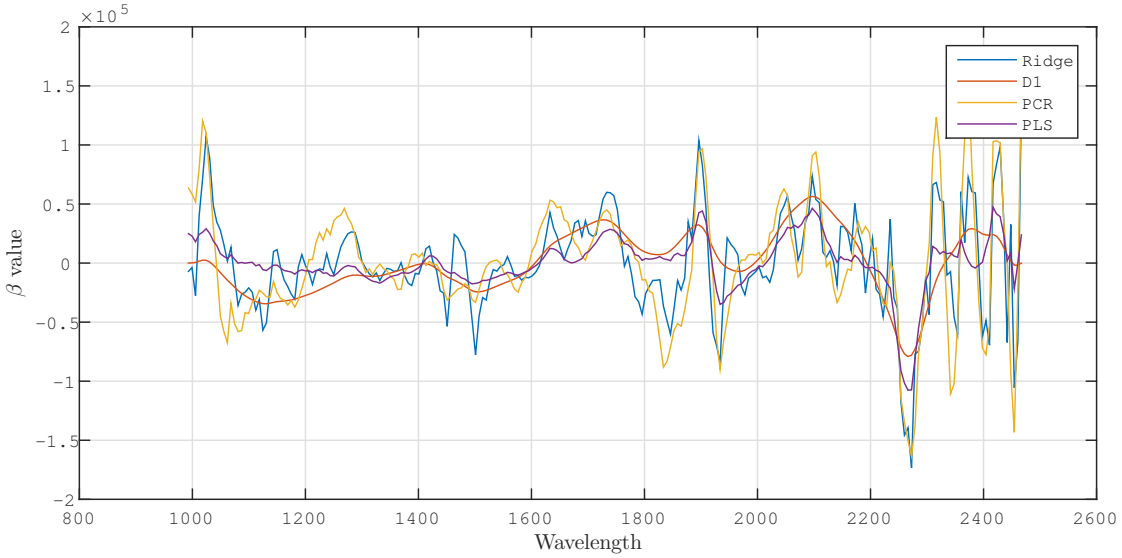
A.1.4 Latewood, cleaned

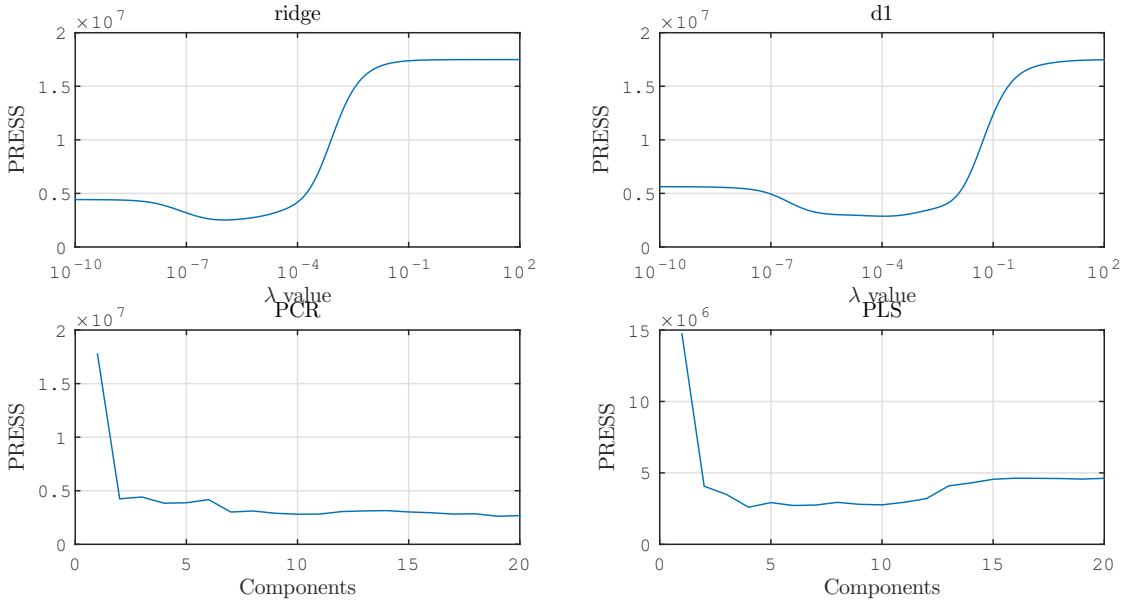
```
clear, clc, close all
load /home/smeland/Documents/Datasett_master/outside_v4
lower = find(outside_means.early(:,135) < 0.201);

spectra = outside_means.late(lower,:);
[n, ~] = size(spectra);
aug_spectra = [ones(n,1), spectra];
response = outside_response_sort(lower,4);
```





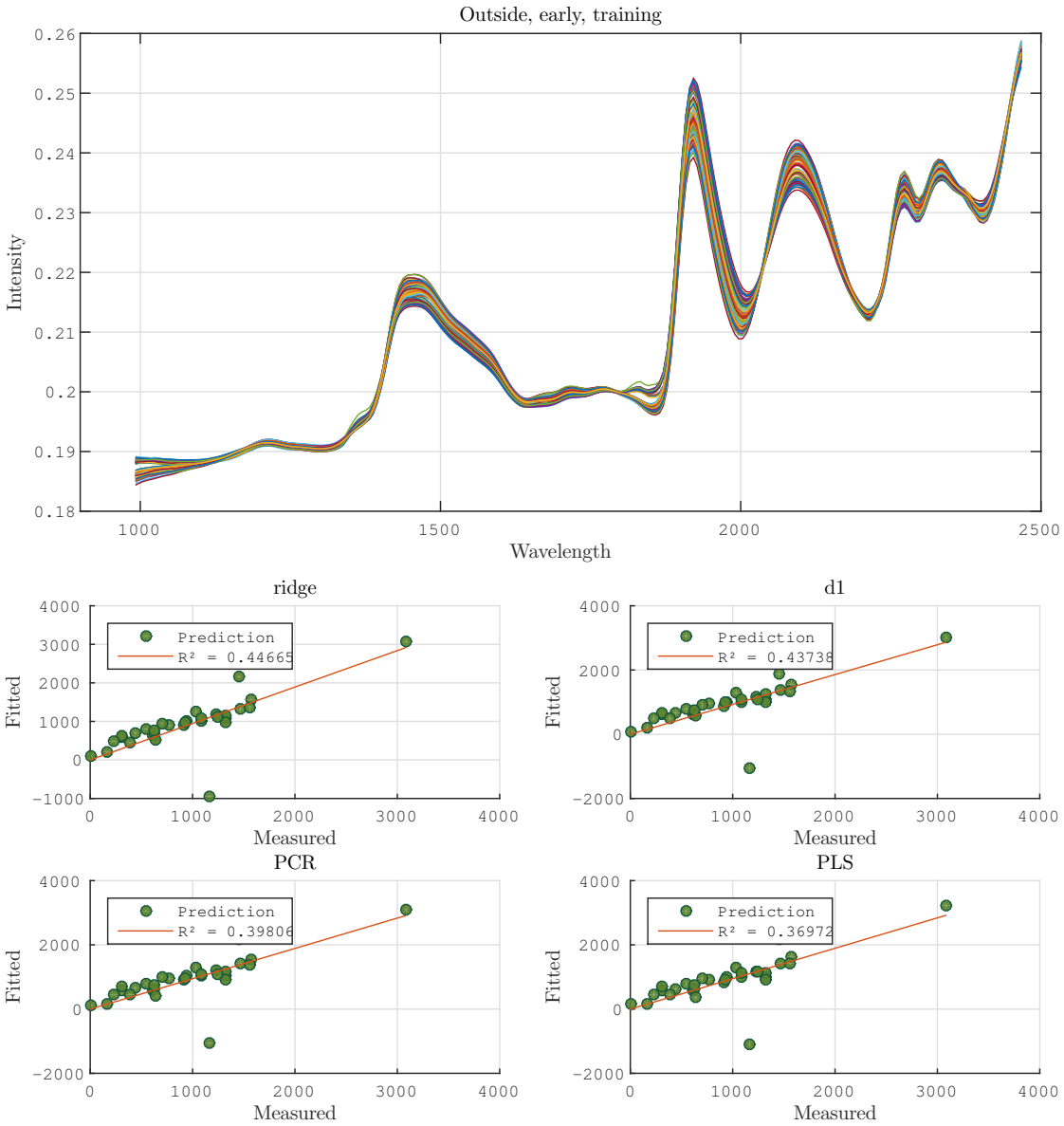

RMSEC
180
160
140
120
100
80
60
40
20
0
ridge
d1
PCR
PLS
Method

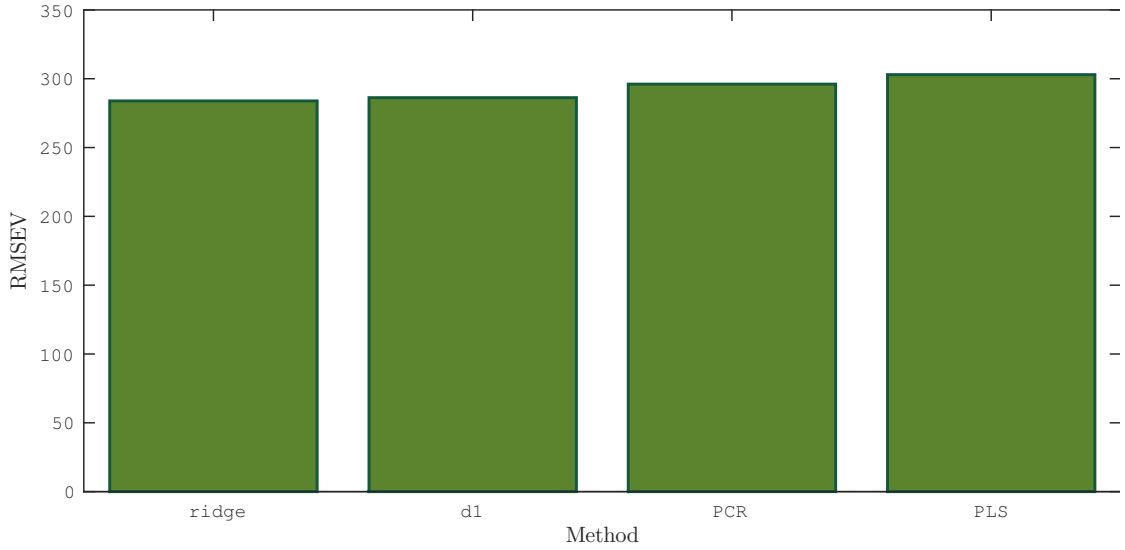

×10^5
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS

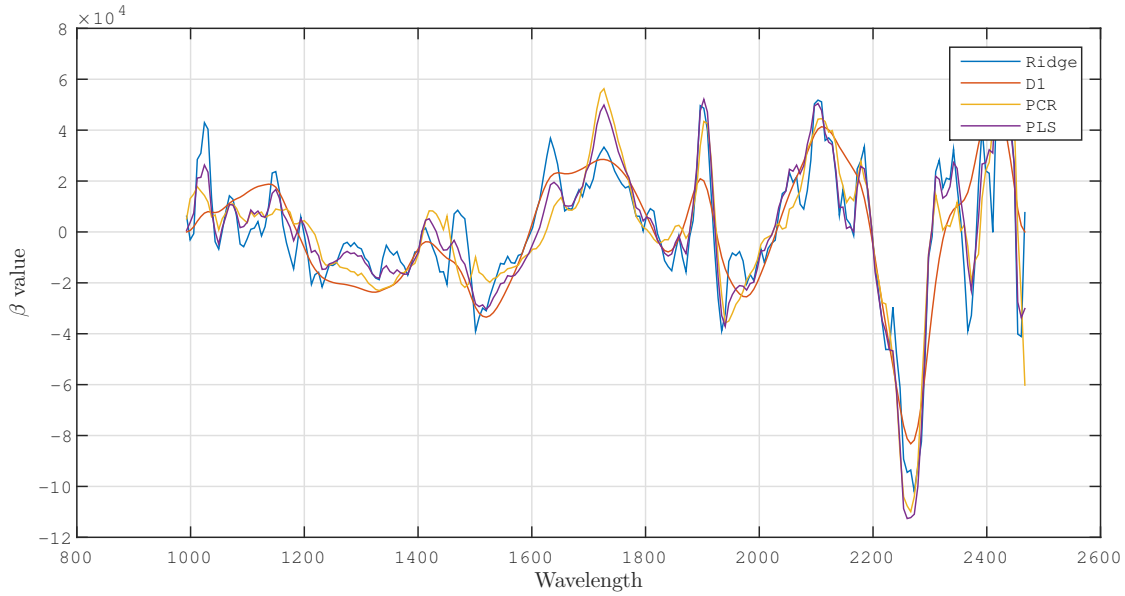

ridge
d1
PCR
PLS
PRESS
λ value
Components
×10^7
×10^6

A.1.5 Earlywood, validation

```
clear, clc, close all
load /home/smeland/Documents/Datasett_master/outside_v4
lower = find(outside_means.early(:,135) < 0.201);

spectra = outside_means.late;
response= outside_response_sort(:,4);
[m, ~] = size(spectra);

split = round(m*0.7);
seq = randperm(m);
spectra_train = spectra(seq(1:split),:);
spectra_test = spectra(seq(split+1:end), :);
response_train = response(seq(1:split));
response_test = response(seq(split+1:end));

aug_spectra = [ones(size(spectra_test,1),1), spectra_test];
n = size(spectra_train, 1);

% Predict with "best" coefficient vectors
ridge.yhat = aug_spectra*ridge.beta_coeffs;
d1.yhat = aug_spectra*d1.beta_coeffs;
PCR.yhat = aug_spectra*PCR.beta_coeffs;
PLS.yhat = aug_spectra*PLS.beta_coeffs;

% Calculate prediction errors
ridge.res = response_test - ridge.yhat;
d1.res = response_test - d1.yhat;
PCR.res = response_test - PCR.yhat;
PLS.res = response_test - PLS.yhat;

% Calculate root mean squared errors of prediction (RMSEP)
ridge.RMSEV = sqrt(sum(ridge.res.^2)/n);
d1.RMSEV = sqrt(sum(d1.res.^2)/n);
PCR.RMSEV = sqrt(sum(PCR.res.^2)/n);
PLS.RMSEV = sqrt(sum(PLS.res.^2)/n);

% Coefficients of determination
k = {ridge.yhat, d1.yhat, PCR.yhat, PLS.yhat};
for i = 1:4
    r2(i) = calc_r2(k{i}, response_test);
end
```

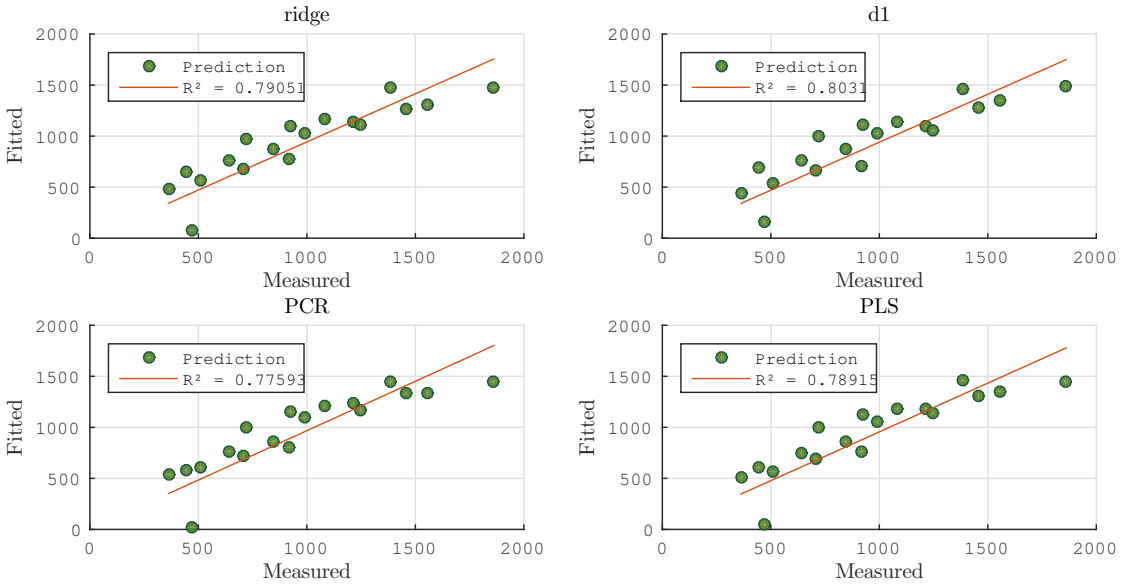

Outside, early, training
Intensity
Wavelength
ridge
Prediction
R² = 0.44665
d1
Prediction
R² = 0.43738
PCR
Prediction
R² = 0.39806
PLS
Prediction
R² = 0.36972
Fitted
Measured

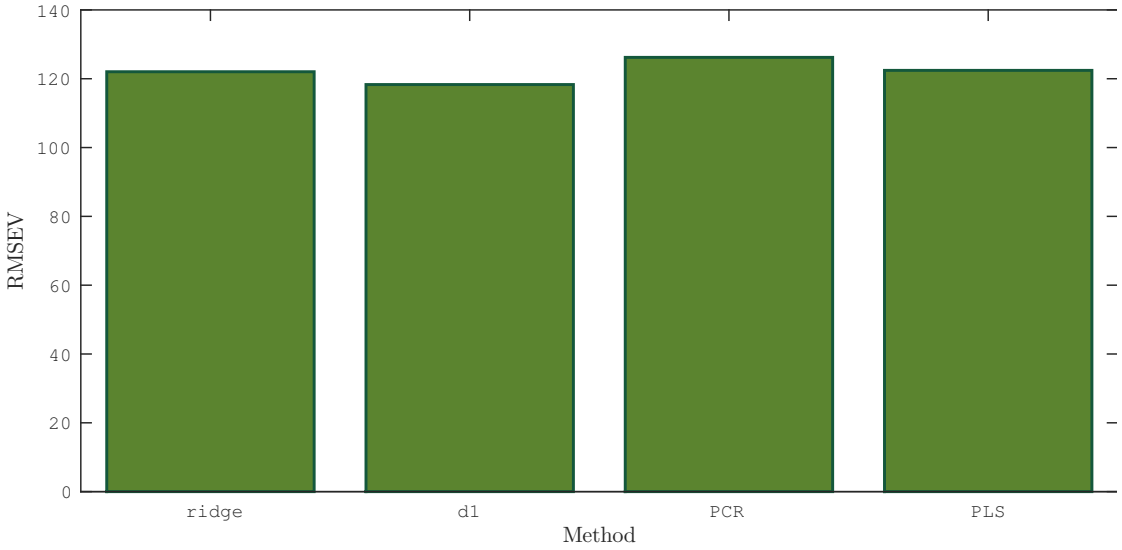

350
300
250
200
150
100
50
0
RMSEV
ridge
d1
PCR
PLS
Method

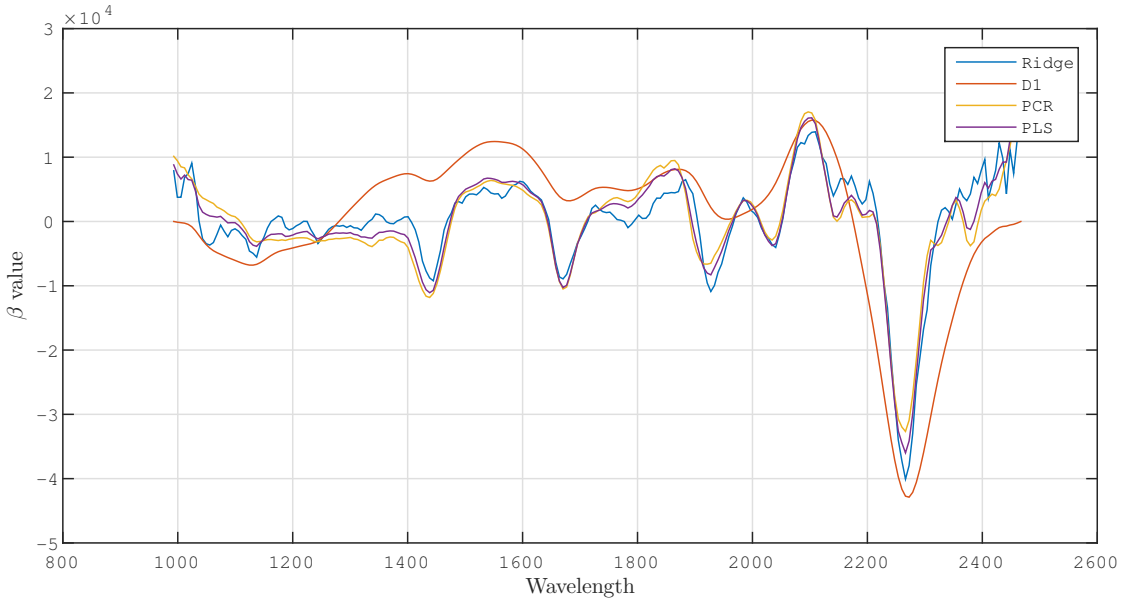

×10^4
8
6
4
2
0
−2
−4
−6
−8
−10
−12
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS

A.1.6 Earlywood, validation, cleaned

```
clear, clc, close all
load /home/smeland/Documents/Datasett_master/outside_v4
lower = find(outside_means.early(:,135) < 0.201);

spectra = outside_means.early(lower,:);
response= outside_response_sort(lower,4);
[m, ~] = size(spectra);

split = round(m*0.7);
seq = randperm(m);
spectra_train = spectra(seq(1:split),:);
spectra_test = spectra(seq(split+1:end), :);
response_train = response(seq(1:split));
response_test = response(seq(split+1:end));

aug_spectra = [ones(size(spectra_test,1),1), spectra_test];
n = size(spectra_train, 1);
```

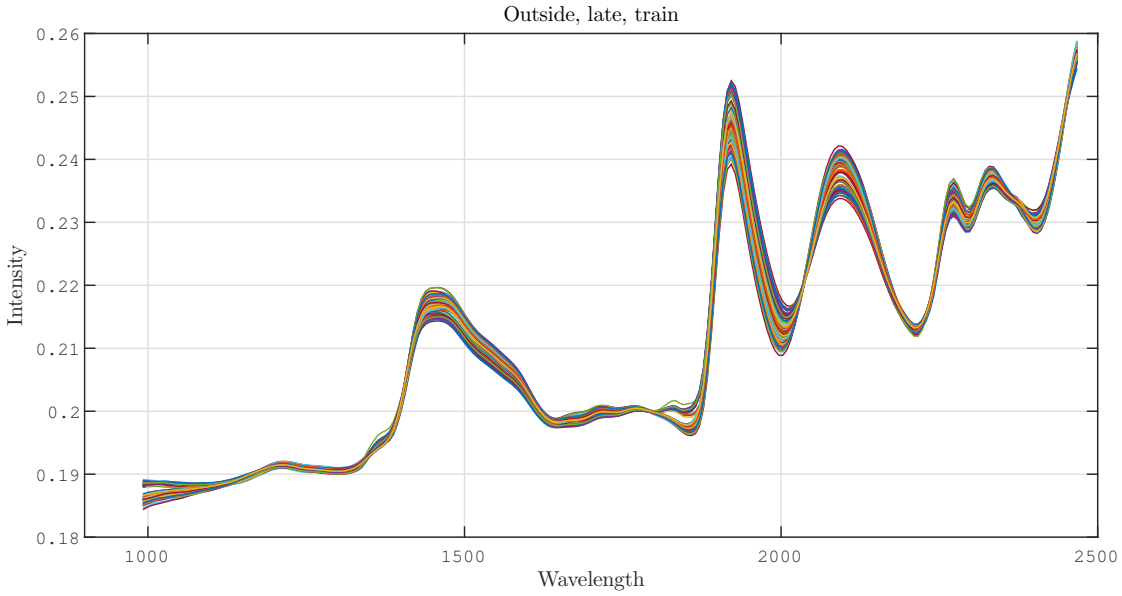


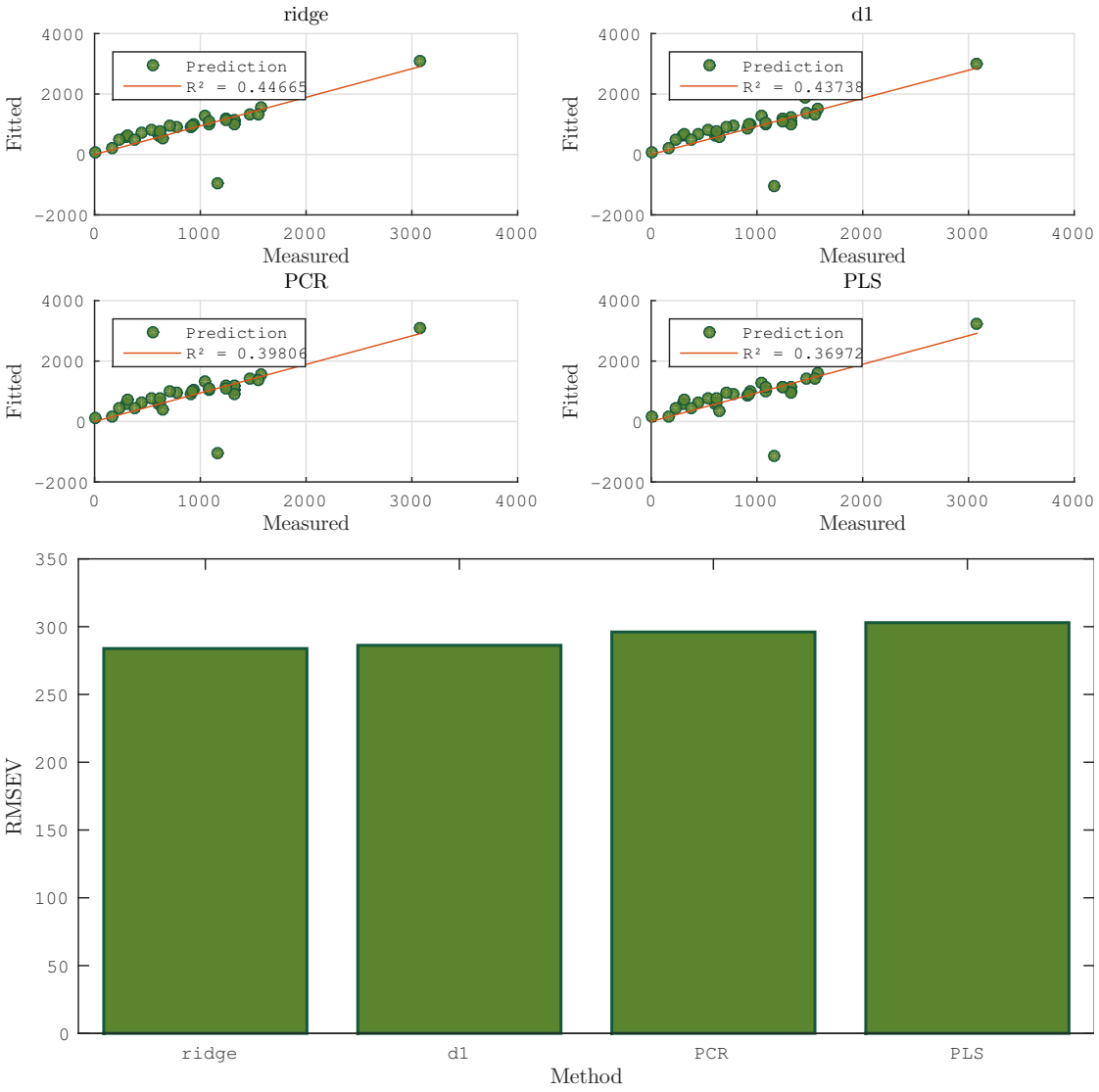

RMSEV
140
120
100
80
60
40
20
0
ridge
d1
PCR
PLS
Method

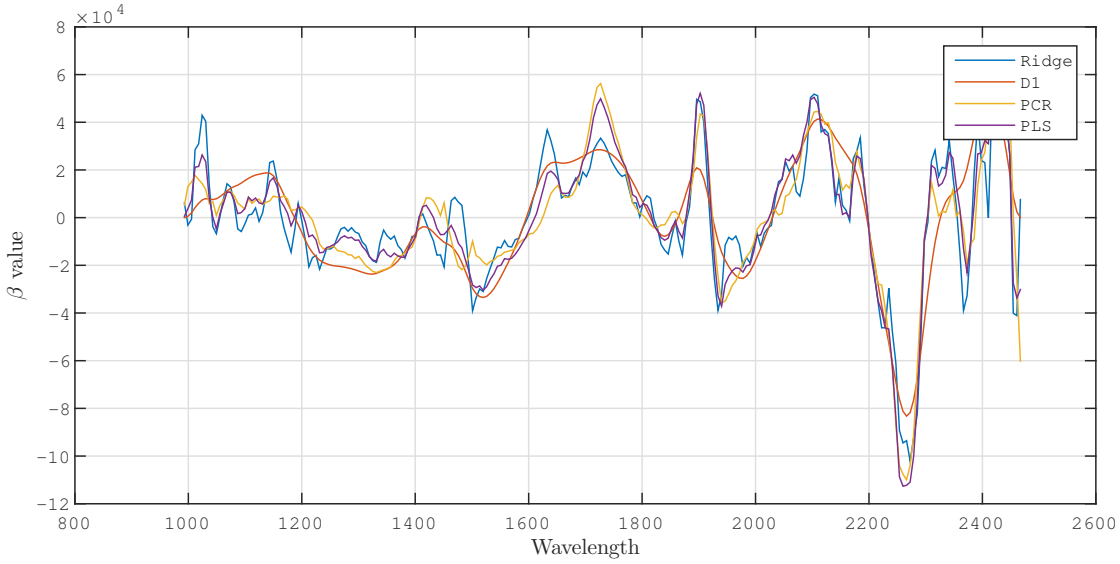

×10^4
3
2
1
0
−1
−2
−3
−4
−5
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS

A.1.7 Latewood, validation

```
lower = find(outside_means.early(:,135) < 0.201);

spectra = outside_means.late;
response= outside_response_sort(:,4);
[m, ~] = size(spectra);

% Randomly assign observations to either test or
% training set.
fraction_training = 0.7;
split = round(m*fraction_training);
seq = randperm(m);
spectra_train = spectra(seq(1:split),:);
spectra_test = spectra(seq(split+1:end), :);
response_train = response(seq(1:split));
response_test = response(seq(split+1:end));

aug_spectra = [ones(size(spectra_test,1),1),...
spectra_test];
n = size(spectra_train, 1);
```

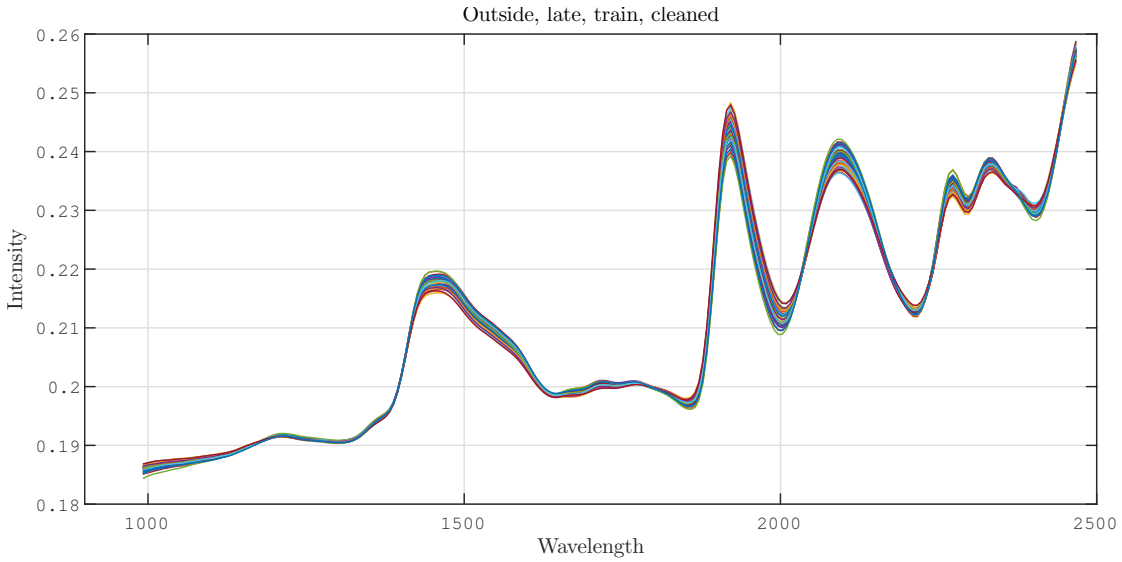


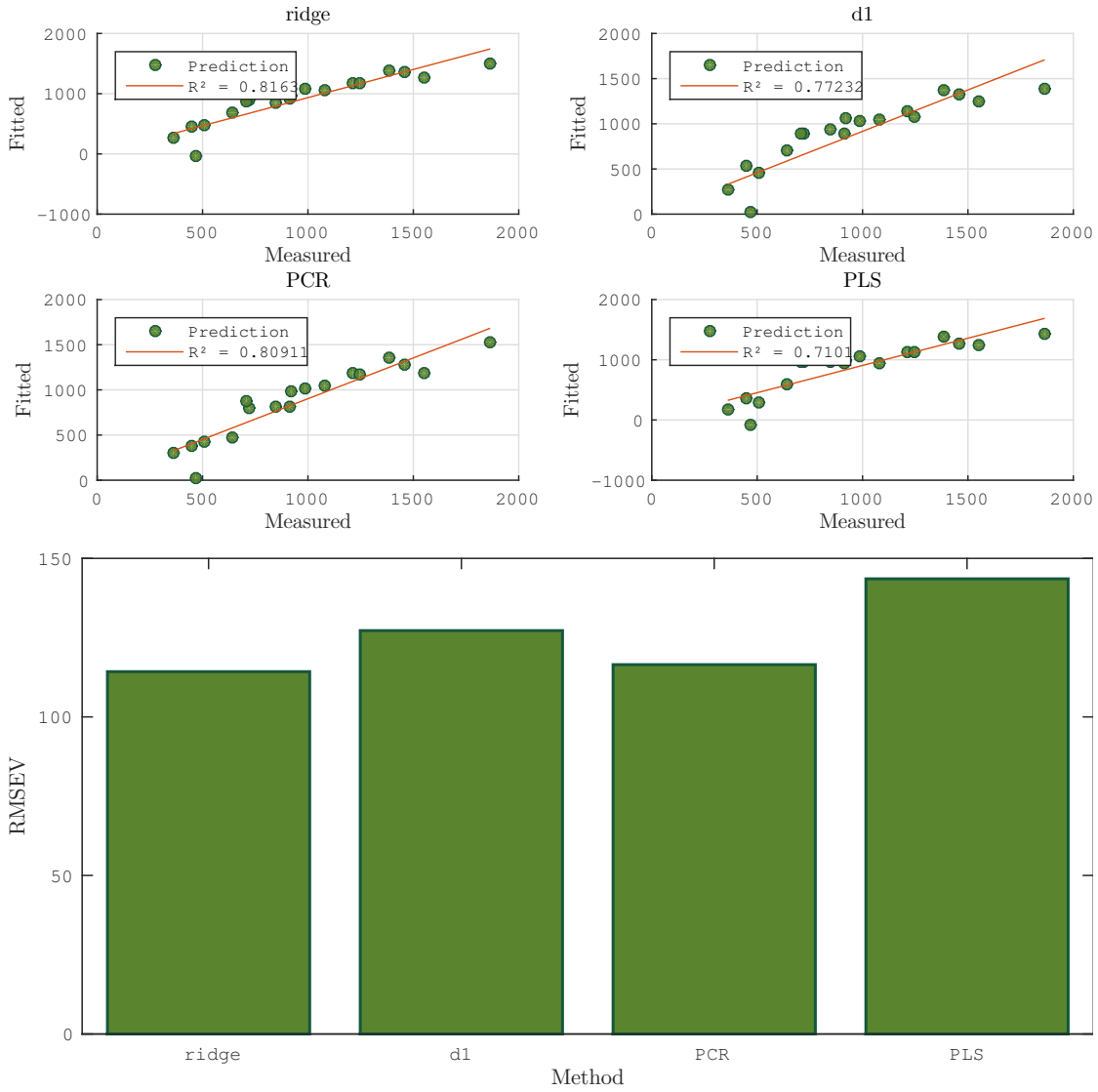

ridge
Prediction
R² = 0.44665
Fitted
Measured
d1
Prediction
R² = 0.43738
Fitted
Measured
PCR
Prediction
R² = 0.39806
Fitted
Measured
PLS
Prediction
R² = 0.36972
Fitted
Measured
RMSEV
ridge
d1
PCR
PLS
Method

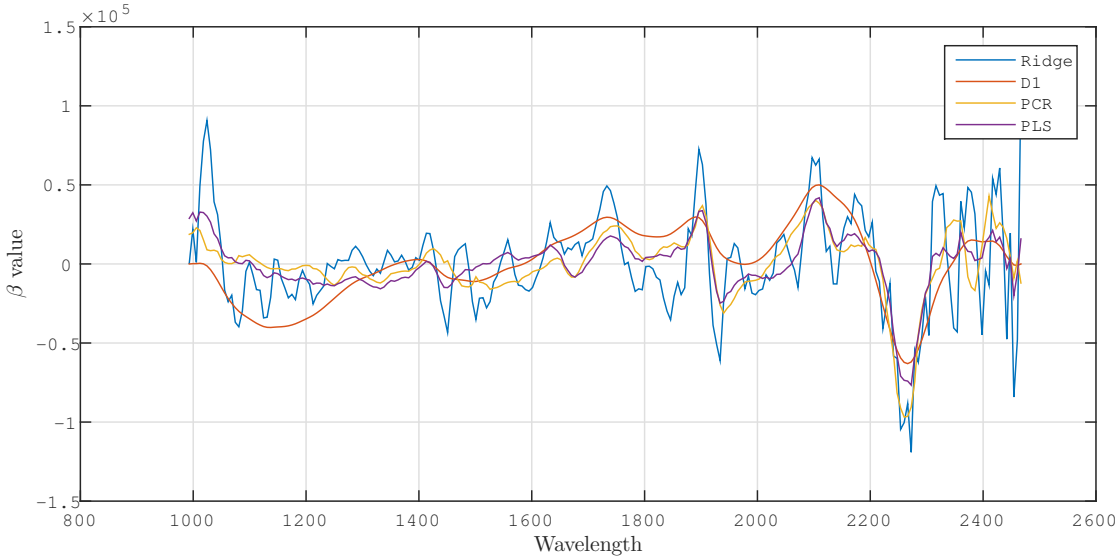

×10^4
8
6
4
2
0
−2
−4
−6
−8
−10
−12
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS

A.1.8 Latewood, validation, cleaned

```
% Identify the anomalous observations
lower = find(outside_means.early(:,135) < 0.201);

% Exclude from data
spectra = outside_means.late(lower,:);
response= outside_response_sort(lower,4);
[m, ~] = size(spectra);

split = round(m*0.7);
seq = randperm(m);
spectra_train = spectra(seq(1:split),:);
spectra_test = spectra(seq(split+1:end), :);
response_train = response(seq(1:split));
response_test = response(seq(split+1:end));

aug_spectra = [ones(size(spectra_test,1),1),...
spectra_test];
n = size(spectra_train, 1);
```



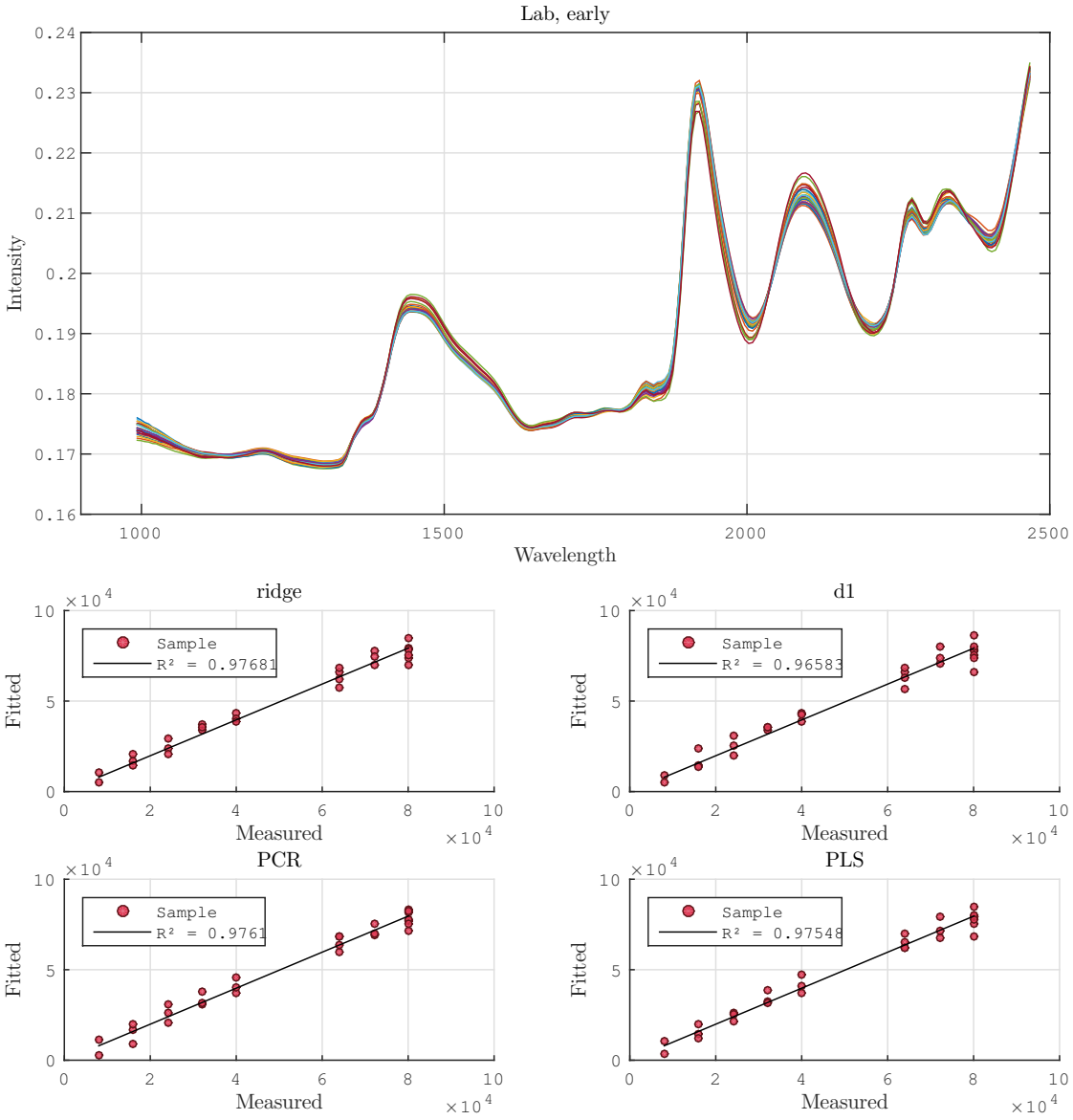

ridge
Prediction
R² = 0.8163
Fitted
Measured
d1
Prediction
R² = 0.77232
Fitted
Measured
PCR
Prediction
R² = 0.80911
Fitted
Measured
PLS
Prediction
R² = 0.7101
Fitted
Measured
RMSEV
ridge
d1
PCR
PLS
Method

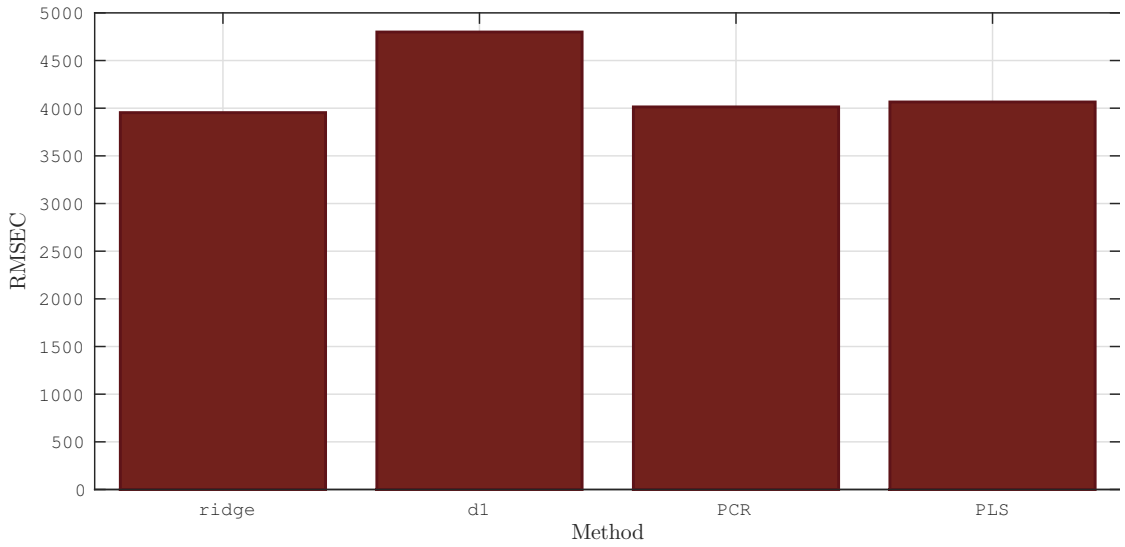

×10^5
1.5
1
0.5
0
−0.5
−1
−1.5
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS

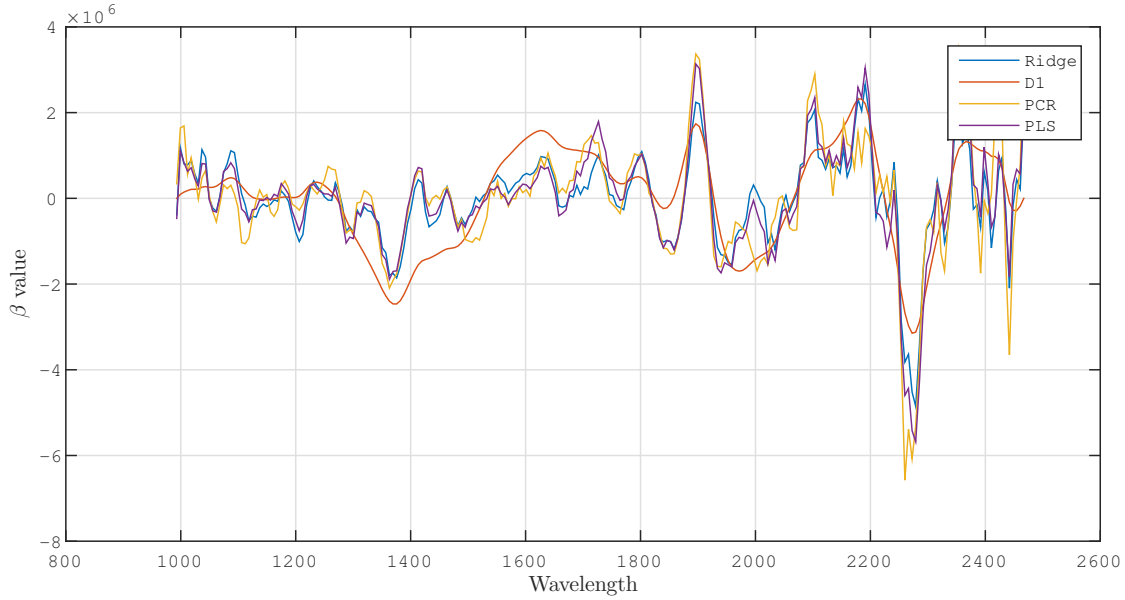
A.2 Lab samples

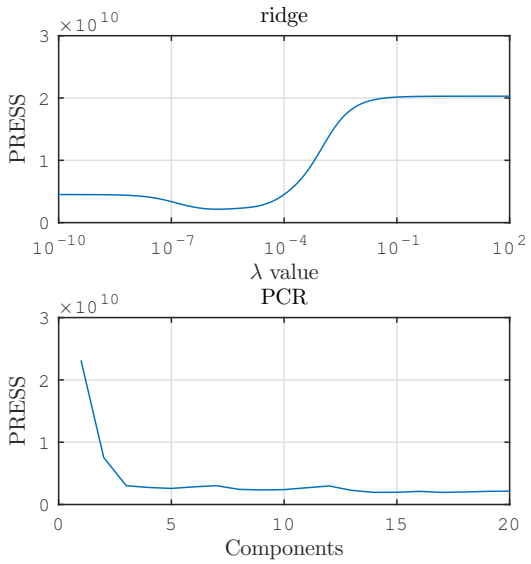
A.2.1 Earlywood

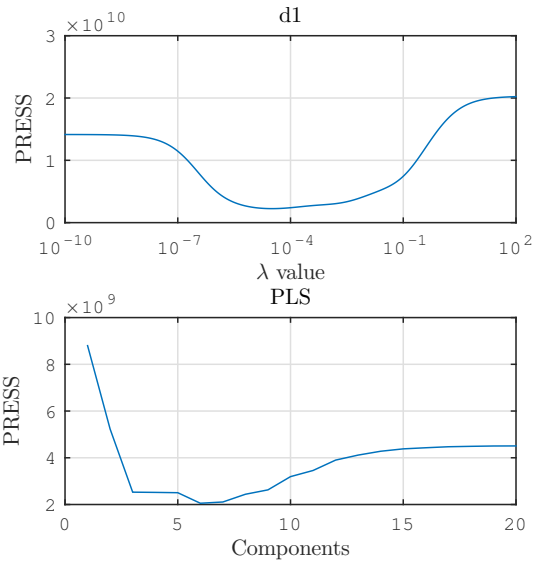
```
load /home/smeland/Documents/Datasett_master/uv_pre_v3
spectra = uv_means.early;
[n, ~] = size(spectra);
aug_spectra = [ones(n,1), spectra];
response = uv_log_sort(:,3);
```




5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
RMSEC
ridge
d1
PCR
PLS
Method


×10^6
4
2
0
−2
−4
−6
−8
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS


ridge
×10^10
PRESS
λ value

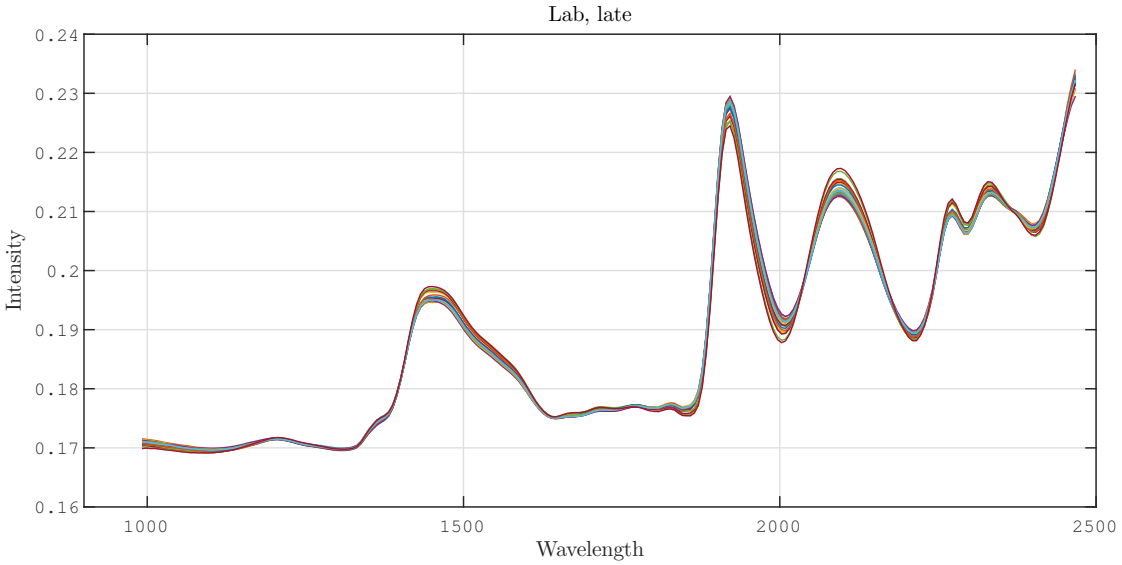

d1
×10^10
PRESS
λ value

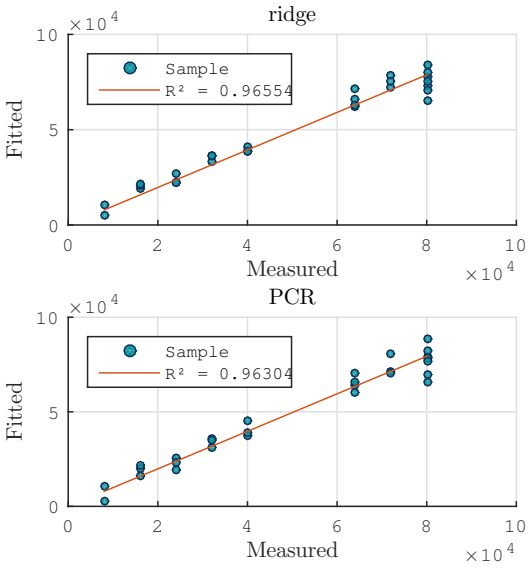
PCR
×10^10
PRESS
Components

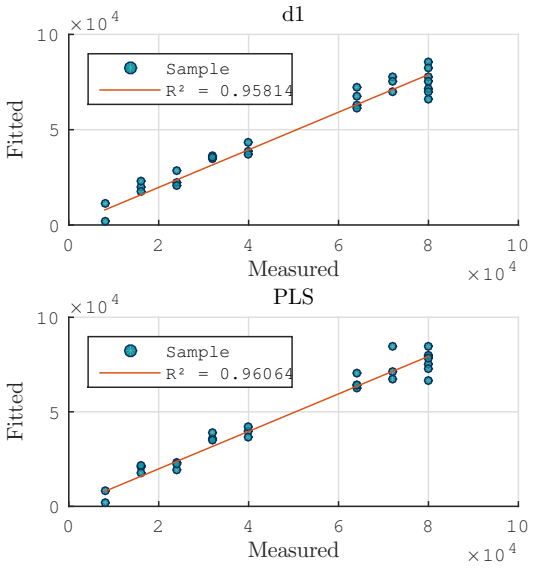
PLS
×10^9
PRESS
Components

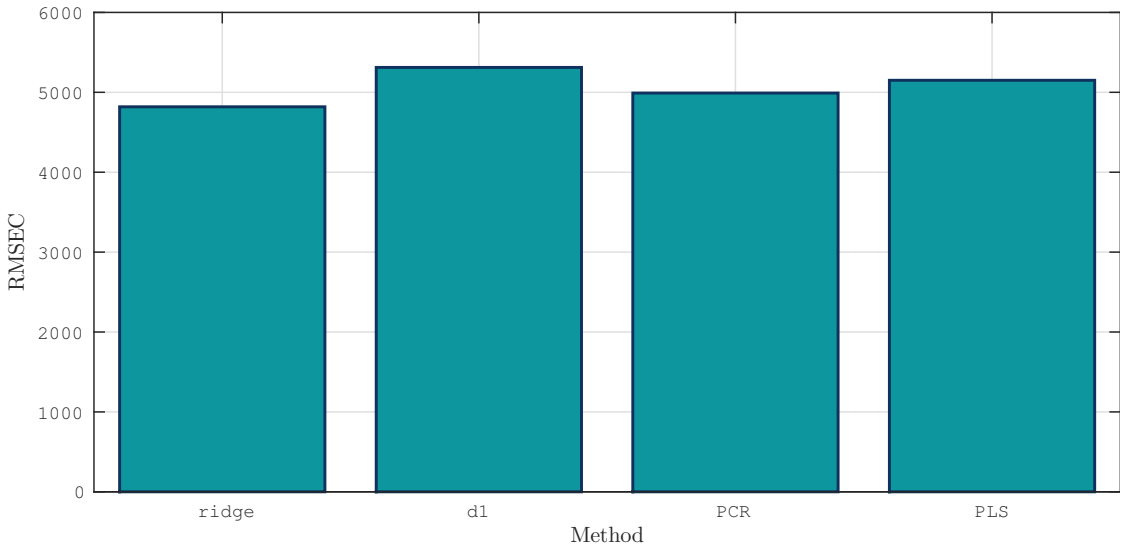
A.2.2 Latewood

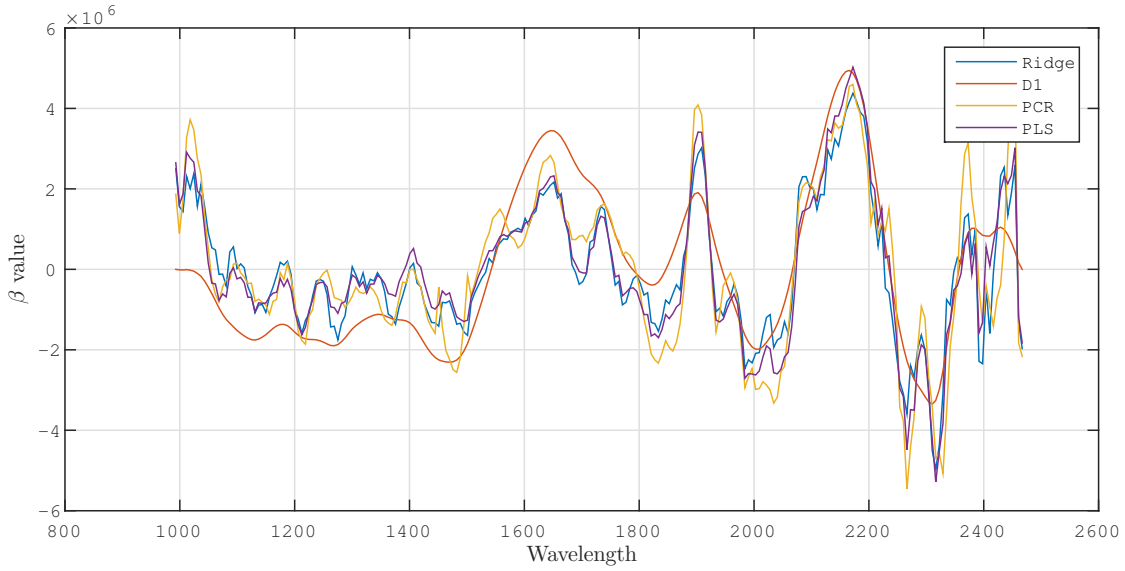
```
load /home/smeland/Documents/Datasett_master/uv_pre_v3
spectra = uv_means.late;
[n, p] = size(spectra);
aug_spectra = [ones(n,1), spectra];
response = uv_log_sort(:,3);
```

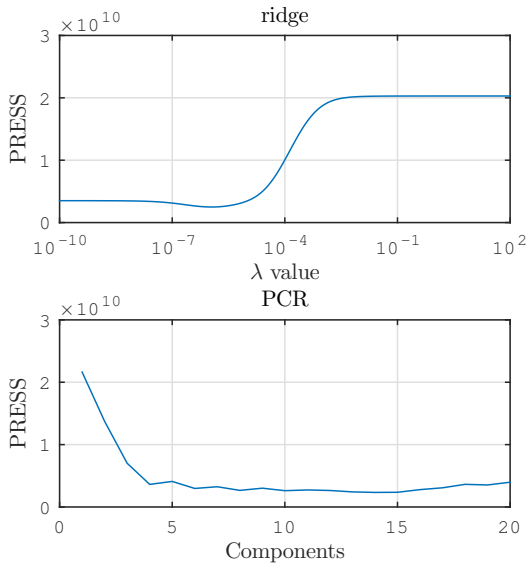


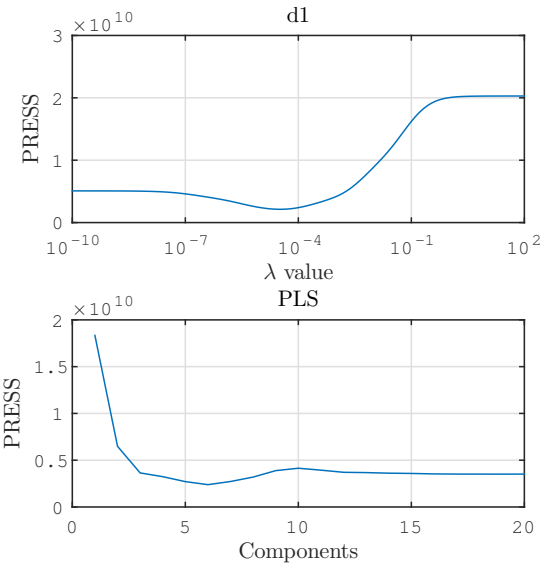





6000
5000
4000
3000
2000
1000
0
RMSEC
ridge
d1
PCR
PLS
Method


×10^6
6
4
2
0
−2
−4
−6
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS


ridge
×10^10
PRESS
λ value

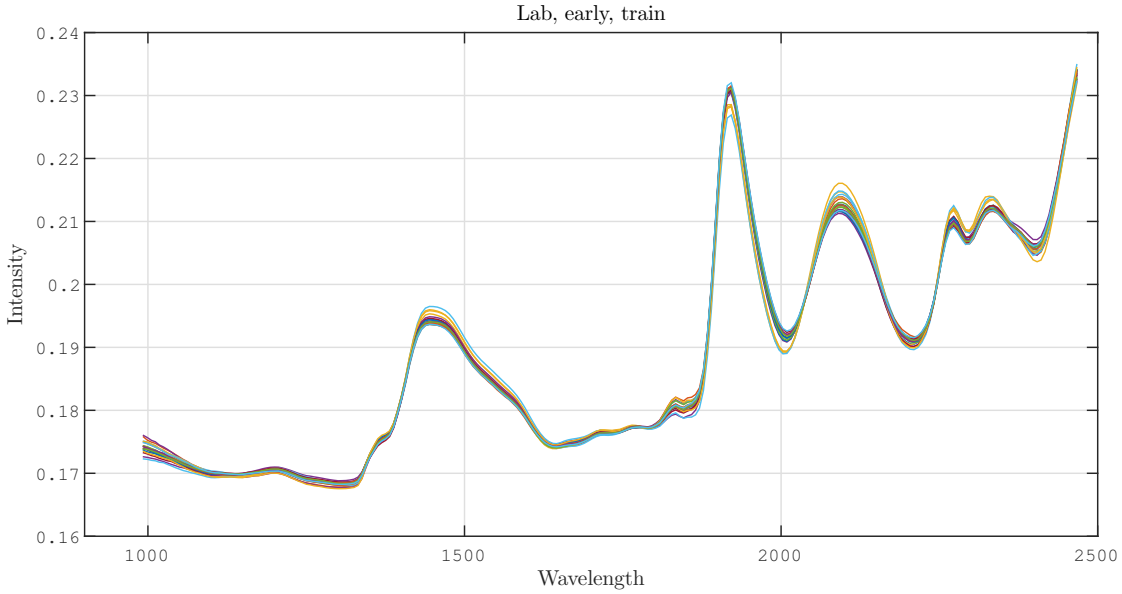

d1
×10^10
PRESS
λ value

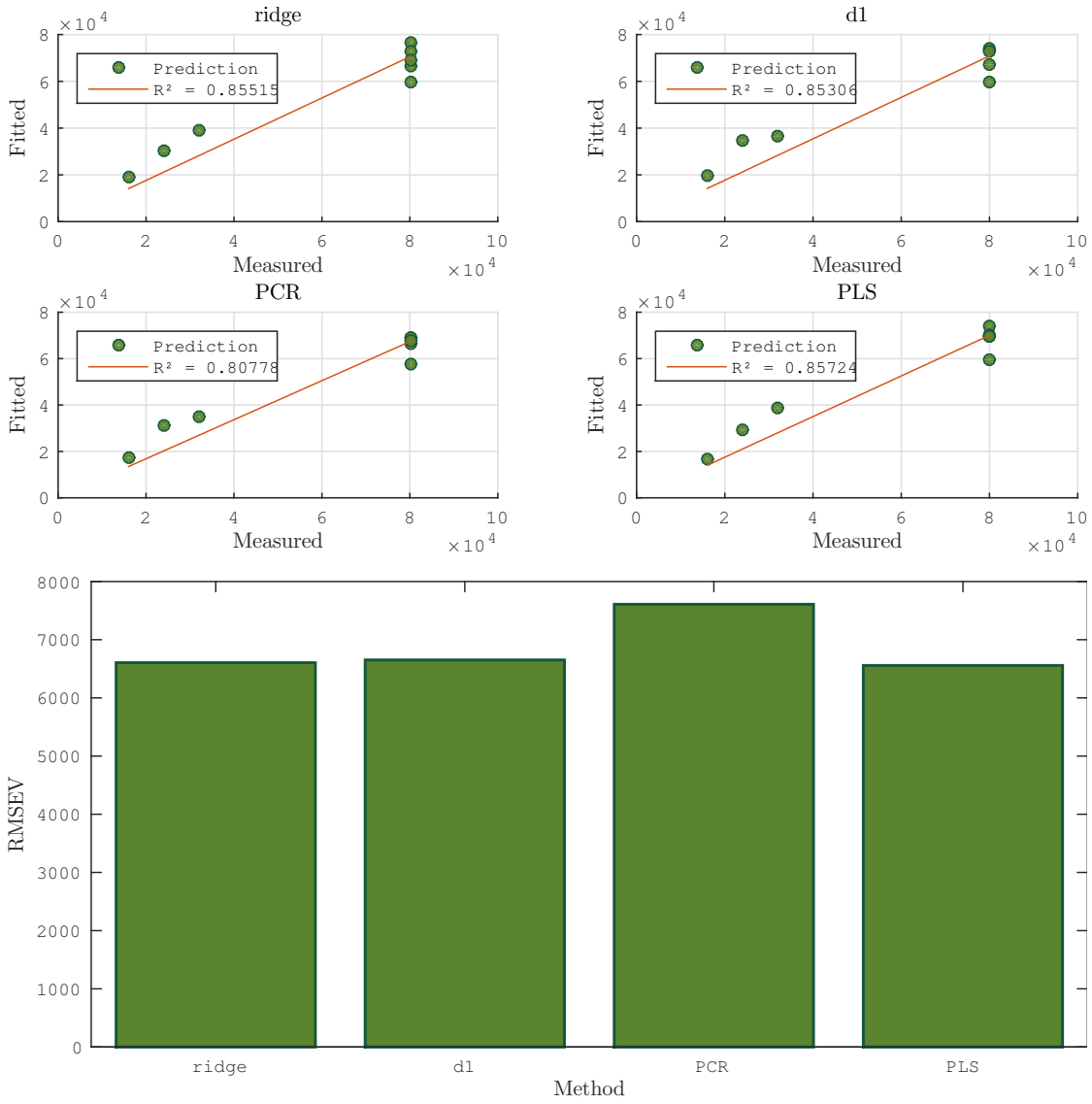
PCR
×10^10
PRESS
Components

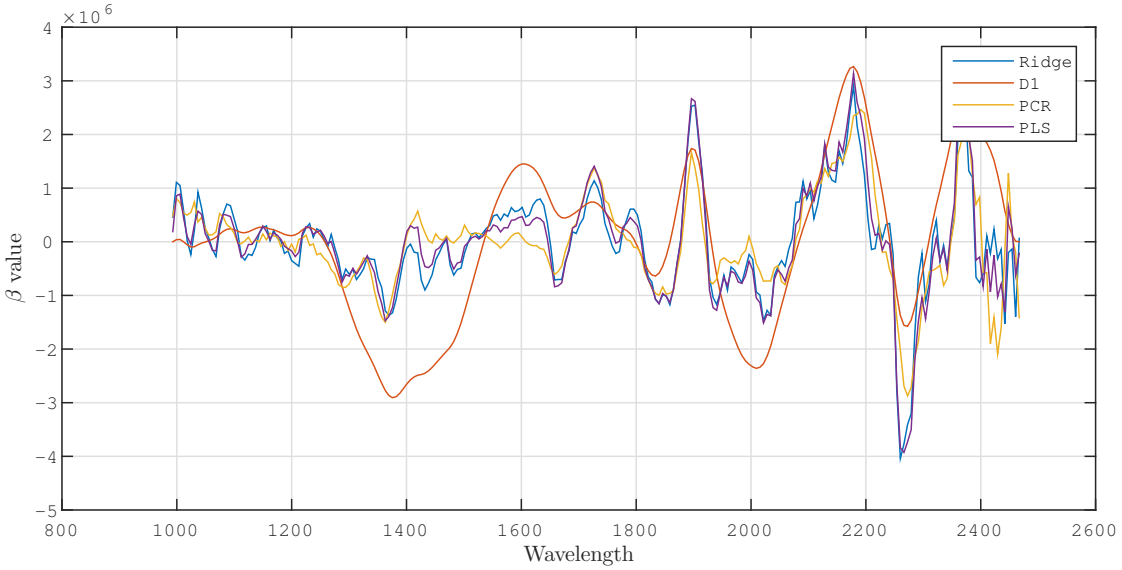
PLS
×10^10
PRESS
Components

A.2.3 Earlywood, validation

```
clear, clc, close all
load /home/smeland/Documents/Datasett_master/uv_pre_v3


spectra = uv_means.early;
response= uv_log_sort(:,3);
[m, ~] = size(spectra);

split = round(m*0.7);
seq = randperm(m);
spectra_train = spectra(seq(1:split),:);
spectra_test = spectra(seq(split+1:end), :);
response_train = response(seq(1:split));
response_test = response(seq(split+1:end));

aug_spectra = [ones(size(spectra_test,1),1),...
spectra_test];
n = size(spectra_train, 1);
```

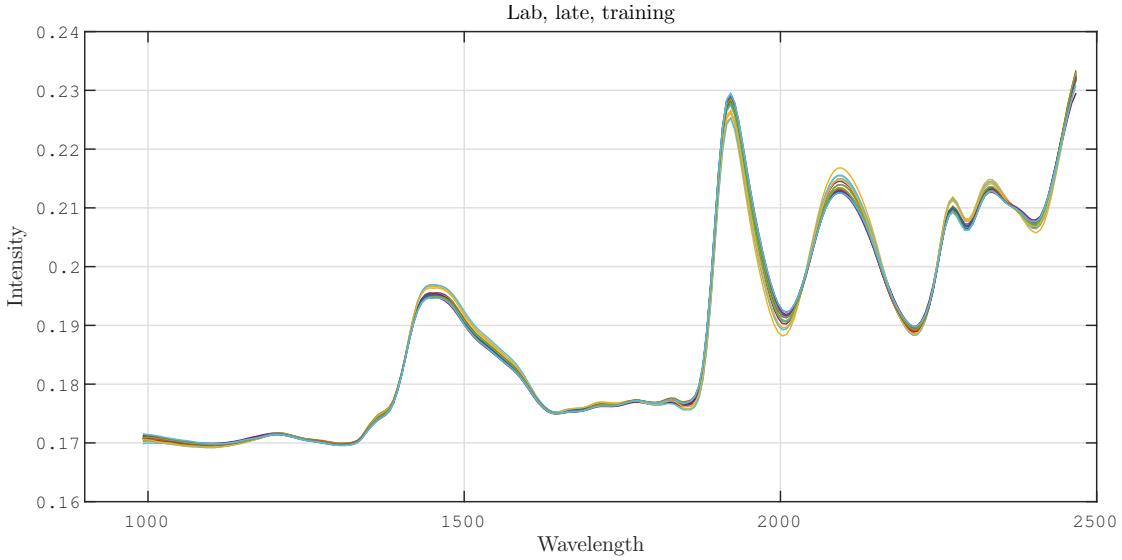


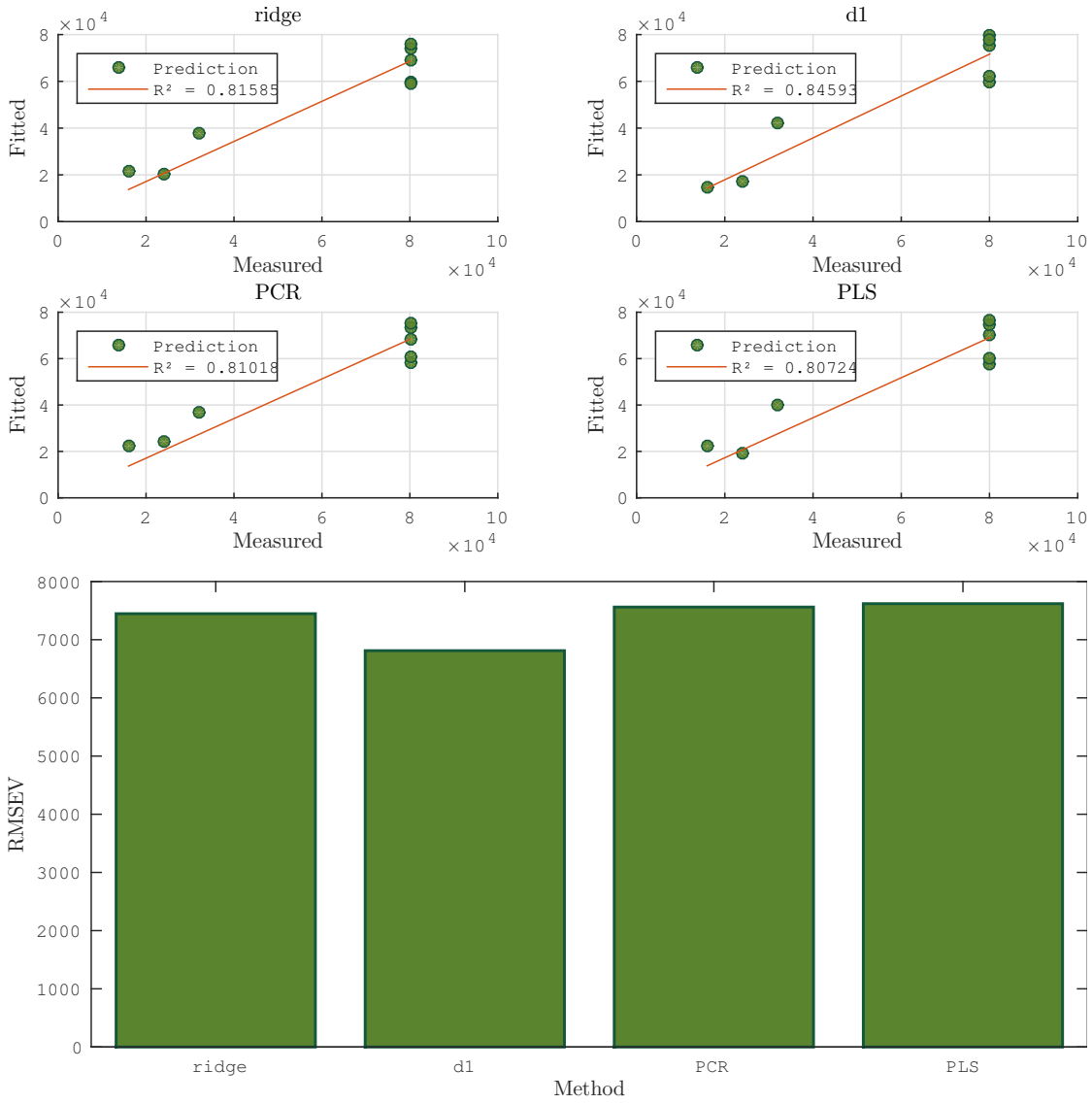

ridge
Prediction
R² = 0.85515
Fitted
Measured
×10^4
d1
Prediction
R² = 0.85306
Fitted
Measured
×10^4
PCR
Prediction
R² = 0.80778
Fitted
Measured
×10^4
PLS
Prediction
R² = 0.85724
Fitted
Measured
×10^4
RMSEV
ridge
d1
PCR
PLS
Method

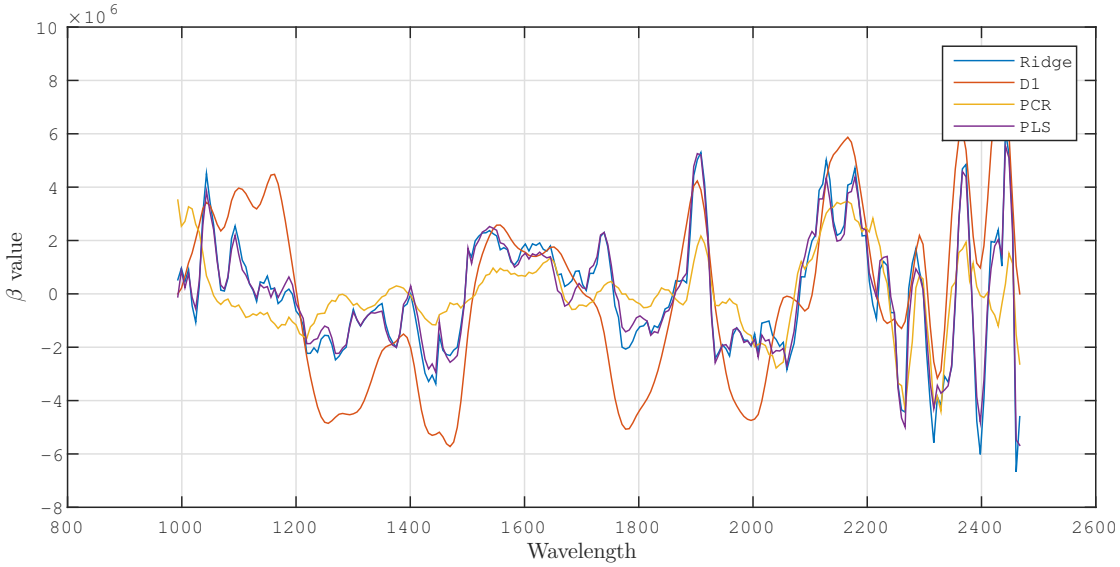

×10^6
4
3
2
1
0
−1
−2
−3
−4
−5
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS

A.2.4 Latewood, validation

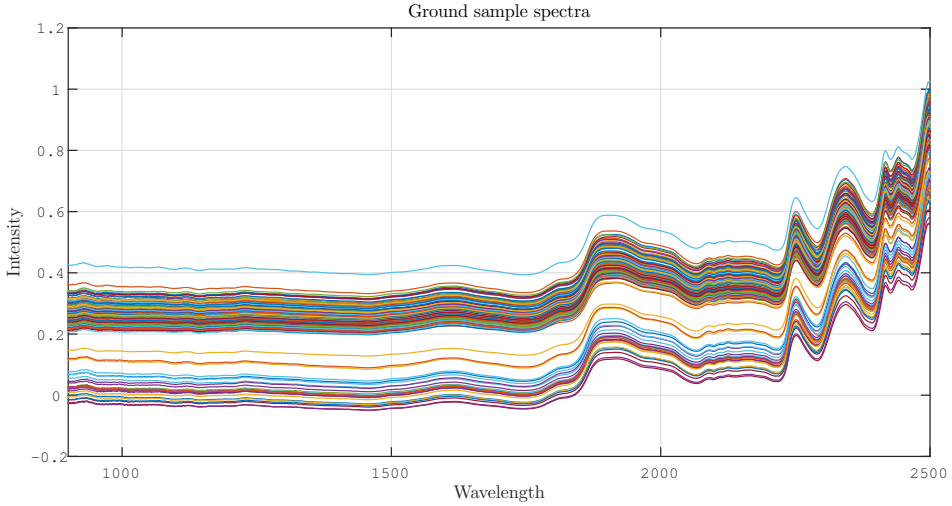
```
spectra = uv_means.late;
response= uv_log_sort(:,3);
[m, ~] = size(spectra);

split = round(m*0.7);
seq = randperm(m);
spectra_train = spectra(seq(1:split),:);
spectra_test = spectra(seq(split+1:end), :);
response_train = response(seq(1:split));
response_test = response(seq(split+1:end));

aug_spectra = [ones(size(spectra_test,1),1),...
spectra_test];
n = size(spectra_train, 1);
```

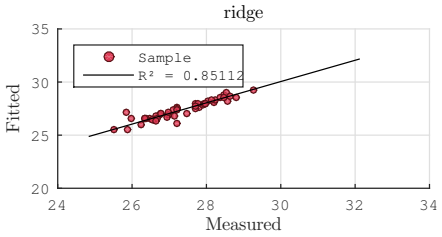


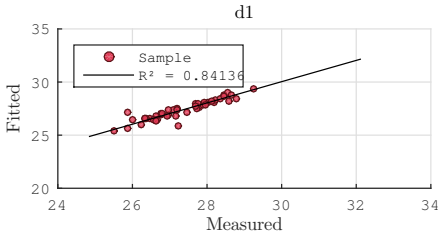

ridge
Prediction
R² = 0.81585
Fitted
Measured
×10^4
d1
Prediction
R² = 0.84593
Fitted
Measured
×10^4
PCR
Prediction
R² = 0.81018
Fitted
Measured
×10^4
PLS
Prediction
R² = 0.80724
Fitted
Measured
×10^4
RMSEV
ridge
d1
PCR
PLS
Method

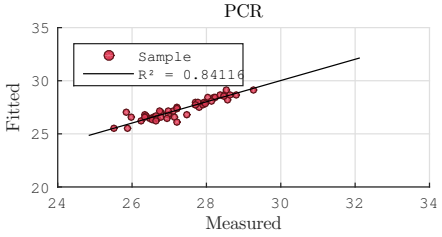

×10^6
10
8
6
4
2
0
−2
−4
−6
−8
β value
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
Wavelength
Ridge
D1
PCR
PLS

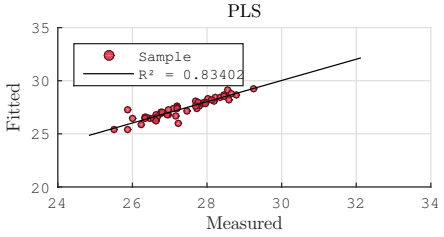
A.3 Ground wood samples

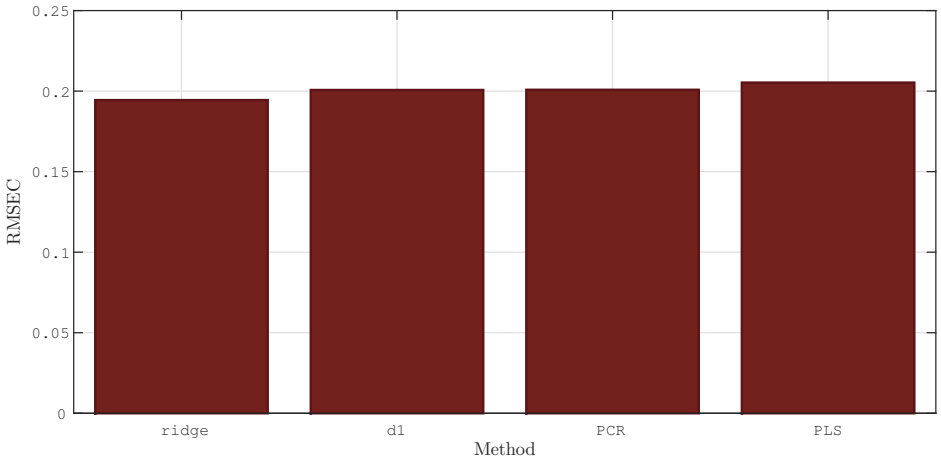


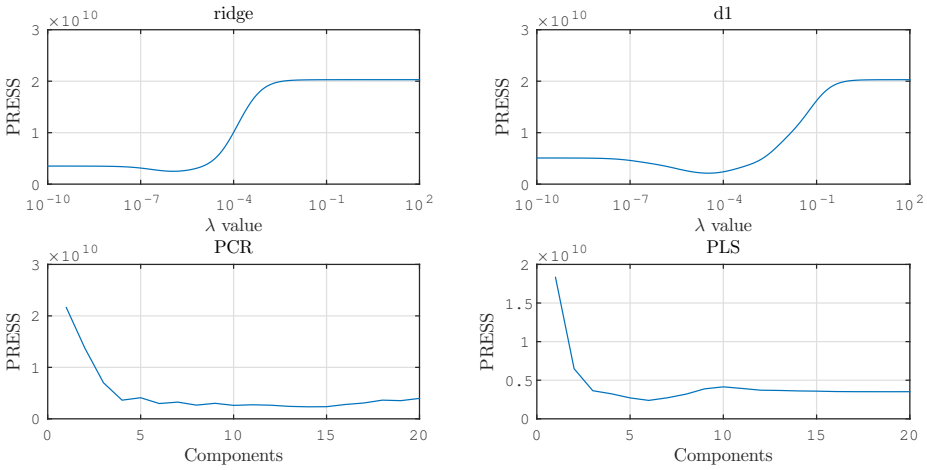


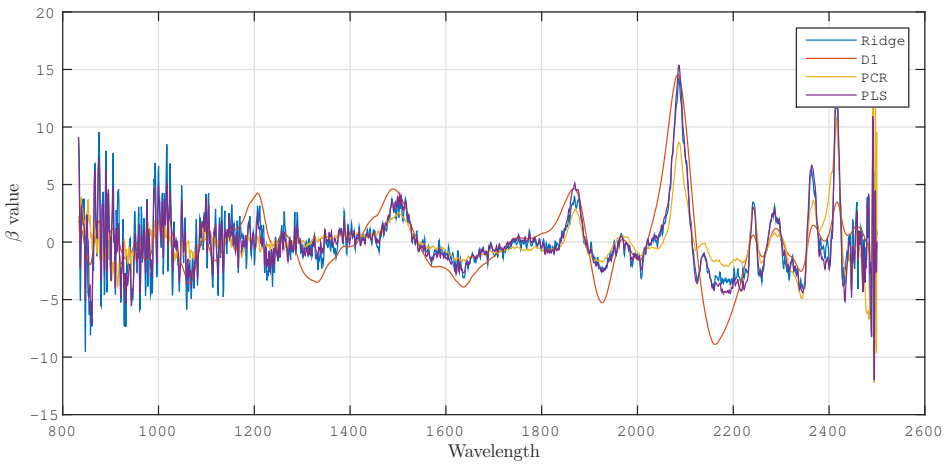







RMSEC
0.25
0.2
0.15
0.1
0.05
0
ridge
d1
PCR
PLS
Method


ridge
d1
PCR
PLS
PRESS
λ value
Components


Ridge
D1
PCR
PLS
β value
Wavelength

B Preprocessing

```
clear; clc; close all
load E:\Dokumenter\MATLAB\Datasett_master\outside_v2
scutoff = 10; ecutoff = 10; % Cut-off values for wavebands

images = outside_samples;
[nrows, ncols, nbands, nsamples] = size(images);
nbandstrim = nbands-scutoff-ecutoff;
wave_bin_size = 6.3; % According to the hdr files
wavelengths = linspace(930 + scutoff*wave_bin_size,...
 2530 - ecutoff*wave_bin_size, nbandstrim);
%% Choice of values

% Values for pre-processing
prepro.nterms = 4; % Number of terms for emsc
prepro.window = 5; % Window size for smoothing
prepro.polynom = 1; % Savitzky-Golay polynomial degree
prepro.deriv = 0; % Derivatives for smoothing

%% Create data structures
% Containers
outside_means.early = zeros(nsamples, nbandstrim);
outside_means.late = zeros(nsamples, nbandstrim);
masks.early = zeros(nrows, ncols, nsamples);
masks.late = zeros(nrows, ncols, nsamples);
emsc_params = zeros(nrows*ncols, prepro.nterms, nsamples);
post_scores = zeros(nrows*ncols, nsamples);
nmasked.late = zeros(1, nsamples);
nmasked.early = zeros(1,nsamples);
maskidx.early = zeros(nrows*ncols, nsamples);
maskidx.late = maskidx.early;
smoothed = zeros(nrows*ncols, nbands, nsamples);
maskidx.early = cell(1, nsamples);
maskidx.late = maskidx.early;

smooth.window =
[floor(prepro.window/2) -floor(prepro.window/2)];


%% Preprocessing
% Create masks and calculate means

for sample = 1:nsamples
    im = images(:,:,:,sample);
    [im_score, ~, ml, ~, msl, ~, maskidx_late] =
make_mask(im,...
        'both', [0.5 0.2],...
        [0.45 0.75], 172);
```

```
    [~, me, ~, mse, ~, maskidx_early, ~] =
make_mask(im,...
       'both', [0.3 0.2],...
       [0.25 0.85], 172);
    post_scores(:, sample) = im_score;

    maskidx.early{sample}= maskidx_early;
    maskidx.late{sample} = maskidx_late;

    masks.early(:,:,sample) = mse;
    masks.late(:,:,sample) = msl;

    nmasked.early(sample) = sum(me);
    nmasked.late(sample) = sum(ml);
end

% Log transform
images = -log10(images);

% Smoothing and SG derivatives
for sample = 1:nsamples
    im = htdeadwavelengths(images(:,:,:,sample), 6, 3, 0);
    smoothed(:,:,sample) = filpoly(reshape(
    im, nrows*ncols, nbands),...
[], smooth.window, prepro.polynom,prepro.deriv);
end

% EMSC
smoothed = smoothed(:, 1+scutoff:end-ecutoff,:);
nbandstrim = size(smoothed,2);
[corrected.images, corrected.params, par_names, corr.model] =
hyspec_emsc(reshape(smoothed, nrows, ncols, nbandstrim, nsamples));
corrected.spectra = reshape(corrected.images, nrows*ncols, nbandstrim, nsamples);

for sample = 1:nsamples
    outside_means.early(sample, :) = mean(
    corrected.spectra(maskidx.early{sample}, :, sample));
    outside_means.late(sample, :) = mean(
    corrected.spectra(maskidx.late{sample}, :, sample));
end

% For montage of images
scores = reshape(post_scores, nrows, ncols, nsamples);
masks_mosaic.early = reshape(masks.early,nrows, ncols, 1, nsamples);
masks_mosaic.late = reshape(masks.late,nrows, ncols, 1, nsamples);
%%


%% Plot result
```

```
figure;
imagesc(scores(:,:,49))
title('Score values, second component')
colorbar
figure;
montage(masks_mosaic.early);
title('Mask for early wood')
figure;
montage(masks_mosaic.late)
title('Mask for late wood')

% Sample corrected spectra
figure;
sample_no = 3;
to_show = randsample(maskidx.early{sample_no}, sample_no);
early = corrected.spectra(maskidx.early{sample_no},:,sample_no);
plot(linspace(992, 2468, nbandstrim), early')
title(['Sample corrected spectra, image no.
' num2str(sample_no) ' (10 random earlywood pixels)'])
xlabel('Wavelength'); ylabel('Intensity')

% Mean spectra
figure('Name', 'Early')
plot(outside_means.early')% linspace(992, 2468, nbandstrim),
title('Early wood')
xlabel('Wavelength'); ylabel('Intensity')
figure('Name', 'Late')
plot(linspace(992, 2468, nbandstrim),outside_means.late')
title('Late wood')
xlabel('Wavelength'); ylabel('Intensity')
```

C Functions and scripts

C.1 Import

Script for importing and matching all the images and respective response variables

```
%% Read and import images
% Read and convert .raw files to matlab matrices, and get filenames for the
% for the files to be used to identify sample numbers.
clear, clc

image_folder_path = '/home/smeland/Documents/Datasett_master/strips_post';
white_balance = '/home/smeland/Documents/Datasett_master/whitebalance/whitebalance-1';

% Values for cropping
xlow = 120; xhigh = 220;
ylow = 185; yhigh = 285;

[filenames, varnames] = read_folder(image_folder_path);
images = read_raws(filenames(1:n), white_balance, [xlow xhigh ylow yhigh]);

%% Sort response data to correspond to sample spectra

% UV chamber

% Convert cell array of strings to vector of integers
varnames = cellfun(@str2num, varnames);
varnames = varnames';

uv_log = csvread('/home/smeland/Documents/Datasett_master/uv_log');

% Match images and response variable
[~, ia_uv, ib_uv] = intersect(uv_log(:,1), varnames);
uv_samples = all_samples(:,:,:,ib_uv);
varnames = varnames(ib_uv);
uv_log = uv_log(ia_uv,:);

% Sort in ascending order in terms of total UV exposure
[~, idxsort] = sort(uv_log(:,3));
varnames_sort = varnames(idxsort);
uv_log_sort = uv_log(idxsort,:);
uv_samples = uv_samples(:,:,:,idxsort);


% Outside samples
load outside_responses
[filenames, varnames] = read_folder(image_folder_path);
varnames = cellfun(@str2num, varnames);
varnames = varnames';
```

```
[common, ia_out, ib_out] = intersect(outside_response(:,1), varnames);
outside_samples = all_samples(:,:,:,ib_out);
varnames = varnames(ib_out);
outside_response = outside_response(ia_out,:);
[x, idxsort] = sort(outside_response(:,4));
outside_response_sort = outside_response(idxsort,:);
varnames = varnames(idxsort);
outside_samples = outside_samples(:,:,:,idxsort);
```

Function for getting a cell object of file names from a specified folder.

```
function [filenames, varnames] = read_folder(folder_path)
%READ_FOLDER Generate a cell array containing file names of .raw images in a folder.

    files = dir(fullfile(folder_path, '*.raw'));
    files = {files.name};
    filenames = strrep(files, '.raw', ''); %filenames = filenames(1:n);
    varnames = strrep(filenames, '_post-1', '');
end
```

Function for reading .raw images from a location, identified by a cell containing the file names. Outputs a Matlab 4D array object.

```
function image_array = read_raws(filenames, whitebalance, crop)
%READ_RAWS Create image object from a list of .raw files.
% Creates a 4D array from color-corrected .raw files. If only one file is
% supplied, a 3D array is created instead.
%
% INPUT
% ===================================
% filenames:    A cell object containing filenames of images, extension
%               excluded.
% whitebalance: Name of the image to be used for color correction.
% crop          The upper and lower bounds for columns and rows to be
%               cropped, respectively.

    white = load_and_subtract(whitebalance, crop);

    % Get size from the first image
    im1 = corrcrop2(filenames{1}, white, crop);
    [nrows, ncols, nbands] = size(im1);
    image_array = zeros(nrows, ncols, nbands, length(filenames));

    if length(filenames) > 1
        for i = 2:length(filenames)
            image_array(:,:,:,i) = corrcrop2(filenames{i}, white, crop);
```

```
        end
        image_array(:,:,:,1) = im1;
    else
        image_array = im1;
    end
end
```

The function used in `read_raws` to convert each image.

```
function [corrected_image, cropped_image] = load_and_subtract(filename, varargin)
%LOAD_AND_SUBTRACT Load and correct a .raw image.
%
% Takes a .raw image and corresponding .hdr file and outputs a black-level
% corrected image in .mat format.
%
% INPUT
%     filename:   Name of the file to be corrected, extension excluded.
%
%     varargin:   Two additional arguments can be supplied. Each one must
%                 contain two scalars. The first one defines the first and last row of
%                 the original image to be included, and the second defines the columns.
%                 If one of the arguments are empty ([]), the image is not cropped along that
%                 axis.
%
%     If any additional arguments are included, both must be included.
%
% OUTPUT
%     corrected_image:    The (cropped) corrected image.
%     cropped_image:      The cropped image, without correction


  % Read the header file and import the raw file
   header = readenvihdr([filename '.hdr']);
   original_image = multibandread([filename '.raw'],...
      [str2num(header.lines),...
      str2num(header.samples),...
      str2num(header.bands)],...
      'uint16',...
      0,...
      header.interleave,...
      'ieee-le');

   cropped_image = original_image;

  [n_rows_original, n_columns_original, n_bands_original] = size(original_image);
  [n_rows, n_columns, n_bands] = size(cropped_image);

  corrected_image = zeros(n_rows, n_columns, n_bands);
```

```
  % Extract mean values for rows in the black reference area
  % (assumes the black area to be in the bottom of the image).
  %mean_black_value = squeeze(mean(cropped_image(n_rows_original-9:n_rows_original, :, :), 1)); %
  mean_black_value = mean(cropped_image(n_rows_original-9:n_rows_original, :, :), 1);

  % Subtract mean black values from the image.
  mean_black_array =  repmat(mean_black_value, n_rows,1,1);

  corrected_image = cropped_image - mean_black_array;
end
```

Function used in load_and_subtract to crop and perform white-level correction.

```
function corrected = corrcrop2(image, white, varargin)
    %CORRCROP2 Corrects and crops an image, with supplied white balance.
    % If a third argument is used, it is a vector of four values, with lower and
    % higher column values and lower and higher row values, in that order.
    corrected_image = load_and_subtract(image);
    wb = white;
    if isempty(varargin)
        wb = wb;
    else
        wb = wb(varargin{1}(3): varargin{1}(4),varargin{1}(1):varargin{1}(2), :);
    end
    if isempty(varargin)
        cropped_image = corrected_image;
    else
        cropped_image = corrected_image(varargin{1}(3): varargin{1}(4),varargin{1}(1):varargin{1}
    end

    [nrows, ncols, nbands] = size(cropped_image);
    corrected = cropped_image./wb;
end
```

C.2 Masking

Function for performing masking on an image, based on PCA score values.

```
function [im_scores, varargout] = make_mask(image, tail, qvalue, exclude, band)
    %MAKE_MASK Create a logical mask from image.
    %
    % Creates mask(s) from an image, based on SVD. Which part of the image
    % and the quantile values can be chosen, as well as a cut-off for
    % insensities in the image.
    %
    % INPUT:
    % ===========================
    % image:        The image, in the form of a matlab matrix.
    % tail:      Whether to keep lower, upper, or both scores ('lower'|'upper'|'both')
    % qvalue:       Quantile for thresholding. Can be a two-element vector
    %               if thresholding both tails, else the same value is used
    %               for both.
    % exclude:      Values for exluding background. A two-element vector indicating
    %               lower and upper cutoff values. Intensities below or above this
    %               are excluded. The values are compared to intensities in the
    %               first waveband.
    % band:         The band on which to perform cut-off
    %
    % OUTPUT:
    % ===========================
    % im_scores:    Score values on which thresholding is performed.
    %
    % Following arguments in this order. If 'tail' is set to 'both', two of
    % each will result, one for each tail (lower, then upper):
    % ---------------------------
    % mask:         A logical vector of pixels to be masked (for reshaped, long
    %               form image).
    % mask_surf:    Masked pixel surface, for visual inspection.
    % mask_idx:     Index of pixels to be masked. When used as an
    %               index vector, only values at these index values will be
    %               included.
    %
    % Example: [score, mask1, mask2, masksurf1, masksurf2, midx1, midx2] =
    %               make_mask(image, 'both', [0.25 0.30], [0.15 0.85])


    im_unfolded = reshape(image, size(image,1)*size(image,2), size(image,3));
    mask = image(:,:,band) > exclude(1) & image(:,:,band) < exclude(2);
    mask_1 = reshape(mask, size(image,1)*size(image,2),1);
    [~, ~, V] = svds(im_unfolded(mask_1,:), 2);
    im_scores = im_unfolded*V(:,2);

    switch tail
```

```
        case 'both'
            if length(qvalue) == 1
                qvalue(2) = qvalue;
            end
            mask_2 = mask_1;
            mask_1(im_scores < quantile(im_scores, 1-qvalue(1))) = 0;
            varargout{1} = mask_1;
            varargout{3} = reshape(mask_1, size(image,1), size(image,2));
            varargout{5} = find(mask_1 == 1);

            mask_2(im_scores > quantile(im_scores, qvalue(2))) = 0;
            varargout{2} = mask_2;
            varargout{4} = reshape(mask_2, size(image,1), size(image,2));
            varargout{6} = find(mask_2 == 1);

        case 'lower'
            if length(qvalue) > 1
                error('Only one qvalue is accepted for tail value "lower".');
            else
                mask_1(im_scores > quantile(im_scores, qvalue)) = 0;
                varargout{1} = mask_1;
                varargout{2} = reshape(mask_1, size(image,1), size(image,2));
                varargout{3} = find(mask_1 == 1);
            end

        case 'upper'
            if length(qvalue) > 1
                error('Only one qvalue is accepted for tail value "upper".');
            else
                mask_1(im_scores < quantile(im_scores, 1-qvalue)) = 0;
                varargout{1} = mask_1;
                varargout{2} = reshape(mask_1, size(image,1), size(image,2));
                varargout{3} = find(mask_1 == 1);
            end
    end
end
```

C.3 Regression

C.3.1 Ridge regression

The code for performing the ridge regression, utilizing fast LooCV. Programmed by Ulf Geir Indahl.

```
function [press, bcoefs, b, lambda, h, U, s, V] = RregsLooCV(X,y,lambdas)
%RREGSLOOCV Search and optimize ridge regression solution.
% Fast Ridge-regression routine finding the best choice of regularization
% paramter value lambda wrt to the PRESS-statistic (leave-one-out crossvalidation).
%
% Programmed by Ulf Geir Indahl, NMBU
%
% INPUT
% ==================================================
% X:        Predictor variables, a matrix.
% y:        Response variable, a vector.
% lambdas:  Choices of regularization parameters values, a vector.
%
% OUTPUT
% ==================================================
% PRESS:    PRESS values for each choice of regularization parameter. A
%           vector.
% bcoefs:   Least squares solution coefficients for the system, constant
%           term (intercept) not included. A vector.
% b:        Same as bcoefs, but with constant term included.
% lambda:   Optimal regularization parameter value, the one minimizing
%           PRESS value. A scalar.
% h:        leverage values for the best model.
% U:        Right singular vectors of centered X. A matrix.
% s:        Singular values of centered X. A vector.
% V:        Left singular values of centered X. A matrix.

[n,p] = size(X);                % Size of data matrix
k = length(lambdas);            % Number of parameter values to be tested
H = zeros(n,k);                 % Matrix of leverage values
mX = mean(X);  my = mean(y);    % mean of X and y
X = X-ones(n,1)*mX; y = y-my;   % Centering of X and y
[U, S, V] = svd(X,'econ');      % SVD of centered X-data (PCA of X).
s = diag(S); s2 = s.^2;
bcoefs = zeros(p,k);
for i = 1:k
bcoefs(:,i) = V*((s./(s.^2+lambdas(i))).*(U'*y));
end
% For more speed use this for the collection of regression coeffs:
% bcoefs = V*(repmat(s.*(U'*y),1,k)./(repmat(s.^2,1,k) + repmat(lambdas, min(n,p),1)));

% The associated leverage and "press"-values:
```

```
for i = 1:k
H(:,i) = sum(bsxfun(@times,U,(s./sqrt(s2+lambdas(i)))').^2,2)+1/n;
end
press = sum( ((repmat(y,1,k)-X*bcoefs)./(1-H)).^2 )'; % The LooCV sum of squared errors (LooCV-SS
% ------------------------------------------------------
% Identify and look up parameters of the LooCV-best model:
% ------------------------------------------------------
[~, id] = min(press); lambda = lambdas(id);  % lambda-value chosen by press
b = [my-mX*bcoefs(:,id); bcoefs(:,id)];      % Regression coeffs with constant term
h = H(:,id);                                 % The associated leverages
```

C.3.2 Tikhonov Regularization, derivatives operator

```
function [press, bcoefs, b, lambda, h, U, s, V, VH] = TregsLooCV(X,y,lambdas,type)
%TREGSLOOCV Search and optimize Tikhonov regularized regression alternate
%operators
% Finds the optimal solution for a least squares problem using Tikhonov
% regularization with either first- or second derivatives operators. That
% is, finds optimal solutions with respect to the length of the vector of
% either the first or the second derivatives of the coefficient vector.
% Programmed by Ulf Geir Indahl, NMBU
%
% INPUT
% ==================================================
% X:        Predictor variables, a matrix.
% y:        Response variable, a vector.
% lambdas:  Choices of regularization parameters values, a vector.
% type:     Wether to use first or second derivatives operator.
%           Options: 1|2
%
% OUTPUT
% ==================================================
% PRESS:    PRESS values for each choice of regularization parameter. A
%           vector.
% bcoefs:   Least squares solution coefficients for the system, constant
%           term (intercept) not included. A vector.
% b:        Same as bcoefs, but with constant term included.
% lambda:   Optimal regularization parameter value, the one minimizing
%           PRESS value. A scalar.
% h:        leverage values for the best model.
% U:        Right singular vectors of centered X. A matrix.
% s:        Singular values of centered X. A vector.
% V:        Left singular values of centered X. A matrix.
% VH:       The variance of the leverage values.


[n,p] = size(X);                % Size of data matrix
mX = mean(X);  my = mean(y);    % mean of X and y
X = X-ones(n,1)*mX; y = y-my;   % Centering of X and y
if type == 1
    L = diff([eye(p);zeros(1,p)]); % L =-eye(p,p)+diag(ones(p-1,1),1);
    X = X/L;%*pinv(L);
elseif type == 2
    L = diff([eye(p);zeros(2,p)],2); %L = eye(p)-diag(2*ones(p-1,1),1)+diag(ones(p-2,1),2);
    X = X/L;
end
[U, S, V] = svd(X,'econ');      % SVD of centered X-data (PCA of X).
s = diag(S); s2 = s.^2;

% bcoefs = V*(repmat(s.*(U'*y),1,k)./(repmat(s.^2,1,k) + repmat(lambdas, min(n,p),1)));
```

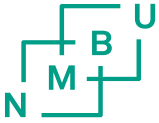
```
bcoefs = V*bsxfun(@times,s.*(U'*y), 1./bsxfun(@plus,s2,lambdas));
% The associated leverage and "press"-values (the 1/n term is included in
% the "press"-calculations):
U2 = U.^2;
H = U2*bsxfun(@times,s2,1./(bsxfun(@plus,s2,lambdas)));
VH = var(H);

% The LooCV sum of squared errors (LooCV-SSE)
% press = sum( ((repmat(y,1,k)-X*bcoefs)./(1-H-1/n)).^2 )';
press = sum(bsxfun(@times,bsxfun(@minus,y,X*bcoefs),1./((1-1/n)-H)).^2)';
% -------------------------------------------------------
% Identify and look up parameters of the LooCV-best model:
% -------------------------------------------------------
if type >0
    bcoefs = L\bcoefs;
end
[~, id] = min(press); lambda = lambdas(id);  % lambda-value chosen by press
b = [my-mX*bcoefs(:,id); bcoefs(:,id)];      % Regression coeffs with constant term
h = H(:,id)+1/n;                             % The associated leverages
```

C.3.3 PCR

```
function b = pcr(X,y,k)
% Function for computing PCR based on k principal componets. Programmed by Ulf G. Indahl
n = size(X,1);
mX = mean(X);
X = X - ones(n,1)*mX;      % Centering of X
[U, S, V] = svds(X,k);     % Extracting first k PCA-components
s = diag(S);
b = V*(s.\(U'*y));         % Same as b = V*inv(S)*(U'*y);
b = [mean(y)-mX*b; b];     % The final regression coeffs (with constant term)

%b = cumsum(bsxfun(@times,V,(s.\(U'*y))'),2);
%b = [mean(y)-mX*b;b];
```

C.3.4 PLS

```
function [b, W, T] = plsNIP(X, y, k)
% Function for computing PLS with k componets. Programmed by Ulf G. Indahl
[n, m] = size(X); mX = mean(X); my = mean (y);
X = X - ones(n,1)*mX;      % Centering of X
y = y - my;                % Centering of y
T     = zeros(n,k);             % Orthonormal scores
W     = zeros(m,k); P = W;      % Orthonormal weights and X-loadings
q     = zeros(k,1);
% ----- Initialization -----
w = X'*y; W(:,1) = w/norm(w);
t = X*W(:,1); T(:,1)=t/norm(t);
P(:,1) = X'*T(:,1); q(1) = T(:,1)'*y;
if k > 1
X = X-T(:,1)*P(:,1)';
    for a = 2:k
        w = X'*y;
        W(:,a) = w/norm(w);
        t = X*W(:,a);
        T(:,a) = t/norm(t);
        P(:,a) = X'*T(:,a);
        X = X-T(:,a)*P(:,a)';
        q(a) = T(:,a)'*y;
        y = y-T(:,a)*q(a);
    end
end
b = W/(P'*W)*q;
b = [my-mX*b; b];     % The final regression coeffs (with constant term)
% It's easy also to compute the regression coeffs for all PCR-models of <= k
% components:
% b = cumsum(bsxfun(@times,W/(P'*W),q'),2);
% b = [my-mX*b;b];
```



Norwegian University
of Life Sciences

Postboks 5003
NO-1432 Ås, Norway
+47 67 23 00 00
www.nmbu.no