

Faculté des sciences

Development of a Supervised non-negative Matrix Factorization (NMF) method for multivariate data with application to chemical kinetics data

Author: **Julien VAN ROY**
Supervisor: **Prof. Bernadette GOVAERTS**
Readers: **Michel THIEL, Prof. Rainer VON SACHS**
Academic year 2020-2021

Contents

1	Introduction and objectives	7
2	General problem and case studies	9
2.1	Artificial dataset	9
2.1.1	True quantities	9
2.1.2	True profile	10
2.1.3	Observed spectra	11
2.1.4	Batch generation	12
2.2	Janssen dataset	13
2.3	Chemium dataset	13
3	Unsupervised methods	14
3.1	Principal Component Analysis (PCA)	15
3.1.1	Description	15
3.1.2	Methodology	15
3.1.3	Algorithms	16
3.2	Non-negative Matrix Factorization (NMF)	17
3.2.1	Aims	17
3.2.2	Applications	17
3.2.3	Similarity measures	19
3.2.4	Algorithms	20
3.2.5	Variety of NMF algorithms	22
3.2.6	Complexity	23
3.2.7	Initialization	24
3.2.8	Stopping criterion	24
3.2.9	Choice of r	24
3.2.10	R packages	24
3.3	Methods comparison on batch data	28
3.3.1	Individual approach	28
3.3.2	Global approach	28
3.3.3	Leave-one-batch-out approach	28
3.3.4	Case of centered data	28
4	Supervised methods and Prediction	30
4.1	Partial Least Squares (PLS)	31
4.1.1	Description	31
4.1.2	Methodology	31
4.1.3	Algorithms	34
4.1.4	Interpretation of the results	35
4.2	Supervised NMF in the literature	36
4.2.1	Supervised NMF for ICU mortality classification	36
4.2.2	Semi Supervised NMF for EEG classification task	36
4.2.3	Supervised NMF for spectrometry data classification	37
4.2.4	Discussion and comparison	39
4.3	New algorithm for Supervised NMF	40
4.3.1	Objective function	40
4.3.2	Deriving the gradients	40
4.3.3	Gradient descent rules	40

4.3.4	Multiplicative Update rules	41
4.4	Prediction and methods comparison on new dataset	44
4.4.1	One step PLS	44
4.4.2	Predicting a new observation after the (S)NMF	44
4.4.3	Two steps (NMF/SNMF) + regression	45
4.4.4	Model quality assessment	46
5	Methods application on case studies	47
5.1	Methodology	47
5.1.1	Structure	47
5.1.2	Objective	48
5.1.3	R packages implementation	48
5.2	Artificial dataset	49
5.2.1	SNMF parameters	49
5.2.2	Analyze of the SNMF algorithms	50
5.2.3	Data visualization and decomposition	52
5.2.4	Prediction	56
5.3	Janssen dataset	59
5.4	Chemium dataset	60
5.4.1	SNMF parameters	60
5.4.2	Analyze of the SNMF algorithms	60
5.4.3	Data visualization and decomposition	62
5.4.4	Prediction	67
6	Further work and conclusion	73
6.1	Further work: Two steps (S)NMF + regression	73
6.2	Conclusion	75
7	Appendix	76
7.1	Non-negative Matrix Factorization (NMF)	76
7.2	Methods application on case studies	76
7.2.1	Janssen dataset	76
7.2.2	Chemium dataset	77
7.3	Code documentation	82
7.3.1	The functions files	82
7.3.2	Organizations of the case studies files	82
7.3.3	functions.R	83
7.3.4	CV_functions.R	85
7.3.5	utils.R	90
7.3.6	create_data.R	94
	Bibliography	96

List of Abbreviations

ALS	Alternating Least Squares
ANLS	Alternating Nonnegative Least Squares
ACLS	Alternating Constrained Least Squares
EEG	Electroencephalogram
GC	Gas Chromatography
GKLD	Generalized Kullback–Leibler divergence
HALS	Hierarchical Alternating Least Squares
ICU	Intensive Care Unit
LoBo	Leave-one-Batch-out
MALDI	Matrix-Assisted Laser Desorption/Ionization
MIR	Mid-Infrared (spectroscopy)
MSI	Mass Spectrometry Imaging
MU	Multiplicative Update (rules)
NIR	Near-Infrared (spectroscopy)
NMF	Non-negative Matrix Factorization
NNLS	Nonnegative Least Squares
NNSVD	Nonnegative Double Singular Value Decomposition
OLS	Ordinary Least Squares
PCA	Principal Component Analysis
PLS	Partial Least Squares
RMSECV	Root Mean Squared Error in Cross-Validation
SNMF	Supervised Non-negative Matrix Factorization
SVD	Singular Value Decomposition
SED	Squared Euclidean Distance
UPLC	Ultra-Pressure Liquid Chromatography

Notations

$\circ$	Component-wise multiplication, $(A \circ B)_{ij} = A_{ij}B_{ij}$
$\frac{[.]}{[.]}$	Component-wise division, $\left(\frac{[A]}{[B]}\right)_{ij} = \frac{A_{ij}}{B_{ij}}$
$(.)^T$	Transpose of a matrix, $(A^T)_{ij} = A_{ji}$
∇f	Gradient of the function f
$\mathbf{1}_{\mathbf{n} \times \mathbf{p}}$	Vector of ones of dimension $n \times p$
I_n	Identity matrix of dimension n
$\ \cdot\ _1$	$\ell 1$ -norm, $\ a\ _1 = \sum_{i=1}^n a_i $
$\ \cdot\ _2$	$\ell 2$ -norm, $\ a\ _2 = \sqrt{\sum_{i=1}^n a_i^2}$
$\ \cdot\ _F$	Frobenius norm, $\ A\ _F = \sqrt{\sum_{i=1}^n \sum_{j=1}^p A_{ij}^2}$

Acknowledgements

I would like to thank my supervisor Pr. Bernadette Govaerts and Michel Thiel for their support, help and the numerous meetings that enabled me to make progress in this master's thesis. I would also like to thank my parents for their unconditional support.

Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

Chapter 1

Introduction and objectives

Monitoring chemical reactions is crucial in the pharmaceutical and chemical industry. For this purpose, several types of strategies exist: off-line or on-line techniques.

The off-line technique typically consists in sampling in the reactor and analyze the sample in a laboratory further away from the reactor, yielding a time delay making it impossible to monitor a reaction in real time or to control a large amount of samples. The Ultra Pressure Liquid Chromatography (UPLC) is an example of off-line technique.

The on-line technique, on the other hand, uses a sensor to sample and analyze in close proximity to the reactor. Thus, monitoring the reaction in real-time is possible and there is no time delay. An example of on-line technique is spectroscopy, which can focus on different parts of the spectrum, for instance the Infrared (Near, Mid or Far depending on the spectral window) or Ultraviolet.

Nevertheless, the off-line technique will directly and precisely estimate the quality attributes in the reactor at the sampling time whereas the on-line technique will only generate a large amount of data which needs to be processed to eventually estimate the chemical concentrations.

Fortunately, methods exist to process high-dimensional and correlated variables (such as PLS), as well as to visualize and reduce the dimensionality in a meaningful way (such as PCA or NMF).

In this master thesis, the spectra resulting from an on-line technique such as Near-infrared spectroscopy (or NIR) are used to monitor the quality attributes of chemical reactions.

In this context, a new supervised penalty term is incorporated into an NMF objective function and the update rules are derived. Then, this new algorithm is applied on real and artificial datasets, along with existing unsupervised and supervised methods (PCA, NMF, PLS) which already proved their efficiency for this class of data. The new method, as an NMF, will produce scores and loadings (or basis vectors, coefficients vectors depending on the author).

The new proposed algorithm will be evaluated on the basis of two objectives:

1. The method should provide easily interpretable scores and loadings. That is to say, the scores should reflect the evolution of the chemical concentrations of each component and the loadings should reflect their true profiles. Regarding the scores, they can be compared to the true quantities by superimposing the true quantities measured off-line or known in advance in the case of an artificial dataset) and the scores on the same graph (possibly multiplied by a factor). For the loadings, they can be compared to the true profiles (known by chemists or by design in the case of an artificial dataset).
2. The prediction of the target chemical component(s) based on the scores and loadings produced should be accurate. To assess this property, the RMSE will be computed in a precise cross-validated scheme detailed in the next sections and several methods will also be compared. An extra step will also be required to produced regression coefficients based on the scores and loadings resulting from a

dimensionality reduction technique such as NMF or Supervised NMF. This will also be extensively discussed in the next sections.

The second chapter presents the general problem and the three datasets which will be used in this master's thesis.

Then, the chapter 3 introduces the unsupervised methods which will be applied to these datasets: the PCA and NMF. In addition, it also presents the methodology for comparing the reconstruction error implied by the scores and loadings (with the individual, global and leave-one-batch-out approaches).

Next, the chapter 4 describes the supervised methods: PLS and supervised NMF (SNMF) in the literature, and exposes the steps to construct a new SNMF algorithm with an objective function in line with the two objectives above. This chapter also explains how the different methods can be compared and used to predict the chemical concentrations (the “target” or “response” variable) in a one step or two steps approach.

Finally, the chapter 5 presents the whole methodology applied to the three datasets and shows the detailed results for each of them.

The conclusion contains the main findings and lessons learned from this thesis.

Chapter 2

General problem and case studies

Three different datasets will be used in this work to illustrate the methodology. Together, they are representative of the more general problem presented in the preceding section. Two datasets result from real experiments in the chemical industry, while the third was artificially generated. With those 3 examples, the aim is to provide an application framework as comprehensive as possible. The following sections detail them. The notation used is also introduced below.

X_G	Matrix of spectra ($n \times p$)
Y_G	Matrix of target concentrations ($n \times k$)
X_i	Matrix of spectra for batch/fold i ($n_i \times p$)
Y_i	Matrix of target concentrations for batch/fold i ($n_i \times k$)
$X_{G/i}$	Matrix of spectra for all but batch/fold i ($(n - n_i) \times p$)
$Y_{G/i}$	Matrix of target concentrations for all but batch/fold i ($(n - n_i) \times k$)

Special case: the target values y are not available for every spectra x

Y_G^* Matrix of target concentrations ($s \times k$) with $s < n$

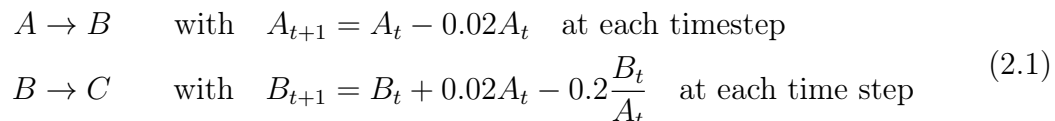
X_G^* Matrix of spectra corresponding to the target concentrations ($s \times p$)

2.1 Artificial dataset

The artificial dataset has been generated to reproduce a batch design where reactants are introduced once at the beginning of the reaction and then react during a certain amount of time to produce output chemical components.

In the following subsections, the construction of the dataset is thoroughly detailed. The result is a X_G matrix of dimensions 300×200 with X_i (50×200) stacked vertically. The target Y_G is of size 300×3 . Noise is added to the observed spectra but none is added to Y_G .

2.1.1 True quantities



The first step is to define the fictional chemical equations which will generate the observed spectra. In this reaction, 3 components will be present: a reactant, an intermediate reactant and a product (A, B, C respectively). The reactions will be performed in batches, over a determined number of timesteps. Thus, this gives the artificial equations 2.1. More formally, it yields the iterative update rules for the theoretical true quantities of each component over time, see equation 2.2.

$$\begin{aligned}
\Delta A_t &= 0.02A_t && \text{the consumed part of A at each timestep} \\
\Delta B_t &= 0.2\frac{B_t}{A_t} && \text{the consumed part of B at each timestep} \\
\text{Hence } \begin{cases} A_{t+1} = A_t - \Delta A_t \\ B_{t+1} = B_t - \Delta B_t + \Delta A_t \\ C_{t+1} = C_t + \Delta B_t \end{cases} &&& (2.2)
\end{aligned}$$

Using the above equations, one can generate the true quantities as shown in figure 2.1. At timestep 1, 100 units of component A are present, 0 units for components B and C , and the max timestep is set to 300. A speed factor is also added under the form of an additional multiplicative constant such that $\Delta A_t = 0.02A_t \times \frac{300\text{speed_factor}}{\text{max_timestep}}$ and $\Delta B_t = 0.2\frac{B_t}{A_t} \times \frac{300\text{speed_factor}}{\text{max_timestep}}$. Consequently, it allows to vary the maximum number of timestep and keeping the evolution of each component as shown in figure 2.1. For instance, in order to ease the computation time, a maximum timestep of 50 might be used instead of 300. In addition, the speed factor will tune the speed of reaction as in a real setup where the components do not react at the same pace for every batch.

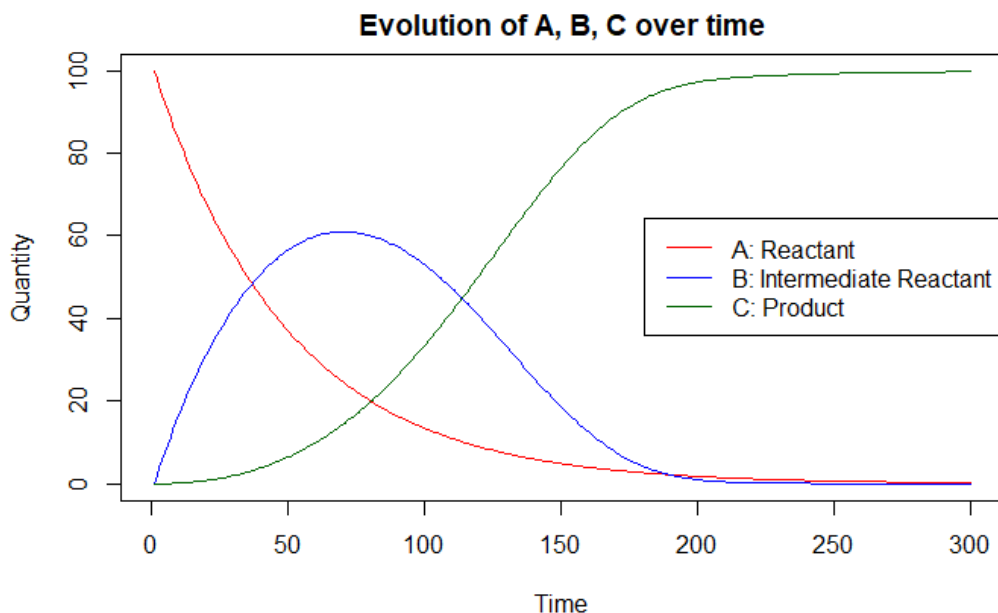


Figure 2.1: Example of evolution of the true quantities over time

2.1.2 True profile

Turning now to the spectra, the true ones need to be defined. They will be used to generate the observed spectra (after adding noise and variance). It simply reduces to randomly generating three sequences of scalars: one will be the location of each peak, the second will be the variability around those peaks, the last will be a factor multiplying the density obtained with the two firsts parameters. The process is repeated for each of the 3 chemical components. A normal density is generated based on the location of each peak and its associated variability. The i -th peak of a component is generated as:

$$\text{peak}_i = \frac{\text{intens}_i}{(\sigma_i)\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_i)^2}{2(\sigma_i)^2}\right) \quad (2.3)$$

This gives the true spectra as can be seen in figure 2.2. Note that the true profile of each component is normalized such that the area under the curve is 1.

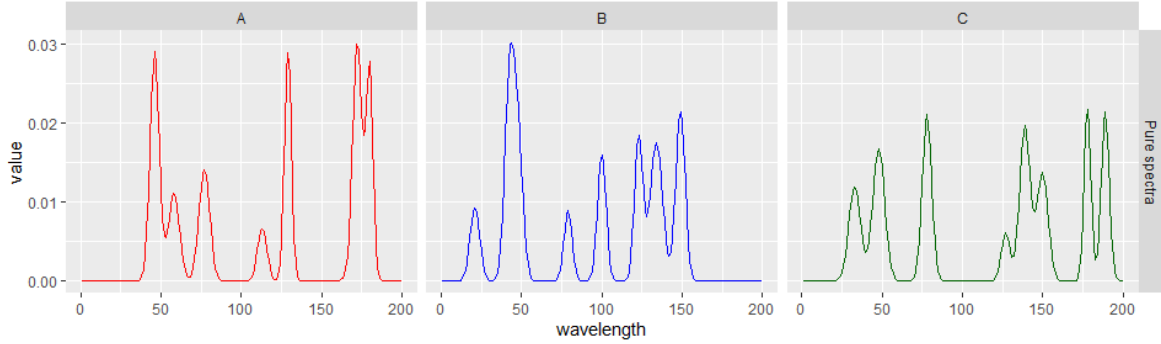


Figure 2.2: True spectra

2.1.3 Observed spectra

To generate the observed spectra, two steps are required.

1. First, for each component, an approximate profile is generated based on its true parameters (location, variability, intensity of the peaks) and the parameters of noise. Thus, the i -th peak of a component is generated as:

$$peak_i = \frac{intens_i}{(\sigma_i + e_\sigma)\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_i + e_\mu)^2}{2(\sigma_i + e_\sigma)^2}\right) \quad (2.4)$$

There are 3 types of noise that can be tuned:

- noise on the location of the peak:
 $e_\mu \sim \mathcal{N}(noise_peak_mean, noise_peak_sd^2)$
- noise on the variability of the peaks:
 $e_\sigma \sim \max(0, \mathcal{N}(noise_var_mean, noise_var_sd^2))$
- noise everywhere on the spectrum, after generating all the peaks and summing them, a lognormal noise is added to the whole profile.
This noise $\sim \ln\mathcal{N}(noise_lognorm_mean, noise_lognorm_sd^2)$

Following this rationale, the approximate spectrum for every component in the chemical reaction is generated.

2. Second, the spectra for each of the 3 components at timestep t can be combined to give the observed spectra. One simply multiplies the approximate spectrum of each component by its respective true quantity at timestep t . Then, the spectrum is normalized such that its area under the curve is 1.

The observed spectra for batch 1 for the case study can be seen in figure 2.3. One can observe the smooth evolution of spectra in time (from timestep 1 to 50). The lognormal noise added can also be noticed in the leftmost part of the spectral window.

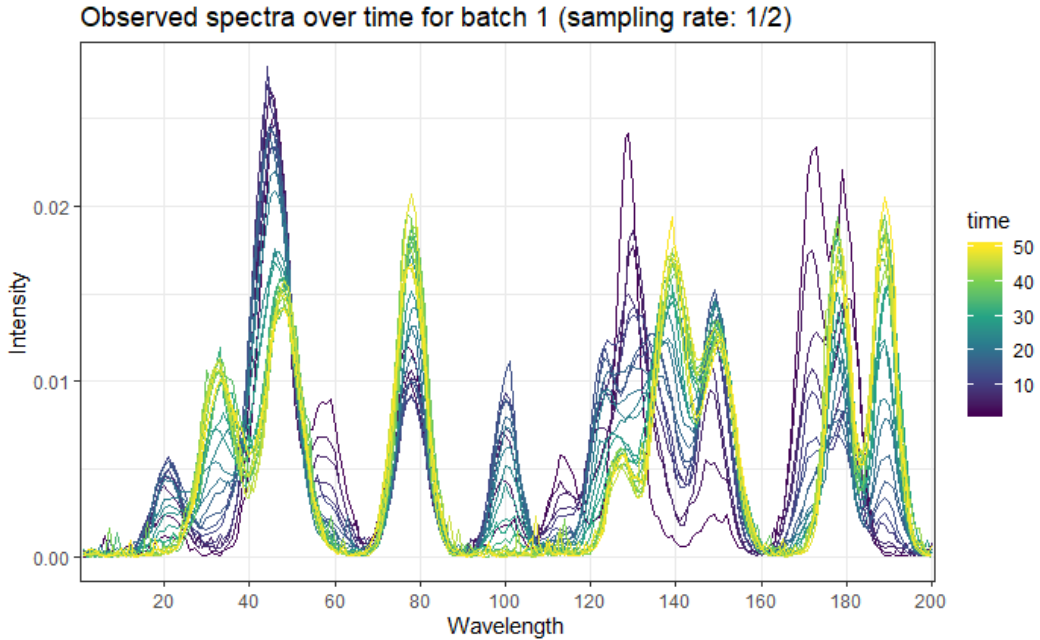


Figure 2.3: Observed spectra for batch 1, where 1 spectra is shown each 2 timesteps

2.1.4 Batch generation

Having designed functions to automate the creation of the spectra based on the parameters described above, let us now turn to the generation of the batches which will be used in this case study.

The default parameters are the following: there are 50 timesteps (so 50 spectra), 100 units of A and none for B and C at timestep 1, 8 peaks per chemical component, the speed factor is 0.9, the noise on the location of the peaks $e_\mu \sim \mathcal{N}(0, 0.4^2)$, the noise on the variability of the peaks $e_\sigma \sim \mathcal{N}(0.3, 0.2^2)$ (the mean is not 0 because the aim is to add variability most of the time), and the background noise is the lognormal $\sim \ln\mathcal{N}(0, 1)$. Those parameters were chosen because they resulted in reasonable observed spectra.

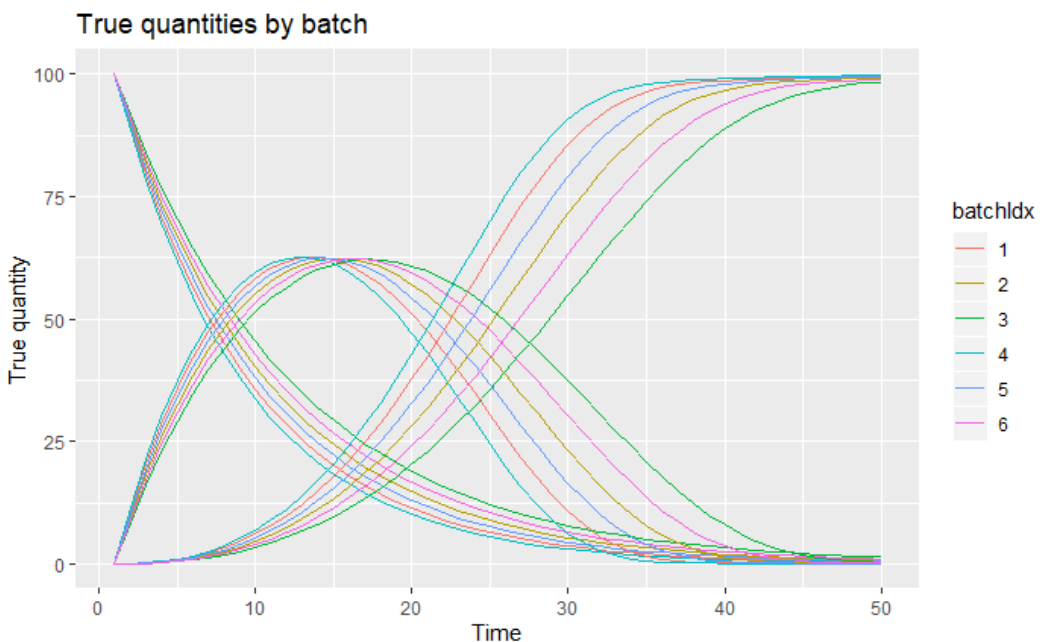


Figure 2.4: True quantities for all batches

Now, 6 batches are generated. The first batch has a speed factor of 0.9, the second 0.8, the third 0.7. For the subsequent batches, the parameters are modified, $e_\mu \sim \mathcal{N}(0, 0.6^2)$ and $e_\sigma \sim \mathcal{N}(0.2, 0.2^2)$, so there are more variability on the position of

the peaks, but the variability around those peaks is less noisy since the mean of e_σ is slightly lower. The fourth, fifth and sixth batches are based on these new parameters and their respective speed factor is 0.95, 0.85, 0.75.

For the batch 1, the spectra are represented in the figure 2.3. The true quantities for each batch are also shown in figure 2.4. The first objective of visualization and interpretation can be checked for this dataset as both the true quantities and pure spectra for the chemical components in the batches are known. The second objective of accuracy of the prediction will also be checked and compared.

2.2 Janssen dataset

For confidentiality reasons, this section is only available under request. Please contact Bernadette Govaerts: bernadette.govaerts@uclouvain.be for more information.

2.3 Chemium dataset

The last dataset comes from Chemium, in the context of the development of a process and a reaction producing a continuous flow of chemical product. Two reactants (A and B) are continuously introduced and they produce a product (C) and side products (D and E). The stoichiometric equations are shown below.



At the reactor exit, the effluent continuously goes through a flow cell and a NIR (Near-Infrared) sensor generates a spectra every minute. Then, this effluent fills a container and after the generation of the 6 spectra, a sample of what is in the container is analyzed off-line by Gas Chromatography (GC) to quantify the mass concentration of A, C, D (a reactant, the target product and a side product, respectively). It is important to note that when the effluent exits the reactor, there is no more reaction possible and the concentration of each component remains the same. Thus, each off-line measurement is associated to each set of 6 spectra.

Chemium provided a series of excel sheets, each one containing one spectrum (the wavenumber from 4003 to 9997 cm^{-1} and its associated intensities), the title of the document containing the date of sampling, the concentrations as measured off-line as well as the density (g/kg). The samples are collected over 3 days.

Thus, after some data management, a spectral matrix X_G containing the 174 (29×6) spectra with 778 wavelengths is obtained. “Batches” will denote the set of 6 spectra coming from the same sample. Hence, there are 29 batches of spectra stacked vertically in X_G , and X_i has dimensions 6×778 .

Regarding the Y_G matrix, 3 compounds are measured by GC and a true value is associated to each spectra. Then, Y_G has dimensions 174×3 with individual Y_i of size 6×3 stacked vertically. Indeed, the 6 spectra for each batch all refer to the same off-line measurement, the sample is isolated from the reaction (as it goes to the flow cell) and 6 NIR spectra are generated from it (waiting a small amount of time between each NIR sample).

For this dataset, the true spectrum of each chemical component is not known and the quantity of the chemical components in each batch is constant. Thus, the first objective of visualization and interpretation is not easily verifiable. On the other hand for the second objective, the accuracy of the prediction may still be checked using the true quantities as the target variables.

Chapter 3

Unsupervised methods

In this section, two unsupervised methods which can be applied to spectral datasets are presented: the Principal Component Analysis (PCA) and the Non-negative Matrix Factorization (NMF). The second one is presented in more details as it will serve as a basis to construct a new NMF algorithm with an extra penalty term.

Then, the section 3.3 presents different ways of fitting and evaluating both dimensionality reduction techniques. One can fit a model and evaluate the reconstruction error on a single batch of the dataset (the individual approach), or fit a single model on all batches (the global approach), or finally fit a model on all but one batch and evaluate the reconstruction error on the left-out batch (the leave-one-batch-out approach). This methodology is the one presented and used in Thiel et al. (2021).

3.1 Principal Component Analysis (PCA)

3.1.1 Description

The Principal Component Analysis is one of the most commonly used and well known linear unsupervised dimension reduction technique. In particular, this method is very convenient for chemometrics applications, where the data matrix comprises quantitative correlated variables with potentially more variables than observations (high dimensional dataset). The PCA finds a new coordinate system where the axes (called the latent variables or the principal axes) are orthogonal to each other (i.e. they are no longer correlated) and maximize the variance of the projected data in the subspace constructed by the latent variables.

The PCA, by drastically reducing the amount of variables, allows to visualize the projected observations and variables, identify existing groups or patterns, or reduce the amount of data needed to represent the information. It is also possible to use the low dimension result of a PCA as input for other methods.

3.1.2 Methodology

Let us first introduce the notation used (the usual chemometrics one).

X	$n \times p$	Data matrix (mean centered on the columns, possibly scaled)
T	$n \times R$	Score matrix of X
t_r	$n \times 1$	Score vector r of X (r : 1,..., R)
P	$p \times R$	Loading matrix of X
p_r	$p \times 1$	Loading vector r of X (r : 1,..., R)
E	$n \times p$	Error matrix
λ_r	scalar	r^{th} eigenvalue of the matrix $X^T X$
Λ	$R \times R$	diagonal matrix with R highest eigenvalues of $X^T X$ in decreasing order
R	scalar	Number of principal components

Before applying PCA, the dataset should be column centered. As PCA is sensitive to the scale of variables, the dataset might also be scaled. The PCA decomposes the X centered matrix into scores and loadings, such that:

$$X = TP^T + E \quad (3.1)$$

where the score matrix is computed as:

$$T = XP \quad (3.2)$$

The first principal component is in the direction which maximize the variance of the projected observations. Projecting the observation using the principal components yields the scores. Thus, $t_1 = Xp_1$ such that the variance of the scores $t_1^T t_1$ is maximized. Then, the second principal component should be orthogonal to the first one and maximize the variance of the scores $t_2^T t_2$. The principal components are uncorrelated.

Note that the loadings are orthogonal to each others AND normalized, while the scores are only orthogonal. Hence, $P^T P = I_R$ and $T^T T$ is diagonal.

Finally, the eigenvalues have a particular interpretation. Indeed, λ_r is the r^{th} eigenvalue of $X^T X$ and is equal to the norm of t_r : $\|t_r\| = t_r^T t_r$. The variance of the projected observations on the r^{th} principal axes is also equal to λ_r . The trace of the matrix $X^T X$ is equal to the sum of the eigen values and is called the 'total inertia': $\lambda_1 + \dots + \lambda_R = I$. Then, the percentage of total variance explained by the r^{th} axis is: $\lambda_r / I \times 100\%$ (Segers, 2019).

3.1.3 Algorithms

Turning now to the algorithms to perform PCA, several exist: the covariance method, the singular value decomposition, NIPALS. In this section, the Singular Value Decomposition and the NIPALS iterative algorithm are briefly detailed.

Singular Value Decomposition (SVD)

The Singular Value Decomposition is the preferred method for numerical accuracy. It can be efficiently computed. It is also the method used by the function *prcomp* in the *stats* package in R, consequently it is the first presented. The SVD for any matrix X of size $n \times p$ is:

$$X = T_0 S P^T \quad (3.3)$$

where T_0 ($n \times n$) and P ($p \times p$) contains the singular vectors and S ($n \times p$) is diagonal and contains the singular values. Regarding the interpretation, T_0 contains the **orthogonal** and **normalized** scores such that $T_0^T T_0 = I_p$. Then, the diagonal matrix S contains the singular values. Thereafter, the **orthogonal** and **normalized** loadings are in the matrix P such that $P^T P = I_p$. The not-normalized scores are obtained as: $T = T_0 S$.

Non-linear iterative partial least-squares algorithm (NIPALS)

Let us now turn to the NIPALS algorithm. This algorithm is very similar to others presented afterwards. In this scheme, the X matrix is iteratively deflated/peeled and the principal components are computed on the residual matrix. In the Partial Least Squares section, the same concept will be applied. The NIPALS is shown below (Govaerts and Martin, 2018).

(0)	X	Start with the initial mean-centered data matrix X
(1)	$u = x_j$	Initialize the candidate score vector using any column of X
(2a)	$b = X^T u$	Compute the approximate loading vector
(2b)	$b = b / \ b\ $	Normalize the approximate loading vector
(2c)	$u^* = X b$	Calculate an improved score vector
(2d)	$\Delta u = u^* - u$	check the convergence of u
(2e)	$u = u^*$	If $\Delta u > threshold$, go to step (2a) and repeat
(3a)	$t_r = u^*$	Define the r^{th} score vector
(3b)	$p_r = b$	Define the r^{th} loading vector
(4a)	$X_{res} = X - u b^T$	Deflate the X matrix
(4b)	$X = X_{res}$	Replace X by X_{res} and go to step (1) to derive the next component

3.2 Non-negative Matrix Factorization (NMF)

3.2.1 Aims

Non-negative Matrix Factorization (sometimes also called non-negative matrix approximation) is a linear dimensionality reduction technique allowing to extract meaningful features from a list of p -dimensional non-negative data vectors. It was first introduced by Paatero and Tapper in 1994 (under the name of “positive matrix factorization”) and became increasingly popular after the publication of the article of Lee and Seung in 1999. Although they were the first to introduce the NMF, Paatero and Tapper are rarely cited compared to the work of Lee and Seung. According to Berry et al. in 2007 this is partly due to the name (“positive matrix factorization”) Paatero and Tapper used whereas their algorithm produces a NMF.

The NMF proceeds by factorizing a non-negative matrix X into two non-negative matrix (usually denoted W and H or U and V) which matrix product approximates the initial matrix X . X is hence said to be approximated by a low-rank matrix UV .

$$X = UV + E \quad (3.4)$$

With the initial matrix $X \in \mathbb{R}_+^{n \times p}$ with n observations and p variables, it results in $U \in \mathbb{R}_+^{n \times r}$ (the scores), $V \in \mathbb{R}_+^{r \times p}$ (the loadings) and the error matrix $E \in \mathbb{R}^{n \times p}$. From the equation 3.4, it means for a particular data-point that $X_i \approx U_i V$ where X_i is a particular data point and U_i is the corresponding row in U . That is to say, each data point is approximated by a linear combination of the rows of V weighted by the components of U_i (Lee and Seung, 2001). Note that in the literature, the data points are often stacked vertically (as columns) in X while in this study they are stacked horizontally (as rows) in X . Consequently, the interpretation of the factorization is modified: if X was $p \times n$, then U would be $p \times r$: the loadings and not the scores.

The parameter r ($r \leq \min(n, p)$) is the number of basis elements and vectors. The matrix V thus contains r basis vectors and the coordinates of each observations in the basis V are given by the rows of U (Gillis, 2014).

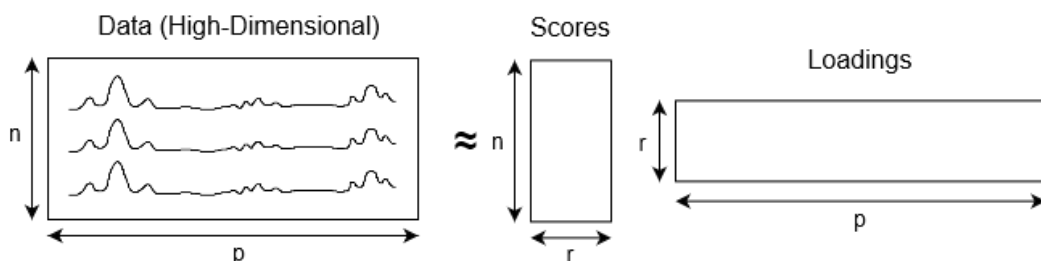


Figure 3.1: NMF schema for high dimensional data (e.g. MIR spectrum)

A good approximation is only found if the basis vectors reflect a structure, a potential relationship between the observations and the variables in the high-dimensional data. These basis vectors will be further denoted as “the loadings” while the basis elements will be called the “the scores”. The learning is by definition unsupervised. Lots of applications exist for the NMF, the next section presents some of the popular ones.

3.2.2 Applications

The applications of NMF are numerous and the interpretability of the results obtained makes it a very persuasive method.

Image processing

One of the most popular example of the NMF in practice is the image decomposition. For the data input, a set of images is given. Each image is nothing but a vector with pixel intensities, where every pixel intensity should be non-negative. The NMF will extract a number of features from the data input images (for instance: having glasses, a beard, big eyes, etc) which will be stored in the loadings V and for each input image, a list of scores (the rows of U) will determine the importance of each feature for the particular input image. The approximation of the input image is the linear combination of the features in the loadings (also known as the basis images) and the scores of the image, as shown in figure 3.2. The square of 7×7 pixels represents the scores for the particular observation i with $U \in \mathbb{R}_+^{2429 \times 49}$. The square representing the features is actually another view of the loadings $V \in \mathbb{R}_+^{49 \times 19^2}$ where each small square representing a specific feature is a row from V .

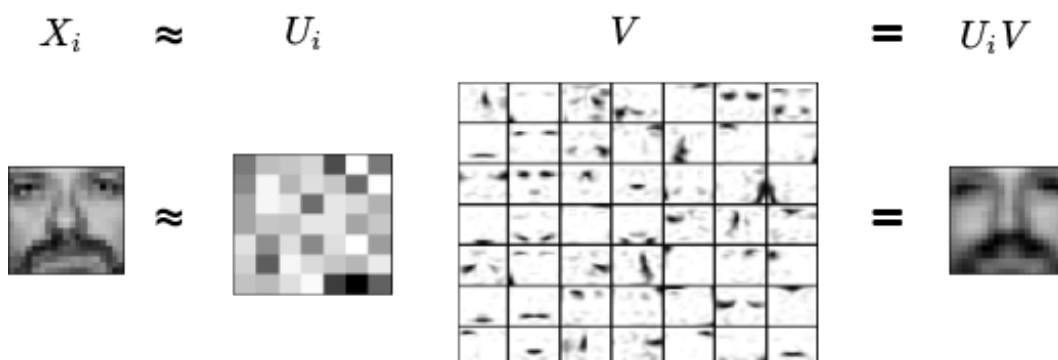
$$X_i \approx U_i V = U_i V$$


Figure 3.2: Decomposition of faces, MIT Center For Biological and Computation Learning (2429 gray-level 19-by-19 pixels images) with $r = 49$ (Lee and Seung, 1999)

According to Gillis in 2014, NMF seems promising in the context of face recognition. For instance, it has been shown more robust than PCA with occluded faces (e.g. wearing sunglasses).

Text Mining

A typical task in text mining can be the identification of the topic of documents. One can use the term-frequency (TF) matrix, the matrix where each row is a document, each column is a word, and each entry is the number of occurrences of a word in a document. Of course, using the raw counts of words is often not optimal and other weighting techniques exist (for instance, term frequency-inverse document frequency (TF-IDF) gives more weight to the words occurring less frequently).

The NMF on the TF matrix will discover the different r topics among the documents. A document X_i will be approximated by its r scores and the r basis topics. Each basis topic is of size p , the number of words in the TF matrix. Thus, $X_i \approx U_i V$ and the r rows of V represents the r topics identified. Each basis topic can be interpreted as a “base document”. For instance when approximating a dataset of books, one base topic could represent a crime, a science-fiction or a romance novel by having a large value for the words “guns”, “wizard” or “love”, respectively. By combining the different base topics with a positive score, one can reconstruct a document. Looking at those scores for a particular document allows to classify it among the different topics.

As a conclusion, Gillis states that “given a set of documents, NMF identifies topics and simultaneously classifies the documents among these different topics” (2014, p.4).

Recommendation Systems

NMF can also be used to recommend items based on which other users have bought/-liked. Such a case can be illustrated with a movie dataset in which rows are users and

columns are movies, a cell is 0 if the user did not watch the movie and 1 otherwise (alternatively, the user can give a rating to seen movies from 1 to 5 and an unseen movie is marked with a 0). The NMF decomposition (using a tuned r parameter) will allow to suggest movies to users/predict the score of unseen movies based on the reconstruction. The same can be applied to a dataset of items bought by customers in a supermarket. Again, not only allowing suggestion of items to the customers but also dividing the customers into several profiles (r profiles contained in V).

Spectroscopic data

Finally, in the context chemical reactions monitoring, NMF provides an effective way of decomposing spectral data. As shown by Thiel et al. (2021), the non-negativity constraint enhances the interpretability of the results compared to another classical decomposition: PCA. Indeed, the spectral data typically consists of intensities measured at a set of wavelengths, the intensities being non-negative. Then, the scores and loadings produced will also be non-negative. The loadings will reflect the r basis elements found in the spectra dataset. If the r is well chosen, the basis elements can be interpreted as the chemical components in the reaction. The loadings are then expected to be close to the true spectral profiles of the components while the scores are expected to give an idea of the proportion of each basis elements in the reaction.

Thiel et al. show among others, that the scores from NMF can indeed be used to visualize the evolution of the chemical components over time during the reaction. They also show that a PLS model can be fitted to accurately predict the concentrations in the reaction (measured by ultra-pressure liquid chromatography, UPLC). A regression model can also be fitted based on the scores of a NMF decomposition. Thus, they built and compared calibration models to predict the concentrations of the components in the reaction over time without having to use "off-line" techniques such as UPLC, where each sample needs to be processed and takes time, not allowing for a continuous monitoring of the reaction (Thiel et al., 2021).

3.2.3 Similarity measures

Regarding the formulation of the NMF problem, the difference between the reconstruction UV and the original matrix X has to be measured using a similarity measure. This similarity measure is also used as a cost function (or objective function) for the optimization problem. The most widely chosen are the Frobenius norm (Squared Euclidean Distance or SED)(3.5) and the Generalized Kullback–Leibler divergence (GKLD) (3.6) according to Wang and Zhang (2013). Gillis (2014) however states that the Frobenius norm is the most widely chosen in the literature.

$$D_F(X||UV) = \frac{1}{2}\|X - UV\|_F^2 = \frac{1}{2} \sum_{ij} (X_{ij} - [UV]_{ij})^2 \quad (3.5)$$

$$D_{KL}(X||UV) = \sum_{ij} \left(X_{ij} \ln \frac{X_{ij}}{[UV]_{ij}} - X_{ij} + [UV]_{ij} \right) \quad (3.6)$$

For both similarity measures, the lower bound is 0 and the similarity vanishes if and only if $X = UV$. On the other hand, the GKLD is not a distance but rather a divergence since it is not symmetric: $D_F(X||UV) \neq D_F(UV||X)$. The GKLD also exhibits some drawbacks. In particular, the gradients used to obtain the scores and the loadings depend on the scales of the matrices. Hence, the convergence speed will be very low as many iterations are required (Wang and Zhang, 2013). Scaling the data is thus needed.

To conclude this section, two possible problems can be considered for the NMF:

- **Problem 1.** Minimize $\|X - UV\|_F^2$ with respect to U and V , with constraints $U, V \geq 0$.

- **Problem 2.** Minimize $D_{KL}(X||UV)$ with respect to U and V , with constraints $U, V \geq 0$.

3.2.4 Algorithms

Several existing algorithms to perform NMF are presented below. It is worth noting that no algorithm is expected to find a global minimum since the objective function is convex in either U or V but not in both together (Lee and Seung, 2001), but the local minimums are in general satisfactory. In this section, we will mainly focus on the Frobenius norm objective.

Gradient descent

The gradient descent is an easy to implement algorithm. A gradient can be seen as the derivative for multi-dimensional function. It gives the direction of the largest increase or the steepest slope. Computing the gradient of a function allows to reach local extrema by iteratively moving towards a local maximum or minimum using the gradient multiplied by a small step.

In the *Algorithm 1*, γ_U and γ_V are the learning rates, the step sizes for the gradient descent. The gradient of the function F with respect to U is noted ∇F_U where F is the objective function to minimize (thus the subtraction at each iteration). In this algorithm, the iterations go until the maximum number of iterations (to be determined) is reached or the convergence is reached (the criterion needs also to be determined). Note that nothing prevents us from obtaining negative elements during a classic gradient descent, thus an additional step is required to project the negative elements back into the non-negative space.

Algorithm 1: Gradient Descent Algorithm

Input: X, r

Output: U, V

Initialization: U, V as non-negative matrix of size $n \times r$ and $r \times p$

while stopping criterion not satisfied **do**

$U \leftarrow \max(0, U - \gamma_U \nabla F_U);$

$V \leftarrow \max(0, V - \gamma_V \nabla F_V);$

end

For the Frobenius norm similarity (note the $\frac{1}{2}$ to cancel out the factor 2 in the gradient), the gradients of $\frac{1}{2}\|X - UV\|_F^2$ are

$$\begin{aligned}\nabla_U \frac{1}{2}\|X - UV\|_F^2 &= -XV^T + UVV^T \\ \nabla_V \frac{1}{2}\|X - UV\|_F^2 &= -U^T X + U^T UV\end{aligned}\tag{3.7}$$

For the Generalized Kullback–Leibler divergence, the gradients of $D_{KL}(X||UV)$ are:

$$\begin{aligned}\nabla_U D_{KL}(X||UV) &= -\left(\frac{[X]}{[UV]} - \mathbf{1}_{n \times p}\right) V^T \\ \nabla_V D_{KL}(X||UV) &= -U^T \left(\frac{[X]}{[UV]} - \mathbf{1}_{n \times p}\right)\end{aligned}\tag{3.8}$$

The choice for the learning rates at each iteration is fundamental. If the steps are too small, the convergence may never occur while the extremum may be missed if the steps are too large. There exists two cases: having a static or a dynamic step size. In a dynamic fashion, the step size will decrease when the algorithm iterates and thus is expected to reach the extremum more rapidly. Berry et al. in 2007 explain for instance that some algorithms start with a learning rate of 1 and divide it by 2 at each iteration. The convergence towards a local minimum is not guaranteed, in particular the projection step in the non-negative space makes it even less likely. In practice, these algorithms produce poor results.

Multiplicative update rules

The multiplicative update rules were introduced by Lee and Seung in 2001. Its ease of implementation contributed to make the NMF more popular. These update rules also have the advantage to ensure that the non-negativity constraints are respected if the U, V initial matrix are also non-negative. In fact, these rules are nothing but a gradient descent with an adaptive learning rate. The multiplicative update rules are presented below for the Frobenius norm (3.9). It has been proved that the Frobenius norm $\|X - UV\|_F$ is non-increasing using these rules.

$$U \leftarrow U \circ \frac{[XV^T]}{[UVV^T]}, \quad V \leftarrow V \circ \frac{[U^T X]}{[U^T UV]} \quad (3.9)$$

The gradient descent expression being

$$U \leftarrow U - L_U \circ (-XV^T + UVV^T), \quad V \leftarrow V - L_V \circ (-U^T X + U^T UV) \quad (3.10)$$

Replacing in the expression 3.10 above $L_U = \frac{[U]}{[UVV^T]}$ and $L_V = \frac{[V]}{[U^T UV]}$ yields the multiplicative update as shown in 3.9. The details are provided in the Appendix.

Algorithm 2: Multiplicative Update Algorithm

Input: X, r

Output: U, V

Initialization: U, V as non-negative matrix of size $n \times r$ and $r \times p$

while stopping criterion not satisfied **do**

$U \leftarrow U \circ \frac{[XV^T]}{[UVV^T]};$
 $V \leftarrow V \circ \frac{[U^T X]}{[U^T UV]};$

end

The algorithm offers no guarantee to reach a local minimum although Lee and Seung initially stated the opposite, but it is usually the case. Indeed, nothing prevents reaching a saddle point (Berry et al., 2007). Although this algorithm is often used as a baseline against new algorithms, it converges slowly and the complexity for each iteration is in $O(npr)$ (Berry et al., 2007). One also realizes that once elements are set to 0, they will never be able to be updated anymore. The algorithm in the next section deals with this drawback.

Alternating Least Squares

In the family of Alternating Least Squares (ALS), least squares steps will be iteratively performed to solve the problem for U and then for V , the theoretical justification being that the NMF problem is convex in U and V but not in both together. The algorithm was first used by Paatero and Tapper (1994) and since then several modifications have been proposed. The basic ALS algorithm is presented below (3).

Algorithm 3: Alternating Least Squares Algorithm

Input: X, r

Output: U, V

Initialization: U as non-negative dense matrix of size $n \times r$

while stopping criterion not satisfied **do**

Solve for V in equation $U^T UV = U^T X$ which is $V = (U^T U)^{-1} U^T X$;
Projection step: set all negative elements in V to 0 ;
Solve for U in equation $V V^T U^T = V X^T$ which is $U^T = (V V^T)^{-1} V X^T$;
Projection step: set all negative elements in U to 0 ;

end

This algorithm has the advantage that a value in U or V set to 0 can escape this situation and explore other solutions while it was not the case for the Multiplicative

Update algorithm (Albright et al. described it as the “locking phenomenon”). In addition, ALS is faster than the previous algorithms. In contrast, the solution produced lacks sparsity and the projection steps into the non-negative space again is incompatible with a proof for convergence. Algorithms based on ALS have thus been created incorporating a **sparsity** or a **nonnegativity constraint**.

Regarding the **nonnegativity constraint**, the Alternating Nonnegative Least Squares (ANLS) (see *algorithm 4*) is the ALS solving each time a nonnegative least squares (NNLS) problem. NNLS problems are very common in optimization and can be efficiently solved (an algorithm is given by Lawson and Hanson (1995)). The ANLS provably converges towards a local minimum. However, solving the NNLS problems is expensive and very slow (Albright et al. (2014), Gillis (2014)), to the point that in practice, the simpler ALS with its unconstrained least squares problems will often be preferred. There is a compromise to make between the convergence theory and the speed.

Algorithm 4: Alternating Nonnegative Least Squares Algorithm

Input: X, r

Output: U, V

Initialization: U as non-negative dense matrix of size $n \times r$

while stopping criterion not satisfied **do**

$U \leftarrow \arg \min_{U \geq 0} \|X - UV\|_F;$

$V \leftarrow \arg \min_{V \geq 0} \|X - UV\|_F;$

end

A more complex algorithm was proposed to circumvent the issue of the slowness of the NNLS problems: the Hierarchical Alternating Least Squares (HALS) (Cichocki and Phan, 2009). This algorithm, unlike the ALS, ensures that a stationary point will be reached. It is also one of the most efficient in practice. The drawback is that it is sensitive to scaling, unlike the MU rules or the ANLS (Gillis and Glineur, 2011).

Turning now to the **sparsity**, Albright et al. in 2014 introduced an additional constraint in their Alternating Constrained Least Squares (ACLS) algorithm. The objective becomes

$$\min_{h_j} \|x_j - u_j V\|_F^2 + \lambda_U \|u_j\|_F^2 \quad s.t. \quad \lambda_H \geq 0, u_j \geq 0 \quad (3.11)$$

where x_j and u_j are rows of X and U . Now, keeping the nonnegative projection steps, the least squares steps are given by

- Solve for V in equation $(U^T U + \lambda_V I)V = U^T X$
- Solve for U in equation $(V V^T + \lambda_U I)U^T = V X^T$

Thus the ACLS alternatively performs a Ridge regression to enhance sparsity. We note that using a L1 norm penalty term is more adequate to achieve sparsity in the sense of obtaining features exactly equal to 0. Nevertheless, a L2 penalty will also shrink the estimates toward 0 when λ increases, but it is less likely that they will be **exactly** 0. Ridge is also known to be able to estimate coefficients for high dimensional datasets (more features than observations) or when the variables are correlated, while OLS (Ordinary Least Squares) cannot. The user has to decide by trial and error which value of the regularization parameters λ_V and λ_U to use. Albright et al. show that the speed of their algorithm is very high, but there is still no guarantee to reach a local minimum.

3.2.5 Variety of NMF algorithms

A wide class of NMF algorithms exists in the literature, as demonstrated in the work of Wang and Zhang (2013). They distinguish 4 families:

1. The Basic NMF, where the only constraint is the non-negativity one.
2. The Constrained NMF, where regularization terms are added. It comprises the Sparse NMF, the Orthogonal NMF, the Discriminant NMF (adding a classification term) and the NMF on manifold (adding a penalty term to preserve some properties on the distance between neighbors in high dimension and in low dimension).
3. The Structured NMF, where the standard factorization formulation is modified. For instance, weighting some elements from the data input (Weighted NMF), taking the time into account (Convolutional NMF), or producing a factorization of 3 factors instead of 2 (Nonnegative Matrix Trifactorization).
4. The Generalized NMF, where the data type or the factorization is unconventional. For instance, the data input is in higher dimension (a tensor), or the non-negativity constraint is relaxed on one of the factor of the NMF, or even derive a non-linear version of the NMF using Kernel.

In the literature, the usual and convenient way to derive a Supervised NMF is to add a penalty term in the objective function. Possibly, several other penalty terms are added as well. The Constrained NMF class is thus further detailed. The Constrained NMF can be expressed as:

$$D_C(X||UV) = D(X||UV) + \alpha J_1(U) + \beta J_2(V) \quad (3.12)$$

where the modified objective is the objective of the Basic NMF, $D(X||UV)$ (the Kullback Leibler or the Frobenius divergence) to which penalty terms are added balanced by small factors α and β .

The **Sparse NMF** will typically add a L1 penalty term on U or V , the choice depending on the application. Note that L2 norm penalty term are also used to enhance sparsity in some cases, but this might lead to low instead of sparse values Wang and Zhang (2013).

Then, an **Orthogonal** penalty on either U or V is a way to minimize the redundancy. Since there is a non-negativity constraint, the orthogonality will also lead to sparseness Wang and Zhang (2013). The orthogonality constraint can be put on the scores: $U^T U = I$, enhancing the clustering accuracy, or on the loadings: $V V^T = I$, leading to the most distinct basis being found. It is particularly useful for clustering tasks.

The **Discriminant NMF** turns the Basic unsupervised NMF problem into a supervised one. It has been successfully used for classification tasks. There exists several ways to do it (but it is still a quite recent field): adding a Fisher discriminant constraint (the difference between the within and the between class scatter) or a maximum margin classifier (especially a Support Vector Machine). Using the Projected Gradient Descent algorithm (Lin, 2007) ensures the convergence to a stationary point, otherwise, the Discriminant NMF algorithm offers no guarantee to converge toward a local optimum.

Finally, **Weighted NMF** emphasizes the importance of some data points by means of a weight matrix: $W \odot X = W \odot (UV) + E$. It is typically used when the dataset comprises missing or unobserved values. It has a successfully application in collaborative filtering. It can also be used in combination with other penalties.

3.2.6 Complexity

Most algorithms for NMF run in $O(npr)$ (Gillis, 2014) as the Multiplicative Updates of Lee and Seung. The same applies for the ACLS and the HALS methods, but it has been shown that they are faster than the MU rules in practice. Note that the order of operations in the MU matters to speed up the process, see equation 3.13. $V V^T$ and $U^T U$ will both result in small $r \times r$ matrices, hence they must be calculated first in the denominators.

$$U \leftarrow U \circ \frac{[XV^T]}{[U(VV^T)]}, \quad V \leftarrow V \circ \frac{[U^T X]}{[(U^T U)V]} \quad (3.13)$$

3.2.7 Initialization

In every algorithm, U and V or at least one of them (for the ALS class) has to be initialized. A random initialization from a uniform distribution is always possible, but other techniques exist and are expected to converge faster and/or towards a better factorization. Nevertheless, Gillis in 2014 reminds that there usually exists no proof that it will indeed be the case. He presents 3 families of possible strategies below.

- *Singular Value Decomposition.* Every matrix $X \in \mathbb{R}^{n \times p}$ of rank $k \leq \min(n, p)$ can be expressed as $\sum_{j=1}^k s_j t_j p_j^T$ with $s_1 \geq \dots \geq s_k > 0$ being the singular values and t_j and p_j the singular vectors. In matrix form, this is equivalent to $X = TSP^T$ with T ($n \times k$), S (diagonal matrix $k \times k$), P ($p \times k$). The optimal rank- r approximation of X is given by the r first triplets: $X^{(r)} = \sum_{j=1}^r s_j t_j p_j^T$.
- *Clustering techniques.* Use the centroids produced by a clustering e.g. use k-means to initialize the rows of V , each of the r rows being p -dimensional.
- *Column subset selection.* Initialize the rows of V using some set of observations from X .

In particular, Boutsidis and Gallopoulos in 2008 proposed the Nonnegative Double Singular Value Decomposition (NNSVD). It starts from a Singular Value Decomposition (SVD) and extracts the r leading triplets of X , allowing to form the matrices $\{u_j v_j^T\}_{j=1}^r$ of singular vectors. From the latter, negative elements could be present, hence it extracts the positive section ($[x]_+ = \max(x, 0)$) and computes their singular triplet information to be used as initialisation for U and V .

3.2.8 Stopping criterion

Once the choice of algorithm and initialisation is made, the stopping criterion remains to be defined. One can monitor the evolution of the objective function and set a lower bound on the difference between consecutive iterations. The norm of the projected gradient can also be measured (Lin, 2007). In addition, a time or a maximum number of iterations can be used.

3.2.9 Choice of r

There exists no clear rule for the choice of the number of components. Trial an error can be used, or looking at singular values of the SVD on the input data (if using the Frobenius norm for the objective function) and choose the component when the decay is slower Gillis (2014) but deciding on a clear threshold might be difficult. Alternatively, this number might be chosen according to the number of latent variables expected in the input data. For instance, if 3 chemical components are present in a reaction for which a set of spectra are available, one could try $r = 3$. Doing so, V will contain 3 basis vectors. Finally, this number might also be chosen relying on the results of other methods like PCA or ICA.

3.2.10 R packages

NMF package

The NMF package seems to be one of the most used in R (Gaujoux and Seoighe, 2010). It comes with a technical documentation (Gaujoux and Seoighe, 2020a) and vignettes (Gaujoux and Seoighe, 2020b). This package implements several algorithms, 4 seeding methods, is optimized and memory efficient (C++ implementations), and performs parallel computations. Last but not least, it provides a unified framework to create and

test new NMF algorithms (Gaujoux and Seoighe, 2020b).

The seeding methods available are

- **ica** : the result of an Independent Component Analysis (only keeps the positive elements)
- **nnsvd** : Nonnegative Double Singular Value Decomposition (see section 3.2.7 above)
- **none** : allows the user to fix the seed manually
- **random** : initialize with a random uniform distribution $[0, \max(X)]$ (used by default)

The available algorithms are

- **lee** : based on Frobenius norm, the Multiplicative Update (MU) rules as introduced by Lee and Seung (2001)
- **offset** : based on Frobenius norm, modified version of **lee** where an intercept is fitted
- **brunet** : based on Kullback-Leibler divergence, modified version of the MU rules (Lee and Seung, 2001) to avoid numerical overflow (used by default)
- **nsNMF** : Non-smooth NMF, based on Kullback-Leibler divergence, modified version of the MU rules (Lee and Seung, 2001) increasing sparsity
- **pe-NMF** : Pattern-Expression NMF, based on Frobenius norm, adds regularization “for effective expression of patterns with basis vectors” (Gaujoux and Seoighe, 2020b)
- **snmf/r**, **snmf/l** : an Alternating Least Squares algorithm, meant to be much faster than the previous techniques

NNLM package

This package was removed from the CRAN on 2020-02-05 as “check issues were not corrected in time” (Lin, 2020b). However, the source code is still accessible on other repositories as well as the vignettes.

In the manual, the author argues that the **NMF** package is in practice very slow when the number of rows increases to tens of thousands, probably because of its **R** implementation (Lin, 2020a). Lin thus implements in this package an adapted version of the ALS problem using the Alternating Non-Negative Least Square (NNLS) algorithm as proposed by Franc et al. (2005) and **Rcpp** (a **R** package which enables the integration of C++ code in **R**).

The package provides 2 main functions, **nnlm** and **nnmf**. The **nnlm** function solves a non-negative least square (NNLS) problem as below

$$\min_{\beta \geq 0} \|Y - XB\|_F \quad (3.14)$$

While the **nnmf** function performs a NMF but allows the addition of other penalization terms in the objective on U and V (with A being either U or V):

- a Ridge penalty: $\frac{1}{2} \sum_{ij} A_{ij}^2$ to control magnitudes and smoothness (Lin, 2020a)
- a LASSO penalty: $\sum_{ij} |A_{ij}|$ to control sparsity and magnitude
- a penalty to minimize the correlations between columns of U or V : $\sum_{i < j} (X_{.i})^T X_{.j}$

Then the author adapts the Lee and Seung MU rules and a sequential coordinate descent (SCD) from the objective function. So the user might essentially define its own penalization, choose between SCD or MU, and choose a Frobenius norm objective or a Kullback-Leibler one. Clearly, this package offers more flexibility with respect to the penalty terms than the `NMF` package. The speed of each package should be compared but its removal from the CRAN does not encourage its use.

spectralAnalysis package

The package `spectralAnalysis` is available on the CRAN, seems very well maintained and appears highly qualitative (Oirbeek et al., 2020). It implements NMF with the purpose of performing it on spectral dataset. The package includes several preprocessing functions for the spectra (baseline correction, normalization, time alignment, interpolation, etc) and enables the creation of pipelines for preprocessing multiple datasets. Focusing on the NMF, the package actually imports other packages to perform NMF: the packages `NMF`, `hNMF` and `nnls`.

The `spectralNMF` function makes it possible to specify the pure spectra for the initialisation of the algorithm. Many methods for the NMF are available, in fact they call the methods from the `NMF` and `hNMF` packages. The default method is the PGNMF: an alternating non-negative least squares method using projected gradients Lin (2007).

As a conclusion, `spectralAnalysis` allows to perform multiple NMF and conveniently provides a single function to use the methods from other existing packages.

hNMF package

The author of the `hNMF` package, Nicolas Sauwen is also one of the author of the package `spectralAnalysis`. It imports the package `NMF`, but in contrast with the package `spectralAnalysis`, well known NMF algorithms from the scientific literature are actually implemented in it.

The methods available comprise

- the accelerated hierarchical alternating least squares (HALSAcc) from the PhD thesis of Nicolas Gillis in 2011
- the hierarchical non-negative matrix factorization (HALS) (Cichocki and Phan, 2009)
- the alternating non-negative least squares using projected gradients of Lin in 2007
- a semi-NMF method based on the MU (the X matrix is no longer constrained to be nonnegative) (Ding et al., 2010)

Alongside these algorithms, there also exists a function to initialize the NMF models with spectral data as well as one to ensure the conditions for NMF on the input matrix are respected. This package is clearly the most up-to-date with respect to the scientific literature.

nnls package

Finally, the package `nnls` (M. Mullen and van Stokkum, 2020) is available on the CRAN. It is a small package, which only implements a solver for non-negative least squares (NNLS) problems (this is a subproblem of the Alternating Least Squares (ALS) algorithm). The function is based on the algorithm of Lawson and Hanson (1995) for solving NNLS problems. It solves

$$\arg \min_b \|y - Xb\|_F \quad (3.15)$$

where b is non-negative, y is the univariate response and X is a matrix $n \times p$. In other words, this package is not properly implementing a NMF algorithm but an ALS enforcing the nonnegativity constraints is only a sequence of NNLS problems.

3.3 Methods comparison on batch data

This section presents the different approaches possible when fitting models and assessing them on batch data. As stated above, 3 approaches are considered: the individual, the global and the leave-one-batch-out, as in Thiel et al. (2021). For each, the reconstruction error is shown.

In the case studies, the reconstruction error will be compared as well as the interpretability of the scores and loadings produced by each models.

3.3.1 Individual approach

The individual model consists in fitting the dimensionality reduction technique on a particular batch i : X_i ($n_i \times p$). Then, the decomposition is $X_i = U_i V_i + E_i$ where U_i is the scores matrix, V_i is the loadings matrix and E_i the error matrix. Finally, the percentage of error for this batch is given by: $\%error_i = \frac{\|E_i\|_F^2}{\|X_i\|_F^2} \times 100$.

3.3.2 Global approach

The global model to the contrary, will fit the dimensionality reduction technique on the whole dataset of spectra X_G ($n \times p$). Thus, the decomposition is $X_G = U_G V_G + E_G$, and the error for the batch i is extracted from the matrix E_G : E_{Gi} . The percentage of error for the batch i is: $\%error_i = \frac{\|E_{Gi}\|_F^2}{\|X_i\|_F^2} \times 100$.

3.3.3 Leave-one-batch-out approach

Finally, the leave-one-batch-out strategy will fit the dimensionality reduction technique on all but one batch: $X_{G/i}$ ($(n - n_i) \times p$) and for this particular batch: X_i ($n_i \times p$), the error is measured. The decomposition is: $X_{G/i} = U_{G/i} V_{G/i} + E_{G/i}$. To compute the error, we need the scores for the batch i , it thus requires an additional projection step.

By projecting X_i on the space of $X_{G/i}$ components, the estimated scores for the left-out batch $\hat{U}_i$ are obtained, using least squares yields: $\hat{U}_i = X_i V_{G/i}^T (V_{G/i} V_{G/i}^T)^{-1}$ (Thiel et al., 2021). The percentage of error for the batch i is: $\%error_i = \frac{\|X_i - \hat{U}_i V_{G/i}\|_F^2}{\|X_i\|_F^2} \times 100$. This last approach is particularly suitable to assess the expected generalization error when applying a given model to new unseen batches.

3.3.4 Case of centered data

For some techniques as PCA, the dataset should first be centered. Consequently, one needs to add the column means of the data input (X_G , X_i , $X_{G/i}$ for the global, individual and leave-one-batch-out model, respectively) when reconstructing the spectra of the batch considered X_i .

- For the **individual approach**, X_i is column centered using its own column means: $X_i^C = X_i - \mathbf{1}_{n_i \times 1} m_i^T$ with m_i ($p \times 1$) being the means of the column of X_i . The decomposition is: $X_i^C = U_i^C V_i^C + E_i^C$ where the C index indicates the column centering. The error measured is: $\%error_i = \frac{\|E_i^C\|_F^2}{\|X_i\|_F^2} \times 100$.
- For the **global approach**, the X_G matrix is column centered using m_G the column means of X_G : $X_G^C = X_G - \mathbf{1}_{n \times 1} m_G^T$. Then, the dimensionality reduction: $X_G^C = U_G^C V_G^C + E_G^C$. The error measured for each batch is: $\%error_{Gi} = \frac{\|E_{Gi}^C\|_F^2}{\|X_{Gi}\|_F^2} \times 100$.
- For the **leave-one-batch-out approach**, the $X_{G/i}$ matrix is column centered using $m_{G/i}$ such that $X_{G/i}^C = X_{G/i} - \mathbf{1}_{(n-n_i) \times 1} m_{G/i}^T$. Then, the dimensionality

reduction technique is applied: $X_{G/i}^C = U_{G/i}^C V_{G/i}^C + E_{G/i}^C$. Next, the extra projection step is required using X_i^C , the left-out batch centered using $m_{G/i}$:

$$\begin{aligned} X_i^C &= X_i - \mathbf{1}_{n_i \times 1} m_{G/i}^T \\ \hat{U}_i^C &= X_i^C (V_{G/i}^C)^T (V_{G/i}^C (V_{G/i}^C)^T)^{-1} \end{aligned} \quad (3.16)$$

Note that with orthogonal and normalized loadings (such as with PCA), the equation of the projection becomes:

$$\hat{U}_i^C = X_i^C (V_{G/i}^C)^T \quad (3.17)$$

Finally, the error can be computed using the projected scores: $\%error_i = \frac{\|X_i^C - \hat{U}_i^C V_{G/i}^C\|_F^2}{\|X_i^C\|_F^2} \times 100$. To compare the various models on the basis of their errors, this error percentage is computed using X_i and not X_i^C .

Since the centering step is advantageous, the methods using non-centered data with rank n will be compared to methods using centered data with rank $n - 1$.

Chapter 4

Supervised methods and Prediction

In this section, supervised methods which can be used in a chemometrics context are presented.

First, the Partial Least Squares (PLS) is detailed, which produces coefficients which maximize the covariance between X (the data) and Y (one or several targets). To do so, the PLS will decompose the data and the target(s) into interpretable scores and loadings. Thus, the PLS is of interest since it produces scores and loadings (as the previous unsupervised methods) and at the same time strives to predict a target.

Then, supervised extensions of the classical NMF problem are presented. The modification essentially consists in the addition of penalization terms in the objective function. The methodology and the results of such studies are also presented and discussed.

In a third subsection, a supervised penalty term is added in the objective function of the NMF problem. The gradient descent additive update rules and the multiplicative update rules are provided following the rationale of Lee and Seung (2001) in their famous paper introducing the multiplicative update rules for the basic NMF problem.

Finally, the last subsection presents the methodology to compare and assess the performance of this new algorithm. It is easy to predict new data using a fitted PLS model since one obtains regression coefficients, but the NMF and the new Supervised NMF only produce scores and loadings. Thus, extra steps are required to compare the performance of the models in a cross-validated scheme.

4.1 Partial Least Squares (PLS)

4.1.1 Description

The Partial Least Squares (PLS) model was introduced around 1975 by the statistician Herman Wold (Varmuza and Filzmoser, 2009). The application of PLS in chemometrics was then introduced by his son Svante Wold and others. It is a linear method which finds regression coefficients to predict a target matrix or vector Y using a data matrix X . It will proceed by decomposing X (and possibly Y) into linear latent variables. Then, the objective is to maximize the covariance between the scores of X and Y (or the scores of Y). Thus, it is equivalent to maximizing the **variance** of the X -scores **and** the **correlation** between the X -scores and Y (or the Y -scores).

The method is widely used in chemometrics, as it accurately deals with multicollinear and high-dimensional datasets (more variables than observations). There exists no analytical solution for the PLS problem but iterative algorithms are used (the best known being SIMPLS or NIPALS). The components of the PLS model are constructed sequentially. This is known as “peeling” or “deflating”: at each iteration of the algorithm, after estimating the current component, the matrices X and Y are deflated from the variation associated to this current component. Then, the remaining part of X and Y are used to estimate the next component (Varmuza and Filzmoser, 2009).

The number of latent variables is the only parameter which needs be determined and tuned before fitting the model. It is usually estimated using cross-validation to avoid over-fitting when using the model to predict the target values of new unseen observations.

4.1.2 Methodology

Let us consider a matrix X with n observations and p variables (dimension: $n \times p$) and a response variable y for each observations. This is the PLS1 problem. If we have several response variable, Y of dimension $n \times k$, this constitutes the PLS2 problem. Note that both matrices X and Y should be mean centered on their columns.

The objective is to find the regression coefficients of X to predict Y by decomposing those variables into scores and loadings. That is to say, the data matrix and the target matrix will be projected to a new space. The scores of X should be such that the reconstruction of X is accurate and at the same time an accurate prediction on Y can be made. The notations below will be used.

Notations

X	$n \times p$	Data matrix (mean centered on the columns)
Y	$n \times k$	Target matrix (mean centered on the columns)
B^{PLS}	$p \times k$	Regression coefficients of the model (if $k = 1$, a vector: b^{PLS})
T	$n \times R$	Score matrix of X
t_r	$n \times 1$	Score vector r of X ($r: 1, \dots, R$)
P	$p \times R$	Loading matrix of X
U	$n \times R$	Score matrix of Y
u_r	$p \times 1$	Score vector r of Y ($r: 1, \dots, R$)
Q	$p \times R$	Loading matrix of Y
C	$R \times k$	Coefficients for the “inner relationship” between U and T (PLS2)
c	$R \times 1$	Coefficients for the “inner relationship” between y and T (PLS1)

Objective

The PLS2 problem aims at finding

$$Y = XB^{PLS} + E \quad (4.1)$$

where B^{PLS} is the regression coefficient matrix, and E is the error matrix. The PLS1 problem is equivalent: $y = Xb + e$ where b is the vector of coefficients and e is the vector of errors. The figure 4.1 presents the typical workflow when fitting a PLS to a dataset. In order to find B , the X and Y matrices will be decomposed into the score and loading matrices, such that:

$$\begin{aligned} X &= TP^T + E_X \\ Y &= UQ^T + E_Y \end{aligned} \quad (4.2)$$

where T and U are the score matrices, P and Q the loading matrices, and E_X and E_Y the error matrices, for X and Y respectively. The score and loading matrices each have R columns, which is the number of component for the PLS model, with $R \leq \min(n, p, q)$ (usually chosen by cross-validation). Note also that the scores are **orthogonal** to each other but in general not the loadings. Hence, the R latent variables that will be obtained (contained in matrix T) are uncorrelated.

The **objective of PLS2**, is to maximize the covariance between X and Y scores, that is to say:

$$\max \text{cov}(t_r, u_r)^2 = \max \text{corr}(t_r, u_r)^2 \text{var}(t_r) \text{var}(u_r) \quad (4.3)$$

While the **PLS1 objective** is to maximize the covariance between the scores of X and y :

$$\max \text{cov}(t_r, y)^2 = \max \text{corr}(t_r, y)^2 \text{var}(t_r) \text{var}(y) \quad (4.4)$$

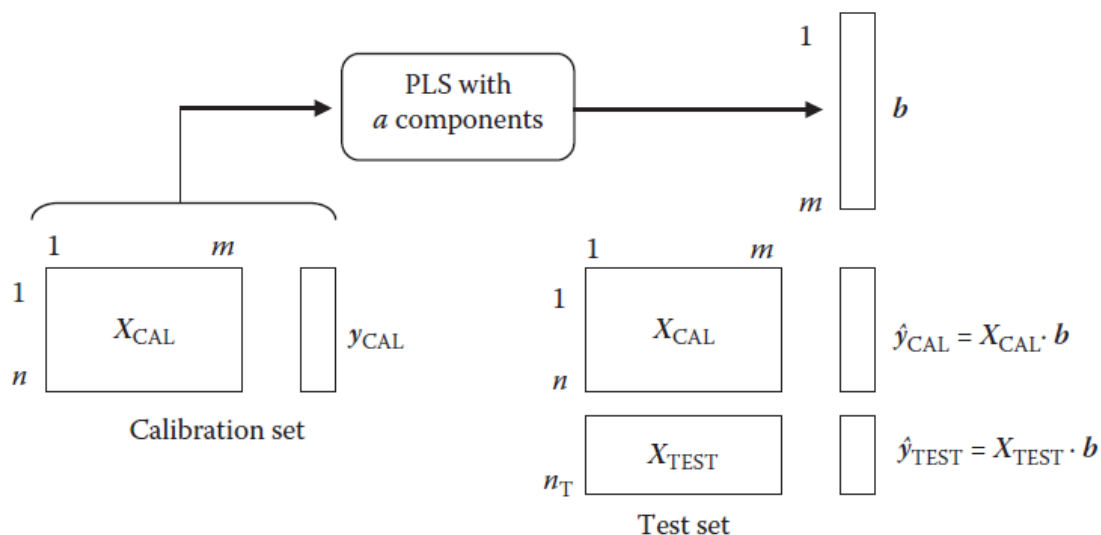


Figure 4.1: Predicting the variable y using a PLS model, taken from Varmuza and Filzmoser (2009), note that n is the number of observations and m is the number of variables

Inner Relationship

In addition, there is an inner linear relationship (Varmuza and Filzmoser, 2009) between U and T such that (for PLS2):

$$U = TC + E \quad (4.5)$$

where C is a diagonal matrix containing the coefficients of the regression and E is the residual matrix. In the particular case where only one component is used for the PLS, this inner relationship becomes $u_r = c_r t_r + e_r$. For PLS1, as T should be a good predictor for y , the inner relationship is:

$$y = Tc + e \quad (4.6)$$

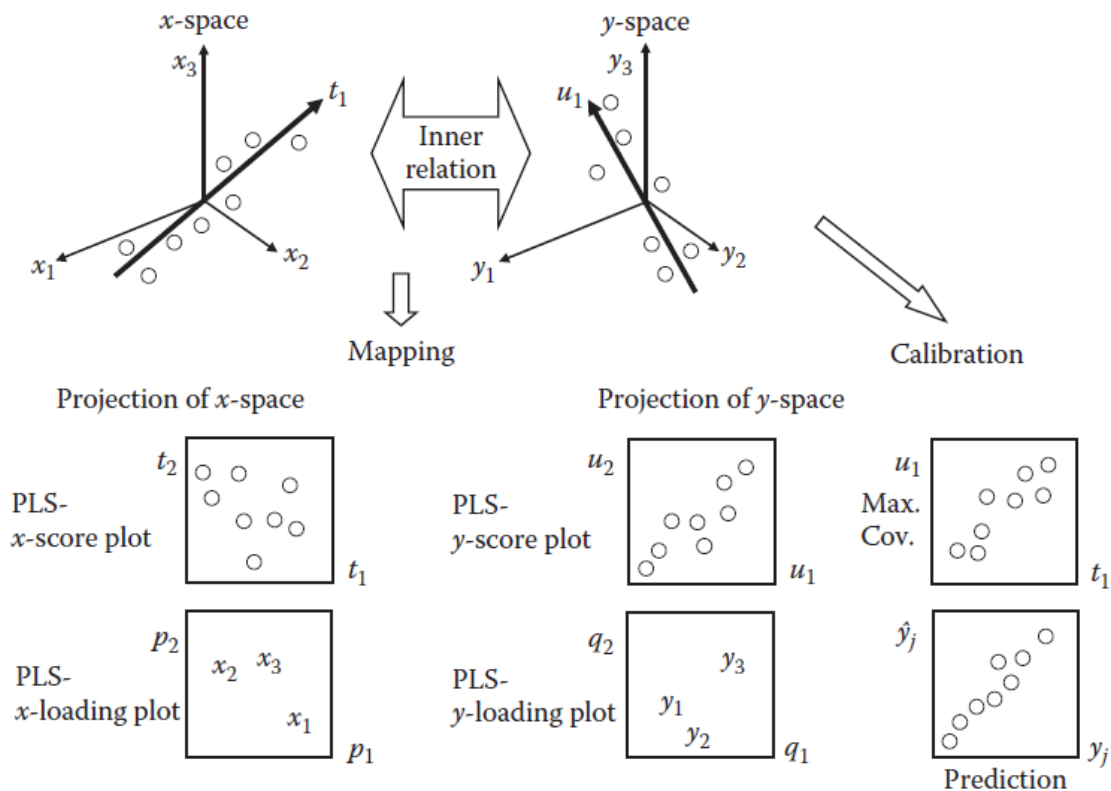


Figure 4.2: PLS2 on X and Y , both have 3 dimensions, the scores are t, u and the loadings p, q for X and Y respectively, taken from Varmuza and Filzmoser (2009)

Obtaining the PLS1 coefficients

Let us now show how the PLS1 regression coefficients can be obtained (see Govaerts, 2018 and Segers, 2019).

Remind that $X = TP^T + E_X$, with the X -scores being linear combinations of X , such that:

$$T = XW \quad (4.7)$$

Starting from the inner relationship:

$$y = Tc + e \quad (4.8)$$

The equation 4.7 is plugged into equation 4.8 and yields:

$$y = XWc + e \quad (4.9)$$

As the objective is to obtain the coefficients such that:

$$y = Xb^{PLS} + e \quad (4.10)$$

With the 2 equations above, the coefficients are defined as:

$$b^{PLS} = Wc \quad (4.11)$$

Finally, from the inner relationship (equation 4.8), $c = (T^T T)^{-1} T^T y$. Hence, the PLS1 coefficients are:

$$b^{PLS} = W(T^T T)^{-1} T^T y \quad (4.12)$$

Note that with orthogonal and normalized X -scores, $T^T T = I_R$ and the coefficients reduce to $b^{PLS} = W T^T y$.

In the following subsection, one of the most popular algorithm for fitting a PLS model is presented: the SIMPLS. However, other algorithms exist: for instance the NIPALS or the Kernel algorithm.

4.1.3 Algorithms

In this section, the SIMPLS for fitting a PLS is presented. As stated above, the X matrix should first be column centered. But first, a small comparison of the existing algorithms is shown.

Indeed, one should be cautious as no analytical solution exists for this problem and the existing algorithms do not always give the same results (but the final regression coefficients B^{PLS} or b^{PLS} are the same for the NIPALS, SIMPLS and Kernel algorithms). The X -scores are orthogonal for both algorithms. For SIMPLS, they are in addition normalized such that: $T^T T = I_R$. The X -loadings are not orthogonal for both algorithms. Their norm is on the other hand different. The equivalence is given by: $P_S = P_N T_N^T T_N$ (with the S index for the SIMPLS algorithm and N for the NIPALS) (Govaerts, 2018).

SIMPLS

The SIMPLS algorithm stands for "straight forward implementation of a statistically inspired modification of the PLS method according to a simple concept" and was introduced by de Jong in 1993.

It solves the objective of maximum covariance as stated above with the constraint of the orthogonality of and normalized T , the X -scores for the different components. The deflation step is different than the one in NIPALS. Indeed, it is done on the covariance matrix $S = X^T y$ (X and y being column **centered**). The result of this algorithm may then be slightly different than NIPALS for the components except the first one (Varmuza and Filzmoser, 2009).

(0) $s_1 = X^T y$	The covariance matrix to be deflated
iterate for r in $1, \dots, R$	
(1) $w_r = s_r / \ s_r\ $	The weight vector is the normalized s_j vector
(2) $t_r = X w_r$	Score vector t_r is a linear combination of X
(3) $t_r = t_r / \ t_r\ $	Normalize the X -score vector
(4) $p_r = X_r^T t_r$	r^{th} loading vector obtained by regressing X_r on t_r
(5) $s_{r+1} = s_r - p_r (p_r^T p_r)^{-1} p_r^T s_r$	Deflation step, go to step 1
Build the matrices	
$T = (t_1, \dots, t_R)$	The X -scores matrix ($n \times R$)
$P = (p_1, \dots, p_R)$	The X -loadings matrix ($p \times R$)
$W = (w_1, \dots, w_R)$	The matrix of weights ($p \times R$)

Note that the step 4 is actually $p_r = X_r^T t_r (t_r^T t_r)^{-1}$ but it simplifies since the scores are orthonormal, such that: $p_r = X_r^T t_r$. Finally, the coefficients are given by

$$B^{PLS} = W T^T Y \quad (4.13)$$

The SIMPLS algorithm exhibits several advantages compared to the NIPALS algorithm. It is first quite simple to implement. In addition, it is also very fast as it

does not require matrix inversion and it does not need a breakdown of the X matrix Varmuza and Filzmoser (2009).

4.1.4 Interpretation of the results

The results of the PLS comprise scores (T for X , U for Y in the case of PLS2) and loadings (P for X , Q for Y in the case of PLS2). Thus, as for a PCA or NMF, one can plot the scores and the loadings. For instance with spectra dataset, the loadings of X will give R curves which represent R latent variables in the dataset. The X -scores will represent the projection of each data-point on the latent variables. Hence, the observations can be compared based on their scores, dissimilar scores indicating that the observations are far from each other in the latent space.

The coefficients B^{PLS} (or b^{PLS} for PLS1) will allow to predict the responses for new data-point. Note that the X and Y matrices were centered on their columns before applying the algorithm, so one needs to add the column means of Y to the prediction, such that (for PLS1): $\hat{y} = \bar{y} + Xb^{PLS}$ with $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$. The coefficients B^{PLS} can also give a detailed view of the influence of each component of the model on the target variable (plotting the coefficients for several different components number). The Root Mean Squared Error of Prediction (RMSEP) can be estimated using a cross-validation technique, for instance the Leave-One-Out (LOO) strategy. Then, n models will be repeatedly fitted on all but one observations and the accuracy (MSE) will be measured on that particular observations. The RMSEP in cross-validation can also be analysed for different number of component in the model. In addition to the cross-validated RMSEP, the percentage of explained variance of both X and Y can be analysed for each component of the model.

To contrast with the NMF, a dataset of spectra could be used to fit a PLS model, using as much components as the number of chemical product in the reaction. Then, the loadings P will comprise R latent variables, but these can hardly be interpreted as the true spectra of each of the chemical product. Indeed, there is no non-negativity constraint in the model and the loadings, coefficients and scores can be negative. Still, one can look for known spectral peaks or profile (Mevik and Wehrens, 2007).

4.2 Supervised NMF in the literature

In the preceding section 3.2.5, the variety of NMF algorithms (including supervised versions) in the literature was first presented. In this section, recent articles using a Supervised NMF algorithm (possibly with additional penalty terms such as sparsity or orthogonality) are presented along with their methodology, algorithm, and field of application.

4.2.1 Supervised NMF for ICU mortality classification

The first paper from Chao et al. in 2018 is focusing on ICU mortality risk prediction. Their objective is an accurate prediction as well as an easy interpretation of the complex temporal data they have at disposal. They propose to add a logistic regression loss into the NMF objective function and derive the gradient update rules.

Using the same notation as in the section presenting the classical unsupervised NMF (see section 3.2), their objective function is shown below.

$$\begin{aligned}
 \min_{U, V, b, b_0} \quad & L_f + L_{lr} + L_r \quad \text{s.t.} \quad U \geq 0, V \geq 0 \\
 L_f = & \frac{1}{2} \|X - UV\|_F^2 \\
 L_{lr} = & \alpha \sum_{i=1}^n \ln \left(1 + \exp(-y_i (\sum_{j=1}^r b_j u_{ij} + b_0)) \right) \\
 L_r = & \frac{1}{2} \beta \left(\sum_{j=1}^r b_j^2 + b_0^2 \right) + \frac{1}{2} \gamma \|U\|_F^2
 \end{aligned} \tag{4.14}$$

The L_f term is the classical NMF problem, L_{lr} is the logistic regression term (using the scores to predict the target labels y , w and b are the weights and bias, respectively). Then, L_r prevents overfitting by adding regularization on the weights, bias and the scores. In this paper, y is a vector of 0 or 1. The parameters α , β and γ weight the penalty terms. The authors finally derive the gradients of the objective function with respect to each parameter: U , V , w , b . As a gradient descent can introduce negative elements in U or V , the authors simply take the positive part for these 2 at each iteration. The initialization for U and V is the result of a NNSVD (Boutsidis and Gallopoulos, 2008) which is a common choice for NMF algorithms.

Finally, the algorithm is applied to simulated data and to ICU data. Each time, they split the dataset into training and test set, fitting a (S)NMF on the train set, and then using the V_{train} to project the test data and obtain estimated scores for the test set (as in section 3.3.3). Then, they fit a logistic regression on top of these scores. For the simulated data, they generate scores and loadings and then add noise to create the data. They use the Area Under Curve (AUC) as metric and vary the amount of noise. They find that their algorithm performs better than the Fisher discriminant NMF and than the standard NMF. They try several values for the parameters α , β and γ , but no information is given about the learning rates used. For ICU dataset, they compare their algorithm to the classical NMF and measure the AUC on a test data, the new algorithm performs better. In their conclusion, the authors state that there is a benefit using the new algorithm, yet the difference is not so big compared to the classical NMF.

4.2.2 Semi Supervised NMF for EEG classification task

In this study, Lee et al. in 2010 also derive a Supervised NMF by incorporating a penalty term based on the target label but they also take into account that the target label is missing for some observations. Thus, they call their algorithm a 'Semi Supervised NMF' but it is in fact a Weighted Supervised NMF. The difference is also that they compute multiplicative update rules instead of a simple gradient descent. They first detail the

weighted part of their algorithm and then the Supervised part. Their obtained objective function is as below.

$$\begin{aligned} \min_{U,V,B} \quad & L_f + L_s \quad \text{s.t.} \quad U \geq 0, V \geq 0 \\ & L_f = \|W \circ (X - UV)\|_F^2 \\ & L_s = \lambda \|L \circ (Y - UB)\|_F^2 \end{aligned} \quad (4.15)$$

Here, $W \in \mathbb{R}^{n \times p}$ (the same dimensions as X), $Y \in \mathbb{R}^{n \times k}$ (the target values, multi class classification with k possible classes, and the class is One-Hot-Encoded: if x_i belongs to class c , then $Y_{ic} = 1$ and the other elements of that row in Y are 0), L also has the same dimensions as Y , and $B \in \mathbb{R}^{r \times k}$ (the coefficients to predict the targets based on the scores). They do not use a logistic penalty term and the gradients are easier to derive, but they minimize a sum of squared errors although their aim is to solve a classification task. The λ parameter balances the two terms in the loss. A variation introduced is the weights matrices W and L on X and Y respectively. Both are only constituted of 0 and 1. If X_{ij} is not observed, then $W_{ij} = 0$, otherwise $W_{ij} = 1$.

$$W_{ij} = \begin{cases} 1, & \text{if } X_{ij} \text{ is observed} \\ 0, & \text{otherwise} \end{cases}$$

Regarding L , the row i of L is filled with 0 if the class of x_i is not known, otherwise it is filled with ones (1_k is a vector of size k with ones).

$$L_{i,:} = \begin{cases} 1_k, & \text{if the class of } x_i \text{ is known} \\ 0, & \text{otherwise} \end{cases}$$

Based on this objective function, they derive multiplicative update rules after computing the gradients with respect to U , V and B , mimicking the approach of Lee and Seung (2001). It has the advantage to eliminate the learning rates from the algorithm, hence there are fewer parameters to choose.

Turning now to their numerical experiments, they compare the Fisher NMF with their algorithm. They partition the dataset to obtain X_{train} and X_{test} , they apply the NMF algorithms to the train set and obtain U_{train} and V_{train} , then they obtain the scores for the left aside test set using a projection step, as Thiel et al. (2021) in section 3.3.3. Finally, they fit a classifier (SVM) on the obtained U_{train} and test it using $\hat{U}_{test}$, the scores obtained after the projection. The metric used is the classification accuracy. The conclusion is that the Semi Supervised NMF algorithm performs better than the Fisher Discriminant NMF.

4.2.3 Supervised NMF for spectrometry data classification

The last paper was published in 2018 by Leuschner et al. and aims at decomposing a MALDI imaging dataset after adding supervised classification and various unsupervised penalty terms. Their paper is particularly well documented and they compared a high number of algorithms derived from the NMF problem. On the other hand, they did not try other classical methods to compare against their proposed algorithms. Their target problem is a binary classification (tumor detection) on MALDI MSI data.

Their study is divided between unsupervised and supervised extensions of the NMF problem. The objective functions all have the form:

$$\min_{U,V} \frac{1}{2} \|X - UV\|_F^2 + \text{penalty terms} \quad (4.16)$$

Unsupervised extensions

First, the **unsupervised extensions** define new objective functions integrating ℓ_1 and ℓ_2 penalty terms. Namely, the first objective comprises a ℓ_2 -penalty on U and V and a

ℓ_1 -penalty on V , increasing the sparsity of the basis elements found: $\frac{\lambda}{2}\|V\|_1 + \frac{\mu}{2}\|U\|_F^2 + \frac{\nu}{2}\|V\|_F^2$. Then, they define another expression by adding a constraint on the orthogonality of the loadings V , to obtain less correlated patterns by means of 2 penalties (instead of the single penalty: $\|I - VV^T\|_F^2$): $\frac{\sigma_1}{2}\|I - VW^T\|_F^2 + \frac{\sigma_2}{2}\|W - V\|_F^2$. The latter orthogonality penalties add a new parameter W to the minimization problem.

Then, they show how to obtain MU rules (following the approach of majorize-minimization algorithms which is not detailed here). In addition they precisely describe how to use the result of their regularized NMF problems as the input of a classifier. Their method comprises two steps, first the decomposition is applied on a training set and results in $X_{train} = U_{train}V_{train}$, then the feature vector (F) is defined as:

$$X_{train}V_{train}^T = F_{train}$$

And the feature vector F_{train} is used with the target labels Y_{train} to fit a classifier (a linear discriminant analysis or a logistic regression). New data X_{test} are then classified by first extracting the feature vector:

$$X_{test}V_{train}^T = F_{test}$$

And feeding F_{test} as the input of the previously trained classifier. Nevertheless with this approach, the loadings matrix only contain the most dominant patterns in the data, which are not necessarily the most discriminative. Thus, this motivates the next part of their paper in which they integrate supervised penalty terms in the objective function of the NMF.

Supervised extensions

Turning now to the **supervised extensions**, they add to the classical NMF problem the penalty of a linear discriminant analysis or the logistic regression. But the input of those 2 classification penalties are the feature vectors: XV^T and not the scores U as in the previous articles ! For instance, the linear discriminant analysis penalty is:

$$\frac{\gamma}{2}\|Y - XV^TB\|_F^2$$

with B the coefficients of the regression. Then, they also incorporate various penalties to compare models with Orthogonality constraints, sparsity constraints, etc. The MU rules are also derived.

Finally, they also need to design a pipeline to train and test their supervised dimensionality reduction technique. Here, they slightly change their approach with respect to the unsupervised part. Indeed, one can either keep the B coefficients at the last iteration of the algorithm or optimize a new one. When keeping the coefficients, then the classifier can be directly used to predict the labels of the training data, the input being the feature data: $X_{train}V_{train}^T = F_{train}$. The prediction for the test data is simply given using the coefficients and the input: $X_{test}V_{train}^T$. For instance with the linear discriminant analysis, one computes $X_{test}V_{train}^TB$ and obtains real values, for each value, if it is below a certain threshold, Y_{test} is 0, otherwise it is 1.

On the other side, when ignoring the B coefficients, a supervised NMF is fitted on X_{train} , producing U_{train} and V_{train} . The feature data $X_{train}V_{train}^T = F_{train}$ and the training labels Y_{train} are used as inputs to fit a linear discriminant or a logistic regression classifier, giving the coefficients B . Lastly, the prediction for a new set of observations X_{test} is given by feeding $X_{test}V_{train}^T$ to the trained classifier.

For the case study, they use 8-folds cross-validation on the dataset and measure the balanced classification accuracy (this metric has the advantage to be independent on the class priors). Overall, the supervised extensions perform better than the unsupervised ones. The authors are also able to identify some mass spectrometric markers by visualizing the patterns: V^TB .

4.2.4 Discussion and comparison

From the three presented articles, the aim of the authors was always to predict a class label (binary or multi-class). Nevertheless, the supervised penalty terms used were sometimes a classification one (for instance, a logistic regression), or sometimes a sum of squared errors (which is more common in regression problems). A very important difference is the input for the classification models: the scores (U) can be used but the last paper proved that XV^T can also be used.

The three studies also showed us that the extra step required to use the result of the (S)NMF is not always well documented although it is crucial to allow the reproducibility and intelligibility of the results. Moreover, the literature does not seem to be unanimous on this subject. Thus, the pipeline for fitting and testing the extended NMF problem needs to be carefully designed. It is also a function of the input taken by the classification model: the MALDI imaging article using feature data as input shows a simple way to extend the classification model to new data while the other articles using the scores as input requires a projection as seen in Thiel et al. (2021).

However, the projection step of Thiel et al. can be seen as a generalization of the “feature data” by Leuschner et al.. Indeed, if the loadings are orthogonal and normalized, then both approaches are equivalent (see section 3.3.4):

$$\begin{aligned} F_{test} &= X_{test} V_{train}^T && \text{(input for classifier in Leuschner et al. 2018)} \\ \hat{U}_{test} &= X_{test} V_{train}^T (V_{train} V_{train}^T)^{-1} && \text{(input for regressor in Thiel et al. 2021)} \end{aligned} \quad (4.17)$$

with orthonormal loadings: $V_{train} V_{train}^T = I$, both inputs are equivalent, but this is not the case in the study of Leuschner et al..

In addition, the methods used to compare the new methods presented in the 3 papers above do not comprise a one-step approach such as a PLS-DA. The authors only compare their new methods to other NMF related methods. One can thus reasonably ask the question: would these methods perform better than PLS ?

In the next section, the NMF problem will be extended to a regression problem, using a sum of squared error penalty term (or ℓ_2 -penalty). The input for the regression will be the scores U (as Chao et al. 2018, Lee et al. 2010) and not the feature data (as Leuschner et al. 2018). The Gradient update rules and MU rules will be derived.

4.3 New algorithm for Supervised NMF

4.3.1 Objective function

The usual NMF notation is used in this section. A X matrix containing the n observations and p variables, $X \in \mathbb{R}_+^{n \times p}$. We will obtain the decomposition $X \approx UV$ with $U \in \mathbb{R}_+^{n \times r}$ (the scores) and $V \in \mathbb{R}_+^{r \times p}$ (the loadings) such that we minimize $\|X - UV\|_F^2$. At the same time, we want to minimize the linear regression error between the target variables $Y \in \mathbb{R}^{n \times k}$ (Y includes k variables with n observations each) and the scores U . That is to say, we add $\|Y - UB\|_F^2$ to the objective function. The objective function to minimize is thus

$$\min_{U, V, B} \frac{1}{2} \|X - UV\|_F^2 + \frac{\lambda}{2} \|Y - UB\|_F^2 \quad (4.18)$$

For clarity purpose, the following parts are defined: $\Theta = \frac{1}{2} \|X - UV\|_F^2$ and $\Omega = \frac{\lambda}{2} \|Y - UB\|_F^2$.

4.3.2 Deriving the gradients

First, the gradient of Θ is derived. Using $\|X\|_F^2 = \text{Tr}(X^T X)$, leads:

$$\begin{aligned} \frac{1}{2} \|X - UV\|_F^2 &= \frac{1}{2} \text{Tr}((X - UV)^T (X - UV)) \\ &= \frac{1}{2} \text{Tr}(X^T X - X^T UV - V^T U^T X + V^T U^T UV) \\ &= \frac{1}{2} (\text{Tr}(X^T X) - \text{Tr}(X^T UV) - \text{Tr}(V^T U^T X) + \text{Tr}(V^T U^T UV)) \end{aligned} \quad (4.19)$$

Using $\nabla_X \text{Tr}(AX) = A^T$ and $\nabla_X \text{Tr}(X^T A) = A$, the gradients of $\|X - UV\|_F^2$ with respect to V and U are:

$$\nabla_V \Theta = \frac{1}{2} (2U^T UV - 2U^T X) \quad (4.20)$$

$$\nabla_U \Theta = \frac{1}{2} (2UVV^T - 2XV^T) \quad (4.21)$$

Second, the gradient of Ω is derived. As above, it yields:

$$\begin{aligned} \frac{\lambda}{2} \|Y - UB\|_F^2 &= \text{Tr}(\frac{\lambda}{2} (Y - UB)^T (Y - UB)) \\ &= \frac{\lambda}{2} \text{Tr}(Y^T Y - Y^T UB - B^T U^T Y + B^T U^T UB) \end{aligned} \quad (4.22)$$

The gradients of $\lambda \|Y - UB\|_F^2$ with respect to B and U are:

$$\nabla_B \Omega = \frac{\lambda}{2} (2U^T UB - 2U^T Y) \quad (4.23)$$

$$\nabla_U \Omega = \frac{\lambda}{2} (2UBB^T - 2YB^T) \quad (4.24)$$

4.3.3 Gradient descent rules

Rules

From the gradients in the above subsection, the (additive) update rules for a Gradient Descent are defined as:

$$\begin{aligned} U &\leftarrow U - \gamma_U (UVV^T - XV^T + \lambda(UBB^T - YB^T)) \\ V &\leftarrow V - \gamma_V (U^T UV - U^T X) \\ B &\leftarrow B - \gamma_B \lambda (U^T UB - U^T Y) \end{aligned} \quad (4.25)$$

Fitting an intercept to B

One may want to fit an intercept to the coefficients B such that $\tilde{B} \in \mathbb{R}^{r+1 \times k}$. It is a bit more challenging than it seems. We propose the following method to fit an intercept. A column of 1 is added to the scores U and one coefficient is added to B , but then problems arise with the dimensions as $\tilde{B} \in \mathbb{R}^{r+1 \times k}$ and $\tilde{U} \in \mathbb{R}^{n \times r+1}$. The rules change for U and $\tilde{B}$ but not for V . The tilde marks a modification from the original matrices caused by adding the intercept.

For the update rule of U , the last term is $\tilde{U}\tilde{B}\tilde{B}^T - Y\tilde{B}^T$, where $Y : n \times k$ and $\tilde{B} : r+1 \times k$. The first column of this term is excluded (noted $[-1]$ in the equation). Hence, the scores $U \in \mathbb{R}^{n \times r}$.

Finally, the weights $\tilde{B}$ are obtained by adding to the scores U a first column full of 1 such that $\tilde{U} \in \mathbb{R}^{n \times r+1}$, thus $B \in \mathbb{R}^{r \times k}$ becomes $\tilde{B} \in \mathbb{R}^{r+1 \times k}$.

$$\begin{aligned} U &\leftarrow U - \gamma_U(UVV^T - XV^T + \lambda(\tilde{U}\tilde{B}\tilde{B}^T - Y\tilde{B}^T)_{[-1]}) \\ V &\leftarrow V - \gamma_U(U^TUV - U^TX) \\ \tilde{B} &\leftarrow \tilde{B} - \gamma_{\tilde{B}}\lambda(\tilde{U}^T\tilde{U}\tilde{B} - \tilde{U}^TY) \end{aligned} \tag{4.26}$$

Nonnegativity constraint

A Gradient Descent algorithm will not be able to take the nonnegativity constraints on U and V into account. A projection step will therefore be needed after the update so that every element of U and V remain nonnegative (if an element ≤ 0 then replace it by 0.0001).

4.3.4 Multiplicative Update rules

Methodology

In a previous section, the Multiplicative Update rules of Lee and Seung (2001) were presented. The idea behind these being that they are equivalent to a rescaled gradient descent. There is another way of understanding the MU which is used to elaborate our own MU rules.

As explained by Lee et al. in 2010, the Multiplicative Update rules can be obtained using another rationale. If the gradient of a function f has a decomposition of the form

$$\nabla_{\alpha}f = [\nabla_{\alpha}f]^+ - [\nabla_{\alpha}f]^- \tag{4.27}$$

with $[\nabla_{\alpha}f]^+ > 0$ and $[\nabla_{\alpha}f]^- > 0$. Then MU for parameters α has the form

$$\alpha \leftarrow \alpha \circ \left(\frac{[\nabla_{\alpha}f]^-}{[\nabla_{\alpha}f]^+} \right) \tag{4.28}$$

Applying 4.28 leads indeed to the MU. This is also put in other words by Gillis (2014). He shows (taking for the example the MU for U : $U \leftarrow U \circ \frac{[XV^T]}{[UVV^T]}$) that

$$\frac{[XV^T]_{ik}}{[UVV^T]_{ik}} \geq 1 \iff (\nabla_U f)_{ik} \leq 0 \tag{4.29}$$

where $\nabla_U f = -XV^T + UVV^T$ (see 3.7). This means that for each element of U , the MU rules either “(i) increase it if its partial derivative is negative, (ii) decrease it if its partial derivative is positive, or (iii) leave it unchanged if its partial derivative is equal to zero” (Gillis, 2014, p. 8). Consequently, the cost function will decrease when using the MU rules.

Rules

From the section 4.3.2 above, the 'parts' of the gradients are:

$$\begin{aligned}\nabla_V \Theta &= [\nabla_V \Theta]^+ - [\nabla_V \Theta]^- \\ &= U^T UV - U^T X\end{aligned}\tag{4.30}$$

$$\begin{aligned}\nabla_U \Theta &= [\nabla_U \Theta]^+ - [\nabla_U \Theta]^- \\ &= UVV^T - XV^T\end{aligned}\tag{4.31}$$

$$\begin{aligned}\nabla_B \Omega &= [\nabla_B \Omega]^+ - [\nabla_B \Omega]^- \\ &= \lambda U^T UB - \lambda U^T Y\end{aligned}\tag{4.32}$$

$$\begin{aligned}\nabla_U \Omega &= [\nabla_U \Omega]^+ - [\nabla_U \Omega]^- \\ &= \lambda UBB^T - \lambda YB^T\end{aligned}\tag{4.33}$$

Using these gradients and the methodology in 4.3.4, the following MU rules are defined as:

$$\begin{aligned}U &\leftarrow U \circ \frac{XV^T + \lambda YB^T}{UVV^T + \lambda UBB^T} \\ V &\leftarrow V \circ \frac{U^T X}{U^T UV} \\ B &\leftarrow B \circ \frac{U^T Y}{U^T UB}\end{aligned}\tag{4.34}$$

Fitting an intercept to B

As in the previous section, B will become $\tilde{B} \in \mathbb{R}^{r+1 \times k}$ and U becomes $\tilde{U} \in \mathbb{R}^{n \times r+1}$.

For the update rule of U , the second term of the numerator is $\lambda Y \tilde{B}^T$, $Y : n \times k$ and $\tilde{B} : r+1 \times k$, from this term the first column is excluded (noted $[-1]$ in the equation). Same is done for the denominator, after adding a column of 1 to U for the second term ($\lambda \tilde{U} \tilde{B} \tilde{B}^T$, $\tilde{U} : n \times r+1$ and $\tilde{B} : r+1 \times k$). The scores U after the update are thus $U \in \mathbb{R}^{n \times r}$.

Finally, the weights $\tilde{B}$ are obtained using $\tilde{U}$ and $\tilde{B}$ such that the update rules return $\tilde{B}$.

$$\begin{aligned}U &\leftarrow U \circ \frac{XV^T + (\lambda Y \tilde{B}^T)_{[-1]}}{UVV^T + (\lambda \tilde{U} \tilde{B} \tilde{B}^T)_{[-1]}} \\ V &\leftarrow V \circ \frac{U^T X}{U^T UV} \\ \tilde{B} &\leftarrow \tilde{B} \circ \frac{\tilde{U}^T Y}{\tilde{U}^T \tilde{U} \tilde{B}}\end{aligned}\tag{4.35}$$

Stopping criterion

Both for the Gradient Descent or the Multiplicative Update rules, the stopping criterion should be defined. A common choice is to measure the objective/loss function at each iteration and stop the algorithm when the difference between consecutive iterations is too low. This is the retained approach for the SNMF algorithm. The parameter measuring this difference is ϵ with the condition: if $|loss_k - loss_{k-1}| \leq \epsilon$ then stop the algorithm (k is the index of the iteration number).

Nonnegativity constraint

The Multiplicative Update rules enforce the nonnegativity constraints on U and V but it also does on B . Nevertheless, having only positive coefficients in B is not necessarily desired.

Exploring solutions to this problem is necessary. However, using the MU rules for U and V and another one relaxing the nonnegativity constraint for B will lead to the loss of the nonnegativity guarantee on U and V . Indeed, if elements in B are negative, negative elements are introduced in the update rule of U , and if U may contain negative elements, so may V . The projection step will then be necessary if only the MU rules for U and V are used. Regarding the update rule of B , a linear regression $Y \sim UB$ could be fitted at each iteration (with or without intercept). But then, the update rules are no longer guaranteed to reduce the loss function at each iteration, this is only valid using the MU rules for all parameters. A gradient descent with a carefully chosen learning rate could also be used.

As a conclusion, there is a choice to make. On the one hand, one can have the guarantee of reducing the objective function at each iteration with the MU for all parameters. On the other hand, the B coefficients can then only be non-negative. Note that when predicting several targets with opposite effects such as the concentration of a product and a reactant, we can flip one of the target (do: $100 - \text{product_concentration}$, for instance) so that nonnegative coefficients of the scores are easier to find. Indeed, both targets are now varying into the same direction.

4.4 Prediction and methods comparison on new dataset

In this section, the methodology used to compare the supervised algorithms is detailed. The first is a one step PLS. The second and the third are a two steps dimensionality reduction + regression based on the scores and loadings produced by the NMF or the proposed Supervised NMF.

4.4.1 One step PLS

The one step approach is straightforward, it simply consists in fitting a PLS model using the SIMPLS algorithm on the dataset of spectra and measuring the error of prediction on each batch. Thus, a PLS is trained on $X_{G/i}$ (the spectra of all but one batch) and $Y_{G/i}$ (the corresponding target values). It results in the PLS model: $\hat{Y}_{G/i} = X_{G/i} \hat{B}_{G/i}^{PLS}$.

For the particular case when y is not available for every spectra x , the subset from matrix $X_{G/i}$ is extracted where the spectra corresponding to the off-line target are gathered in the matrix $X_{G/i}^*$. Hence, $X_{G/i}^*$ and $Y_{G/i}^*$ have the same number of rows. The PLS model is: $\hat{Y}_{G/i}^* = X_{G/i}^* \hat{B}_{G/i}^{PLS}$. The coefficients $\hat{B}_{G/i}^{PLS}$ can be used to predict Y for a new set of spectra.

4.4.2 Predicting a new observation after the (S)NMF

Regarding the comparison of the prediction accuracy between a supervised method such as PLS with a decomposition such as NMF or SNMF, one issue remains. Indeed, the PLS, after internally decomposing the dataset, returns coefficients which can be directly applied on new unseen data. On the other hand, a NMF or a SNMF only produces scores and loadings for the dataset on which the method was fitted. In other words, training a dimensionality reduction technique on X_{train} yields U_{train} and V_{train} . An extra step is required to obtain coefficients to apply to a new validation set X_{valid} .

The needed procedure is called the two steps regression approach. The figure 4.3 summarizes it, using the *train* and *valid* indices (note that it is equivalent to the G/i and i indices). It can be seen that $\hat{Y}_{valid}$ can only be obtained using the coefficients $\hat{B}_{train}$ from the projection step. On the other hand, $\hat{Y}_{train}$ can be obtained in two different ways: by using the coefficients $\hat{B}_{train}$ or the regression coefficients $\hat{C}_{train}$ one step ahead of the projection.

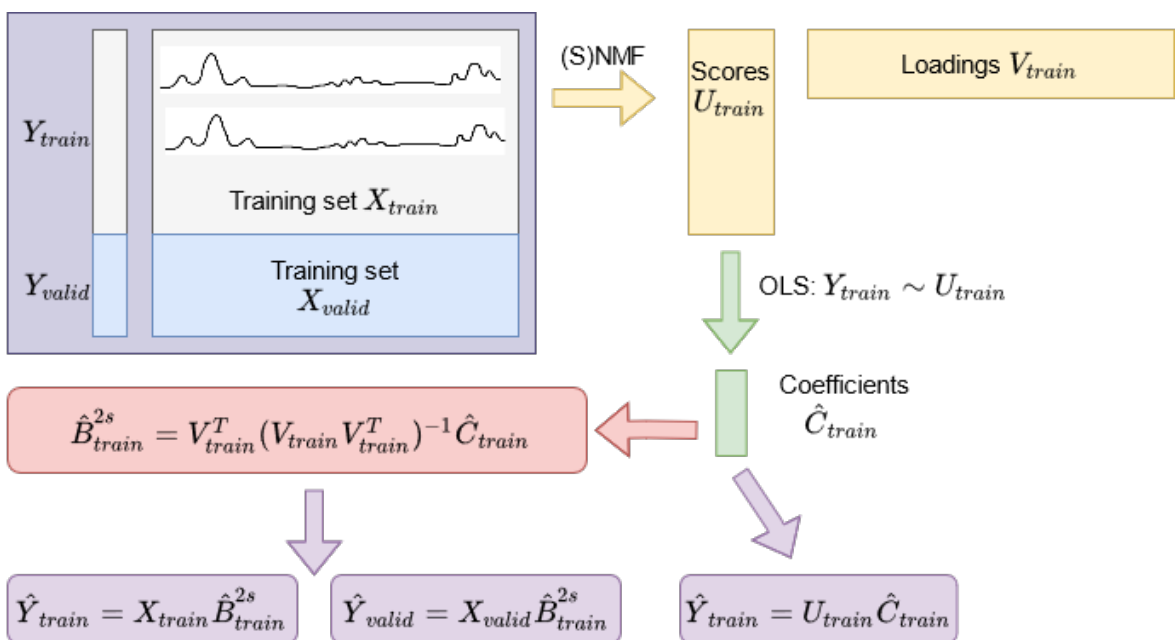


Figure 4.3: Two steps regression

The two steps regression is further detailed in the next subsection, using the batch notation previously defined.

4.4.3 Two steps (NMF/SNMF) + regression

The two steps approach consists in applying first a dimensionality reduction technique on the dataset of spectra before a regression step.

1. The first step is the dimensionality reduction performed using an NMF or a SNMF. It yields:

$$X_{G/i} = U_{G/i}V_{G/i} + E_{G/i} \quad (4.36)$$

with $U_{G/i}$ being the scores, $V_{G/i}$ the loadings and $E_{G/i}$ the error matrix.

If the dataset has as many target values as spectra, the particular case is not met and the step 2 is the following:

2. A. The second step is the regression one. It uses the scores obtained $U_{G/i}$ to fit the linear regression model:

$$\begin{aligned} Y_{G/i} &= U_{G/i}C_{G/i} + E \\ \hat{C}_{G/i} &= (U_{G/i}^T U_{G/i})^{-1} U_{G/i}^T Y_{G/i} \end{aligned} \quad (4.37)$$

Defining $\hat{U}_{G/i} = X_{G/i}V_{G/i}^T(V_{G/i}V_{G/i}^T)^{-1}$, (see 3.3.3) it yields:

$$Y_{G/i} = \hat{U}_{G/i}\hat{C}_{G/i} + \hat{E} = X_{G/i}V_{G/i}^T(V_{G/i}V_{G/i}^T)^{-1}\hat{C}_{G/i} + \hat{E} \quad (4.38)$$

Finally, the regression coefficients $\hat{B}_{G/i}^{2S}$ are obtained as:

$\hat{B}_{G/i}^{2S} = V_{G/i}^T(V_{G/i}V_{G/i}^T)^{-1}\hat{C}_{G/i}$ and the two-steps model is:

$$\hat{Y}_{G/i} = X_{G/i}\hat{B}_{G/i}^{2S} \quad (4.39)$$

If the dataset has the particular property that there is less target values than spectra, the step 2 is slightly different. After the first step, $U_{G/i}^*$ is obtained as the subset of $U_{G/i}$ where the corresponding spectra have an offline measurement (a target value). The step 2 is the following:

2. B. The second step if the particular case is met uses the scores obtained $U_{G/i}^*$ to fit the linear regression model:

$$\begin{aligned} Y_{G/i}^* &= U_{G/i}^*C_{G/i} + E \\ \hat{C}_{G/i} &= (U_{G/i}^{*T} U_{G/i}^*)^{-1} U_{G/i}^{*T} Y_{G/i}^* \end{aligned} \quad (4.40)$$

Defining $\hat{U}_{G/i}^* = X_{G/i}^*V_{G/i}^T(V_{G/i}V_{G/i}^T)^{-1}$, (see 3.3.3) it yields:

$$Y_{G/i}^* = \hat{U}_{G/i}^*\hat{C}_{G/i} + \hat{E} = X_{G/i}^*V_{G/i}^T(V_{G/i}V_{G/i}^T)^{-1}\hat{C}_{G/i} + \hat{E} \quad (4.41)$$

Finally, the regression coefficients $\hat{B}_{G/i}^{2S}$ are obtained as:

$\hat{B}_{G/i}^{2S} = V_{G/i}^T(V_{G/i}V_{G/i}^T)^{-1}\hat{C}_{G/i}$ and the two-steps model is:

$$\hat{Y}_{G/i}^* = X_{G/i}^*\hat{B}_{G/i}^{2S} \quad (4.42)$$

4.4.4 Model quality assessment

As the aim is to predict the concentration of chemical components, the metric used for this regression problem is the RMSECV (Root Mean Squared Error in Cross Validation).

$$RMSECV_{ik} = \sqrt{\frac{\sum_{j=1}^{n_i} (Y_{ikj} - \hat{Y}_{ikj})^2}{n_i}}$$

with Y_i the matrix of true target for the batch i and $\hat{Y}_i$ the predicted target. n_i is the number of observations in this batch. Then, the index k stands for the column number of Y_i , since several target concentrations may be predicted. Finally, the index $j : 1, \dots, n_i$ iterates over the row number of Y_{ik} .

Note that for a given target k the mean RMSECV of the RMSECVs over several batches/folds defined as follows:

$$RMSECV_k = \frac{\sum_{i=1}^I RMSECV_{ik}^2}{L}$$

with i the index of the RMSECV and I the total number of different RMSECVs for batches or folds.

Chapter 5

Methods application on case studies

5.1 Methodology

Before analyzing the three datasets, let us present the detailed methodology in this section. The structure, objective of the new SNMF and R implementation is shown below.

5.1.1 Structure

The structure of the application of the methods to each case study is presented in the figure 5.1. Instead of dividing it between the unsupervised and the supervised techniques, the preferred organization is the following:

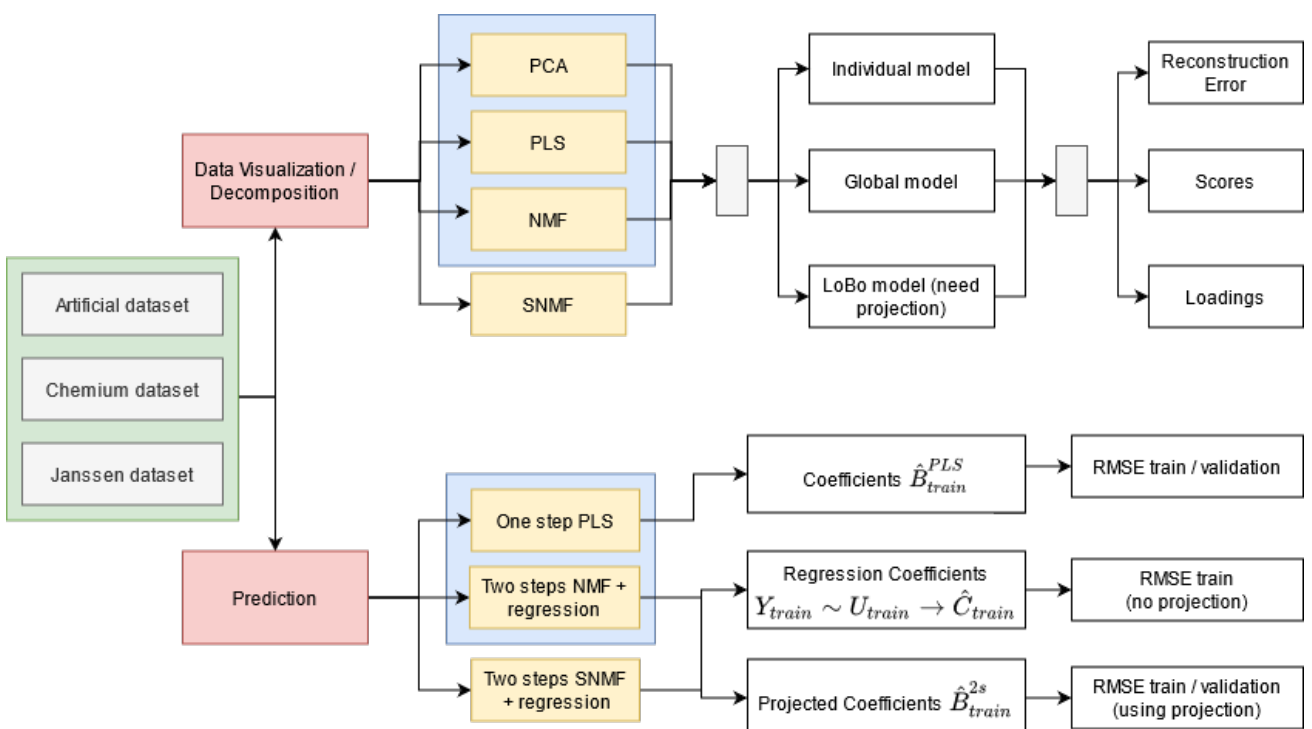


Figure 5.1: Work Plan for the case studies

1. **Data Visualisation/Decomposition:** the scores and loadings produced for the data matrix are compared for the PCA, NMF, PLS and SNMF. Then, the reconstruction error is also studied. Obviously, the unsupervised methods are expected to perform better at the reconstruction error because it is their only objective. On the other hand, it is interesting to analyse the interpretability of our new algorithm and its reconstruction error compared to a classical NMF and the other methods. This is the reason why the first part of the analysis is not limited to the unsupervised methods only.
2. **Prediction:** the supervised methods are fitted and evaluated in cross-validation. The metric used is the RMSE as explained in section 4.4.4. The one step PLS,

and the two steps (S)NMF are compared. Note that PLS produces regression coefficients directly so the RMSE train and validation can easily be obtained.

On the other hand, as seen in the figure 4.3, the first step of the two steps methods is to produce the OLS regression coefficients $\hat{C}_{train}$ which result from:

$$\hat{Y}_{train} = U_{train} \hat{C}_{train}$$

thus the training RMSE can be measured on each batch/fold (and not the validation RMSE, since one needs the scores which are only available for the training set !).

To obtain regression coefficients applicable to new observations (and not to scores), the extra projection step is required and produces $\hat{B}_{train}^{2s}$ (see figure 4.3). Then the RMSE train using the projected coefficients results from:

$$\hat{Y}_{train} = X_{train} \hat{B}_{train}^{2s}$$

and the RMSE on the validation set (RMSECV) using the projected coefficients results from:

$$\hat{Y}_{valid} = X_{valid} \hat{B}_{train}^{2s}$$

5.1.2 Objective

The objective for the new SNMF algorithm is to have an equivalent reconstruction error as the classical NMF problem (yet, it is expected to slightly be larger since the reconstruction is no longer the only objective of the loss function), have a good interpretability in terms of scores and loadings.

At the same time, the new algorithm + regression should perform better at predicting Y compared to a simple NMF + regression model. If it is the case in cross-validation, it means that the new algorithm is able to capture more relevant patterns to predict the target than the unsupervised NMF.

5.1.3 R packages implementation

The whole implementation was performed using **R** (the PCA, NMF, PLS and the proposed Supervised NMF).

- For the PCA, the function `prcomp` was used (from the package `stats` which comes directly with R). The algorithm used by `prcomp` is a SVD.
- For the PLS, the function `pls` from the `pls` package was used. The algorithm chosen was SIMPLS.
- For the NMF, the Brunet algorithm from the `nmf` function from the `NMF` package is used (remind that it is based on the Kullback-Leibler divergence, not on the Frobenius norm, but it is an enhanced version of the MU of Lee and Seung).

5.2 Artificial dataset

In this section, the artificial dataset described in the second chapter is analyzed. The details of its construction can be found in the section 2.1. But first, let us define some general parameters for this case study.

- The target concentration to predict by the supervised model is the component C (the product).
- For the NMF and the SNMF, 3 components will be used (indeed, it is known by design that there are 3 latent components in the data)
- For the PLS and the PCA, as the data must be first centered, only 2 components will be used.

5.2.1 SNMF parameters

Regarding the SNMF, many parameters are at stake. We chose to study only 3 different configurations of the SNMF for this case study. The table 5.1 shows the main differences between these models. The regression coefficients B in the supervised penalty term $\frac{1}{2}\|Y - UB\|_F^2$ can be initialized and updated according to various rules. Having a full Multiplicative Update algorithm for U , V and B has the advantage that the loss should decrease at each iteration while at the same time enforcing the non-negativity constraint and having an automatic learning rate. On the other hand, as discussed in the section 4.3.4, having non-negative B is not necessarily desirable.

Consequently, in addition to the classic MU algorithm 'SNMF1' with positive B , the second SNMF model 'SNMF2' enables the coefficients to be negative. Indeed, At initialization, U_{init} and V_{init} are the result of an NMF of X , and B_{init} is set as the OLS regression coefficients of $Y \sim U_{init}$, hence possibly negatives. Then, B is updated by the classic MU rules. Note that there is no longer a guarantee that U and V are non-negative since the update rules of U depends on B , so at each iteration the negative elements of U are replaced by 0.0001. Then, the 'SNMF3' model experiments a new way of updating the coefficients B by setting them to the OLS coefficients from $Y \sim U_k$ at each iteration k of the algorithm ($B_k = (U_k^T U_k)^{-1} U_k^T Y$). Again, the negative elements of U are replaced by 0.0001 at each iteration. The table 5.1 summarizes the main differences of the 3 models.

SNMF Model Name	init B	update B
MU with random B init (SNMF_1)	random uniform $[0, 1]$	multiplicative
MU with smart B init (SNMF_2)	OLS	multiplicative
Mixed MU with OLS (SNMF_3)	OLS	OLS

Table 5.1: 3 different models of SNMF

The other SNMF parameters are fixed for all three models. The update of U, V is done by the MU rules, the maximum number of iterations/epochs is 100, the initialization of U, V is the result of an NMF on X with the Brunet algorithm. Regarding the value of λ , the weight for the supervised penalty, a grid search is performed and the mean validation RMSE (the mean RMSECV) are compared in a leave-one-batch-out cross-validation.

The figure 5.2 shows the evolution of the mean validation RMSE depending on the value of lambda for SNMF1. It can be seen that the lowest mean validation RMSE are obtained with very low values for lambda (thus very low weights on the supervised penalty term). The chosen λ is the one giving the best mean RMSECV. The λ is $1e-5$, $1e-4$, $5e-6$ for the SNMF1, SNMF2 and SNMF3, respectively. The parameter for the stopping criterion ϵ is fixed to 0.001 (when the difference of consecutive objective function is lower than 0.001, the algorithm stops).

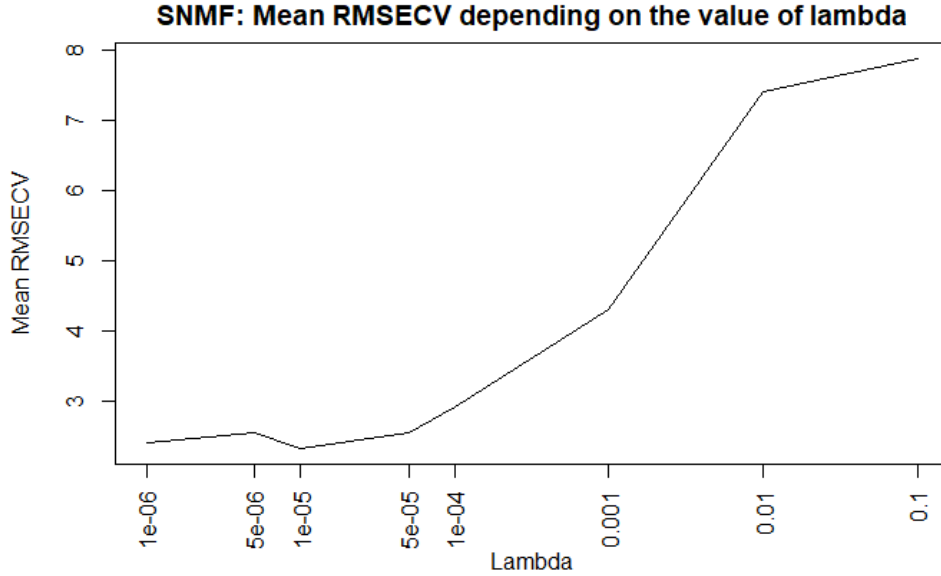


Figure 5.2: Evolution of the mean RMSECV for SNMF1

5.2.2 Analyze of the SNMF algorithms

Before describing the output of the SNMF models, let us focus on the values of the objective function when varying the balancing factor λ . From the objective function of the SNMF, two terms can be distinguished:

- The penalty 1 ('p1') is the NMF loss function: $\frac{1}{2}\|X - UV\|_F^2$
- The penalty 2 ('p2') is the supervised penalty: $\frac{1}{2}\|Y - UB\|_F^2$

The figure 5.3 is obtained by fitting global models with different values for λ and measuring the values of 'p1' and 'p2' at the termination of the SNMF1 algorithm. This figure clearly shows the trade off between the supervised penalty term and the NMF cost function. When λ is small, the NMF loss function 'p1' is small. On the other hand, increasing λ gives more weight to the supervised penalty term ('p2') thus the value of 'p2' is lower but the value of 'p1' increases. Note that the supervised penalty term can take very large values because the coefficients for the linear regression $Y = UB + E$ may not have converged yet depending on the value of λ . For instance, if λ is very small, the regression coefficients B at the end of the algorithm may still be close to the random initial ones, therefore making no sense.

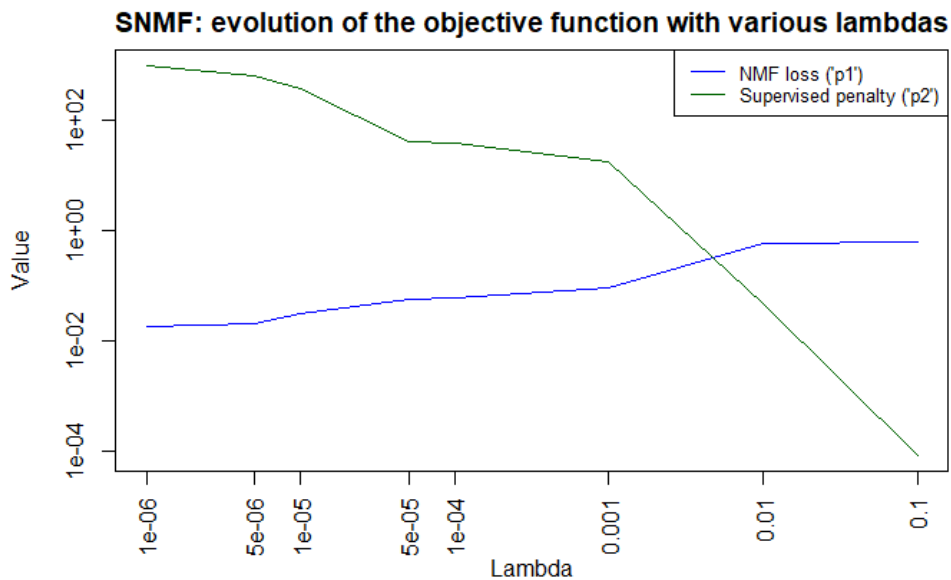


Figure 5.3: Evolution of 'p1' and 'p2' (after the last iteration) for SNMF1 depending on λ (global model)

Let us now focus on the number of iterations and the value of the two terms in the objective function during the algorithm. The analyse is performed for the global model, on all batches. The figure 5.4 shows the evolution of the objective function for the SNMF1 algorithm using the optimal value for λ ($1e - 5$).

The NMF loss function starts from a very low value, as it is the result of the Brunet NMF algorithm and then increases, while the supervised penalty and the objective function regularly decrease. The minimum for the penalty 1, 2 and the objective function is also shown on the figure through the dots of the same color.

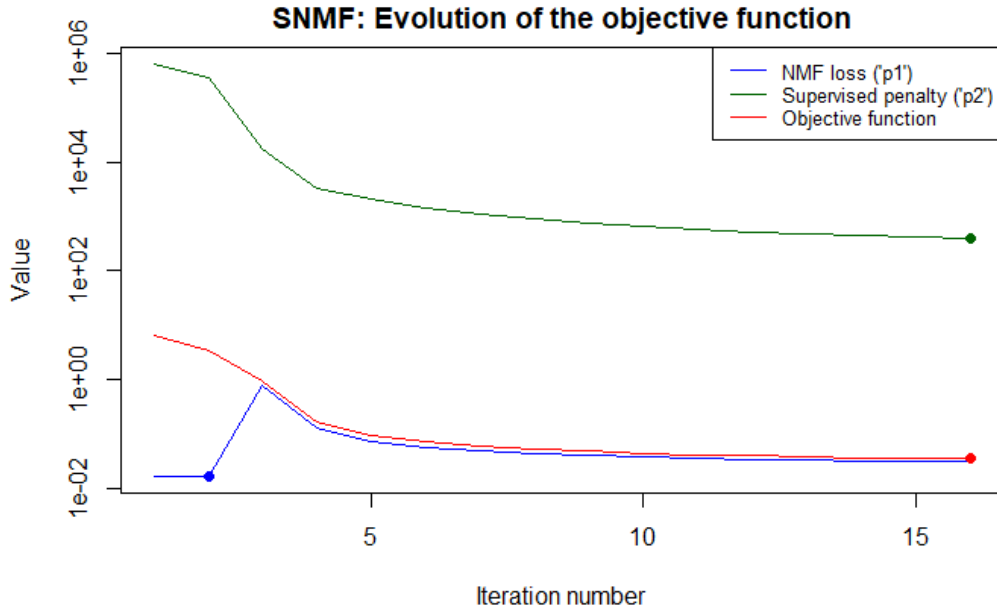


Figure 5.4: Evolution of the objective function for SNMF1 with the optimal λ (global model)

The table 5.2 shows the statistics (value at the first iteration and at the last iteration) for both penalties, the loss function and the number of iterations for the first batch of data. As the SNMF1 starts with a random initialization for B , it is expected that the first value of the penalty 2 is very high. Then, B is adapted with the MU rules such that the penalty 2 is reduced. Yet, the value of the supervised penalty remains high compared to the SNMF2 and SNMF3. Using the "clever" initialization for B for SNMF2 and SNMF3, results in a very low number of iterations. The SNMF2 arrives at a very low value of the supervised penalty using the MU for B , on the other hand SNMF3 reaches a higher value for the same penalty using an OLS update at each iteration.

Thus, the MU rules for B seem to perform better than using an OLS update at each iteration. In addition, it can be seen that 'smartly' initializing the regression coefficients B reduces the number of iterations needed and can yield lower values for 'p2' at the end of the algorithm.

The final value of the loss function is equivalent for SNMF2 and SNMF3 but higher for SNMF1 (both because of a higher NMF cost function and a higher supervised penalty).

	λ	iter	p1.start	p1.end	p2.start	p2.end	loss.start	loss.end
SNMF_1	1e-5	16	0.017	0.031	638538	392.2	6.403	0.035
SNMF_2	1e-4	3	0.017	0.017	164.9	3.467	0.034	0.018
SNMF_3	5e-6	2	0.017	0.017	164.9	29.45	0.018	0.018

Table 5.2: Statistics over the SNMF algorithms for the global model

Remind also that the initialization for U and V is the result of a NMF, so a low number of iterations is expected. For the SNMF2 and SNMF3, the results are thus

expected to be very close to the NMF ones. Increasing the number of iterations by using a different λ value had the effect that the RMSECV for the validation sets were higher. It is already a clue that the number of iterations should remain low in order to optimize the RMSECV, in other words, the algorithm struggles to find an optimal decomposition which differs from the original NMF one.

5.2.3 Data visualization and decomposition

Reconstruction error

The **table 5.3** shows the reconstruction error (%) for the 3 classical methods: PCA, NMF and PLS. Yet, the PCA and the NMF are unsupervised methods while the PLS objective also focuses on a target Y , thus the PLS is obviously expected to do worse than the 2 latter methods.

The PCA with 2 components captures 71.7% (first component) and 21.3% (second component) of the variability in the data. Adding a component would only increases this amount by 1.3%. Thus, 2 components seems a reasonable choice for PCA (although, this choice was already made before the analysis, as explained above).

For PCA and PLS, the individual approach has the smallest error, then the global approach, then the leave-one-batch-out (LoBo) has the biggest. It is what is expected since the individual approach fits a model for each batch and then adjust to the specificity of each. The global approach fits only one model for all batches so the specificity of each batch may be lost. The LoBo approach fits a model on all but one batch and then uses the loadings and the data of this batch to obtain projected scores. Thus the reconstruction error is expected to be higher.

However, for NMF, the LoBo error is lower than the individual and the global one. This could be explained by the very high similarity between the batches. Overall, the PCA, NMF and PLS performs roughly equivalently well. The NMF can be highlighted as it exhibits the smallest LoBo error. The PLS also performs almost as well as PCA although its objective also depends on the prediction of the target concentration C .

	PCA			NMF			PLS		
Batch	Individual	Global	LoBo	Individual	Global	LoBo	Individual	Global	LoBo
1	0.748	0.749	0.757	0.766	0.773	0.756	0.748	0.755	0.757
2	0.718	0.714	0.727	0.735	0.743	0.727	0.718	0.725	0.727
3	0.675	0.669	0.689	0.691	0.703	0.688	0.675	0.685	0.688
4	1.396	1.428	1.411	1.428	1.435	1.404	1.402	1.408	1.411
5	1.352	1.37	1.361	1.384	1.389	1.356	1.355	1.36	1.362
6	1.288	1.295	1.298	1.319	1.325	1.294	1.289	1.296	1.298
mean	1.029	1.037	1.041	1.054	1.061	1.037	1.031	1.038	1.041

Table 5.3: Error percentage per batch and methodology (%)

The **table 5.4** shows the reconstruction error for the 3 SNMF models. SNMF1 shows the worse reconstruction error for the individual and global approach, but its LoBo error is on the other hand roughly equivalent to the other methods. The SNMF2 and SNMF3 are very close in terms of error and it is also expected since they both have a low number of iterations starting with the results of a NMF and taking into account the additional supervised penalty term. However, the SNMF2 and SNMF3 errors compared to the NMF errors are almost equal, meaning that the quality of the reconstruction is not significantly affected by this new supervised term. In contrast, for SNMF1, which has more iterations, this effect is noticeable but only for the individual and the global approach.

SNMF1			SNMF2			SNMF3			
Batch	Individual	Global	LoBo	Individual	Global	LoBo	Individual	Global	LoBo
1	1.027	1.55	0.78	0.77	0.778	0.753	0.765	0.773	0.754
2	1.016	1.566	0.75	0.739	0.746	0.723	0.734	0.741	0.724
3	0.994	1.592	0.711	0.693	0.703	0.684	0.689	0.698	0.685
4	1.686	2.206	1.435	1.428	1.439	1.404	1.423	1.432	1.404
5	1.648	2.193	1.387	1.386	1.391	1.355	1.379	1.384	1.355
6	1.603	2.182	1.327	1.316	1.327	1.292	1.313	1.319	1.292
mean	1.329	1.881	1.065	1.056	1.064	1.035	1.05	1.058	1.036

Table 5.4: Error percentage per batch and methodology (%)

Finally, the errors of each batch are in direct relation with the construction of the observed spectra. Remind that the batch 4, 5, 6 have more variability on the position of their peaks. The reconstruction error for each batch are then probably just reflecting the artificially added error which is independent of the pattern in the data.

Scores and Loadings

Let us now turn to the scores and loadings produced by the methods. The **figure 5.5** shows the true quantities of each component (A, B, C) used to generate the data and the scores for the individual model for the first batch, for the PCA and the PLS with 2 components. The similarity between the two models is striking, as the first component is roughly the same and mimicks the evolution of the product (C), while the second captures the evolution of the intermediate reactant B and probably the reactant A. The second component for PLS is just the symmetric as the PCA one. In both models, the scores are not smooth and then captures a part of the artificial noise added to the observed data.

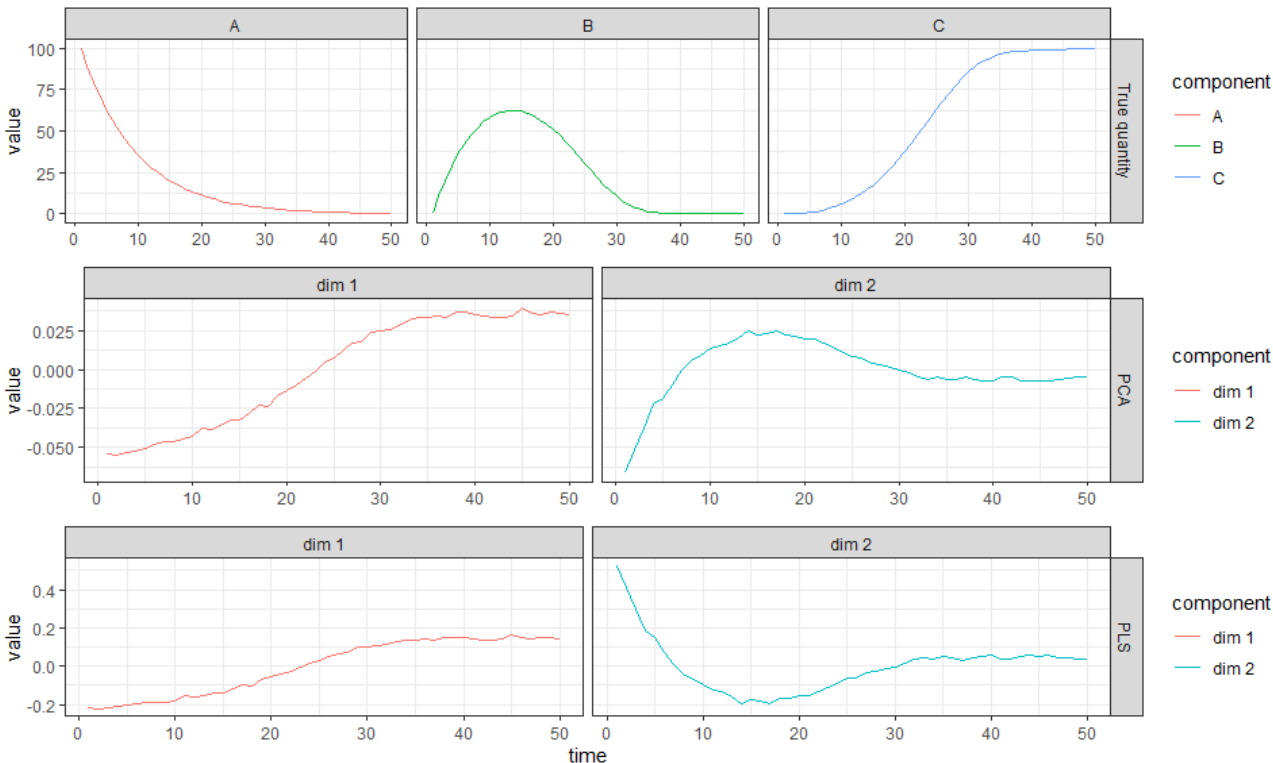


Figure 5.5: Scores for batch 1: individual model

Then, the **figure 5.6** shows the true quantities of each component and the scores for the individual model of the batch 1 for the NMF and the SNMF1 (the one which differs the most from the initial NMF) with 3 components. Here, the NMF almost perfectly

captures each chemical component through each of its dimensions. The SNMF1 also does, but in addition it also gets closer to the true scale of each chemical component, see the difference between the third component for NMF and SNMF1. As for the previous models, noise is also captured by NMF and SNMF1 in the scores.

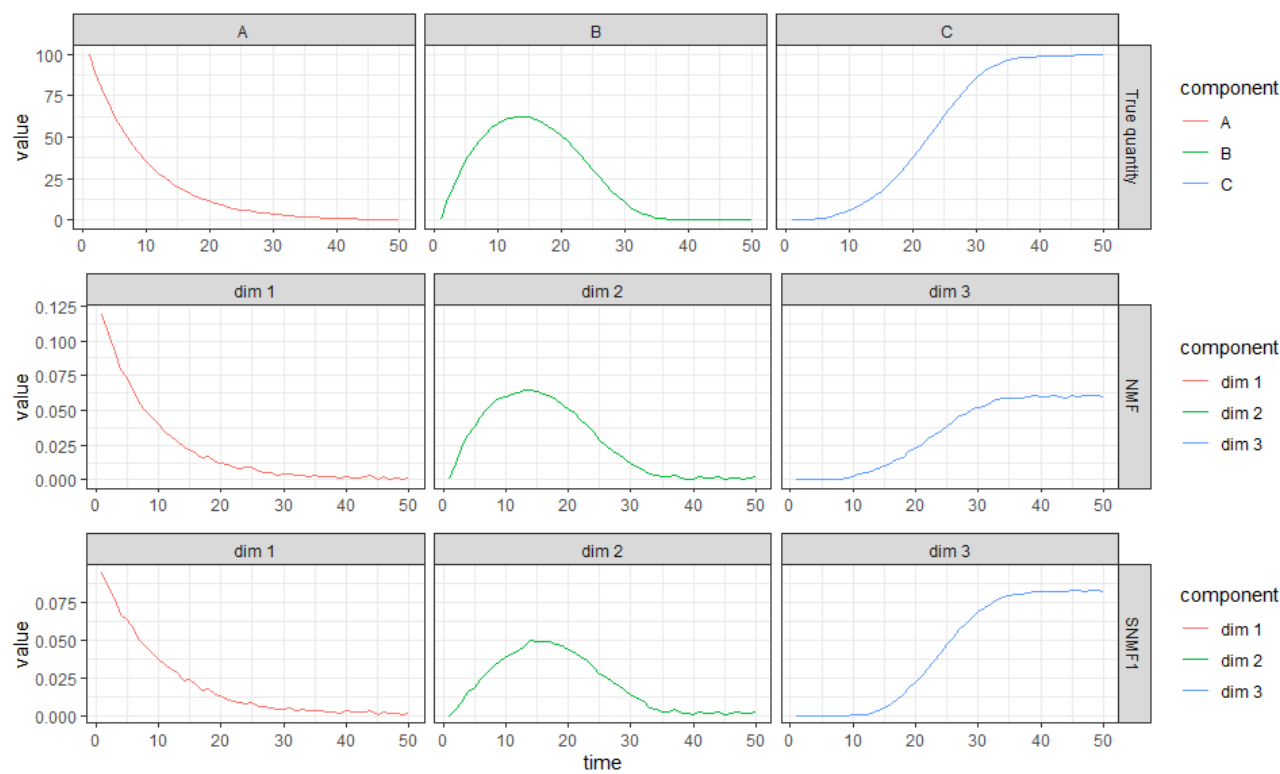


Figure 5.6: Scores for batch 1: individual model

The **figure 5.7** shows the loadings for the individual model for the batch 1. Again, the similarity between the PCA and the PLS is very noticeable: the first component seems the same and the second for PLS is the symmetric of the second for PCA (but the intensities diverge).

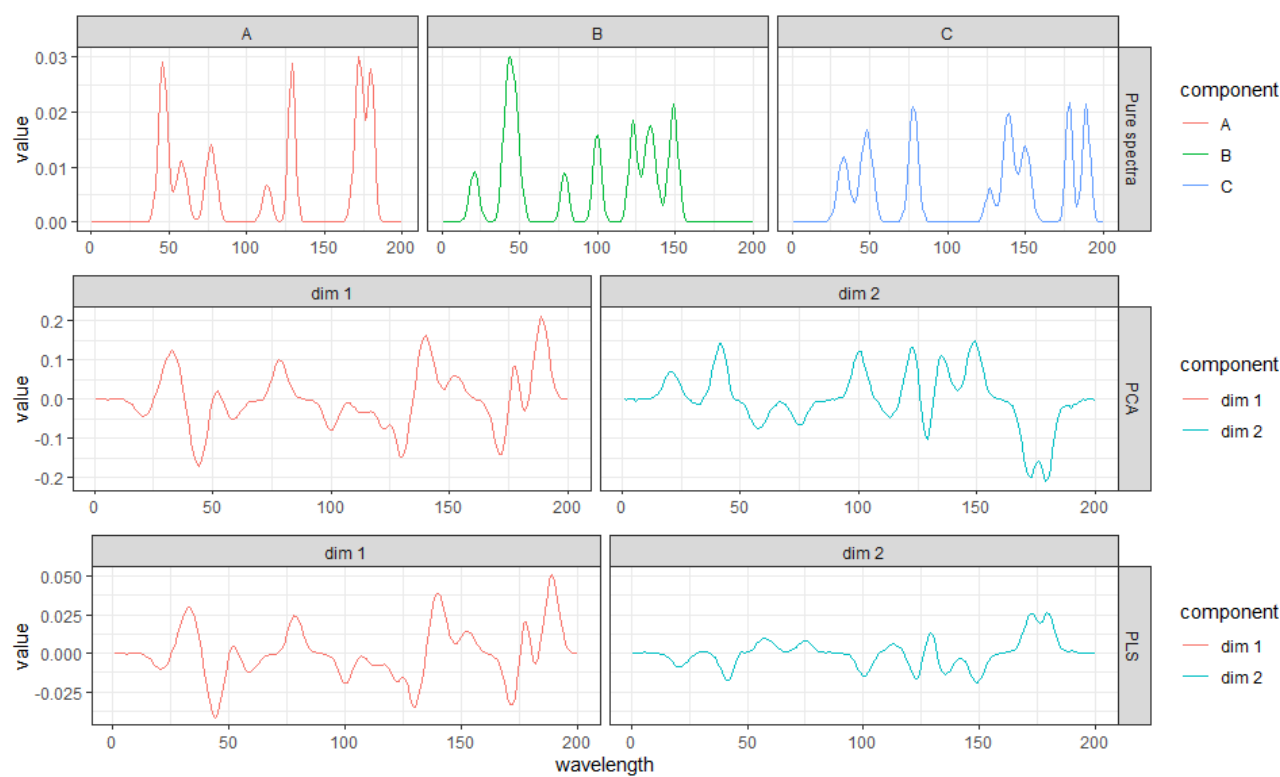


Figure 5.7: Loadings for batch 1: individual model

The **figure 5.8** shows the loadings for the individual model for the batch 1 for NMF and SNMF1. In both cases, the loadings of each component clearly corresponds to peaks of the true spectra. A bit of noise is captured in the loadings for instance in the dimension 1 and 2 of NMF and SNMF1. The scale between the 2 models' loadings differs: the dimension 1 in SNMF1 is closer to the true spectra regarding the intensity of each peak, the dimension 2 intensity in SNMF1 is in contrast too wide, and the dimension 3 in SNMF1 is very close to the true scale of the true spectra of the component C. Thus, as for the scores, there seems to be a scaling effect for the loadings of the SNMF compared to the NMF. However the peaks are the same for both models.

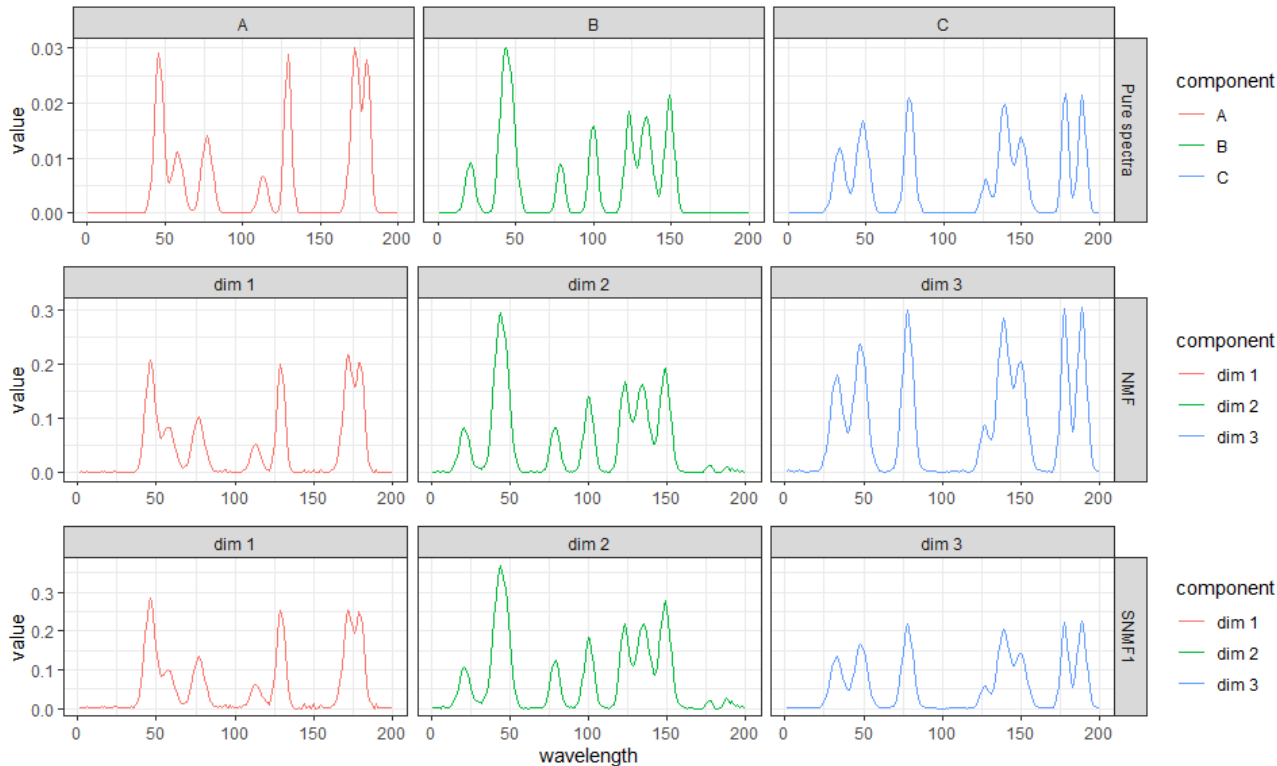


Figure 5.8: Loadings for batch 1: individual model

NMF VS SNMF

The **figure 5.9** shows the difference between the NMF and the SNMF1 results for the individual model for the batch 1. The difference in the loadings is mainly on the intensity of each peak but the peaks identified are the same. The intensity for the dimension 1 and 2 is increased but reduced for the dimension 3 (the dimension representing the response variable). Regarding the scores, an extra black curve represents the evolution of the true quantities (divided by a factor 1200) represented by the matching dimensions. It can be seen that for the dimension 3 (representing the product, the target used for fitting the SNMF algorithm), the SNMF scores is shifted such that it better matches the true quantity of the product. The scores for the dimension 1 and 2 are also (but more slightly) shifted towards the true quantities.

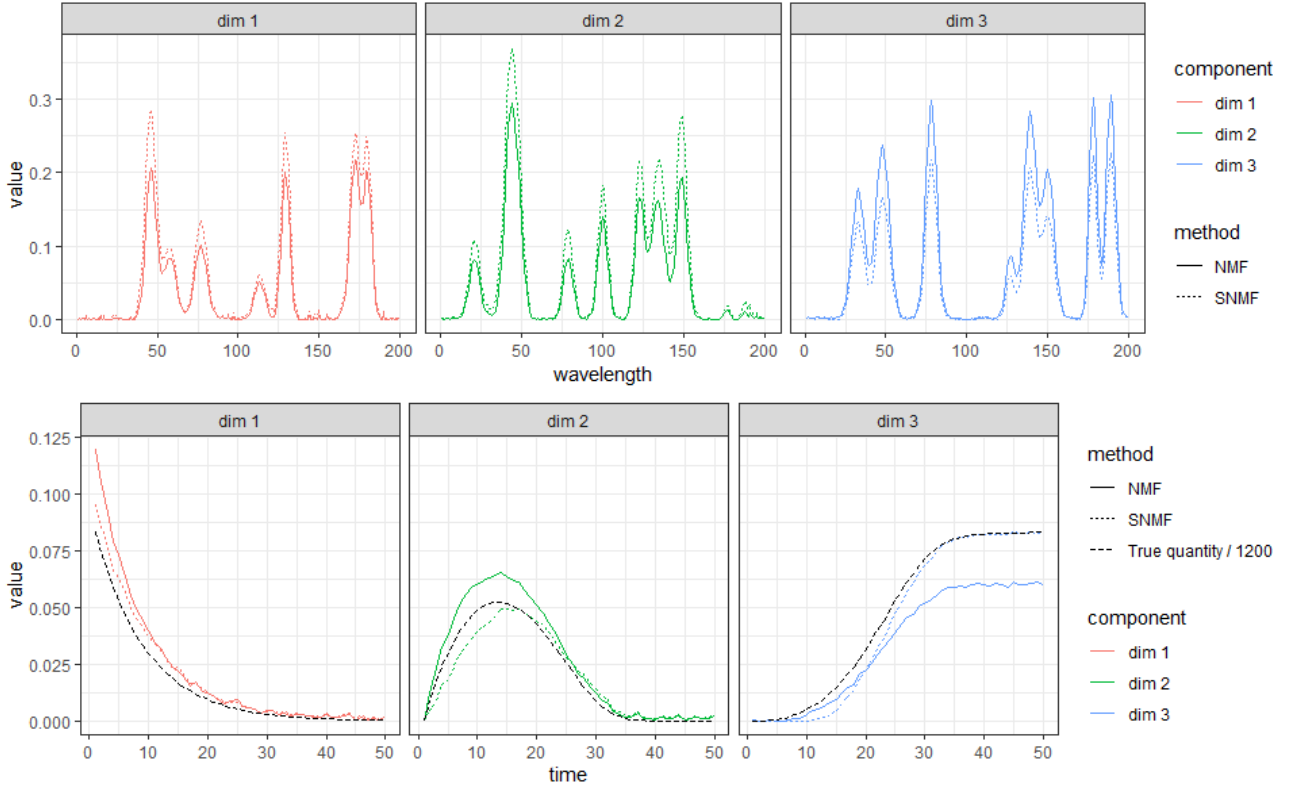


Figure 5.9: SNMF1 versus NMF for batch 1: individual model

5.2.4 Prediction

In this subsection, the prediction accuracy for the product C is investigated. The models used are the one step PLS, the two steps NMF + regression and the two steps SNMF + regression. The RMSECV are compared as well as the final regression coefficients over the variables (the wavelengths).

RMSE

Remind that two different predictions can be produced for the training set for the two steps models: see figure 5.1 and 4.3. Hence, two different RMSE train can be obtained for the two steps methods: using $\hat{C}_{G/i}$ or using $\hat{B}_{G/i}^{2s}$. Note that, in the tables below, each line corresponds to one fitted model. For instance, to predict the batch 1, a model is fitted on the batches 2 to 6 and the RMSE validation is the error of prediction on the batch 1 (written in the tables on the line 'Pred. Batch 1'). On the other hand, the RMSE train is the error of this model when predicting the batches 2 to 6, but the result is also written on the same line as the batch 1 ! So, the line X should be read as “the training and validation RMSE when predicting the batch X”.

The **table 5.5** first shows the RMSECV for the one step PLS and the two steps NMF. For the PLS with 2 components, the validation RMSE is 1.907 and the training RMSE is slightly lower, but not much overfitting is observed. The batches 4 to 6 seem harder to predict: the validation RMSE is higher and the training RMSE is lower, meaning that predicting one of those batches (and hence disregarding it for the training set) yields better training errors. For the NMF + regression, the validation RMSE is higher (2.01) as well as the training RMSE using $\hat{B}_{G/i}^{2s}$ (1.993), again not much overfitting is observed.

The training RMSE using $\hat{C}_{G/i}$ is however very low, much lower than the training RMSE using the coefficients of PLS, but this comes at a price: the coefficients $\hat{C}_{train}$ are optimal (OLS) for the specific training set only, but they are easily interpretable, as one coefficient corresponds to one component. For instance, when predicting the first batch, an NMF is fitted on the batches 2 to 6 and produces scores and loadings. The coefficients $\hat{C}_{G/i}$ (with $\tilde{Y}_{G/i} = U_{G/i}\hat{C}_{G/i}$, see figure 4.3) are 2.448, -5.180, 1451.676 for

the 3 vectors of scores respectively. The target being the product concentration, this indicates that its concentration mainly depends on the third dimension of the scores and is in addition proportional to it.

	PLS		NMF + regression		
<i>Pred. Batch</i>	<i>Train</i>	<i>Valid</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>
1	1.929	1.715	1.066	2.026	1.835
2	1.932	1.675	1.089	2.035	1.779
3	1.932	1.669	1.071	2.043	1.735
4	1.856	2.073	1.054	1.954	2.188
5	1.851	2.091	1.05	1.951	2.204
6	1.838	2.147	1.055	1.943	2.246
mean	1.89	1.907	1.064	1.993	2.01

Table 5.5: One step PLS & Two steps NMF + regression: RMSECV

The **table 5.6** shows the RMSE for the SNMFs models. Among the batches, lower training errors are mostly observed when predicting the batches 4 to 6 (as the preceding table). Training errors using $\hat{C}_{G/i}$ widely differ but are always lower than for the two steps NMF (and this is expected since it is in the objective function). Using the coefficients $\hat{B}_{G/i}^{2s}$, the training and validation errors are a bit higher than for the PLS. However, the two steps SNMF2 and SNMF3 shows better training and validation errors than the two steps NMF.

One also notices the difference between the training errors using $\hat{C}_{G/i}$ for SNMF2 and SNMF3. Indeed, the lowest is achieved for SNMF2, which used MU rules for the coefficients instead of OLS at each iteration like SNMF3. In this sense, the MU rules for the coefficients seem to be more useful for finding optimal training regression coefficients (remind that the MU rule for the scores depends on the regression coefficients, thus the scores might have been adapted in SNMF2 in a different way than in SNMF3 and yield better regression coefficients).

As a conclusion, the two steps NMF is slightly outperformed by the two steps SNMF2 and SNMF3 (with low number of iterations) but not by SNMF1 (full MU with random init for B). None of the models beats the PLS for the training and validation RMSE (using the regression coefficients $\hat{B}_{G/i}^{PLS}$ versus $\hat{B}_{G/i}^{2s}$).

	SNMF1			SNMF2			SNMF3		
<i>Pred. Batch</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>
1	0.966	2.367	2.233	0.079	2.023	1.813	0.361	2.02	1.829
2	0.973	2.329	2.169	0.095	2.032	1.761	0.364	2.03	1.771
3	0.969	2.32	2.183	0.096	2.04	1.723	0.363	2.037	1.731
4	0.963	2.303	2.383	0.081	1.95	2.19	0.357	1.948	2.176
5	0.963	2.288	2.429	0.08	1.948	2.209	0.357	1.946	2.195
6	0.949	2.338	2.59	0.07	1.937	2.242	0.348	1.936	2.244
mean	0.964	2.324	2.336	0.084	1.989	2.002	0.358	1.987	2.003

Table 5.6: Two steps SNMF + regression: RMSECV

Regression coefficients

Finally, the regression coefficients $\hat{B}_{G/i}^{2s}$ or $\hat{B}_{G/i}^{PLS}$ are compared in the **figure 5.10** below. They result from fitting a model on the batches 2 to 6 in order to predict the left aside batch 1. The same peaks appear for the one step PLS and the two steps models, but the intensities of each peak differ. In fact, the PLS coefficients always are equal or below the two steps coefficients. This might be caused by the additional constraint of nonnegativity of the scores and loadings they were derived from in the case of the NMF and SNMF. Among the SNMFs models, the SNMF1 differs a bit from the two others but the SNMF1, SNMF2 and NMF coefficients are very much alike (and it is expected since the low number of iterations of SNMF1 and SNMF2).

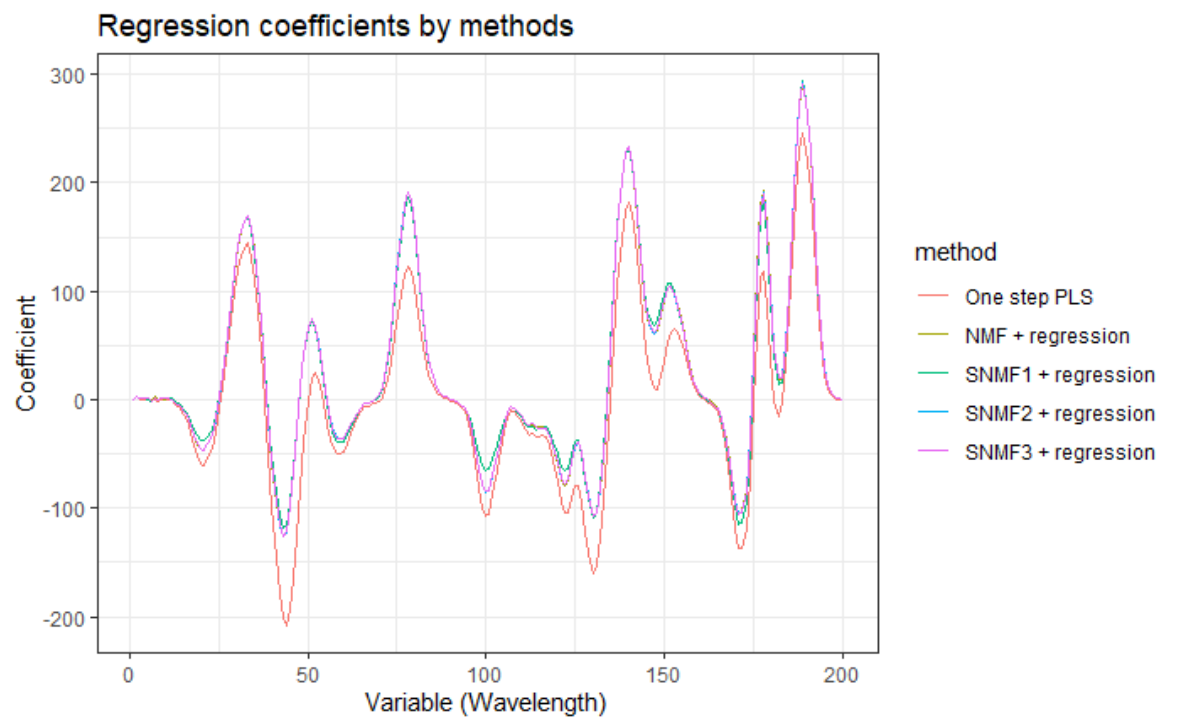


Figure 5.10: Regression coefficients on the variables for the different methods

5.3 Janssen dataset

For confidentiality reasons, this section is only available under request. Please contact Bernadette Govaerts: bernadette.govaerts@uclouvain.be for more information.

5.4 Chemium dataset

As explained in the second chapter, the Chemium dataset has the particularity to result from a flow reaction. From the specific design of the data collection, 29 sets of 6 spectra are gathered and each spectrum of the same set has the same target values (A, C, D). For the purpose of our case study, only C (the product) will be predicted.

The dataset has 174 observations (6×29) and 778 variables, ranging from 4003 to 9997 cm^{-1} . In a previous extensive research lead by Alexis Rousseaux as part of his master thesis in chemical engineering (Rousseaux and Nagy, 2021), the spectral window was filtered to exclude the part where the spectra saturate (see figures 5.11 and 5.12). This is the only pretreatment which is applied in this case study. Thus, the dataset has now 616 variables (this is almost triple the previous case).

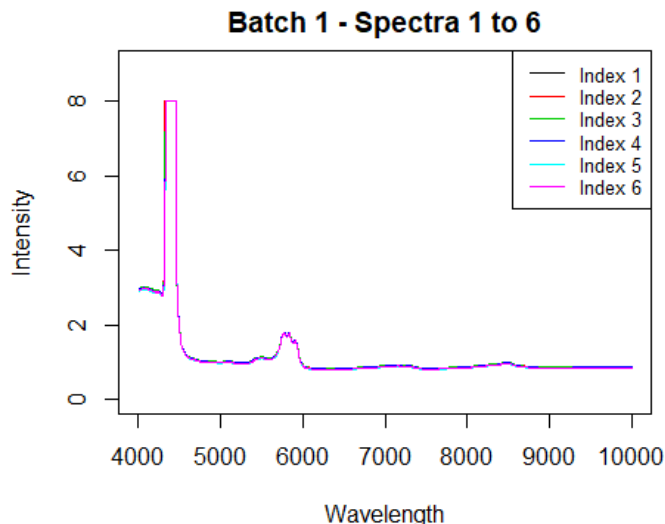


Figure 5.11: Unfiltered spectra for batch 1

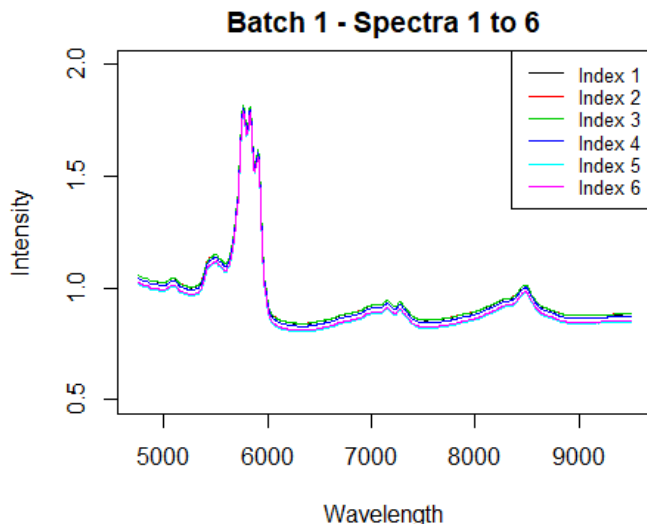


Figure 5.12: filtered spectra for batch 1

The number of components is chosen as 3, as 3 chemical components are present in the samples (yet the second and third in much lower proportions than the product). In addition, one notes that a PLS with 3 components is optimal in a leave-one-batch-out cross-validation strategy (but remind that $r - 1 = 2$ components will be used for PLS and PCA in order to compare with NMF and SNMF).

5.4.1 SNMF parameters

As usual, 3 SNMF algorithms will be tested (SNMF1, SNMF2 and SNMF3: see table 5.1). A grid search is used to find the optimal λ parameters balancing the two terms in the objective function. For each λ value, the mean RMSECV is measured in a leave-one-batch-out cross-validation using the two steps SNMF + regression. The optimal values are 0.01, 0.6 and 0.5 for SNMF1, SNMF2 and SNMF3, respectively. The ϵ (threshold for the termination) is maintained at 0.001, the maximum number of iterations is 100, and the initialization for the scores and loadings is the result of a NMF with the Brunet algorithm. The multiplicative update rules are used for U and V .

5.4.2 Analyze of the SNMF algorithms

Using the above parameters for each SNMF algorithm, a SNMF is fitted on the whole dataset of spectra and using the product as the target. The **table 5.7** is obtained. One immediately sees that none of the algorithms match the termination threshold and thus iterates 100 times. The question of convergence thus arises.

Although the λ parameter is quite high compared to the previous case studies (hence, high importance on the supervised penalty), the value for the NMF loss function ('p1')

decreases for the three SNMF algorithms. As previous, the values for the second penalty at the end of the algorithms are larger for SNMF1 as SNMF2 and SNMF3 used a smart initialization for their regression coefficients (OLS for $Y \sim U_{init}$). In addition, SNMF3 performed an OLS regression to update the regression coefficients B at each iteration, therefore it is not suprising that the penalty 2 is minimal for SNMF3. Finally, the last value of the objective function is the highest for SNMF1 although the λ is the smallest (0.01).

	λ	iter	p1.start	p1.end	p2.start	p2.end	loss.start	loss.end
SNMF1	0.01	100	1.8	1.7	27611	323	278	5
SNMF2	0.6	100	1.8	1.5	329	0.00022	199	1.5
SNMF3	0.5	100	1.8	1.4	329	2e-06	166	1.4

Table 5.7: Statistics over the SNMF algorithms for the global model

The **figures 5.13, 5.14 and 5.15** show the evolution of the parts of objective function for each algorithm. For SNMF1, there is a smooth (slow) decay for the first penalty (not seen in this graph because of the log scale). The decay for the second penalty is huge at the beginning of the algorithm because a bit a time is needed to find reasonable regression coefficient B , then the decay slows down. For SNMF2, both penalties widely increase during the first iterations ! Then, they both decrease to reach lower values than at the first iteration. Turning now to SNMF3, jumps of values are observed and finally yield lower values for both terms in the objective function. The jumps are not surprising since the regression coefficients B are obtained by doing an OLS at each iteration, thus there might be drastic changes between iterations (see the next paragraph for further analysis). An encouraging fact is that the minimal values for the parts of objective function are given at the last iteration (the dots in the plots) for the three algorithms.

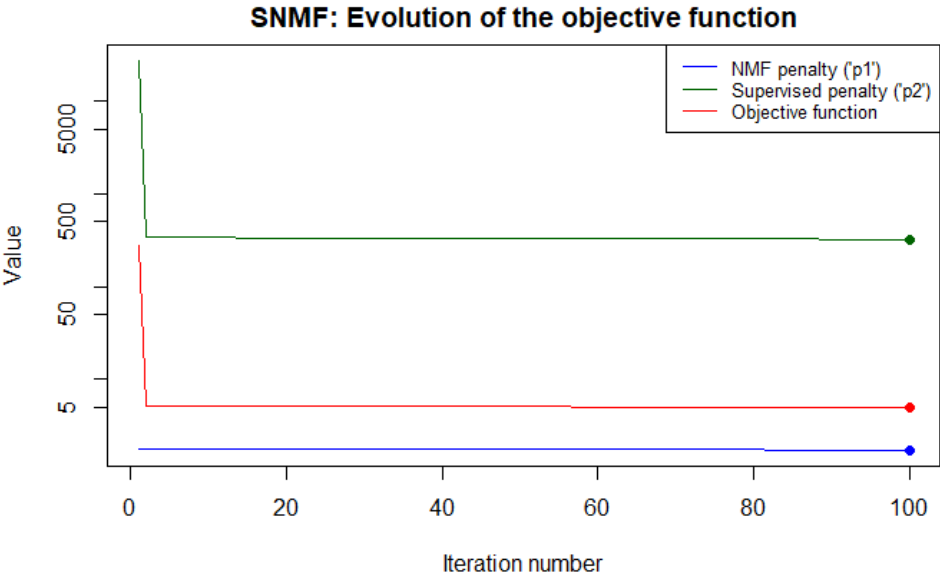


Figure 5.13: Evolution of the objective function for SNMF1 (global model)

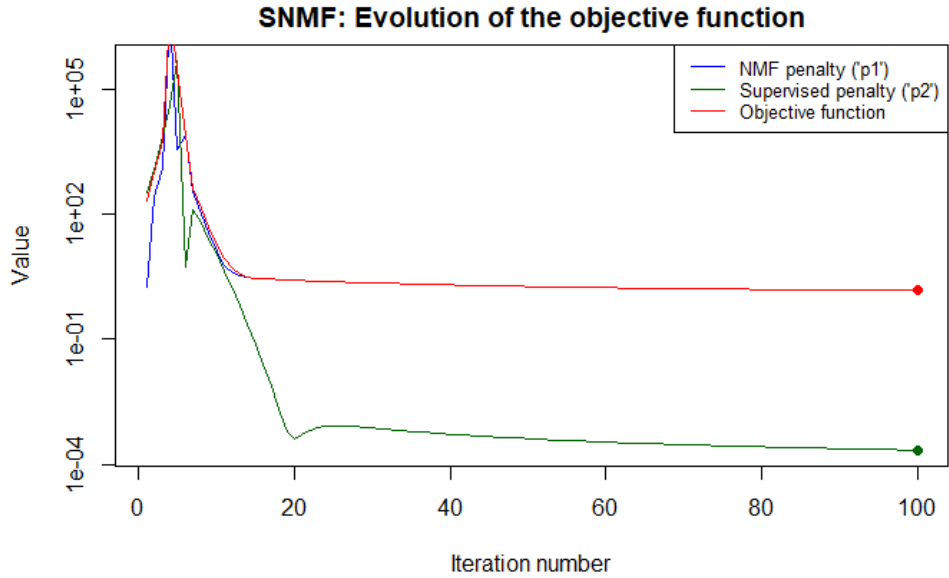


Figure 5.14: Evolution of the objective function for SNMF2 (global model)

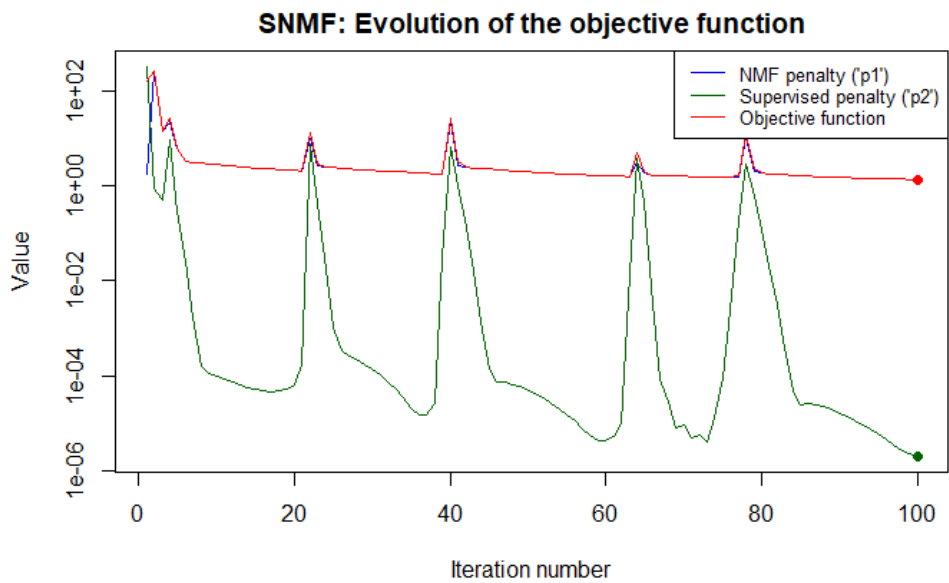


Figure 5.15: Evolution of the objective function for SNMF3 (global model)

After observing the successive plots of scores, loadings and regression coefficients at each iteration around the iteration 15 to 30 for SNMF3, a number of observations can be made. First, there is a jump in the values of the regression coefficients at the iteration 21. Until the iteration 20 and after the iteration 22 (included), the weights follow a smooth path. The loadings remain steady during the algorithm, only the scores vary. Each dimension of the scores seem to form one smooth line (when ordering the scores by the respective index of their spectra), but at the iteration 21, a very short number of values jump out of it. As the weights are updated after the scores, an OLS regression needs to shift its coefficients such that the MSE is minimized. During the subsequent iterations, the extreme values for the scores are progressively brought back such that each dimension form a smooth line, and the regression coefficients are also slowly adapted.

5.4.3 Data visualization and decomposition

Reconstruction error

The **table 5.8** presents the reconstruction error for the three benchmark methods: PCA and PLS (2 components both) and NMF (3 components). The full table with the 29 batches can be found in the appendix (table 7.1). Note also that the individual

approach for PLS is missing because it is not allowed by the package `pls` to fit on less than 7 observations (here: 6 observations). The first finding is that PCA and PLS perform equally well for the global and LoBo approaches, and the individual approach for PCA results in a very low mean reconstruction error ($3e - 7$). On the other hand, the global and LoBo approaches for NMF produce low reconstruction errors (0.0035 and 0.0017) but it is still roughly 100 times more than for PCA and PLS ! The same applies for the individual approach of NMF ($7.7e - 5$) compared to PCA ($3e - 7$).

	PCA - 2 comps			NMF - 3 comps			PLS - 2 comps	
	Indiv	Global	LoBo	Indiv	Global	LoBo	Global	LoBo
1	6e-07	7.87e-05	0.0002157	0.00093	0.015	0.016	8e-05	0.00022
2	3.6e-06	8.04e-05	0.0001069	0.00039	0.012	0.0081	8.4e-05	0.00011
3	5e-07	3.94e-05	5.87e-05	4.4e-05	0.0049	0.0047	4e-05	5.9e-05
...	...	...	...	...	...	...	...	...
27	1e-07	3.4e-06	3.9e-06	5e-07	0.0026	5.2e-05	3.4e-06	3.9e-06
28	2e-07	7.8e-06	9.3e-06	5e-07	0.0031	0.00058	7.8e-06	9.3e-06
29	1e-07	7.6e-06	9.1e-06	5e-07	0.0034	0.0035	7.6e-06	9.1e-06
mean	3e-07	1.91e-05	2.83e-05	7.7e-05	0.0035	0.0017	2e-05	2.8e-05

Table 5.8: Error percentage per batch and methodology (%)

Regarding the SNMF algorithms, the **table 5.9** shows their respective reconstruction errors. The full table with the 29 batches can again be found in the appendix (table 7.2). It can be seen that there is an extreme value for the individual model for SNMF2, causing the mean individual error to be much higher. Apart from that, the mean errors for SNMFs are of the same order of magnitude, and slightly larger than the NMF results for the global and LoBo approaches. On the other hand, the error for the individual approach is deteriorated when going from NMF to SNMF. The SNMF2 performs the closest to NMF for the LoBo approach.

	SNMF1			SNMF2			SNMF3		
	Indiv	Global	LoBo	Indiv	Global	LoBo	Indiv	Global	LoBo
1	0.00069	0.017	0.017	0.0018	0.014	0.00034	0.00089	0.009	0.017
2	0.00051	0.013	0.013	84	0.011	0.00035	0.00058	0.0076	0.013
3	5.2e-05	0.0057	0.0059	0.00047	0.0036	0.00018	0.00013	0.0024	0.0058
...	...	...	...	...	...	...	...	...	...
27	2.4e-05	0.0024	0.002	1.6e-06	0.0023	0.002	0.00027	0.0025	0.0022
28	2.4e-05	0.0036	0.0025	2.5e-06	0.0029	2e-05	2.5e-06	0.0032	0.0027
29	2.4e-05	0.0031	0.0026	1.4e-06	0.0031	8.1e-06	1.7e-06	0.0033	0.003
mean	0.00011	0.0036	0.0033	2.9	0.0031	0.0019	0.00027	0.0043	0.0036

Table 5.9: Error percentage per batch and methodology (%)

Scores and Loadings

As the data comes from a flow reaction, there is no evolution of the chemical concentrations for a given batch. Hence, looking at the scores from the individual models is not meaningful (and even not possible for PLS). This section thus looks at the scores and loadings for the global model, and investigates whether it is possible to distinguish the batches of data and obtain smooth, pure spectra like, loadings. First the scores for PCA, PLS, NMF and SNMF2 are shown. Then, their loadings are compared.

The PCA scores and PLS scores are shown in the **figures 5.16 and 5.17**. A zoom of both of these figures can be found in the appendix (figures 7.1 and 7.2). Both scores

show very similar arrangements, but different (for instance, the batch 7 and 11). In addition, the range of values taken by the scores for PCA and PLS also differ. The batches of data are clearly grouped together for both methods (see the zoom for the upper part of both figures in the appendix). This is encouraging since it is possible to well separate the groups of spectra even using an unsupervised dimensionality reduction technique such as PCA. The evolution in time of the scores of each batch can also be seen: the first batches are well distinguished but then the batches all grouped together.

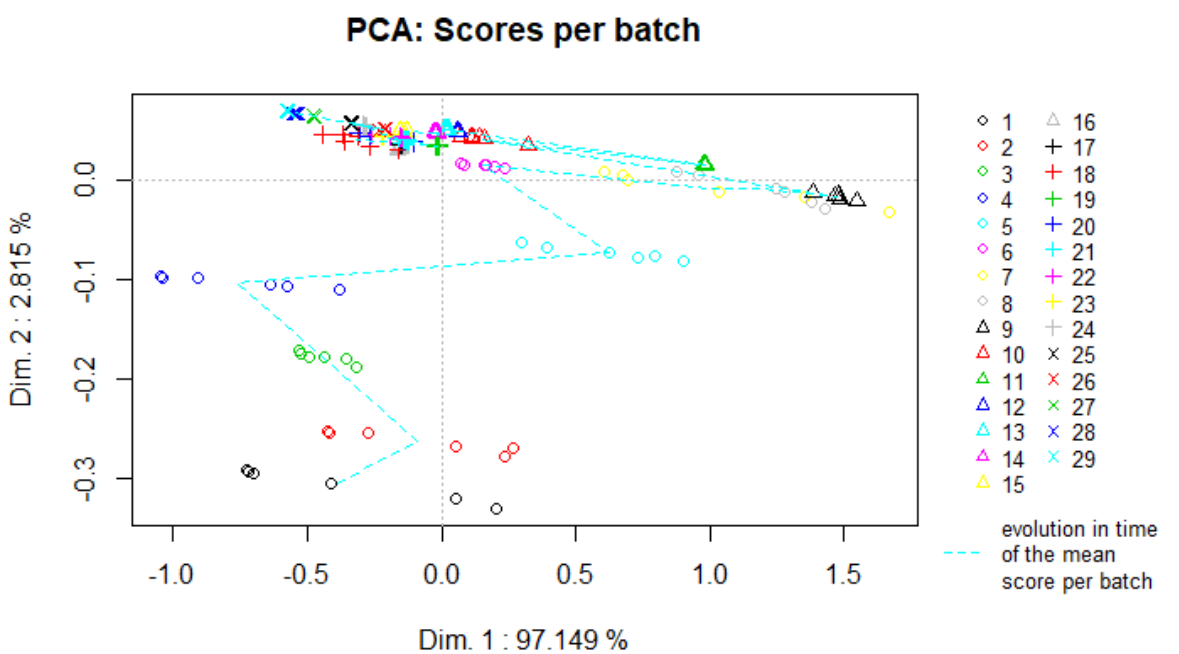


Figure 5.16: PCA scores for the global model

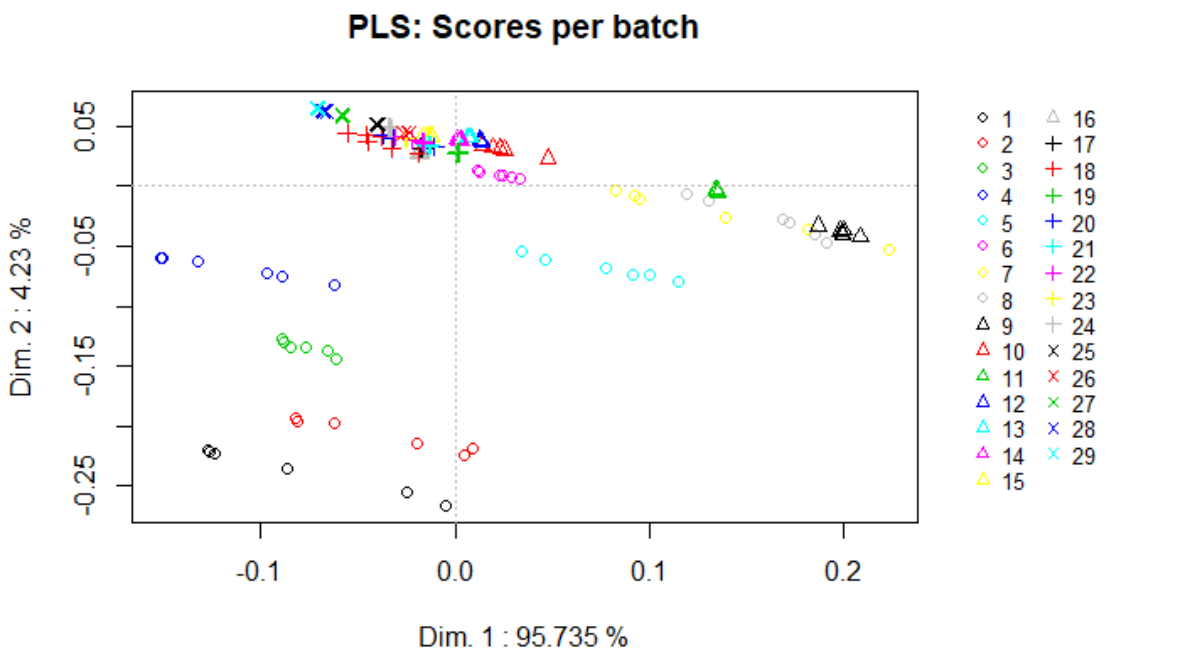


Figure 5.17: PLS scores for the global model

The NMF and the SNMF2 scores are then shown in the **figures 5.18, 5.19 and 5.20, 5.21**. The SNMF2 is chosen because it is the algorithm which has the smallest mean validation RMSE (see below). Remind that 3 dimensions are now used, the figures plotting the dimension 2 against the first one and the dimension 3 against the second

one are then represented. The scores for the NMF do not show a well separation as for PLS or PCA, only some sets of spectra break away from the “crowd”. Comparing to SNMF2, one observes that the scores have been adapted such that the separation is now a bit clearer for both figures (dim1 versus dim2, and dim2 versus dim3). From the sole point of view of the scores for the global model with the purpose of clustering the batches, the SNMF2 seems to add value to the dimensionality reduction.

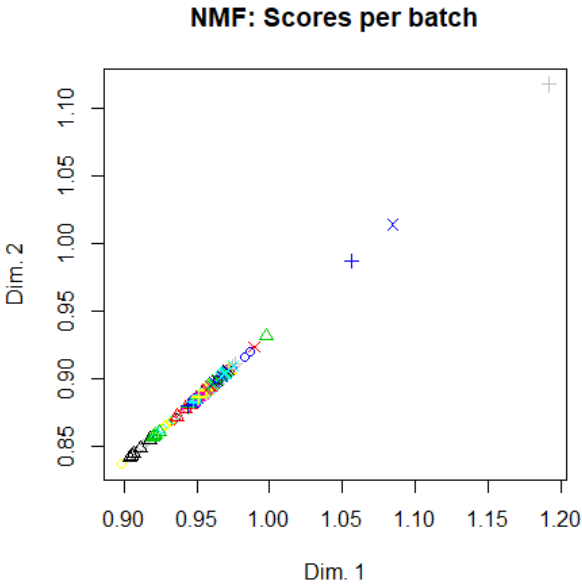


Figure 5.18: NMF scores for the global model

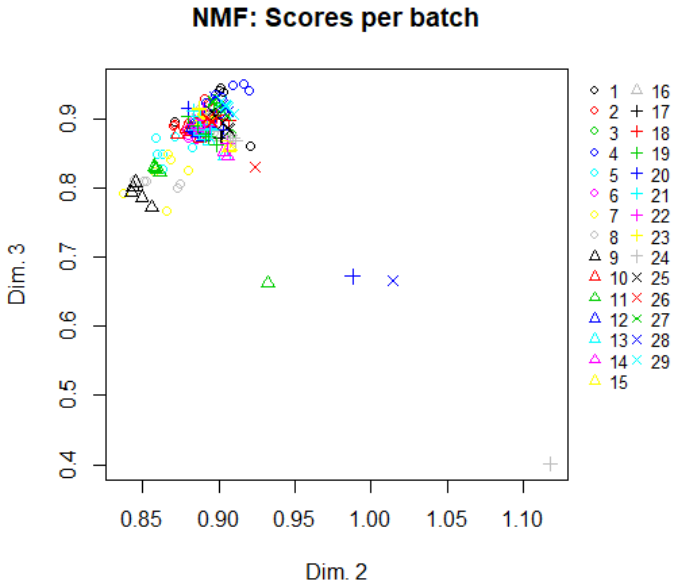


Figure 5.19: NMF scores for the global model

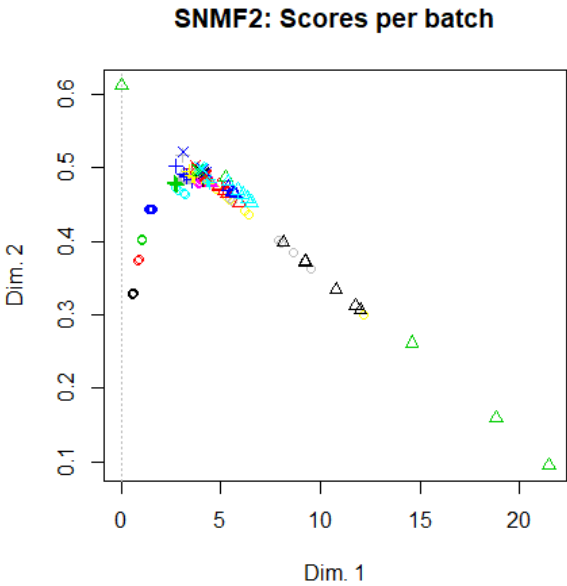


Figure 5.20: SNMF2 scores for the global model

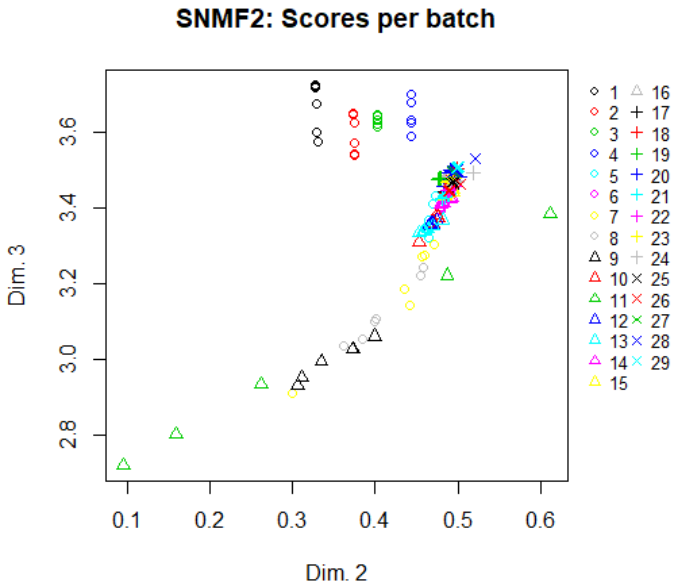


Figure 5.21: SNMF2 scores for the global model

To refine the analysis, the **figures 5.22 and 5.23** show the evolution of the scores sorted by index, which also represents the time. The dimension of the scores for the NMF all have the same order of magnitude and exhibits some peaks for some spectra. On the other hand for the SNMF2, the first dimension takes very high values around the spectra number 50. The dimension 2 has steady values slightly above 0 while the dimension 3 is just below 5. The variations of these two dimensions are much lower than for the first one.

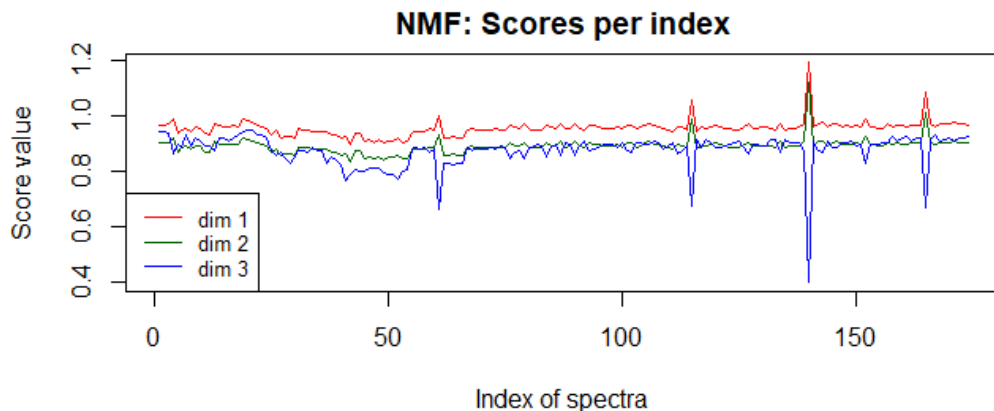


Figure 5.22: NMF scores per index for the global model

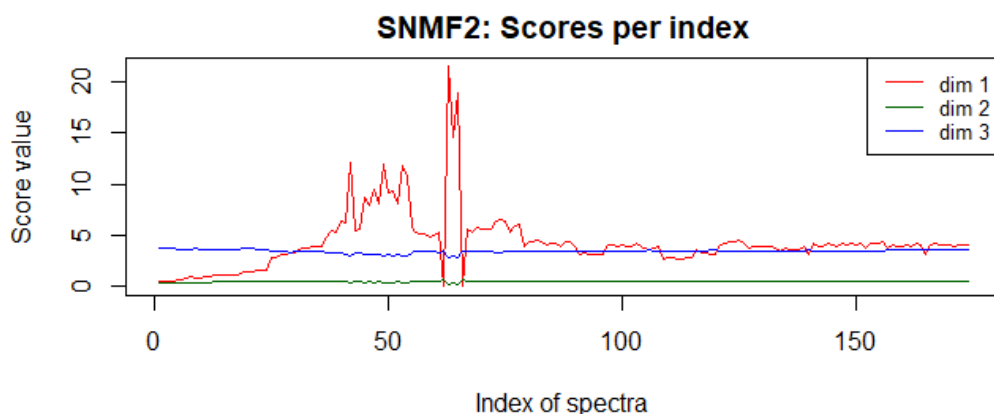


Figure 5.23: SNMF2 scores per index for the global model

Let us now turn to the loadings for the above methods (still for the global approach). The **figures 5.24 and 5.25** compares them. The PCA and PLS loadings are closely related for the dimension 2, only the dimension 1 differs in the intensity of its peaks, but the same peaks are found. On the other hand, the loadings for NMF are noisy and highly not smooth, only the dimension 3 is relatively smooth. An NMF with 2 components was also tested but gave the same noisy dimensions as the one in the figure 5.24. In addition for NMF, the peaks identified in the loadings of PCA and PLS are not retrieved.

Regarding the SNMF loadings, the SNMF1 and SNMF3 are similar to the NMF loadings, but the SNMF2 shows smooth loadings curves for the dimension 1 and 3. The dimension 3 seems to mimic the shape of spectra in the data, but some tiny peaks exist in the first dimension. The intensity for the noise in the dimension 2 is also reduced. One should keep in mind that the loadings for PLS and PCA cannot be obtained for NMF because they contain several positive and negative peaks. Starting from the scores and loadings of an NMF, it is encouraging to observe that SNMF2 is able to smooth V , at least for some of the dimensions.



Figure 5.24: Loadings for the global model

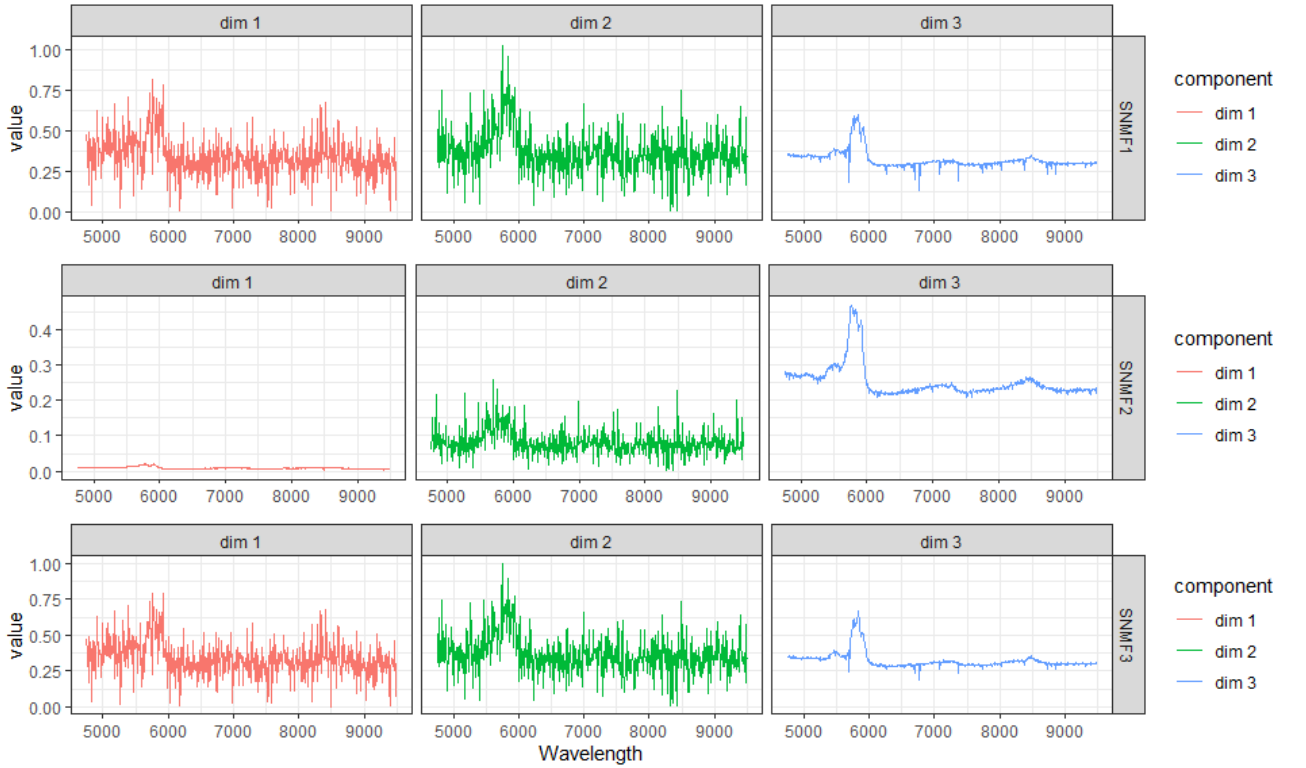


Figure 5.25: Loadings for the global model

5.4.4 Prediction

Now, the focus is placed on the assessment of the prediction's accuracy for the product. First, a comparison between the KFold and the LoBo cross-validation is presented, highlighting the amount of difference in the optimal parameter resulting from the two. Then, the RMSE in LoBo cross-validation are obtained for each usual method. The regression coefficients $\hat{B}_{G/i}^{2s}$ or $\hat{B}_{G/i}^{PLS}$ are also compared. Finally, the leave-one-batch-out predictions and the actual values are visually shown for all spectra in the dataset.

KFold vs LoBo

The use of the particular leave-one-batch-out cross-validation can be highlighted with the striking example below. Let us fit a PLS on the filtered spectra to predict the mass concentration of the product C. One then wishes to find the optimal number of components to fit the model, such that a new unseen set of spectra will yield a good prediction using the previously fitted model.

First, a simple KFold cross-validation is used (with 10 folds), measuring the mean RMSE on the validation and training sets for each number of component. The figure 5.26 is obtained. The optimal number of component is the one which minimizes the mean RMSE on the validation sets (the RMSECV). In this case, this optimal number would be 21.

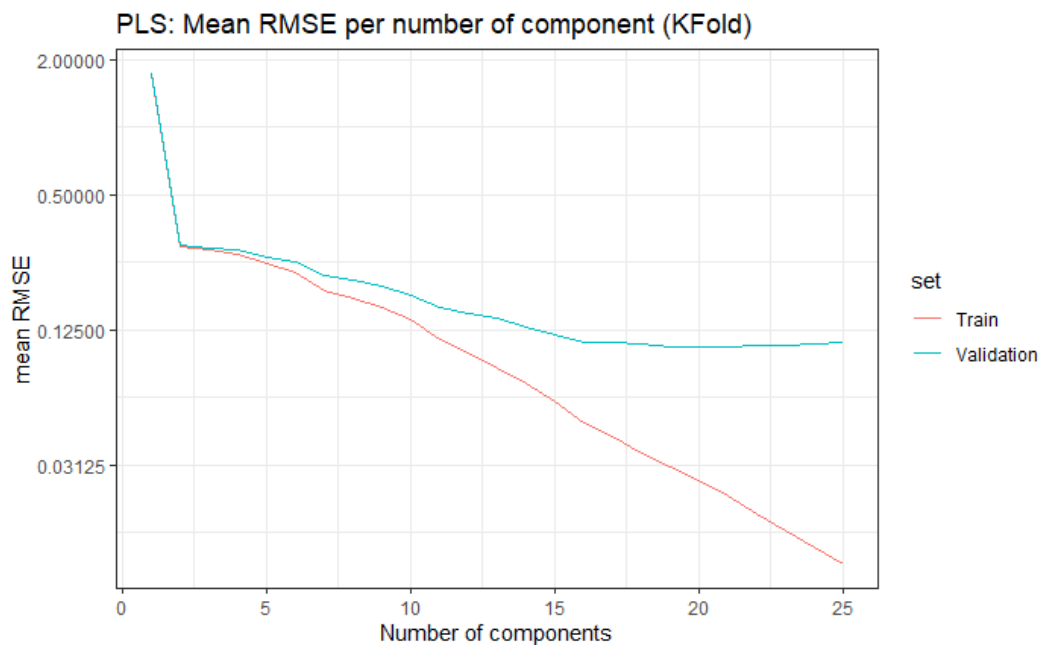


Figure 5.26: PLS - mean RMSE in KFold (10) cross-validation

Then, the leave-one-batch-out cross-validation is used, and the mean RMSE on the training and validation sets are also measured for each number of component (see the figure 5.27). Now, the optimal is widely different and is only 3.

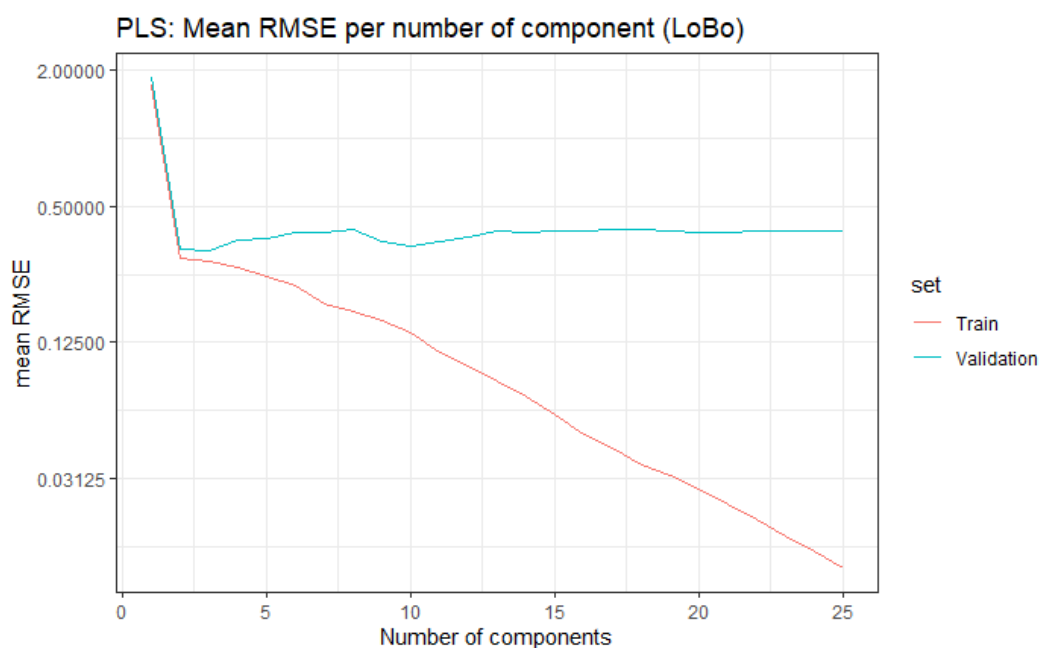


Figure 5.27: PLS - mean RMSE in leave-one-batch-out cross-validation

This example demonstrates the dependence between the spectra in the same set of

6. Indeed, if the spectra (the observations) were independent, the KFold and the LoBo cross-validation would produce roughly the same results. Here, one observes that using KFold (and thus predicting the product for a spectra using a model which was fitted on some of the spectra from the same set) leads to overfitting, which would be similarly observed if a model were evaluated on the same set it was fitted on.

RMSE

The **tables 5.10 and 5.11** show the RMSE for the product concentration. The full tables can be found in the appendix (tables 7.3 and 7.4). As already stated, for the two steps models (S)NMF + regression, the training RMSE can be obtained using $\hat{C}_{G/i}$ regression coefficients applied to the scores $U_{G/i}$. The training RMSE can also be obtained using $\hat{B}_{G/i}^{2s}$ on $X_{G/i}$. The validation RMSE can only be obtained by using $\hat{B}_{G/i}^{2s}$ on X_i .

In this case study, the difference between the performance of the PLS (2 components) and the two steps (S)NMF (3 components) is huge. For the two steps NMF, one notes that the mean training RMSE using $\hat{C}_{G/i}$ is larger than using the regression coefficients $\hat{B}_{G/i}^{2s}$. This was never the case in the previous case studies and is quite surprising.

	PLS		NMF		
<i>Pred. Batch</i>	<i>Train</i>	<i>Valid</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>
1	0.297	0.411	0.311	0.814	0.733
2	0.296	0.351	1.805	1.756	5.288
3	0.3	0.139	1.497	0.411	0.116
...	...	...	...	...	...
27	0.301	0.052	2.003	0.558	0.045
28	0.298	0.257	1.57	0.611	0.686
29	0.298	0.231	1.675	0.476	0.698
mean	0.296	0.321	1.689	0.939	1.115

Table 5.10: One step PLS & Two steps NMF + regression: RMSE

	SNMF1			SNMF2			SNMF3		
<i>Pred. Batch</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>
1	0.101	1.047	4.85	1.113	0.717	2.125	1.09	1.443	6.563
2	0.1	1.211	3.327	1.14	0.61	2.129	0.602	1.682	4.917
3	0.102	1.287	2.859	0.1	0.635	0.195	0.005	1.784	3.568
...	...	...	...	...	...	...	...	...	...
27	0.103	1.374	0.547	0.004	1.039	0.056	0.007	1.717	1.161
28	0.102	1.366	0.741	0.638	0.436	0.077	0.003	1.693	1.613
29	0.103	1.366	0.678	1.414	0.558	0.706	0.103	1.758	1.397
mean	0.105	1.326	1.384	0.546	1.525	1.117	0.26	1.668	1.949

Table 5.11: Two steps SNMF + regression: RMSE

None of the two steps SNMF algorithms beat the two steps NMF in terms of mean RMSE on the validation set (RMSECV, 1.115). Nevertheless, SNMF2 obtains a similar mean RMSECV (1.117). Using the SNMF algorithms clearly leads to a reduction of the training RMSE using $\hat{C}_{G/i}$ compared to a NMF (which mean training RMSE was 1.689). On the other hand, the training performance using the regression coefficients $\hat{B}_{G/i}^{2s}$ is not improved. Finally, another surprising fact is the mean training RMSE for SNMF2 (using $\hat{B}_{G/i}^{2s}$: 1.525) being larger than the mean validation RMSE. Clearly, some results in both tables are not expected. Below are compared the regression coefficients, which will help to understand these figures.

Coefficients

Now, the **figures 5.28 and 5.29** show the regression coefficients $\hat{B}_{G/i}^{2s}$ obtained when fitting on all but the first batch to predict this first batch. These coefficients for NMF, SNMF1 and SNMF3 are plotted together as they all are very noisy with large values for the coefficients of each wavelength. This is particularly true for NMF, less for SNMF1 and even less for SNMF3. These coefficients leave room for little to no interpretation and are expected to hazardously generalize. On the other hand, the SNMF algorithms contributed to reduce the intensities of the regression coefficients compared to the initial NMF ones, but it is still not enough.

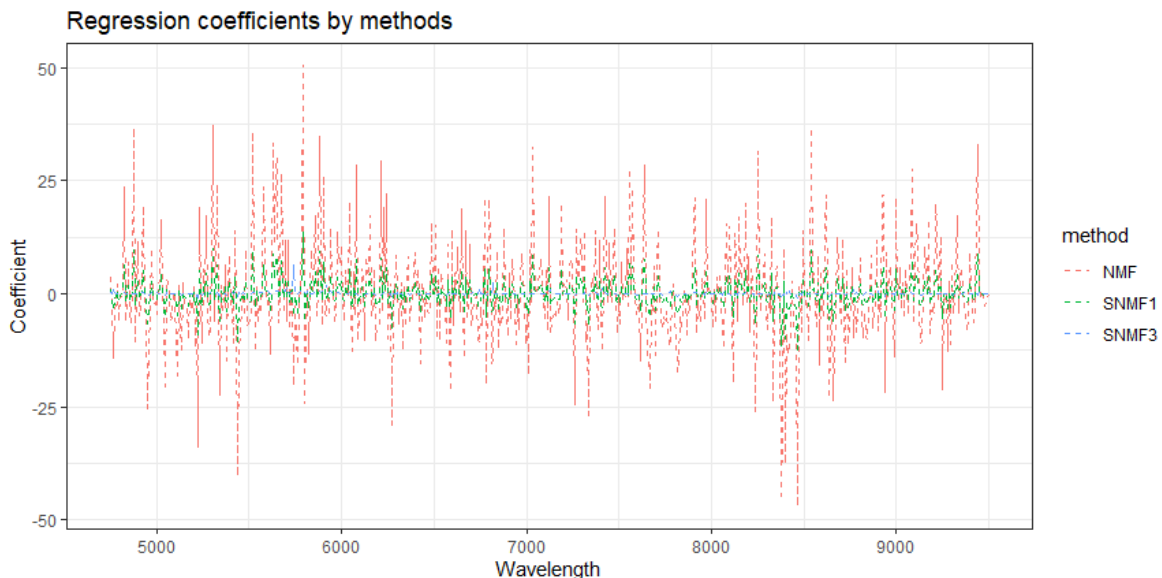


Figure 5.28: Regression coefficients for the different methods

The second figure shows the one step PLS and the two steps SNMF2 + regression coefficients. Now, the similarity between both is very important, and the peaks are roughly identical, with variations mainly on the intensity on each wavelength. On the other hand, there is still a wide gap between the two models when predicting a new unseen set of spectra, and even the two steps NMF performs slightly better than the two steps SNMF2. Thus, having smooth regression coefficients does not imply a significantly better RMSECV, but it would probably be unwise to use the two steps NMF coefficients simply because the RMSECV is slightly higher. It is also really encouraging to observe that the SNMF2 produces smooth and similar regression coefficients as PLS even though the coefficients derived from a simple NMF were very noisy.

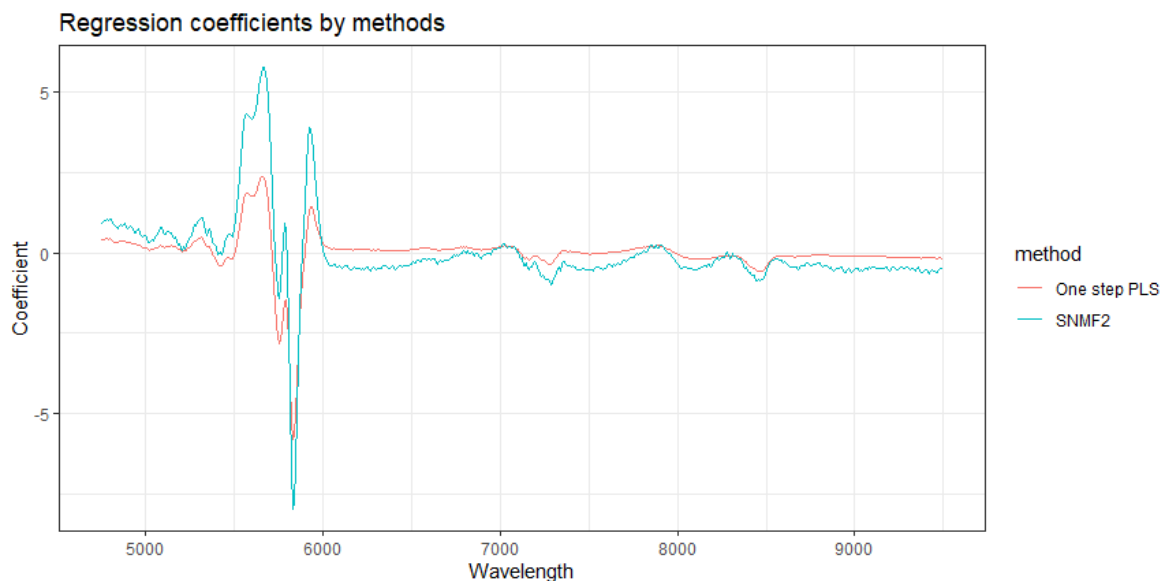


Figure 5.29: Regression coefficients for the different methods

Prediction plot

Finally, the prediction power of each model is visually compared below. For the 29 batches of spectra, the leave-one-batch-out prediction is computed and plotted (for the batch 1, the prediction results from the model fitted on all but the first batch, and so forth). The **figures 5.30 and 5.31** are obtained.

In the **figure 5.30**, the superiority of the one step PLS is clearly seen compared to the two steps NMF. In particular, the first batches which have very low values for the target are correctly predicted, while the two steps NMF completely fails when predicting the batch 2. In addition, the predictions for a given batch is not always consistent when using the two steps NMF (for instance, the predictions for the first batch are very different from one to the other).

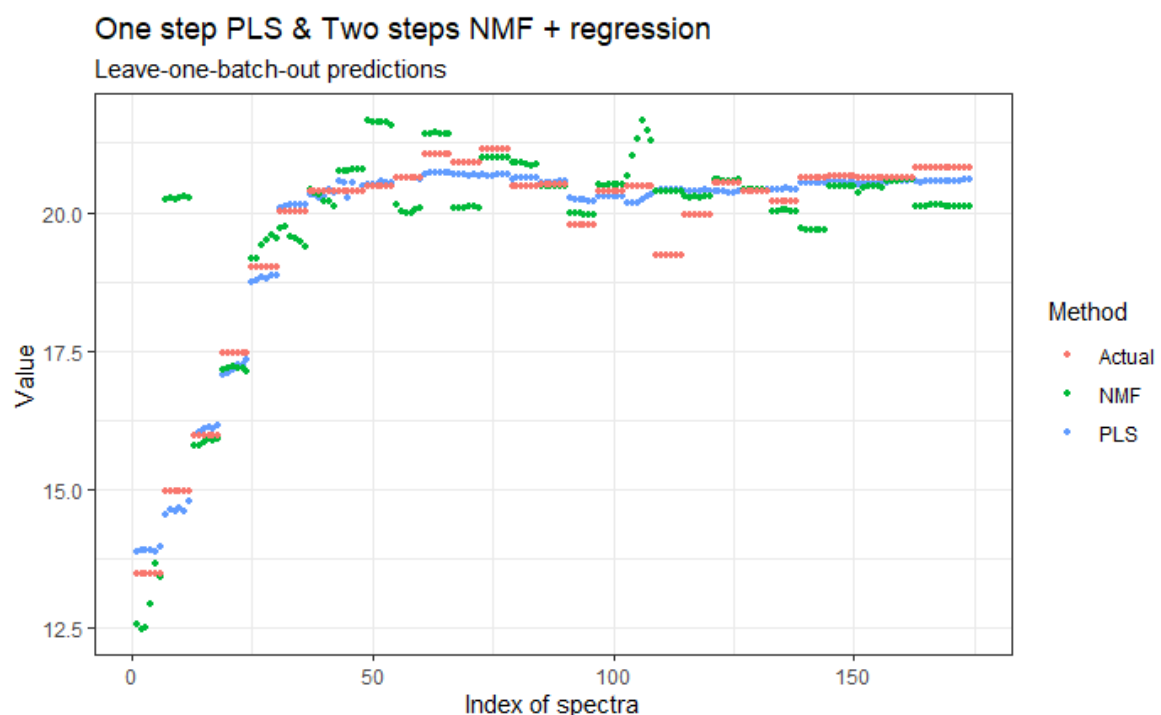


Figure 5.30: Prediction for all batches - one step PLS & two steps NMF

In the **figure 5.31**, the problem of consistency inside the predictions for a given batch is still present for SNMF2. SNMF1 and SNMF3 are unable to predict the concen-

trations for the first, unusual values, and their predictions remain steady. The SNMF2 predicts differently the first batches but still with very large errors.

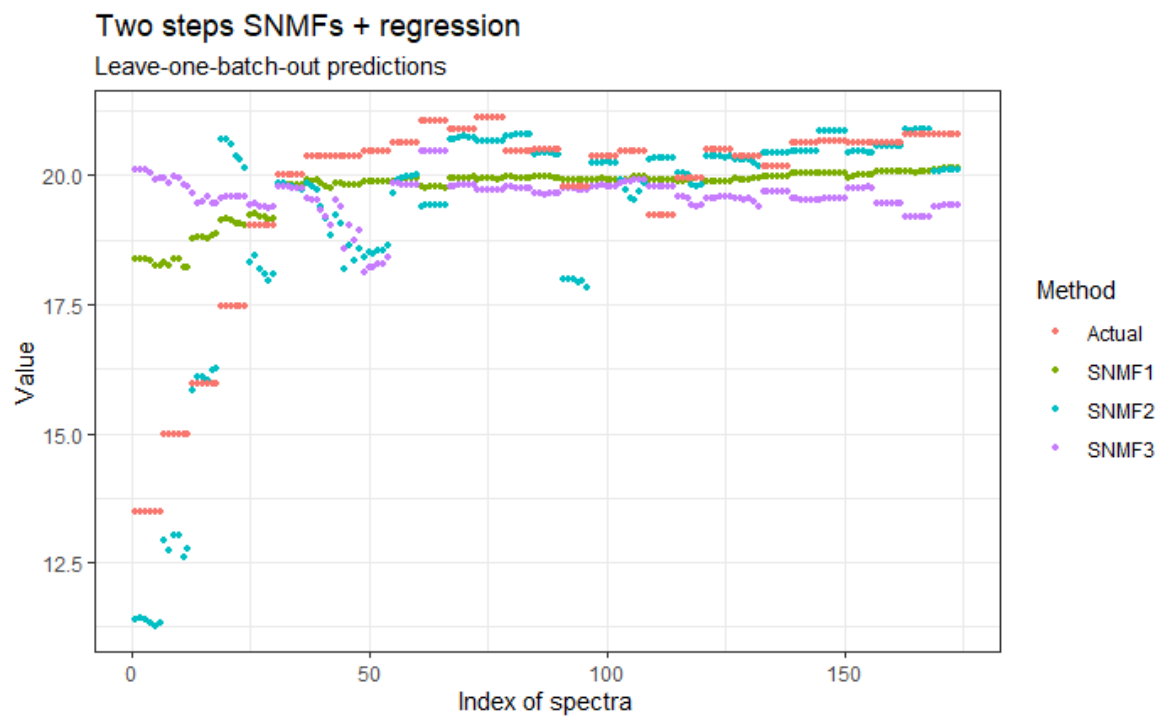


Figure 5.31: Prediction for all batches - two steps SNMFs

Chapter 6

Further work and conclusion

6.1 Further work: Two steps (S)NMF + regression

In the (S)NMF + regression methods, the projection step required to obtain the regression coefficients seemed to somehow lead to loss of information. Indeed, in the simulations, the RMSE on the training set using the projected coefficients (B_{train}^{2s}) were always larger than the training RMSE using the regression coefficients from $Y_{train} \sim U_{train}$ (the unprojected coefficients: $\hat{C}_{train}$) (see the illustration 4.3). Thus, an attempt was to mimick the approach of the PLS to obtain regression coefficients. This attempt was unsuccessful but still deserves to be shown such that it could serve as a basis or be corrected and improved in further researchs.

From the section 4.1.2, let us assume there is an inner relationship between the targets and the scores, such that:

$$Y = UC + E \quad (6.1)$$

Also, assume that the scores are linear combinations of X (with W being the weights):

$$U = XW + E \quad (6.2)$$

Then, the equation 6.2 is plugged into equation 6.1, hence:

$$Y = XWC + E \quad (6.3)$$

Now, W and C can be estimated using least squares:

$$\begin{aligned} \hat{C} &= (U^T U)^{-1} U^T Y \\ \hat{W} &= (X^T X)^{-1} X^T U \end{aligned} \quad (6.4)$$

Finally, the regression coefficients:

$$\begin{aligned} \hat{Y} &= X \hat{B}^{2s} \\ \hat{B}^{2s} &= \hat{W} \hat{C} = (X^T X)^{-1} X^T U (U^T U)^{-1} U^T Y \end{aligned} \quad (6.5)$$

Remind that the first part of the second step in Thiel et al. (2021) (see the figure 4.3) also performed a regression $Y \sim U$, therefore $\hat{C}$ is the same as in the 'classic' two steps regression. Only the second part is different (the computation of $\hat{W}$). To summarize, the classic regression coefficients are $\hat{B}^{2s} = V^T (V V^T)^{-1} \hat{C}$ and the alternative ones are $\hat{B}^{2s} = (X^T X)^{-1} X^T U \hat{C}$.

The new method has the big advantage that the RMSE train with or without projection (using $\hat{Y}_{G/i} = X_{G/i} \hat{B}_{G/i}^{2s}$ or $\hat{Y}_{G/i} = U_{G/i} \hat{C}_{G/i}$) are roughly the same ! Unfortunately, the coefficients $\hat{B}_{G/i}^{2s}$ are no longer smooth and the interpretability is lost, in addition they vary a lot accross folds. On the other hand, the RMSECV (the RMSE on the

validation sets) are very close to the RMSE train (for the artificial dataset) so although the coefficients are not interpretable and messy, they still generalize well (at least for the case of the artificial dataset).

The simulation was performed on the artificial dataset when predicting the component C. The results for the two steps NMF (3 components) + regression using the Thiel's method ('Classic two steps') or the above method ('Alternative two steps') are shown below with a leave-one-batch-out cross validation.

Regarding the classic two steps, the table 6.1 shows the gap between the RMSE on the training sets using the training coefficients $\hat{C}_{G/i}$ or $\hat{B}_{G/i}^{2s}$. On the other hand, the validation RMSE is very close to the training RMSE using the $\hat{B}_{G/i}^{2s}$ coefficients. These coefficients fitted on the last 5 batches to predict the first one are shown in the figure 6.1 and are convenient for interpretation.

	Alternative two steps			Classic two steps		
<i>Pred. Batch</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>
1	1.066	1.066	1.025	1.066	2.026	1.835
2	1.089	1.088	1.012	1.089	2.035	1.779
3	1.071	1.07	1.044	1.071	2.043	1.735
4	1.054	1.054	1.061	1.054	1.954	2.188
5	1.05	1.05	1.087	1.05	1.951	2.204
6	1.055	1.055	1.143	1.055	1.943	2.246
mean	1.064	1.064	1.063	1.064	1.993	2.01

Table 6.1: RMSE: Alternative & Classic two steps NMF + regression (3 components)

Regarding the alternative two steps regression, the RMSE in cross validation using the regression coefficients $\hat{B}_{G/i}^{2s}$ are very similar for the train and the validation sets. In addition, there is no gap between the training RMSE using $\hat{C}_{G/i}$ or $\hat{B}_{G/i}^{2s}$. On the other hand, the coefficients $\hat{B}_{G/i}^{2s}$ are no longer interpretable nor smooth. They also vary a lot when considering different folds, see the figures 6.2 below (coefficients for predicting the first batch).

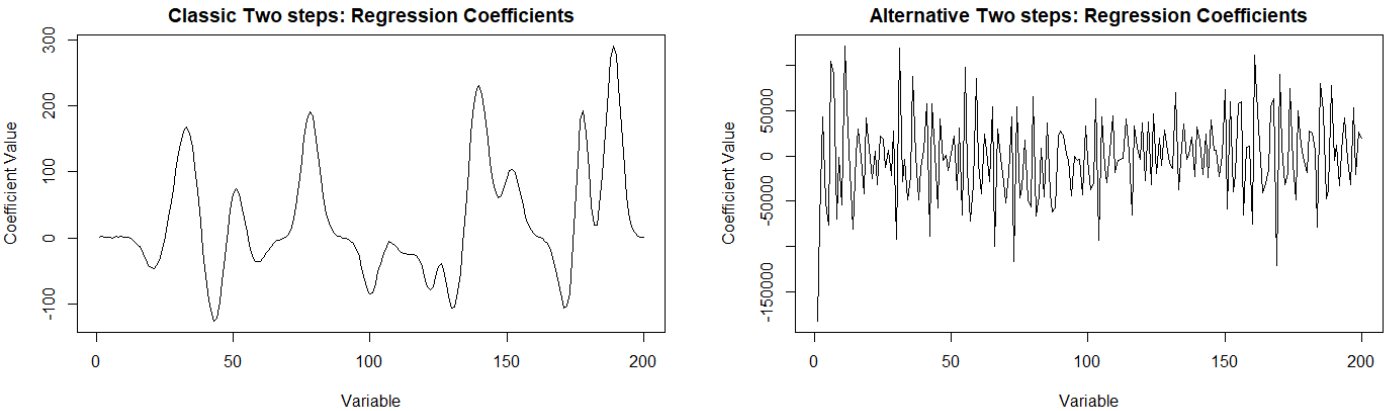


Figure 6.1: Classic Two steps NMF + regression with 3 components

Figure 6.2: Alternative Two steps NMF + regression with 3 components

As a conclusion, although the mean RMSECV for the alternative two steps method is

very low, there are many doubts about the capacity of the coefficients $\hat{B}_{G/i}^{2s}$ to generalize well to new data. This method was thus not retained.

6.2 Conclusion

In this master’s thesis, the starting point was to monitor chemical reactions over time using on-line methods such as infrared spectra. In this context, the usefulness of NMF was shown for the convenience of the interpretation of the scores and loadings it produced, due to its non-negativity constraint.

Thus, after a review of the existing NMF and SNMF algorithms, a new SNMF objective function with an extra supervised penalty term was proposed as well as gradient descent and multiplicative update rules. The theory and algorithms for PCA and PLS (which are used in the case studies) are also presented.

The extra step required to obtain regression coefficients from the scores and loadings of a dimensionality reduction technique such as NMF or SNMF was explained. Hence, it allows to compare multiple algorithms (one-step PLS and two steps (S)NMF) and the accuracy of their prediction for the target variable in a cross-validated scheme.

Then, three datasets were analyzed in the Chapter 5, following a precise methodology. For each case study, the three SNMF algorithms were fitted and tuned, then the loadings, scores and the reconstruction errors of PCA, (S)NMF and PLS were compared. The prediction accuracy and the regression coefficients of the one step PLS and the two steps (S)NMF were also shown.

The main conclusions from these case studies are as follows. First, the reconstruction errors obtained are generally the lowest for PCA or PLS, but those of the NMF and SNMF remain small. Second, the scores and loadings for PLS and PCA show important similarities, and widely differ from the interpretable scores and loadings from (S)NMF. Then, the prediction accuracy in leave-one-batch-out cross-validation was always the best using PLS, on its side, the two steps NMF performs less well but the errors stay reasonable (but remind that the data is centered prior to the PLS). The two steps SNMF is usually able to reduce the training prediction error but it does not translate in a reduction of the validation errors (except for the artificial data), thus reflecting the lack of new relevant patterns identified. On the other hand, the comparison of the regression coefficients was surprising as the one step PLS and the two steps (S)NMF all produced similar ones (the same peaks were always identified, only the intensity of those peaks varied). Yet, the last case study showed an exception where the NMF regression coefficients were unsatisfactory while the SNMF ones were smoothed and enhanced. Overall, NMF is shown to work really well for spectra datasets and seems difficult to improve. It is also probably impossible to reach the same regression coefficients as the PLS, as the scores and loadings of (S)NMF are constrained.

Finally, the trade-off between the accuracy of the prediction and the interpretation of the scores and loadings remains, although the two steps NMF of Thiel et al. (2021) can be used to obtain reasonable prediction errors. Further work could include the search for a new two steps method to construct the regression coefficients. A search for new initializations for the scores and loadings and new objective functions could also be undertaken. It may also be useful to analyze the effect of adding noise to the spectra when decomposing them. Nevertheless, the results are expected to be only marginally better (there is a trade-off as explained above and no free lunch exists).

Chapter 7

Appendix

7.1 Non-negative Matrix Factorization (NMF)

Going from the gradient descent rules to the multiplicative updates using adaptive learning rates (Lee and Seung, 2001)

$$\begin{aligned} U_{ia} &\leftarrow U_{ia} + \eta_{ia}[(XV^T)_{ia} - (UVV^T)_{ia}] \\ &\leftarrow U_{ia} + \frac{U_{ia}}{(UVV^T)_{ia}}[(XV^T)_{ia} - (UVV^T)_{ia}] \\ &\leftarrow U_{ia} + \frac{U_{ia}(XV^T)_{ia}}{(UVV^T)_{ia}} - \frac{U_{ia}(UVV^T)_{ia}}{(UVV^T)_{ia}} \\ &\leftarrow U_{ia}(1 + \frac{(XV^T)_{ia}}{(UVV^T)_{ia}} - 1) \\ U_{ia} &\leftarrow U_{ia} \frac{(XV^T)_{ia}}{(UVV^T)_{ia}} \end{aligned} \tag{7.1}$$

$$\begin{aligned} V_{a\mu} &\leftarrow V_{a\mu} + \eta_{a\mu}[(U^T X)_{a\mu} - (U^T UV)_{a\mu}] \\ &\leftarrow V_{a\mu} + \frac{V_{a\mu}}{(U^T UV)_{a\mu}}[(U^T X)_{a\mu} - (U^T UV)_{a\mu}] \\ &\leftarrow V_{a\mu} + \frac{V_{a\mu}(U^T X)_{a\mu}}{(U^T UV)_{a\mu}} - \frac{V_{a\mu}(U^T UV)_{a\mu}}{(U^T UV)_{a\mu}} \\ &\leftarrow V_{a\mu}(1 + \frac{(U^T X)_{a\mu}}{(U^T UV)_{a\mu}} - 1) \\ V_{a\mu} &\leftarrow V_{a\mu} \frac{(U^T X)_{a\mu}}{(U^T UV)_{a\mu}} \end{aligned} \tag{7.2}$$

7.2 Methods application on case studies

7.2.1 Janssen dataset

For confidentiality reasons, this subsection is only available under request. Please contact Bernadette Govaerts: bernadette.govaerts@uclouvain.be for more information.

7.2.2 Chemium dataset

	PCA - 2 comps			NMF - 3 comps			PLS - 2 comps	
	Indiv	Global	LoBo	Indiv	Global	LoBo	Global	LoBo
1	6e-07	7.87e-05	0.0002157	0.00093	0.015	0.016	8e-05	0.00022
2	3.6e-06	8.04e-05	0.0001069	0.00039	0.012	0.0081	8.4e-05	0.00011
3	5e-07	3.94e-05	5.87e-05	4.4e-05	0.0049	0.0047	4e-05	5.9e-05
4	2e-07	5.43e-05	6.74e-05	0.00014	0.004	0.004	5.3e-05	6.7e-05
5	3e-07	3.32e-05	4.5e-05	0.00033	0.0044	3e-05	3.7e-05	4.5e-05
6	1e-07	2.98e-05	3.42e-05	1e-05	0.00031	3.1e-05	3.2e-05	3.4e-05
7	2e-07	2.26e-05	3.24e-05	1.9e-06	0.0083	0.0071	2.6e-05	3.2e-05
8	2e-07	1.43e-05	1.99e-05	0.00017	0.01	0.00056	1.7e-05	2e-05
9	1e-07	6.4e-06	1.36e-05	1.5e-05	0.015	0.001	7.6e-06	1.4e-05
10	1e-07	2e-05	2.31e-05	4.1e-05	0.00046	4.4e-05	2.1e-05	2.3e-05
11	1e-07	1.21e-05	1.86e-05	1.1e-05	0.0061	0.0018	1.4e-05	1.9e-05
12	1e-07	5.9e-06	6.8e-06	3e-07	0.00063	1.3e-05	6.3e-06	6.8e-06
13	1e-07	4e-06	4.6e-06	4e-07	0.00057	0.00046	4.2e-06	4.6e-06
14	1e-07	1.13e-05	1.29e-05	6e-07	0.00054	1.9e-05	1.2e-05	1.3e-05
15	1e-07	1.19e-05	1.36e-05	7e-07	0.00066	1.3e-05	1.2e-05	1.4e-05
16	1e-07	1.12e-05	1.26e-05	4e-07	0.00046	3e-04	1.2e-05	1.3e-05
17	1e-07	1.86e-05	2.1e-05	1.7e-06	0.00043	1.8e-05	1.9e-05	2.1e-05
18	2e-07	1e-05	1.08e-05	6.9e-05	0.0012	6e-04	1e-05	1.1e-05
19	1e-07	6.1e-06	6.9e-06	5e-07	0.00039	1.2e-05	6.4e-06	6.9e-06
20	1e-07	1.78e-05	2e-05	5.1e-05	0.00082	1.6e-05	1.8e-05	2e-05
21	1e-07	1.3e-05	1.47e-05	9e-07	0.00058	1.1e-05	1.3e-05	1.5e-05
22	1e-07	1.55e-05	1.74e-05	1.7e-05	0.00076	1.8e-05	1.6e-05	1.7e-05
23	1e-07	2e-06	2.2e-06	3e-07	0.00081	7.1e-05	2e-06	2.2e-06
24	1e-07	7.3e-06	8.4e-06	4e-07	0.0014	0.0012	7.5e-06	8.4e-06
25	1e-07	4.1e-06	4.7e-06	4e-07	0.0016	0.00021	4.2e-06	4.7e-06
26	1e-07	6.5e-06	7.3e-06	1.3e-06	0.0011	5.5e-06	6.7e-06	7.2e-06
27	1e-07	3.4e-06	3.9e-06	5e-07	0.0026	5.2e-05	3.4e-06	3.9e-06
28	2e-07	7.8e-06	9.3e-06	5e-07	0.0031	0.00058	7.8e-06	9.3e-06
29	1e-07	7.6e-06	9.1e-06	5e-07	0.0034	0.0035	7.6e-06	9.1e-06
mean	3e-07	1.91e-05	2.83e-05	7.7e-05	0.0035	0.0017	2e-05	2.8e-05

Table 7.1: Error percentage per batch and methodology (%)

	SNMF1			SNMF2			SNMF3		
	Indiv	Global	LoBo	Indiv	Global	LoBo	Indiv	Global	LoBo
1	0.00069	0.017	0.017	0.0018	0.014	0.00034	0.00089	0.009	0.017
2	0.00051	0.013	0.013	84	0.011	0.00035	0.00058	0.0076	0.013
3	5.2e-05	0.0057	0.0059	0.00047	0.0036	0.00018	0.00013	0.0024	0.0058
4	0.00011	0.0038	0.0038	0.00039	0.0017	0.0039	0.00019	0.0021	0.0036
5	2e-04	0.0043	0.0037	0.00045	0.005	0.0034	0.00027	0.005	0.0045
6	3.5e-05	0.00028	0.00024	0.00041	0.00044	0.00022	1.8e-05	0.00028	0.00023
7	6e-04	0.0078	0.007	0.0037	0.0052	0.0076	0.004	0.0077	0.0083
8	0.00038	0.0098	0.0091	0.0021	0.0059	0.011	0.00097	0.014	0.011
9	4.1e-05	0.015	0.014	7.8e-05	0.0082	0.015	7.8e-05	0.048	0.016
10	4.3e-05	0.00054	0.00049	6.2e-05	0.00022	0.00043	6.2e-05	0.001	0.00047
11	3.1e-05	0.0056	0.0048	5e-07	0.015	0.0053	5e-07	0.006	0.0048
12	2.7e-05	0.00048	0.00046	3e-07	3e-04	4.6e-06	3e-07	0.00034	0.00044
13	2.7e-05	0.00052	0.00048	1e-06	0.00052	9.1e-06	1e-06	0.00096	0.00048
14	2.5e-05	0.00047	0.00045	1.7e-05	0.00047	2.7e-05	1.2e-05	0.00033	0.00045
15	2.5e-05	0.00066	6e-04	2.8e-05	0.00067	4e-04	2.8e-05	6e-04	0.00064
16	2.2e-05	0.00036	0.00032	4e-07	0.00041	3.1e-05	4e-07	0.00042	0.00036
17	2.4e-05	0.00043	0.00038	1.2e-06	0.00045	0.00028	4e-07	0.00044	0.00043
18	5.1e-05	0.0011	0.00092	0.00026	0.001	4.8e-05	4.4e-05	0.0012	0.0011
19	2.1e-05	0.00025	0.00023	4e-07	0.00054	2.2e-05	3e-07	0.00042	0.00024
20	4.5e-05	0.0012	0.00068	6.4e-05	0.00075	4e-05	1e-04	0.002	0.00073
21	2.5e-05	0.00051	0.00045	3.2e-06	0.00053	0.00032	3.3e-06	0.00054	0.00048
22	3.7e-05	0.00069	0.00061	6.6e-05	0.00069	2.2e-05	6.7e-05	0.00073	0.00066
23	2.3e-05	0.00077	0.00067	1e-06	0.00073	3e-06	1.4e-06	0.00094	0.00075
24	2.4e-05	0.0038	0.0011	1.2e-06	0.0012	9e-04	1.3e-06	0.002	0.0012
25	2.4e-05	0.0015	0.0013	2.3e-06	0.0015	0.0011	3.3e-06	0.0015	0.0014
26	2.6e-05	0.00096	0.00082	2.8e-06	0.00094	0.00066	2.9e-06	0.00092	0.00091
27	2.4e-05	0.0024	0.002	1.6e-06	0.0023	0.002	0.00027	0.0025	0.0022
28	2.4e-05	0.0036	0.0025	2.5e-06	0.0029	2e-05	2.5e-06	0.0032	0.0027
29	2.4e-05	0.0031	0.0026	1.4e-06	0.0031	8.1e-06	1.7e-06	0.0033	0.003
mean	0.00011	0.0036	0.0033	2.9	0.0031	0.0019	0.00027	0.0043	0.0036

Table 7.2: Error percentage per batch and methodology (%)

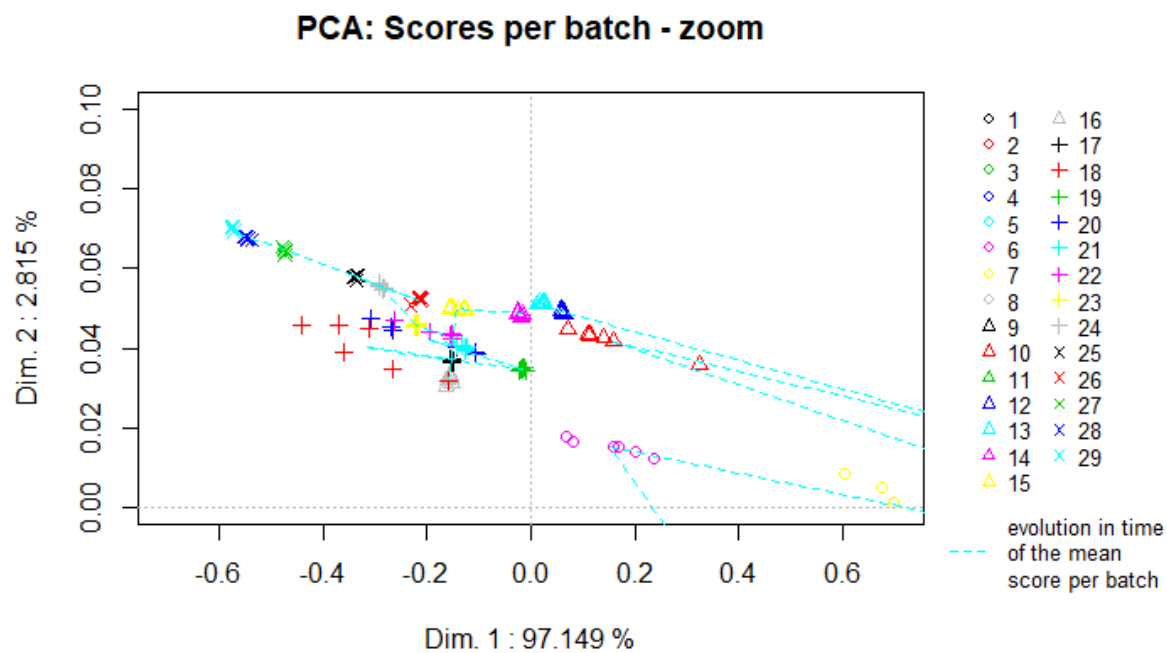


Figure 7.1: PCA scores for the global model - zoom

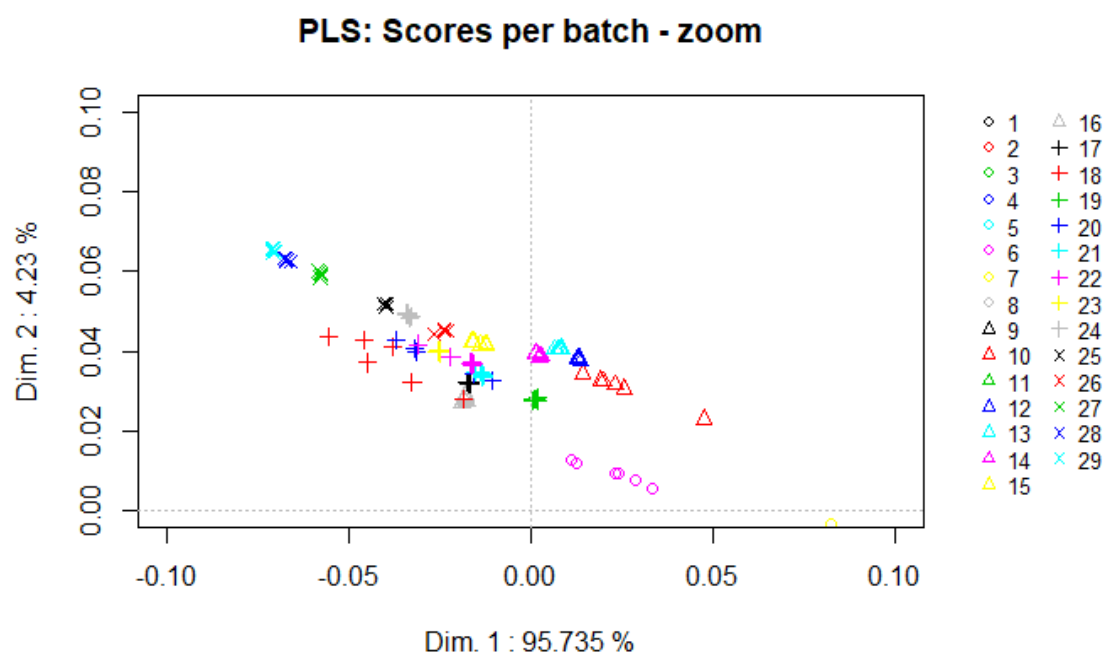


Figure 7.2: PLS scores for the global model - zoom

	PLS		NMF		
<i>Pred. Batch</i>	<i>Train</i>	<i>Valid</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>
1	0.297	0.411	0.311	0.814	0.733
2	0.296	0.351	1.805	1.756	5.288
3	0.3	0.139	1.497	0.411	0.116
4	0.297	0.284	1.911	0.329	0.289
5	0.299	0.225	1.426	0.55	0.41
6	0.301	0.092	1.963	1.716	0.486
7	0.301	0.068	0.839	0.362	0.148
8	0.3	0.124	1.412	0.34	0.369
9	0.301	0.071	1.977	0.557	1.16
10	0.301	0.023	1.786	2.19	0.596
11	0.295	0.344	1.946	0.634	0.375
12	0.298	0.237	1.41	1.533	0.818
13	0.289	0.466	1.912	0.765	0.14
14	0.3	0.149	0.477	0.637	0.412
15	0.301	0.046	2.014	0.37	0.024
16	0.29	0.451	1.873	0.971	0.2
17	0.301	0.097	1.761	0.489	0.1
18	0.298	0.244	1.7	1.851	0.851
19	0.207	1.183	2.009	0.287	1.167
20	0.29	0.446	1.694	0.384	0.34
21	0.3	0.151	2.021	0.77	0.055
22	0.301	0.032	1.542	0.321	0.023
23	0.298	0.23	2.009	0.762	0.163
24	0.301	0.093	1.279	1.196	0.946
25	0.301	0.116	1.527	0.729	0.198
26	0.301	0.103	1.994	0.319	0.196
27	0.301	0.052	2.003	0.558	0.045
28	0.298	0.257	1.57	0.611	0.686
29	0.298	0.231	1.675	0.476	0.698
mean	0.296	0.321	1.689	0.939	1.115

Table 7.3: One step PLS & Two steps NMF + regression: RMSE

	SNMF1			SNMF2			SNMF3		
<i>Pred. Batch</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>	<i>Train ($\hat{C}_{G/i}$)</i>	<i>Train ($\hat{B}_{G/i}^{2s}$)</i>	<i>Valid ($\hat{B}_{G/i}^{2s}$)</i>
1	0.101	1.047	4.85	1.113	0.717	2.125	1.09	1.443	6.563
2	0.1	1.211	3.327	1.14	0.61	2.129	0.602	1.682	4.917
3	0.102	1.287	2.859	0.1	0.635	0.195	0.005	1.784	3.568
4	0.105	1.319	1.647	0.001	1.965	3.016	0.132	1.745	2.124
5	0.11	1.295	0.19	0.005	0.598	0.847	0.008	1.489	0.391
6	0.108	1.34	0.209	0.019	1.059	0.245	0.008	1.778	0.258
7	0.117	1.264	0.518	0.098	0.575	0.98	0.001	1.561	1.03
8	0.119	1.25	0.559	0.002	0.797	1.757	0.109	0.729	1.394
9	0.123	1.217	0.58	0.016	0.696	1.931	0.554	0.796	2.205
10	0.106	1.35	0.714	0.001	1.427	0.737	0.005	1.818	0.807
11	0.109	1.283	1.277	0.029	0.675	1.632	0.088	1.7	0.579
12	0.104	1.358	0.947	0.207	1.689	0.184	0.153	1.768	1.101
13	0.1	1.353	1.203	0.305	0.566	0.471	0.099	1.744	1.41
14	0.103	1.369	0.51	0.26	1.182	0.32	0.006	1.736	0.708
15	0.104	1.364	0.531	0.002	1.199	0.078	0.004	1.726	0.85
16	0.1	1.367	0.148	0.489	6.232	1.82	0.006	1.795	0.035
17	0.104	1.365	0.463	0.002	1.334	0.136	0.032	1.786	0.589
18	0.105	1.353	0.521	0.324	0.784	0.762	0.007	1.714	0.558
19	0.078	1.372	0.69	0.577	1.258	1.114	0.006	1.804	0.561
20	0.103	1.361	0.031	0.173	0.735	0.119	0.112	1.734	0.457
21	0.103	1.366	0.626	0.688	0.921	0.155	0	1.68	0.944
22	0.104	1.366	0.445	0.968	0.437	0.123	0.039	1.728	0.863
23	0.103	1.364	0.211	1.165	0.335	0.257	0.048	1.768	0.501
24	0.105	1.359	0.585	0.001	1.625	0.161	0.016	1.668	1.09
25	0.106	1.352	0.615	0.007	0.574	0.209	0	1.645	1.113
26	0.104	1.361	0.639	0.002	1.334	0.176	0.005	1.804	0.87
27	0.103	1.374	0.547	0.004	1.039	0.056	0.007	1.717	1.161
28	0.102	1.366	0.741	0.638	0.436	0.077	0.003	1.693	1.613
29	0.103	1.366	0.678	1.414	0.558	0.706	0.103	1.758	1.397
mean	0.105	1.326	1.384	0.546	1.525	1.117	0.26	1.668	1.949

Table 7.4: Two steps SNMF + regression: RMSE

7.3 Code documentation

7.3.1 The functions files

This section presents the most important functions which were written for this master's thesis. The functions are divided into 3 files:

- `functions.R` : contains the SNMF algorithm
- `CV_functions.R` : contains the various functions allowing to perform cross-validation (k-fold or leave-one-batch-out) for PLS, and (S)NMF, using an intermediate function to do the two steps regression (Thiel et al., 2021)
- `utils.R` : contains the functions to generate the tables presenting the reconstruction errors for PLS, NMF, SNMF, and PCA. In addition, it contains helper functions to plot the scores, loadings and coefficients of the individual models (as presented in chapter 5).

Finally, another file (contained in the artificial dataset folder) comprises the functions used to generate the data for this case study: `create_data.R`.

7.3.2 Organizations of the case studies files

The implementation of the case studies follows the same logic each time. For each dataset, there are at least two Rmd files containing the simulations and they use the functions comprised in the files above.

The first Rmd (for instance: 'artificial_data_part1') contains the code to write the section 'Data visualization and decomposition' (reconstruction errors tables, graphs for the individual model). The second Rmd contains the code to write the sections 'SNMF parameters', 'Analyze of the SNMF algorithms' and 'Prediction' which requires cross-validation and more intensive computations.

There are also other Rmd files for every data set which were used to test other parameters, preprocess the data (for instance with the Chemium dataset), test a new two steps regression procedure (see the Artificial dataset), create the artificial dataset ('create_data.R' and 'creating_data.Rmd'), etc.

7.3.3 functions.R

SNMF_multiple_strategies

Description

This function allows to call the SNMF algorithm with many parameters directly using the datasets in matrix form.

Usage

```
PARAMS = list("X" = X, "Y" = y, "r" = 3, "lambda" = 1e-5,
               "max_epochs" = 100, "min_epsilon" = 1e-3,
               "fit_intercept" = FALSE, "init" = "NMF_default",
               "plot" = "artificial", "verbose" = FALSE,
               "seed" = 42, "U_strat" = "multiplicative",
               "V_strat" = "multiplicative",
               "w_strat" = "multiplicative",
               "random_weights" = TRUE)
```

```
res = do.call(SNMF_multiple_strategies, PARAMS)
```

```
res$U
res$V
res$initial_U
res$initial_V
res$stats_algo
```

Arguments

X	Matrix of observations $n \times p$
Y	Matrix or vector of target values $n \times k$
r	Numeric, the number of component/latent variables to use
lambda	Numeric, the weight to give to the supervised penalty in the objective function of the SNMF
max_epochs	Numeric, the maximum number of iterations
min_epsilon	Numeric, the value for the stopping criterion: if the difference between the objective function in the current iteration and in the preceding is lower than min_epsilon , then the algorithm stops
random_weights	Logical, to indicate whether the regression coefficients (B in the objective function) should be randomly initialized or set using OLS ($Y \sim UB$)
U_strat	Character, type of update rule to use for U (the scores): "multiplicative" or "gradient_descent"
V_strat	Character, type of update rule to use for V (the loadings): "multiplicative" or "gradient_descent"
w_strat	Character, type of update rule to use for B (the regression coefficients/weights): "multiplicative", "gradient_descent", "linear_regression"

<code>learning_rate_UV</code>	Numeric, only used if <code>U_strat</code> and <code>V_strat</code> are set as "gradient_descent". Learning rate to use for the gradient descent of U and V
<code>learning_rate_w</code>	Numeric, only used if <code>w_strat</code> is set as "gradient_descent". Learning rate to use for the gradient descent of the weights/regression coefficients B
<code>fit_intercept</code>	Logical, to indicate whether to fit an intercept for the weights/regression coefficients B
<code>plot</code>	Character, to indicate the dataset being analysed, produces specific graphs depending on the dataset. Can currently take the values "janssen", "artificial" or "none"
<code>verbose</code>	Logical, to control the level of verbosity
<code>init</code>	Character, indicating whether to use the result of an NMF Brunet ("NMF_default") or an NMF HALSacc ("NMF_HALSacc")
<code>seed</code>	Numeric, set the seed at the beginning of the algorithm

Returns

A list comprising:

<code>U</code>	Matrix of scores
<code>V</code>	Matrix of loadings
<code>w</code>	Matrix of regression coefficients/weights B
<code>initial_U</code>	Matrix of the initial scores
<code>initial_V</code>	Matrix of the initial loadings
<code>stats_algo</code>	Matrix with insights about the algorithm: iter, p1.start, p1.end, p2.start, p2.end, loss.start, loss.end (as in the table 5.2)

7.3.4 CV_functions.R

PLS_cross_validation

Description

The function performs a cross-validation and repeatedly fits PLS models (simpls algorithm) using the `pls` function from the PLS package.

Usage

```
pls.cv = PLS_cross_validation(X = X, Y = y, ncomp = 2,
                             batch_index = batchIdx,
                             CV_type = "leave_one_batch_out")

pander(pls.cv$RMSE)
```

Arguments

X	Matrix of observations $n \times p$
Y	Matrix or vector of target values $n \times k$
ncomp	Numeric, the number of component/latent variables to use
CV_type	Character, type of CV to perform: "kfold" or "leave_one_batch_out"
batch_index	Vector of numeric, if leave-one-batch-out CV, index of the batches for the observations. For instance: <code>c(1,1,2,2,2)</code> : the first two observations are taken from batch 1 and the others are taken from batch 2
nb_fold	Numeric, if k-fold CV, number of folds to use
seed	Numeric, set the seed
scale	Logical, whether to scale or not in the <code>pls</code> function
center	Logical, whether to center or not in the <code>pls</code> function

Returns

A list comprising:

RMSE	A table showing the training and validation RMSE for each batch and the mean
nb_fold_CV	The number of folds/batches which were used
prediction_valid	A table with the prediction for each batch/fold using the models fitted without this batch/fold
coefficients	A table with the pls coefficients of each fitted model
pls.models	A list with all the pls model fitted, as returned by the <code>pls</code> function

Description

This is an intermediate function used by the (S)NMF_cross_validation functions. It performs the two steps regression as introduced by Thiel et al. (2021) and summarized in the figure 4.3. Note that the function uses the 'train', 'test' distinction because it is more general than the 'train', 'validation' which only refers to a cross-validated scheme, but the computations are the same.

Usage

```
res = TWO_STEP_regression(U_scores = U, V_loadings = V,
                          X_train = X_train, Y_train = Y_train,
                          X_test = X_valid, Y_test = Y_valid)

res$rmse.test
res$rmse.train
res$coefficients
res$prediction
```

Arguments

U_scores	Matrix of scores
V_loadings	Matrix of loadings
X_train	Matrix, training set of the matrix of observations
Y_train	Matrix, training set of the matrix of response/targets
X_test	Matrix, test set of the matrix of observations (only used to compute the RMSE on the test set)
Y_test	Matrix, test set of the matrix of response/targets (only used to compute the RMSE on the test set)
verbose	Logical, to control the level of verbosity

Returns

A list comprising:

rmse.train	Numeric, RSME for the training set
rmse.test	Numeric, RMSE fot the test set
coefficients	Matrix, regression coefficients ($\hat{B}_{train}^{2s}$)
prediction	Matrix, prediction for the targets of the test set ($\hat{Y}_{test}$)

Description

The function performs a cross-validation and repeatedly fits NMF models using the Brunet or the HALSacc algorithms from the NMF and hNMF packages. Note that a variation of this function exists (NMF_cross_validation_JANSSEN) where only some of the observations have a response variable. The initial NMF is fitted on the whole training dataset, then the scores corresponding to the observations which have target values are extracted and the two steps regression is applied.

Usage

```
nmf.cv = NMF_cross_validation(X = X, Y = y, ncomp = 3,
                             batch_index = batchIdx, CV_type = "leave_one_batch_out",
                             NMF_type = "default", two_steps_type = "classic")

pander(nmf.cv$RMSE)
```

Arguments

X	Matrix of observations $n \times p$
Y	Matrix or vector of target values $n \times k$
ncomp	Numeric, the number of component/latent variables to use
CV_type	Character, type of CV to perform: "kfold" or "leave_one_batch_out"
batch_index	Vector of numeric, if leave-one-batch-out CV, index of the batches for the observations. For instance: c(1,1,2,2,2) : the first two observations are taken from batch 1 and the others are taken from batch 2
nb_fold	Numeric, if k-fold CV, number of folds to use
NMF_type	Character, type of NMF to perform: "default" (Brunet) or "HAL-Sacc"
seed	Numeric, set the seed
two_steps_type	Character, type of two steps to perform: "classic" (as seen in figure 4.3, Thiel et al.) or "alternative" (as in the conclusion, but it has been shown that this does not work properly)

Returns

A list comprising:

RMSE	Matrix, RSME for the training (using $\hat{C}_{train}$ or $\hat{B}_{train}^{2s}$) and validation sets
coefficients	Matrix, regression coefficients ($\hat{B}_{train}^{2s}$) for each batch/fold
predictions	Matrix, prediction for the targets of the test set for each batch/-fold ($\hat{Y}_{test}$)
C_hat	Matrix, regression coefficients $\hat{C}_{train}$ for each batch/fold
nb_fold	Numeric, number of folds/batches used

Description

The function performs a cross-validation and repeatedly fits SNMF models. It works similarly as `NMF_cross_validation` but requires more parameters.

Usage

```
# the parameters for the SNMF
PARAMS = list("X" = X, "Y" = y, "r" = 3, "lambda" = 1e-5,
              "max_epochs" = 100, "min_epsilon" = 1e-3,
              "fit_intercept" = FALSE, "init" = "NMF_default",
              "plot" = "artificial", "verbose" = FALSE,
              "seed" = 42, "U_strat" = "multiplicative",
              "V_strat" = "multiplicative",
              "w_strat" = "multiplicative",
              "random_weights" = TRUE)
```

```
# adding the parameters for the CV
PARAMS$verbose = FALSE
PARAMS$plot = "none"
PARAMS$batch_index = batchIdx
PARAMS$CV_type = "leave_one_batch_out"
PARAMS$seed = 1
```

```
snmf.cv = do.call(SNMF_cross_validation, PARAMS)
```

```
pander(snmf_1.cv$RMSE)
```

Arguments

<code>CV_type</code>	Character, type of CV to perform: "kfold" or "leave_one_batch_out"
<code>batch_index</code>	Vector of numeric, if leave-one-batch-out CV, index of the batches for the observations. For instance: <code>c(1,1,2,2,2)</code> : the first two observations are taken from batch 1 and the others are taken from batch 2
<code>nb_fold</code>	Numeric, if k-fold CV, number of folds to use
<code>seed</code>	Numeric, set the seed
<code>two_steps_type</code>	Character, type of two steps to perform: "classic" (as seen in figure 4.3, Thiel et al.) or "alternative" (as in the conclusion, but it has been shown that this does not work properly)
<code>...</code>	Arguments to be passed to the SNMF function (see <code>SNMF_multiple_strategies</code>)

Returns

A list comprising:

<code>RMSE</code>	Matrix, RSME for the training (using $\hat{C}_{train}$ or $\hat{B}_{train}^{2s}$) and validation sets
<code>coefficients</code>	Matrix, regression coefficients ($\hat{B}_{train}^{2s}$) for each batch/fold
<code>nb_fold_CV</code>	Numeric, number of folds/batches used

<code>stats_algo</code>	Matrix, table of statistics about the SNMF algorithms fitted for each batch/fold, as in the table 5.2
<code>args_snmf</code>	List, arguments used for the SNMF algorithm (excluding the arguments for the cross validation)

7.3.5 utils.R

`rmse_report_PCA`

Description

The function computes the reconstruction errors for the individual, global and leave-one-batch-out approaches (see section 3.3) for each batch/fold using a PCA and the `prcomp` function. The data needs not be centered when it is passed to the function as it is centered before any call to `prcomp`. We recognize that the name of the function is not well chosen because it is not really computing a RMSE but rather a reconstruction error.

Usage

```
rmse.pca = rmse_report_PCA(df_spectrum, batchIdx = df_metadata$Batch,
                           ncomp = 2)
pander(rmse.pca)
```

Arguments

<code>X</code>	Matrix of observations $n \times p$
<code>batchIdx</code>	Vector of numeric, index of the batches for the observations. For instance: <code>c(1,1,2,2,2)</code> : the first two observations are taken from batch 1 and the others are taken from batch 2
<code>ncomp</code>	Numeric, the number of component/latent variables to use
<code>seed</code>	Numeric, set the seed

Returns

A matrix with the individual, global and leave-one-batch-out reconstruction errors where one line corresponds to one batch/fold. In addition, the last line is the mean for each approach.

Description

The function computes the reconstruction errors for the individual, global and leave-one-batch-out approaches (see section 3.3) for each batch/fold using an NMF Brunet or HALSacc (either the `NMF` or the `hNMF` package). We recognize that the name of the function is not well chosen because it is not really computing a RMSE but rather a reconstruction error.

Usage

```
rmse.nmf = rmse_report_NMF(df_spectrum, batchIdx = df_metadata$Batch,
                           ncomp = 3, nmf = "NMF_default")
pander(rmse.nmf)
```

Arguments

<code>X</code>	Matrix of observations $n \times p$
<code>batchIdx</code>	Vector of numeric, index of the batches for the observations. For instance: <code>c(1,1,2,2,2)</code> : the first two observations are taken from batch 1 and the others are taken from batch 2
<code>ncomp</code>	Numeric, the number of component/latent variables to use
<code>nmf</code>	Character, the NMF algorithm to use: "NMF_default" (for the Brunet algorithm) or "NMF_HALSacc"
<code>seed</code>	Numeric, set the seed

Returns

A matrix with the individual, global and leave-one-batch-out reconstruction errors where one line corresponds to one batch/fold. In addition, the last line is the mean for each approach.

Description

The function computes the reconstruction errors for the individual, global and leave-one-batch-out approaches (see section 3.3) for each batch/fold using a PLS with the 'simpls' algorithm (the function `pls_r` from the package `PLS` is used). We recognize that the name of the function is not well chosen because it is not really computing a RMSE but rather a reconstruction error.

Usage

```
error.pls = rmse_report_PLS(X = df_spectrum, y = df_metadata$C,  
                             batchIdx = df_metadata$Batch, ncomp = 2)  
pander(error.pls)
```

Arguments

<code>X</code>	Matrix of observations $n \times p$
<code>y</code>	Matrix or vector of target values $n \times k$
<code>batchIdx</code>	Vector of numeric, index of the batches for the observations. For instance: <code>c(1,1,2,2,2)</code> : the first two observations are taken from batch 1 and the others are taken from batch 2
<code>ncomp</code>	Numeric, the number of component/latent variables to use
<code>seed</code>	Numeric, set the seed

Returns

A matrix with the individual, global and leave-one-batch-out reconstruction errors where one line corresponds to one batch/fold. In addition, the last line is the mean for each approach.

Description

The function computes the reconstruction errors for the individual, global and leave-one-batch-out approaches (see section 3.3) for each batch/fold using an SNMF (see the function `SNMF_multiple_strategies`). We recognize that the name of the function is not well chosen because it is not really computing a RMSE but rather a reconstruction error.

Usage

```
# the parameters for the SNMF
PARAMS = list("X" = X, "Y" = y, "r" = 3, "lambda" = 1e-5,
              "max_epochs" = 100, "min_epsilon" = 1e-3,
              "fit_intercept" = FALSE, "init" = "NMF_default",
              "plot" = "artificial", "verbose" = FALSE,
              "seed" = 42, "U_strat" = "multiplicative",
              "V_strat" = "multiplicative",
              "W_strat" = "multiplicative",
              "random_weights" = TRUE)

# adding the parameters for the reconstruction error function
PARAMS$batchIdx = batchIdx

error.snmf = do.call(rmse_report_SNMF, PARAMS)
pander(error.snmf)
```

Arguments

<code>batchIdx</code>	Vector of numeric, index of the batches for the observations. For instance: <code>c(1,1,2,2,2)</code> : the first two observations are taken from batch 1 and the others are taken from batch 2
<code>...</code>	Arguments to be passed to the SNMF function (see <code>SNMF_multiple_strategies</code>)

Returns

A matrix with the individual, global and leave-one-batch-out reconstruction errors where one line corresponds to one batch/fold. In addition, the last line is the mean for each approach.

7.3.6 create_data.R

create_data

Description

The function creates the artificial data. First, it generates the true quantities for each component (using `evolution_components`). Then, it generates the true peaks, the intensity and variability of each peak (using `generate_true_profile`). Then, it calls N (number of timesteps) times the function generating the approximate profile (`generate_approximate_profile`). The latter generates the approximate profile for each component (with noise), then these profiles are normalized such that the area under the curve is 1 for each component. Finally, the spectrum at timestep i is obtained by a linear combination of the approximate profiles of each components, weighted by the quantity of each.

Usage

```
PARAMS_SIMU = list(max_timestep = 50, nbr_peaks = 8,
                    noise_peak_sd = 0.4, noise_variability_mean = 0.3,
                    noise_lognorm_mean = 0, noise_lognorm_sd = 1)

data = do.call(create_data, c(PARAMS_SIMU,
                              speed_factor = 0.9,
                              verbose = T))

df_spectra = data$df_spectra
df_true_profile = data$df_true_profiles
df_quantity = data$true_quantities
```

Arguments

<code>initial_A</code>	Numeric, quantity of A at the first timestep
<code>initial_B</code>	Numeric, quantity of B at the first timestep
<code>initial_C</code>	Numeric, quantity of C at the first timestep
<code>max_timestep</code>	Numeric, maximum number of timesteps
<code>speed_factor</code>	Numeric, the speed of reaction of the chemical components
<code>delay_B</code>	Numeric, the number of timesteps until the component B starts to increase
<code>delay_C</code>	Numeric, the number of timesteps until the component C starts to increase
<code>nbr_peaks</code>	Numeric, the number of peaks for the true profile of the chemical components
<code>verbose</code>	Logical, to control the level of verbosity

For `noise_peak_mean`, `noise_peak_sd`, `noise_variability_mean`, `noise_variability_sd`, `noise_lognorm_mean`, `noise_lognorm_sd`: Numeric, parameter to control the noise added to the observed spectra, see the section 2.1.3.

Returns

A list comprising:

<code>df_spectra</code>	DataFrame, the artificial observed spectra over time
<code>df_true_profiles</code>	DataFrame, the true profiles for the chemical components
<code>true_quantities</code>	DataFrame, the true quantities for each chemical components over time

Bibliography

- Albright, R., Cox, J., Duling, D., Langville, A., and Meyer, C. (2014). Algorithms, initializations, and convergence for the nonnegative matrix factorization. *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Disc. & Data Mining*.
- Berry, M. W., Browne, M., Langville, A. N., Pauca, V. P., and Plemmons, R. J. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173.
- Boutsidis, C. and Gallopoulos, E. (2008). Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362.
- Chao, G., Mao, C., Wang, F., Zhao, Y., and Luo, Y. (2018). Supervised nonnegative matrix factorization to predict icu mortality risk. *IEEE International Conference on Bioinformatics and Biomedicine*, 2018:1189–1194.
- Cichocki, A. and Phan, A.-H. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions*, 92-A:708–721.
- Ding, C. H. Q., Li, T., and Jordan, M. I. (2010). Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55.
- Franc, V., Hlavac, V., and Navara, M. (2005). Sequential coordinate-wise algorithm for the non-negative least squares problem. pages 407–414.
- Gaujoux, R. and Seoighe, C. (2010). A flexible r package for nonnegative matrix factorization. *BMC Bioinformatics*, 11(1):367.
- Gaujoux, R. and Seoighe, C. (2020a). *The package NMF: manual pages*. R package version 0.23.0.
- Gaujoux, R. and Seoighe, C. (2020b). *Using the package NMF*. R package version 0.23.0.
- Gillis, N. (2014). The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12.
- Gillis, N. and Glineur, F. (2011). Nonnegative matrix factorization : complexity, algorithms and applications. *PhD thesis, UCLouvain*.
- Govaerts, B. (2018). *Lecture notes - PLS card*. UCLouvain.
- Govaerts, B. and Martin, M. (2018). *Lecture notes - PCA card*. UCLouvain.
- Lawson, C. and Hanson, R. (1995). *Solving least squares problems*. Classics in Applied Mathematics.
- Lee, D. and Seung, H. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–91.
- Lee, D. and Seung, H. (2001). Algorithms for non-negative matrix factorization. *Adv. Neural Inform. Process. Syst.*, 13.
- Lee, H., Yoo, J., and Choi, S. (2010). Semi-supervised nonnegative matrix factorization. *IEEE Signal Processing Letters*, 17(1):4–7.

- Leuschner, J., Schmidt, M., Fernsel, P., Lachmund, D., Boskamp, T., and Maass, P. (2018). Supervised non-negative matrix factorization methods for MALDI imaging applications. *Bioinformatics*, 35(11):1940–1947.
- Lin, C.-J. (2007). Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19:2756–79.
- Lin, E. X. (2020a). *NNLM: A package For Fast And Versatile Nonnegative Matrix Factorization*. Vignettes.
- Lin, E. X. (2020b). *NNLM: Fast and Versatile Non-Negative Matrix Factorization*. R package.
- M. Mullen, K. and van Stokkum, I. H. M. (2020). *Package ‘nnls’: The Lawson-Hanson algorithm for non-negative least squares (NNLS)*.
- Mevik, B.-H. and Wehrens, R. (2007). The pls package: Principal component and partial least squares regression in r. *Journal of Statistical Software, Articles*, 18(2):1–23.
- Nicolas Sauwen (2020). *Package ‘hNMF’: Hierarchical Non-Negative Matrix Factorization*.
- Oirbeek, R. V., Adriaan, B., Sauwen, N., Maes, T., Dijkmans, J., Cuypers, J., Khamiakova, T., Tiel, M., and Beleites, C. (2020). *Package spectralAnalysis: Pre-Process, Visualize and Analyse Process Analytical Data, by Spectral Data Measurements Made During a Chemical Process*.
- Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.
- Rousseaux, A. and Nagy, A. (2021). *Optimisation du monitoring par spectroscopie proche infrarouge (NIR) d’un réacteur flow de synthèse*. Haute Ecole Louvain en Hainaut (HELHa).
- Segers, J. (2019). *Lecture notes in LSTAT2110, Analyse des données*. UCLouvain.
- Thiel, M., Sauwen, N., Khamiakova, T., Maes, T., and Govaerts, B. (2021). Comparison of chemometrics strategies for the spectroscopic monitoring of active pharmaceutical ingredients in chemical reactions. *Chemometrics and Intelligent Laboratory Systems*, 211:104273.
- Varmuza, K. and Filzmoser, P. (2009). *Introduction to Multivariate Statistical Analysis in Chemometrics*. Taylor & Francis - CRC Press.
- Wang, Y. and Zhang, Y. (2013). Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté des sciences

Place des sciences, 2 bte L6.06.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/sc