UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Use of multivariate statistical methods for the analysis of metabolomic data

July 2019

Author: David Hervás Marín

Directors: José Manuel Prats Montalbán
Agustín Lahoz Rodríguez

Acknowledgments

Writing a thesis always involves a lot of working, fighting against adversity and enduring numerous obstacles. This one is no exception, and it wouldn't have been finished without the help of my two directors José Manuel Prats Montalbán and Agustín Lahoz Rodríguez.

I want to thank also the help and insights given by Alberto Ferrer Riquelme, who has greatly contributed with his knowledge to the contents of this work.

Special thanks to Victoria Fornés Ferrer for her help with the valencian version of the abstract and for being such a great colleage.

Finally, some lines to my parents Ana and Tomás, my brother Raúl, my wife Lucía and my precious daughter Laura. Thank you for your support, for your patience and your love. This thesis is dedicated to you all.

Resumen

En las últimas décadas los avances tecnológicos han tenido como consecuencia la generación de una creciente cantidad de datos en el campo de la biología y la biomedicina. A día de hoy, las así llamadas tecnologías "ómicas", como la genómica, epigenómica, transcriptómica o metabolómica entre otras, producen bases de datos con cientos, miles o incluso millones de variables.

El análisis de datos ómicos presenta una serie de complejidades tanto metodológicas como computacionales que han llevado a una revolución en el desarrollo de nuevos métodos estadísticos específicamente diseñados para tratar con este tipo de datos.

A estas complejidades metodológicas hay que añadir que, en la mayor parte de los casos, las restricciones logísticas y/o económicas de los proyectos de investigación suelen conllevar que los tamaños muestrales en estas bases de datos con tantas variables sean muy bajos, lo cual no hace sino empeorar las dificultades de análisis, ya que se tienen muchísimas más variables que observaciones.

Entre las técnicas desarrolladas para tratar con este tipo de datos podemos encontrar algunas basadas en la penalización de los coeficientes, como lasso o elastic net, otras basadas en técnicas de proyección sobre estructuras latentes como PCA o PLS y otras basadas en árboles o combinaciones de árboles como random forest.

Todas estas técnicas funcionan muy bien sobre distintos datos *ómicos* presentados en forma de matriz (IxJ). Sin embargo, en ocasiones los datos ómicos pueden estar expandidos, por ejemplo, al tomar medidas repetidas en el tiempo sobre los mismos individuos, encontrándonos con estructuras de datos que ya no son matrices, sino arrays tridimensionales o *three-way* ($IxJxK$). En estos casos, la mayoría de las técnicas citadas pierden parte de su aplicabilidad, quedando muy pocas opciones viables para el análisis de este tipo de estructuras de datos.

Una de las técnicas que sí es útil para el análisis de estructuras *three-way* es N-PLS, que permite ajustar modelos predictivos razonablemente precisos, así como interpretarlos mediante distintos gráficos.

Sin embargo, relacionado con el problema de la escasez de tamaño muestral relativa al desorbitado número de variables, aparece la necesidad de realizar una selección de variables relacionadas con la variable respuesta. Esto es especialmente cierto en el ámbito de la biología y la biomedicina, ya que no solo se quiere poder predecir lo que va a suceder, sino entender por qué sucede, qué variables están implicadas y, a poder ser, no tener que volver a recoger los cientos de miles de variables para realizar una nueva predicción, sino utilizar unas cuantas, las más importantes, para poder diseñar kits predictivos coste/efectivos de utilidad real. Por ello, el objetivo principal de esta tesis es mejorar las técnicas existentes para el análisis de datos ómicos, específicamente las encaminadas a analizar datos *three-way*, incorporando la capacidad de selección de variables, mejorando la capacidad predictiva y mejorando la interpretabilidad de los resultados obtenidos. Todo ello se implementará además en un paquete de R completamente documentado, que incluirá todas las funciones necesarias para llevar a cabo análisis completos de datos *three-way*.

El trabajo incluido en esta tesis por tanto, consta de una primera parte teórico-conceptual de desarrollo de la idea del algoritmo, así como su puesta a punto, validación y comprobación de su eficacia; de una segunda parte empírico-práctica de comparación de los resultados del algoritmo con otras metodologías de selección de variables existentes, y de una parte adicional de programación y desarrollo de *software* en la que se presenta todo el desarrollo del paquete de R, su funcionalidad y capacidades de análisis.

El desarrollo y validación de la técnica, así como la publicación del paquete de R, que ya cuenta con varios cientos de usuarios, ha permitido ampliar las opciones actuales para el análisis de datos ómicos *three-way* abriendo un gran número de líneas futuras de investigación.

Resum

En les últimes dècades els avançaments tecnològics han tingut com a conseqüència la generació d'una creixent quantitat de dades en el camp de la biologia i la biomedicina. A dia d'avui, les anomenades tecnologies "òmiques", com la genòmica, epigenòmica, transcriptòmica o metabolòmica entre altres, produeixen bases de dades amb centenars, milers o fins i tot milions de variables.

L'anàlisi de dades 'òmiques' presenta una sèrie de complexitats tant metodològiques com computacionals que han portat a una revolució en el desenvolupament de nous mètodes estadístics específicament dissenyats per a tractar amb aquest tipus de dades.

A aquestes complexitats metodològiques cal afegir que, en la major part dels casos, les restriccions logístiques i / o econòmiques dels projectes de recerca solen comportar que les magnituts de les mostres en aquestes bases de dades amb tantes variables siguen molt baixes, el que no fa sinó empitjorar les dificultats d'anàlisi, ja que es tenen moltíssimes més variables que observacions

Entre les tècniques desenvolupades per a tractar amb aquest tipus de dades podem trobar algunes basades en la penalització dels coeficients, com lasso o elastic net, altres basades en tècniques de projecció sobre estructures latents com PCA o PLS i altres basades en arbres o combinacions d'arbres com random forest.

Totes aquestes tècniques funcionen molt bé sobre diferents dades 'òmiques' presentats en forma de matriu (IxJ), però, en ocasions les dades òmiques poden estar expandits, per exemple, cuan ni ha mesures repetides en el temps

sobre els mateixos individus, trobant-se amb estructures de dades que ja no són matrius, sinó arrays tridimensionals o *three-way* ($IxJxK$). En aquestos casos, la majoria de les tècniques mencionades perden tota o bona part de la seua aplicabilitat, quedant molt poques opcions viables per a l'anàlisi d'aquest tipus d'estructures de dades.

Una de les tècniques que sí que és útil per a l'anàlisi d'estructures *three-way* es N-PLS, que permet ajustar models predictius raonablement precisos, així com interpretar-los mitjançant diferents gràfics.

No obstant això, relacionat amb el problema de l'escassetat de mostres relativa al desorbitat nombre de variables, apareix la necessitat de realitzar una selecció de variables relacionades amb la variable resposta. Això és especialment cert en l'àmbit de la biologia i la biomedicina, ja que no només es vol poder predir el que va a succeir, sinó entendre per què passa, quines variables estan implicades i, si pot ser, no haver de tornar a recollir els centenars de milers de variables per realitzar una nova predicció, sinó utilitzar unes quantes, les més importants, per poder dissenyar kits predictius cost / efectius d'utilitat real. Per això, l'objectiu principal d'aquesta tesi és millorar les tècniques existents per a l'anàlisi de dades òmiques, específicament les encaminades a analitzar dades *three-way*, incorporant la capacitat de selecció de variables, millorant la capacitat predictiva i millorant la interpretabilitat dels resultats obtinguts. Tot això s'implementarà a més en un paquet de R completament documentat, que inclourà totes les funcions necessàries per a dur a terme anàlisis completes de dades *three-way*.

El treball inclòs en aquesta tesi per tant, consta d'una primera part teorica-conceptual de desenvolupament de la idea de l'algoritme, així com la seua posada a punt, validació i comprovació de la seua eficàcia, d'una segona part empíric-pràctica de comparació dels resultats de l'algoritme amb altres metodologies de selecció de variables existents i d'una part adicional de programació i desenvolupament de programació en la qual es presenta tot el desenvolupament del paquet de R, la seua funcionalitat i capacitats d'anàlisi.

El desenvolupament i validació de la tècnica, així com la publicació del paquet de R, que ja compta amb diversos centenars d'usuaris, ha permès ampliar les opcions actuals per a l'anàlisi de dades 'òmiques' *three-way* obrint un gran nombre de línies futures de recerca .

Abstract

In the last decades, advances in technology have enabled the gathering of an increasingly amount of data in the field of biology and biomedicine. The so called "-omics" technologies such as genomics, epigenomics, transcriptomics or metabolomics, among others, produce hundreds, thousands or even millions of variables per data set.

The analysis of 'omic' data presents different complexities that can be methodological and computational. This has driven a revolution in the development of new statistical methods specifically designed for dealing with these type of data.

To this methodological complexities one must add the logistic and economic restrictions usually present in scientific research projects that lead to small sample sizes paired to these wide data sets. This makes the analyses even harder, since there is a problem in having many more variables than observations.

Among the methods developed to deal with these type of data there are some based on the penalization of the coefficients, such as lasso or elastic net, others based on projection techniques, such as PCA or PLS, and others based in regression or classification trees and ensemble methods such as random forest.

All these techniques work fine when dealing with different 'omic' data in matrix format (IxJ), but sometimes, these IxJ data sets can be expanded by taking, for example, repeated measurements at different time points for each individual, thus having IxJxK data sets that raise more methodological com-

plications to the analyses. These data sets are called three-way data. In this cases, the majority of the cited techniques lose all or a good part of their applicability, leaving very few viable options for the analysis of this type of data structures.

One useful tool for analyzing three-way data, when some **Y** data structure is to be predicted, is N-PLS. N-PLS reduces the inclusion of noise in the models and obtains more robust parameters when compared to PLS while, at the same time, producing easy-to-understand plots.

Related to the problem of small sample sizes and exorbitant variable numbers, comes the issue of variable selection. Variable selection is essential for facilitating biological interpretation of the results when analyzing 'omic' data sets. Often, the aim of the study is not only predicting the outcome, but also understanding why it is happening and also what variables are involved. It is also of interest being able to perform new predictions without having to collect all the variables again. Because all of this, the main goal of this thesis is to improve the existing methods for 'omic' data analysis, specifically those for dealing with three-way data, incorporating the ability of variable selection, improving predictive capacity and interpretability of results. All this will be implemented in a fully documented `R` package, that will include all the necessary functions for performing complete analyses of three-way data.

The work included in this thesis consists in a first theoretical-conceptual part where the idea and development of the algorithm takes place, as well as its tuning, validation and assessment of its performance. Then, a second empirical-practical part comes where the algorithm is compared to other variable selection methodologies. Finally, an additional programming and software development part is presented where all the `R` package development takes place, and its functionality and capabilities are exposed.

The development and validation of the technique, as well as the publication of the R package, which has already hundreds of users, has opened many future research lines.

List of Figures

List of Tables

Contents

Part I

Introduction and literature review

Chapter 1

Introduction, contents and aims of the thesis

1.1 Introduction

Data sets have experienced a rapid growth in size and complexity during the last years, and biology and biomedicine have been no exception (Marx 2013; Costa 2014). Few decades ago, a typical data set consisted in dozens of variables, and a large data set was one with some few hundreds of variables. Nowadays, the different 'omic' technologies have evolved producing data sets ranging from hundreds of variables, in the case of specific panels or targeted analyses, to thousands or even hundreds of thousands of variables for untargeted or whole genome analyses. A specific characteristic of 'omic' data sets, compared to data sets from other disciplines is their heterogeneous nature. Individual variability is huge and experiments have to be repeated many times to reach conclusions (Lay Jr et al. 2006). This has lead to the so called "reproducibility crisis", which considers that many research findings could be false (John PA Ioannidis 2005; Begley and John PA Ioannidis 2015). In fact, as shown in Figure 1.1, some studies have estimated that more than 70% of published works in the field of biology may be not reproducible (Baker 2016).

Among the causes for lack of reproducibility, the most prominent ones are the use of inadequate statistical methods, the misuse of adequate statistical methods and the misunderstanding of statistical results (J. Ioannidis 2017; Gagnier and Morgenstern 2017; Diong et al. 2018). There is also an inertia to continue using the same outdated methods that have been used during the last 30 years (Leek et al. 2017). These issues are prevalent in all scientific branches, but in biology, and most specifically in the field of 'omic' data, they are critical because of the enormous variability in the data (both technical and biological), the large number of variables and the prevailing low sample sizes. It is still very common to find publications in top journals where no corrections for multiple comparisons have been applied after performing hundreds of tests or using statistical methods such as the *t test* without even considering their assumptions (normality and homocedasticity) (Marino 2014; Eklund, Nichols, and Knutsson 2016). During the last decades, a wide array of statistical methods have been specifically developed to deal with the issues and hurdles present when analyzing 'omic' data sets, but their adoption has been pretty slow in most fields. In the following chapters, the different methods for analyzing 'omic' data will be presented and critically reviewed before presenting a new method for the analysis of multi-way arrays that also implements variable selection.

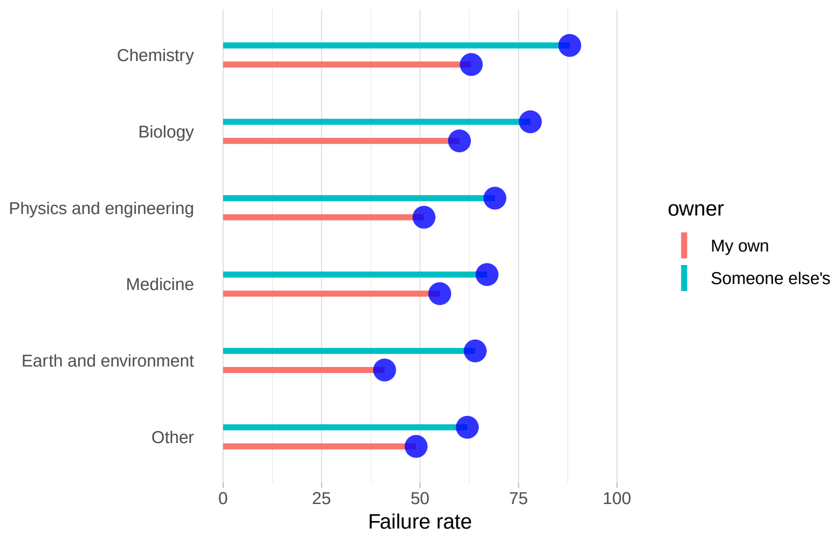

Figure 1.1: Reproducibility problems in different branches of science (data from Baker 2016)

1.2 Characteristics of 'omic' data

The word "omics" makes reference to the study of some characteristics of different families of biological molecules, such as genes, proteins, etc. (Palsson 2002). The basic aspect of the 'omic' approaches is that a complex system can be understood more thoroughly if considered as a whole. Thanks to this approach, the advances that have been achieved in the last decades in the understanding of the different biological processes are huge, and today the future of medicine and biology cannot be understood outside the view of the different 'omics' approaches (Karnebeek et al. 2018). In few years, 'omics' technologies have given us an unimaginable deep knowledge of many cellular processes and pathways, helping the scientific community to make huge progresses in the understanding of living systems. Each year new technologies are developed allowing for the gathering of more information, increasing the resolution of the analyses and improving the precision of the determinations. As emerging fields, the different 'omics' are still expanding and defining themselves but, in a general view, 'omic' data can be classified according to the nature of the elements being analyzed. Following this approach, 5 different fields can be delimited (Figure 1.2), (Horgan and Kenny 2011; Barh and Azevedo 2018):

- Genomics: Focused at the structure, function, evolution and mapping of genomes, that is, the complete set of DNA of an organism.
- Epigenomics: Studies the complete set of epigenetic reversible modifications of the genome, which regulate expression.
- Transcriptomics: Focused at the whole set of RNA transcripts of an organism, that is, the expression of an organism's genes.
- Proteomics: Focused at the whole set of proteins produced or modified by an organism. Proteomics studies protein composition, structure and activity.
- Metabolomics: Studies metabolites in a biological entity, which are the end products of cellular processes.

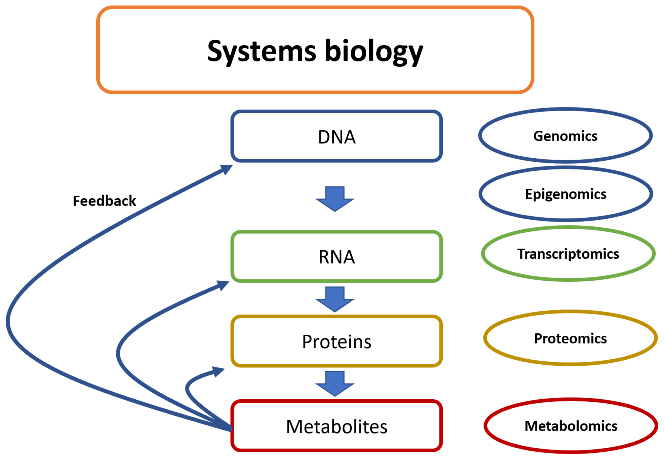

Figure 1.2: Different 'omic' technologies and their relationship

1.2.1 Metabolomic data

The term '*metabolome*' is used to address the entire set of metabolites present in an organism and '*metabolomics*' is defined as the comprehensive and quantitative analysis of all metabolites of the biological system under study (Fiehn 2001). Metabolomics studies the downstream products of the so called "omics cascade" presented in Figure 1.2, so its information is influenced by the actions of genomic, epigenomic, transcriptomic and proteomic mechanisms. Because of this, metabolomics is the closest approximation to phenotype among all 'omic' technologies, thus providing more information about the actual status of the organism than the other 'omic' technologies (Beger et al. 2016).

In general metabolomic studies can be classified as targeted or untargeted depending on whether the researcher measures and quantifies a specific set of known metabolites or the largest possible number of metabolites contained in a biological system (Orešič 2009). Among all 'omic' technologies, metabolomics is the most complex and heterogeneous in terms of chemical diversity. More specifically, targeted metabolomic studies focus on accurate identification and quantification of a defined set of metabolites in biological samples which was predetermined by the scientific question formulated by the researcher or, in some cases, by the size of metabolite library that is available in the software used for the analyses. On the other hand, untargeted metabolomics focuses on measuring and comparing as many signals as possible in a biological samples, and then assigning these signals to specific metabolites by using annotation databases such as HMDB (D. Wishart, Tzur, Knox, et al. 2007) and Metlin (Smith et al. 2005). It is important to mention that a significant portion of the detected signals cannot be identified as the metabolome is not fully annotated (Viant et al. 2017). A typical raw data spectrum of a metabolomics analysis is presented in Figure 1.3.

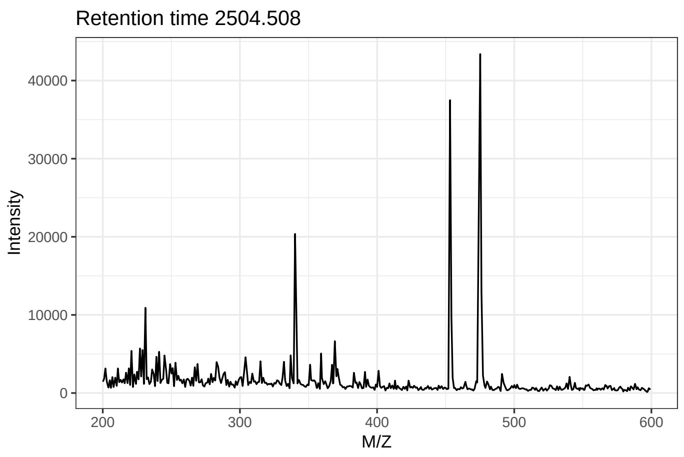

Figure 1.3: Raw spectrum of a metabolomics analysis

In genomics, epigenomics and transcriptomics, measurement procedures are reasonably standardized. This is not the case in metabolomics, with many different technologies being used, which lead to different study designs and different produced data types (Moco et al. 2007). The commonly used analytical techniques for metabolomic studies are nuclear magnetic resonance (NMR), spectroscopy and mass spectrometry (Büscher et al. 2009). Another specific hurdle of metabolomics is the identification of the metabolites in untargeted analyses. Many of the metabolites found in an untargeted analysis can not be identified, because the data bases for associating a specific mass and retention time to a concrete metabolite are still very immature (Mathew and Padmanaban 2013). In fact, identification of unknowns is considered as the bottleneck of untargeted metabolomics (Bingol 2018). In any case, although heterogeneous, metabolomic data share a common set of properties which define their most important characteristics for their statistical analysis:

- High correlation among variables
- Complex pre-processing of the signal to obtain the data

- High range of detection values among variables (i.e., some metabolites are found in very small concentrations and others are found in very high concentrations)
- Variables (metabolites) found in one analysis are not guaranteed to be found in a subsequent analysis (i.e., lack of reproducibility).

All these properties add to the difficulty of analyzing metabolomic data, and will be further developed in the following sections.

Chapter 2

Common strategies for the analysis of metabolomic data

2.1 Classical bivariate tests

As exposed in section 4.1, many metabolomic data analysis are performed using rudimentary statistical methods such as bivariate tests (Saccenti et al. 2014). These tests, such as the well know *t test* consist in the test of a specific hypothesis of equality for each of the means of the variables being studied. The hypothesis test assesses for each metabolite if its mean (or median depending on the test being used) is different among the groups being compared. This way, in a study searching for biomarkers to discern between controls and patients of a specific disease, one hypothesis test (Equation 2.1) for differences between both groups is performed for each of the metabolites in the data set, thus performing hundreds or thousands of hypothesis test in the analysis.

$$\begin{aligned} H_0 &: \mu_{controls} = \mu_{patients} \\ H_A &: \mu_{controls} \neq \mu_{patients} \end{aligned} \tag{2.1}$$

The advantages of bivariate tests are the ease of application and interpretation of the results (Vetter and Mascha 2018), but they suffer from a lot of issues. One of the most important issues when performing bivariate tests is the integration of results (Katz 2011). Usually, even when finding several statistically significant biomarkers in a study, each of them has a small effect not being able by itself of discriminating between the groups being studied. Since the analysis was performed independenly on each variable, there is no way of combining the different selected biomarkers to perform a prediction, or even to understand the behavior of the studied biological system. Another important issue is the inability of controlling for confounders, i.e., variables correlated with both the biomarker and the response (Heinze and Dunkler 2017). Most sudies are observational, so groups are usually not directly comparable. Bivariate tests are not able to take into account possible confounders and this renders many of their results invalid. There is also no way of assessing possible interactions among variables (Hassall and Mead 2018). As seen in Figure 2.1, the effect of one variable in the response can depend on the value of another. All this important information is completely lost when analyzing data one variable at a time.

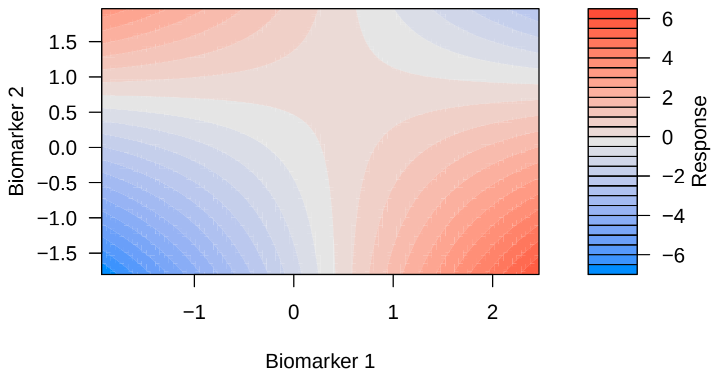

Figure 2.1: Interaction between two variables (biomarker 1 and biomarker 2). Higher values of one of the two biomarkers are associated to higher values of the response, but only if the other biomarker has not high values either.

Finally, there is also a concern with the question being asked by the bivariate tests. Usually, what these analysis are doing is flipping the question. If one is looking for biomarkers for detecting a specific disease, it wants to be able to predict disease with the value of the biomarkers. This is something inherently different as the question answered by the bivariate hypothesis test: Are the means (medians) for this biomarker different in each group? (Equation 2.2).

$$\begin{gathered} Group \sim Biomarker_i \\ vs. \\ Biomarker_i \sim Group \end{gathered} \tag{2.2}$$

In the first case, the response variable is the group and the predictors are the different biomarkers. In the second case, the responses are the different biomarkers and the predictor is the group. Since the approaches are different, it is not surprising that the fact that the means are different is not enough to achieve a good discrimination power between groups and, on the other hand, the fact that the means are equal does not mean that the biomarker is not appropriate for discriminating between groups. Figure 2.2 shows clear examples of both cases.

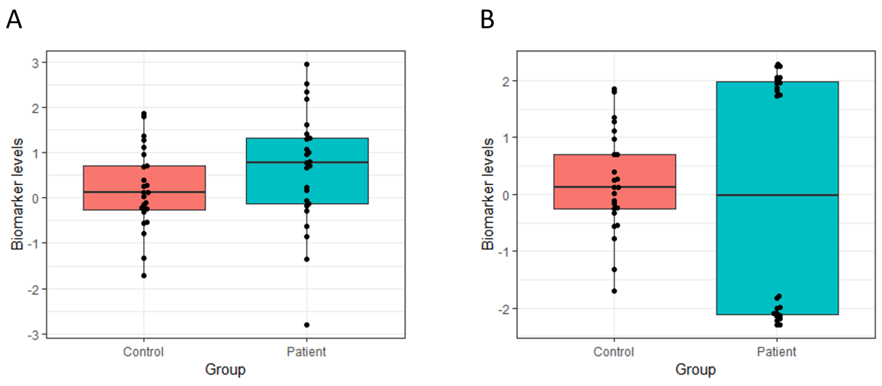

Figure 2.2: In **A** means are different (p < 0.001) but values of the biomarker are not valid for discriminationg between controls and patients. In **B** means are exactly equal, but values of the biomarker allow for a perfect discrimination between both groups.

In summary, while compelling for their simplicity and ease of use, classical bivariate tests are not appropriate for the analysis of omic data and should only be used in the context of exploratory analysis or complementary secondary analyses.

2.2 Linear modelling of omic data

Linear models are a generalization of the well known statistical tests *t test* and *ANOVA*, where more predictors can be included in the studied association with the response variable. In fact, the standard *t test* and *ANOVA* are just a linear regression with one categorical predictor (Figure 2.3)

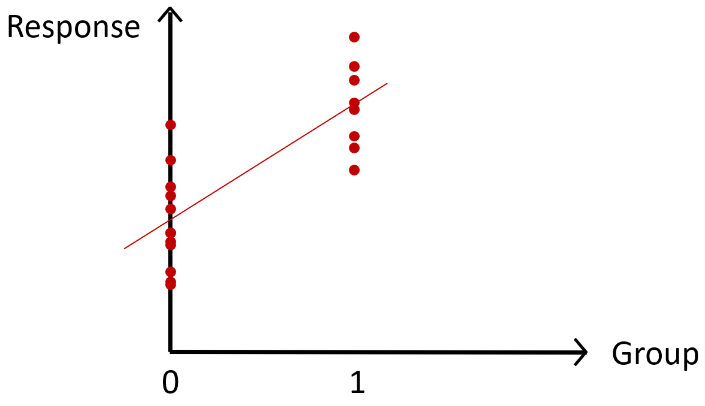

Figure 2.3: Equivalence between the *t test* and a linear regression with one categorical predictor. It is easy to notice that the slope of the regression line, defined as $\Delta y/\Delta x$, equals the difference between means from group 0 and group 1.

One advantage of linear models compared to their simpler versions *t test* and *ANOVA* is their ability to control for confounding variables. This overcomes one of the important issues raised in section 2.1 regarding non-comparable groups in observational studies. The linear model has the form showed in Equation 2.3. Where β_0 is the intercept and β_i are the slopes of the different variables included in the model. The linear model can also accommodate interactions by introducing a multiplicative term between variables.

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \epsilon \tag{2.3}$$

At first glance, it seems like the linear model could solve most of the problems presented in section 2.1, including integration of results, capturing interactions and answering the right question. One of the assumptions of linear models is that the response variable is continuous and has a linear relationship with the predictors, but this assumption can be relaxed using a well known generalization of the linear models known as generalized linear models (McCulloch 2000). This way, it would seem possible to perform regressions to predict or explain relationships of the different metabolites with a continuous variable, and also for discriminating between different groups just by including all studied metabolites in the model as covariates. Unfortunately, linear models cannot accommodate an unlimited number of variables, because they are limited by their degrees of freedom and suffer greatly from overfitting (Babyak 2004; Hawkins 2004). Thus, in the context of omic data analysis, linear models are only useful for the control of confounding variables. This means that they are used in a similar fashion as the bivariate tests (Equation 2.4). That is, performing as many regressions as predictor variables are in the data set.

$$Biomarker_i \sim Group + Confounder + \epsilon \tag{2.4}$$

2.3 Corrections for multiple comparisons

Each time a hypothesis test is performed, there is a α probability of a false positive (usually $\alpha = 0.05$). That means that, when using the techniques presented in section 2.1 and section 2.2, the probability of false positives grows to unacceptable levels. The relationship between the number of performed hypothesis tests and the overall probability of a false positive is presented in Equation 2.5.

$$Pr(False\ positive) = 1 - (1 - \alpha)^{n_{tests}} \tag{2.5}$$

This entails that, in a data set with just 100 variables, one expects five false positives in average. Noteworthy, metabolomic data sets can have up to thousands of variables producing docens or even hundreds of false positives (Broadhurst and Kell 2006). This renders the results of the performed hypothesis tests useless, because there is no way of discriminating between false positives and true positives.

2.3.1 Bonferroni correction

One method for dealing with the problem of the large number of false positives is the Bonferroni correction (Bland and Altman 1995). In this method, the α value for each comparison is set to α/n_{tests} in order to control the family wise error rate (the probability of making one or more false discoveries, when performing all the hypothesis tests). So this method is just setting a lower threshold for the significance level. The more hypotheses are tested, the lower is the significance level threshold. This is a very basic correction, and has a major drawback: it greatly increases the number of false negatives. It is straightforward to see that only with 20 tests, the significance level is lowered to $0.05/20 = 0.0025$, so only medium to large effects will be detected assuming that sample size is large enough. But, in the case of metabolomic studies, data sets tend to have many more variables. With 500 variables, the significance level would be $0.05/500 = 0.0001$ and with 2000 (typical for an untargeted analysis) it would be $0.05/2000 = 0.000025$. Taking into account the limited sample sizes present in metabolomic studies, only the hugest effects will be detected at such a low significance level (Johnson et al. 2010).

2.3.2 False Discovery Rate

As explained in the previous paragraph, Bonferroni correction greatly increases the number of false negatives. The explanation for this increase can be deduced from the pseudo formula presented in Equation 2.6. In this pseudo formula, both error types are at the same place of the equation and the other parameters are fixed for a specific data set. This means that, when one decreases one of the two error types, the other has to increase accordingly.

$$N \sim \frac{variance}{effect\ size + Type\ I\ error + Type\ II\ error} \tag{2.6}$$

In the case of Bonferroni correction, the lowering of type I error is huge, so the increase in type II error is correspondingly large. In practice, this usually means that the analysis yields no positive results after a Bonferroni correction. To overcome this problem, an alternative, less aggressive correction named False Discovery Rate was developed (Benjamini and Hochberg 1995). The false discovery rate method is not concerned with the family wise error rate (FWER) as the Bonferroni method. It is focused in controlling the number of false positives among all the discovered positives. So, in the case of false discovery rate, it is assumed that some false positives are going to happen, but one wants to control their proportion in relation with all the positives. The difference between this approach and the classical control of type I error is explained in Table 2.1.

	H_0 is true	**H_0 is false**
Rejected H_0	FP	TP
Not rejected H_0	TN	FN

Table 2.1: Difference between FWER and FDR. FP, TP, TN and FN stand for false positives, true positives, true negatives and false negatives, respectively. In the classic FWER procedures such as Bonferroni, one is interested in maintaining the overall proportion of false positives under a specific threshold α. In FDR, on the other hand, the interest is in maintaining the ratio of FP/(FP+TP) under a specific threshold.

It is easily shown that controlling the ratio FP/(FP+TP) is less stringent than controlling the overall proportion of false positives. Thus, FDR has greater statistical power, although it has a larger amount of type I error compared to Bonferroni. This is expected since FDR is also subject to the formula presented in Equation 2.6.

2.3.3 Other corrections

As explained in the preceding paragraph, there is no way out of the formula presented in Equation 2.6 when performing hypothesis tests. Several alternative methods have been provided during the last decades (Benjamini and Yekutieli 2001; Gao, Starmer, and Martin 2008; Castro-Conde and Uña-Álvarez 2015), but all of them are just playing with the balance between type I and type II error rate. Some researchers even support that the loss of power is so high, even when performing false discovery rate, that in many cases it would be better to not correct at all and consider the results as exploratory or hypotheses generators instead of forcing an unacceptable low power for confirming hypotheses (Bender and Lange 2001). In summary, hypothesis testing for the analysis of metabolomic data suffers from many issues that are difficult or, in some cases, impossible to overcome. From the exposed methods in this section, the linear modelling approach seems a promising tool, specially when considering its ability to use the different metabolites as predictors. However, issues concerning overfitting and limited number of variables, due to the degrees of freedom, should be considered when using this approach.

2.4 Principal Component analysis

It has already been exposed that linear models could be an appropriate solution for analyzing metabolomic data sets. One of the important issues concerning their use is the saturation of their degrees of freedom when the number of variables is large. Principal component analysis (PCA) is a dimensionality reduction technique, that can be used to reduce the number of variables prior to modeling, thus helping to overcome one of the problems of linear models in the context of high dimensionality data sets (Hotelling 1933; K. Wang and Abbott 2008). An illustration of the dimensionality reduction capability of PCA is provided in Figure 2.4.

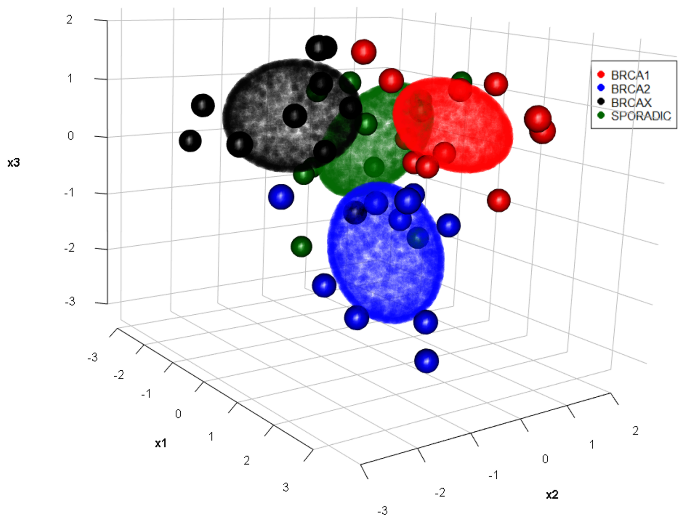

Figure 2.4: Plot of the first three principal components of a miRNA dataset. These components carry enough information to be able to discriminate four different groups of samples.

Formally, PCA is defined as an orthogonal linear transformation that projects the data in a new coordinate system such that the greatest variance of the data lies in the first dimension, the second greatest variance in the second dimension, and so on until the number of components is equal to the number of original dimensions. The full principal components decomposition of $\mathbf{X}$ is given by (Equation 2.7).

$$\mathbf{T} = \mathbf{X}\mathbf{P} \tag{2.7}$$

Where $\mathbf{P}$ is a $p \times p$ matrix of loadings whose columns are the eigenvectors of $\mathbf{X}^T\mathbf{X}$. Since the last components are the ones accounting for the least part of the variability, they carry almost no information from the original data matrix. Thus, it is reasonable to remove them keeping only the first components and effectively reducing the dimensionality of the data. Principal component regression (PCR) then uses these first components as predictors in a linear regression model to predict the response (Jolliffe 1982).

Another common use of PCA is as an exploratory analysis method (F. S. Tsai and Chan 2007). Since it is very hard to represent more than three dimensions in a surface such as a sheet of paper or a screen, multidimensional data sets such as those coming from 'omic' technologies are difficult to visualize. The dimensionality reduction produced by PCA allows for the representation of such data sets and the performing of exploratory analyses such as outlier detection, batch effect assessment, etc. (Meglen 1992).

Chapter 3

Modern techniques for the analysis of metabolomic data

3.1 Projection based methods

In the previous chapter, the projection technique known as PCA has been introduced. As explained in section 2.4, this method works by performing a dimension reduction to an original data matrix with dimensions $I \times J$, to a projected data matrix with dimensions $I \times M$, where $M \leq J$. Each new component of the projected data matrix is built trying to capture as much of the variability from the original data matrix as possible. So PCA is a projection technique that tries to reduce the dimensionality of the data while trying to maximize the information kept from the original data. This approach can be great for performing exploratory analysis of a data set or unsupervised analysis, but can be inefficient for performing supervised analysis such as regression or classification. The reason for this is that, while variables with the highest variability carry most of the information of the original data matrix, they do not necessarily have to be the ones able to explain the response that one is trying to predict (Kettaneh, Berglund, and Wold 2005). A conceptual explanation of this is presented in Figure 3.1.

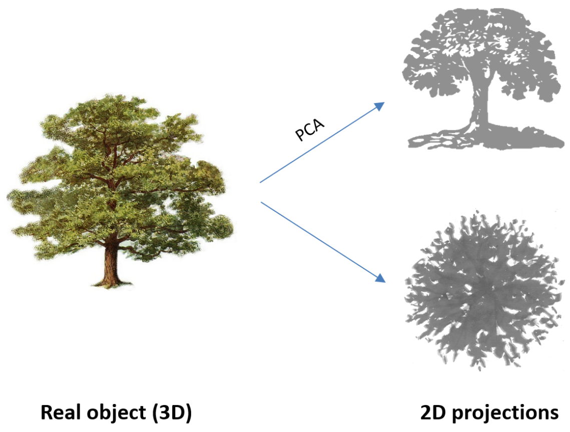

Figure 3.1: Variables with less variability can be the ones related to the response, thus being infrarepresented in the first components of PCA. In this example, if one wanted to predict the amount of shadow dropped by the tree, the PCA projection would not be efficient, while the other presented projection fully captures the target prediction. On the other hand, the PCA projection makes the tree fully recognizable, while the other projection makes it hard to even guess there is a tree.

Thus, if the aim is prediction, projection for maximizing the explained variability on the original data matrix is not the best approach. It is much better to maximize covariance with the response. This is how the technique Partial Least Squares (PLS) works.

3.1.1 Partial Least Squares

In linear regression, there is a limit on the number of variables that can enter a model for a specific sample size. Recent studies have shown that a minimum of two independent observations per variable are needed for estimating regression coefficients with reasonable bias (Austin and Steyerberg 2015). Metabolomic data sets usually have many more variables than observations, so linear regression models are not a viable alternative for analyzing these data sets. Linear regression also suffers from multicollinearity, so when highly correlated predictors are used together in a model their coefficients get unstable and standard errors grow wildly (Alin 2010). PLS can be seen as an extension of multiple linear regression designed to overcome the mentioned drawbacks. It can analyze data sets with strongly correlated variables and handle situations where the number of variables far exceeds the number of observations (Wold, Sjöström, and Eriksson 2001). Additionally, PLS can also simultaneously model several response variables. It is important to note that results of PLS (and of projection methods in general) depend on the scaling of the data so, before the analyses, pre-processing of the **X** and **Y** variables should be considered. A scheme of the PLS model is depicted in Figure 3.2.

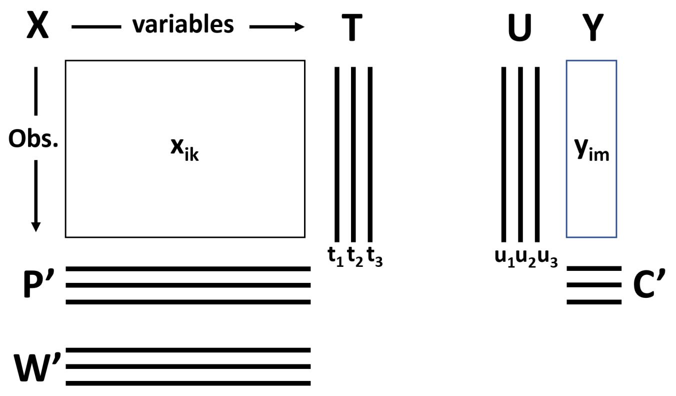

Figure 3.2: Scheme of the PLS model (Wold, Sjöström, and Eriksson 2001). **X** is the matrix of predictors and **Y** is the matrix of responses. Additional matrices are generated at the different modelling steps.

The PLS model finds latent variables (denoted by t_a, where a = 1, 2, . . . , A) that are predictors of **Y** (with dimensions $I \times M$) and also model **X** (with dimensions $I \times K$), by maximizing the covariance between their latent structures. These latent variables are also called X-scores and are orthogonal. They are estimated as linear combinations of the original variables using different weights denoted by w^*_{ka} (Equation 3.1).

$$\mathbf{T} = \mathbf{X}\mathbf{W}^* \tag{3.1}$$

X-scores are multiplied by the loadings p_{ak} in order to obtain the residuals in **X**, denoted as **E** (Equation 3.2).

$$\mathbf{X} = \mathbf{T}\mathbf{P}' + \mathbf{E} \tag{3.2}$$

Analogously, the Y-scores are multiplied by the weights (denoted by c_{am}) minimizing residuals in **Y**, denoted as **G** (Equation 3.3).

$$\mathbf{Y} = \mathbf{U}\mathbf{C}' + \mathbf{G} \tag{3.3}$$

As stated before, X-scores are good predictors of **Y** (Equation 3.4).

$$\mathbf{Y} = \mathbf{T}\mathbf{C}' + \mathbf{F} \tag{3.4}$$

Where **F** is the residuals matrix from **Y**. Alternatively, Equation 3.4 can be rewritten as follows (Equation 3.5):

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^{*}\mathbf{C}' + \mathbf{F} = \mathbf{X}\mathbf{B} + \mathbf{F} \tag{3.5}$$

Which resembles a multiple linear regression model where the predictors matrix **X** is multiplied by the coefficients matrix **B** to predict **Y**.

PLS is, consequently, a modelling technique able to effectively deal with metabolomic data. Correlated variables and data sets with many more variables than observations can be properly analyzed and results are easily interpreted with the help of useful plots of the scores, weights and predictions of the model (Figure 3.3).

3.2 Penalization methods

Another set of techniques able to manage metabolomic data sets are the penalization methods. These methods are motivated by the relationship between model complexity and prediction error, also known as bias-variance trade-off (Hastie, Friedman, and R. Tibshirani 2001). As depicted in Figure 3.4, prediction error in the same data set used to fit a specific model always decreases when increasing the complexity of the model. On the other hand, prediction error in new data decreases at first, reaching a minimum at a specific model complexity, and increases later again as model complexity keeps growing larger.

As previously described, metabolomic data sets are characterized by having a large number of variables and relatively a low number of observations. Trying to fit a model with all the variables in this situation would correspond to the extreme right side of the plot in Figure 3.4 with low bias and high variance. This is the reason why multiple regression models cannot be fit to these data, variance grows to infinite as the number of predictor variables increases in the model. But this relationship between bias-variance and model complexity

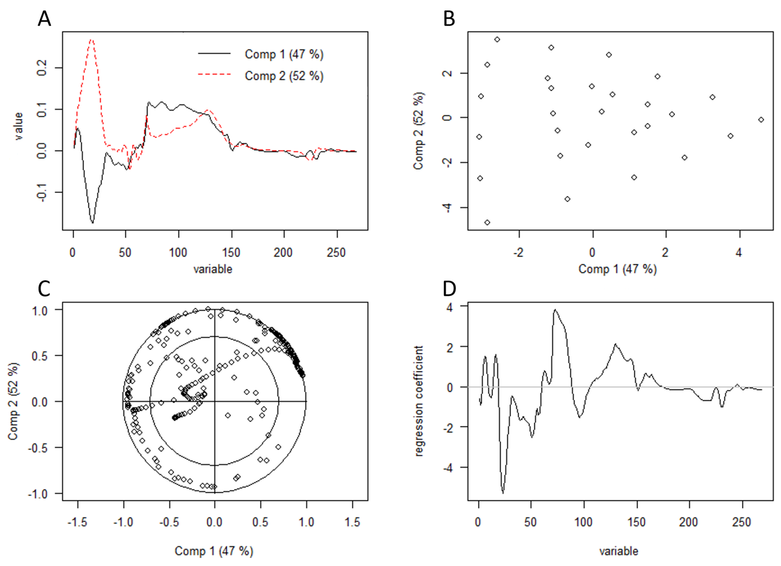

Figure 3.3: Plots of a PLS model. **A** shows the weights for each variable in the two first components. **B** shows the scores for each observation in the first two components. **C** represents the correlations between each variable and the first two components. **D** shows the regression coefficients of the model.

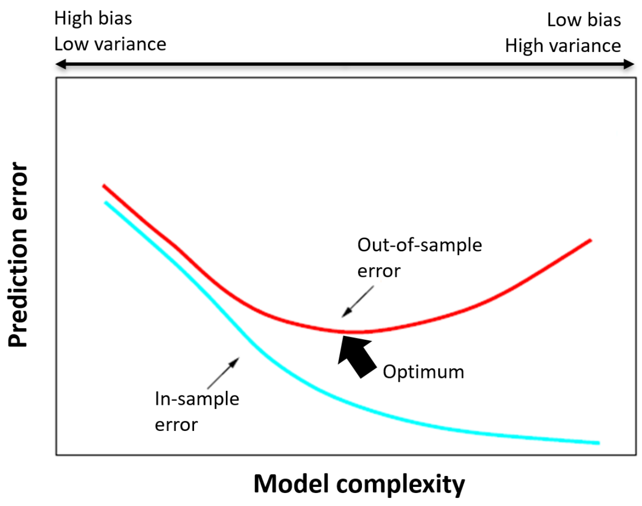

Figure 3.4: Behavior of prediction error in training (in-sample) and test (out-of-sample) data.

gives us the key for improving the standard linear models: increasing bias will reduce the variance, potentially obtaining a lower out-of-sample prediction error because model complexity has been decreased. Thus, the solution for being able to fit linear models to metabolomic data is to bias them. The full reasoning behind this last sentence can be better understood with the explanation provided in Figure 3.5.

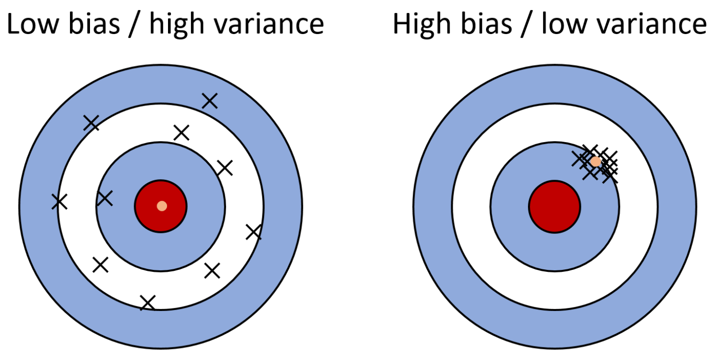

Figure 3.5: In both targets each black cross represents an estimation of a specific model to different samples of the same population and the orange dot represents the average estimation of all of them. In the left target there is no bias, with the orange dot right at the bullseye, but estimation from the different samples are wildly spread with a very high variability. In the right target, there is a perceptible amount of bias, the orange dot lies out of the bullseye, but all estimations from the different samples are concentrated in a small area around the average estimation. Usually, only one sample (one data set) is available for performing the estimation, so the advantage of introducing bias to lower variance is evident.

3.2.1 Ridge regression

The first developed penalization method for linear regression was ridge regression (Hoerl and Kennard 1970). This method consists in solving the ordinary least squares problem, but introduces a restriction to the solution. This restriction consists in setting an upper bound on the sum of the squared coefficients (L2 norm) (Equation 3.6).

$$\begin{gathered}\hat{\beta}^{ridge} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \\ \text{Subject to the restriction: } \sum_{j=1}^{p} \beta_j^2 \leq s\end{gathered} \tag{3.6}$$

The inclusion of this restriction to the ordinary least squares problem imposes a reduction in the value of the coefficients, thus biasing their estimation and producing a reduction in the complexity of the model. A graphical explanation of the functioning of the method is represented in Figure 3.6.

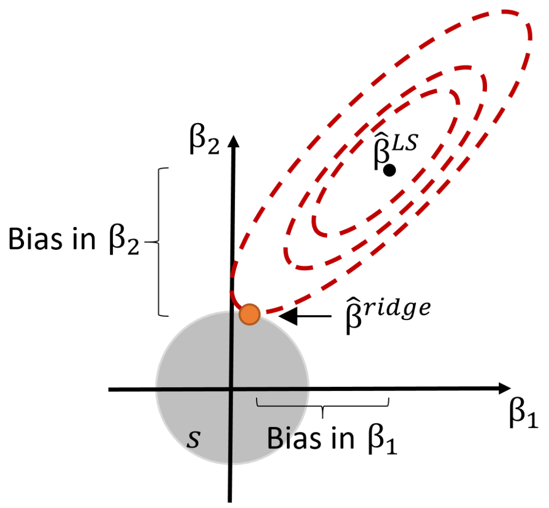

Figure 3.6: Ridge regression for a linear model with two predictors. As the restriction is intensified (lower value of s), the estimation of the coefficients is more biased to zero. Dashed red lines represent least squares contour lines. Grayed area represents the restriction imposed by s.

The solution offered by ridge regression is very similar to that obtained by PLS. A regression model suitable for metabolomic data sets with many, highly correlated variables. Both methods use all the variables for adjusting the model and both models shrink the estimates of the coefficients (De Jong 1995), thus reducing model complexity and therefore variance at a cost of increasing bias.

3.2.2 Lasso

In many studies there is the need or interest of not only fitting a predictive model, but also selecting the most important predictors among all the variables present in the data. As it has been exposed in the previous sections, neither PLS nor ridge regression provide a direct method for variable selection. They both use all the variables in the data set for adjusting the model. It is true that there are some post-model-fitting methods for assessing variable importance in the case of PLS (Mehmood et al. 2012), but they require and additional filtering step after the model has been fitted and are not inherent to the PLS model. The method known as lasso is similar to ridge regression. But instead of setting a bound on the sum of the squared coefficients, it is set on the sum of the absolute values of the coefficients (L1 norm) as seen in Equation 3.7 (R. Tibshirani 1996). This change in the method of penalization of the coefficients allows that some of them are set to zero when the model is adjusted, thus producing a variable selection at the model-fitting step.

$$\begin{gathered}\hat{\beta}^{lasso} = \arg\min_{\beta} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \\ \text{Subject to the restriction: } \sum_{j=1}^{p} |\beta_j| \leq s\end{gathered} \tag{3.7}$$

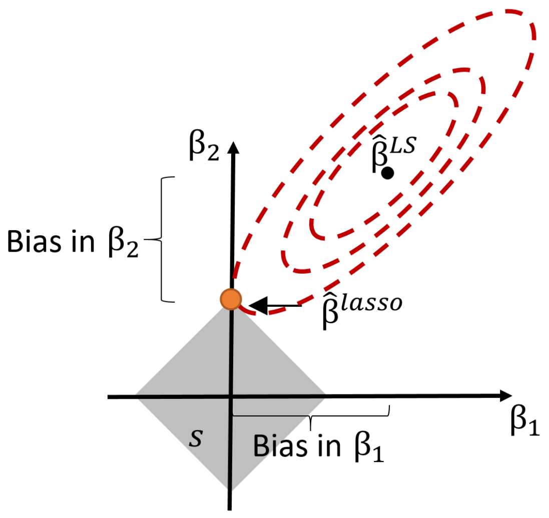

Figure 3.7: Lasso regression for a linear model with two predictors. As the restriction is intensified (lower value of s), the estimation of the coefficients is more biased to zero. Since it is more easier to land in a vertex than in an edge, reducing s forces one of the coefficients to zero. Dashed red lines represent least squares contour lines. Grayed area represents the restriction imposed by s.

Figure 3.7 shows how increasing the penalization by reducing s forces the parameters to zero, producing a simpler model by deselecting some features. Thus, assuming data are standardized (i.e., mean centered and scaled to unit variance), Lasso automatically selects the most relevant features and discards the others.

The fact that lasso allows for variable selection makes it a very appealing solution for modelling 'omic' data. Unfortunately, unlike PLS and ridge regression, lasso is not able to deal with multicollinearity in an optimal manner (Chong and Jun 2005), which is a prominent characteristic of metabolimic data. When two (or more) variables are highly correlated, lasso will select one of them discarding the others and, potentially, losing predictive performance and important variables.

3.2.3 Elastic net

Two different penalization methods have been presented, each one with its own benefits and drawbacks. Ridge is able to deal with multicollinearity but does not perform variable selection and lasso performs variable selection but does not deal adequately with multicollinearity. The elastic net algorithm (EN) was developed in order to merge the good qualities of ridge and elastic net and overcome their limitations (Zou and Hastie 2005). EN is a flexible combination of the two constrains defined for ridge and lasso. The L1 penalty provides variable selection and the L2 penalty provides stability in the selection of correlated variables. The balance between both penalties is defined by the parameter α (Equation 3.8). When $\alpha = 0$, elastic net is equivalent to ridge regression, since the L1 constraint is set to zero. On the other hand, when $\alpha = 1$, elastic net is equivalent to lasso, since the L2 constraint is set to zero. Values between $\alpha = 0$ and $\alpha = 1$ provide different balances between L1 (more variable selection) and L2 (better handling of multicollinearity) penalizations.

$$\begin{gathered}\hat{\beta}^{elasticnet} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \\ \text{Subject to the restriction: } \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \leq s\end{gathered} \tag{3.8}$$

Figure 3.8 depicts the elastic net in a two-variable case and shows how, while keeping the vertices present in lasso, the edges are convex and allow for grouped selection of correlated variables. This is a very attractive solution for the modelling of metabolomic data sets and has been used widely during the last years (Lankinen et al. 2010; Bowling and Thomas 2014; X. Liu et al. 2015; Ferrario et al. 2016). The only drawback of elastic net compared to lasso and ridge is the inclusion of another parameter in the model. Instead of optimizing only for the value of s, elastic net models have also to optimize for the value of α, thus requiring more tuning for an appropriate fitting of the data.

Penalization methods have been used by the author of this thesis for the analysis of different 'omic' data sets with very positive results (Yáñez et al. 2015; Gonzalez-Billalabeitia et al. 2017; Ibáñez et al. 2018).

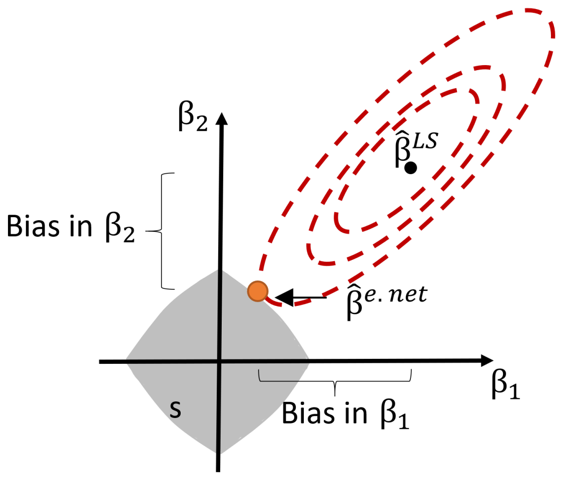

Figure 3.8: Elastic net regression for a linear model with two predictors. As the restriction is intensified (lower value of s), the estimation of the coefficients is more biased to zero. As in lasso, it is more easier to land in a vertex than in an edge, so many coefficients can be potentially set to zero. But, as in ridge, edges are convex, so highly correlated variables will be selected together. Dashed red lines represent least squares contour lines. Grayed area represents the restriction imposed by s.

3.3 Other methods

Projection based methods and penalization methods are extensions of the standard linear model and can also be extended to generalized linear models and survival models without much effort (Nygård et al. 2008; Simon et al. 2011). But there are other methods worth mentioning when thinking about dealing with metabolomic data sets. Among them stand out neural networks (Dayhoff and DeLeo 2001), which are highly parametrized models inspired by the functioning of the human brain, support vector machines (Mahadevan et al. 2008) based on kernel methods, bayesian networks (Bartel, Krumsiek, and Theis 2013) and tree based methods such as regression and classification trees (Venables and Ripley 2002; Loh 2014), Figure 3.9, random forests (Breiman 2001) (based on bootstrap resampling (Efron and R. J. Tibshirani 1994) of classification trees), and boosting (Freund, Schapire, et al. 1996; Auret and Aldrich 2011).

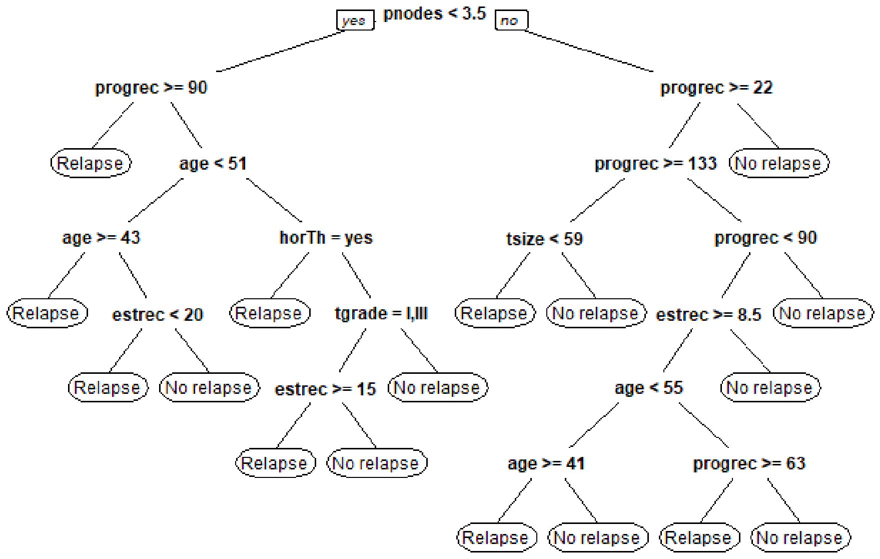

Figure 3.9: Classification tree for predicting relapse in breast cancer patients. Starting from the top one, each node tests a boolean condition which determines the direction of the next node to test until a terminal node is reached and a prediction is made.

Chapter 4

Challenges for analyzing metabolomics data. Strategies and aims of the thesis

4.1 Challenges for analyzing metabolomic data

Traditionally, 'omic' data analyses have been performed using classical statistical methods from last century. In most cases, these methods consisted in bivariate tests such as *t test* or *Wilcoxon-Mann Whitney test*, followed by some sort of multiple comparissons correction such as *Bonferroni* or *False Discovery Rate* (Hochberg and Benjamini 1990; Benjamini and Hochberg 1995). These approaches suffer from several drawbacks including lack of statistical power, lack of interpretability of results, and omission of complex relationships among variables (Strasak et al. 2007). Thus, classical statistical methods based on bivariate tests are not adequate to extract all the information available in 'omic' data sets. Some of the mentioned problems such as the lack of interpretability or the omission of complex relationships could be adressed using statistical models such as linear models or generalized linear models, but these methods also suffer from other problems when dealing with 'omic' data, such as the large number of variables and the low sample size, which produces overfitting, and the high correlation among variables, which produces multicollinearity. All these limitations associated with 'omic' data analysis have motivated the development of numerous novel statistical techniques during the last decades specifically aimed at solving them. Prediction methods such as PLS (Wold, Ruhe, et al. 1984), lasso (R. Tibshirani 1996), elastic net (Zou and Hastie 2005) and random forest (Breiman 2001), among others, can deal with most of the problems exposed in the preceding lines. Nevertheless, other challenges remain partially unresolved, such as the analysis of repeated measures data or, more generally, data organized in multiway arrays. Also, in a discipline where the number of variables is so large, the development of variable selection methods is of utmost importance.

4.2 Strategies and aims of the thesis

As presented in the previous sections, 'omic' data sets and, more specifically, metabolomic data sets contain a large amount of information in the form of huge data structures. The magnitude of this data sets dificults their handling as well as their analysis and interpretation. Additionally, metabolomics presents specific singularities that make necessary the use of complex pre-processing and analysis techniques. First, depending on the technology used, baseline correction, phasing, peak alignment and binning in the case of NMR or peak detection, aligmnment and gap filling in the case of MS have to be performed on the raw data. Then, different normalization and data pretreatments methods are applied to the data depending on the aims of the study (Karaman 2017).

The aim of metabolomic analyses is the identification of those metabolites related to the specific biological question being studied. As of today, the majority of statistical analyses performed on metabolomic data sets are based on classical methods or, in the best cases, on adequate but limited modern statistical methods. Among these adequate methods stand out principal component analysis (PCA), partial least squares (PLS), penalization methods such as ridge or lasso and other machine learning techniques such as random forest or boosting. However, these techniques are not able no deal effectively with all the particularities of metabolomic data sets. Their limitations get clearly exposed when dealing with data sets with repeated measurements over time or, more generally, three-way or multi-way data sets. An important part of the work on this thesis consisted on critically reviewing an assessing the usefulness of the different presented tools for metabolomic data analysis.

Some useful tools for analyzing three-way data are the Tucker3 and PARAFAC models and, when some **Y** data structure is to be predicted, the N-PLS model. Related to the problem of these datasets with a large number of features, comes the issue of variable selection. Variable selection is essential for facilitating e.g. biological interpretation of the results when analyzing '-omic' data sets. It is often the case that the aim of these analyses is to find a new biomarker or a specific set of biomarkers, also called signature, to diagnose or predict the onset of a disease. For these cases, the N-PLS algorithm does not provide variable selection implemented within the algorithm, although some methods have been developed to perform it after the fitting of the model. One of the main research lines of this thesis will be the introduction of L1-penalization in the N-PLS algorithm to allow for variable selection within the model-fitting step. This approach should not only facilitate interpretation by producing a reduced model including fewer variables, but should also reduce prediction error by completely eliminating noise features instead of downweighting them as N-PLS does. The full list of objectives for this thesis is presented below:

- Try, assess and evaluate different preprocessing techniques for metabolomic data
- Study and critically evaluate the use of different modern data analysis techniques for dealing with metabolomic data sets.
- Develop novel algorithms for embedding variable selection methods with N-PLS for the analysis of three- or multi-way datasets that allow for variable selection at the model-fitting step
- Develop a software package based in the R language, that implements the new algorithm along with all the extra functions needed for performing complete analyses of three-way data sets and their posterior interpretation.

Part II

Materials and methods

Chapter 5

Multiway data, methods for its analysis

5.1 From 2D matrices to 3D arrays

In most fields, data sets are usually represented by two dimensional matrices, where the rows are the observations and the columns are the different variables ($I \times J$). But in metabolomics, and in general in many other 'omic' sciences, sometimes these $I \times J$ data sets can be expanded by taking, for example, repeated measurements at different K time points for each observation and producing a three-dimensional data structure ($I \times J \times K$) called three-way array (Figure 5.1) or, more generally, a multidimensional data structure called multi-way array (Kroonenberg et al. 2016). Most of the modelling methods that have been explained in chapter 3 are not able of dealing with such data structures, which raise many methodological complications to the analyses. Therefore, different three-way or multi-way specific methods have been developed during the last decades to deal with this sort of data.

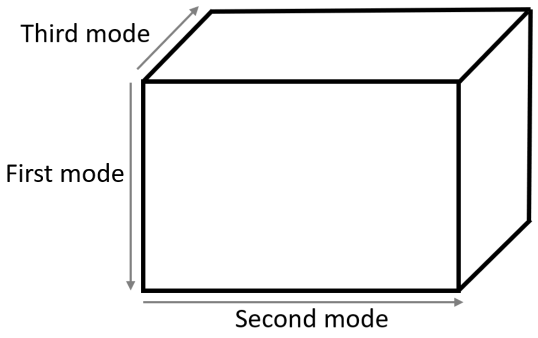

Figure 5.1: Representation of a three-way array.

5.2 Exploration and description of N-way arrays

5.2.1 Unfolding

One of the straightforward solutions for dealing with N-way arrays is performing an unfolding of the array into a two-dimensional matrix. This unfolding is performed by regrouping the elements of the higher order dimensions, also called modes, by extending the dimension of the second mode, as showed in Figure 5.2. After the unfolding, all statistical methods valid for two-dimensional matrices are applicable to the new structure, although the tridimensional structure of the data is lost, leading to a list of potential issues such as more complex models (too many parameters), loss of predictive power, loss of the multi-way information and greater difficulty in the interpretation of results.

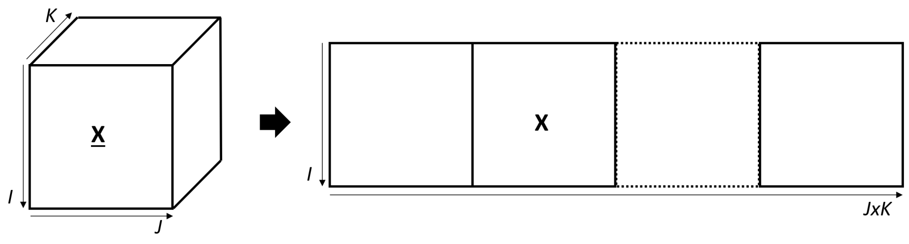

Figure 5.2: Unfolding of a three-way array ($I \times J \times K$) into a two-dimensional matrix ($I \times JK$).

5.2.2 Tucker3

When studying multi-way structures, the most general model that can be used is the Tucker3 model (Tucker 1966), also explained in detail by Kiers and Mechelen (2001).

Assuming a three-way structure of the data, the Tucker3 model has the structure defined in Figure 5.3. This figure shows that the model is a weighted sum of all possible outer products, where the loadings of the outer product between the ith factor from $\mathbf{A}$, the jth factor from $\mathbf{B}$ and the kth factor from $\mathbf{C}$ is determined by element g_{ijk} of the core.

Model parameters are estimated by minimizing the squared sum of residuals e_{lmn} as shown in Equation 5.1

$$x_{ijk} = \sum_{l=1}^{w_1} \sum_{m=1}^{w_2} \sum_{n=1}^{w_3} a_{il} b_{jm} c_{kn} + e_{lmn} \tag{5.1}$$

If the Kronecker product is used, defined as $\otimes$, unfolding the $\underline{\mathbf{X}}$ array, the $\underline{\mathbf{G}}$ core and the residuals $\underline{\mathbf{E}}$ into matrices $\mathbf{X}$, $\mathbf{G}$ and $\mathbf{E}$ by fixing the first mode, the matrix expression of the model is as follows (Equation 5.2):

$$\mathbf{X} = \mathbf{AG}(\mathbf{C}^T \otimes \mathbf{B}^T) + \mathbf{E} \tag{5.2}$$

where $\mathbf{A}$ is the loadings matrix corresponding to the first mode, $\mathbf{G}$ is the matrix obtained after unfolding the $\underline{\mathbf{G}}$ core fixing the first mode, $\mathbf{B}$ and $\mathbf{C}$ are the loading matrices of the second and third mode respectively and $\mathbf{E}$ is the residuals matrix. The fact that the number of components is not forced to be the same in the different modes allows for the different loadings matrices to have different dimensions for each mode.

Of note, the compression performed by the model can be applied also to the first mode. This way, the $\underline{\mathbf{G}}$ core can be considered as an approximation to $\underline{\mathbf{X}}$, which approximates the original array by the three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$.

It is important to take into account that the model has not a unique solution, because it has rotational freedom. In fact, the model can be so redundant that many elements of $\underline{\mathbf{G}}$ can be set to zero without altering the fit of the model, thus meaning they are associated to noise (José Manuel Prats-Montalbán 2005).

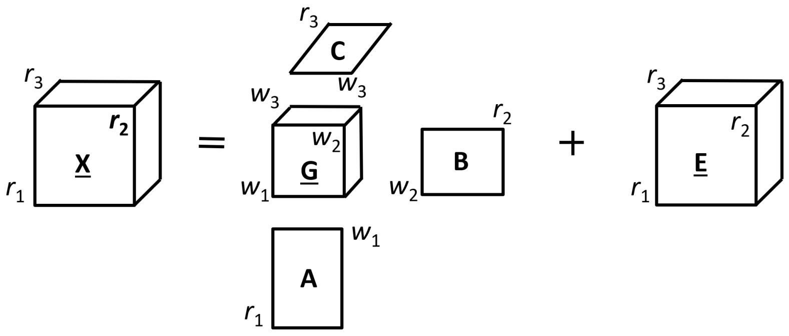

Figure 5.3: Scheme of the Tucker3 model. The model consists on a weighted sum of outer products between the different factors stored in matrices **A**, **B** and **C**.

5.2.3 PARAFAC

The PARAFAC method is also a decomposition method for multi-way arrays. It can be derived from the Tucker 3 model by imposing superdiagonallity and identity in the structure of the $\underline{\mathbf{G}}$ core. Essentially, it is a generalization of PCA to three- or multi-way data structures (Bro 1997). The method was first developed by Harshman (1970) and Carroll and Chang (1970), who named the method as CANDECOMP. In this method, the decomposition of the data is made into trilinear components, with each component consisting of one score vector and two loading vectors. The general structure of the PARAFAC model is presented in Figure 5.4.

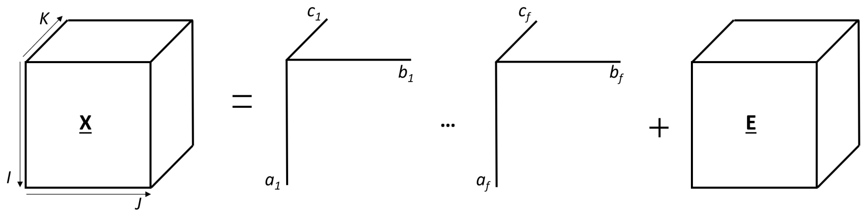

Figure 5.4: General structure of a PARAFAC model.

A PARAFAC model of a three-way array is given by three loading matrices, **A**, **B** and **C** with elements a_{if}, b_{if} and c_{kf}. The model minimizes the sum of squares of residuals in Equation 5.3.

$$x_{ijk} = \sum_{f=1}^{F} a_{ij} b_{jf} c_{kf} + e_{ijk} \tag{5.3}$$

The matricial expression of the PARAFAC model can be obtained by Using the Khatri-Rao product (S. Liu and Trenkler 2008) and by unfolding the $\underline{\mathbf{X}}$ array into matrix $\mathbf{X}$ (Equation 5.4).

$$\mathbf{X} = \mathbf{A}(\mathbf{C}| \otimes |\mathbf{B})^T + \mathbf{E} = \sum_{f=1}^{F} a_f (c_f^T \otimes b_f^T) + \mathbf{E} \tag{5.4}$$

In the PARAFAC model, a_f, b_f and c_f are the fth columns of the loading matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, respectively.

The main advantage of PARAFAC over Tucker3 models is the impossibility of alternative solutions, that is, there is no posibility of rotation. This makes PARAFAC models more easily interpretable.

5.3 Regression methods for *N*-way arrays, *N*-PLS

The N-PLS model is an extension of the standard PLS model to multi-way arrays based in the decomposition performed by the PARAFAC model (Bro 1996). Its aim is studying relationships between some three-way (or N-way) $\underline{\mathbf{X}}$ (e.g. $I \times J \times K$) data structure and any $\underline{\mathbf{Y}}$ (e.g. $I \times L \times M$) data structure by maximizing the covariance between $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ data arrays. For achieving this, a multilinear model is fitted simultaneously to the $\underline{\mathbf{X}}$ and the $\underline{\mathbf{Y}}$ arrays with a regression model relating them together. The structure of the N-PLS model is presented in Figure 5.5.

Considering $\mathbf{X}$ ($I \times JK$) the unfolded version of $\underline{\mathbf{X}}$, N-PLS tries to find latent spaces $\mathbf{W}^J$ and $\mathbf{W}^K$ that maximize the covariance between $\mathbf{X}$ and $\mathbf{Y}$, so it can be expressed as:

$$\mathbf{X} = \mathbf{T}(\mathbf{W}^K| \otimes |\mathbf{W}^J)^T + \mathbf{R} \tag{5.5}$$

Afterwards decomposing $\underline{\mathbf{X}}$ from $\mathbf{X}$ using the improved N-PLS version expression (Bro, A. K. Smilde, and Jong 2001), in order to obtain residuals with better statistical properties:

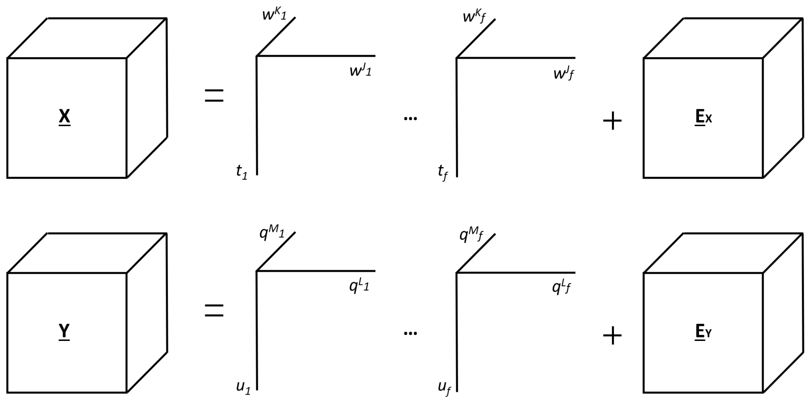

Figure 5.5: General structure of a N-PLS model.

$$\mathbf{X} = \mathbf{TGu}(\mathbf{W}^K \otimes \mathbf{W}^J)^T + \mathbf{R'} \tag{5.6}$$

In the same way, $\underline{\mathbf{Y}}$ can be decomposed by unfolding $\underline{\mathbf{Y}}$ $(I \times L \times M)$ into $\mathbf{Y}$ $(I \times LM)$ as:

$$\mathbf{Y} = \mathbf{U}(\mathbf{Q}^M | \otimes | \mathbf{Q}^L)^T + \mathbf{R''} \tag{5.7}$$

In this case, $\mathbf{W}^K$ and $\mathbf{W}^J$ refer to the weights of the third and of the second mode, respectively; whereas $\mathbf{T}$ matrix gathers the scores of the samples at each component extracted, in the first mode. $|\otimes|$ is the Khatri-Rao product and $\otimes$ the Kronecker product, which forbid or allow (respectively) to take interactions between the different modes components into account. $\mathbf{Gu}$ is the core array (unfolded) of a Tucker3 decomposition when using $\mathbf{T}$, $\mathbf{W}^K$ and $\mathbf{W}^J$ as loadings, in order to obtain a better (or at least not worse) approximation of the $\underline{\mathbf{X}}$ array (A. Smilde, Bro, and Geladi 2005). Finally, $\mathbf{R'}$ incorporates the residuals. Analogously, $\mathbf{U}$ refers to the $\mathbf{Y}$ scores, and $\mathbf{Q}^M$ and $\mathbf{Q}^L$ to the loadings of the array $\underline{\mathbf{Y}}$. Finally, from the scores $\mathbf{T}$ and $\mathbf{U}$, as well from the $\mathbf{W}$ weights, a $\mathbf{B}_{PLS}$ regression matrix can be obtained (Bro 1998) so

$$\mathbf{Y} = \mathbf{XB}_{PLS} + \mathbf{R'''} \tag{5.8}$$

In summary, N-PLS allows for the application of the well known standard PLS model to multi-way arrays, being able to exploit the multidimensional structure of the data, reducing the inclusion of noise, and producing more parsimonious models than standard PLS models applied to unfolded matrices.

5.4 Software for analyzing three-way data

5.4.1 R programming language

R (Ihaka and Gentleman 1996; R Core Team 2018) is a programming language designed for statistical programming, derived from the S language, which was developed in the Bell Laboratories by Rick Becker, John Chambers and Allan Wilks in the 60s. It is free software and open source, and was designed specifically for performing data analysis and statistical tasks. Since it is open source, R has thousands of developers, which results in more than 10000 packages in the main software repository for R named *CRAN* (Figure 5.6) and more than 1300 packages in the *Bioconductor* repository, specialized in packages for the analysis of 'omic' data. For this reasons, R has become a standard for data analysis in many scientific fields such as physics, biology and medicine among others (Goztepe 2017).

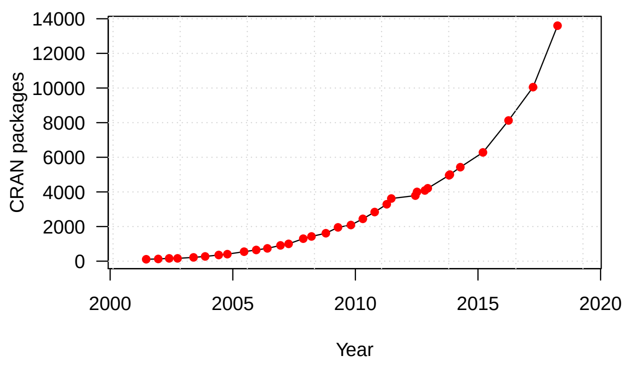

Figure 5.6: Number of R packages on CRAN has been growing at an exponential rate since 2002.

With the huge number of packages at the disposal of `R` users, many advanced functions have already been programmed and are available for their use or reuse when programming new functionalities. Apart from the main repository (CRAN) there are other important repositories with thousands of packages more such as *R-Forge*, *Bioconductor* or even *Github*. All this functionallity and the fact that R is open source motivated the election of `R` as the language for developing all the techniques and tools presented in this thesis.

5.4.2 `R` packages for the analysis of three-way data

Most three-way data procedures are implemented in MATLAB® (MATLAB 2018) (or previous versions of the software). Concretely, the N-way toolbox by Andersson and Bro (2000) provides a comprehensive set of functions for dealing with three- and multi-way arrays in MATLAB®. `R`, on the other hand, only has two packages dedicated to the analysis of three- or multi-way data. The packages `PTAk` (Leibovici 2010) and `ThreeWay` (Giordani, Kiers, Del Ferraro, et al. 2014), offer a suit of functions for handling three-way arrays in `R`. These functions include Tucker3 and PARAFAC models and also some processing methods for three-way data such as centering, scaling, unfolding and normalizing arrays. However, N-PLS is notably missing from these packages, so no prediction method for three-way arrays was implemented in `R` before the release of the `sNPLS` package developed in this thesis. In this way, with the development of the package, it has not only been implemented a newly developed method such as sparse N-PLS, but also enabled the use of standard N-PLS regression models to `R` users.

Chapter 6

Variable selection techniques

6.1 VIP scores

As explained in the previous chapter, although N-PLS does not provide variable selection at the model-fitting step, some methods have been developed to implement variable selection on the fitted model. The variable importance in the projection (VIP) method is a ranking method for variable selection that was first developed for standard PLS regression models in two-way data matrices (Wold, Sjöström, and Eriksson 2001; Chong and Jun 2005). For this two-way case, the variable importance in the projection is a measure of the influence of each of the variables in the data matrix $\mathbf{X}$ on the response matrix $\mathbf{Y}$ (Equation 6.1)

$$VIP_j^2 = S_f w_{jf}^2 \cdot SSY_f \cdot J/(SSY_{tot.expl.} \cdot F) \tag{6.1}$$

where J is the number of variables in $\mathbf{X}$ and F is the number of latent variables in the model.

When the response is a vector $\mathbf{y}$ it holds that

$$SSY_f = b_{ff}^2 \mathbf{t}_f^T \mathbf{t}_f \tag{6.2}$$

and

$$SSY_{tot.expl.} = \mathbf{b}^2\mathbf{T}^T\mathbf{T} \tag{6.3}$$

where $\mathbf{T}$ is the score matrix and $\mathbf{b}$ are the PLS coefficients. Following this, VIP values for each variable can be computed using the PLS weight w_{jf} based on how much of $\mathbf{y}$ is explained in each latent variable. It is considered that VIP values greater than one have an above average influence and should be considered as relevant variables in explaining $\mathbf{Y}$, although the limit is arbitrary and can be modified depending on the requirements of the study (Akarachantachote, Chadcham, and Saithanu 2014).

In their work, Favilla, Durante, et al. (2013), expanded the applicability of the VIP scores to the three-way case as follows:

In the case of a two-dimensional $\mathbf{Y}$ $(I \times M)$, the relation presented on Equation 6.1 applies to each mode of a three-way $\underline{\mathbf{X}}$ array. For each latent variable f and for each y-variable y_m it holds that:

$$VIP_j^2 = \Sigma_f w_{jf}^2 \cdot SSY_{mf} \cdot J/(SSY_{tot.expl.,m} \cdot F) \tag{6.4}$$

and

$$VIP_k^2 = \Sigma_f w_{kf}^2 \cdot SSY_{mf} \cdot K/(SSY_{tot.expl.,m} \cdot F) \tag{6.5}$$

where

$$SSY_{tot.expl.,m} = \Sigma_i(\mathbf{T}_{(I\times F)}\mathbf{B}_{(F\times F)}\mathbf{q}^T_{(m,F)})^2 \tag{6.6}$$

and

$$SSY_{mf} = \Sigma_i(\mathbf{t}_f b_{ff} q_{mf})^2 \tag{6.7}$$

where I is the number of samples, J is the number of variables in the second mode of $\underline{\mathbf{X}}$ and K is the number of elements of the third mode in $\underline{\mathbf{X}}$. $\mathbf{T}$ is the first mode score matrix, $\mathbf{B}$ the inner relation coefficients matrix and $\mathbf{Q}$ the loadings matrix of $\mathbf{Y}$.

For any model dimension f and variable j the corresponding VIP value is estimated by the squared weight w_{jf}^2 multiplied by the percent of $\mathbf{Y}$ explained by that dimension. Then, importance values are normalized constraining the VIP^2 value to equal the number of variables.

Taking Equation 6.6 and Equation 6.7, when considering all $\mathbf{Y}$ variables together, it holds that

$$SSY_{tot.expl.} = \Sigma_i(\mathbf{T}_{(I\times F)}\mathbf{B}_{(F\times F)}\mathbf{Q}^T_{(M,F)})^2 \tag{6.8}$$

$$SSY_f = \Sigma_m\Sigma_i(\mathbf{t}_f b_{ff}\mathbf{q}^T_{mf})^2 \tag{6.9}$$

and

$$VIP_j^2 = \Sigma_f w_{jf}^2 \cdot SSY_f \cdot J/(SSY_{tot.expl.} \cdot F) \tag{6.10}$$

From this, extension to other $\underline{\mathbf{Y}}$ modes can be easily obtained.

An example of the VIP score computation results of a PLS analysis performed on an arbitrary data set is provided in Figure 6.1, where the first eleven variables and the 28th variable have a VIP score greater than one and are considered as relevant for predicting $\mathbf{y}$.

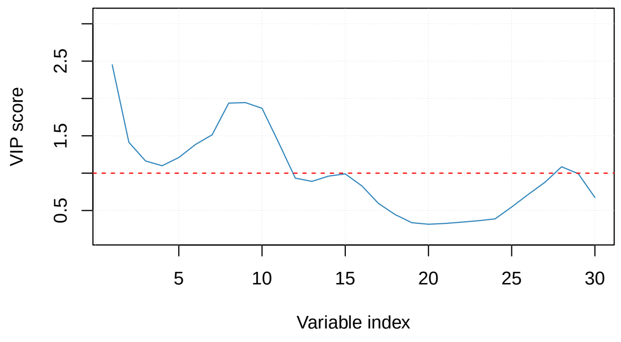

Figure 6.1: Plot of the VIP scores from the different variables in a PLS model. The dashed red line marks the VIP=1 threshold.

6.2 Selectivity Ratio

In the majority of cases, several components are needed to adequately fit a model for a specific response $\mathbf{y}$. By using the $\mathbf{y}$ vector as a target, it is possible to transform the PLS components to obtain a single predictive target-projected component, which is analogous to the predictive component in orthogonal partial least squares (Bylesjö et al. 2006). The selectivity ratio (SR) is estimated by calculating the ratio between explained and residual variance of the variables on that target-projected component (Rajalahti et al. 2009). Briefly, by selecting the normalized regression coefficients as PLS weights

$$\mathbf{w}_{TP} = \mathbf{b}_{PLS}/\|\mathbf{b}_{PLS}\| \tag{6.11}$$

and calculating target-projected scores by projection on the matrix of variables

$$\mathbf{t}_{TP} = \mathbf{X}\mathbf{w}_{TP} = \mathbf{X}\mathbf{b}_{PLS}/\|\mathbf{b}_{PLS}\| = \hat{\mathbf{y}}/\|\mathbf{b}_{PLS}\| \tag{6.12}$$

it is showed that the target-projected scores are proportional to the vector of predicted responses.

The target-projected loadings can be calculated as

$$\mathbf{p}_{TP}^{T} = \mathbf{t}_{TP}^{T}\mathbf{X}/(\mathbf{t}_{TP}^{T}\mathbf{t}_{T}P) = \mathbf{b}_{PLS}^{T}/\|\mathbf{b}_{PLS}\| * (\mathbf{X}^{T}\mathbf{X})/\|\mathbf{t}_{TP}\|^{2} \tag{6.13}$$

This shows that the target-projected loadings are the product of the normalized regression coefficients and the covariance matrix of the variables scaled by the inverse of the variance of the target-projected scores. So the target-projected loading vector combines the predictive ability of a variable with its $\mathbf{X}$ explanatory power.

The target projection model can then be expressed as

$$\mathbf{X} = \mathbf{X}_{TP} + \mathbf{E}_{TP} = \mathbf{t}_{TP}\mathbf{p}_{TP}^{T} + \mathbf{E}_{TP} \tag{6.14}$$

From here, explained and residual variance ($v_{expl,j}$ and $v_{res,j}$ can be computed for each variable j. Then the selectivity ratio is obtained by dividing explained and residual variance as follows

$$SR_j = v_{expl,j}/v_{res,j},\ j = 1,2,3\ldots \tag{6.15}$$

As with the case of VIP scores and any other ranking method, the limit or threshold of SR value for considering a variable as relevant is arbitrary. Nevertheless, the authors of the original paper considered a 75% of explained variance, corresponding to a SR of three, a reasonable threshold for selecting variables as relevant in predicting **y**.

An example of the SR computation results on the same PLS analysis presented in Figure 6.1 for VIP scores is provided in Figure 6.2. It can be seen that conclusions from this method regarding relevant variables are coincident with those obtained using the VIP scores methods, so there is good agreement between both methods. There is only a slight disagreement regarding the first three variables, which are selected by the VIP scores methods and discarded by the selectivity ratio when using the standard threshold values for both methods.

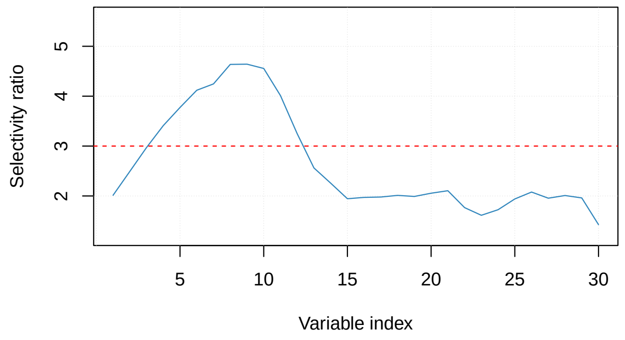

Figure 6.2: Plot of the SR from the different variables in a PLS model. The dashed red line marks the SR=3 threshold.

As of today. there are no published studies comparing the use of both methods in a comprehensive set of different data sets for the three-way case. However, a study of Farrés et al. (2015) compared both methods regarding variable selection performance and prediction capability in different two-way data sets. This study concluded that, for most data sets, the SR method selected fewer variables than the VIP method. In many cases, some of the variables selected by the VIP method were false positive candidates and some of the variables not selected by the SR were false negative candidates. Regarding prediction accuracy, each method was better that the other in a similar number of data sets, so final decision about the best of the two approaches should be based on the aims of the specific study being performed.

6.3 Variable selection through L1 penalization

As explained in chapter 3, lasso applies a bias to the model fitting step of a linear or generalized linear model by shrinking all the coefficients using L1-penalization. This shrinkage reduces the complexity of the model, reducing variance and forcing some of the coefficients to be exactly zero, effectively performing variable selection at the same time as it increases predictive capacity reducing prediction error. Unfortunately the application of lasso, or its generalization elastic net, to three-way or multi-way data is not straightforward. The only approximations of these techniques for dealing with three-way data have been based on unfolding of the $\underline{\mathbf{X}}$ array (Chiu and Yao 2013).

In chapter 5, the standard N-PLS algorithm has been presented as an appealing technique for modelling three- or multi-way data sets when there is a response to be predicted. However, in this chapter it has been exposed that, although there are some post modelling methods for estimating variable importance such as VIP (Favilla, Durante, et al. 2013) and selectivity ratio (Rajalahti et al. 2009), it does not perform in-model variable selection, which would greatly increase its utility for analyzing metabolomic data sets. A suitable approach, recently published in (Hervás, J. Prats-Montalbán, Lahoz, et al. 2018), would consist in applying the L1 penalization from lasso inside the N-PLS algorithm to be able to perform variable selection at the model-fitting step while maintaining the capability of N-PLS of dealing with multi-way arrays.

For this, it is proposed to use the soft-thresholding operator which can be derived from the lasso problem as follows:

1. Assuming $\mathbf{X}$ (matrizied version of $\underline{\mathbf{X}}$) is composed of orthogonal columns, the least squares solution is

$$\hat{\beta}^{LS} = (X^T X)^{-1} X^T y = X^T y \tag{6.16}$$

2. Using the Lagrangian form, an equivalent problem to that considered would be

$$\min_{\beta} \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1 \tag{6.17}$$

3. Expansion of the first term gives

$$\frac{1}{2}y^T y - y^T X\beta + \frac{1}{2}\beta^T \beta \tag{6.18}$$

Since $y^T y$ does not contain any of the variables of interest, it can be discarded, and one can consider the following equivalent problem

$$\min_{\beta}(-y^T X\beta + \frac{1}{2}||\beta||_2) + \lambda||\beta||_1 \tag{6.19}$$

Which can be rewritten as

$$\min_{\beta} \sum_{j=1}^{p} -\hat{\beta}_j^{LS}\beta_j + \frac{1}{2}\beta_j^2 + \lambda|\beta_j| \tag{6.20}$$

So, there is a sum of objectives as the objective function. Since each of them corresponds to a separate β_j, this means that each variable may be solved individually.

4. For a certain j, the aim is to minimize

$$\mathcal{L}_j = -\hat{\beta}_j^{LS}\beta_j + \frac{1}{2}\beta_j^2 + \lambda|\beta_j| \tag{6.21}$$

If $\hat{\beta}_j^{LS} > 0$, then $\beta_j \geq 0$, otherwise one could just change its sign and get a lower value for the objective function. Correspondingly, if $\hat{\beta}_j^{LS} < 0$, then $\beta_j \leq 0$

5. In the first case, if $\hat{\beta}_j^{LS} > 0$ and $\beta_j \geq 0$, then

$$\mathcal{L}_j = -\hat{\beta}_j^{LS}\beta_j + \frac{1}{2}\beta_j^2 + \lambda\beta_j \tag{6.22}$$

After differentiating respect to β_j amd setting equal to zero, the result is $\beta_j = \hat{\beta}_j^{LS} - \lambda$. Since $\beta_j \geq 0$, the right-hand side must be non-negative, so the solution would be

$$\beta_j^{lasso} = sgn(\beta_j^{LS})(|\beta_j^{LS}| - \lambda)^+ \tag{6.23}$$

Which is the soft-thresholding operator.

6. In the other case, if $\hat{\beta}_j^{LS} < 0$ and $\beta_j \leq 0$, then

$$\mathcal{L}_j = -\hat{\beta}_j^{LS}\beta_j + \frac{1}{2}\beta_j^2 - \lambda\beta_j \tag{6.24}$$

After differentiating respect to β_j and setting equal to zero the result is $\beta_j = \hat{\beta}_j^{LS} + \lambda$. Since it is needed that $\beta_j \leq 0$ the solution is

$$\beta_j^{lasso} = sgn(\beta_j^{LS})(|\beta_j^{LS}| - \lambda)^+ \tag{6.25}$$

Which, again, gives the soft-thresholding operator.

The soft-thresholding operator is well known in signal processing and image analysis, since it is used as a denoising filter in noisy images to obtain an approximation of the original image which gives the minimum mean square error (Khare and Tiwary 2005; Joy, Peter, and John 2013). To understand how it works, the soft-thresholding function for a threshold of 3 has been represented in Figure 6.3. Its use to create sparsity in projection methods was first introduced by Zou, Hastie, and R. Tibshirani (2006) in sparse Principal Component Analysis.

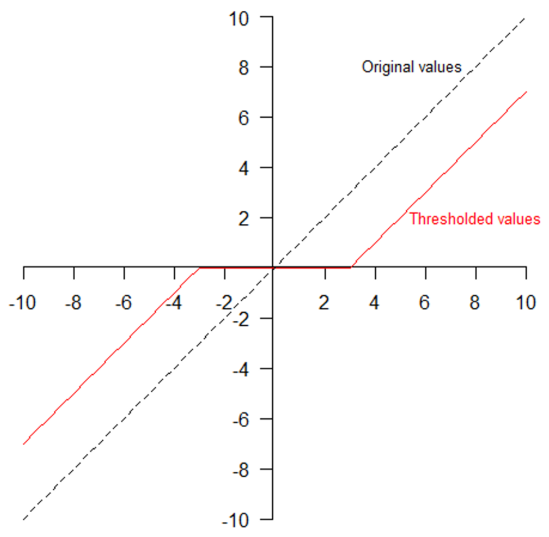

Figure 6.3: Soft-thresholding function in the range x = [-10, 10] with a threshold λ of 3. Values with $|x|$ higher than λ have the threshold value substracted and values with $|x|$ equal or less than λ are set to zero. Black dashed line represents the original values and red line represents the thresholded values.

Part III

Results

Chapter 7

Sparse N-PLS, a method for variable selection in multiway data sets

7.1 Sparse N-PLS, integration of L1 penalization in the N-PLS algorithm

As commented in the previous chapter, it is proposed to introduce the L1-penalization in the N-PLS algorithm. Since there are three modes, the aim is to perform selection not only on the second mode (variables), but also on the third mode. To this aim, a similar approach to that of Lê Cao et al. 2008 is used. Briefly, to achieve sparse versions of $\mathbf{w}^{\mathrm{J}}$ and $\mathbf{w}^{\mathrm{K}}$ for each latent variable, the soft-thresholding penalty function $\beta_j^{lasso} = sgn(\beta_j^{LS})(|\beta_j^{LS}| - \lambda)^{+}$ is introduced in the N-PLS algorithm right after the SVD at the $\mathbf{w}^{\mathrm{J}}$ and $\mathbf{w}^{\mathrm{K}}$ determination. The complete algorithm is as follows (Figure 7.1):

Center $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$, and unfold $\underline{\mathbf{X}}$ (and $\underline{\mathbf{Y}}$ when necessary) into a two-way matrix.

Let $\mathbf{u}$ be some column of $\mathbf{Y}$, and set f=1

```
1. $\mathbf{w}^{\mathrm{T}} = \mathbf{u}^{\mathrm{T}}\mathbf{X}/\mathbf{u}^{\mathrm{T}}\mathbf{u}$
2. Build $\mathbf{Z}$ by refolding $\mathbf{w}$ according to the modes dimensions
3. Determine $\mathbf{w}^{\mathrm{J}}$ and $\mathbf{w}^{\mathrm{K}}$ by SVD
4. L1-penalization inclusion
    (a) Apply soft-thresholding on $\mathbf{w}^{\mathrm{J}}$: $\hat{w}_i^{j*} = sgn(\hat{w}_i^j)(|\hat{w}_i^j| - \lambda_j)^+$
    (b) Apply soft-thresholding on $\mathbf{w}^{\mathrm{K}}$: $\hat{w}_i^{k*} = sgn(\hat{w}_i^k)(|\hat{w}_i^k| - \lambda_k)^+$
    (c) Input the new $\mathbf{w}$ as kronecker$(\mathbf{w}^{\mathrm{K}}, \mathbf{w}^{\mathrm{J}})$
5. $\mathbf{t} = \mathbf{X}\mathbf{w}/\mathbf{w}^{\mathrm{T}}\mathbf{w}$
6. $\mathbf{q} = \mathbf{Y}^{\mathrm{T}}\mathbf{t}/\mathrm{norm}(\mathbf{Y}^{\mathrm{T}}\mathbf{t})$
7. $\mathbf{u} = \mathbf{Y}\mathbf{q}$
8. Check for convergence. If it is achieved, continue; otherwise, go to 1
9. $\mathbf{b} = (\mathbf{T}^{\mathrm{T}}\mathbf{T})^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{u}$; where $\mathbf{T} = [\mathrm{t1\ t2} \ldots \mathrm{t}_f]$
10. $\mathbf{X} = \mathbf{X} - \mathbf{t}\mathbf{w}^{\mathrm{T}}$ and $\mathbf{Y} = \mathbf{Y} - \mathbf{t}\mathbf{b}\mathbf{q}^{\mathrm{T}}$
11. $f = f+1$. Continue from step 1 until a good description of $\mathbf{Y}$
```

This algorithm is applicable to both the standard regression (continuous response) and the discriminant version of the *N*-PLS model, i.e. *N*-PLS-DA. In the case of *N*-PLS-DA, $\underline{\mathbf{Y}}$ is a $\mathbf{y}$ vector formed by ones and zeros, each of the two values related to one of the two classes to be segregated.

7.2 Hyperparameters of the sparse *N*-PLS algorithm

Hyperparameters are tuneable parameters of the model, whose values are specified before starting the algorithm instead of determined inside the algorithm as standard parameters. L1-penalization has one hyperparameter, called the penalization factor (the amount of L1-penalization to apply). However, since the L1-penalization is applied to both $\mathbf{w}^{\mathrm{J}}$ and $\mathbf{w}^{\mathrm{K}}$ the number of hyperparam-

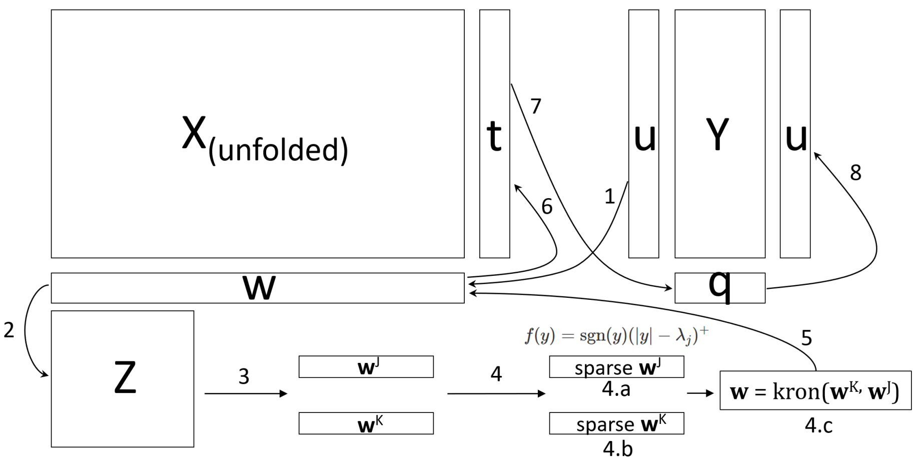

Figure 7.1: Scheme detailing the different steps of the sNPLS algorithm as explained in the previous page.

eters for a sparse N-PLS model (sNPLS) is three: the number of components (shared with the standard N-PLS model), the number of variables to select at each component, and the number of elements of the third mode to select at each component. The number of components can range between one and J, and the number of variables per component or elements of the third mode per component can range between zero and J and zero and K, respectively. It is necessary that at least one component includes one selected variable and one element of the third mode to fit a valid model. A sparse N-PLS model where all variables and all elements of the third mode are forced to be selected reduces to a general N-PLS model, where only the number of components acts as hyperparameter.

7.2.1 Tuning of the parameters

Optimization of the hyperparameter values is necessary to achieve optimal prediction power or optimal variable selection performance in the case of sNPLS. One important point to take into account is that consistency (selecting the right variables) and minimizing prediction error appear to be non-compatible (Yang 2005), so the tuning criterion has to be adapted depending on the main objective of the analysis. For the objective of variable selection, Zou and Hastie (2005) mention the option of just choosing the desired number of non-zero coefficients as a viable alternative for interpretation purposes. In the present approach, the focus is set on minimizing prediction error, so to perform this tuning of the parameters, a grid search of the hyperparameter space (Lameski et al. 2015) is proposed guided by the mean squared prediction error evaluated by cross-validation (Duarte and Wainer 2017). More specifically, the approach consists on performing K-fold cross-validation repeatedly and averaging their results, to alleviate instability in the selection of parameters because of high variance in the single cross-validation results (Krstajic et al. 2014). This high variance entails that different runs of the cross-validation procedure can yield different results regarding the estimated best set of the parameters and also that the estimation of the best set of parameters is prone to overfitting (bias-variance tradeoff).

Briefly, K-fold cross-validation consists on randomly splitting the data set in a number of folds, K, and perform K iterations of model fitting and testing, where a different fold is used as test set in each iteration while the other folds are used to train the model. This way, an out-of-sample estimate for the prediction error of the model is estimated at each iteration. Later, all estimates are averaged to get the mean cross-validated error.

The procedure is performed as follows:

1. Divide the data set in K subsets (ideally of equal size)
2. For each k in 1, . . . K:
   - Train the model on (x_i, y_i) where $i \notin F_k$
   - Estimate prediction error on (x_k, y_k) as E_k
3. Average all errors to get the CVE estimate

A scheme of the procedure for K-fold cross-validation is represented in Figure 7.2.

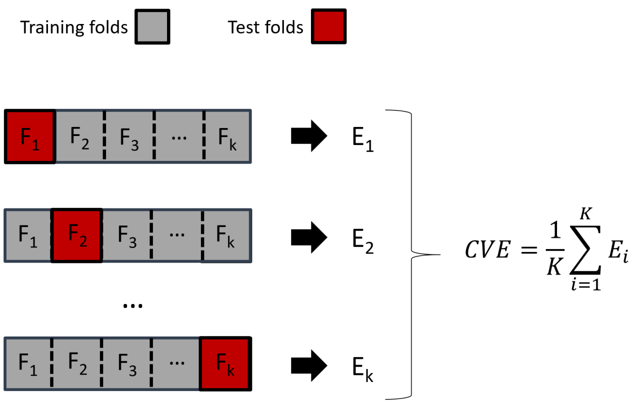

Figure 7.2: Scheme of a K-fold cross-validation procedure. At each iteration, one different fold acts as test fold while all the others act as training folds. At the end, the different prediction errors are averaged.

There are alternative ways to select or tune the hyperparameters apart from cross-validation (Vujačić, Abbruzzo, and Wit 2015; Zhang, Li, and C.-L. Tsai 2010). These alternatives have the advantage of being computationally faster than cross-validation, but one important advantage of cross-validation is the possibility to estimate the standard error of the estimated cross-validated error performing the following steps:

1.
$$E_k = (y - \hat{y})^2 \tag{7.1}$$

2.
$$CVE = \frac{1}{K}\sum_{i=1}^{K} E_i \tag{7.2}$$

3.
$$SD_{CVE} = \sqrt{var(E_1, E_2, \ldots, E_k)} \tag{7.3}$$

4.
$$SE_{CVE} = \frac{SD_{CVE}}{\sqrt{K}} \tag{7.4}$$

Performing a grid search on a three-dimensional hyperparameter step is computationally very demanding, so the fact that grid search is an embarrassingly parallel task (McGibbon et al. 2016) is exploited to parallelize the cross-validation procedure and be able to perform the repeated cross-validation in a reasonable time for medium and large data sets. Details on the parallelization algorithm will be presented in next chapter.

Chapter 8

sNPLS package, a comprehensive software for $\boldsymbol{N}$-PLS and sparse $\boldsymbol{N}$-PLS analysis

8.1 Implementation of the sNPLS algorithm in `R`

In this section the key functions of the code for the *sNPLS* package implementing the sNPLS method will be exposed and commented. These include the main sNPLS function, the R-matrix function and the cross-validation functions. The full code for the package will be included in the appendix.

8.1.1 Main algorithm

This is the code of the main function of the package. It performs N-PLS regression as explained in section 5.3 and includes an L1-penalization step at the determination of $\mathbf{w}^J$ and $\mathbf{w}^K$ as explained in section 7.1.

```r
sNPLS <- function(XN, Y, ncomp = 2, conver = 1e-16, max.iteration = 10000,
                  keepJ = rep(ncol(XN), ncomp),
                  keepK = rep(rev(dim(XN))[1], ncomp),
                  scale.X=TRUE, center.X=TRUE, scale.Y=TRUE, center.Y=TRUE,
                  silent = F) {

  mynorm <- function(x) sqrt(sum(diag(crossprod(x))))
  if (length(dim(Y)) == 3) Y <- unfold3w(Y)
  if (length(dim(XN)) != 3)
    stop("'XN' is not a three-way array")
  if (!is.null(rownames(XN)))
    y.names <- x.names <- rownames(XN) else {
    y.names <- x.names <- 1:dim(XN)[1]
    }
    if (!is.null(colnames(XN)))
      var.names <- colnames(XN) else {
      var.names <- paste("X.", 1:dim(XN)[2], sep = "")
      }
      if (!is.null(dimnames(XN)[[3]]))
        x3d.names <- dimnames(XN)[[3]] else {
        x3d.names <- paste("Z.", 1:dim(XN)[3], sep = "")
        }
        if (!is.null(colnames(Y)))
          yvar.names <- colnames(Y) else {
          yvar.names <- paste("Y.", 1:dim(Y)[2], sep = "")
          }
          if(!center.X) center.X <- rep(0, ncol(XN)*dim(XN)[3])
          if(!center.Y) center.Y <- rep(0, ncol(Y))
          if(!scale.X) scale.X <- rep(1, ncol(XN)*dim(XN)[3])
          if(!scale.Y) scale.Y <- rep(1, ncol(Y))

          # Matrices initialization
          Tm <- U <- Q <- WsupraJ <- WsupraK <- X <- P <- NULL
          Yorig <- Y
          Y <- scale(Y, center = center.Y, scale = scale.Y)
          y_center <- attr(Y, "scaled:center")
          y_scale <- attr(Y, "scaled:scale")
          B <- matrix(0, ncol = ncomp, nrow = ncomp)
          Gu <- vector("list", ncomp)
          S <- svd(Y)$d
          u <- Y[, S == max(S)]  #Column with the highest variance
          # Unfolding of XN en 2-D
          X <- unfold3w(XN)
          #Check for zero variance columns and fix them with some noise
          if(any(apply(X, 2, sd)==0)){
            X[,apply(X, 2, sd)==0] <- apply(X[,apply(X, 2, sd)==0, drop=F],
                                            2, function(x) jitter(x))
          }
          # Center and scale
          Xd <- scale(X, center = center.X, scale = scale.X)
          x_center <- attr(Xd, "scaled:center")
          x_scale <- attr(Xd, "scaled:scale")

          # Main loop for each component
          for (f in 1:ncomp) {
            nj <- ncol(XN) - keepJ[f]
            nk <- dim(XN)[3] - keepK[f]
            it = 1
            while (it < max.iteration) {
              Zrow <- crossprod(u, Xd)
              Z <- matrix(Zrow, nrow = dim(XN)[2], ncol = dim(XN)[3])
              svd.z <- svd(Z)
              wsupraj <- svd.z$u[, 1]
              # L1 penalization for wsupraj
              if (nj != 0) {
                wsupraj <- ifelse(abs(wsupraj) >
                                  abs(wsupraj[order(abs(wsupraj))][nj]),
                                  (abs(wsupraj) -
                                  abs(wsupraj[order(abs(wsupraj))][nj])) *
                                  sign(wsupraj), 0)
              }
              ##########
              wsuprak <- svd.z$v[, 1]
              # L1 penalization for wsuprak
              if (nk != 0) {
                wsuprak <- ifelse(abs(wsuprak) >
                                  abs(wsuprak[order(abs(wsuprak))][nk]),
                                  (abs(wsuprak) -
                                  abs(wsuprak[order(abs(wsuprak))][nk])) *
                                  sign(wsuprak), 0)
              }
              ##########
              tf <- Xd %*% kronecker(wsuprak, wsupraj)
              qf <- crossprod(Y, tf)/mynorm(crossprod(Y, tf))
              uf <- Y %*% qf
              if (sum((uf - u)^2) < conver) {
                if (!silent) {
                  cat(paste("Component number ", f, "\n"))
                  cat(paste("Number of iterations: ", it, "\n"))
                }
                it <- max.iteration
                Tm <- cbind(Tm, tf)
                WsupraJ <- cbind(WsupraJ, wsupraj)
                WsupraK <- cbind(WsupraK, wsuprak)
                bf <- MASS::ginv(crossprod(Tm)) %*% t(Tm) %*% uf
                B[1:length(bf), f] <- bf
                Q <- cbind(Q, qf)
                U <- cbind(U, uf)
                TM <- MASS::ginv(crossprod(Tm)) %*% t(Tm)
                WkM <- MASS::ginv(crossprod(WsupraK)) %*% t(WsupraK)
                WjM <- MASS::ginv(crossprod(WsupraJ)) %*% t(WsupraJ)
                Gu[[f]] <- TM %*% X %*% kronecker(t(WkM), t(WjM))
                P[[f]] = t(as.matrix(Gu[[f]]) %*%
                           t(kronecker(WsupraK, WsupraJ)))
                Y <- Y - Tm %*% bf %*% t(qf)
                S <- svd(Y)$d
                u <- Y[, S == max(S)]
              } else {
                u <- uf
                it <- it + 1
              }
            }
          }
          Yadjsc <- Tm %*% B %*% t(Q)
          Yadj <- Yadjsc * y_scale + y_center
          SqrdE <- sum((Yorig - Yadj)^2)
          rownames(WsupraJ) <- var.names
          rownames(WsupraK) <- x3d.names
```

```
119           rownames(Q) <- yvar.names
120           rownames(Tm) <- rownames(U) <- x.names
121           colnames(Tm) <- colnames(WsupraJ) <- colnames(WsupraK) <-
122                           colnames(B) <- colnames(U) <- colnames(Q) <-
123                           names(Gu) <- names(P) <- paste("Comp.", 1:ncomp)
124           output <- list(T = Tm, Wj = WsupraJ, Wk = WsupraK, B = B, U = U,
125                          Q = Q, P = P, Gu = Gu, ncomp = ncomp, Yadj = Yadj,
126                          SqrdE = SqrdE,
127                          Standarization = list(ScaleX = x_scale,
128                                                CenterX = x_center,
129                                                ScaleY = y_scale,
130                                                CenterY = y_center))
131           class(output)<-"sNPLS"
132           return(output)
133 }
```

Listing 8.1: sNPLS main function

Lines 1-5 define the parameters of the function. Parameters with default values have that value assigned after an equal sign. Line 7 defines an internal function for normalization. Lines 8-30 check for inconsistencies in the data structure, unfold $\underline{\mathbf{Y}}$ (when necessary) and assign variable names. After this, all necessary matrices are initiallized in lines 33-52, $\underline{\mathbf{X}}$ is unfolded into $\mathbf{X}$ and optional different centering and scaling methods are applied on $\mathbf{Y}$ and $\mathbf{X}$ (e.g., autoscaling, blockscaling, scaling within any mode, etc.). The following lines (55-113) correspond to the N-PLS algorithm described in section 7.1. Last lines (114-133) include the estimation of $\mathbf{Y}$, the computing of the squared prediction error, and the generation of the output object of the function, defined as an S3 object of class *sNPLS*. This object includes the different output matrices of the model as well as all the information regarding the model fit such as centering on $\mathbf{X}$ and/or $\mathbf{Y}$ and used values for the hyperparameters.

8.1.2 R matrix computation

This function is used for computing the coefficients used for prediction of $\mathbf{Y}$ in the `predic.sNPLS` function.

```
Rmatrix<-function(x) {
  WsupraK <- x$Wk
  WsupraJ <- x$Wj
  R <- matrix(nrow = dim(x$Wj)[1] * dim(x$Wk)[1], ncol = x$ncomp)
  ncomp <- x$ncomp
  kroneckers<-sapply(1:x$ncomp, function(x) {
                     kronecker(WsupraK[, x], WsupraJ[, x]))
                     }
  tkroneckers<-apply(kroneckers, 2, function(x) t(x))
  R[,1] <- kroneckers[,1]
  if(ncomp>1){
    for(i in 2:ncomp){
      pi <- pi0 <- Matrix::Matrix(diag(dim(R)[1]), sparse=TRUE)
      for (j in 1:(i - 1)) {
```

```
15          pi <- Matrix::Matrix(pi %*% pi0 - kroneckers[,j] %*%
16          t(tkroneckers[,j]), sparse=TRUE)
17        }
18        w <- kroneckers[, i]
19        pi <- pi %*% w
20        R[, i] <- Matrix::as.matrix(pi)
21      }
22    }
23    return(R)
24  }
```

Listing 8.2: R matrix function

Of note, most of the computations performed inside the `Rmatrix` function make use of sparse matrices (lines 12-19), taking advantage of the sparsity created by the `sNPLS` function when adjusting the model to greatly reduce computing times. The `Rmatrix` function is an internal function (not exported to the end user) called by the `sNPLS` and `predict.sNPLS` functions to get the regression coefficients in order to be able to perform predictions from new $\underline{\mathbf{X}}$ as explained in Leardi (2005):

The **R** matrix is defined as:

$$\mathbf{R} = [\mathbf{w}_1(\mathbf{I} - \mathbf{w}_1\mathbf{w}_1^T)\mathbf{w}_2 \ldots (\prod_{f=1}^{F-1}(\mathbf{I} - \mathbf{w}_f\mathbf{w}_f^T)\mathbf{w}_F)] \tag{8.1}$$

From this, it derives that

$$\mathbf{T} = \mathbf{XR} \tag{8.2}$$

So, from the equation

$$\hat{\mathbf{y}} = \mathbf{Tb} \tag{8.3}$$

it is obtained

$$\mathbf{b}_{NPLS} = \mathbf{Rb} \tag{8.4}$$

8.1.3 Cross validation (cv_snpls)

The `cv_snpls` function is used to estimate the best combination of parameters for adjusting sNPLS models using K-fold cross-validation. Since the hyperparameter space can be huge when considering a grid search for three different hyperparameters, it has been parallelized using the *parallel* package to greatly speed up computation times. For simple problems or small data sets it can also be run in single thread mode.

```r
cv_snpls <- function(X_npls, Y_npls, ncomp = 1:3, keepJ = 1:ncol(X_npls),
                     keepK = 1:dim(X_npls)[3], nfold = 10, parallel = TRUE,
                     free_cores = 2, ...) {
  #Creation of cluster and exportation of data to the nodes
  if (parallel & (parallel::detectCores()>1)) {
      cl <- parallel::makeCluster(max(2,
                              parallel::detectCores() - free_cores))
      parallel::clusterExport(cl, list(deparse(substitute(X_npls)),
                                  deparse(substitute(Y_npls))))
      parallel::clusterCall(cl, function() require(sNPLS))
  }
 if(length(dim(Y_npls)) == 3) Y_npls <- unfold3w(Y_npls)
 top <- ceiling(dim(X_npls)[1]/nfold)
  foldid <- sample(rep(1:nfold, top), dim(X_npls)[1], replace = F)
  #Expansion of hyperparameters grid
  search.grid <- expand.grid(list(ncomp = ncomp,
                              keepJ = keepJ,
                              keepK = keepK))
  SqrdE <- numeric()
  #Main function
  applied_fun <- function(y) {
      sapply(1:nfold, function(x) {
          tryCatch(cv_fit(xtrain = X_npls[x != foldid, , ],
                        ytrain = Y_npls[x != foldid, , drop = FALSE],
                        xval = X_npls[x == foldid, , ],
                        yval = Y_npls[x == foldid, , drop = FALSE],
                        ncomp = y["ncomp"],
                        keepJ = rep(y["keepJ"], y["ncomp"]),
                        keepK = rep(y["keepK"], y["ncomp"]), ...),
                    error=function(x) NA)
        })
  }
  #Parallelization of main function
  if (parallel) {
      cv_res <- parallel::parApply(cl, search.grid, 1, applied_fun)
      parallel::stopCluster(cl)
  } else cv_res <- pbapply::pbapply(search.grid, 1, applied_fun)
  cv_mean <- apply(cv_res, 2, function(x) mean(x, na.rm = TRUE))
  cv_se <- apply(cv_res, 2, function(x) sd(x, na.rm=TRUE)/sqrt(nfold))
  best_model <- search.grid[which.min(cv_mean), ]
  output <- list(best_parameters = best_model, cv_mean = cv_mean,
              cv_se = cv_se, cv_grid = search.grid)
  class(output)<-"cvsNPLS"
  return(output)
}
```

Listing 8.3: Cross validation function

Of note, regarding the `cv_snpls` function, is the code related to the parallelization of the procedure using the `parallel` package (lines 5-11 for the definition of the cluster and lines 34-37 for the execution of the parallelized procedure). The function makes use of the `parApply` function, which splits the search grid in n nodes and runs the cross-validation procedure on each split independently, combining all results at the end of the computations. Lines 21-32 of the function correspond to the cross-validation procedure defined in section 7.2.

8.1.4 Repeated Cross validation (repeat_cv)

The `repeat_cv` calls the `cv_snpls` function repeated times, but it changes the way of parallelizing the computations.

In this function, each `cv_snpls` call is run in single thread mode, but assigned to a different node. So, instead of on subsets of the search grid, the parallelization is performed on repetitions of the procedure on the whole grid. This handling of the parallelization allows for a more efficient distribution of the jobs across the different nodes of the defined cluster, assuming the number of repetitions is greater than the number of folds K of the K-fold cross-validation procedure.

```r
repeat_cv<-function(X_npls, Y_npls, ncomp = 1:3, keepJ = 1:ncol(X_npls),
                    keepK = 1:dim(X_npls)[3], nfold = 10,
                    parallel = TRUE, free_cores = 2, times=30, ...){
  #Definition of the cluster and exportation of data to the nodes
  if(parallel & (parallel::detectCores()>1)){
    cl <- parallel::makeCluster(max(2,
                                    parallel::detectCores() - free_cores))
    parallel::clusterExport(cl, list(deparse(substitute(X_npls)),
                                     deparse(substitute(Y_npls))))
    parallel::clusterCall(cl, function() require(sNPLS))
    #Parallelization of the repetitions of K-fold cross-validation
    rep_cv<-parallel::parSapply(cl, 1:times, function(x){
                                  cv_snpls(X_npls, Y_npls, ncomp=ncomp,
                                  keepJ = keepJ, keepK = keepK,
                                  parallel = FALSE, nfold = nfold, ...))
                                  }
    parallel::stopCluster(cl)
  } else {
    #Single threaded version (Not use, slow)
    rep_cv<-pbapply::pbreplicate(times, cv_snpls(X_npls, Y_npls,
                                 ncomp=ncomp, keepJ = keepJ,
                                 keepK = keepK, parallel = FALSE,
                                 nfold = nfold, ...))
  }
  resdata<-data.frame(ncomp=sapply(rep_cv[1,], function(x) x[[1]]),
                      keepJ=sapply(rep_cv[1,], function(x) x[[2]]),
                      keepK=sapply(rep_cv[1,], function(x) x[[3]]))
  class(resdata)<-c("repeatcv", "data.frame")
  return(resdata)
}
```

Listing 8.4: Repeated cross validation function

8.2 The sNPLS package

In the following section, the use of the different functions of the `sNPLS` package will be presented, as well as examples of their application to the different parts of the analysis in two data sets. A special emphasis will be given to the main `sNPLS` function, as well as to both cross-validation functions for the selection of the hyperparameters and to the different plot functions for the interpretation of results. This section is based on the published paper for the *sNPLS* package (Hervás, J. Prats-Montalbán, Lahoz, et al. 2018).

8.2.1 Functions

sNPLS function

Function `sNPLS` is used to fit N-PLS and sNPLS models to three-way data, depending on the input settings of the algorithm. The following R code shows an example of a model fit to a simulated three-way dataset with fifty observations (I), 50 variables (J) and three elements in the third mode (K).

```
R> library("sNPLS")
R> X_npls <- array(rpois(7500, 10), dim=c(50, 50, 3))
R> Y_npls <- matrix(2+0.4*X_npls[,5,1]+0.7*X_npls[,10,1]-
+     0.9*X_npls[,15,1] + 0.6*X_npls[,20,1] - 0.5*X_npls[,25,1]+
+     rnorm(50), ncol=1)
R> fit <- sNPLS(X_npls, Y_npls, ncomp=3, keepJ = rep(2,3),
+     keepK = rep(1,3))
```

Note that the function `sNPLS` needs a N-way array for $\underline{\mathbf{X}}$ and an N-way array, a two-dimensional matrix or a numeric vector for $\mathbf{Y}$ as inputs. If data is in another format the function will stop and throw an error. In the following paragraph, a generic call with a short description of each of the arguments of the function is presented.

```
sNPLS(XN, Y, ncomp = 2, conver = 1e-16,
+     max.iteration = 10000, keepJ = rep(ncol(XN), ncomp),
+      keepK = rep(rev(dim(XN))[1], ncomp), scale.X=TRUE,
+      center.X=TRUE, scale.Y=TRUE, center.Y=TRUE, silent = F)
```

XN	N-dimensional array containing the predictors
Y	Array containing the response(s)
ncomp	Number of components to use in the projection
conver	Convergence criterion
max.iteration	Maximum allowed number of iterations to achieve convergence
keepJ	Number of variables to keep at each component. If all variables are kept, NPLS regression is performed, if any variable is removed then sNPLS is performed.
keepK	Number of elements of the third mode to keep at each component.
scale.X	Perform scaling on **X**?
center.X	Perform centering on **X**?
scale.Y	Perform scaling on **Y**?
center.Y	Perform centering on **Y**?
silent	Allows to choose if information regarding number of iterations should be displayed

The function `sNPLS` produces an `S3 sNPLS` object with defined `coef`, `predict` and `plot` methods that will be discussed later. The object consists of a list containing the following components: [1-6] The **T**, $\mathbf{W}^J$, $\mathbf{W}^K$, **B** (regression coefficients between $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$), **Y** and **Q** matrices, [7-8] **P** and **G**u (the unfolded **G** core array of the Tucker decomposition), [9] The number of components, [10] Fitted values, [11] Squared error, [12] Scale and centering information performed on $\underline{\mathbf{X}}$ and **Y**.

`cv_sNPLS` and `repeat_cv` functions

Selecting parameter values for the `sNPLS` function requires choosing values for the number of components, the number of variables to select and the number of elements of the third mode to select. An appropriate way of selecting these parameters is performing a grid search with cross-validation. The function `cv_snpls` performs cross-validation on a grid of different `ncomp`, `keepJ` and `keepK` values estimating $RMSE$ (Root Mean Square Error) for each combination of values and selecting the best set producing the lowest $RMSE$.

```
R> X_npls<-array(rpois(7500, 10), dim=c(50, 50, 3))
R> Y_npls<-matrix(2+0.4*X_npls[,5,1]+0.7*X_npls[,10,1]-
+     0.9*X_npls[,15,1]+0.6*X_npls[,20,1]- 0.5*X_npls[,25,1]+
+     rnorm(50), ncol=1)
R> cv1<- cv_snpls(X_npls, Y_npls, ncomp=1:2, keepJ = 1:10,
+     keepK = 1:3, parallel = FALSE)
```

The generic call for `cv_snpls` is the following:

```
cv_snpls(X_npls, Y_npls, ncomp = 1:3, keepJ = 1:ncol(X_npls),
+    keepK = 1:dim(X_npls)[3], nfold = 10, parallel = FALSE)
```

It contains the same parameters as `sNPLS` but now admits vectors of values for each parameter in `ncomp`, `keepJ` and `keepK`. Since grid search can be computationally intensive, it makes use of the `parallel` package. Being able to perform computations in parallel greatly reduces running times, being able to finish between 1.5 and 6 times faster in an eight-core computer, depending on the data set. Parallel mode is activated by setting `parallel` argument to `TRUE` and selecting a suitable number of free cores. As explained in subsection 8.1.4, in the case of `cv_snpls`, the parallelization applies to the grid search, not to the different folds of the cross-validation.

To further reduce computation times and improve memory use, sparse matrices from the `R` package `Matrix` (Bates and Maechler 2018) are used whenever possible in the matrix multiplication steps of the function. Sparse matrices achieve these goals by using an alternative representation to that of dense matrices: instead of being stored as two-dimensional arrays, only their non-zero values are stored, along with an index linking these values with their location in the matrix. The function `cv_snpls` returns a `cvsnpls` object which is a list with a component containing the best combination parameters and other components containing information about the grid and its corresponding `RMSE`.

`cv_snpls` objects can be plotted for better interpretation of the results. The resulting plot is a grid of scatterplots with the different combinations of `keepJ`, `keepK` and `ncomp` and their resulting cross-validation errors (Figure 8.1).

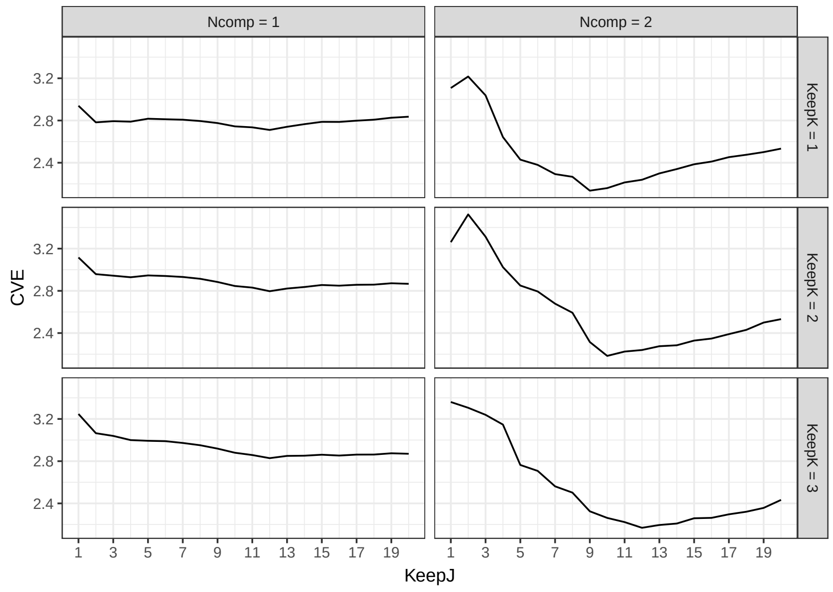

Figure 8.1: Results of cross-validation. Lines depicting the cross-validated error (CVE) for each combination of the parameters are presented in a grid layout combining `KeepJ` values (number of variables selected), `KeepK` values (number of elements of the third mode) and `Ncomp` values (number of components).

A known issue of cross-validation is the variance in its results (Krstajic et al. 2014), which entails that different runs of the cross-validation procedure can yield different results regarding the estimated best set of parameters and, more importantly, that the cross-validation procedure can overfit in the estimation of the error. This overfitting would entail that repeating the same cross-validation procedure on another sample of the population would yield totally different results. A reasonable solution is performing repeated cross-validation and selecting the most frequently selected set of parameters along a round of different runs. The function `repeat_cv` performs repeated cross-validation by calling the function `cv_snpls` repeated times and storing each result in a `data.frame` object. The syntax is the same as in `cv_snpls` with an extra argument `times` for specifying the number of repetitions to perform.

In the case of `repeat_cv` the parallelization applies to the replicates instead of applying to the grid search. The following code explains how to perform repeated cross-validation with `repeat_cv`.

```
R> repcv <- repeat_cv(X_npls, Y_npls, ncomp=1:2, keepJ = 1:10,
+     keepK = 1:3, parallel = FALSE, times=10)
```

The result of the `repeat_cv` call is a `data.frame` storing the results (best combination of `ncomp`, `keepJ` and `keepK` of each cross-validation repetition (Listing 8.5).

```
R> repcv
```

```
   ncomp keepJ keepK
1      2     9     1
2      2     9     1
3      2    10     1
4      2     9     1
5      2    11     2
6      2    12     1
7      2     7     1
8      2    10     1
9      2     8     1
10     2    12     2
```

Listing 8.5: Results of `repeat_cv` function.

This output can be presented as a cross-tabulation table with the absolute frequencies of each combination of the different parameters (Listing 8.6. In this example, the table shows that the combination `ncomp`=2, `keepJ`=9 and `keepK`=1 is the most recurrent (3 out of 10 repetitions), followed by the combination `ncomp`=2, `keepJ`=10 and `keepK`=1 with two repetitions and five other different combinations with just one occurrence each. It can also be noticed that none of the best combinations included only one component.

```
R> ftable(table(repcv))
```

```
            keepK 1 2
ncomp keepJ
2     7           1 0
      8           1 0
```

```
    9              3  0
    10             2  0
    11             0  1
    12             1  1
```

Listing 8.6: Cross-tabulation table of the results of `repeat_cv`.

Results of the `repeat_cv` function can also be plotted to obtain a kernel density plot, which can be one-, two- or three-dimensional depending of the number of constant parameters obtained in the repeated cross-validation procedure. This density plot depicts the most frequently selected parameters (Figure 8.2).

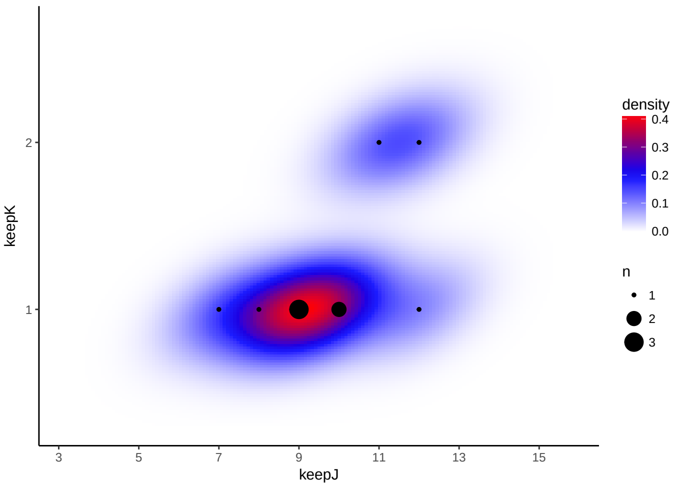

Figure 8.2: Kernel density plot with the result of the repeated cross-validation. Highest density lies around `keepJ`=9 and `keepK`=1. Points scaled in size by frequency of appearance are additionally included on top of the density plot. The number of components is not represented since it was constant at 2 in all repetitions of the cross-validation.

In this example, according to the plot, the most likely combination after all repetitions of the cross-validation was between 9 and 10 in the case of `keepJ` and `keepK`=1. Since the optimal number of components did not vary along all the repetitions, this parameter is not represented in the plot. Instead, the message `'ncomp is(are) constant with a value of 2'` is returned by the function. The density plot leads to a similar interpretation of the results as the cross-tabulation table, but the smoothing can result in more sensible estimates in the case of multiple and/or wide spread modes. It also provides a visual measure of the uncertainty in the selection of the best combination of parameters.

Full example functionality

To exhibit all package functionality, next it is provided a complete analysis of the `bread` dataset (Bro 1998) included in the *sNPLS* package. This data set consists on data of five different breads that were baked in duplicate giving a total of ten samples. Eight different judges assessed the breads with respect to eleven different sensorial attributes. The data can be regarded as a three-way array (10 x 11 x 8). The data are quite noisy as opposed to, e.g., spectral data. The salt content of each bread was also measured and it is considered as the response variable **y**.

```
R> data(bread)
R> Xbread <- bread$Xbread
R> Ybread <- bread$Ybread
R> cv_bread <- repeat_cv(Xbread, Ybread, ncomp=1:3, keepJ = 1:11,
+    keepK = 1:8, parallel = FALSE, times=50, nfold=3)
```

After importation of the data set and preparation of the $\underline{\mathbf{X}}$ array and **y** vector (first three lines of the code above), repeated cross-validation is performed on the data to select the optimal combination of hyperparameters. 50 repetitions of 3-fold cross-validation should be enough to get stable estimates, but using such a large number of repetitions would increase computing times dramatically. Therefore, using the option `parallel=TRUE` when performing repeated cross-validation is recommended. Results of this procedure yield a cross-tabulation table with the appearance frequencies for each combination of parameters Listing 8.7.

```
R> ftable(table(cv_bread))
```

```
              keepK 1 2 3 4 5 6 7 8
ncomp keepJ
1     1             1 2 2 4 3 3 1 1
      2             0 0 1 1 2 1 0 2
2     1             0 0 0 0 0 0 0 1
      2             0 0 0 0 0 0 0 2
      3             0 0 0 0 0 1 0 4
      4             0 0 0 0 0 0 1 1
      5             1 0 0 0 0 0 0 1
```

```
        6          0  0  0  0  0  0  1  0
        7          0  0  0  0  0  0  1  0
        8          0  0  0  0  0  0  0  2
        9          0  0  0  0  0  0  0  1
3       1          0  0  1  0  0  2  0  1
        2          0  0  0  0  0  0  1  1
        3          0  0  0  0  0  0  0  1
        5          0  0  0  0  0  0  0  1
```

Listing 8.7: `repeat_cv` results for the `bread` data set.

This table shows that there are two possible combinations of parameters that are almost equally selected by cross-validation. One is `ncomp`=1, `keepJ`=1 and `keepK`=4 and the other is `ncomp`=2, `keepJ`=3 and `keepK`=8, both with 4 appearances out of 50 repetitions. Also most of the other combinations are close to one of these two mentioned 'hotspots'. A plot of the `cv_bread` object (Figure 8.3) reveals a similar information with a representation of a sliced three-dimensional kernel density estimate.

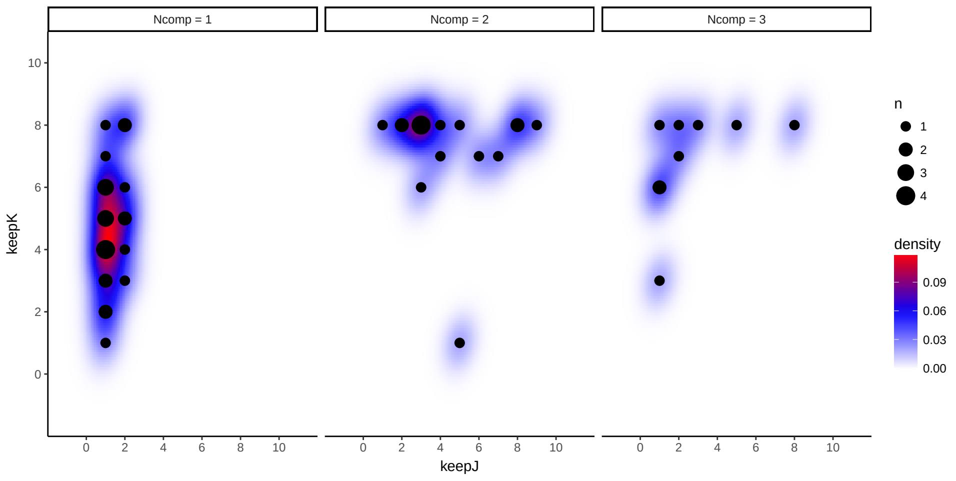

Figure 8.3: Results of the repeated cross-validation function performed on the `bread` dataset. The plot consists on the faceted representation of a three-dimensional density plot sliced in different planes (one per number of components). The kernel density estimation is created with the observed frequencies of the combinations of the different parameters. Additionally, points scaled in size by frequency of appearance are added to the density plots.

Next step would be fitting the `sNPLS` model using the `sNPLS` function with the selected set of parameters. As discussed before, the results point to two possible combinations and, based on the density plot, the option with `ncomp`=1, `keepJ`=1 and `keepK`=4 seems more likely. Nevertheless, the combination with `ncomp`=2 will be used to get working examples of the `plot` function (which needs, at least, two dimensions). It is important to take into account that `keepJ` and `keepK` have to be specified for each component, so they must be a vector of length equal to the number of components. Note that, in this case, the same number of attributes and judges for each component have been chosen, although this is not necessarily the case.

```
R> fit <- sNPLS(Xbread, Ybread, ncomp = 2, keepJ = rep(3, 2),
+    keepK = rep(8, 2), silent = F)
```

The summary of the fit shows the number of components, the estimated squared error and a matrix with rows corresponding to attributes (second mode) and columns corresponding to judges (third mode). In this matrix there are five rows with non-zero coefficients which correspond to the 4th, 6th 7th 8th and 9th attributes from all the judges (no selection is performed on the third mode).

```
R> summary(fit)
```

```
sNPLS model with 2 components and squared error of 0.047

 Coefficients:
         Z.1     Z.2     Z.3     Z.4     Z.5     Z.6     Z.7     Z.8
X.1    0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
X.2    0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
X.3    0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
X.4   -0.041  -0.041  -0.043  -0.049  -0.074  -0.040  -0.042  -0.029
X.5    0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
X.6    0.019   0.026   0.024   0.027   0.021   0.025   0.018   0.036
X.7    0.070   0.089   0.086   0.095   0.087   0.088   0.069   0.115
X.8   -0.030  -0.038  -0.037  -0.041  -0.038  -0.038  -0.030  -0.050
X.9   -0.009  -0.009  -0.010  -0.011  -0.017  -0.009  -0.009  -0.006
X.10   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
X.11   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
```

Listing 8.8: Summary with the coefficients of the fitted model.

To better understand and interpret the results, the plot function can be used to display different visualizations of the results. Values of the **T** (Figure 8.4), **U** (Figure 8.5), $\mathbf{W}^J$ (Figure 8.6 and Figure 8.8), and $\mathbf{W}^K$ (Figure 8.7 and

Figure 8.9) matrices can be plotted by changing the `type` parameter of the function. 1st and 2nd components are plotted by default, but they can be changed using the `comps` parameter.

```
R> plot(fit, type="T", cex.axis=1.2, cex.lab=1.2, cex=1.2,
+    las=1, bty="L")
R> plot(fit, type="U", cex.axis=1.2, cex.lab=1.2, cex=1.2,
+    las=1, bty="L")
```

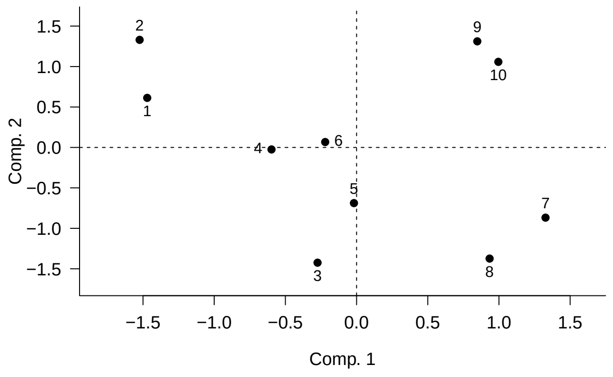

Figure 8.4: Score plot of the two first components in the **T** matrix

Figure 8.4 shows how the 10 samples spread over the two first components. It can be seen how the samples evolve every two observations, from left to right, which seems reasonable attending to their different composition. Figure 8.5 is similar, but related to the **y** scores.

```
R> plot(fit, type="Wj", cex.axis=1.2, cex.lab=1.2, cex=1.2,
+    las=1, bty="L")
R> plot(fit, type="Wk", cex.axis=1.2, cex.lab=1.2, cex=1.2,
+    las=1, bty="L")
```

On the other hand, Figure 8.6 presents the attributes and Figure 8.7 the judges that help in predicting the **y** variable, for each of the two components. It can be seen that the first component of the attributes mode is related to attributes 7, 8 and 6 (in descending order in absolute values); whereas attributes 4, 9 and 6 are related to the second component. This can be also derived from

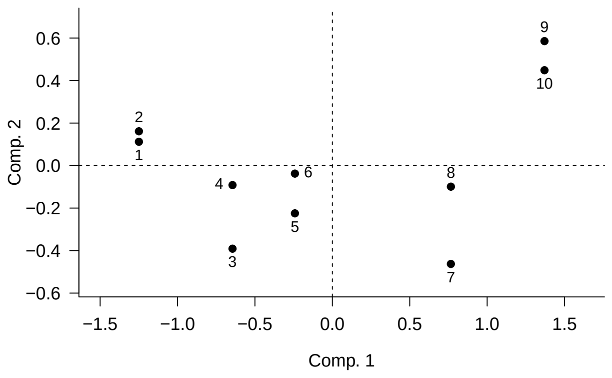

Figure 8.5: Score plot of the two first components in the **U** matrix

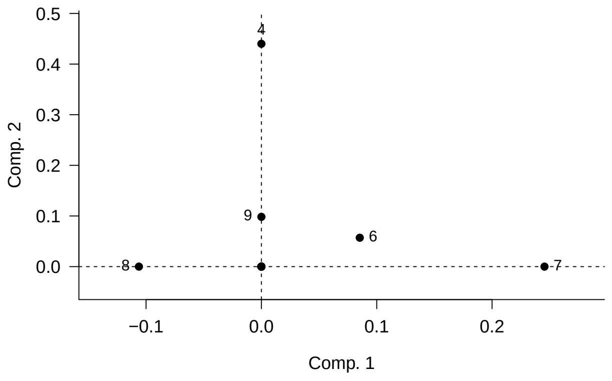

Figure 8.6: Weights Plot of the $\mathbf{W}^J$ weights matrix

Figure 8.8, which produces an equivalent graph. In this case, when trying to see what kind of relationship these attributes have with the judges' mode, Figure 8.9 provides easier-to-interpret results.

```r
R> plot(fit, type="variables", cex.axis=1.2, cex.lab=1.2,
+   cex=1.2, las=1, bty="L", lwd=2)
R> plot(fit, type="time", cex.axis=1.2, cex.lab=1.2, cex=1.2,
+   las=1, bty="L", lwd=2, xlab="Judge")
```

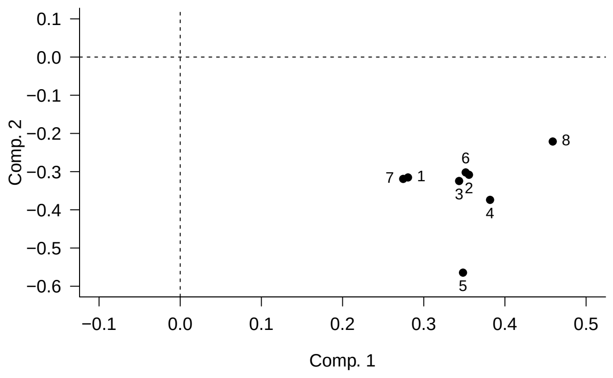

Figure 8.7: Weights Plot of the $\mathbf{W}^K$ weights matrix

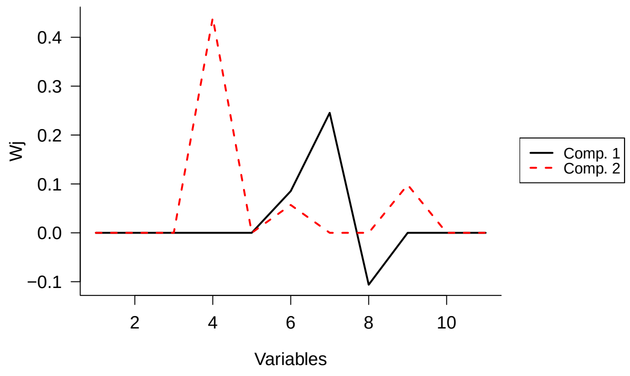

Figure 8.8: Plot of the second mode

It can be seen how, for the first component, there is approximately the same effect of all judges with respect to variables 7, 8 and 6 (those related to the first component in the attributes mode) when trying to predict the salt content. In this case, since the judges' weights show positive values, the higher the value of attributes 7 and 6, and the lower the value of attribute 8, the higher the salt content. For the second component, attributes 4, 9 and 6 are more influenced by judge 5, even though the rest of judges also have a similar effect on the salt content scoring (y variable). Since the weights of the judges' mode are all negative for the second component, the higher the value of attributes 4, 9 and 6, the lower the salt content. In this case, Figure 8.7 is less interpretable. However, depending on the case, one representation or another might provide easier-to-interpret results; so it is decided to keep both graphs in the package.

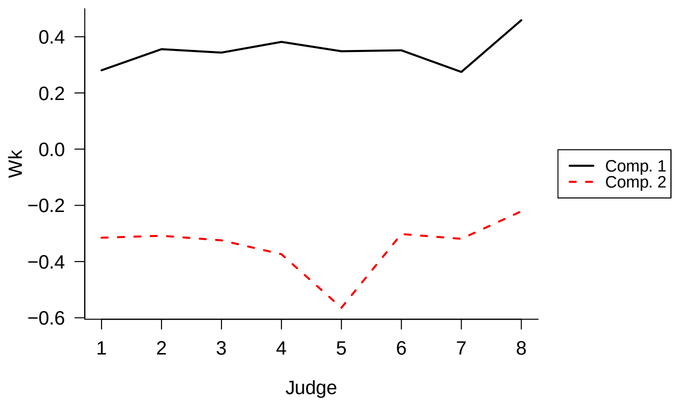

Figure 8.9: Plot of the third mode

As seen on these examples, all plots produced by the `plot.sNPLS` function are fully customizable, being able to change the titles, labels, orientation of labels, axis, size of the plot, etc., by using *base* plot parameters such as `las`, `cex`, `bty`, etc. Since they are base `R` plots, they can be affected by all `par` options and also be combined in a single figure by using the `mfrow` and `mfcol` options or the `layout` function.

Finally, the `predict` function can be used to make predictions using new $\underline{\mathbf{X}}$ data. As explained before, it makes use of the coefficients matrix $\mathbf{B}$ computed by the `Rmatrix` function (subsection 8.1.2). Here it is used to make a prediction from a new random $\underline{\mathbf{X}}$ array with 10 new observations (Listing 8.9). This function also has an optional parameter, `scale`, which defaults to `TRUE` for controlling the final scale of the predictions (original scale vs. post-processing scale).

```
newX <- array(sample(0:5, 10*11*8, replace = TRUE),
+     dim = c(10, 11, 8))
```

```r
predict(fit , newX)
```

```
              Y.1
 [1,]  1.4196482
 [2,]  0.7819288
 [3,]  0.9619593
 [4,]  1.0523621
 [5,]  1.2585099
 [6,]  0.7243910
 [7,]  0.6458718
 [8,]  1.2693292
 [9,]  0.8194011
 [10,]  1.2753320
```

Listing 8.9: Predictions of the model on new data `newX`.

The output of the function is a **Y** matrix with the 10 predicted values.

8.2.2 Performance

The package offers a complete set of functions for tuning, fitting and interpreting N-PLS and sNPLS models. As tuning is a computationally intensive method, all cross-validation functions in the package allow the use of parallelization through the *parallel* package and also use sparse matrices from the *Matrix* package. These two optimizations allow for speedups of up to 20 times faster computation times compared to non-parallelized computations with dense matrices. The use of sparse matrices also alleviates the use of RAM memory which, in the case of large data sets, could be a limiting factor in some computers. Figure 8.10 shows the improvements in computing times by using parallelization with different number of cores. As seen in the figure, the multicore scaling allows for tuning a model with a search grid length of 900 in an eight-core machine in less time than it takes to tune a model with a search grid of 90 in a single core running at the same speed.

The package also greatly benefits from the use of specialized Basic Linear Algebra Subprograms (BLAS) such as OpenBLAS, ATLAS or Intel® MKL, which can produce additional speedups of up to 10 times faster computation times (Xianyi, Qian, and Chothia 2014; E. Wang et al. 2014). In this case, the potential benefit is left to the decision of the end user, that should install and link one of these BLAS libraries to `R`.

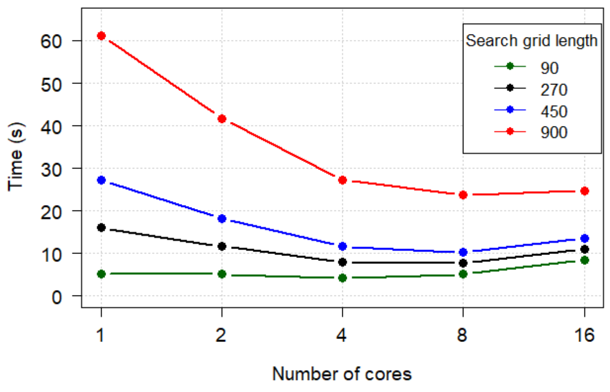

Figure 8.10: Computing times of `cv_snpls` under different conditions of grid length and number of cores. The function scales efficiently up until 8 cores

Chapter 9

Validation of the sparse N-PLS method and sNPLS package

In order to show and assess the performance of the sparse N-PLS model, different validations were performed to evaluate its properties regarding prediction accuracy and variable selection capabilities in different scenarios. It also was necessary to assess the usability of the sNPLS package in a real-world data set analysis. For this, simulated and real data sets were used, where the performance of N-PLS against the performance of sparse N-PLS was also compared regarding predictive capability. The following sections are based on the recent publication (Hervás, J. Prats-Montalbán, Garcia-Canaveras, et al. 2019) regarding the validation of the sNPLS method.

9.1 Random synthetic data sets

The performance of the implementation of the sparse N-PLS was first tested using data simulation. In total, fourteen different scenarios with different signal-to-noise ratios and different distributions of the data were assessed. Tuning of the hyperparameters of the models was performed with the `repeat_cv` function using 20 repetitions of 10-fold cross-validation following the methodology presented in chapter 8. Each simulation generated a three-way $\underline{\mathbf{X}}$ array with I=50 samples, J=50 variables and K=3 times, where variables were randomly sampled from different kinds of distributions (Poisson, Normal and Uniform) with varying parameters as follows:

1. If Normal: $\mathcal{X} \sim \mathcal{N}(\mu, \sigma)$ where $\mu \sim \mathcal{N}(10, 10)$ and $\sigma \sim \Gamma(5, 1)$
2. If Poisson: $\mathcal{X} \sim \mathcal{P}(\lambda)$ where $\lambda \sim \mathcal{N}(10, 2.5)$
3. If Uniform: $\mathcal{X} \sim \mathcal{U}(a, b)$ where $a \sim \mathcal{N}(10, 10)$ and $b \sim \mathcal{N}(100, 10)$

Only 5 out of the 50 variables were used to construct the response $\underline{\mathbf{Y}}$. They were chosen randomly from the $\underline{\mathbf{X}}$ array and assigned randomly the following coefficients: 0.4, 0.5, 0.6, 0.7 and 0.9. In the first run, only one of the three times of the third mode was involved in the creation of, in this case, vector $\mathbf{y}$. In the second run, the three times were involved, but with different coefficients for each variable. In all simulations, random Normal and Poisson errors were added in different amounts to $\mathbf{y}$. For each combination of type and amount of random error, simulations were repeated 100 times. Ability to select the real variables involved in $\mathbf{y}$ generation as well as median and 1^{st} and 3^{rd} quartiles of the mean squared error were estimated for each simulation run.

Results of the different simulations carried out on the synthetic data sets are provided in Table 9.1. Sparse N-PLS outperformed N-PLS regarding prediction error, with a mean squared error which was significantly lower in almost all simulation scenarios. This difference in prediction error increased as the signal-to-noise ratio decreased. Regarding variable selection, when performing sparse N-PLS, the true variables were almost always included in the selected model. Since simulations were repeated 100 times for each scenario, their results were summarized by giving descriptive statistics of the distribution of the results. Median number of true variables selected in each model was 5 (100%) in most of the simulations (9 out of 14). In the other 5 simulations, the median number of true variables selected was 4 (80%). These simulations where one of the true predictor variables was not selected were the ones consisting in the more complex and noisy models, with the three times of the third mode affecting $\mathbf{y}$ and lower signal-to-noise ratios. Also, a varying amount of other noise variables were erroneously included in the models. The amount of false positives (noise variables) that were included in the sparse N-PLS models increased remarkably as the signal-to-noise ratio of the data decreased, ranging from a median of 2 (4.4%) in the case of high signal-to-noise ratio data sets, to a median of 7 (15.6%) noise variables included in the low signal-to-noise, worst-case simulations.

	Mean Squared Error			**Variable selection**	
	N-PLS	**sNPLS**	**95% CI for difference**	**True variables selected**	**Noise variables selected**
One time					
Normal error (sd=1)	85.58 (71.36, 96.24)	66.23 (52.72, 90.54)	[-21.97, -9.72]	5 (5, 5)	3 (1, 6)
Normal error (sd=1.5)	103.16 (85.98, 114.21)	90.71 (70.54, 121.8)	[-16.32, 0.61]	5 (5, 5)	2 (1, 6)
Normal error (sd=2)	116.2 (101.05, 135.47)	99.17 (84.34, 121.62)	[-23.93, -8.02]	5 (4.75, 5)	4 (2, 12)
Normal error (sd=3)	166.33 (134.1, 199.73)	149.57 (117.21, 198.9)	[-26.84, 2.10]	5 (3, 5)	4 (1, 11)
Poisson error (mean=1)	86.46 (70.46, 99.98)	60.03 (45.83, 101.65)	[-26.66, -9.52]	5 (5, 5)	3 (1, 5)
Poisson error (mean=3)	109.21 (92.02, 131.57)	83.08 (67.22, 107.67)	[-32.09, -16.21]	5 (5, 5)	3 (1, 6)
Poisson error (mean=5)	138.29 (108.9, 153.86)	99.55 (78.81, 125.76)	[-40.58, -21.47]	5 (4, 5)	5 (2, 11)
Three times					
Normal error (sd=1)	165.1 (137.81, 202.86)	107.97 (91.79, 126.32)	[-68.02, -46.07]	5 (3, 5)	5 (2, 11)
Normal error (sd=1.5)	171.95 (147.6, 206.47)	115.75 (98.58, 133.18)	[-69.71, -49.51]	5 (3, 5)	6 (2, 12)
Normal error (sd=2)	195.57 (159.96, 230.1)	122.73 (100.32, 145.4)	[-85.29, -59.27]	4 (3, 5)	7 (3, 19)
Normal error (sd=3)	245.71 (199.71, 292.2)	130.97 (107.48, 169.2)	[-125.2, -92.03]	4 (3, 4.5)	7 (2, 18)
Poisson error (mean=1)	153.67 (135.87, 193.6)	103.92 (85.16, 123.6)	[-65.17, -45.07]	5 (3, 5)	5 (2, 5)
Poisson error (mean=3)	186.22 (154.9, 213.56)	115.77 (91.33, 134.17)	[-79.64, -57.47]	4 (3, 5)	6 (3, 13)
Poisson error (mean=5)	205.18 (169.3, 236.78)	118.58 (94.83, 140.54)	[-98.13, -73.24]	4 (3, 5)	6 (3, 15)

Table 9.1: Results of the analyses performed using N-PLS and sparse N-PLS on the different simulations. Median (1^{st}, 3^{rd} quartile) of the mean squared error and a 95% confidence interval for the difference in mean squared error between sparse N-PLS and N-PLS is also provided. True variables selected column indicates the median of the occasions these are included in the models, as well as the 1^{st} and 3^{rd} quartiles (True positives). Noise variables selected column presents analogous results for the Noise variables (False positives)

9.2 Data driven synthetic data sets

To further test the performance of the method in selecting highly correlated variables, simulated data resembling the ones of a toxicogenomics data set (Heijne et al. 2004) was analysed. In this work, the effect of the hepatotoxicant bromobenzene in rats was studied. Groups of rats were treated with different doses of this toxic compound dissolved in corn oil for a 48 hours period. At three time points from the start of the treatment, rats were sacrificed. Liver samples were used to extract mRNA for microarray profiling and blood and urine was used for metabolite profiling. Additionally, 21 physiological parameters were recorded: Glucose, A/G ratio, GSH, Body Weight, Creatin, GGT, Urea, Kidneys, Kidney/BW, Triglycerides, Liver, Albumin, Total Protein, ALP, Liver/BW, Bilirubin, LDH, Phospholipids, Cholesterol, ASAT, ALAT.

For this study, simulated profiles from 14 of these 21 physiological parameters were used to discriminate between two of the different groups (High and Low doses evolution). In order to compute these profiles, since they were not shown in (Heijne et al. 2004), it was taken advantage of those shown in Conesa et al. 2010, which correspond to the same dataset. 25 samples for high doses and 25 for low doses were simulated, adding to the pattern random normal noise, with standard deviation 0.1.

Figure 9.1 represents time course levels of the seven patterns corresponding to the 14 physiological parameters evaluated for high and low dose treatment groups.

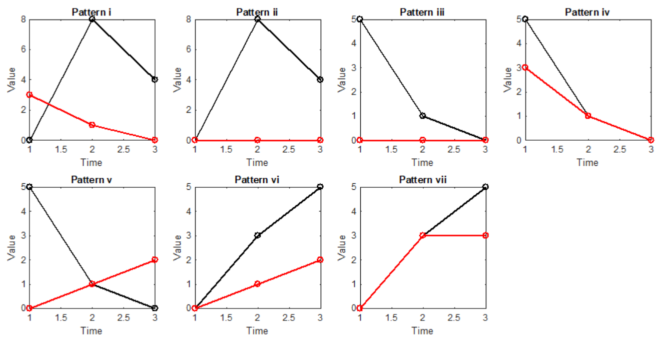

Figure 9.1: Patterns for the 14 different variables in both groups: High doses (red lines) and low doses (black lines).

These variables were grouped attending to their common patterns in the following seven groups: i) ALT, AST, LDH and GSH; ii) Creatin and Albumin; iii) Kidney and Cholesterol; iv) Liver, Phospholipids and Triglycerides; v) Glucose; vi) A/G Ratio and vii) Urea.

The goal with these simulations based on a real data set was to test the ability of the sparse N-PLS model in selecting those relevant variables that were associated to the response (discrimination between both groups) even in the case that these variables were correlated. On one hand, since the original lasso for least squares is known to suffer from multicollinearity as explained in chapter 3, there was a concern regarding the performance of the sparse N-PLS model in such scenarios. But, on the other hand, since the penalization is included at the model-fitting step of N-PLS, which deals excellently with multicollinearity, there was confidence that the method would be able to treat correlated variables adequately. More concretely, the expectations were that sNPLS would behave similarly to elastic net, where L1 and L2 penalty are combined chapter 3, since PLS is very similar to L2 penalization in the way it shrinks the coefficients (De Jong 1995). The results, presented in Table 9.2 which summarizes the model coefficients obtained in the analysis, confirmed the hypothesis of good performance in the presence of collinearity.

Variable	T1	T2	T3
Glucose	0	0	0
Phospoliphids	0	0	0
Kidney	0	0	0
Liver	0	0	0
Cholesterol	0	0	0
Tryglycerids	0	0	0
A/G ratio	0	0	0.077
Urea	0	0	0
Creatinine	0	0	0.082
Albumin	0	0	0.076
ALT	0	0	0.221
AST	0	0	0.221
LDH	0	0	0.221
GSH	0	0	0.101

Table 9.2: Coefficients of the model

Overall, results showed good agreement with the structure of the data. All the variables following the pattern i (i.e., ALT, AST, LDH and GSH) were selected by the method and similar coefficients were assigned. Additionally, Creatinine and Albumin (pattern ii) were also selected with similar coefficients. Interestingly, the A/G ratio (pattern vi), a variable uncorrelated to all the others was also selected. However, kidney and cholesterol were not selected by the model, probably because selection was also performed on the third mode and only the third element of the third mode was selected.

9.3 Real data sets

Simulated data sets provide a useful suitable first approach to test the performance of the new sNPLS method. However, to exemplify sNPLS and its utility in a more complex context, the proposed method was faced to the analysis of a real dataset, which was derived from a metabolomics study. In metabolomics data sets usually hundreds of variables, with high noise and high correlation, are obtained. This dramatically hinders biomarker discovery and variable selection for predictive models building. Thus, real processed rat serum samples were used to artificially generate two different groups by adding a set of standards at different final concentrations that additionally showed different trends along time (Figure 9.2). It was expected that this experimental design provided a suitable frame work for assessing sNPLS capabilities when facing real '-omics' data sets.

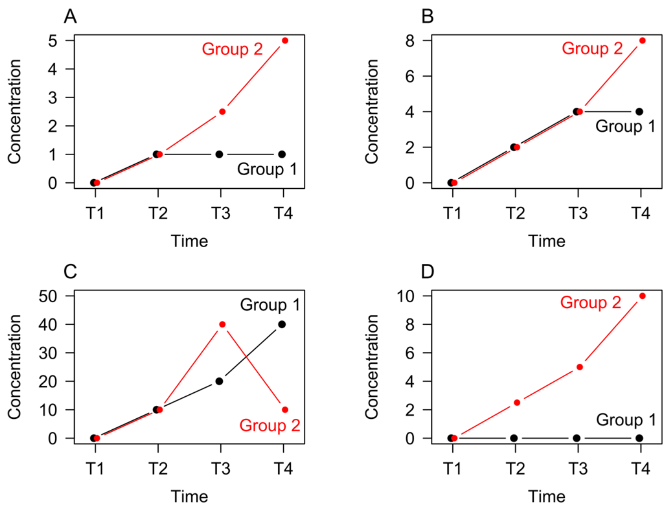

Figure 9.2: Expected patterns for the four different metabolite classes in group 1 (n=8) and group 2 (n=6).

In the following paragraphs, the whole experimental procedure for obtaining the real metabolomics data set is thoroughly described.

Liquid chromatography–mass spectrometry (LC-MS) analyses of rat serum samples were performed in an Agilent 1290 Infinity LC system coupled to an Agilent 6550 Q-TOF mass spectrometer equipped with an ESI source (Agilent Technologies, Santa Clara, CA, USA). LC-MS grade solvents (i.e. water, acetonitrile and methanol) were acquired from Fisher Scientific (Loughborough, UK). All the LC-MS additives and standards were acquired from Sigma–Aldrich/Fluka (Madrid, Spain). Metabolites were separated on an Zorbax SB-Aq column (100 x 2.1 mm; 1.8 µm (Agilent Technologies, Santa Clara, CA, USA). Mobile phases consisted of (A) 1mM ammonium fluoride and (B) acetonitrile. The separation was conducted under the following gradient at a flow of 0.3 mL/min: 0 min 3% (B); 0–2 min 40% (B); 2-5 min 7% (B); 5-7 min 50% (B); 7-12 min 100% (B); 12-16min 100% (B); 16-16.5 min 3% (B); 16.5-18 min 3% (B). Sample and column temperatures were maintained at 4 ºC and 40ºC, respectively. The injection volume was 5 µL. The instrument was tuned in the 50-1700 m/z range using an Agilent tune mix in 2GHz extended dynamic range mode (mass resolving power 25,000 FWHM). Detection was performed in ESI (-) mode in the 50-1000 m/z range. A reference solution (m/z 119.0360 and m/z 980.0164) was used to correct small mass drifts during acquisition. The following conditions were employed: capillary voltage, 3.5 kV; nozzle voltage -1.0 kV; fragmentor voltage, 175 V; gas temperature, 200 ºC; drying gas (nitrogen), 14 L/min; nebulizer gas (nitrogen), 35 psi; sheath gas temperature, 350 ºC; and sheath gas flow (nitrogen), 11 L/min. The acquisition rate was set at 4 spectra/s in all cases. Data preprocessing was performed using ProgenesisQI software (Nonlinear Dynamics, UK).

Six-week-old male Oncins France Strain A (OFA) rats (200–240 g) were purchased from Charles River (Barcelona, Spain) and acclimatized to laboratory conditions for at least 7 days. Animals were housed (12-h light-dark cycle, 21–25°C, 30–70% humidity, woodchip bedding) and fed ad libitum with a standard chow diet (Scientific Animal Food and Engineering, Augy, France). Rats were anesthetized with sodium thiobarbital (0.1 g/kg), and blood was collected by cardiac puncture. After coagulation and centrifugation (1,000 g for 10 min at 4°C), serum samples were aliquoted and stored at - 80°C until the analysis. All the experimental protocols were approved by the Institutional Animal Ethics Committee. 40 µL of serum sample were mixed with 120 µL of methanol. After vortexing, samples were kept at -20 ºC for 20 min. Samples were centrifuged (14000 g, 4 ºC, 15 min) and the supernatants transferred to clean tubes and evaporated to dryness. Samples were resuspended in 80 µL of water, centrifuged (14000 g, 4 ºC, 5 min), and the clean supernatants transferred to HPLC vials for their LC-MS analysis. Rats serum samples were separated in two groups of sizes 8 and 6 and subsequently fortified with a set of metabolites to generate the patterns already presented in Figure 9.2. Metabolites and final concentrations used are summarized in Table 9.3.

Variable Class	**Metabolites**
A	Capric Acid, Lauric, Acid, Myristic Acid, Myristoleic Acid, Palmitic Acid, Palmitoleic Acid, Octadecanoic Acid, Oleic Acid, Linoleic Acid, Linolenic Acid
B	Cholic acid, Glycocholic acid, Taurocholic acid, Chenoeoxycholic acid, Glycochenodeoxycholic acid, Taurochenodeoxycholic acid, Deoxycholic acid, Glycodeoxycholic acid, Taurodeoxycholic acid, Lithocholic aid, Glycolithocholic acid, Taurolithocholic acid
C	Valine, Leucine, Isoleucine, Phenylalanine, Methionine, Cysteine, Proline, Tyrosine, Aspartic acid, Alanine, Glycine, Lysine
D	Ornithine, Glutamate, Glutamine, Citrulline, Arginine, Argininosuccinic Acid, γ-glutamyl-glutamic acid, γ-glutamyl-glutamine, γ-glutamyl-2aminobutyric acid, ophthalmic acid

Table 9.3: List of metabolites for each variable class. Metabolites are grouped attending to their physical and chemical properties. Class **A**, comprises fatty acid; Class **B**, comprises bile acids; Class **C**, comprises amino acids; Class **D** comprises miscellaneous compounds

Results of the real data set analysis

The cross-validation procedure (20 repetitions of 5-fold cross-validation) selected as the optimum parameter values 30 features of $\mathbf{W}^{\mathrm{J}}$, 3 features of $\mathbf{W}^{\mathrm{K}}$ and 2 components. Therefore, 60 variables among the initial 1220 obtained from the LC-MS analysis (30 in each component) were selected by the final sparse N-PLS model. Out of the four variable classes which were different between both groups by design (Table 9.3), the model included at least one representative variable for each class. The model also included other variables not present among the four controlled variable classes, but many of them showed similar patterns to those included and could be derivatives or adducts of the original metabolites. Overall, the selection provided by the new model showed a quite feasible result, where not only the real assignable variables (added metabolites) but also those interfering ones could be selected. A list of all the selected variables is presented in Table 9.4. The first column list those variables selected by sparse N-PLS, while the second column indicates on which component these variables were selected. The third column shows wether these variables belong or not to one of the classes described in Table 9.3. Finally, column four shows whether those variables that do not belong to any of the assayed classes follows or not a pattern similar to those variables included in Table 9.3.

Variable	**Component**	**Variable Class**	**Profile similar to class**
V8, V16	1	A	-
V27, V28, V32	1	B	-
V54	1	C	-
V58	2	D	-
V187, V466, V853	1	-	A
V470	1	-	B
V388, V405, V422, V660, V661, V672	1	-	C
V112, V151, V179, V434, V449, V587, V608, V612, V967, V990	2	-	D
V95, V180, V527, V955, V1034, V1056, V1165, V1183, V1512, V2041, V2463, V2520, V2683	1	-	-
V897, V1235, V1322, V1354, V1378, V1389, V1535, V1601, V1627, V1647, V1711, V1715, V1729, V1873, V1935, V1945, V2011, V2077, V2180, V2616	2	-	-

Table 9.4: Variables selected by the final sNPLS model and their corresponding assigned variable classes

Interestingly, variables of the classes A, B and C and its derivatives or analogues were all exclusively selected in the first component and variables of the class D and its derivatives or analogues were all exclusively selected in the second component. Variables with different patterns to those of the four experimentally generated classes were included in both components, but were more prominent in the second one (13 in the first component versus 20 in the second).

Finally, the performance of the model was compared with the standard N-PLS model. To this end, the metabolomics data set was analyzed using both approaches. The sNPLS model clearly discriminated between the two rat groups (Figure 9.3A). However, similar groups' separation was also obtained by using the standard N-PLS (Figure 9.3B). The differences between both models appear when comparing Figure 9.3B vs Figure 9.3G, and Figure 9.3C vs Figure 9.3H, related to $\mathbf{W}^{\mathrm{J}}$ and $\mathbf{W}^{\mathrm{K}}$, respectively; or Figure 9.3D vs Figure 9.3I, and Figure 9.3E vs Figure 9.3J respectively, which are alternative representations. For interpretation purposes, it seems better to compare Figure 9.3B vs Figure 9.3G for $\mathbf{W}^{\mathrm{J}}$, and Figure 9.3E vs Figure 9.3J for $\mathbf{W}^{\mathrm{K}}$. For $\mathbf{W}^{\mathrm{J}}$, it seems quite clear that the selection made from sparse N-PLS allows a clear interpretation of the metabolites responsible for the separation between the two groups. In the first component are represented those metabolites belonging to the classes 1, 2 and 3. While, the second component is related to completely independent metabolites (with respect to component one), which could be related to the separation of rats 6 and 13. Many of these metabolites are from class 4, although some of them are not apparently related to any of the designed variable groups. These interpretations are much more hard to do when using standard N-PLS due to the high number of variables to deal with in the second (metabolites) mode, so from this perspective the proposed approach seems to improve the standard N-PLS model when trying to directly select the variables of interest (metabolites in this case). However, it should be highlighted that variable selection is out of the N-PLS scope. These results show that when variable selection is of prior relevance for interpretation or validation purposes sparse N-PLS come up as a valid alternative For the interpretation of $\mathbf{W}^{\mathrm{K}}$, Figure 9.3E and Figure 9.3J have been selected. These plots show, for the first component, a similar pattern, although a slight shift downwards is observed for the standard N-PLS. The similar trend observed for both methods strengths the use of the sparse N-PLS results, as it provides extra information as discussed above. However, regarding the second component, they did not provide the same result, which could be related to the clear separation of rats 6 and 13 observed in sparse N-PLS (Figure 9.3A).

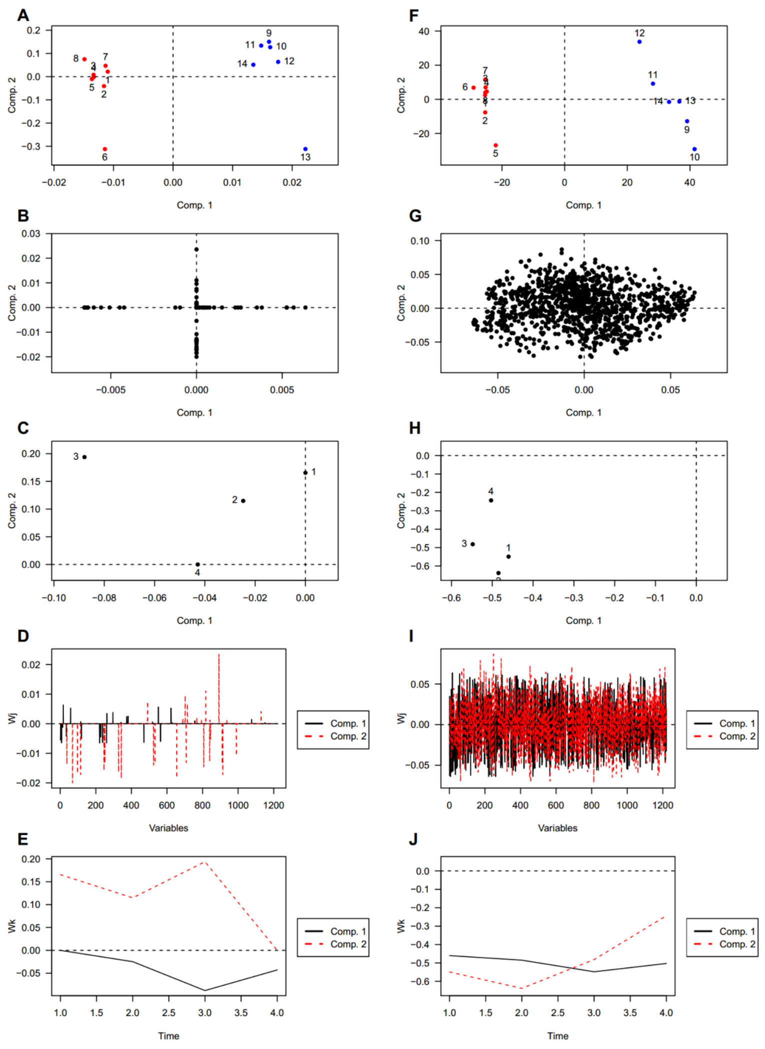

Figure 9.3: Plots of the sparse N-PLS model (left) and the standard N-PLS model (right). Score plots of the two first components in the **T** matrix (A, F); weighting plots of the $\mathbf{W}^{\mathrm{J}}$ matrix (B, G); weighting plot of the $\mathbf{W}^{\mathrm{K}}$ matrix (C, H); plots of the loadings of the second (D, I) and third (E, J) modes.

9.4 Comparison with other methods

As exposed in chapter 5, although N-PLS does not provide variable selection at the model fitting step, some methods have been developed to perform variable selection after the fitting of the model. Variable Importance in Projection (VIP) scores (Favilla, Durante, et al. 2013) and Selectivity Ratio (Rajalahti et al. 2009) have been used successfully in different studies (Favilla, Huber, et al. 2014; Mostafapour and Parastar 2015; Yun et al. 2016).

9.4.1 The hepatic toxicity data set

To better get a view of the similarities and differences between the sNPLS method and the VIP scores and Selectivity Ratio methods, a complex biological data set was analyzed using the three methods and compared their results. This data set consisted on measures of eight different toxicity parameters at four different time points in samples of hepatic cells treated with fifteen different drugs at three different concentrations. The study was designed to evaluate the hepatotoxicity in Upcyte human hepatocytes (UHH) in response to their exposure to the different selected drugs with Cmax (therapeutic peak plasmatic concentration), 0.2 Cmax and 5 Cmax concentrations. A scheme of the structure of the data is depicted in Figure 9.4. A complete description of the data set can be found in Tolosa et al. (2018).

Exploration of the data set showed clear trends over time for most of the different toxicity parameters. Some of these trends were highly dependent on concentration levels and others were more independent (Figure 9.5).

9.4.2 Data pre-processing and preparation

Based on the exploratory analysis and on the nature of the variables, all toxicity parameters were log-transformed prior to the analysis except for the viability variable, which was left untransformed. One of the drugs, methotrexate, was also excluded from the study because it had a high proportion (over 40%) of missing values. The $\underline{\mathbf{X}}$ array was set up including not only the main variables, but also the interaction between all toxicity variables and the concentration. The array was also mean centered and scaled to unit variance prior to modelling.

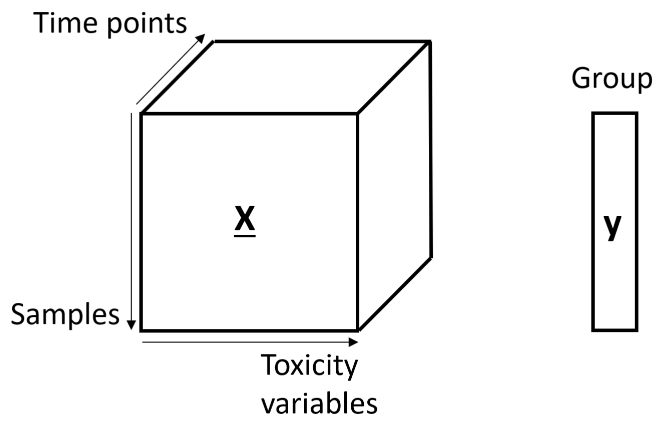

Figure 9.4: Structure of the toxicity outcomes data set. It is organized as a three-way array, with 42 samples (I), 8 variables (K) and 4 time points (K). The two drug groups, conform the response vector $\mathbf{y}$.

9.4.3 sNPLS results

Results of the repeated cross-validation estimated that the optimal combination of hyperparameter values was `ncomp=2`, `keepJ=4` and `keepK=4`. This combination was selected as the best in 21 of the 30 performed repetitions of the cross-validation procedure (Figure 9.6). Interestingly, the selection of the hyperparameter values shows a very low uncertainty compared to other problems with real data such as the one previously presented in this chapter (section 9.3) or even the simple bread data set presented in chapter 8.

With these results, a sNPLS model was adjusted using the suggested optimal combination of hyperparameter values. The coefficients of the model are presented in Table 9.5. As can be seen in the table, the variables MMP, lipids and viability were not selected by the model. This is consistent with the data exploration depicted in Figure 9.5, where these variables show no clear differences between both groups. Another interesting result is the selection of the mitosox variable only for the highest (5Cmax) concentration level. This effect also agrees with the data representation in Figure 9.5. For the other variables, the day 14 seems to have the highest effect, getting the largest coefficients in each case. To further interpret the results, different plots of the model are represented in Figure 9.7.

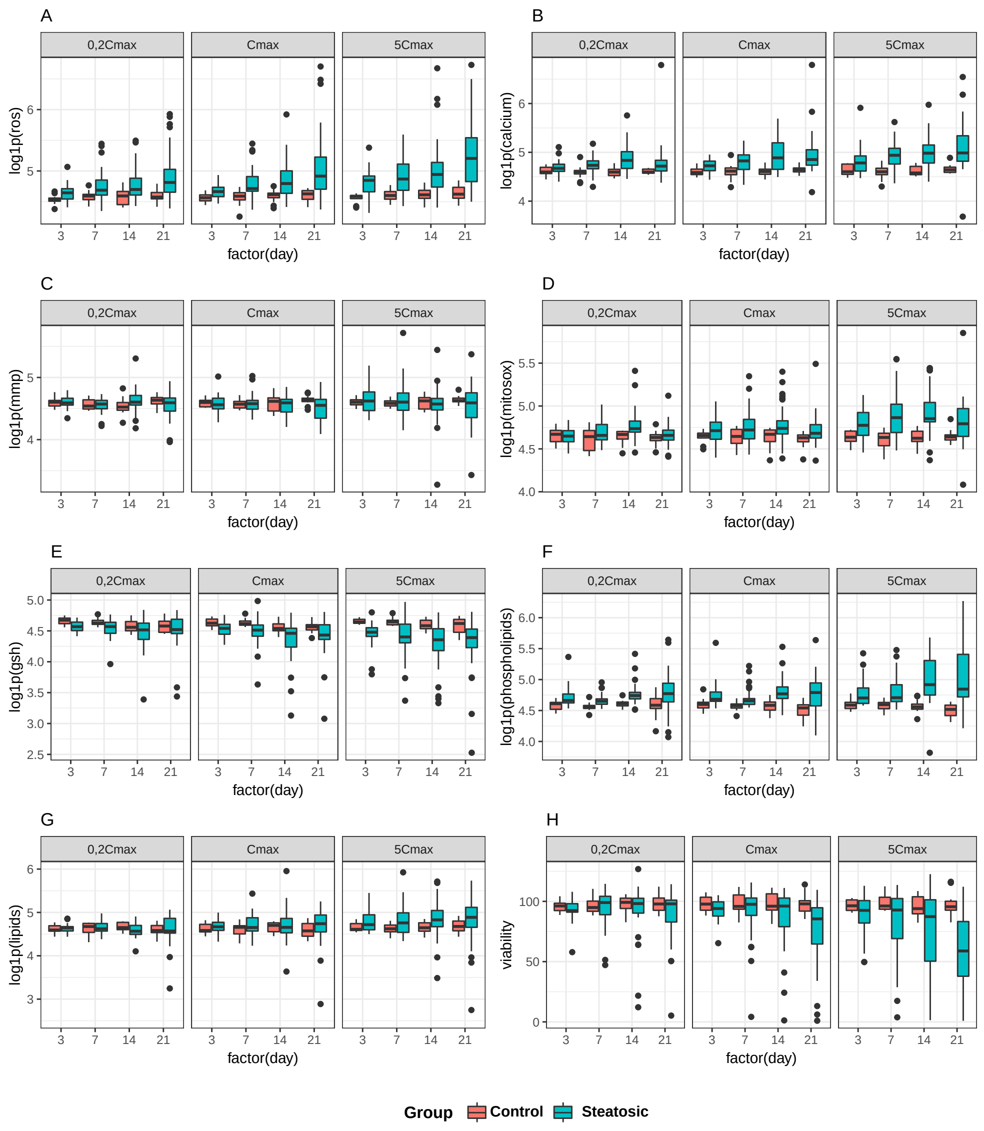

Figure 9.5: Exploratory analysis of the toxicity data set. Trends over time at each studied concentration for the two drug groups for the different toxicity parameters. (**A** Reactive Oxygen Species, **B** Calcium, **C** Matrix Metalloproteinases, **D** Mitochondrial Superoxide Indicator, **E** Reduced glutathione, **F** Phospholipids, **G** Lipids and **H** Viability.

	Day 3	**Day 7**	**Day 14**	**Day 21**
ROS | 0.2Cmax	0.059	0.069	0.090	0.061
Calcium | 0.2Cmax	0.008	0.009	0.012	0.008
MMP | 0.2Cmax	0	0	0	0
Mitosox | 0.2Cmax	0	0	0	0
GSH | 0.2Cmax	0	0	0	0
Phospholipids | 0.2Cmax	0.049	0.048	0.051	0.04
Lipids | 0.2Cmax	0	0	0	0
Viability | 0.2Cmax	0	0	0	0
ROS | Cmax	0.024	0.028	0.037	0.025
Calcium | Cmax	0.034	0.040	0.052	0.036
MMP | Cmax	0	0	0	0
Mitosox | Cmax	0	0	0	0
GSH | Cmax	-0.008	-0.009	-0.012	-0.008
Phospholipids | Cmax	0.069	0.067	0.072	0.057
Lipids | Cmax	0	0	0	0
Viability | Cmax	0	0	0	0
ROS | 5Cmax	0	0	0	0
Calcium | 5Cmax	0.028	0.03	0.036	0.026
MMP | 5Cmax	0	0	0	0
Mitosox | 5Cmax	0.006	0.006	0.006	0.005
GSH | 5Cmax	-0.105	-0.102	-0.11	-0.087
Phospholipids | 5Cmax	0.029	0.028	0.030	0.024
Lipids | 5Cmax	0	0	0	0
Viability | 5Cmax	0	0	0	0

Table 9.5: Coefficients of the model for predicting drug group in the toxicity data set. Notably, the variables MMP, Lipids and Viability are not selected by the model in any concentration level.

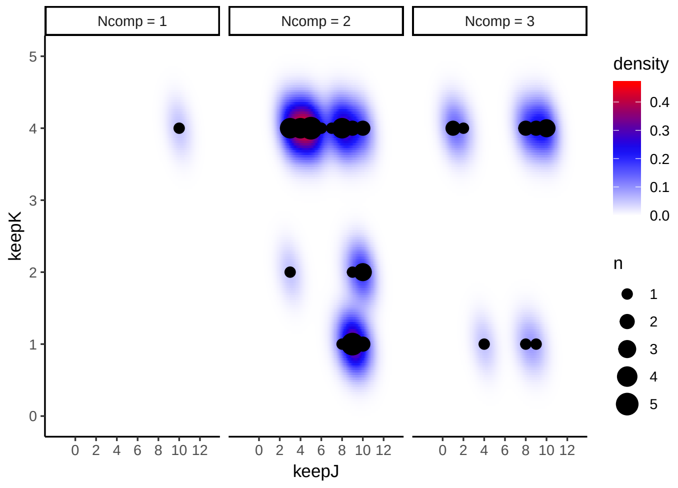

Figure 9.6: Results of the repeated cross-validation procedure for the toxicity data set. The highest density is located at the combination `ncomp=2`, `keepJ=4` and `keepK=4`.

9.4.4 Selectivity ratio results

After fitting the sNPLS model, a standard N-PLS model was fitted to the data and the Selectivity ratio for each variable was estimated. Code for performing SR on N-way arrays was implemented in `R` based on Rajalahti et al. (2009) and is included in the `sNPLS` package since version 0.4. SR values for each variable at each component are presented in Table 9.6.

For this data set, estimation of the SR does not yield highly discriminative results between the different variables. Only MMP at 0.2 Cmax, Lipids at 0.2 Cmax and, to a lesser extent, MMP at 5 Cmax, display a lower SR value compared to the other variables. This two variables, Lipids and MMP, were not selected by the sNPLS model, so there is a slight coincidence in these results. A plot of the different SR values for each variable in each component is depicted in Figure 9.8.

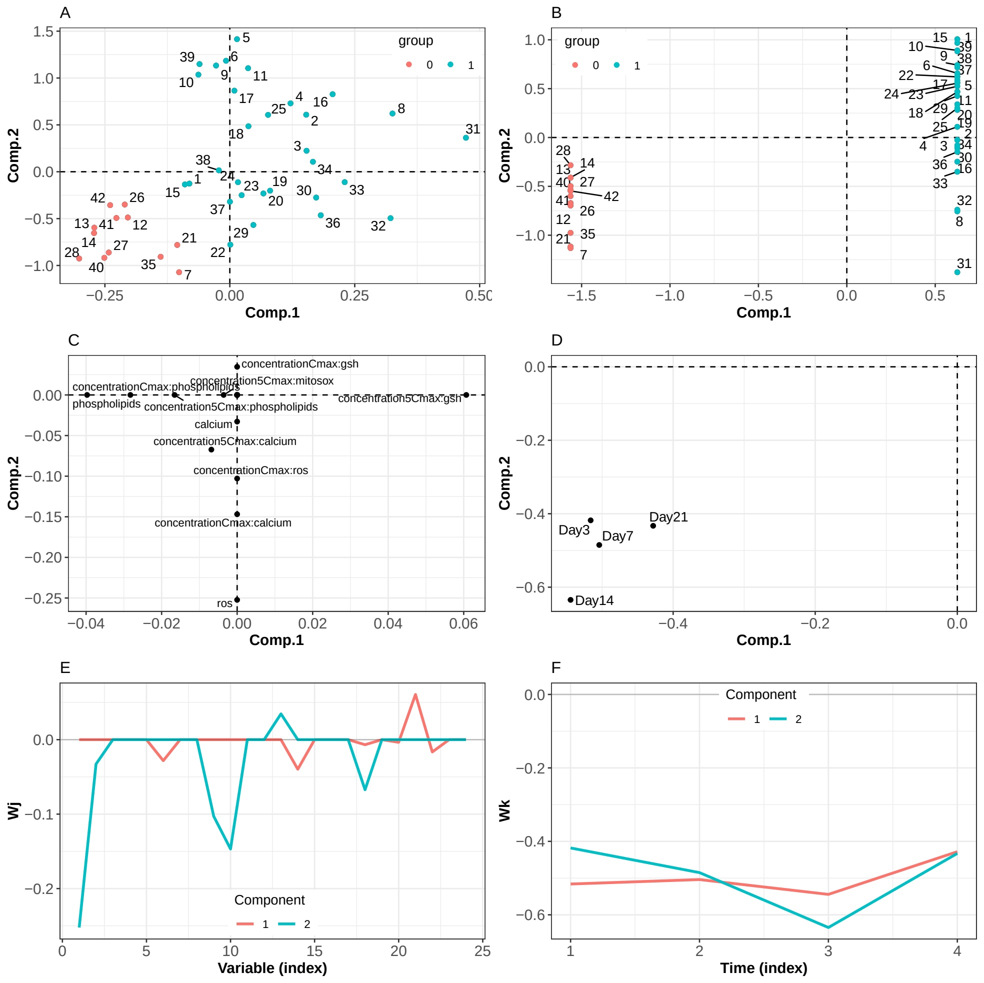

Figure 9.7: Plots of the fitted sNPLS model for the toxicity data set. Score plot of the two first components in the **T** matrix (A); Score plot of the two first components in the **U** matrix (B); weighting plot of the $\mathbf{W}^{\mathrm{J}}$ matrix (C); weighting plot of the $\mathbf{W}^{\mathrm{K}}$ matrix (D); plots of the loadings of the second (E) and third (F) modes

	Component 1	**Component 2**
ROS | 0.2Cmax	1.044	1.052
Calcium | 0.2Cmax	1.047	1.044
MMP | 0.2Cmax	0.532	1.029
Mitosox | 0.2Cmax	1.065	1.035
GSH | 0.2Cmax	1.039	1.043
Phospholipids | 0.2Cmax	1.036	1.068
Lipids | 0.2Cmax	0.717	1.078
Viability | 0.2Cmax	0.966	1.046
ROS | Cmax	1.059	1.051
Calcium | Cmax	1.055	1.046
MMP | Cmax	1.030	1.045
Mitosox | Cmax	1.065	1.046
GSH | Cmax	1.041	1.040
Phospholipids | Cmax	1.042	1.064
Lipids | Cmax	1.085	1.061
Viability | Cmax	1.074	1.049
ROS | 5Cmax	1.067	1.049
Calcium | 5Cmax	1.069	1.047
MMP | 5Cmax	0.979	0.953
Mitosox | 5Cmax	1.068	1.045
GSH | 5Cmax1.060	1.060	1.048
Phospholipids | 5Cmax	1.058	1.048
Lipids | 5Cmax	1.076	1.058
Viability | 5Cmax	1.080	1.048

Table 9.6: Selectivity Ratio values for the N-PLS model fitted to the toxicity data set.

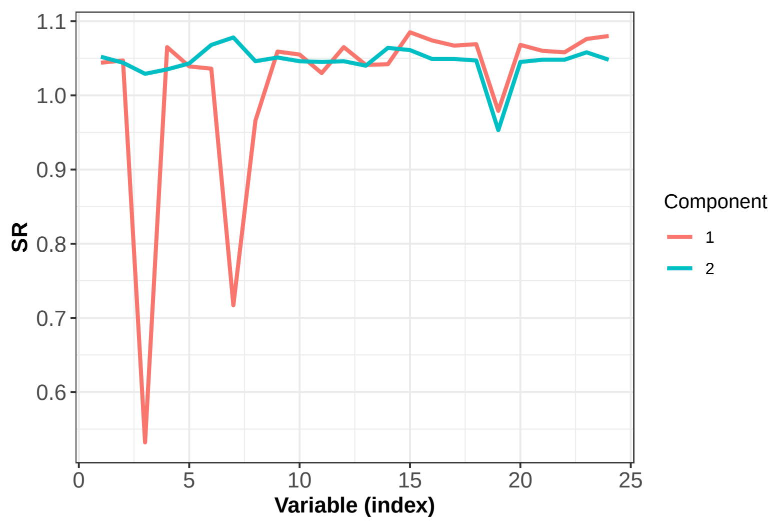

Figure 9.8: SR plot for the N-PLS model of the toxicity data set. Almost all variables get a similar score, so almost no variable selection can be performed.

9.4.5 VIP results

Additionally to the SR results, VIP scores were also estimated for the fitted standard N-PLS model. Code for estimating VIP scores on N-way arrays was implemented in `R` based on the paper by Favilla, Durante, et al. (2013) and is included in the `sNPLS` package since version 0.4.1. VIP scores for each variable at each component are presented in Table 9.7.

Interestingly, in the case of VIP scores, the concordance with sNPLS is greater compared to the concordance between SR and sNPLS. VIP scores, clearly selects the variables ROS Cmax, calcium Cmax, ROS 5Cmax, calcium 5Cmax, mitosox 5Cmax, phospholipids 5Cmax and lipids 5Cmax as important variables. Of these seven variables, six were also selected by the sNPLS method. sNPLS also selected ROS 0.2Cmax, calcium 0.2Cmax, phospholipids 0.2Cmax, GSH Cmax and phospholipids Cmax as important variables. For these, the VIP scores method gives a relatively high score to ROS 0.2Cmax and Calcium 0.2Cmax variables, but clearly considers non-important phospholipids 0.2Cmax (VIP score of 0.31), GSH Cmax (VIP score of 0.13) and phospholipids Cmax (VIP score of 0.43). A plot comparing the results of the VIP score method and the sNPLS method is presented in Figure 9.9.

	VIP score
ROS | 0.2Cmax	0.78
Calcium | 0.2Cmax	0.83
MMP | 0.2Cmax	0.028
Mitosox | 0.2Cmax	0.073
GSH | 0.2Cmax	0.066
Phospholipids | 0.2Cmax	0.31
Lipids | 0.2Cmax	0.041
Viability | 0.2Cmax0.013	0.013
ROS | Cmax	2.47
Calcium | Cmax	1.92
MMP | Cmax	0.033
Mitosox | Cmax	0.21
GSH | Cmax	0.13
Phospholipids | Cmax	0.43
Lipids | Cmax	0.27
Viability | Cmax	0.13
ROS | 5Cmax	5.09
Calcium | 5Cmax	3.42
MMP | 5Cmax	0.079
Mitosox | 5Cmax	1.25
GSH | 5Cmax1.060	0.47
Phospholipids | 5Cmax	3.48
Lipids | 5Cmax	1.97
Viability | 5Cmax	0.50

Table 9.7: VIP score values for the *N*-PLS model fitted to the toxicity data set.

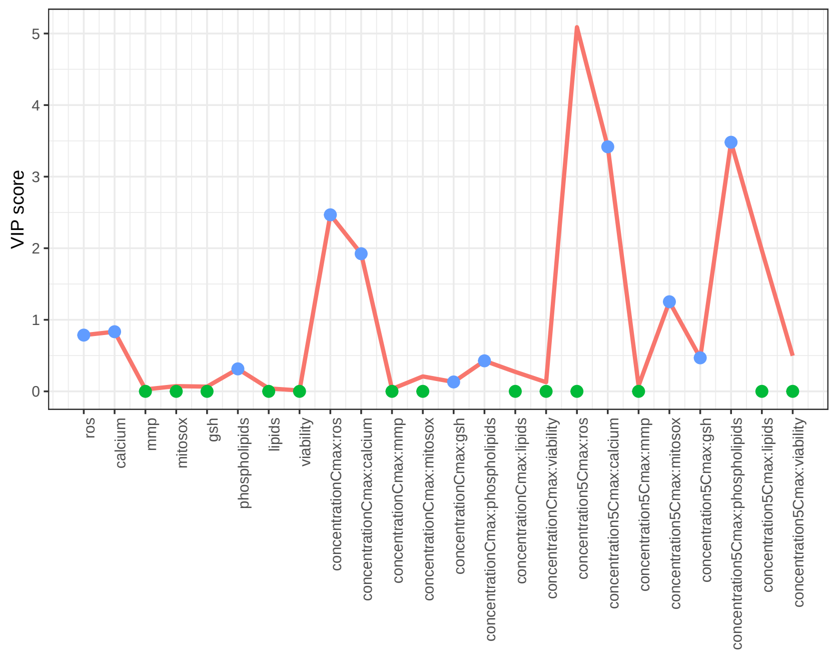

Figure 9.9: Comparison of variable selection results of sNPLS method and VIP scores method for the N-PLS model of the toxicity data set. Variables selected by sNPLS are depicted as a blue point at the same height as the VIP score value for that variable and non-selected variables are depicted as a blue point at zero. Ideally, VIP score peaks and blue points should match.

Although the agreement is not perfect, most blue points in Figure 9.9 match a peak of VIP score so, in this specific data set, both methods give comparable results.

Part IV

Conclussions and future work

Chapter 10

Conclusions and future work

10.1 Developed topics and contributions

This thesis has covered the issues and difficulties in analyzing metabolomic data and the different techniques able to deal with those problems. A special focus has been given to the treatment and analysis of three-way and, by extension, multi-way data sets and the development and implementation of a variable selection procedure integrated at the model-fitting step.

First, a review of the properties and particularities of 'omic' data and, more specifically, of metabolomic data was performed. The main characteristics of these data are their multicollinearity, the large number of variables and their organization in complex data structures. There are a wide range of methods available for analyzing metabolomic data sets. In this thesis, many of these techniques have been presented and explained; after a review of the limitations of the classical statistical methods, such as univariate tests, the usefulness of multivariate statistical projection methods (PCA, PLS) as well as penalization methods (ridge regression, lasso and elastic net) for the analysis of metabolomic data has been assessed. After this, the thesis has focused on complex data structures such as three-way or multi-way arrays and the tools for their analysis regarding compression methods such as Tucker3 and PARAFAC and also predictive modelling methods such as N-PLS.

A new method for combining the N-PLS algorithm with L1-penalization has been developed. Additionally, other variable selection methods such as VIP and SR have been implemented within the N-PLS algorithm. Related to the variable selection, these methods allow for the construction of simpler models than standard N-PLS. This makes the models more interpretable and less prone to overfitting.

Furthermore, the variable selection performance of the three new methods has been compared, obtaining similar results with sNPLS and VIP and potentially different results with SR; with any of them being superior to the other two in some cases and inferior in other cases.

The new developed methods have been implemented in an R package called `sNPLS`. This package includes not only the main sNPLS algorithm but also a complete set of functions for fitting predictive models with three-way data structures, including parallelized cross-validation and repeated cross-validation procedures, processing of the data such as centering, scaling and unfolding and a wide array of plots for interpretation of the modelling results. Additionally, this is the only package for performing N-PLS regression models in `R`, so with its development the applicability of three-way methods to the `R` user community has been expanded.

The methods and their implementation in `R` has also been validated using different synthetic data sets, showing lower prediction error than standard N-PLS and good performance in variable selection. The method has also been validated for the case of high multicollinearity between variables, proving that the combination of L1-penalization and N-PLS projection behaves similarly to elastic net, in the sense that is it is able to select all correlated variables instead of discarding most of them and selecting only one as happens with lasso.

10.2 Future work and research lines

The current version of the package is the 0.4.01, which is the 59th update of the initial package. At its current state, the package is fully functional, being able of fitting sparse and standard N-PLS models, efficiently performing cross-validation and repeated cross-validation and providing useful plots for the interpretation of results. It also incorporates VIP and SR variable selection procedures within the N-PLS algorithm. The package recently incorporated generic base `R` functions for models such as `coef`, `summary` and `predict` and also has fixed all the detected bugs of the first versions of the software. According to the statistics given by *CRAN*, the `sNPLS` package averages around

275 downloads per month and has accumulated more than 6700 downloads. These are similar statistics to the two other `R` packages for three-way analysis, so reception of the package seems to be great among the three-way community of `R` users.

For future versions of the package one of the priorities is including more options for the application of the penalization factor. Currently, the L1-penalty is determined by choosing the number of non-zero elements in $\mathbf{w}^{\mathrm{J}}$ and $\mathbf{w}^{\mathrm{K}}$, which can be determined by K-fold cross-validation or arbitrarily by the user. Ideally, the L1-penalty values should come from a continuous distribution, being able to take any value, not just discrete values as of now. Related to the application of penalty, there is ongoing work being performed on studying the possibility of adding an option for L0 penalty, This penalty can be applied using the hard-thresholding operator (Zou 2006), so it should be straightforward to implement in future versions of the package.

Also the cross-validation procedure does not currently try all the possible combinations of the hyperparameter space, because when there is more than one component, all components are forced to have the same number of `keepJ` and `keepK` values. This constrain is only applied because of computational constrains, so different alternatives are being studied to be able to improve efficiency of the cross-validation procedure, thus removing the need for constraining the hyperparameter space. One of such alternatives, which will be implemented in future versions of the package will be the substitution of the grid search procedure for a random search procedure which has been demonstrated to be more efficient and also provide better estimates of uncertainty (Bergstra and Bengio 2012).

Another addition regarding the tuning of the parameters and the cross-validation will be the possibility of personalizing the objective metric. Currently, the $RMSE$ is the only option for the optimization of the hyperparameter space. However, in the case of N-PLS-DA, other metrics related to classification methods such as missclassification rate or AUC (Bradley 1997) would be more reasonable and should be implemented in future versions of the package.

Related to the optimization of the model by tuning of the hyperparameter space, alternative methods to cross-validation will be studied and implemented, such as information criteria based methods like Akaike information criterion (AIC) (Akaike 1974) or Bayesian information criterion (BIC) (Schwarz et al. 1978). These alternative methods should be much more efficient computationally, so they would be a great addition to the package.

It has also been already experimented with the estimation of confidence intervals for the estimates based on resampling, so this will be another addition to the package in the upcoming versions. Related to this, the possibility of estimating the uncertainty in the variable selection procedure will be estudied and implemented if feasible.

Other planned addition is the modification of the plot function so that it creates objects with all the information about the plot instead of only producing them. This way, the user will be able to fully personalize their plots and also make their plots fully reproducible without the new of repeating a large analysis.

Finally, another objective is adding more useful functions to the package for providing a comprehensive framework for three-way analyses. Functions for performing Tucker and PARAFAC models like those already implemented in the `ThreeWay` and `PTAk` packages, more pre-processing functions and an extended documentation of all of them, including some vignettes, will be introduced in future versions of the package.

The development of a new method for performing variable selection at the model-fitting step in N-PLS and its software implementation as an `R` package has opened a large range of research lines that should be explored in the future. These research lines can be framed in three different categories: methodological developments, software development and practical application of the method to real problems in biomedical research.

Regarding methodological developments, future research lines include:

1. Study of other possible penalizations such as L0 or adaptive L1 penalization. The field of penalized regression methods is a very active research branch. Since the sNPLS method is a combination of a projection based method with a penalization method, advances in both fields could be incorporated into this technique. Adaptive L1 penalization is specially appealing for its oracle properties (Zou 2006).

2. Development of alternative methods for the selection and/or tuning of the hyperparameter space. Using $RMSE$ for the tuning of the hyperparameters results in a theoretically optimal prediction error performance at a cost of non-optimal variable selection performance. Finding an appropriate criterion for achieving optimal variable selection performance would greatly increase the flexibility and utility of the method.

3. Derivation of standard errors and confidence intervals for sNPLS to further improve the inference capabilities of the method.

In the case of software development, the proposed future research lines include:

1. Improvement of the `sNPLS R` package as exposed in section 10.2. This includes adding all the functionality developed by the methodological research lines, but also improving the efficiency of the computations and adding more data processing options and plots.
2. Developing an alternative to the search grid method. Random search is one candidate, but other options should be studied such as gradient search or genetic algorithms.
3. Study the impact of the use of different BLAS libraries on the performance of the package and implement optimizations in the algorithm based on this results.

Finally, regarding practical application of the method to real biomedical problems, the potential number of research lines is immense, so a generic sample is provided:

1. Application of the sNPLS method to biomarkers search in longitudinal studies.
2. Development of prediction models for outcome in repeated measures studies.
3. Use of the sNPLS method for the understanding of complex biological systems

Bibliography

Akaike, Hirotugu (1974). “A new look at the statistical model identification”. In: *IEEE transactions on automatic control* 19.6, pp. 716–723 (cit. on p. 111).

Akarachantachote, Noppamas, Seree Chadcham, and Kidakan Saithanu (2014). “Cutoff threshold of variable importance in projection for variable selection”. In: *International Journal of Pure and Applied Mathematics* 94.3, pp. 307–322 (cit. on p. 46).

Alin, Aylin (2010). “Multicollinearity”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.3, pp. 370–374 (cit. on p. 20).

Andersson, Claus A and Rasmus Bro (2000). “The N-way toolbox for MATLAB”. In: *Chemometrics and intelligent laboratory systems* 52.1, pp. 1–4 (cit. on p. 44).

Auret, Lidia and Chris Aldrich (2011). “Empirical comparison of tree ensemble variable importance measures”. In: *Chemometrics and Intelligent Laboratory Systems* 105.2, pp. 157–170 (cit. on p. 29).

Austin, Peter C and Ewout W Steyerberg (2015). “The number of subjects per variable required in linear regression analyses”. In: *Journal of clinical epidemiology* 68.6, pp. 627–636 (cit. on p. 20).

Babyak, Michael A (2004). “What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models”. In: *Psychosomatic medicine* 66.3, pp. 411–421 (cit. on p. 13).

Baker, Monya (2016). “1,500 scientists lift the lid on reproducibility”. In: *Nature News* 533.7604, p. 452 (cit. on pp. 3, 4).

Barh, Debmalya and Vasco Azevedo, eds. (2018). *Omics Technologies and Bio-Engineering.* Academic Press, p. iv. ISBN: 978-0-12-804659-3. DOI: `https://doi.org/10.1016/B978-0-12-804659-3.00047-6` (cit. on p. 5).

Bartel, Jörg, Jan Krumsiek, and Fabian J Theis (2013). “Statistical methods for the analysis of high-throughput metabolomics data”. In: *Computational and structural biotechnology journal* 4.5, e201301009 (cit. on p. 29).

Bates, Douglas and Martin Maechler (2018). *Matrix: Sparse and Dense Matrix Classes and Methods.* R package version 1.2-14 (cit. on p. 72).

Beger, Richard D et al. (2016). “Metabolomics enables precision medicine:“a white paper, community perspective””. In: *Metabolomics* 12.9, p. 149 (cit. on p. 6).

Begley, C Glenn and John PA Ioannidis (2015). “Reproducibility in science: improving the standard for basic and preclinical research”. In: *Circulation research* 116.1, pp. 116–126 (cit. on p. 3).

Bender, Ralf and Stefan Lange (2001). “Adjusting for multiple testing—when and how?” In: *Journal of clinical epidemiology* 54.4, pp. 343–349 (cit. on p. 15).

Benjamini, Yoav and Yosef Hochberg (1995). “Controlling the false discovery rate: a practical and powerful approach to multiple testing”. In: *Journal of the royal statistical society. Series B (Methodological)*, pp. 289–300 (cit. on pp. 14, 31).

Benjamini, Yoav and Daniel Yekutieli (2001). “The control of the false discovery rate in multiple testing under dependency”. In: *Annals of statistics*, pp. 1165–1188 (cit. on p. 15).

Bergstra, James and Yoshua Bengio (2012). “Random search for hyper-parameter optimization”. In: *Journal of Machine Learning Research* 13.Feb, pp. 281–305 (cit. on p. 111).

Bingol, Kerem (2018). “Recent Advances in Targeted and Untargeted Metabolomics by NMR and MS/NMR Methods”. In: *High-throughput* 7.2, p. 9 (cit. on p. 7).

Bland, J Martin and Douglas G Altman (1995). “Multiple significance tests: the Bonferroni method”. In: *Bmj* 310.6973, p. 170 (cit. on p. 14).

Bowling, Francis G and Mervyn Thomas (2014). “Analyzing the metabolome”. In: *Clinical bioinformatics*. Springer, pp. 31–45 (cit. on p. 28).

Bradley, Andrew P (1997). “The use of the area under the ROC curve in the evaluation of machine learning algorithms”. In: *Pattern recognition* 30.7, pp. 1145–1159 (cit. on p. 111).

Breiman, Leo (2001). “Random forests”. In: *Machine learning* 45.1, pp. 5–32 (cit. on pp. 29, 32).

Bro, Rasmus (1996). “Multiway calibration. multilinear pls”. In: *Journal of chemometrics* 10.1, pp. 47–61 (cit. on p. 41).

— (1997). “PARAFAC. Tutorial and applications”. In: *Chemometrics and intelligent laboratory systems* 38.2, pp. 149–171 (cit. on p. 40).

— (1998). “Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications”. PhD thesis. Københavns UniversitetKøbenhavns Universitet, LUKKET: 2012 Det Biovidenskabelige Fakultet for Fødevarer, Veterinærmedicin og NaturressourcerFaculty of Life Sciences, LUKKET: 2012 Institut for FødevarevidenskabDepartment of Food Science, LUKKET: 2012 Kvalitet og TeknologiQuality & Technology (cit. on pp. 42, 76).

Bro, Rasmus, Age K Smilde, and Sijmen de Jong (2001). “On the difference between low-rank and subspace approximation: improved model for multilinear PLS regression”. In: *Chemometrics and Intelligent Laboratory Systems* 58.1, pp. 3–13 (cit. on p. 41).

Broadhurst, David I and Douglas B Kell (2006). “Statistical strategies for avoiding false discoveries in metabolomics and related experiments”. In: *Metabolomics* 2.4, pp. 171–196 (cit. on p. 13).

Büscher, Jörg Martin et al. (2009). “Cross-platform comparison of methods for quantitative metabolomics of primary metabolism”. In: *Analytical chemistry* 81.6, pp. 2135–2143 (cit. on p. 7).

Bylesjö, Max et al. (2006). “OPLS discriminant analysis: combining the strengths of PLS-DA and SIMCA classification”. In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 20.8-10, pp. 341–351 (cit. on p. 48).

Carroll, J Douglas and Jih-Jie Chang (1970). “Analysis of individual differences in multidimensional scaling via an N-way generalization of “Eckart-Young” decomposition”. In: *Psychometrika* 35.3, pp. 283–319 (cit. on p. 40).

Castro-Conde, Irene and Jacobo de Uña-Álvarez (2015). “Adjusted p-values for SGoF multiple test procedure”. In: *Biometrical Journal* 57.1, pp. 108–122 (cit. on p. 15).

Chiu, Chih-Chiun and Yuan Yao (2013). “Multiway elastic net (MEN) for final product quality prediction and quality-related analysis of batch processes”. In: *Chemometrics and Intelligent Laboratory Systems* 125, pp. 153–165 (cit. on p. 50).

Chong, Il-Gyo and Chi-Hyuck Jun (2005). “Performance of some variable selection methods when multicollinearity is present”. In: *Chemometrics and intelligent laboratory systems* 78.1-2, pp. 103–112 (cit. on pp. 27, 45).

Conesa, Ana et al. (2010). “A multiway approach to data integration in systems biology based on Tucker3 and N-PLS”. In: *Chemometrics and Intelligent Laboratory Systems* 104.1, pp. 101–111 (cit. on p. 88).

Costa, Fabricio F (2014). “Big data in biomedicine”. In: *Drug discovery today* 19.4, pp. 433–440 (cit. on p. 3).

Dayhoff, Judith E and James M DeLeo (2001). “Artificial neural networks: opening the black box”. In: *Cancer: Interdisciplinary International Journal of the American Cancer Society* 91.S8, pp. 1615–1635 (cit. on p. 29).

De Jong, Sijmen (1995). “PLS shrinks”. In: *Journal of chemometrics* 9.4, pp. 323–326 (cit. on pp. 25, 89).

Diong, Joanna et al. (2018). “Poor statistical reporting, inadequate data presentation and spin persist despite editorial advice”. In: *PloS one* 13.8, e0202121 (cit. on p. 4).

Duarte, Edson and Jacques Wainer (2017). “Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters”. In: *Pattern Recognition Letters* 88, pp. 6–11 (cit. on p. 60).

Efron, Bradley and Robert J Tibshirani (1994). *An introduction to the bootstrap.* CRC press (cit. on p. 29).

Eklund, Anders, Thomas E Nichols, and Hans Knutsson (2016). “Cluster failure: why fMRI inferences for spatial extent have inflated false-positive rates”. In: *Proceedings of the National Academy of Sciences*, p. 201602413 (cit. on p. 4).

Farrés, Mireia et al. (2015). “Comparison of the variable importance in projection (VIP) and of the selectivity ratio (SR) methods for variable selection and interpretation”. In: *Journal of Chemometrics* 29.10, pp. 528–536 (cit. on p. 49).

Favilla, Stefania, Caterina Durante, et al. (2013). “Assessing feature relevance in NPLS models by VIP”. In: *Chemometrics and Intelligent Laboratory Systems* 129, pp. 76–86 (cit. on pp. 46, 50, 96, 103).

Favilla, Stefania, Alexa Huber, et al. (2014). “Ranking brain areas encoding the perceived level of pain from fMRI data”. In: *Neuroimage* 90, pp. 153–162 (cit. on p. 96).

Ferrario, Manuela et al. (2016). “Mortality prediction in patients with severe septic shock: a pilot study using a target metabolomics approach”. In: *Scientific reports* 6, p. 20391 (cit. on p. 28).

Fiehn, Oliver (2001). “Combining genomics, metabolome analysis, and biochemical modelling to understand metabolic networks”. In: *International Journal of Genomics* 2.3, pp. 155–168 (cit. on p. 6).

Freund, Yoav, Robert E Schapire, et al. (1996). “Experiments with a new boosting algorithm”. In: *Icml*. Vol. 96. Citeseer, pp. 148–156 (cit. on p. 29).

Gagnier, Joel J and Hal Morgenstern (2017). “Misconceptions, misuses, and misinterpretations of p values and significance testing”. In: *JBJS* 99.18, pp. 1598–1603 (cit. on p. 4).

Gao, Xiaoyi, Joshua Starmer, and Eden R Martin (2008). “A multiple testing correction method for genetic association studies using correlated single nucleotide polymorphisms”. In: *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society* 32.4, pp. 361–369 (cit. on p. 15).

Giordani, Paolo, Henk AL Kiers, Maria Antonietta Del Ferraro, et al. (2014). “Three-way component analysis using the R package ThreeWay”. In: *Journal of Statistical Software* 57.7, pp. 1–23 (cit. on p. 44).

Gonzalez-Billalabeitia, Enrique et al. (2017). “Prognostic significance of venous thromboembolic events in disseminated germ cell cancer patients”. In: *JNCI: Journal of the National Cancer Institute* 109.4 (cit. on p. 28).

Goztepe, Kerim (2017). “De Facto Language of Data Science: The R Project”. In: *Journal of Management and Information Science* 4.4, pp. 104–107 (cit. on p. 43).

Harshman, Richard A (1970). “Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis”. In: (cit. on p. 40).

Hassall, Kirsty L and Andrew Mead (2018). “Beyond the one-way ANOVA for’omics data”. In: *BMC bioinformatics* 19.7, p. 199 (cit. on p. 10).

Hastie, Trevor, Jerome Friedman, and Robert Tibshirani (2001). “Model assessment and selection”. In: *The elements of statistical learning*. Springer, pp. 193–224 (cit. on p. 22).

Hawkins, Douglas M (2004). “The problem of overfitting”. In: *Journal of chemical information and computer sciences* 44.1, pp. 1–12 (cit. on p. 13).

Heijne, Wilbert HM et al. (2004). “Bromobenzene-induced hepatotoxicity at the transcriptome level”. In: *Toxicological Sciences* 79.2, pp. 411–422 (cit. on p. 88).

Heinze, Georg and Daniela Dunkler (2017). “Five myths about variable selection”. In: *Transplant International* 30.1, pp. 6–10 (cit. on p. 10).

Hervás, D, JM Prats-Montalbán, JC Garcia-Canaveras, et al. (2019). “Sparse N-way Partial Least Squares by L1-penalization”. In: *Chemometrics and Intelligent Laboratory Systems* (cit. on p. 85).

Hervás, D, JM Prats-Montalbán, A Lahoz, et al. (2018). “Sparse N-way partial least squares with R package sNPLS”. In: *Chemometrics and Intelligent Laboratory Systems* (cit. on pp. 50, 70).

Hochberg, Yosef and Yoav Benjamini (1990). “More powerful procedures for multiple significance testing”. In: *Statistics in medicine* 9.7, pp. 811–818 (cit. on p. 31).

Hoerl, Arthur E and Robert W Kennard (1970). “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1, pp. 55–67 (cit. on p. 24).

Horgan, Richard P and Louise C Kenny (2011). “‘Omic’technologies: genomics, transcriptomics, proteomics and metabolomics”. In: *The Obstetrician & Gynaecologist* 13.3, pp. 189–195 (cit. on p. 5).

Hotelling, Harold (1933). “Analysis of a complex of statistical variables into principal components.” In: *Journal of educational psychology* 24.6, p. 417 (cit. on p. 16).

Ibáñez, Mariam et al. (2018). “The modular network structure of the mutational landscape of Acute Myeloid Leukemia”. In: *PLOS ONE* 13.10, pp. 1–16. DOI: `10.1371/journal.pone.0202926` (cit. on p. 28).

Ihaka, Ross and Robert Gentleman (1996). “R: a language for data analysis and graphics”. In: *Journal of computational and graphical statistics* 5.3, pp. 299–314 (cit. on p. 43).

Ioannidis, John PA (2005). “Why most published research findings are false”. In: *PLoS medicine* 2.8, e124 (cit. on p. 3).

Ioannidis, JP (2017). “Statistical biases in science communication: What we know about them and how they can be addressed”. In: *The Oxford Handbook of the Science of Science Communication*, p. 103 (cit. on p. 4).

Johnson, Randall C et al. (2010). “Accounting for multiple comparisons in a genome-wide association study (GWAS)”. In: *BMC genomics* 11.1, p. 724 (cit. on p. 14).

Jolliffe, Ian T (1982). “A note on the use of principal components in regression”. In: *Applied Statistics*, pp. 300–303 (cit. on p. 17).

Joy, Jeena, Salice Peter, and Neetha John (2013). “Denoising using soft thresholding”. In: *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 2.3, pp. 1027–1032 (cit. on p. 52).

Karaman, Ibrahim (2017). “Preprocessing and pretreatment of metabolomics data for statistical analysis”. In: *Metabolomics: From Fundamentals to Clinical Applications*. Springer, pp. 145–161 (cit. on p. 32).

Karnebeek, Clara DM van et al. (2018). “The role of the clinician in the multi-omics era: are you ready?” In: *Journal of Inherited Metabolic Disease*, pp. 1–12 (cit. on p. 5).

Katz, Mitchell H (2011). *Multivariable analysis: a practical guide for clinicians and public health researchers*. Cambridge university press. Chap. 10, pp. 140–159 (cit. on p. 10).

Kettaneh, Nouna, Anders Berglund, and Svante Wold (2005). “PCA and PLS with very large data sets”. In: *Computational Statistics & Data Analysis* 48.1, pp. 69–85 (cit. on p. 19).

Khare, Ashish and Uma Shanker Tiwary (2005). “Soft-thresholding for denoising of medical images—a multiresolution approach”. In: *International Journal of Wavelets, Multiresolution and Information Processing* 3.04, pp. 477–496 (cit. on p. 52).

Kiers, Henk AL and Iven Van Mechelen (2001). “Three-way component analysis: Principles and illustrative application.” In: *Psychological methods* 6.1, p. 84 (cit. on p. 39).

Kroonenberg, Pieter M et al. (2016). “My multiway analysis: From Jan de Leeuw to TWPack and back”. In: *Journal of Statistical Software* 73, p. 22 (cit. on p. 37).

Krstajic, Damjan et al. (2014). “Cross-validation pitfalls when selecting and assessing regression and classification models”. In: *Journal of cheminformatics* 6.1, p. 10 (cit. on pp. 60, 73).

Lameski, Petre et al. (2015). “SVM parameter tuning with grid search and its impact on reduction of model over-fitting”. In: *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing.* Springer, pp. 464–474 (cit. on p. 60).

Lankinen, Maria et al. (2010). “Dietary carbohydrate modification alters serum metabolic profiles in individuals with the metabolic syndrome”. In: *Nutrition, Metabolism and Cardiovascular Diseases* 20.4, pp. 249–257 (cit. on p. 28).

Lay Jr, Jackson O et al. (2006). “Problems with the “omics””. In: *TrAC Trends in Analytical Chemistry* 25.11, pp. 1046–1056 (cit. on p. 3).

Lê Cao, Kim-Anh et al. (2008). “A sparse PLS for variable selection when integrating omics data”. In: *Statistical applications in genetics and molecular biology* 7.1 (cit. on p. 57).

Leardi, Riccardo (2005). “Multi-way analysis with applications in the chemical sciences, age smilde, Rasmus Bro and Paul Geladi, Wiley, Chichester, 2004, ISBN 0-471-98691-7, 381 pp”. In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 19.2, pp. 119–120 (cit. on p. 67).

Leek, Jeff et al. (2017). “Five ways to fix statistics”. In: *Nature* 551.7682, pp. 557–559 (cit. on p. 4).

Leibovici, Didier G (2010). "Spatio-temporal multiway decompositions using principal tensor analysis on k-modes: The R package PTAk". In: *Journal of Statistical Software* 34.10, pp. 1–34 (cit. on p. 44).

Liu, Shuangzhe and Gõtz Trenkler (2008). "Hadamard, Khatri-Rao, Kronecker and other matrix products". In: *Int. J. Inf. Syst. Sci* 4.1, pp. 160–177 (cit. on p. 41).

Liu, Xiaojing et al. (2015). "High resolution metabolomics with acyl-CoA profiling reveals widespread remodeling in response to diet". In: *Molecular & cellular proteomics*, mcp–M114 (cit. on p. 28).

Loh, Wei-Yin (2014). "Fifty years of classification and regression trees". In: *International Statistical Review* 82.3, pp. 329–348 (cit. on p. 29).

Mahadevan, Sankar et al. (2008). "Analysis of metabolomic data using support vector machines". In: *Analytical Chemistry* 80.19, pp. 7562–7570 (cit. on p. 29).

Marino, Michael J (2014). "The use and misuse of statistical methodologies in pharmacology research". In: *Biochemical pharmacology* 87.1, pp. 78–92 (cit. on p. 4).

Marx, Vivien (2013). *Biology: The big challenges of big data* (cit. on p. 3).

Mathew, AK and VC Padmanaban (2013). "Metabolomics: the apogee of the omics trilogy". In: *Int J Pharm Pharm Sci* 5.2, pp. 45–8 (cit. on p. 7).

MATLAB (2018). *version 9.5 (R2018b)*. Natick, Massachusetts: The MathWorks Inc. (cit. on p. 44).

McCulloch, Charles E (2000). "Generalized linear models". In: *Journal of the American Statistical Association* 95.452, pp. 1320–1324 (cit. on p. 13).

McGibbon, Robert T et al. (2016). "Osprey: Hyperparameter optimization for machine learning". In: *J. Open Source Software* 1.5, p. 00034 (cit. on p. 62).

Meglen, Robert R (1992). "Examining large databases: a chemometric approach using principal component analysis". In: *Marine Chemistry* 39.1-3, pp. 217–237 (cit. on p. 17).

Mehmood, Tahir et al. (2012). "A review of variable selection methods in partial least squares regression". In: *Chemometrics and Intelligent Laboratory Systems* 118, pp. 62–69 (cit. on p. 26).

Moco, Sofia et al. (2007). "Metabolomics technologies and metabolite identification". In: *TrAC Trends in Analytical Chemistry* 26.9, pp. 855–866 (cit. on p. 7).

Mostafapour, Sara and Hadi Parastar (2015). "N-way partial least squares with variable importance in projection combined to GC× GC-TOFMS as a reliable tool for toxicity identification of fresh and weathered crude oils". In: *Analytical and bioanalytical chemistry* 407.1, pp. 285–295 (cit. on p. 96).

Nygård, Ståle et al. (2008). "Partial least squares Cox regression for genome-wide data". In: *Lifetime Data Analysis* 14.2, pp. 179–195 (cit. on p. 29).

Orešič, M (2009). "Metabolomics, a novel tool for studies of nutrition, metabolism and lipid dysfunction". In: *Nutrition, Metabolism and Cardiovascular Diseases* 19.11, pp. 816–824 (cit. on p. 6).

Palsson, Bernhard (2002). "In silico biology through "omics"". In: *Nature biotechnology* 20.7, p. 649 (cit. on p. 5).

Prats-Montalbán, José Manuel (2005). "Control estadistico de procesos mediante analisis multivariante de imagenes". In: (cit. on p. 39).

R Core Team (2018). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria (cit. on p. 43).

Rajalahti, Tarja et al. (2009). "Biomarker discovery in mass spectral profiles by means of selectivity ratio plot". In: *Chemometrics and Intelligent Laboratory Systems* 95.1, pp. 35–48 (cit. on pp. 48, 50, 96, 100).

Saccenti, Edoardo et al. (2014). "Reflections on univariate and multivariate analysis of metabolomics data". In: *Metabolomics* 10.3, pp. 361–374 (cit. on p. 9).

Schwarz, Gideon et al. (1978). "Estimating the dimension of a model". In: *The annals of statistics* 6.2, pp. 461–464 (cit. on p. 111).

Simon, Noah et al. (2011). "Regularization paths for Cox's proportional hazards model via coordinate descent". In: *Journal of statistical software* 39.5, p. 1 (cit. on p. 29).

Smilde, Age, Rasmus Bro, and Paul Geladi (2005). *Multi-way analysis: applications in the chemical sciences.* John Wiley & Sons (cit. on p. 42).

Smith, Colin A et al. (2005). "METLIN: a metabolite mass spectral database". In: *Therapeutic drug monitoring* 27.6, pp. 747–751 (cit. on p. 7).

Strasak, Alexander M et al. (2007). "Statistical errors in medical research–a review of common pitfalls". In: *Swiss medical weekly* 137.3-4, pp. 44–49 (cit. on p. 31).

Tibshirani, Robert (1996). "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288 (cit. on pp. 26, 32).

Tolosa, Laia et al. (2018). "Long-term and mechanistic evaluation of drug-induced liver injury in Upcyte human hepatocytes". In: *Archives of Toxicology*, pp. 1–14 (cit. on p. 96).

Tsai, Flora S and Kap Luk Chan (2007). "Dimensionality reduction techniques for data exploration". In: *Information, Communications & Signal Processing, 2007 6th International Conference on.* IEEE, pp. 1–5 (cit. on p. 17).

Tucker, Ledyard R (1966). "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* 31.3, pp. 279–311 (cit. on p. 39).

Venables, William N and Brian D Ripley (2002). "Tree-based methods". In: *Modern Applied Statistics with S.* Springer, pp. 251–269 (cit. on p. 29).

Vetter, Thomas R and Edward J Mascha (2018). “Unadjusted Bivariate Two-Group Comparisons: When Simpler is Better”. In: *Anesthesia & Analgesia* 126.1, pp. 338–342 (cit. on p. 10).

Viant, Mark R et al. (2017). “How close are we to complete annotation of metabolomes?” In: *Current opinion in chemical biology* 36, pp. 64–69 (cit. on p. 7).

Vujačić, Ivan, Antonino Abbruzzo, and Ernst Wit (2015). “A computationally fast alternative to cross-validation in penalized Gaussian graphical models”. In: *Journal of Statistical Computation and Simulation* 85.18, pp. 3628–3640 (cit. on p. 61).

Wang, Endong et al. (2014). “Intel math kernel library”. In: *High-Performance Computing on the Intel® Xeon Phi™*. Springer, pp. 167–188 (cit. on p. 83).

Wang, Kai and Diana Abbott (2008). “A principal components regression approach to multilocus genetic association studies”. In: *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society* 32.2, pp. 108–118 (cit. on p. 16).

Wishart, DS, D Tzur, C Knox, et al. (2007). “HMDB: the Human Metabolome Database. Nucleic Acids Res”. In: *Database* D521-6 (cit. on p. 7).

Wold, Svante, Arnold Ruhe, et al. (1984). “The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses”. In: *SIAM Journal on Scientific and Statistical Computing* 5.3, pp. 735–743 (cit. on p. 32).

Wold, Svante, Michael Sjöström, and Lennart Eriksson (2001). “PLS-regression: a basic tool of chemometrics”. In: *Chemometrics and intelligent laboratory systems* 58.2, pp. 109–130 (cit. on pp. 20, 21, 45).

Xianyi, Zhang, Wang Qian, and Zaheer Chothia (2014). “OpenBLAS”. In: *URL: http://xianyi. github. io/OpenBLAS* (cit. on p. 83).

Yáñez, Yania et al. (2015). “Two independent epigenetic biomarkers predict survival in neuroblastoma”. In: *Clinical epigenetics* 7.1, p. 16 (cit. on p. 28).

Yang, Yuhong (2005). “Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation”. In: *Biometrika* 92.4, pp. 937–950 (cit. on p. 60).

Yun, Yong-Huan et al. (2016). “Variable importance analysis based on rank aggregation with applications in metabolomics for biomarker discovery”. In: *Analytica chimica acta* 911, pp. 27–34 (cit. on p. 96).

Zhang, Yiyun, Runze Li, and Chih-Ling Tsai (2010). “Regularization parameter selections via generalized information criterion”. In: *Journal of the American Statistical Association* 105.489, pp. 312–323 (cit. on p. 61).

Zou, Hui (2006). “The adaptive lasso and its oracle properties”. In: *Journal of the American statistical association* 101.476, pp. 1418–1429 (cit. on pp. 111, 112).

Zou, Hui and Trevor Hastie (2005). “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320 (cit. on pp. 27, 32, 60).

Zou, Hui, Trevor Hastie, and Robert Tibshirani (2006). “Sparse principal component analysis”. In: *Journal of computational and graphical statistics* 15.2, pp. 265–286 (cit. on p. 52).