Statistical Methods:
Prediction of Soil Parameters through Near Infrared Spectroscopy

Kazimir Menzel
Markus Pawellek

August 19, 2016

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

Statistical Methods:
Prediction of Soil Parameters through Near Infrared Spectroscopy

Kazimir Menzel
kazimir.menzel@me.com

Markus Pawellek
markuspawellek@gmail.com

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1 Introduction

Through soil analyses or soil testing one can get certain chemical and physical information about the used soil like the concentration of soil organic carbon (SOC) or the pH-value. With these Measurements, known as soil parameters, it is possible to assist in solving soil-related problems such as optimizing plant growth.

However the direct measurement of soil parameters is very costly and error-prone. Therefore methods for fast and cheap determination of these parameters play a fundamental and vital role in particular fields like agriculture, geochemistry and ecology.

Albeit developed in the 1960s, Near Infrared Spectroscopy (NIRS) only gained traction during the 1990s as sufficient computing power became increasingly available. NIRS provides a cheap alternative to other methods for the analysis of soil composition .

After a short summary of the physical background of NIRS, we will lay out some of the difficulties in applying NIRS to real world problems. The rest of the paper is dedicated to a methodological framework to address some of those obstacles. In particular we propose a computationally efficient way to identify optimal parameters on a training sample with known data to be used for further analysis of NIRS on soil samples.

2 Background

2.1 Soil Parameters

Let A be any substance in a given soil sample dissolved in a solution of volume $V \in (0, \infty)$ and let $n_A \in (0, \infty)$ be the amount of A in the sample. Then the molar concentration c_A is given by

$$c_A \coloneqq \frac{n_A}{V}$$

Now let c_0 be the molar concentration of this whole sample and n_0 the amount of the whole sample. We define the amount-of-substance fraction (ASF) p_A of A as

$$p^{(A)} \coloneqq 100\% \cdot \frac{c_A}{c_0} = 100\% \cdot \frac{n_A}{n_0}$$

In this report, we concentrate on three important soil parameters that are given by existing NIRS-measurements. The first two parameters $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$ are the ASFs relating to soil organic carbon (SOC) and nitrogen (N) in a given soil sample. SOC refers to the carbon in the sample that is bound in an organic compound. The third parameter is the pH-value that specifies the acidity of an aqueous solution. It links to the concentration of hydronium ions $c_{\mathrm{H_3O^+}}$.

$$\mathrm{pH} \coloneqq -\lg c_{\mathrm{H_3O^+}} = -\frac{\ln c_{\mathrm{H_3O^+}}}{\ln 10}$$

2.2 Near Infrared Spectroscopy

NIRS uses electromagnetic waves,famously known as light, with wavelengths ranging from $780\,\mathrm{nm}$ to $3000\,\mathrm{nm}$, the so called near infrared spectrum. This area is called the near infrared region and is the most energetic one of the infrared light.

An emitted light wave with wavelength $\lambda \in (0, \infty)$ interacts with a soil sample in three ways: It can be reflected, absorbed or transmitted. For most soil samples measuring the transmittance of light waves is not sensible as the thickness of these samples varies. Since the absorptance cannot be determined directly, reflectance is used.

Reflection can be split in specular and diffuse reflection. NIRS uses the latter as it penetrates the sample the most. As a consequence, diffuse reflected light is hemispherically scattered and contains information about the soil sample composition. For a more detailed view on this topic please refer to [Tutorial].

The reflectance

$$\varrho\colon (0,\infty) \to (0,\infty), \qquad \varrho(\lambda) \coloneqq \frac{P_\mathrm{r}(\lambda)}{P_0}$$

of a surface depending on wavelength λ of a light wave is given by the amount of radiation power $P_\mathrm{r}(\lambda)$ that is reflected from the surface divided by the initial power P_0 of the light wave. In our case, this function ϱ is defined as the near infrared spectrum of the soil sample.

Absorptance itself originates from the existence of vibrational modes in molecules. A photon with a wavelength $\lambda \in (0,\infty)$ can only be absorbed if the appropriate frequency f exactly matches a multiple of the transition energy of the bond or group that vibrates.This is why the spectra of soil samples are formed of overtones and combination bands.

Due to the similarity of diffuse reflected and transmitted light we can use Beer-Lambert's law as a relation of the attenuation of light and the properties of samples. Let $n \in \mathbb{N}$ be the count of different substances in a sample and c_i be the molar concentration of the ith substance for $i \in \mathbb{N}, i \leq n$. Is again $\lambda \in (0,\infty)$ the wavelength of the used light then there exist certain coefficients $\varepsilon_i(\lambda)$ for all $i \in \mathbb{N}, i \leq n$ such that

$$-\ln \varrho(\lambda) = \sum_{i=1}^{n} \varepsilon_i(\lambda) c_i$$

3 Methodology

3.1 Measured Data

As a single spectrum contains overlapping information, it is necessary to determine both relevant wavelengths and the respective parameters to apply NIRS to practical problems. To select wavelengths and determine parameters we use an example data set, which contains $p^{(\mathrm{SOC})}, p^{(\mathrm{N})}$, pH and wave reflectances of 319 wavelengths ranging from $1400\,\mathrm{nm}$ to $2672\,\mathrm{nm}$ by steps of $4\,\mathrm{nm}$ for 533 samples.

We define Λ as the set of all measured wavelengths. The reflectance $\varrho(\lambda)$ of a sample at a wavelength $\lambda \in \Lambda$ is recorded as

$$-\lg \varrho(\lambda) = -\frac{\ln \varrho(\lambda)}{\ln 10}$$

Figure 1 shows six randomly chosen soil spectra in a diagram.

3.2 Statistical Model

Let $n \in \mathbb{N}$ be the size of the data set and $k \in \mathbb{N}$ with $k \leq n$ the number of measured wavelengths. We define ϱ_i as the soil spectrum of the ith sample for every $i \in \mathbb{N}, i \leq n$. λ_j is the jth measured wavelength for every $j \in \mathbb{N}, j \leq k$. We will alternatively refer to these as predictors.Then according to section 3.1 the measured reflectance values are x_{ij} with

$$x_{ij} \coloneqq -\lg \varrho_i(\lambda_j)$$

for every $i, j \in \mathbb{N}, i \leq n, j \leq k$.

We define the measured ASF of SOC of the ith sample for every $i \in \mathbb{N}, i \leq n$ as $p_i^{(\mathrm{SOC})}$ to which we will also refer to as response variable. To simplify notation, we then define the n-dimensional vector

$$p^{(\mathrm{SOC})} \coloneqq \left(p_i^{(\mathrm{SOC})}\right)$$

The Beer-Lambert law allows us to make assumptions on the relations between soil spectra and the response variable. We saw in section 2.2 that the logarithmised reflectance can be written as a linear combination of molar concentrations. Hence, it seems plausible to assume that an ASF can be represented by a linear combination of logarithmised reflectance values.

Now let $P^{(\mathrm{SOC})}$ be the appropriate random vector to $p^{(\mathrm{SOC})}$. Then under the above assumption the expected values are given for all $i \in \mathbb{N}, i \leq n$ by

$$\mathbb{E}\, P_i^{(\mathrm{SOC})} \coloneqq \beta_0^{(\mathrm{SOC})} + \sum_{j=1}^{k} x_{ij} \beta_j^{(\mathrm{SOC})}$$

which simplifies with an $\mathbb{X} \in \mathbb{R}^{n\times(k+1)}$, called design matrix, and a parameter vector $\beta^{(\mathrm{SOC})} \in \mathbb{R}^{k+1}$ to

$$\mathbb{E}\, P^{(\mathrm{SOC})} = \mathbb{X}\beta^{(\mathrm{SOC})}$$

To capture the stochastic part of $P^{(\mathrm{SOC})}$, we extend the model to

$$\begin{aligned} P^{(\mathrm{SOC})} &= \mathbb{X}\beta^{(\mathrm{SOC})} + \varepsilon^{(\mathrm{SOC})} \\ \mathbb{E}\,\varepsilon^{(\mathrm{SOC})} &= 0, \qquad \operatorname{cov} \varepsilon^{(\mathrm{SOC})} = (\sigma^2)^{(\mathrm{SOC})}\mathrm{I} \end{aligned}$$

where $(\sigma^2)^{(\mathrm{SOC})} \in (0,\infty)$. Following common practice in physics and chemistry, we further assume that

$$\varepsilon^{(\mathrm{SOC})} \sim \mathcal{N}\left(0, (\sigma^2)^{(\mathrm{SOC})}\mathrm{I}\right)$$

This results in the complete model

$$P^{(\mathrm{SOC})} \sim \mathcal{N}\left(\mathbb{X}\beta^{(\mathrm{SOC})}, (\sigma^2)^{(\mathrm{SOC})}\mathrm{I}\right)$$

The model for the second response variable $P^{(\mathrm{N})}$ is constructed in analogy.

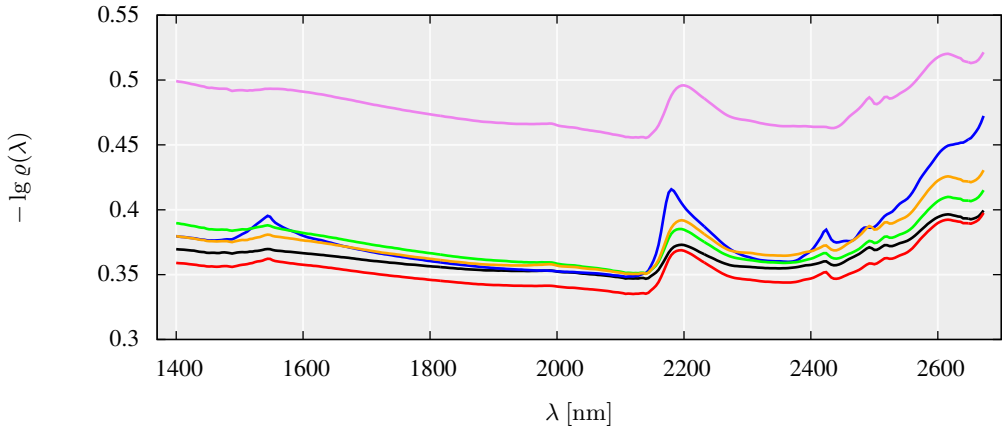


Figure 1: Six near infrared soil spectra of randomly chosen soil samples obtained from the data set, where λ is the wavelength and $\rho(\lambda)$ the corresponding reflectance and each colour refers to one sample

The case for the pH is slightly different, though. When modelling the corresponding random variable we have to adjust the model as the pH is a logarithmised molar concentration. We therefore have to include this into the expected value of the corresponding random variables

$$\mathbb{E}\,\overline{\mathrm{pH}}_i \coloneqq \beta_0^{(\mathrm{pH})} + \sum_{j=1}^{k} \ln(x_{ij})\beta_j^{(\mathrm{pH})}$$

and denote the corresponding matrix by $\mathbb{X}_{\ln}$.

3.3 Multivariate Linear Regression

Multiple linear regression (MLR) or multivariate linear regression is a statistical method for estimating parameters of linear relations between a response variable and a set of predictors and to use these to predict new responses. Let $\mathbb{X} \in \mathbb{R}^{n\times(k+1)}, n, k \in \mathbb{N}, k < n$ be the design matrix, $\sigma^2 \in (0,\infty)$ and Y be the random vector variables with

$$Y \sim \mathcal{N}\left(\mathbb{X}\beta, \sigma^2 \mathrm{I}_n\right)$$

for a certain $\beta \in \mathbb{R}^{k+1}$ Then through the maximum-likelihood-method and a correction we get two best unbiased estimators $\hat{\beta}, \hat{\sigma}^2$ for β and σ^2

$$\hat{\beta}(Y) = \left(\mathbb{X}^{\mathrm{T}}\mathbb{X}\right)^{-1}\mathbb{X}^{\mathrm{T}}Y$$
$$\hat{\sigma}^2(Y) = \frac{1}{n-(k+1)}\left\|Y - \mathbb{X}\hat{\beta}(Y)\right\|^2$$

Now let $y \coloneqq (y_i) \in \mathbb{R}^n$ be a realization of Y. Then we define

$$\hat{y} \coloneqq \mathbb{X}\hat{\beta}(y) = \mathbb{X}\left(\mathbb{X}^{\mathrm{T}}\mathbb{X}\right)^{-1}\mathbb{X}^{\mathrm{T}}y$$
$$\hat{\sigma}^2 \coloneqq \hat{\sigma}^2(y)$$

3.4 Mallows' C_{p}

At this point, the model is specified using $k+1 = 320$ predictors for each response variable, using the whole measured spectra for each soil sample. The reflectances of neighbouring lightwaves are correlated. This might lead to overfitting, i.e. the variance of our estimated parameters $\hat{\beta}_i(Y)$ might be too large, compromising their usability for future measurements. To address this problem, it is sensible to limit each actual model to a "good" subset of the predictors. Hence, our task becomes to select the best or at least a "sufficiently" good model M defined by

$$M \subset \Lambda \cup \{\lambda_0\} =: \overline{\Lambda}$$

where λ_0 stands for the intercept. We denote the design matrix for each M by $\mathbb{X}^{(M)}$. Applying MLR to the new design matrix yields the new estimators

$$\hat{\beta}^{(M)}(Y) = \left(\mathbb{X}^{(M)\mathrm{T}}\mathbb{X}^{(M)}\right)^{-1}\mathbb{X}^{(M)\mathrm{T}}Y$$
$$\left(\hat{\sigma}^2\right)^{(M)}(Y) = \frac{1}{n-\mathrm{p}}\left\|Y - \mathbb{X}^{(M)}\hat{\beta}^{(M)}(Y)\right\|_2^2,$$

where $\mathrm{p} \in \{2,\dots,k\}$ corresponds to the number of predictors included in M.

$$\mathrm{SPSE} \coloneqq \sum_{i=1}^{n}\mathbb{E}\left(Y_{n+i} - \hat{Y}_i^{(M)}\right)^2,$$

which simplifies to

$$\mathrm{SPSE} = n\sigma^2 + \mathrm{p}\sigma^2$$

As σ^2 is unobservable, we have to estimate it so that our estimator for the true SPSE becomes

$$\widehat{\mathrm{SPSE}} \coloneqq \left\|Y - \mathbb{X}^{(M)}\hat{\beta}^{(M)}(Y)\right\|_2^2 + 2\mathrm{p}\left(\hat{\sigma}^2\right)^{(\overline{\Lambda})}$$

To determine the "goodness" of M, we use Mallow's C_p given by

$$C_\mathrm{p}^{(M)} := \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} \left(y_i - \hat{y}_i^{(M)} \right)^2 - n + 2\mathrm{p}$$

The minimization this value is equivalent to the minimization of the sum of predicted squared errors ($\widehat{\mathrm{SPSE}}$).

3.5 Simulated Annealing

As the set of predictors is comparatively large, $n-k < k$, the full-sized model might overfit the data and hence lower the confidence in our parameter estimator. The proposed solution in **??** is to reduce the number of predictors. Still, with a size of the hypothesis space of $|\mathcal{H}| = 2^{319}$, complete search or even best k approaches are beyond feasibility. To reduce the time spent on model search, we will compare two *do we?* model selection algorithms, simulated annealing and bidirectional elimination. We want to find a subset M such that for all $N \in \mathcal{H}$ the inequality

$$C_\mathrm{p}^{(M)} \leq C_\mathrm{p}^{(N)}$$

holds.

Simulated annealing (SANN) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimisation in large search spaces. It simulates the slow cooling of a thermodynamic system through random numbers. With this algorithm it is possible to find a "good" local minimum in a short time.

The algorithm is applicable to arbitrary sets, in our case $\mathcal{H}$. Let $x_0 \in \mathcal{H}$ be the initial set of predictors, $T_0 \in (0, \infty)$ be the initial temperature of the system and $i_\mathrm{max} \in \mathbb{N}$ be the maximal number of time steps. Then the algorithm requires the following functions:

- $\mathrm{cost}\colon \mathcal{H} \to \mathbb{R}$
  Calculates the cost of a given predictor set.
- $\mathrm{temp}\colon \mathbb{R} \times \mathbb{N}^2 \to (0, \infty)$
  Calculates the temperature according to the given initial temperature and time steps. It is a monotonically decreasing function in the second parameter.
- $\mathrm{nbr}\colon \mathcal{H} \to \mathcal{H}$
  Generates a random neighbor of a given predictor set.
- $\mathrm{prob}\colon \mathbb{R}^2 \times (0, \infty) \to [0, 1]$
  Calculates the probability of changing the current set or state to the neighbor.
- $\mathrm{rnd}(0, 1)$
  Returns a random number in the interval $[0, 1]$.

The listing shows one variant of the pseudocode of the SANN algorithm.

Listing: SANN algorithm

```
c_0 = cost(x_0)

for (i = 1, i ≤ i_max) {
    T = temp(T_0, i, i_max)

    x_1 = nbr(x_0)
    c_1 = cost(x_1)

    if (prob(c_0, c_1, T) ≥ rnd(0, 1)) {
        x_0 = x_1
        c_0 = c_1
    }
}
```

3.6 Model Validation

3.7 Simulation

In the following section, we assume that μ and σ^2 are known. true SPSE To generate pseudoobservations, we use the selected model's estimators instead μ and σ^2 by adding a Gaussian error term to the predictions by the selected model. Then we forget both and proceed to select a best model and estimate its parameters for each simulation. We then proceed to calculate the $\widehat{\mathrm{SPSE}}$ for each simulation. We then compute the arithmetic mean of the $\widehat{\mathrm{SPSE}}$ and compare it to the known true SPSE.

4 Implementation

4.1 Choosing a Neighbour

We stated in section 3.5 that we want to select a "good" model for the prediction. To this goal, we have to define the functions and parameters of the algorithm. The most important one is the nbr function whose purpose is to choose a neighbour efficiently since the final solution depends on a sequence of neighbours. In most cases it is best to select a neighbour not too far away from the given subset.

Our nbr-function generates a random natural number $r \in \{2, \ldots, k+1\}$ that represents the index of a measured wavelength. If this predictor is already in our current subset then we remove it. If not, we include it to the new subset. That way, new neighbours are not too far away from the current parameter vector. The pseudocode is shown in the following listing.

4.2 Additional Functions

All other functions were defined following a standard scheme. It follows from 3.4 that

$$\mathrm{cost}(M) \coloneqq {\mathrm{C_p}}^{(M)}$$

In most applications prob is defined in analogy to the transition of a physical system.

$$\mathrm{prob}(c_0, c_1, T) \coloneqq \exp\left(\frac{c_0 - c_1}{T}\right)$$

Details of temp are not really important as long as it monotonically decreases in the second parameter. So let $\alpha \in (0, 1)$.

$$\mathrm{temp}(T_0, i, i_{\max}) \coloneqq T_0 \alpha^i$$

4.3 Preprocessing

From 3 becomes that we have plenty of computations to perform. We need to compute the RSS in order to compute the C_p. To compute the RSS we have to compute the $\hat{Y}$. difficult better: we then only have to precompute X'X, X'y [most important precomputations] we use a function of the indices to the models we also precompute the full model σ^2

we define an injective, monotone increasing function $\delta_M : \{0, \ldots, |M| - 1\} \to \{0, \ldots, k\}$ to describe the current model in terms of the indices of the included parameters. Then resulting X_M is full rank.

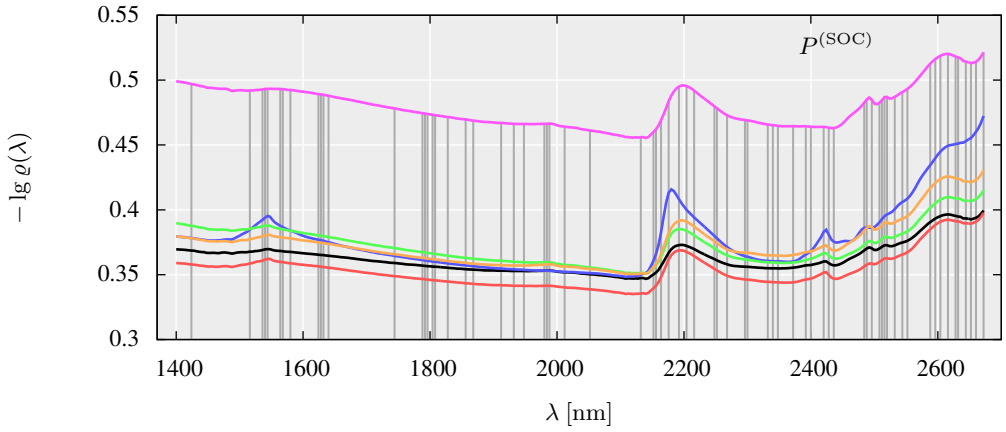
5 Model Selection

5.1 Calibration

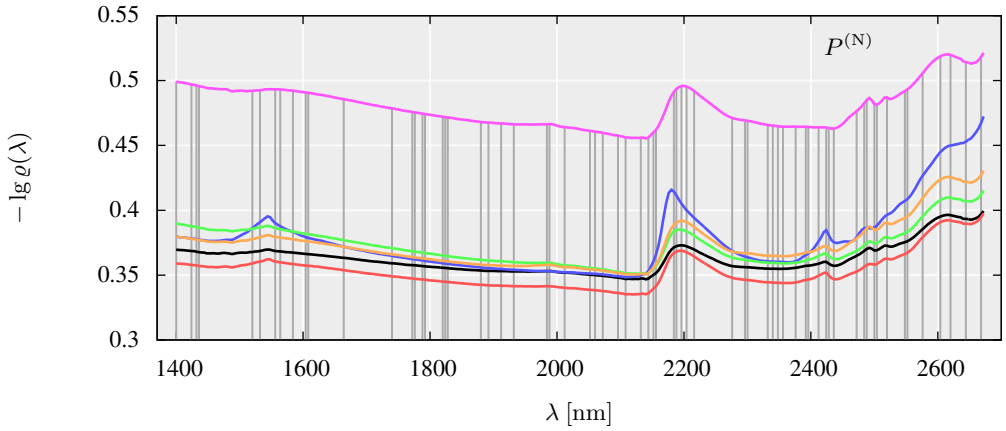
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.2 Goodness of Fit

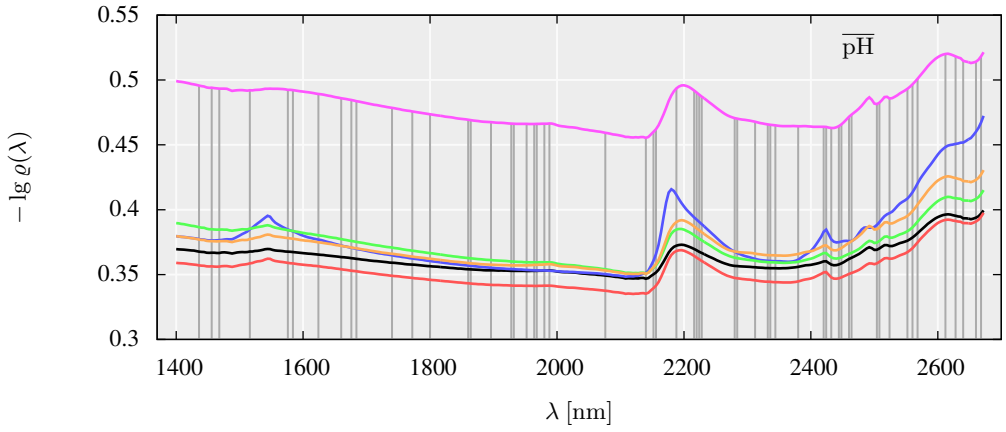
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



(a) $p^{(\mathrm{SOC})}$ with $\mathrm{p} = 72$

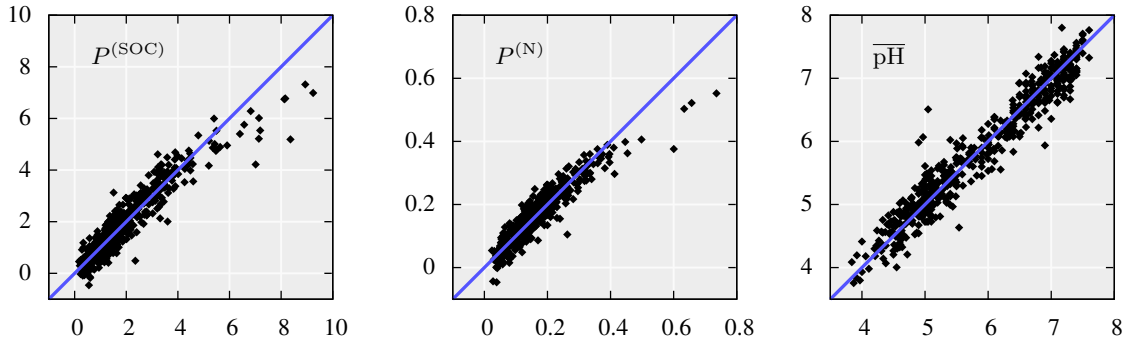


(b) $p^{(\mathrm{N})}$ with $\mathrm{p} = 69$



(c) $p^{(\mathrm{pH})}$ with $\mathrm{p} = 58$

Figure 2: Displaying the spectra from figure 1 with wavelength included in the selected models for each response highlighted by vertical grey lines



(a) $\hat{p}^{(\mathrm{SOC})} \sim p^{(\mathrm{SOC})}$
$(R^2)^{(\mathrm{SOC})} = 0.871$

(b) $\hat{p}^{(\mathrm{N})} \sim p^{(\mathrm{N})}$
$(R^2)^{(\mathrm{N})} = 0.861$

(c) $\hat{p}^{(\mathrm{pH})} \sim p^{(\mathrm{pH})}$
$(R^2)^{(\mathrm{pH})} = 0.940$

Figure 3: Correlation diagrams plotting $\hat{y}$ on y and the BLUE line representing the id_y

6 Simulation

7 Conclusion

References

A Prediction Parameters

Table 1: Estimated model parameters of $P^{(\mathrm{SOC})}$ on selected model

λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$
—	-1.47103	1808	1991.63	2204	-2319.71	2496	1956.13
1424	-811.326	1828	1568.91	2216	1075.21	2508	-5057.56
1516	953.795	1856	-2676.75	2248	2709.34	2512	5247.59
1536	1922.53	1868	1233.57	2252	-2048.76	2516	-4250.17
1540	-1869.18	1912	-1977.43	2268	-1583.37	2520	2886.04
1544	822.492	1932	1234.36	2296	-1973.03	2532	662.481
1564	-1390.49	1948	1194.36	2300	2819.68	2544	-2327.63
1568	896.58	1980	-2148.19	2332	-1751.19	2552	1768.67
1580	-822.831	1984	3493.29	2340	2932.77	2588	959.206
1624	1701.71	1988	-3187.35	2348	-1887.18	2596	-1089.79
1628	-2347.86	2012	1796.7	2372	2027.39	2604	-1012.46
1632	1901.09	2052	-2107.2	2392	-1418.5	2616	1163.92
1640	-1242.78	2132	1980.87	2400	827.711	2628	1104.94
1744	1615.77	2152	-2584.24	2420	-1222.28	2632	-850.572
1788	-2916.59	2156	1574.49	2428	1366.11	2644	-1025.13
1792	3481.22	2164	560.331	2436	-875.673	2652	-750.826
1796	-2024.07	2176	-1002.58	2484	1828.84	2660	543.211
1804	-1387.22	2192	1797.18	2488	-2413.53	2672	757.694

Table 2: Estimated model parameters of $P^{(\mathrm{pH})}$ on selected model

λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$
—	5.57628	1864	-508.298	2220	699.185	2460	545.634
1436	135.244	1896	-623.655	2224	-659.264	2464	-519.611
1456	-218.665	1928	749.04	2228	546.732	2504	318.913
1468	187.833	1932	-494.482	2280	277.156	2508	-207.677
1516	-135.26	1952	301.897	2284	-524.869	2524	-246.765
1576	-142.599	1964	304.809	2312	342.157	2552	-339.66
1584	321.27	1968	-381.735	2332	331.74	2560	292.801
1624	212.938	1980	390.898	2336	-424.459	2568	353.898
1660	-216.045	1988	-241.732	2344	-243.312	2612	-255.161
1676	-189.999	2076	-254.226	2380	249.445	2628	241.185
1684	-170.503	2140	104.988	2420	-184.771	2640	-302.791
1740	-254.442	2152	291.586	2424	114.651	2660	-183.544
1772	374.618	2156	-222.418	2432	162.078	2668	305.323
1800	319.283	2188	75.4672	2444	-735.52		
1860	368.757	2216	-576.162	2448	537.499		

Table 3: Estimated model parameters of $P^{(\mathrm{N})}$ on selected model

λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$
—	-0.0287506	1820	169.949	2156	95.2657	2428	116.231
1400	48.2214	1824	-272.304	2184	-99.54	2436	-60.6976
1424	-56.8242	1828	206.448	2188	122.224	2472	32.1805
1432	51.654	1880	100.173	2196	62.0394	2484	78.6657
1436	-93.4026	1892	-92.3753	2204	-203.738	2488	-98.8639
1520	52.0473	1912	-149.402	2216	131.551	2500	255.264
1532	59.5159	1932	78.6979	2276	-117.447	2504	-301.29
1556	43.1811	1984	100.883	2296	-99.3957	2520	66.8194
1564	-60.6404	1988	-163.811	2300	228.234	2548	-243.675
1584	-75.1836	2012	125.258	2332	-126.408	2552	208.634
1604	-77.3958	2052	-138.936	2340	174.779	2576	44.06
1608	111.822	2060	94.4386	2348	-179.621	2604	-77.1908
1664	-85.9411	2072	92.1081	2356	117.382	2620	69.4575
1740	84.6252	2096	-129.448	2376	72.808	2644	-91.9228
1772	106.938	2108	-157.724	2392	-136.592	2668	69.4473
1776	-88.21	2132	78.4591	2396	121.266		
1788	-148.498	2144	203.587	2412	-55.8742		
1792	105.402	2152	-224.185	2424	-72.9166		

B R Source Code

Listing: utils.r

```
# calculate the gram matrix of a given matrix
# gram.mat <- function(mat){
#   #return
#   t(mat) %*% mat
# }

# mlr.transf.obs.vec <- function(obs_vec, design_mat){

# }

# get matrix for calculating parameters
# mlr.par.mat = function(design_mat){
#   transp_design_mat <- t(design_mat)

#   # return
#   solve(transp_design_mat %*% design_mat) %*% transp_design_mat
# }

# calculate parameters
# mlr.par = function(obs_vec, design_mat){
#   # return
#   as.vector(mlr.par.mat(design_mat) %*% obs_vec)
# }

# initialize observables and design matrix (needed for fast calculation)
init.data = function(obs_vec, design_mat){
    gv_obs_vec <<- obs_vec
```

```
    gv_design_mat <<- design_mat
    gv_sample_size <<- length(gv_obs_vec)
    gv_par_size <<- dim(gv_design_mat)[2]

    # preprocessing
    gv_gram_design_mat <<- t(gv_design_mat) %*% gv_design_mat
    gv_transf_obs_vec <<- t(gv_design_mat) %*% gv_obs_vec
}

# generate pseudo observables (needs init.data; first use another function to calculate global
    variables: gv_expect_vec, gv_sd)
# gen.pseudo.obs.vec = function(){
#   # return
#   rnorm(gv_sample_size, gv_expect_vec, gv_sd)
# }

# initialize global variables for given observables and design matrix (needed for fast calculation
     of multiple linear regression and model selection)
mlr.init = function(){
    # transp_design_mat <- t(design_mat)

    # global variables
    # gv_mlr_design_mat <<- design_mat
    # gv_mlr_par_size <<- dim(design_mat)[2]
    # gv_mlr_gram_design_mat <<- transp_design_mat %*% design_mat

    # gv_mlr_obs_vec <<- obs_vec
    # gv_mlr_sample_size <<- length(obs_vec)
    # gv_mlr_transf_obs_vec <<- transp_design_mat %*% obs_vec
    # gv_mlr_expect_vec <<- gv_design_mat %*% gv_mlr_par_vec

    gv_mlr_par_vec <<- solve(gv_gram_design_mat, gv_transf_obs_vec)
    res_vec <- gv_obs_vec - (gv_design_mat %*% gv_mlr_par_vec)
    gv_mlr_var <<- ( as.numeric( t(res_vec)%*%res_vec) ) / (gv_sample_size - gv_par_size)
    gv_mlr_inv_var <<- 1.0 / gv_mlr_var

    # gv_mlr_var <<- gv_mlr_rss  / (gv_mlr_sample_size - gv_mlr_par_size)
}

#
ms.init.dist = function(idx_vec){
    par_vec <- solve(gv_gram_design_mat[idx_vec,idx_vec], gv_transf_obs_vec[idx_vec])

    # global: model selection expectation vector
    gv_expect_vec <<- as.matrix(gv_design_mat[,idx_vec]) %*% par_vec

    res_vec <- gv_obs_vec - gv_expect_vec

    # global: model selection standard deviation
    gv_sd <<- sqrt( as.numeric( t(res_vec) %*% res_vec ) / (gv_sample_size - length(idx_vec)) )
}

# initialize new observable with the same length (needs mlr.init)
# mlr.init.obs.vec = function(obs_vec){
#   gv_mlr_obs_vec <<- obs_vec
#   # gv_mlr_sample_size <<- length(obs_vec)
#   gv_mlr_transf_obs_vec <<- t(gv_mlr_design_mat) %*% obs_vec
#   gv_mlr_par_vec <<- solve(gv_mlr_gram_design_mat, gv_mlr_transf_obs_vec)
#   gv_mlr_expect_vec <<- gv_mlr_design_mat %*% gv_mlr_par_vec
#   gv_mlr_res_vec <<- obs_vec - gv_mlr_expect_vec
#   gv_mlr_rss <<- as.numeric(t(gv_mlr_res_vec)%*%gv_mlr_res_vec)
#   gv_mlr_var <<- gv_mlr_rss  / (gv_mlr_sample_size - gv_mlr_par_size)
#   gv_mlr_inv_var <<- 1.0 / gv_mlr_var
# }

# mlr.init.design.mat = function(design_mat){
```

```
# }

# get hat-matrix of a given design-matrix
# design_mat must have full rank
# mlr.hat.mat = function(design_mat){
#   # return
#   design_mat %*% mlr.par.mat(design_mat)
# }

# multiple linear regression residual sum of squares (rss)
# mlr.rss = function(obs_vec, design_mat){
#   hat_mat <- mlr.hat.mat(design_mat)
#   res <- obs_vec - (hat_mat %*% obs_vec)

#   # return
#   as.numeric(t(res) %*% res)
# }

# multiple linear regression variance estimator
# mlr.var = function(obs_vec, design_mat){
#   # return
#   (mlr.rss(obs_vec, design_mat)) / (length(obs_vec) - dim(design_mat)[2])
# }

ms.par.vec = function(idx_vec){
    #return
    solve(gv_gram_design_mat[idx_vec,idx_vec], gv_transf_obs_vec[idx_vec])
}

ms.expect.vec = function(idx_vec){
    par_vec <- ms.par.vec(idx_vec)

    # return
    as.matrix(gv_design_mat[,idx_vec]) %*% par_vec
}

# get residual sum of squares for given model (needs mlr.init)
ms.rss = function(idx_vec){
    par_vec <- solve(gv_gram_design_mat[idx_vec,idx_vec], gv_transf_obs_vec[idx_vec])
    res_vec <- gv_obs_vec - ( as.matrix(gv_design_mat[,idx_vec]) %*% par_vec )

    # return
    as.numeric( t(res_vec) %*% res_vec )
}

ms.spse.est = function(idx_vec){
    # return
    ms.rss(idx_vec) + (2 * gv_mlr_var * length(idx_vec))
}

# get mallows cp for certain model
ms.cp = function(idx_vec){
    # return
    (ms.rss(idx_vec) * gv_mlr_inv_var) + (2*length(idx_vec)) - gv_sample_size
}

# model selection: forward selection method
# ms.fwd.sel = function(obs_vec, design_mat, invgv_mlr_var){
#   full_idx_vec <- seq(1, dim(design_mat)[2])
#   # first column will be used every time
#   idx_vec <- 1
#   cp <- mallows.cp(obs_vec, design_mat, idx_vec, invgv_mlr_var)

#   repeat{
#       # vector of selection
```

```
#       sel_vec = setdiff(full_idx_vec, idx_vec)

#       if (length(sel_vec) == 0){
#           break
#       }

#       tmp_idx_vec_1 <- c(idx_vec, sel_vec[1])
#       tmp_cp_1 <- mallows.cp(obs_vec, design_mat, tmp_idx_vec_1, invgv_mlr_var)

#       for (i in 2:length(sel_vec)){
#           tmp_idx_vec_2 <- c(idx_vec, sel_vec[i])
#           tmp_cp_2 <- mallows.cp(obs_vec, design_mat, tmp_idx_vec_2, invgv_mlr_var)

#           if (tmp_cp_2 <= tmp_cp_1){
#               tmp_cp_1 <- tmp_cp_2
#               tmp_idx_vec_1 <- tmp_idx_vec_2
#           }
#       }

#       if (cp >= tmp_cp_1){
#           cp <- tmp_cp_1
#           idx_vec <- tmp_idx_vec_1
#       }else{
#           break
#       }

#       # debug information
#       print(idx_vec)
#       print(cp)
#   }

#   # return
#   idx_vec
# }

# model selection: simulated annealing: neighbour function
ms.sa.nbr = function(idx_vec){
    # get random index (1 is not used)
    rand_idx <- sample(2:gv_par_size, size = 1)

    if (rand_idx %in% idx_vec){
        # delete rand_idx in idx_vec
        nbr_idx_vec <- idx_vec[idx_vec != rand_idx]
    }else{
        # add rand_idx to idx_vec
        nbr_idx_vec <- c(idx_vec, rand_idx)
    }

    # return
    nbr_idx_vec
}

# model selection: simulated annealing: probability function
# costs will be minimized
ms.sa.prob <- function(old_cost, new_cost, temp){
    # return
    exp( (old_cost - new_cost) / temp )
}

# model selection: simulated annealing
ms.sa = function(idx_vec = c(1), temp = 100, alpha = 0.99, it_max = 10000, it_exit = 1200){
    old_cost <- ms.cp(idx_vec);
    it_same <- 0

    for (i in 1:it_max){
        nbr_idx_vec <- ms.sa.nbr(idx_vec);
```

```
        new_cost <- ms.cp(nbr_idx_vec)

        if ( ms.sa.prob(old_cost, new_cost, temp) >= runif(1) ){
            idx_vec <- nbr_idx_vec
            old_cost <- new_cost
            it_same <- 0
        }else{
            it_same <- it_same + 1
            if (it_same >= it_exit){
                break
            }
        }

        temp <- alpha * temp

        # debug
        # print(idx_vec)
        # print(old_cost)
        # print(temp)
    }

    # return
    idx_vec
}

#
ms.sim = function(expect_vec, var, sim_count = 10){
    sd <- sqrt(var)
    spse <- 0

    for (i in 1:sim_count){
        # generate and init pseudo observables
        gv_obs_vec <<- rnorm(gv_sample_size, expect_vec, sd)
        gv_transf_obs_vec <<- t(gv_design_mat) %*% gv_obs_vec
        mlr.init()

        # select model
        idx_vec <- ms.sa()
        tmp_spse <- ms.rss(idx_vec) + (2 * gv_mlr_var * length(idx_vec))

        # calculate spse
        spse <- spse + tmp_spse

        # debug
        print("sorted index vector:")
        print(sort(idx_vec))
        print("tmp spse:")
        print(tmp_spse)
        print("tmp mallows' cp:")
        print(ms.cp(idx_vec))
    }

    spse <- spse / sim_count

    # return
    spse
}
```