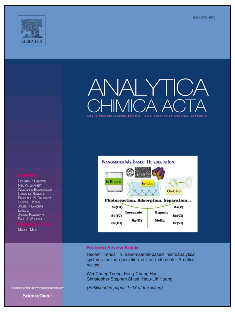


Open-Source Python Module for Automated Preprocessing of Near Infrared Spectroscopic Data

Jari Torniainen, Isaac O. Afara, Mithilesh Prakash, Jaakko K. Sarin, Lauri Stenroth, Juha Töyräs

PII: S0003-2670(20)30204-X

DOI: https://doi.org/10.1016/j.aca.2020.02.030

Reference: ACA 237462

To appear in: *Analytica Chimica Acta*

Received Date: 14 September 2019

Revised Date: 20 December 2019

Accepted Date: 12 February 2020

Credit Author Statement

Jari Torniainen: Investigation; Software; Writing - original draft
Isaac O. Afara: Software; Writing - review & editing
Mithilesh Prakash: Validation; Writing - review & editing
Jaakko K. Sarin: Validation; Writing - review & editing
Lauri Stenroth: Supervision; Writing - review & editing
Juha Töyräs: Supervision; Project administration; Writing - review & editing

Configuration file

```
[SAVGOL]
   filter_win = 101, 131
   deriv_order = 0, 1, 2
   also_skip = True
[LSNV]
   num_windows = 3, 4, 8
   also_skip = True
```

Raw spectra
Samples
400
600
800
1000
1200
Wavelength [nm]
No preprocessing
SAVGOL 101
SAVGOL 131
LSNV 3
LSNV 4
LSNV 8
SAVGOL 101 D1
SAVGOL 131 D1
SAVGOL 101 D2
SAVGOL 131 D2
SAVGOL 101 LSNV 3
SAVGOL 101 LSNV 4
SAVGOL 101 LSNV 8
SAVGOL 131 LSNV 3
SAVGOL 131 LSNV 4
SAVGOL 131 LSNV 8
SAVGOL 101 D1 LSNV 3
SAVGOL 101 D1 LSNV 4
SAVGOL 101 D1 LSNV 8
SAVGOL 131 D1 LSNV 3
SAVGOL 131 D1 LSNV 4
SAVGOL 131 D1 LSNV 8
SAVGOL 101 D2 LSNV 3
SAVGOL 101 D2 LSNV 4
SAVGOL 101 D2 LSNV 8
SAVGOL 131 D2 LSNV 3
SAVGOL 131 D2 LSNV 4
SAVGOL 131 D2 LSNV 8
nippy
Optimize
Feature extraction
Regression / Classification
Model performance
2 3 1.7
2.1 2.8 1.1
1.8 2 0.8
1.5 1.8 1.2
1.6 1.9 1.4
1.1 2 1.2
1.4 0.9
1.3 0.5
1.5 1.1
1.6 1.8
2.4
2
RMSE
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Optimal preprocessing pipeline
SAVGOL 101 D2 LSNV 4
Input
Output

Open-Source Python Module for Automated Preprocessing of Near Infrared Spectroscopic Data

Jari Torniainen[a,b], Isaac O. Afara[a,b], Mithilesh Prakash[a,b], Jaakko K. Sarin[a,b], Lauri Stenroth[a], Juha Töyräs[a,b,c]

[a]*Department of Applied Physics, University of Eastern Finland, Kuopio, Finland*
[b]*Diagnostic Imaging Center, Kuopio University Hospital, Kuopio, Finland*
[c]*School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia*

Abstract

Near infrared spectroscopy (NIRS) is an analytical technique for determining the chemical composition or structure of a given sample. For several decades, NIRS has been a frequently used analysis tool in agriculture, pharmacology, medicine, and petrochemistry. The popularity of NIRS is constantly growing as new application areas are discovered. Contrary to mid infrared spectral region, the absorption bands in near infrared spectral regions are often non-specific, broad, and overlapping. Analysis of NIR spectra requires multivariate methods which are highly subjective to noise arising from instrumentation, scattering effects, and measurement setup. NIRS measurements are also frequently performed outside of a laboratory which further contributes to the presence of noise. Therefore, preprocessing is a critical step in NIRS as it can vastly improve the performance of multivariate models. While extensive research regarding various preprocessing methods exists, selection of the best preprocessing method is often determined through trial-and-error. A more powerful approach for optimizing preprocessing in NIRS models would be to automatically compare a large number of preprocessing techniques (e.g., through grid-search or hyperparameter tuning). To enable this, we present, `nippy`, an open-source Python module for semi-automatic comparison of NIRS preprocessing methods (available at `https://github.com/uef-bbc/nippy`). We provide here a brief overview of the capabilities of `nippy` and demonstrate the typical usage through two examples with public datasets.

Keywords: Near infrared spectroscopy, Preprocessing, Chemometrics

1. Introduction

Near infrared spectroscopy (NIRS) is a widely used vibrational spectroscopic technique for quantitative evaluation of the composition and structure of a given sample. In NIRS, the target sample is illuminated with near infrared (NIR) light (750 – 2500 nm wavelength range) and the reflected and backscattered light are measured with a spectrometer. NIR-active molecular bonds in the sample absorb the incoming light at different overtone and combination spectral bands, thus producing the NIR absorbance spectrum. Compared to other infrared spectroscopy methods, NIRS has increased penetration depth and less stringent requirements for sample preparation [1]. The robustness and portability of NIRS devices enables their use as diagnostic probes outside laboratory environments [1, 2]. NIRS is a relatively old analytical technique with pertinent research spanning well over three decades. Today, NIRS is utilized heavily in multiple fields, including agriculture [3], food processing [4], pharmaceutical industry [5], and medical research [6].

The objective of a typical NIRS analysis is to relate the measured NIR spectra to a target property of the sample which is either directly or indirectly linked to its chemical composition or structure. For instance, a common example is the determination of the octane number of gasoline samples from the corresponding NIR spectrum [7]. In this example, the number of CH-bonds in hydrocarbons, like methyl and methylene, correlates with the gasoline octane rating and can, therefore, be detected using NIRS. Compared to traditional mid infrared spectroscopy, the NIR spectrum is more complex and difficult to analyse due to the presence of overlapping overtones and combination bands. The proper analysis of NIR spectral data, therefore, requires multivariate statistical models. On the other hand, the robust-nature of NIRS instrumentation enables integration of sensors for continuous monitoring of chemical processes or as portable scanners for performing measurement and analysis of solid samples on the field (e.g., soil or crops analysis). The best practices in analysing NIRS data have been the topic of active research in the field of chemometrics [1].

The canonical analysis process of regression-based NIRS models is very similar to other chemometric applications. First, a training set of data (i.e., the calibration set) is collected by measuring both the NIR spectrum and the desired target property. The calibration set should contain a relatively large number of samples (e.g., $N = 100 - 200$, estimate derived from recommendations by Burns et al.[8] and ISO 12099:2017[9]) and cover the entire current (and future) range of expected variation of the spectra and the target property. The calibration dataset is used to construct a calibration model, capable of predicting the target property of future samples. The resulting calibration model should be reported with appropriate figures of merit (root mean squared error, RMSE; coefficient of determination, R^2; etc.) [10, 11] that approximate the real-world performance.

The workflow for generating the calibration model can be divided into three main steps: preprocessing, variable selection (also known as frequency selection or feature extraction), and calibration (i.e., training the classification or regression model). Preprocessing aims to remove all sources of uninformative variance (e.g., instrumentation and scattering effects) from the measured spectrum. Typical preprocessing steps include normalization, linearization, and smoothing [12]. The variable selection aims to reduce the full feature space (i.e., different wavelengths) to a subset of most important features. Variable selection methods include various decomposition methods, such as principal and independent component analysis, sequential methods like uninformative variable elimination, and optimization techniques like genetic algorithms [13, 14]. In the last two cases, the process of variable selection is performed in tandem with the calibration step. Finally, the calibration step trains a classification/regression model which maps the extracted spectral features to explain the property of interest. Each of the three steps has multiple viable methods and configurable parameters which can be tuned for achieving the best possible predictive performance. The combination of preprocessing, variable selection, and calibration producing the model with the best performance depends heavily on the instrumentation and intended application.

As the workflow for constructing the calibration model dictates how well the technique will perform, a large number of studies have focused on optimizing this process. A substantial amount of time and effort has been used to compare different preprocessing [15, 16, 17, 12], feature extraction [14, 13, 18], and classification methods [19, 20, 21] in order to build optimal predictive NIRS models. More recently, the effectiveness of different combinations of preprocessing methods (often termed "preprocessing strategies") has been investigated. Engel et al.[22] produced the first critical review of chemometric preprocessing strategies which demonstrated that the model performance between different preprocessing strategies can vary substantially and highlighted the drawbacks in methods currently in use for selecting these strategies. A study by Gerretzen et al.[23] addressed the problem by utilizing *design-of-experiments* approach for selecting the best preprocessing strategy. This approach was later expanded to cover both preprocessing and variable selection, with the aim of improving not only the model performance but also its interpretability[24]. Earlier, Bocklitz et al.[25] presented a similar solution based on genetic algorithms for finding the optimal preprocessing strategy in Raman spectroscopy. Earlier investigations of various methods have resulted in a collective consensus of good methods in the field of NIRS research. However, in order to combine these methods to form the best possible calibration model, each part of the model (i.e., preprocessing, variable selection, and calibration) should be optimized individually for each application.

Recent advances in the field of machine learning have produced powerful methods (such as automated learning and hyperparameter tuning) for seeking further optimized models. The optimization of calibration models, however, has mainly focused on tuning the variable selection and classification/regression parts of the process with little focus on the preprocessing stage. While preprocessing has a critical impact on model outcomes, it is often not subjected to similar pruning. Likewise, the effect of different parameter values (e.g., filtering window length) on the model performance is rarely exhaustively studied or their effects reported.

To help find an effective combination of preprocessing methods, we present `nippy`: a toolbox to rapidly test different preprocessing combinations in conjunction with different machine learning methods. `nippy` is a Python 3.6+ module, open source under the MIT license, and fully compatible with the Python scientific stack (e.g., `scikit-learn` [26] machine learning module and `scipy` [27]). While nippy is intended for spectral preprocessing, it can be combined with the powerful outlier detection (e.g., elliptic envelopes, isolation forests, etc.), variable selection (e.g., variance thresholding, recursive feature elimination, etc.), and model validation (e.g., classification/regression metrics, dummy estimators, etc.) methods of the `scikit-learn` module in order to produce state-of-the-art calibration models. It implements all recommended preprocessing methods (such as scatter correction, smoothing, and computing derivatives) as well as a framework for testing the effects of different combinations of preprocessing methods and their parameters. To be clear, we are not aiming to produce a comprehensive benchmarking of different preprocessing methods but rather provide a tool for researchers and practitioners to find an effective preprocessing strategy for their data.

We provide here a brief overview of the module and its capabilities and demonstrate the typical usage through two example cases using previously published datasets. Full documentation and the source code of `nippy` are also available at `https://github.com/uef-bbc/nippy`.

2. Materials and methods

2.1. Overview of `nippy`

The *near infrared preprocessing in Python toolbox* (abbreviated as `nippy`) is a Python module that enables the user to easily define multiple different preprocessing strategies for NIRS data. The module is written in Python 3.6 and should, therefore, be compatible with all later versions. The module is also fully compatible with the data format used by the `scikit-learn`-module, enabling easy integration with powerful machine learning methods. A standard configuration file provided with the modules can be used to install `nippy` automatically using the Python package management system (i.e., `pip` or `conda`).

The `nippy` module consists of two main components: a preprocessing module and a handler function. The preprocessing module collects all currently accepted preprocessing methods under one file. Inputs to these methods have been standardized and the general naming convention from NIRS literature has been used when possible. To improve performance and reduce the burden of maintenance, the best methods from existing larger modules (such as `sklearn` and `numpy`) have been used. The modular structure of preprocessing methods also enables the user to implement custom preprocessing methods if they so desire. An overview of the preprocessing methods currently provided by `nippy` can be found in the section 2.2. The handler component of `nippy` constructs preprocessing pipelines per users requests. The user provides a list of methods and a list of parameters for each method through a configuration file. The handler parses the configuration and produces all possible permutations which can then be executed as a combination of appropriate functions of the preprocessing module. The default usage of the handler function is covered in section 2.3.

2.2. Preprocessing methods

In total, six different preprocessing categories are supported in `nippy` and they are performed in the following order.

1. Clipping
2. Scatter correction
3. Smoothing
4. Derivatives
5. Trimming
6. Resampling

It should be noted, that the order in which the preprocessing operations are performed can have an effect on the model performance. The order selected for `nippy` represents the consensus among the authors and is based on the order typically found in literature. If needed, the order of the operations can be changed by modifying the source-code (details for this can be found in the online documentation). Some of the above-mentioned operations (such as scatter correction) can contain multiple alternative methods. The handler component of `nippy` will make sure that only one operation from each category is included in each of the preprocessing pipes.

Clipping operation removes or substitutes data points with values exceeding the user-defined threshold. This method is intended to eliminate short spike-like artifacts that might otherwise distort further analysis. As this method induces short discontinuities in the data, it is advisable to only use it for gross artifacts. Larger noise-contaminated regions can be removed with the trimming operation.

Scatter correction methods aim to counter the particle size effect. In `nippy`, the three standard methods (standard normal variate, SNV [28]; multiplicative scatter correction, MSC [29]; and normalization [12]) have been implemented. The non-parametric version of SNV, known as robust normal variate (RNV) is also implemented [30]. In SNV, the correction is performed according to the mean and standard deviation of the spectrum and in RNV (which is intended for more noisy data) according to the median value and user-specified inter-quartile interval. In addition, localized version of SNV (LSNV) [31] is also included, where SNV operation is performed piece-wise inside user-defined spectral windows. In MSC, correction is only possible to the mean of the spectra. The extended version of MSC (EMSC) can also be used, which takes into account both the linear and quadratic terms when performing the correction. [32, 33] Normalization of the spectra can be performed to the value ranges specified by the user (e.g., between 0 and 1). If no normalization range is provided, each spectrum is normalized with the respective Euclidean norm. Instead of scatter correction, `nippy` also supports baseline correction which only mean-centers the spectra.

Smoothing of the NIR spectra can be helpful for removing environmental or instrumentation-related noise. `nippy` provides both convolution-based and Savitzky-Golay filtering [34] as smoothing methods. The Savitzky-Golay filter is also able to return smoothed derivatives of the original spectrum.

Sometimes using the entire available range of wavelengths is not desired and instead the analysis is constrained to specific optical windows [35].

Trimming operation enables the extraction of continuous and non-continuous wavelength regions from the full spectral data.

As a final step, the spectra can be resampled to a new spectral resolution using the Fourier method. This step can be useful, for instance, when combining spectra acquired with multiple devices having different spectral resolutions.

2.3. Usage of `nippy`

The default format for data entry in `nippy` is `numpy`-matrices. The module requires one vector specifying the wavelengths and a matrix containing the spectra. The first dimension of the matrix must correspond to the wavelengths while the second dimension corresponds to samples.

Pipelines in `nippy` are defined using a configuration file. Structure of the configuration file follows the INI format (`https://en.wikipedia.org/wiki/INI_file`). In the configuration file, preprocessing methods are entered as sections and parameters as *key-value* pairs. Most parameters (such as the length of the filtering window) accept multiple variations of the parameter value which can be separated with a comma. An up-to-date list of preprocessing methods and related parameters can be found in the online documentation (`https://github.com/UEF-BBC/nippy/blob/master/CONFIGURATION.md`). Reading the configuration file will produce a list of nested dictionaries where the dictionaries contain the argument-value pairs of the preprocessing functions.

Once the NIR data and the pipeline configuration have been loaded, they can be passed to `nippy`. The handler module of `nippy` then automatically applies all combinations of preprocessing methods defined in the configuration file. The operation returns a list of copies of the original data, each modified according to the methods in the pipeline list. The preprocessed data is returned as `numpy`-arrays, meaning it can be fed directly in most machine learning methods. The module also supports iteration through a *Preprocessor*-class, which returns one preprocessed version of the original data at a time.

`nippy` also supports data export to other platforms (such as MATLAB or R). The export formats currently supported are MAT, CSV, and Pickle. An in-depth description of the data structure in exported files can be found from the online documentation.

3. Results

The basic operation of `nippy` (fig. 1) is demonstrated through two real-world examples. In both cases, the construction of a calibration model is optimized by finding the best NIRS preprocessing pipeline through grid search. Source code for both examples is available at `https://github.com/uef-bbc/nippy-aca-2019`.

3.1. Example 1: Classification of Ethiopian barley variants using NIR

The first example case is a classification task with a publicly available NIR dataset. The dataset (originally published by Kosmowski et al. [36]) consists of NIR measurements of 1200 samples of Ethiopian barley from 24 different barley cultivar variants. The objective of the example is to classify the barley cultivar variant (Ardu 1260 B, Bahati, Bekoji-1, etc.) of a given sample based on the NIR spectrum. This example represents a typical agricultural application where data collected in a field with a portable device is utilized for quality control.

As in Kosmowski et al., the dataset was first split into training (70%) and testing (30%) sets. The test set was stratified over the 24 classes resulting in 15 samples per class. Preprocessing of the data was performed using `nippy`. Similar to the original analysis by Kosmowski et al., classification of the samples was performed using support vector machines (SVMs). A nu-regularized SVM with a polynomial kernel function was trained separately for each individual preprocessing pipeline and the hyperparameters of each model were optimized using a five-fold cross-validation. As the initial inspection of the NIR spectra looked noise-free and relatively flat (fig. 2a), the preprocessing techniques were restricted to Savitzky-Golay filtering and two scatter correction methods (SNV and RNV). The parameter combinations used to build these preprocessing pipelines are listed in table 1.

In total, 38 different preprocessing combinations (table 1) and one baseline model without preprocessing were tested (see fig. 2a for original data and 2b for different preprocessing methods). The effect of each preprocessing method was quantified by comparing the accuracies of hold-out test and training datasets over different pipelines (fig. 3). The best overall accuracy (82.6% for training and 87.2% for test set) was obtained using a preprocessing pipeline with SNV scatter correction and a first-order derivative Savitzky-Golay filtering (3rd order polynomial, 11-point window length).

The baseline model without any preprocessing yielded an accuracy of 75.0% for training and 80.3% for test set. The confusion matrices of the classification results were computed for the baseline model and the best performing preprocessing method (fig. 4). The classification accuracy of the best performing preprocessing pipeline was identical to the best performing model reported by Kosmowski et al. [36].

3.2. Example 2: Regression model for predicting the instantaneous modulus of equine articular cartilage

The second example focuses on complex and noisy data, where NIR spectra were measured from the articular cartilage surface of equine fetlock joints (fig. 5). Articular cartilage is a layer of viscoleastic connective tissue covering the ends of articulating bones within a joint. Material properties of the cartilage layer (such as instantaneous modulus) are an important indicator of joint health. Ability to determine the material properties of cartilage during an arthroscopic procedure could have substantial diagnostic significance in identifying healthy and degraded regions of the joint.

The dataset was originally published by Sarin et al. [37] and optimal regression and variable selection methods for this data were subsequently investigated by Prakash et al. [38]. NIRS measurements were performed on 869 points from the proximal phalanx and the metacarpal bone of five horses. Instantaneous modulus at each measurement point was determined using a custom material testing device (for details see[37]). The dataset and an in-depth description of different variables can be found in [39].

The objective of this example is to construct a calibration model capable of predicting the instantaneous modulus of cartilage from the NIR spectrum. An earlier investigation of model selection and regression methods for this dataset [38] determined that the best prediction performance ($R^2 = 0.51$, RMSEP = 2.46 MPa) was obtained with a five-component partial least squares regression. The preprocessing used in that analysis consisted of a third order Savitzky-Golay filtering (25 nm window size) and second order spectral derivation. Identical regression technique was used here and the effect of preprocessing was investigated using `nippy`. The model was validated with the same holdout test method (N=70, 9% of the dataset) used in the original study[38]. Baseline test set prediction performance without any kind of pretreatment of the spectra was $R^2 = 0.25$ and RMSEP = 3.06 MPa.

The pipelines (i.e., parameter combinations) used for the preprocessing consisted of 3rd order Savitzky-Golay filtering with (12 window sizes, up to the 2nd derivative) and convolution filtering (12 window sizes). Particle size effects were compensated using MSC, SNV, LSNV (four different window sizes), and RNV (four different interquartile ranges). Preprocessing was performed on the full spectral range, as well as subsets of 700 - 900 nm and 850 - 1050 nm. The absence of each aforementioned preprocessing step was also investigated by leaving them out of the analysis. In total, `nippy` was used to generate 1618 comparable pipelines (table 2) which were then compared using PLSR. Number of components for each PLSR model was determined using five-fold cross-validation. Best test set prediction performance in terms of R^2 was obtained using the wavelength range of 700 – 950 nm, RNV scatter correction (85% – 15% interquartile range), and a Savitzky-Golay filtering with 73 nm window size (see figs. 6A and 6B). In comparison to the baseline PLSR model, the preprocessing pipeline increased the coefficient of determination by approximately 38% ($R^2 = 0.63$, RMSEP = 2.15 MPa, see fig. 6C). Preprocessing also improved the residuals of the model in terms of magnitude, homoscedasticity, and normality (fig. 6D). Compared to the performance reported earlier with this dataset, optimization of the preprocessing step provided a modest improvement of approximately 12% in terms of R^2 and 13% in terms of RMSEP.

Table 1: Configuration parameters for example 1.

Preprocessing operation	Parameter	Values
SNV	`snv_type:`	`snv, rnv`
	`also_skip:`	`True`
SAVGOL	`filter_win:`	`11,21,51,101`
	`deriv_order:`	`0, 1, 2`
	`poly_order:`	`3`
	`also_skip:`	`True`

4. Discussion

Preprocessing of NIR spectrum is a fundamental part of any NIRS application. An optimized preprocessing protocol can substantially improve the predictive capabilities of NIRS models. In this paper, we presented an open-source Python module for semi-automatic exploration and comparison of different preprocessing strategies. Ideally, the tools introduced in this paper should cut down development time when researching or building new NIR-based analytical applications.

While improving the prediction performance is the main use of `nippy`, it also enhances understanding why some preprocessing improves the result by revealing additional details about the underlying phenomena. By comparing

Table 2: Configuration parameters for example 2.

Preprocessing operation	Parameter	Values
MSC	`also_skip:`	`True`
SNV	`also_skip:`	`True`
RNV	`iqr:`	`75-25, 85-15, 65-35, 70-30`
	`also_skip:`	`True`
LSNV	`num_windows:`	`3, 5, 7, 9`
	`also_skip:`	`True`
SAVGOL	`filter_win:`	`11,25,41,55,71,85,101,115,131,145,161,175`
	`deriv_order:`	`0, 1, 2`
	`poly_order:`	`3`
	`also_skip:`	`True`
SMOOTH	`filter_win:`	`11,25,41,55,71,85,101,115,131,145,161,175`
	`also_skip:`	`True`
TRIM	`bins:`	`700 - 950, 850 - 1050`
	`also_skip:`	`True`

which methods work and which do not can yield more insight into what kind of instrumentation is needed by the application. For example, the substantial increase in accuracy as a result of derivation in the Ethiopian barley classification (section 3.1) most likely indicates that the differentiating factors are minute spectral peaks and not the baseline level of the signal.

Software tools for chemometric analysis of NIRS data have existed for a long time and range from proprietary analysis solutions, such as The Unscrambler X (Camo Analytics, Oslo, Norway), OPUS (Bruker Corporation, Billerica, MA, USA), or Pirouette (Infometrix Inc., Bothell, WA, USA), to open-source libraries. While proprietary software can be useful in industrial applications, research of new NIRS analysis techniques is typically conducted with data science and programming oriented methods using tools, such as R, MATLAB, or Python. The main benefits of open-source tools are transparency, customizability, easier access, and lower cost. We have, therefore, limited our comparison of `nippy` to other comparable free or open-source tools, more specifically, `prospectr` [40] and ParLeS [41]

The `prospectr` is one of the most popular R packages for analysing visible and NIR spectroscopic data. The package implements a set of preprocessing and sampling functions in the R language. Preprocessing methods included in `prospectr` are largely similar to those found in `nippy`. The sampling methods present different techniques for selecting training, testing, and validation sets for constructing the calibration models (e.g., Kennard-Stone sampling, DUPLEX sampling, etc.). Variable selection and calibration methods are not included in the `prospectr` package. ParLes is a shareware software solution for constructing NIRS calibration models. It implements the full chain of operations ranging from preprocessing and variable selection to training calibration models. Like `prospectr`, various sampling methods for training the models are also provided.

In comparison, `nippy` shares more features with the `prospectr` package than ParLeS, in the sense that they are both libraries containing functions for preprocessing NIRS data. Main difference between the two is that the `prospectr` package does not enable rapid iteration of multiple comparable preprocessing pipelines. ParLes is more of a stand-alone tool for data exploration that enables the user to perform multiple different manipulations and analysis on the given data. Again, however, the ability to rapidly and programmatically test the effect of different preprocessing pipelines is missing. Furthermore, integration to other analysis platforms (e.g. Python or MATLAB) can not be done directly. As ParLeS is a shareware application, extending the analysis capabilities of the tool is impossible without the help of the original author.

Several earlier chemometric studies have also indicated that preprocessing has a substantial impact on the performance spectroscopic models and depends on such factors as: the preprocessing operations, parameters used, and the order of operations [42, 17, 25, 22, 23, 24]. In addition, Engel et al. [22] pointed out that the sequential optimization of a preprocessing pipeline might not work, as the synergy between different operations can be hard to predict in advance. These studies have also suggested different approaches for finding a suitable preprocessing strategy from multiple comparable alternatives. For instance, Wold et al. [42] suggested utilizing orthogonal signal correction, where preprocessing methods aim to remove linearly uncorrelated spectral components with respect to the target property. Solution proposed by Xu et al. [17] used Monte Carlo sampling for selecting the best preprocessing pipeline from multiple alternatives. Other approaches have suggested genetic algorithms [25] or design of experiments approach [23] for selecting the preprocessing strategy. Combination of preprocessing and variable selection under the same optimization process has also been investigated [24]. While exhaustive grid-search never fails to find the optimal preprocessing combination for a given dataset (provided that the grid of operations and related parameters is dense enough), the operation can be very time-consuming. The proposed hyperparameter tuning methods can converge faster and can enable larger search space for parameters. As a final step, generalization of the optimal solution outside the training data should be verified in order to avoid overfitting.

While the current implementation of `nippy` enables the user to rapidly sift through various combinations of preprocessing operations, it does not provide actual feedback on which of the combinations is the most effective. This feature was intentionally left out as metaheuristics (also known as automated machine learning or hyperparameter search), a relatively recent and very active field of research, can deal with higher level optimization of machine learning pipelines. Metaheuristics can be utilized to find the best solution to a problem by individually tuning the preprocessing, feature extraction, and classification/regression. Several powerful tools, ranging from bayesian optimization [43] to genetic programming [44], already exist for solving this problem. Instead of competing with existing tools, `nippy` was built with focus on compatibility with modern metaheuristic tools. As machine learning tools are nowadays predominantly written in Python and use `numpy`-based matrix structures as its base, the same approach was adopted here. The end goal was to combine decades worth of domain knowledge gained from various NIRS publications and combine it with powerful machine learning frameworks.

In the future, `nippy` could potentially be extended to cover other spectroscopic techniques as well. Raman and mid infrared spectroscopy are very similar to NIRS and, thus, benefit from the same preprocessing operations.

However, as NIRS is often used in industrial and agricultural applications, the most crucial need for optimized preprocessing is with this technique.

5. Conclusions

To conclude, a lot is known about the different ways to eliminate scattering effects and external noise from NIR spectra. Due to the great diversity of different NIRS applications, the best preprocessing strategy is often dependent on the intended use. Finding the correct combination of preprocessing, variable selection, and calibration has been the focus of much recent research. To facilitate the optimization of preprocessing for NIRS models, we have developed `nippy`, a tool that enables rapid iteration of different preprocessing combinations. We feel that tools, such as `nippy`, are important to chemometrics because they provide researchers easy access to current state-of-art NIR preprocessing and thus enable them to focus on optimizing models instead of reinventing the wheel.

6. Acknowledgements

This research received funding from the following sources: SCITECO Doctoral Programme of University of Eastern Finland, Kuopio University Hospital (VTR projects 5041750, 5041744, 5041778, and 5203111, PY210 Clinical Neurophysiology), the Academy of Finland (projects 267551, 310466, and 315820), Orion Research Foundation sr, and Finnish foundation of technology promotion (project 8193).

7. Contributions

Jari Torniainen: Investigation; Software; Writing - original draft

Isaac O. Afara: Software; Writing - review & editing

Mithilesh Prakash: Validation; Writing - review & editing

Jaakko Sarin: Validation; Writing - review & editing

Lauri Stenroth: Supervision; Writing - review & editing

Juha Töyräs: Supervision; Project administration; Writing - review & editing

[1] C. Pasquini, Near infrared spectroscopy: A mature analytical technique with new perspectives – A review, Analytica Chimica Acta 1026 (2018) 8–36.

[2] C. P. Bacon, Y. Mattley, R. DeFrece, Miniature spectroscopic instrumentation: Applications to biology and chemistry, Review of Scientific Instruments 75 (2004) 1–16.

[3] R. V. Rossel, D. Walvoort, A. McBratney, L. J. Janik, J. Skjemstad, Visible, near infrared, mid infrared or combined diffuse reflectance spectroscopy for simultaneous assessment of various soil properties, Geoderma 131 (2006) 59–75.

[4] N. Prieto, R. Roehe, P. Lavin, G. Batten, S. Andres, Application of near infrared reflectance spectroscopy to predict meat and meat products quality: A review, Meat Science 83 (2009) 175–186.

[5] Y. Roggo, P. Chalus, L. Maurer, C. Lema-Martinez, A. Edmond, N. Jent, A review of near infrared spectroscopy and chemometrics in

pharmaceutical technologies, Journal of pharmaceutical and biomedical analysis 44 (2007) 683–700.

[6] V. R. Kondepati, H. M. Heise, J. Backhaus, Recent applications of near-infrared spectroscopy in cancer diagnosis and therapy, Analytical and Bioanalytical Chemistry 390 (2008) 125–139.

[7] J. H. Kalivas, Two data sets of near infrared spectra, Chemometrics and Intelligent Laboratory Systems 37 (1997) 255–259.

[8] D. A. Burns, E. W. Ciurczak, Handbook of near-infrared analysis, CRC press, 2007.

[9] ISO 12099:2017, Animal feeding stuffs, cereals and milled cereal products —- Guidelines for the application of near infrared spectrometry, Standard, International Organization for Standardization, Geneva, CH, 2017.

[10] P. Williams, P. Dardenne, P. Flinn, Tutorial: Items to be included in a report on a near infrared spectroscopy project, Journal of Near Infrared Spectroscopy 25 (2017) 85–90.

[11] M. Larrechi, M. Callao, Strategy for introducing nir spectroscopy and multivariate calibration techniques in industry, TrAC Trends in Analytical Chemistry 22 (2003) 634–640.

[12] Å. Rinnan, F. van den Berg, S. B. Engelsen, Review of the most common pre-processing techniques for near-infrared spectra, TrAC - Trends in Analytical Chemistry 28 (2009) 1201–1222.

[13] Z. Xiaobo, Z. Jiewen, M. J. Povey, M. Holmes, M. Hanpin, Variables selection methods in near-infrared spectroscopy, Analytica chimica acta 667 (2010) 14–32.

[14] R. M. Balabin, S. V. Smirnov, Variable selection in near-infrared spectroscopy: benchmarking of feature selection methods on biodiesel data, Analytica chimica acta 692 (2011) 63–72.

[15] A. Candolfi, R. De Maesschalck, D. Jouan-Rimbaud, P. A. Hailey, D. L. Massart, The influence of data pre-processing in the pattern recognition of excipients near-infrared spectra, Journal of Pharmaceutical and Biomedical Analysis 21 (1999) 115–132.

[16] L. J. Chen, L. Xing, L. J. Han, Influence of Data Preprocessing on the Quantitative Determination of Nutrient Content in Poultry Manure by Near Infrared Spectroscopy, Journal of Environment Quality 39 (2010) 1841.

[17] L. Xu, Y. P. Zhou, L. J. Tang, H. L. Wu, J. H. Jiang, G. L. Shen, R. Q. Yu, Ensemble preprocessing of near-infrared (NIR) spectra for multivariate calibration, Analytica Chimica Acta 616 (2008) 138–143.

[18] L.-L. Wang, Y.-W. Lin, X.-F. Wang, N. Xiao, Y.-D. Xu, H.-D. Li, Q.-S. Xu, A selective review and comparison for interval variable selection in spectroscopic modeling, Chemometrics and Intelligent Laboratory Systems 172 (2018) 229–240.

[19] A. Mouazen, B. Kuang, J. De Baerdemaeker, H. Ramon, Comparison among principal component, partial least squares and back propagation

neural network analyses for accuracy of measurement of selected soil properties with visible and near infrared spectroscopy, Geoderma 158 (2010) 23–31.

[20] R. M. Balabin, R. Z. Safieva, E. I. Lomakina, Comparison of linear and nonlinear calibration models based on near infrared (nir) spectroscopy data for gasoline properties prediction, Chemometrics and intelligent laboratory systems 88 (2007) 183–188.

[21] R. M. Balabin, R. Z. Safieva, E. I. Lomakina, Gasoline classification using near infrared (NIR) spectroscopy data: Comparison of multivariate techniques, Analytica Chimica Acta 671 (2010) 27–35.

[22] J. Engel, J. Gerretzen, E. Szymańska, J. J. Jansen, G. Downey, L. Blanchet, L. M. Buydens, Breaking with trends in pre-processing?, TrAC - Trends in Analytical Chemistry 50 (2013) 96–106.

[23] J. Gerretzen, E. Szymańska, J. J. Jansen, J. Bart, H. J. Van Manen, E. R. Van Den Heuvel, L. M. Buydens, Simple and Effective Way for Data Preprocessing Selection Based on Design of Experiments, Analytical Chemistry 87 (2015) 12096–12103.

[24] J. Gerretzen, E. Szymańska, J. Bart, A. N. Davies, H. J. van Manen, E. R. van den Heuvel, J. J. Jansen, L. M. Buydens, Boosting model performance and interpretation by entangling preprocessing selection and variable selection, Analytica Chimica Acta 938 (2016) 44–52.

[25] T. Bocklitz, A. Walter, K. Hartmann, P. Rösch, J. Popp, How to pre-

process Raman spectra for reliable and stable models?, Analytica Chimica Acta 704 (2011) 47–56.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (2011) 2825–2830.

[27] E. Jones, T. Oliphant, P. Peterson, et al., Scipy: Open source scientific tools for python (2001).

[28] R. J. Barnes, M. S. Dhanoa, S. J. Lister, Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra, Applied Spectroscopy 43 (1989) 772–777.

[29] H. Martens, S. Jensen, P. Geladi, Multivariate linearity transformation for near-infrared reflectance spectrometry, in: Proceedings of the Nordic symposium on applied statistics, Stokkand Forlag Publishers Stavanger, Norway, pp. 205–234.

[30] Q. Guo, W. Wu, D. L. Massart, The robust normal variate transform for pattern recognition with near-infrared data, Analytica Chimica Acta 382 (1999) 87–103.

[31] Y. Bi, K. Yuan, W. Xiao, J. Wu, C. Shi, J. Xia, G. Chu, G. Zhang, G. Zhou, A local pre-processing method for near-infrared spectra, combined with spectral segmentation and standard normal variate transformation, Analytica Chimica Acta 909 (2016) 30–40.

[32] H. Martens, E. Stark, Extended multiplicative signal correction and spectral interference subtraction: new preprocessing methods for near infrared spectroscopy, Journal of pharmaceutical and biomedical analysis 9 (1991) 625–635.

[33] N. K. Afseth, A. Kohler, Extended multiplicative signal correction in vibrational spectroscopy, a tutorial, Chemometrics and Intelligent Laboratory Systems 117 (2012) 92–99.

[34] A. Savitzky, M. J. Golay, Smoothing and differentiation of data by simplified least squares procedures., Analytical chemistry 36 (1964) 1627–1639.

[35] L. A. Sordillo, Y. Pu, S. Pratavieira, Y. Budansky, R. R. Alfano, Deep optical imaging of tissue using the second and third near-infrared spectral windows, Journal of Biomedical Optics 19 (2014) 056004.

[36] F. Kosmowski, T. Worku, Evaluation of a miniaturized nir spectrometer for cultivar identification: The case of barley, chickpea and sorghum in ethiopia, PloS one 13 (2018) e0193620.

[37] J. K. Sarin, M. Amissah, H. Brommer, D. Argüelles, J. Töyräs, I. O. Afara, Near infrared spectroscopic mapping of functional properties of equine articular cartilage, Annals of biomedical engineering 44 (2016) 3335–3345.

[38] M. Prakash, J. K. Sarin, L. Rieppo, I. O. Afara, J. Töyräs, Optimal regression method for near-infrared spectroscopic evaluation of articular cartilage, Applied spectroscopy 71 (2017) 2253–2262.

[39] J. K. Sarin, J. Torniainen, M. Prakash, L. Rieppo, I. O. Afara, J. Töyräs, Dataset on equine cartilage near infrared spectra, composition, and functional properties, Scientific Data 6 (2019) 164.

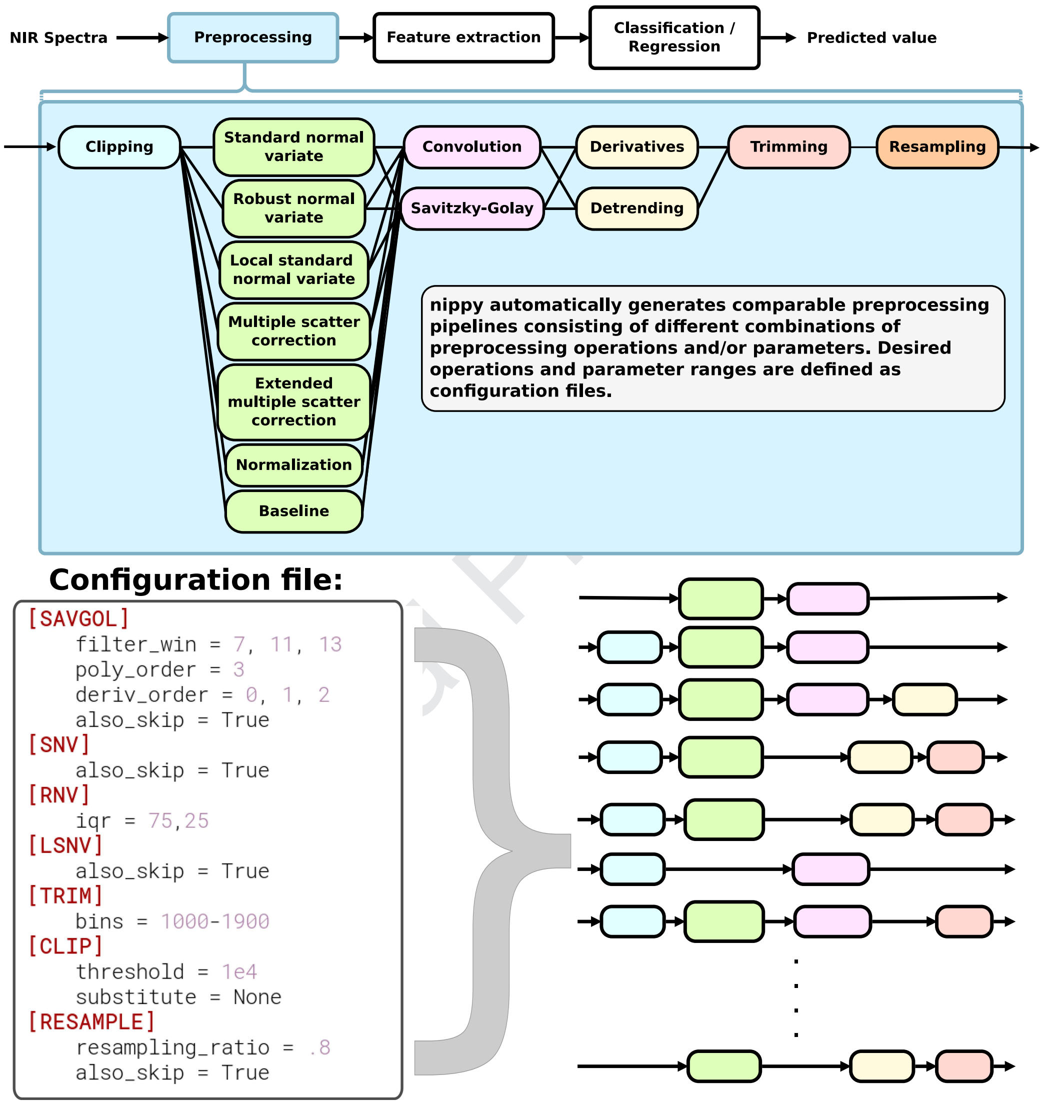
[40] A. Stevens, L. Ramirez-Lopez, An introduction to the prospectr package (2013) 1–22.

[41] R. A. Viscarra Rossel, ParLeS: Software for chemometric analysis of spectroscopic data, Chemometrics and Intelligent Laboratory Systems 90 (2008) 72–83.

[42] S. Wold, H. Antti, F. Lindgren, J. Öhman, Orthogonal signal correction of near-infrared spectra, Chemometrics and Intelligent Laboratory Systems 44 (1998) 175–185.

[43] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: Advances in Neural Information Processing Systems, pp. 2962–2970.

[44] R. S. Olson, J. H. Moore, Tpot: A tree-based pipeline optimization tool for automating machine learning, in: Workshop on Automatic Machine Learning, pp. 66–74.


NIR Spectra
Preprocessing
Feature extraction
Classification / Regression
Predicted value
Clipping
Standard normal variate
Robust normal variate
Local standard normal variate
Multiple scatter correction
Extended multiple scatter correction
Normalization
Baseline
Convolution
Savitzky-Golay
Derivatives
Detrending
Trimming
Resampling
nippy automatically generates comparable preprocessing pipelines consisting of different combinations of preprocessing operations and/or parameters. Desired operations and parameter ranges are defined as configuration files.
Configuration file:

```
[SAVGOL]
    filter_win = 7, 11, 13
    poly_order = 3
    deriv_order = 0, 1, 2
    also_skip = True
[SNV]
    also_skip = True
[RNV]
    iqr = 75,25
[LSNV]
    also_skip = True
[TRIM]
    bins = 1000-1900
[CLIP]
    threshold = 1e4
    substitute = None
[RESAMPLE]
    resampling_ratio = .8
    also_skip = True
```

Figure 1: General operation principle of `nippy`. Several preprocessing options and parameter values can be combined into multiple competing preprocessing strategies (called pipelines). Comparing the effect of different preprocessors on the prediction performance of the NIRS model will yield the most optimal solution for a given application.

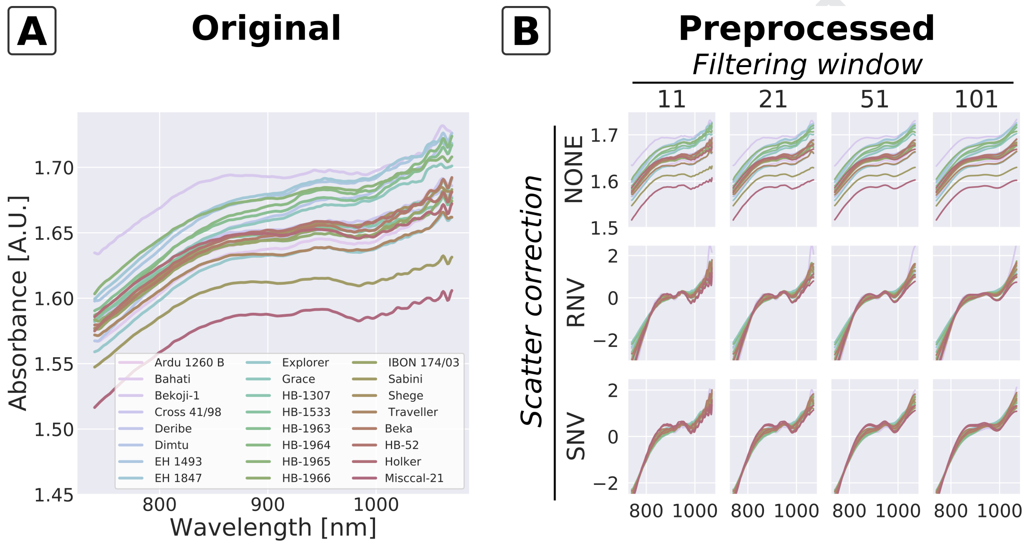


Figure 2: A: Overview of the Etiophian barley NIRS data set. Individual lines correspond to the averaged unprocessed spectra of the 24 different barley variants. B: The effect of different parameter combinations of scatter correction and smoothing to the per-class average spectra of the dataset. Rows correspond to different scatter correction treatment while columns represent different filter window lengths used in Savitzky-Golay filtering (3rd polynomial order, no derivation).

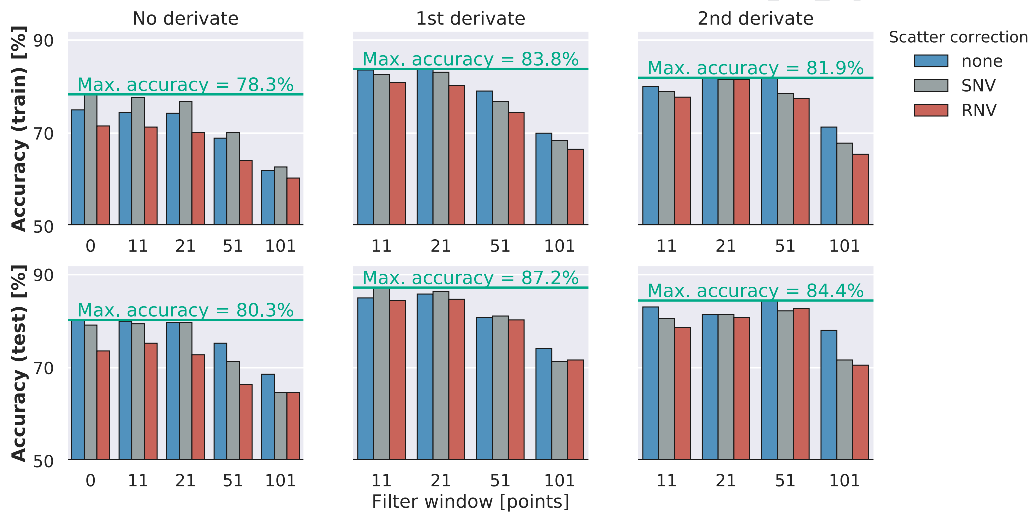


Figure 3: Overall training and testing accuracy with SVM classification for barley cultivars of example 1. Training accuracy was derived from cross-validation while testing accuracy represents the performance of an independent hold-out set. Results have been divided between filtering window length (the effect of omitting filtering was also investigated), scatter correction method, and derivative.

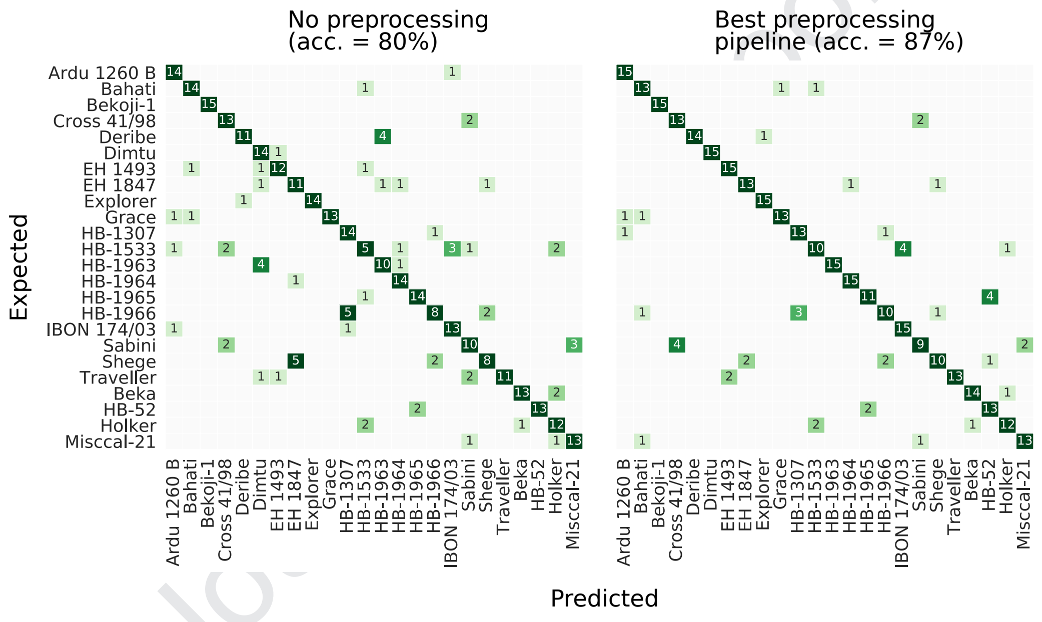
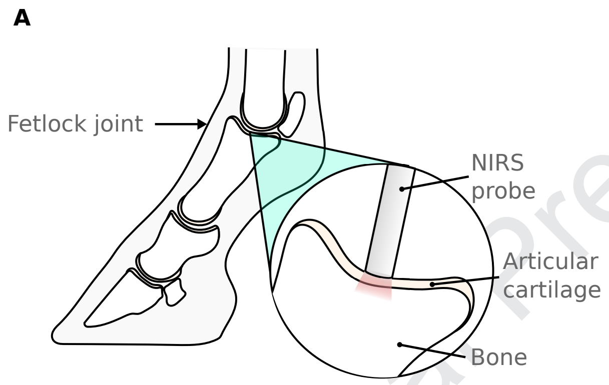


Figure 4: A comparison of confusion matrices between the un-preprocessed data and the data preprocessed with the optimal preprocessing combination of the barley cultivar classification example.



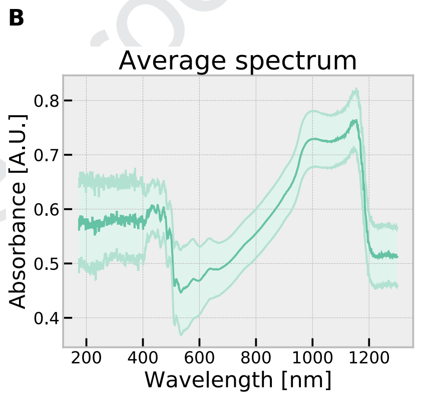


Figure 5: A: NIRS spectra was measured from the cartilage surface of the equine fetlock joint. In total, five joints were measured from 44 different areas of interest (AIs). Each AI consisted between 6–25 measurement points (depending on the joint geometry and cartilage condition) resulting in 861 measurement points. Local instantaneous modulus was determined for each measurement site. B: Average of all the collected NIR spectra. Shaded regions represent the standard deviation.

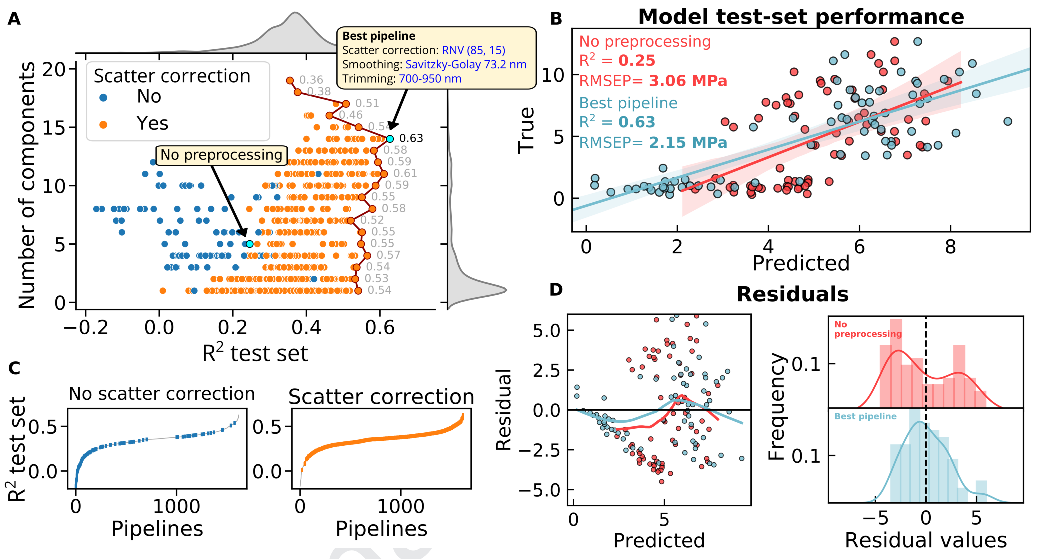
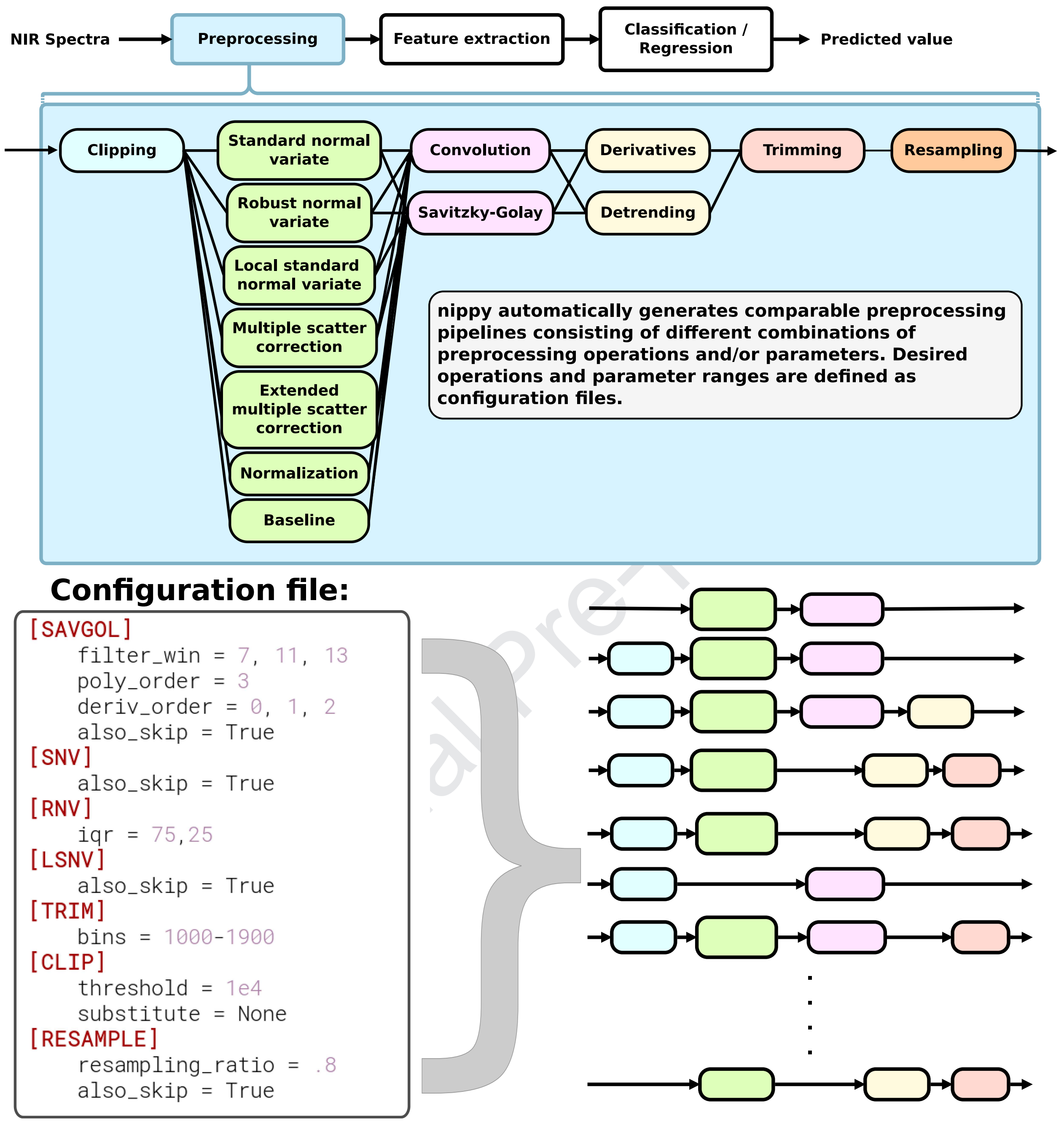
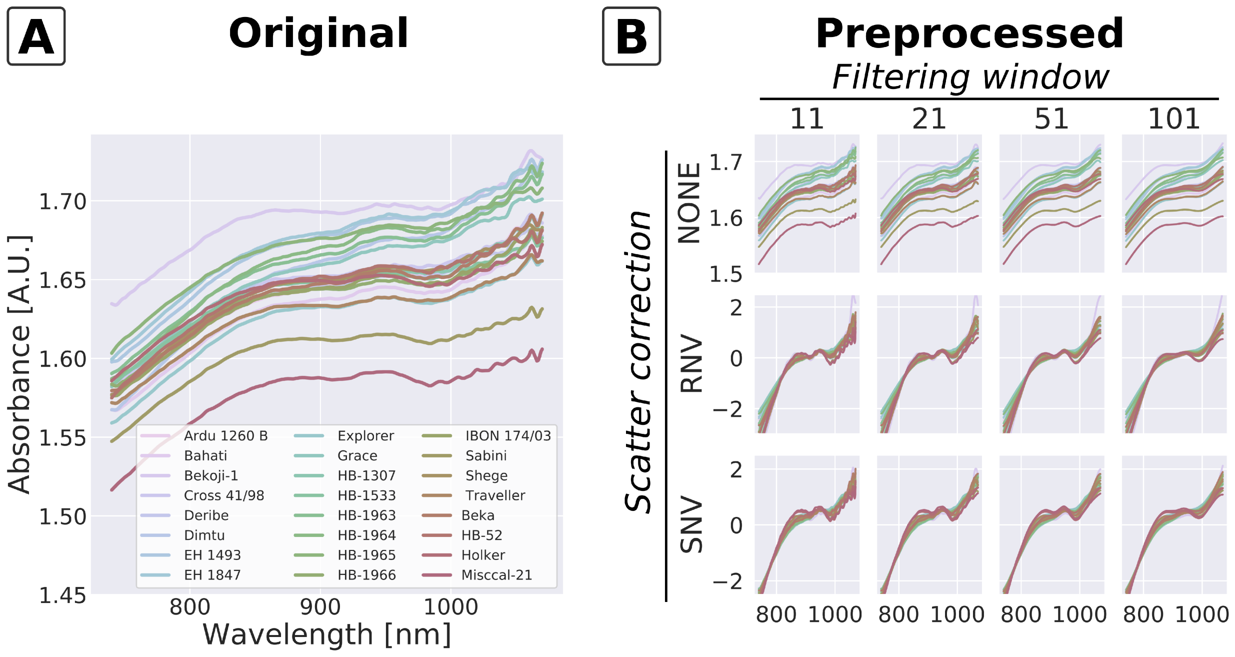


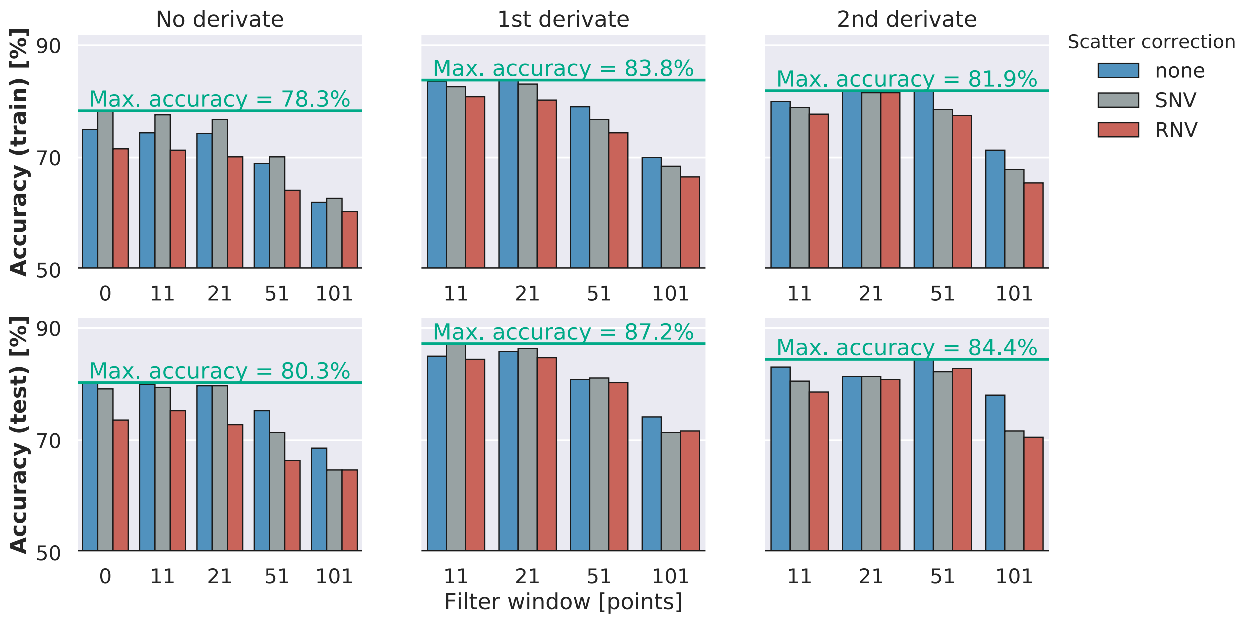
Figure 6: A: All 1618 models generated to predict the instantaneous modulus of articular cartilage presented in terms of model complexity (i.e., the number of latent variables) and test set performance. Color of individual models indicates whether the data was treated with scatter correction. Margin plots indicate the distribution of models in terms of complexity and performance. B: True vs predicted values for the test set between no preprocessing and the best performing pipeline. C: All preprocessing pipelines (scatter correction vs no scatter correction) sorted according to the R^2 values of the test set. D: Residual plots and the distributions of residuals between no preprocessing and the best performing pipeline.



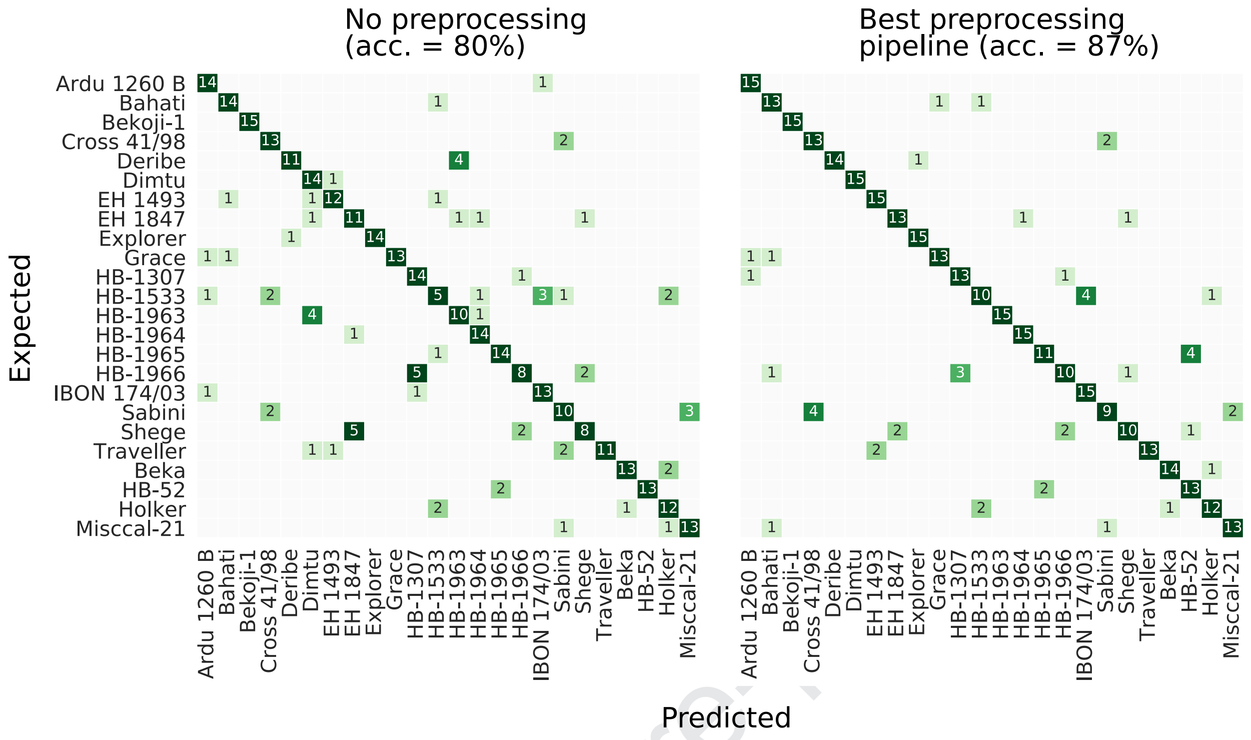
nippy automatically generates comparable preprocessing pipelines consisting of different combinations of preprocessing operations and/or parameters. Desired operations and parameter ranges are defined as configuration files.

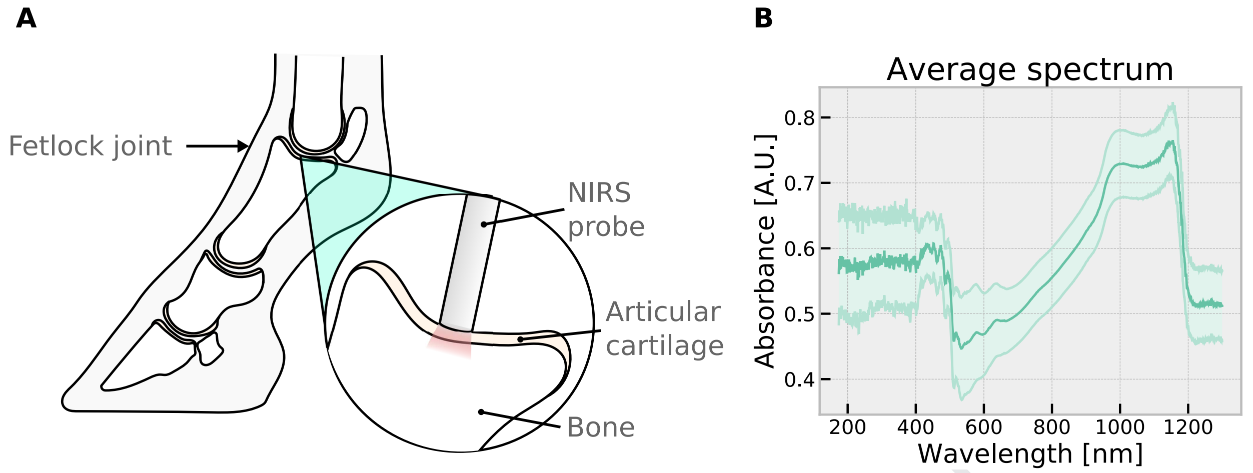
```
[SAVGOL]
    filter_win = 7, 11, 13
    poly_order = 3
    deriv_order = 0, 1, 2
    also_skip = True
[SNV]
    also_skip = True
[RNV]
    iqr = 75,25
[LSNV]
    also_skip = True
[TRIM]
    bins = 1000-1900
[CLIP]
    threshold = 1e4
    substitute = None
[RESAMPLE]
    resampling_ratio = .8
    also_skip = True
```

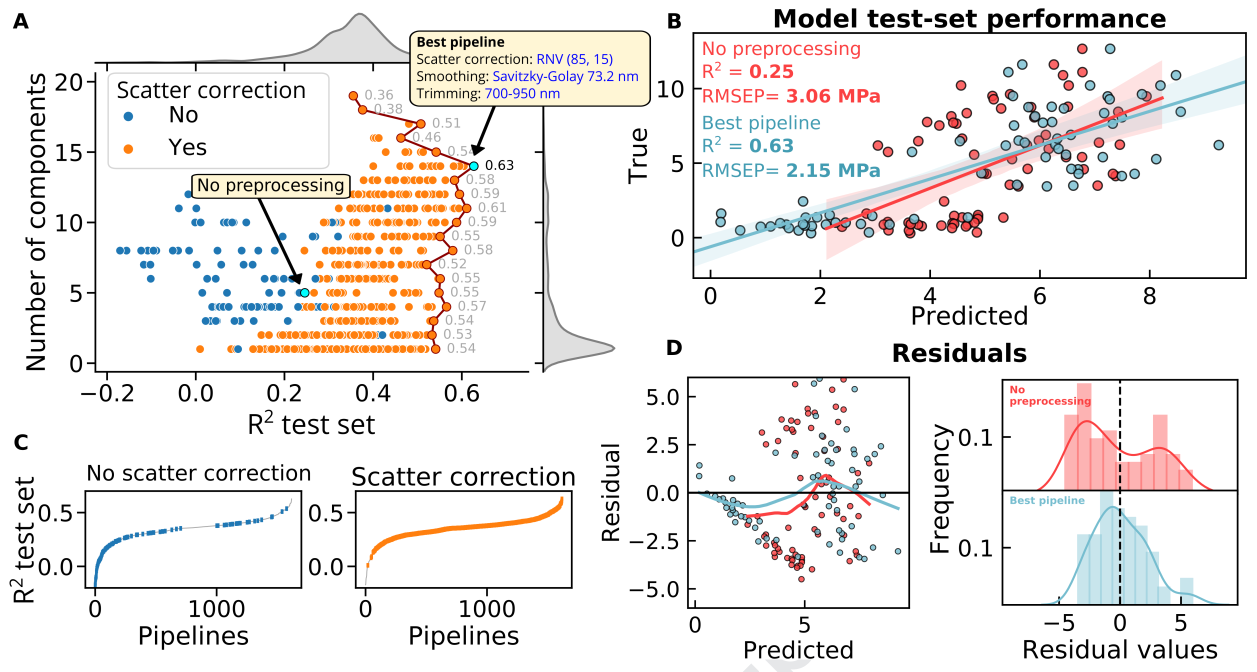

A
Original
Absorbance [A.U.]
1.70
1.65
1.60
1.55
1.50
1.45
800
900
1000
Wavelength [nm]
Ardu 1260 B
Bahati
Bekoji-1
Cross 41/98
Deribe
Dimtu
EH 1493
EH 1847
Explorer
Grace
HB-1307
HB-1533
HB-1963
HB-1964
HB-1965
HB-1966
IBON 174/03
Sabini
Shege
Traveller
Beka
HB-52
Holker
Misccal-21
B
Preprocessed
Filtering window
11
21
51
101
Scatter correction
NONE
RNV
SNV
1.7
1.6
1.5
2
0
−2
800
1000



No derivate
1st derivate
2nd derivate
Scatter correction
none
SNV
RNV
Accuracy (train) [%]
Accuracy (test) [%]
90
70
50
Max. accuracy = 78.3%
Max. accuracy = 83.8%
Max. accuracy = 81.9%
Max. accuracy = 80.3%
Max. accuracy = 87.2%
Max. accuracy = 84.4%
0
11
21
51
101
Filter window [points]


No preprocessing (acc. = 80%)
Best preprocessing pipeline (acc. = 87%)
Expected
Predicted
Ardu 1260 B
Bahati
Bekoji-1
Cross 41/98
Deribe
Dimtu
EH 1493
EH 1847
Explorer
Grace
HB-1307
HB-1533
HB-1963
HB-1964
HB-1965
HB-1966
IBON 174/03
Sabini
Shege
Traveller
Beka
HB-52
Holker
Misccal-21


A
Fetlock joint
NIRS probe
Articular cartilage
Bone
B
Average spectrum
Absorbance [A.U.]
0.4
0.5
0.6
0.7
0.8
200
400
600
800
1000
1200
Wavelength [nm]


A
Best pipeline
Scatter correction: RNV (85, 15)
Smoothing: Savitzky-Golay 73.2 nm
Trimming: 700-950 nm
Scatter correction
No
Yes
No preprocessing
Number of components
R^2 test set
B
Model test-set performance
No preprocessing
R^2 = 0.25
RMSEP= 3.06 MPa
Best pipeline
R^2 = 0.63
RMSEP= 2.15 MPa
True
Predicted
C
No scatter correction
Scatter correction
R^2 test set
Pipelines
Pipelines
D
Residuals
Residual
Predicted
No preprocessing
Best pipeline
Frequency
Residual values

Highlights

- Spectral preprocessing affects the performance of chemometric models
- Selection of best preprocessing strategy depends heavily on the intended application
- nippy module can rapidly explore different preprocessing combinations
- Semi-automatic preprocessing tuning for spectroscopic models

Declaration of interests

[✓] The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

On behalf of all the authors;

[illegible]

JARI TORNIAINEN