Statistical Methods:
Prediction of Soil Parameters through Near Infrared Spectroscopy

Kazimir Menzel
Markus Pawellek

August 25, 2016

Abstract

We describe and assess a procedure to construct and calibrate predictive models in NIRS measurements applied to three different response variables in soil samples. Constructing linear models from known physical and chemical properties, we calibrate them with respect to included predictors by formulating a discrete minimisation problem based on Mallow's C_{p}. Using SANN to solve the minimisation problem, we then assess the effects of two hyperparameters, resolution and training set size, on the method's perfrmance.

Contents

Statistical Methods:
Prediction of Soil Parameters through Near Infrared Spectroscopy

Kazimir Menzel
kazimir.menzel@me.com

Markus Pawellek
markuspawellek@gmail.com

Abstract

We describe and assess a procedure to construct and calibrate predictive models in NIRS measurements applied to three different response variables in soil samples. Constructing linear models from known physical and chemical properties, we calibrate them with respect to included predictors by formulating a discrete minimisation problem based on Mallow's C_p. Using SANN to solve the minimisation problem, we then assess the effects of two hyperparameters, resolution and training set size, on the method's perfrmance.

1 Introduction

Obtaining information on the composition of soil samples, such as the concentration of soil organic carbon (SOC), and other properties like the pH-value is important in answering a great variety of soil-related questions. The direct measurement of soil composition is costly and error-prone [McL99, 1-3]. Therefore methods that allow for fast and cheaply determination of soil properties play a fundamental and vital role in fields with a strong interest in soil analysis [LK01].

Developed in the 1960s, Near Infrared Spectroscopy (NIRS) gained traction in soil science during the 1990s as sufficient computing power became increasingly available. NIRS provides a cheap alternative to other methods for the analysis of soil composition [AHJ10, 247].

Following a short summary of the chemo-physical background of NIRS, we will present some of the peculiarities of its application. We then proceed to describe in detail the methods combined to calibrate an NIRS system on a training set employing Mallow's C_p. To assess the robustness of the method, we investigate the algorithm aided by simulations and variations of two hyperparameters, resolution and training set size. We conclude after a short discussion of the implementation in R [r-f16] and our results with an outline of additional investigations that seem promising for improved results.

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 of A is given by

$$c_A := \frac{n_A}{V}$$

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

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

In this report, we concentrate on three important soil parameters that are given by existing NIRS measurements [AHJ10, Don]. 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 $c_{\mathrm{H_3O^+}}$.

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

2.2 Near Infrared Spectroscopy

NIRS uses electromagnetic waves [AHJ10, 246], famously known as light, with wavelengths ranging from $780\,\mathrm{nm}$ to $3000\,\mathrm{nm}$, the so called near infrared spectrum. 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 [AHJ10, 247-8].

Reflection can be split in specular and diffuse reflection. NIRS uses the latter as it penetrates the sample the most. By 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 [AHJ10, 248-251].

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 [AHJ10, 247].

Due to the similarity of diffuse reflected and transmitted light we can use Beer-Lambert's law as an approximation of the relation of the attenuation of light and the properties of samples [AHJ10, 247-8]. 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 a training 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 [Don].

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 $k < n$ 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 [AHJ10, 247-8, 254]. 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

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

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.

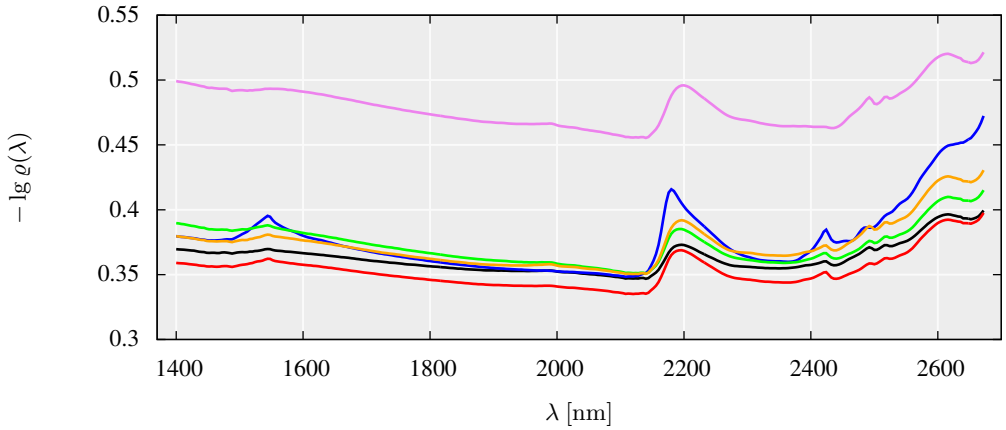


Figure 1: Six near infrared soil spectra of randomly chosen soil samples obtained from the data set, where λ is the wavelength and $\rho(\lambda)$ the corresponding reflectance and each colour refers to one sample

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

Multivariate linear regression (MLR) is a statistical method for estimating parameters of linear relations between a response variable and a set of predictors. These can be reused to predict new responses [DS98, Par12, Sch16b]. Let $\mathbb{X} \in \mathbb{R}^{n\times(k+1)}, n, k \in \mathbb{N}, k < n$ be the design matrix with $\operatorname{rank}\mathbb{X} = k+1$, $\sigma^2 \in (0,\infty)$ and Y be the random vector of a response variable 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

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

3.4 Mallows' C_{p}

At this point, the model is specified using $k+1 = 320$ predictors for each response variable, using the whole domain of measured spectrum for each soil sample. This set of predictors is comparatively large, $n-k < k$. Further, the reflectances of neighbouring lightwaves are correlated [AHJ10, 252]. In these circumstances, we might overfit the data, 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 transform into selecting 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 $\mathrm{p} \in \{1,\ldots,k+1\}$ corresponds to the number of predictors included in M.

The sum of predicted squared errors (SPSE) is a theoretical criterion to compare the merits of different models. The SPSE measures how well a model does in predicting new responses for new observations Y_{n+i} which are iid to Y_i for $i \in \mathbb{N}, i \le n$ and is given by

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

which simplifies to [Sch16b, 29-30]

$$\mathrm{SPSE}^{(M)} = n\sigma^2 + \mathrm{p}\sigma^2 + \left(\mathrm{bias}^2\right)^{(M)}$$

where bias describes the divergence of the estimated expectation from the true expectation. As bias is decreasing in the number of parameters included in M, we assume it to be sufficiently close to zero to neglect it from further considerations [Sch16a].

Unfortunately, the true σ^2 is unobservable so that it has to be estimated by

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

Instead of using the $\widehat{\mathrm{SPSE}}^{(M)}$ directly, we will use Mallow's C_p instead, given by

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

Minimising this value is equivalent to the minimisation of the $\widehat{\mathrm{SPSE}}^{(M)}$.

3.5 Simulated Annealing

We can now formulate the following minimisation problem. Let

$$\mathcal{H} \coloneqq \{A \cup \{\lambda_0\} \mid A \in \mathcal{P}(\Lambda)\} \subset \mathcal{P}\left(\overline{\Lambda}\right)$$

Then the model we want to select is a solution to

$$\Omega \coloneqq \text{minimise } \left\{ C_\mathrm{p}^{(M)} \;\middle|\; M \in \mathcal{H} \right\}$$

This problem is a discrete optimisation problem, which is in general np-hard [Sch98, 245-50]. Unfortunately, $|\mathcal{H}| = 2^{319}$, which is too large for a complete search. Therefore, we need a heuristic search algorithm that does not depend on further information of the cost function.

Simulated annealing (SANN) is such an algorithm that approximates global optima of a given function [PFTV07, 549-54]. Note that the SANN returns only a probabilistically good approximation $\mathcal{M} \in \mathcal{H}$ to a true solution to Ω. Lacking similarly reliable and computationally feasible alternatives we use the solution returned by SANN instead.

The algorithm is applicable to arbitrary sets, in our case $\mathcal{H}$. It simulates the slow cooling of a thermodynamic system. 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 neighbour 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 neighbour.
- $\mathrm{rnd}(0,1)$
  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

To examine the models selected by SANN, we recur to the often used R^2 measure [DS98, 33-4]. For $M \in \mathcal{H}$ it is given by

$$\left(R^2\right)^{(M)} \coloneqq \frac{\sum_{i=1}^{n}\left(\hat{y}_i^{(M)} - \overline{y}\right)^2}{\sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2} \in [0,1]$$

where $\overline{y}$ is the arithmetic mean of y_i for $i = 1,\ldots,n$ and describes how much of the total variation of y is explained by the model M. $\left(R^2\right)^{(M)} = 0$ is equivalent to no, $\left(R^2\right)^{(M)} = 1$ to full agreement of the model with the observations.

We shall note though, that the measure is increasing with $|M|$. In contrast the actual upper bound of R^2 is lowered by the presence of measurement errors. We will therefore consult the correlation diagrams for each response variable observation vector and its respective prediction vector to complement our judgement based on the R^2 measure alone.

3.7 Assessment by Simulation

As the true SPSE is unobservable, we need to assess how well our procedure minimises the true SPSE in general. For limitations of available data we have to resort to so called pseudo-observations of a response variable, which we collect in a random vector as follows

$$\widetilde{Y} \coloneqq \hat{y}^{(\mathcal{M})} + \varepsilon, \qquad \varepsilon \sim \mathcal{N}\left(0, \sigma^2 \mathrm{I}_n\right)$$

where we assign

$$\sigma^2 \coloneqq \left(\tilde{\sigma}^2\right)^{(\mathcal{M})}$$

Having fixed the parameters for the random vector $\widetilde{Y}$ we can now calculate the true SPSE from them by

$$\mathrm{SPSE}^{(\mathcal{M})} = (n + |\mathcal{M}|)\,\sigma^2$$

We can now simulate a new application of our model selection algorithm for

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

treating σ^2 as unknown. The algorithm returns a model $\widetilde{\mathcal{M}}$. We can now calculate $\widehat{\mathrm{SPSE}}^{(\widetilde{\mathcal{M}})}$ and compare it to the true SPSE. Due to the heuristic nature of the selection algorithm, we might end up with an exceptionally bad value. To address this, we repeat this process $q \in \mathbb{N}$ times [Sch16a].

In addition to parameters of the random vector $\widetilde{Y}$, $\widehat{\mathrm{SPSE}}^{(\widetilde{\mathcal{M}})}$ depends on the hyperparameters, resolution $k = |\Lambda|$ and the number of observations, n, as well. To gauge influences of the resolution on the model selection, we augment the procedure above to include the sets

$$\Lambda^{(m)} \coloneqq \{\lambda_{m \cdot i} \in \Lambda \mid i \in \mathbb{N}, m \cdot i \leq k\}$$

where $m \in \{1, 2, 3, 4\}$ and $\Lambda^{(m)}$ takes replace of Λ in 3.4 and 3.5.

To assess the influence of the number of available measurements, we choose randomly selected subsets of the existing measurements of sizes $n \in \{100, 200, 300, 500, 533\}$ and augment the simulation accordingly. In this way, we can compare the $\widehat{\mathrm{SPSE}}^{(\widetilde{\mathcal{M}})}$ with the "true" model's SPSE in 19 situations and on the same r pseudo-observations for each.

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 efficiently choose a neighbour since the final solution depends on the 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 $r \in \{1, \ldots, k\}$ 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.

Listing: nbr function

```
function nbr(M){
    r = rnd {1, ..., k}

    if (λ_r ∈ M){
        M̃ = M \ {λ_r}
    }else{
        M̃ = M ∪ {λ_r}
    }

    return M̃
}
```

4.2 Additional Functions

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

$$\mathrm{cost}(M) \coloneqq C_{\mathrm{p}}^{(M)}$$

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

Implementing the algorithm described in 3.5 takes a sizeable toll on computing power. The most expensive calculations are performed in the computation of the residual sum of squares

$$\left\| y - \mathbb{X}^{(M)} \hat{\beta}^{(M)}(y) \right\|_2^2$$

which requires the computation of

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

Instead of solving this, it is more efficient to solve

$$\mathbb{X}^{(M)^{\mathrm{T}}}\mathbb{X}^{(M)}\hat{\beta}(y) = \mathbb{X}^{(M)^{\mathrm{T}}}y$$

which can be done through QR-decomposition or an adequate alternative.

The design matrices $\mathbb{X}^{(M)}$ are full rank by construction. It follows then from the definition of M that we can define an injective, monotone increasing function

$$\delta_M : \{0, \ldots, |M| - 1\} \to \{0, \ldots, k\}$$

that maps the indices of the design matrix $\mathbb{X}^{(M)}$ to the indices of the full design matrix $\mathbb{X}$. We then only have to precompute $\mathbb{X}^{\mathrm{T}}\mathbb{X}$ and $\mathbb{X}^{\mathrm{T}}y$ and construct $\mathbb{X}^{(M)^{\mathrm{T}}}\mathbb{X}^{(M)}$ and $\mathbb{X}^{(M)^{\mathrm{T}}}y$ by

$$\begin{aligned}\mathbb{X}^{(M)^{\mathrm{T}}}\mathbb{X}^{(M)} &= \left(\left\langle x_{\delta(i)}, x_{\delta(j)}\right\rangle\right) \\ \mathbb{X}^{(M)^{\mathrm{T}}}y &= \left(\left(\mathbb{X}^{\mathrm{T}}y\right)_{\delta(i)}\right)\end{aligned}$$

for all $i, j \in M$ where $x_{\delta(i)}$ is the $\delta(i)$th column vector of $\mathbb{X}$.

The estimate $\tilde{\sigma}^2$ is independent from M, so we can precompute the estimated variance of the complete model and its inverse as well.

5 Calibration

5.1 Model Selection

SANN returned as minimum values for the respective models

$$\begin{aligned}C_{\mathrm{p}}^{(\mathrm{SOC})} &= -17.46 \\ C_{\mathrm{p}}^{(\mathrm{N})} &= -28.51 \\ C_{\mathrm{p}}^{(\mathrm{pH})} &= -57.76\end{aligned}$$

Figures 3a, 3b and 3c in appendix A show the selected wavelengths included in the corresponding models highlighted in grey against the spectra from figure 1. One notes that the density of selected lightwaves varies strongly along the wavelengths for all three response variables, where $p^{(\mathrm{SOC})}$ uses the most and pH the lowest amount of predictors.

For $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$, we find that similar regions seeminlgy relevant for prediction, albeit the predictors for $p^{(\mathrm{N})}$ appear to be slightly more evenly distributed along the whole domain of measured wavelengths. In the selected model for $p^{(\mathrm{SOC})}$, predictors are highly concentrated in the regions from 1550 - 1650 nm, 1790 - 1810 nm, 1980 - 1990 nm, around 2500 nm and between 2600 and 2672 nm.

The distribution of predictors for the pH appears to be visibly distinct from the other two models. In particular, the lower-middle wavelengths seem to have more predictive power than for $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$.

Appendix A hosts the tables with the estimated parameters for the predictors for each model. It is noteworthy that the the value of the intercept for pH lies in a mildly acidic range.

5.2 Goodness of Fit

Figure 2 displays the correlation diagrams introduced in 3.6 for each response variable together with the values for the R^2. Both indicate that our estimation and model selection to predict pH are working best. Taking measurement error into account, an $R^2 = 0.940$ shows a good accordance between predictions by the model and the actual observed values. The clear pattern in diagram 2c corroborates these findings.

We have already seen on several occasions that pH has to be treated slighted different from the other two response variables. The usual divergence between the patterns for the pH and the other two response variables emerges here as well. $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$ display not only lower values for R^2, but their correlation diagrams indicate a coherent divergence from the id.

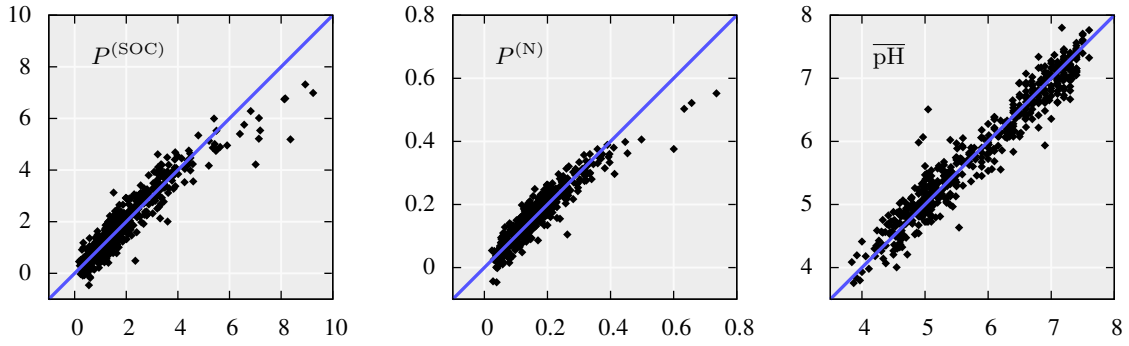
Our predictions seem to underestimate the observed values for the upper third of the observed interval. This could be due to a lack of data, which are considerably scarcer in that region than in the regions where the fits appear good. Only less than 5 % of the whole measurements lie in the upper tiers of both response variables.

Another reason might be that the linear assumption is misplaced, by judging from the visual aspects of the scatter plot alone. As we have strong reason to maintain the linear hypothesis (2.2) and taking into account that for most values the correlation seems even better than in the case of pH, we maintain that it is safe to assume higher measurement errors in as $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$ increase, which in turn contributes to lower R^2 values. In sum, there is not sufficient evidence to reject the selected and estimated models at this point.

6 Simulation

The true SPSE for all simulations calculated from the selected model is 172.85. Applying the model selection algorithm to 1000 pseudo-observations of $p^{(\mathrm{SOC})}$, we find that we underestimate the true SPSE by a margin of 20.22. This is more than one standard deviation below the true value. As we decrease the number of used observations, this difference becomes increasingly and monotonously steeper as shown in the first column of table 1.

Reducing the resolution in contrast leads to overestimation of the SPSE. Again, the difference between mean estimated SPSE and the true value is here monotone and in-



(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) $\widehat{\mathrm{pH}} \sim \mathrm{pH}$ $(R^2)^{(\mathrm{pH})} = 0.940$

Figure 2: Correlation diagrams plotting $\hat{y}$ on y and the blue line representing the id

Table 1: Simulation measurements reporting the mean estimated SPSE in the upper entry of each cell, its standard deviation in the lower entry with n corresponding to the number of measurements included and m to the resolution at a true $\mathrm{SPSE} = 172.85$

n	$m=1$	$m=2$	$m=3$	$m=4$
533	152.63	192.47	226.85	250.58
	12.26	12.94	14.64	15.23
500	142.92	180.17	212.31	237.15
	12.03	12.63	14.39	14.81
400	112.31	138.81	168.64	184.81
	12.28	11.26	12.64	13.28
300	—	104.82	126.13	139.38
	—	10.44	11.47	11.93
200	—	66.34	78.83	89.46
	—	10.15	9.54	10.08
100	—	—	—	40.45
	—	—	—	9.10

creasing as can be seen in the first row of table 1.

Note that the standard deviation (and hence the variance) is increasing in m but decreasing in n. Albeit the estimates for the SPSE differ strongly, standard the deviation remains rather stable and within a close band around 12.5 ± 3

The closest approximation to the true SPSE is reached at 400 measurements and at a resolution of only a third of the original one. This indicates a strong dependence of the calibration on the original measuring setup.

7 Conclusion

Using Mallow's C_p criterion, we calibrated three predictive models for the soil parameters $p^{(\mathrm{SOC})}$, $p^{(\mathrm{N})}$ and pH. To construct these models, we recurred to the physical relationship between these parameters and the NIRS according to Beer-Lambert's law obtaining three linear models. We then calibrated each models with respect to the wavelengths included and their respective parameters by solving a minimisation problem based on Mallow's C_p through simulated annealing.

To assess the effectiveness of our model selection algorithm, we conducted 1000 simulations, each in 19 different setups using the estimated model for $p^{(\mathrm{SOC})}$ and estimated the precision of estimating the "true" SPSE through the algorithm. We found that the resolution and the number of measurements used for estimation both affect the performance of the model selection and estimation strongly. There were further strong signs for high dependance on measurement precision. This indicates a strong need for careful and attentive *a priori* collection of soil data with a particular focus on measurement precision, resolution and training set size.

Nonetheless, the here presented calibration method yields model parameters that are reasonably good for new predictions as long as the need for precision and accuracy does not exceed the limits discussed above. Certainly, these limits are difficult to capture within the design of this investigation and hence retain more the quality of a cautionary note.

In addition to the above, we strongly recommend to compare the here discussed model selection algorithm used for calibration with alternative approaches. Such approaches should vary the algorithm used to solve the minimisation problem in 3.5, the minimisation problem itself, by replacing Mallow's C_p with other criteria and by modifying the model itself by using an approach that estimates better parameters in terms of prediction such as the ridge regression [Sch16a].

References

[AHJ10] Lidia Esteve Agelet and Charles R Hurburgh Jr. A tutorial on near infrared spectroscopy and its' calibration. *Critical Reviews in Analytical Chemistry*, 40:246–260, 2010.

[Don] A Don. Nir data. data set.

[DS98] Norman R Draper and Harry Smith. *Applied regression analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, 3rd edition, 1998.

[LK01] B Ludwig and PK Khanna. Use of near infrared spectroscopy to determine inorganic and organic carbon fractions in soil and litter. In *Assessment Methods for Soil Carbon*, Advances in Soil Science, pages 361–370. CRC Press, 2001.

[McL99] MJ McLaughlin. Soil testing – principles and concepts. In KI Peverill, LA Sparrow, and DJ Reuter, editors, *Soil Analysis – an Interpretation Manual*, pages 1–21. CSIRO Publishing: Melbourne, 1999.

[Par12] Iain Pardoe. *Applied regression modelling*. New York: Wiley & Sons, 2nd edition, 2012.

[PFTV07] William H Press, BP Flannery, SA Teukolsky, and WT Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge Univ. Press, New York, 3rd edition, 2007.

[r-f16] The r language, June 2016.

[Sch98] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[Sch16a] Jens Schumacher. Consultation on simulations. personal communication, August 2016.

[Sch16b] Jens Schumacher. Statistische verfahren. lecture script, 2016.

A Prediction Parameters and Models

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

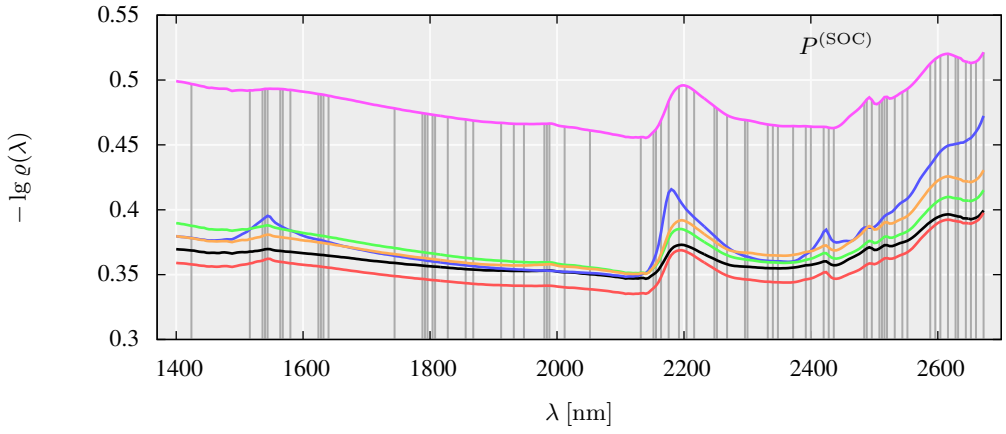
λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$
—	-1.47103	1808	1991.63	2204	-2319.71	2496	1956.13
1424	-811.326	1828	1568.91	2216	1075.21	2508	-5057.56
1516	953.795	1856	-2676.75	2248	2709.34	2512	5247.59
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 3: Estimated model parameters of $\overline{\mathrm{pH}}$ on selected model

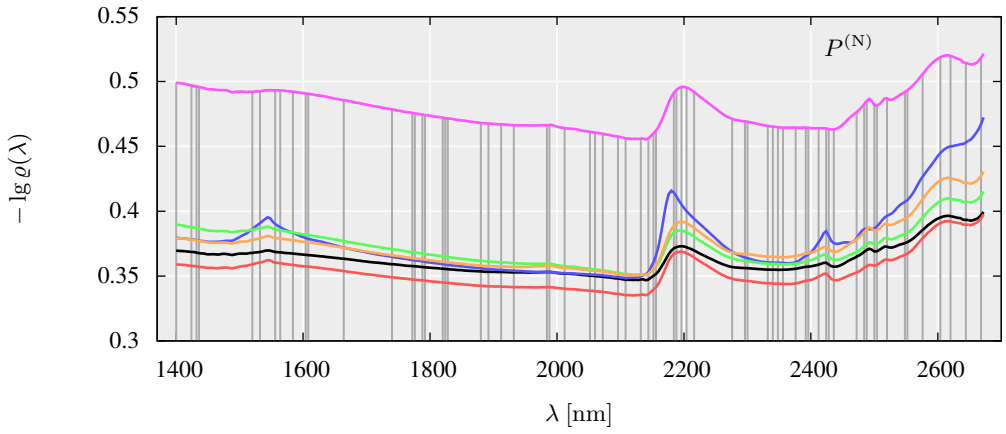
λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$
—	5.57628	1864	-508.298	2220	699.185	2460	545.634
1436	135.244	1896	-623.655	2224	-659.264	2464	-519.611
1456	-218.665	1928	749.04	2228	546.732	2504	318.913
1468	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 4: Estimated model parameters of $P^{(\mathrm{N})}$ on selected model

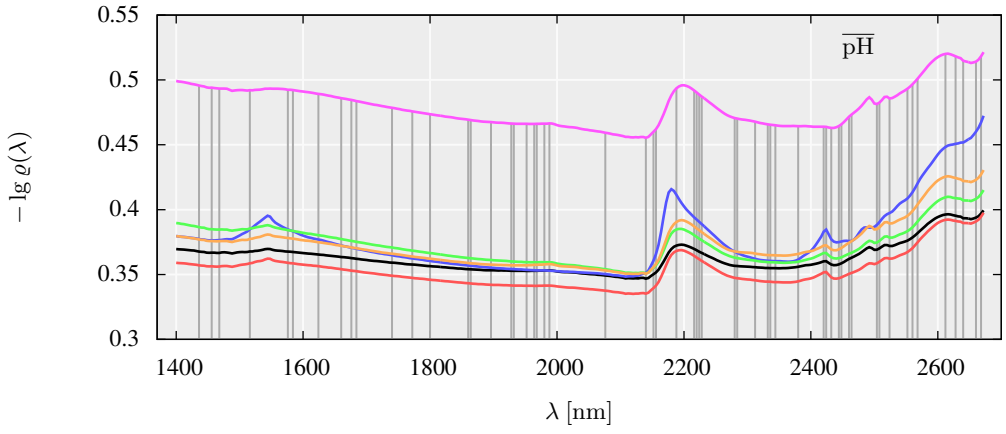
λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$
—	-0.0287506	1820	169.949	2156	95.2657	2428	116.231
1400	48.2214	1824	-272.304	2184	-99.54	2436	-60.6976
1424	-56.8242	1828	206.448	2188	122.224	2472	32.1805
1432	51.654	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		



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



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



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

Figure 3: Displaying the spectra from figure 1 with wavelength included in the selected models for each response highlighted by vertical grey lines

B R Source Code

Listing: utils.r

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

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

# precompute (estimate) parameter vector, variance and inverse variance of full model (needed for
     fast calculation)
mlr.init = function(){
    # global
    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
}

# initialize expectation vector and standard deviation for pseudo observations (as global
     variables: maybe faster)
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)) )
}

# estimate parameter vector for a certain model given through index vector (needs init, mlr.init)
ms.par.vec = function(idx_vec){
    #return
    solve(gv_gram_design_mat[idx_vec,idx_vec], gv_transf_obs_vec[idx_vec])
}

# estimate expectation vector for a certain model (needs init, mlr.init)
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 init, 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 )
}

# estimate spse for given model (needs init, mlr.init)
ms.spse.est = function(idx_vec){
    # return
```

```
    ms.rss(idx_vec) + (2 * gv_mlr_var * length(idx_vec))
}

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

# 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
}

# old variant
```

```
# 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
# }
```

Listing: init.r

```
# initialize random number generator
set.seed(NULL)

# read spectral soil data
soil_spec_data <- read.csv("../pro-files/data/soil-spec.csv", sep=";")

# define data dimensions
# number of wavelengths
wl_count <- dim(soil_spec_data)[2] - 3
# number of samples
sample_count <- dim(soil_spec_data)[1]
# number of observables
obs_count <- 3

# construct all wavelengths (only hard coded; future: has to be read from data)
wl_vec <- seq(1400, 2672, by = 4)
# get matrix of reflectance values
refl_mat <- as.matrix(soil_spec_data[,(obs_count+1):dim(soil_spec_data)[2]])
# get matrix of observables
obs_mat <- as.matrix(soil_spec_data[,1:obs_count])
# set vector of observables (easy to read and write)
soc_vec <- as.vector(obs_mat[,1])
n_vec <- as.vector(obs_mat[,2])
ph_vec <- as.vector(obs_mat[,3])

# design matrices
soc_design_mat <- cbind(1,refl_mat)
n_design_mat <- soc_design_mat
ph_design_mat <- cbind(1,log(refl_mat))
```

Listing: ms-soc.r

```
# load source files
source("utils.r")
source("init.r")


init.data(soc_vec, soc_design_mat)
mlr.init()

t1 <- proc.time()

idx_vec <- sort(ms.sa(it_max=100000,it_exit=12000,temp=10,alpha=0.995))

t2 <- proc.time()

# save relevant data
write.table(idx_vec, "../pro-files/data/gen/ms-sa-soc-idx-vec.csv", sep="\t", col.names=F, row.
    names=F)

wl_const_vec <- c(0, wl_vec)
par_vec <- ms.par.vec(idx_vec)
par_mat <- cbind(wl_const_vec[idx_vec], par_vec)
write.table(par_mat, "../pro-files/data/gen/ms-sa-soc-par.csv", sep="\t", col.names=F, row.names=F
    )

wl_idx_vec <- (idx_vec-1)[-1]
rnd_spec_data <- read.csv("../pro-files/data/soil-spec-rnd.csv", sep="\t", header=F)
write.table(rnd_spec_data[wl_idx_vec,], "../pro-files/data/gen/ms-sa-soc-spec-rnd.csv", sep="\t",
    col.names=F, row.names=F)

expect_vec <- ms.expect.vec(idx_vec)
corr_data <- cbind(gv_obs_vec, expect_vec)
write.table(corr_data, "../pro-files/data/gen/ms-sa-soc-corr.csv", sep="\t", col.names=F, row.
    names=F)

# arithmetic mean of obsevables
obs_mean <- sum(gv_obs_vec)/length(gv_obs_vec)
# total sum of squares
tss <- as.numeric( t(gv_obs_vec - obs_mean) %*% (gv_obs_vec - obs_mean) )
# reidual sum of squares
rss <- as.numeric( t(gv_obs_vec - expect_vec) %*% (gv_obs_vec - expect_vec) )
# explained sum of squares
ess <- as.numeric( t(expect_vec - obs_mean) %*% (expect_vec - obs_mean) )
# compute r^2 value
r2 <- 1-(rss/tss)

# output
print("index vector:")
idx_vec
print("index size:")
length(idx_vec)
print("mallows' cp:")
ms.cp(idx_vec)
print("estimated spse:")
ms.spse.est(idx_vec)
print("computed spse:")
rss * ( (length(idx_vec) + gv_sample_size)/(gv_sample_size - length(idx_vec)) )
print("r^2")
c( r2, ess/tss )
print("time:")
t2-t1
```

Listing: pseudo-soc.r

```
# load source files
source("utils.r")
source("init.r")


init.data(soc_vec, soc_design_mat)
mlr.init()

idx_vec <- as.vector(read.csv("../pro-files/data/gen/ms-sa-soc-idx-vec.csv", header=F)[,1])

idx_vec
length(idx_vec)

ms.init.dist(idx_vec)

# pseudo_obs_mat <- rnorm(gv_sample_size, gv_expect_vec, gv_sd)

# for (i in 2:1000){
#   pseudo_obs_mat <- cbind(pseudo_obs_mat, rnorm(gv_sample_size, gv_expect_vec, gv_sd))
#   print(dim(pseudo_obs_mat))
# }

# write.table(pseudo_obs_mat, "../pro-files/data/gen/pseudo-obs.csv", sep="\t", col.names=F, row.
    names=F)

gv_var <- gv_sd^2

# for usage in the simulation analysis
#introduced by Kaz.

gv_spse_soc <- (gv_sample_size + length(idx_vec)) * gv_var


print("expectation vector:")
gv_expect_vec
print("standard deviation:")
gv_sd
print("variance:")
gv_var
print("spse:")
gv_spse_soc
```

Listing: sim-soc.r

```
# load source files
source("utils.r")
source("init.r")


pseudo_obs_mat <- as.matrix(read.csv("../pro-files/data/gen/pseudo-obs.csv", header=F, sep = "\t")
    )
idx_vec <- as.vector(read.csv("../pro-files/data/gen/ms-sa-soc-idx-vec.csv", header=F)[,1])

# # pseudo_obs_mat
# dim(pseudo_obs_mat)
# idx_vec
# length(idx_vec)


init.data(soc_vec, soc_design_mat)
mlr.init()
```

```
ms.init.dist(idx_vec)


# choose predictor index
step <- 4
pred_idx_vec <- seq(1, dim(soc_design_mat)[2], by=step)

# choose observable index
sample_count <- 100
obs_idx_vec <- sample(dim(soc_design_mat)[1], sample_count)

#
init.data(soc_vec[obs_idx_vec], soc_design_mat[obs_idx_vec,pred_idx_vec])
mlr.init()


sim_count <- dim(pseudo_obs_mat)[2]
# sim_count <- 10

spse_vec <- numeric()

t1 <- proc.time()

for (i in 1:sim_count){
    tmp_t1 <- proc.time()

    # generate and init pseudo observables
    gv_obs_vec <<- as.vector(pseudo_obs_mat[obs_idx_vec,i])
    gv_transf_obs_vec <<- t(gv_design_mat) %*% gv_obs_vec
    mlr.init()

    # select model and estimate spse
    tmp_idx_vec <- ms.sa()
    tmp_spse <- ms.rss(tmp_idx_vec) + (2 * gv_mlr_var * length(tmp_idx_vec))
    spse_vec <- c(spse_vec, tmp_spse)

    tmp_t2 <- proc.time()

    # debug
    # print("sorted index vector:")
    # print(sort(tmp_idx_vec))
    print("tmp spse:")
    print(tmp_spse)
    # print("tmp mallows' cp:")
    # print(ms.cp(tmp_idx_vec))
    print("tmp time:")
    print(tmp_t2-tmp_t1)
    print("status:")
    print(i)
}

t2 <- proc.time()

print("spse arithmetic mean:")
print(sum(spse_vec) / sim_count)
print("spse variance:")
print(var(spse_vec))
print("time:")
print(t2-t1)


write.table(spse_vec, "../pro-files/data/gen/sim-soc-100-s4-spse-vec.csv", sep="\t", col.names=F,
     row.names=F)
```

Statutory Declaration

We herewith declare that we have completed the present report independently making use only of the specified literature and aids. Sentences or parts of sentences quoted literally are marked as quotations. Identification of other references with regard to the statement and scope of the work is quoted. The report in this form or in any other form has not been submitted to an examination body and has not been published.

Jena, 25th of August 2016

Kazimir Menzel Markus Pawellek