Statistical Methods:
Prediction of Soil Parameters through Near Infrared Spectroscopy

Kazimir Menzel
Markus Pawellek

August 22, 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

1 Introduction **1**

2 Background **1**
2.1 Soil Parameters 1
2.2 Near Infrared Spectroscopy 1

3 Methodology **2**
3.1 Measured Data 2
3.2 Statistical Model 2
3.3 Multivariate Linear Regression 3
3.4 Mallows' C_p 3
3.5 Simulated Annealing 4
3.6 Model Validation 4
3.7 Simulation 4

4 Implementation **5**
4.1 Choosing a Neighbour 5
4.2 Additional Functions 5
4.3 Preprocessing 5

5 Model Selection **6**
5.1 Calibration 6
5.2 Goodness of Fit 6

6 Simulation **8**

7 Conclusion **8**

A Prediction Parameters **i**

B R Source Code **ii**

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 [5, 1-3].

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 [4].

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 [1, 247].

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 demonstrate a computationally efficient way to identify near 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 of 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. [1, 2] 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, [1, 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 [1, 247-8].

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 [1, 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 [1, 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 [1, 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 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 [2].

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 [1, 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.

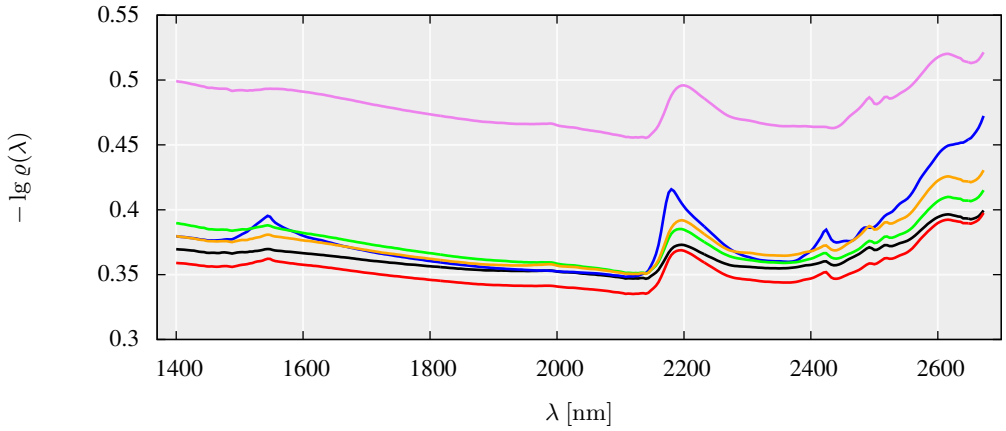


Figure 1: Six near infrared soil spectra of randomly chosen soil samples obtained from the data set, where λ is the wavelength and $\rho(\lambda)$ the corresponding reflectance and each colour refers to one sample

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 finding parameters of linear relations between a response variable and a set of predictors. These can be reused to predict new responses [3, 6, 9] 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 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

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

3.4 Mallows' $\mathrm{C_p}$

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. This set of predictors is comparatively large, $n-k < k$. Further, the reflectances of neighbouring lightwaves are correlated [1, 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 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 $\mathrm{p} \in \{2,\ldots,k\}$ 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 $\mathrm{SPSE}^{(M)}$ measures how well a model does in predicting new responses from new data:

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

which simplifies to [9, 29-30]

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

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

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

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

$$(\Omega) \qquad \underset{M \in \mathcal{H}}{\text{minimise}} \qquad C_\mathrm{p}^{(M)}$$

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

This problem is a discrete optimisation problem, which is in general np-hard [8, 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 a heuristic algorithm that approximates global optima of a given function [7, 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 [3, 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 be 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.

3.7 Simulation

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

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

As we have fixed the parameters for the random vector $\widetilde{Y}$ we can now calculate the true SPSE for the model.

we resort to simulation of pseudo response variable $\tilde{Y}^{(j)}, j \in \mathbb{N}, j \leq 1000$, where we can fix both, $\mathbb{E}\,\tilde{Y}^{(j)}$ and $C^{(j)}$. The resulting values for the response variable are also called pseudo-observations. We then apply our model selection algorithm to each simulation and compare the resulting estimators $\widehat{\mathrm{SPSE}}^{(j)}$ with the "true" SPSE computed from the fixed parameters.

The pseudo-observations of one simulation are generated as follows

where $\hat{y}$ is the selected model from 3.5 and $\left(\widehat{\sigma^2}\right)^{(\mathcal{M})}$ the estimator for σ^2 in said model. We then proceed to use the complete model space $\overline{\Lambda}$ to select the best model for each simulation and estimate the SPSE through $\widehat{\mathrm{SPSE}}^{(j)}$. We then compare the estimates with the fixed "true" SPSE.

To gauge how much the resolution of the NIRS influences the model selection, we repeat the model selection on the simulations, but allow only models from

$$\Lambda^{(m)} \coloneqq \{\lambda_{mi} | \lambda_{mi} \in \Lambda, i = 1, 2, \ldots\} \qquad m \in \{2, 3, 4\}$$

where $\overline{\Lambda}^{(m)}$ takes the place of $\overline{\Lambda}$ in the algorithm.

Just as with resolution, we can also assess the influence of availability of measurements. Choosing a suitable $N \in \mathbb{N}^L$ with entries $N_l > k, l \in \{1, \ldots, L\}$, we randomly select a subset of the simulation data of size N_l and proceed as above. As $n - k \leq k$, we perform this inquiry with $\Lambda^{(3)}$ instead to reduce computational cost and capture a larger variety of L.

Using the same algorithm for all three response variables, we can limit the algorithm examination to one response variable, SOC, to be better able to compare the results and reduce redundant analyses.

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.

Listing: nbr function

```
function nbr(M){
    r = rnd {2, ..., k+1}

    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

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

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

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

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

It is more efficient to compute $\hat{\beta}$ first and then proceed 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

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

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

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

where $x_{\delta(i)}$ is the $\delta(i)$th column vector of $\mathbb{X}$.

As $\left(\sigma^2\right)^{(\overline{\Lambda})}$ and by consequence $\left(\hat{\sigma}^2\right)^{(\overline{\Lambda})}$ are constant, we can precompute the estimated variance of the complete model as well.

5 Model Selection

5.1 Calibration

Figures 2a, 2b and 2c show the selected wavelengths highlighted in grey. One notes that the density of selected lightwaves varies strongly along the wavelength 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 seem relevant for prediction, albeit the predictors for $p^{(\mathrm{N})}$ appear to beslightly more evenly distributed along the whole bandwidth **[Is that the correct term?]**. 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 highly distinct from the other two models. In particular, the lower-middle wavelengths seem to have more predictive power than for $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$.

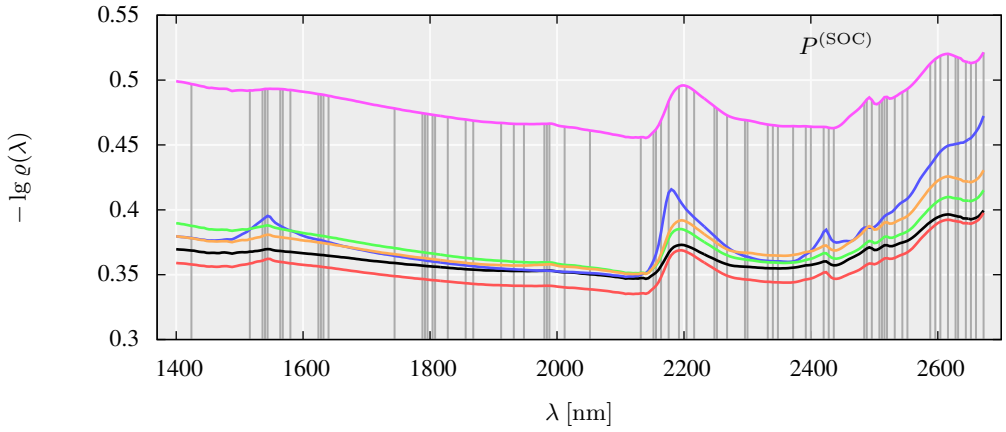
Appendix A hosts the tables with the estimated parameters for the predictors for each model. It is noteworthy to inspect the values of the intercepts for each model. We find that these appear relatively close to commonly known "neutral" figures – 0 in case of $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$ and not too far from 7 in case of the pH.

5.2 Goodness of Fit

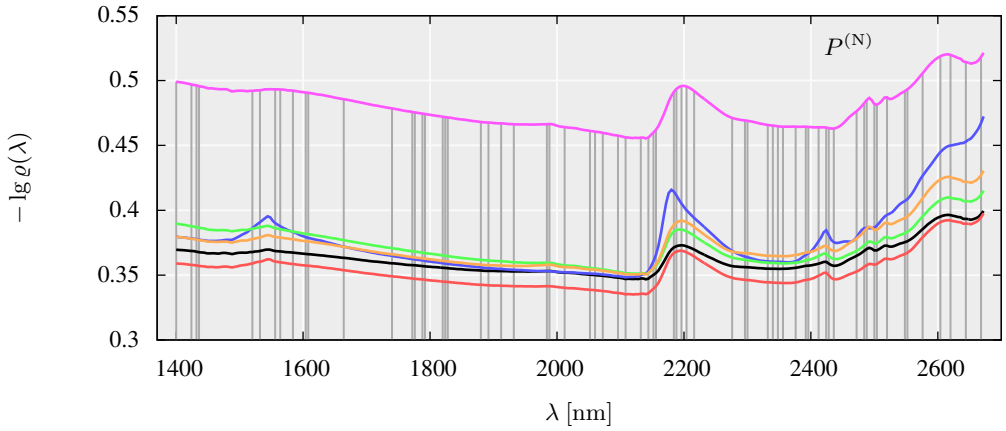
Figure 3 displays the correlation diagrams introduced in 3.6 for each response variable together with the values for the R^2. Both indicate that our estimation and model selection to predict pH are working best. Assuming that pure error exists due to measurement, an $R^2 = 0.940$ shows a good accordance between predictions by the model and the actual observed values. The clear pattern in diagram 3c corroborates these findings further.

We have already seen on several occasions that pH has to be treated slighted different from the other two response variables. This pattern repeats itself here as well. $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$ display not only somewhat lower values for R^2, but their correlation diagrams show a common divergence from the identity. 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.

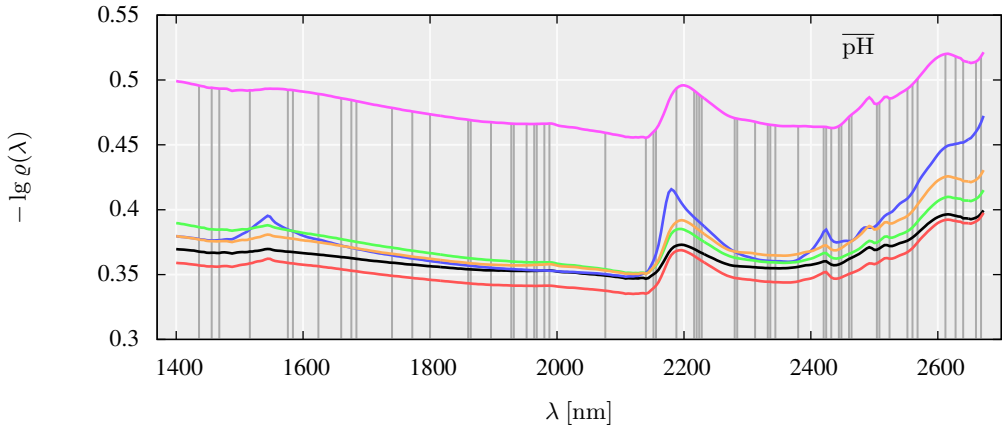
Another reason might be that the linear assumption is misplaced, by judging from the visual aspects of the scatter plot alone. As we have strong reason to maintain the linear hypothesis (cf. 2.2) and taking into account that for most values the correlation seems even better than in the case of pH, we 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 higher levels of pure error and thus to lower values of R^2. In sum, there is not sufficient evidence to reject the selected and estimated models at this point.



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

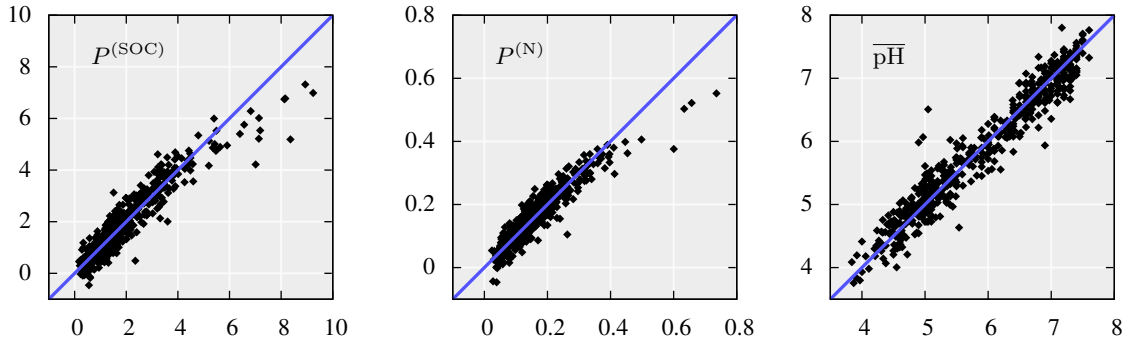


(b) $p^{(\mathrm{N})}$ with $\mathrm{p}^{(\mathrm{N})} = 69$



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

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

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

6 Simulation

Applying SANN on the 1000 simulations with the full design matrix showed that the $\widehat{\mathrm{SPSE}}$ values underestimate the true SPSE as displayed in figure **??**. The mean $\widehat{\mathrm{SPSE}}$ lies about $\mathcal{X}$ below its true value. It shall be noted though that the true SPSE lies within one standard deviation of the estimate.

Reducing the resolution at which we measure the light-waves ameliorates the situation somewhat. Estimates lie much closer to the true SPSE, albeit the risk of obtaining outliers increases dramatically. The best result are yielded at a resolution of every $\frac{1}{2}$ of the original one.

Reducing the data set at resolution $\frac{1}{3}$ shows even more dramatic effects on the variance of the estimates around the true SPSE. Compared to the full data set estimates at the same resolution, we notice a considerably wider interval around the mean. This is to be expected as lesser data increases the risk of variance.

7 Conclusion

References

[1] 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.

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

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

[4] 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.

[5] 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.

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

[7] 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.

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

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

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 $\overline{\mathrm{pH}}$ 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 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
}
```