Universitat Oberta de Catalunya

Integrative learning for heterogeneous block-wise missing omics data

Sergi Baena i Miret

Area 5, subarea 1: statistics and bioinformatics
Master's degree in Bioinformatics and Biostatistics

Ferran Reverter Comes and Esteban Vegas Lozano

June 2, 2022

This license lets others distribute, remix, adapt, and build upon your work, even commercially, as long as they credit you for the original creation. This is the most accommodating of licenses offered. Recommended for maximum dissemination and use of licensed materials.
https://creativecommons.org/licenses/by-nc/3.0

FINAL WORK CARD

Title:	Integrative learning for heterogeneous block-wise missing omics data
Author:	Sergi Baena i Miret
Tutor:	Ferran Reverter Comes and Esteban Vegas Lozano
Date of delivery:	June 2, 2022
Studies:	Master's degree in Bioinformatics and Biostatistics
Area:	Area 5, subarea 1: statistics and bioinformatics
Language:	English
Number of credits:	15
Keywords:	Block-wise missing data, multi-source, optimization regression model, machine learning, omics data, exposome data

Resum

En moltes ocasions la informació que es pot recollir no està completa, ja que per a algunes observacions no totes les *fonts* de dades estan disponibles (el que es coneix com a dades faltants per blocs) per la qual cosa la pregunta que sorgeix és com es podria implementar un procés d'integració amb dades que contenen blocs faltants basat en una aproximació de tipus *Lasso*, que després es podria aplicar a dades òmiques reals. De fet, en aquesta tesi resoldrem un problema d'optimització de regressió consistent en un model d'aprenentatge de característiques unificades per a blocs heterogenis faltants de dades (o fins i tot completes) que realitzin anàlisis tant a nivell de característiques com de fonts simultàniament.

La novetat d'aquesta tesi es basa en que encara que es pot trobar la formulació i l'optimització teòrica del problema, no hem pogut trobar la seva implementació de codi enlloc, per la qual cosa ens ha estat impossible (fins que no hem aconseguit implementar-lo) donar una valoració raonable del model. De fet, per a l'avaluació del model (l'estudi de la seva efectivitat i rendiment) hem utilitzat dades simulades generades per un model de regressió lineal i dades reals extretes d'un nou projecte de recerca col·laboratiu anomenat *Human Early-Life Exposome* (HELIX).

Tot plegat, en aquest manuscrit hem estudiat un model d'aprenentatge binivell de característiques motivat per les dades de l'*exposome* i hem implementat un codi que tant serveix per a dades completes com amb blocs faltants. Concretament, hem introduït un model d'aprenentatge de característiques unificades per a dades completes, que conté diversos models convexos clàssics que s'han estès fàcilment per gestionar el cas més difícil: el de les dades faltants per blocs. Al final hem aconseguit presentar un model d'optimització de regressió que donades les dades completes o faltants per blocs, podem obtenir-ne informació per tal de fer prediccions per a dades que tinguin una estructura similar. En particular, hem observat resultats excel·lents per a les dades simulades i resultats força bons per a les dades d'*exposome*.

Abstract

On many occasions the information that one can gather is not complete, since for some observations not all data sources are available (what is known as block-wise missing data) so the question that arises is how we could implement an integrative process with block-wise missing data based on a Lasso's type approximation that then could be applied to real omics data. Indeed, in this thesis we will solve an optimization regression problem consisting on a unified feature learning model for heterogeneous block-wise missing (or even complete) data that performs both feature-level and source-level analysis simultaneously.

The novelty on this thesis relies on that although one can find the formulation and the theoretical optimization of the problem, we have not been able to find its code implementation anywhere, so it has been impossible for us (until we have succeed implementing them) to give a reasonable evaluation of the model. Indeed, for the evaluation of the model (the study of its effectiveness and performance) we will use synthetic data generated by a linear regression model and real data drawn from a new collaborative research project called the Human Early-Life Exposome (HELIX).

All in all, in this manuscript we have studied a bi-level feature learning model motivated by the exposome data and we have implemented a code that approaches for both complete and block-wise missing data. Specifically, we have introduced a unified feature learning model for complete data, which contains several classical convex models that has been easily extended to handling the more challenging case: the block-wise missing data. At the end we have succeed in presenting an optimization regression model that given complete or block-wise missing data, we can obtain information from it in order to make predictions for similar structured data. In particular, we have observed great results for the simulated data and quite good results for this exposome data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This short chapter is intended to be a brief description of our project.

1.1 Context and justification of the thesis

The *Omics technologies* are high-throughput biochemical assays that, in a comprehensive and simultaneous way, measure molecules of the same type from a biological sample. For example, transcriptomics measure transcripts; metabolomics quantify metabolites while proteomics quantify proteins; genomics profile DNA... Then, *omics data* are those consisting on all the data generated by Omics technologies applied to a set of samples.

Indeed, the "omics" notion refers to the fact that all (or nearly all) instances of the target molecular space are measured in the assay. Initially, omics experiments tended to concentrate on one type of assay (i.e., transcriptomics) so that provide single omics data. However, it is believed that a joint learning of multiple data sources (in that case, from multiple different omics) is beneficial as different data sources may contain complementary information, which should be properly integrated and leveraged. In fact, machine learning algorithms have being increasingly used to analyze multi-source data [5, 20, 26, 28] which has gained great attention in biomedical research (see, for instance, [9]). So now, researchers are combining multiple assays (e.g., genome, transcriptome, proteome, epigenome, metabolome...) from the same set of samples in order to create what is known as *multi-omics data sets*.

Nevertheless, on many occasions the information that one can collect is not complete, since for some assays not all data can be gathered (for some observations some data is not available, that is, there is some information missing from some sources). This is what is known as *block-wise missing data*. Indeed, there has been a growing interest in both data mining and machine learning community, not only for omics data but for general data, to fill the gaps of the missing blocks or, at least, to extract as much as possible the necessary information from the unknown data (see [24, 25, 28]). Now, for the former (filling gaps with imputed information) there exist some well-known missing value estimation techniques like *Expectation-Maximization* (EM) [6], *iterative singular value decomposition* (SVD) and *matrix completion* [12], which perform imputation on the missing part of the data. However, those approaches fail to capture the patterns of the missing data and have to estimate a significant amount of missing values with high-dimensional data, which can lead to unstable performance [28]. Otherwise, one could also apply existing feature learning approaches directly, as discarding all samples that have missing entries, but this can lead to an important lost of information.

This thesis is focused on the challenge about how to effectively integrate information from multiple heterogeneous sources in the presence of block-wise missing data, which is going to be restricted to an optimization problem (see [24, 25]). Then, the main problem that is addressed on this thesis is to implement an integrative process with block-wise missing data based on a Lasso's type approximation [18], which will be applied to either simulated data and real data, so that both will be used for the model evaluation.

1.2 Overview

The main aim of this thesis has been to understand the algorithms proposed on [24] and [25] respectively, which define an integrative process with block-wise missing data based on a Lasso's type approximation that result on some regression models, and to generate a code that implement them so that we can computationally evaluate both its performance and its effectiveness by using simulated data or high-dimensional omics data.

Indeed, a unified *bi-level learning model* has been proposed, which consists on a "bi-level analysis" (which performs simultaneously feature-level and source-level analysis) for multi-source incomplete data, a method that avoids direct imputation of the missing elements. The term *bi-level analysis* was first coined in [4] and, although it has recently drawn increasing attention (see, for instance, [23]) how to extend existing techniques to deal with block-wise missing data remains largely unexplored. Indeed, bi-level learning models provide better performances than usual imputations methods, since the former try to extract complementary information from the data.

This thesis has been developed almost entirely through the use of the R programming language and both R Markdown reports and LaTeX typesetting system. The R language and its development framework has been used to generate the scripts that fulfill the functions of: data download, data simulation, study and treatment of data, training and testing of the bi-level learning model, and generation of packages with functions that works with block-wise missing data.

1.3 State-of-the-art

The novelty on this thesis relies on that although one can find the formulation and the theoretical optimization of the problem, we have not been able to find its code implementation anywhere, so it has been impossible for us (until we have succeed implementing them) to give a reasonable evaluation of the proposed algorithms. Indeed, a model that contemplates either complete or block-wise missing data is still new with no so much references of it (if one does not take into account techniques such as the imputation where part of the information on the data is lost).

1.4 Objectives

In this section we present the general objectives of this thesis, which we have broken down into other more concrete:

(i) Development of a code that implements integrative learning for heterogeneous block-wise missing data:

  a) Read and understand the algorithms proposed on [24] and [25], respectively.

  b) Generate a code that implements an optimization algorithm that models an integrative learning model on block-wise missing (or even complete) data.

(ii) Evaluation of the performance and the effectiveness of the previous code with high-dimensional data, either simulated and real data:

  a) Treat the data that will be used for the evaluation of the code. That is (if necessary) to do data quality control by seeing how the data is distributed using graphs and also to do data normalization.

  b) Generate random and simulated block-wise missing data.

  c) Evaluate the model performance and effectiveness. To do so, it will be made use of evaluation measures such as R square/adjusted R square, mean square error(MSE)/root mean square error(RMSE) or even mean absolute error(MAE)/root mean absolute error(RMAE).

(iii) Improvement of the previous code or finding some variants of it:

  a) Try to improve the performance and effectiveness of the model by changing the parameters used on it or modifying conveniently the data used for it.

  b) Investigate possible variants of the model either by using different models or different approaches (recall that the main code will result on a regression model).

1.5 Approach and method

The approach and methodology to be followed will be of a scientific type, since we are in front of a computational optimization (mathematical) regression problem that will be tackled from a high-dimensional data analysis point of view.

Hence, an approach to the problem to be investigated and how to approach it will be made. Furthermore, theoretical support will be sought through the search for related and interesting studies (references), data will be experimented with in order to find significant results for the study and finally some conclusions will be obtained and provided due to the evaluation of the experiment.

Within this type of methodology, in this thesis a quantitative type will be proposed, where the data used will be subjected to a rigorous analysis (using numerical methods) and its results are going to be analyzed with statistical techniques. In this way, the results obtained with this type of methodology will be objective.

1.6 Planning

In this section we have broken down the tasks that are carried out during this thesis in order to achieve the objectives set and we have proposed a time plan by means of a Gantt chart and by marking milestones.

The main tasks basically correspond to the objectives indicated in Section 1.4. However, within those tasks it is necessary to include others dedicated to the search for references and information, together with the installation and learning of the operation of programming libraries. In addition, the drafting of the *PACs* (plural of the Catalan acronym for continuous assessment test) that make up this thesis must also be taking into account. Both in the tasks related to the objectives and those related to the PACs, an estimation time for their duration has been established.

Therefore, the tasks corresponding to the objectives are defined in this way:

- Development of a code that implements integrative learning for heterogeneous block-wise missing data. (126h)

  T.1 Read and understand the algorithms proposed on [24] and [25], respectively. (36h)

  T.2 Generate a code that implements an optimization algorithm that models an integrative learning model on block-wise missing (or even complete) data. (90h)

- Evaluation of the performance and the effectiveness of the previous code with high-dimensional data, either simulated and real data. (72h)

  T.3 Treat the data that will be used for the evaluation of the code. That is (if necessary) to do data quality control by seeing how the data is distributed using graphs and also to do data normalization. (18h)

  T.4 Generate random and simulated block-wise missing data. (18h)

  T.5 Evaluate the model performance and effectiveness. To do so, it will be made use of evaluation measures such as R square/adjusted R square, mean square error(MSE)/root mean square error(RMSE) or even mean absolute error(MAE). (36h)

- Improving of the previous code or finding some variants of it. (63h)

  T.6 Try to improve the performance and effectiveness of the model by changing the parameters used on it or modifying conveniently the data used for it. (31.5h)

T.7 Investigate possible variants of the model either by using different models or different approaches (recall that the main code will result on a regression model). (31.5h)

Further, the tasks related to carrying out the PACs are defined as follows:

PAC0 TFM proposal. (4.5h)

PAC1 Work's plan. (9h)

PAC2 Work development - phase 1. (13.5h)

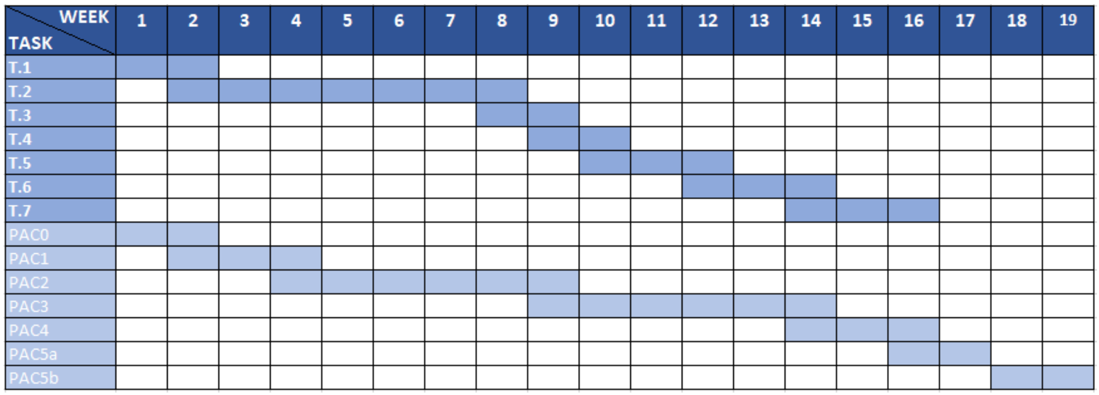
PAC3 Work development - phase 2. (13.5h)

PAC4 Thesis' memory writing. (45h)

PAC5a Preparation of the presentation. (18h)

PAC5b Public thesis defense. (13.5h)

To ease the schedule of the tasks corresponding to the objectives and the PACs, in this section it is showed a calendar (Gantt chart) that follows the notation used above.



WEEK / TASK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
T.1	■	■																	
T.2		■	■	■	■	■	■	■											
T.3								■	■										
T.4									■	■									
T.5										■	■	■							
T.6												■	■	■					
T.7														■	■	■			
PAC0	■	■																	
PAC1		■	■	■															
PAC2				■	■	■	■	■	■										
PAC3									■	■	■	■	■	■					
PAC4														■	■	■			
PAC5a																■	■		
PAC5b																		■	■

The planning shown before has been carried out according to an estimate of the time required. Further, the milestones are set in four date ranges that mark the end of the development of the key objectives set.

- **February 16 - April 12**: Study and development of the main code of this thesis together with the realization of PACs 0, 1 and almost all 2.
- **April 13 - May 9**: Evaluation of the performance and the effectiveness of the code together with the realization of the last of PAC2 and almost all PAC3.
- **May 10 - June 2**: Improving of the previous code or finding some variants of it together with the last of PAC3 and the finalization of the memory's thesis writing (PAC4).
- **June 3 - June 6**: Elaboration of the virtual presentation (PAC5a).

1.7 Brief summary of contributions

The official documents for the UOC consisting on PACs are the following:

- TFM proposal (PAC0), work's plan (PAC1), work development - phase 1 (PAC2), work development - phase 2 (PAC3), thesis' memory writing (PAC4) and TFM presentation (PAC5).

The results from the study are:

- An algorithm that deal with block-wise missing data in order to generate a regression model.
- R scripts (which can be found in the file algorithm_tfm.Rmd) containing all the code of the algorithm together with the simulation of the synthetic data, the reading of the real data and the treatment and summary of both simulated and real data.

1.8 Brief description of each chapter

The manuscript is organized as follows:

In the first chapter we introduce the thesis: in particular, we contextualize and justify the topic to study, we show its importance and what it contributes with the *state-of-the-art*, the main set objectives, the approach and the method followed to obtain the results, the planning that was scheduled before starting with it and a brief summary of the contributions got from it.

In the second chapter, the methodology and materials used are detailed: we highlight the software needed for the correct development of this thesis, we explain the main algorithm which motivates this thesis and consists on a regression model on block-wise missing (or even complete) data, and we summarize the synthetic and the real (exposome) data to be applied to our main algorithm along with the treatment and study of that data.

In the third chapter, the code for the main algorithm can be found. Indeed, there we explain all the mathematics behind the algorithm together its optimization and how to make predictions from the implemented model. For the sake of convenience and clarity of the thesis, throughout this chapter we will combine the mathematical notations and explanations together with its code in R.

In the fourth chapter, we expose the discussion of this thesis together with the applications of the model applied to both simulated and real data. Indeed, we will compare the different scenarios where we will have both complete and block-wise missing data cases by showing all the results obtained from them.

Finally, in the fifth chapter, the conclusions are detailed, along with the future research lines and the schedule tracking.

Chapter 2

Methodology and materials

We devote this chapter to describe the methodology and the materials used along this thesis. In particular, we will talk about the software employed here and we will introduce the model to be studied together with the data (either simulated or real) applied for its proper evaluation.

2.1 Software for the project development

This section explains and justifies the software used on this thesis. Indeed, we will talk about the *R* and *RStudio* software (see Section 2.1.1) and the online *LaTeX* editor called *Overleaf* (see Section 2.1.2).

2.1.1 R and RStudio

Aimed for the analysis of the data, the development of all the code and for its corresponding evaluation on the data, the free software R [16] was used through the RStudio interface [17]. The reason why this software has been chosen is because of the wide variety of statistical models and graphical techniques that they provide. R is an integrated set of software facilities for data manipulation, computation, and graphical display. In addition, it allows users to create extension packages by creating new very useful tools for data analysis. On the other side, the RStudio interface is an integrated development environment for R, which facilitates the use and understanding of the code, in addition to that ease the writing of both the code and its mathematical formulas. Indeed, RStudio presents different areas within the work window where it can be seen data tables, user-defined variables, command console, graph display, and the help tool that prints the manual of the functions integrated in R and in the loaded extension packages.

Within all the extension packages offered by R, we highlight the “glmnet” package [8], which has been used to generate some initial models called β_0 (see Section 3.2.2). Indeed, “glmnet” contains the function *cv.glmnet*, which does k-fold cross-validation to produce a Lasso regression model by setting the parameter *alpha* to 1. All in all, in Appendix A.1.1 can be seen all the packages used for the code of this manuscript.

2.1.2 Overleaf and LaTeX

Overleaf [15] is an open-source online real-time collaborative cloud-based LaTeX editor, while LaTeX [10] is a high-quality typesetting system aimed for the communication and publication of scientific documents. Indeed, Overleaf takes advantage of LaTeX with a multi-panel interface, so that in its left the document can be seen formatted using LaTeX commands (the enriched version) just as it is seen in any domestic text editor and, to its right, it is shown how we will see the document once compiled.

For the writing of this thesis it has been used Overleaf since it makes the whole process of writing, editing and publishing scientific documents, in an structured way, much quicker and easier due to its great variety of packages and environments. Indeed, it allows to write R code together with any kind of mathematical formulas, allowing to obtain a self-contained manuscript. Further, since it integrates LaTeX typesetting, which is in continuous development, it has lots of new functionalities each year and many online resources that can be consulted easily. Besides, LaTeX uses BibTeX as a bibliographic tool to help to organize the user's references and to create a bibliography and, nowadays, almost any book or article citation can be found in that format.

2.2 A unified feature learning model for complete and block-wise missing multi-source data

Given a collection X of n *samples* from S data *sources*; that is,

$$X = [X_1, \dots, X_S] \in \mathbb{R}^{n\times p}, \qquad y \in \mathbb{R}^n,$$

where $X_i \in \mathbb{R}^{n\times p_i}$ is the *data matrix* of the i-th source (which may or not contain missing data) with $p_i \geq 2$ variables (so that $p = p_1 + \cdots + p_S$) and y is the corresponding *outcome* for each sample. We consider the following *linear regression model*:

$$y = \sum_{i=1}^{S} X_i\beta_i + \varepsilon = X\beta + \varepsilon, \tag{2.1}$$

where ε represents the *noise term* and β is the *underlying true model* which is usually unknown in real-world applications. Based on (X, y), we want to use an statistical method called *supervised learning* to learn an estimator of β, denoted as $\hat{\beta}$, whose non-zero elements correspond to the relevant features (in other words, features that correspond to the zero elements of $\hat{\beta}$ are discarded). To do so, in essence, we will consider both the regularization framework

$$\min_{\beta\in\mathbb{R}^p} \mathcal{L}(\beta) + \lambda\Omega(\beta), \qquad \text{for some } \lambda > 0, \tag{2.2}$$

and its constrained form

$$\min_{\beta\in\mathbb{R}^p} \mathcal{L}(\beta) \qquad \text{such that} \qquad \Omega(\beta) \leq \lambda, \qquad \text{for some } \lambda > 0, \tag{2.3}$$

where $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$ is a convex differentiable function with Lipschitz-continuous gradient[1] called *data-fitting term* and $\Omega : \mathbb{R}^p \to \mathbb{R}$ is a sparsity-inducing[2] (typically non-differentiable) norm called the *regularization term*, which encodes our prior knowledge about β. The choice of Ω would enable us to perform a *bi-level analysis*; that is, performing simultaneously both feature-level and source-level analysis. Towards this end, a natural approach is a two-stage model: first we learn different models for each data source and then we combine these learned models properly, where the regularization/constrain should be imposed independently on each stage to assure the bi-level analysis.

2.2.1 Missing blocks and profiles

In most of the cases, the data to be modeled is not complete for every data source but lack one or more data blocks. To apply existing feature learning approaches directly, we can either discard all samples that have missing entries or estimate the missing values based on the observed entries. However, the former approach may significantly reduce the size of the data set while the latter approach heavily relies on our prior knowledge about the missing values. Moreover, both approaches neglect the block-wise missing patterns in the data and therefore could lead to sub-optimal performances. When willing to use the maximum information of the known data, one way is to partition the whole data set into multiple groups according to the availability of data sources.

Given S data sources and assuming that each participant has at least one data source available, then there are $2^S - 1$ possible missing patterns, since

$$\binom{S}{1} + \binom{S}{2} + \cdots + \binom{S}{S-1} + \binom{S}{S} = (1+1)^S - \binom{S}{0} = 2^S - 1.$$

Now, for each participant, based on whether a certain data source is present, we can obtain a binary indicator vector that will simplify the analysis and which is defined as

$$I[1,\ldots,S] = [I(1),\ldots,I(S)] \qquad \text{where} \qquad I(i) = \begin{cases} 1, & i\text{-th data source is available,} \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, it is not needed to store the complete vector for each participant but just to record a single decimal integer (if it is converted this binary vector to a binary number) i.e., the information in the indicator vector can be completely described by a decimal integer, which is called *profile*. All these profiles will be stored in a vector pf of dimension n, where n is the number of samples (see Appendix A.2.1).

Once the availability of data sources is known (due to the profile vector) we can break down the whole data on complete data blocks so that we can extract the maximum information of

[1]That is, there exists a constant $K_{\mathcal{L}}$ such that

$$\|\nabla\mathcal{L}(\beta_1) - \nabla\mathcal{L}(\beta_2)\|_2 \leq K_{\mathcal{L}}\|\beta_1 - \beta_2\|_2, \qquad \forall \beta_1, \beta_2 \in \mathbb{R}^p,$$

with $\|\cdot\|_2$ being the *euclidean norm*, i.e., $\|x\|_2 = \left(x_1^2 + \cdots + x_p^2\right)^{\frac{1}{2}}$ for every $x = (x_1, \ldots, x_p) \in \mathbb{R}^p$.

[2]That is, inducing β to have only a small number of coefficients that are non-zero.

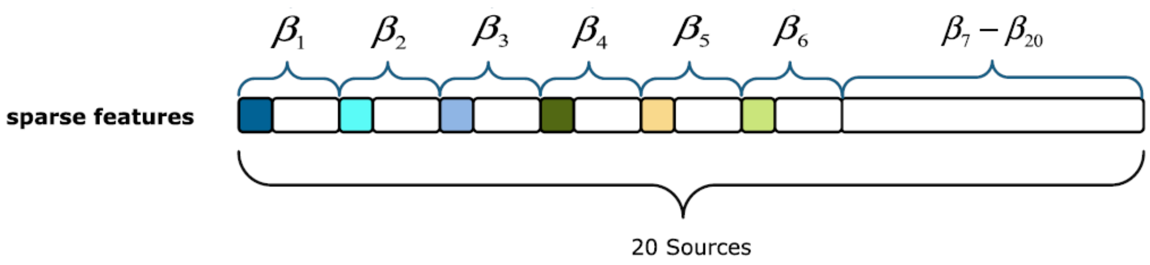


Figure 2.1: Illustration of sparse features [25]. The white blocks represent zero elements, while the non-zero values are represented by different colors.

the known data as highlighted in red boxes on Figure 3.1. To do so, for a given profile m, we will group all the samples which have m as a profile together with those that have complete data in all the sources that are contained in the profile m, i.e., in all the profiles that contains also m as a profile (see Appendix A.2.1).

2.3 Data

This section explains the data used on the study, the variables in which the information of interest is contained and its origin. The data will be used to evaluate the model on Chapter 3 and it will consist on simulated and real data respectively. Indeed, for each data set we will have either complete and block-wise missing data so that we will be able to compare both cases.

2.3.1 Simulated data

The synthetic data that will be used on the analysis is generated by the linear regression model (2.1) and its code can be found in Appendix A.3.1. The parameter setting will follow the similar strategy described in [25]. In particular, it is chosen $n = 1500$ samples and $S = 20$ sources in total, and the underlying true model is

$$\beta = \left[\beta_1^T, \dots, \beta_S^T\right]^T = (\beta_{1,1}, \dots, \beta_{1,p_1}, \dots, \beta_{S,1}, \dots, \beta_{S,p_S})$$

being some of them sparse and with only taking non-zero values in the first six sources (that is, $\beta_i = 0$ for $i \geq 6$) whose values are $\pm 10, \pm 8, \pm 6, \pm 4, \pm 2$ and ± 1 respectively, where the sign of each of its coordinates is chosen randomly (see Figure 2.1).

Further, $\varepsilon \sim N(0, 0.5)$ (that is, it follows the multivariate Gaussian distribution with zero mean and standard deviation of $\sigma = 0.5$). And the same holds for the data matrix $X = [X_1, \dots, X_S]$, where we have simulated three different data matrices according on how correlated the variables are (non-correlated, low-correlated and high-correlated) between them. Besides, we also have imposed missing blocks for those simulated data. We should emphasize here that this distinction on the correlation is aimed to quantify the disagreement of the model (2.1) once we impose each data matrix to have some missing data, i.e., how much affect the quantity of missing blocks as a function of how correlated the data is.

Finally, the outcome y can be computed from (2.1) for each matrix data X by combining the previous parameters in a suitable way.

2.3.2 Exposome data

The real data that will be used on the analysis are drawn from a new collaborative research project called the *Human Early-Life Exposome* (HELIX). In fact, HELIX aims to characterize early-life exposure to multiple environmental factors (early-life exposome) and associate these with omics biomarkers and child health outcomes (see [11, 22] for more information about this topic). The project HELIX used a multilevel study design where the entire study population sums up to 31,472 pairs of mothers and childs, that were recruited during the pregnancy period, distributed in six different cohorts (BiB, MoBa, KANC, EDEN, INMA and RHEA). Further, a subcohort of 1301 pairs of mothers and childs where biomarkers, and child health outcomes were measured at ages ranging between 6 and 11 years.

In that project, there are two available main data sets of *exposome data* (which measures all the exposures of some individuals in a lifetime and how those exposures are related to health) whose variables, to facilitate the analysis, were transformed to approach a normal distribution. One of the data sets is a complete case data (distributed on *exposome* and *covariates* data sets) and the other includes missing data (distributed on *exposomeNA* and *covariatesNA* data sets), both with $n = 1301$ samples. Further, together with both data sets there is an object called *codebook* with all their more important information. Indeed, we see there that, in particular, those data sets have 235 different variables in total from 19 sources (or *families*) classified in five domains, namely *Indoor air*, *Outdoor exposures*, *Covariates*, *Exposure to chemicals* and *Lifestyles*.

Indoor air (BTEX, NO2, PM)
- *Indoor air* with 5 variables.

Outdoor exposures (GIS)
- *Air pollution* with 16 variables.
- *Built environment* with 24 variables.
- *Meteorological* with 12 variables.
- *Natural Spaces* with 9 variables.
- *Noise* with 3 variables.
- *Traffic* with 5 variables.
- *Water DBPs* with 3 variables.

Covariates (potential confounders)
- *Child covariates* with 7 variables.
- *Maternal covariates* with 6 variables.

Exposure to chemicals (biomarkers)
- *Metals* with 20 variables.
- *Organochlorines* with 18 variables.
- *Organophosphate pesticides* with 9 variables.
- *PFAS* with 10 variables.
- *Phenols* with 14 variables.
- *Phthalates* with 22 variables.
- *PBDE* with 4 variables.
- *Tobacco smoke* with 5 variables.

Lifestyles (questionnaire)
- *Lifestyle (Allergens, Diet, Physical activity, Prenatal alcohol, Sleep)* with 39 variables.
- *Social and economic capital* with 4 variables.

Those variables are available at two time points (pregnancy and childhood) except from the covariates, which are available at a single time point (either pregnancy or childhood).

Finally, on both data sets there are variables inside the family *phenotype*, which consists on the health outcome data:

Phenotype (Outcomes)
• *Asthma* (ever) at childhood, 6-11 years (categorical variable).
• *Birth weight* (kg) at birth time (numeric variable).
• *Body mass index* (categories) at childhood, 6-11 years (categorical variable).
• *Body mass index* (z-score) at childhood, 6-11 years (numeric variable).
• *Intelligence quotient - Total correct answers* (RAVEN test) at childhood, 6-11 years (numeric variable).
• *Neuro behaviour - Internalizing and externalizing problems* (CBCL scale) at childhood, 6-11 years (numeric variable).

Now, both data sets (ordered by each source) together with the outcome variables can be declared in R as we did in Appendix A.3.2. There, we observe that all the missing values of the *exposomeNA* data set can be found on the *Covariates* variables, which means that the only missing block that the samples could have correspond to the source *Covariates*. The distribution of the missing values is shown in Figure A.1.

Further, in Appendix A.3.2 it is also made a first brief description of the exposome variables consisting on the smallest data value, the first quantile, the median, the third quantile, and the largest data value of each variable respectively, and we observe that not all variables ranges between the same values, so that it could be a good idea to normalize them. However, since we are in front of a regression problem, and we are aimed to get some predictions, we will let the normalization step as part of the regression algorithm (see Section 3.2.3) so we can keep the values used for such normalization (scaling and translation) for future values oblivious to the current data. Further, we also see that there are both numeric and categorical variables and, indeed, using the object *codebook* (from the exposome data) we are able to see that around the 25.11% are categorical and 74.89% are numeric.

Nevertheless, when dealing with regression problems is advised to work only with numeric variables. That's why we will consider two cases for the previous exposome data sets: one without factors (just numeric variables) and another with the factor variables imposed to be binary and then converted to dummy variables.

- Exposome data without factor variables (numeric variables)

  In this case, we remove from the data (both complete and with missing blocks) the variables that are factors. However, since we need each source having more than two variables, and due to the factor variable removal we obtain sources with just one variable, we add this "only variables" to its more near sources in the sense of those that have closer attributes (see Appendix A.3.2). Indeed, those sources that result to have just one variable are *Noise*, *Social and economic capital* and *Tobacco Smoke*, which are added to the sources *Traffic* (for the former) and *Lifestyle* (for the others).

At this point, before going into details of the "dummy variables" case, taking into account that the cornerstone of the regression problem of Chapter 3 for missing block data consists on getting information for the missing data from the known data, it is important to study how correlated are the numeric variables between them.

First, recall that all the missing values on the *exposomeNA* data set are concentrated on the *Covariates* source; in particular, in Figure A.1 we observe which variables have missing values and with which proportion. However, at the end a sample with some missing value in some variable will mean a sample with missing values in the whole source where this variable belong so the *Covariates* source will be considered as a missing block for all the samples with missing values.

Now, in Figures A.2, A.3, A.4 and A.5, we see the four sources that are more correlated with the *Covariates* source (which are *Air Pollution*, *Metals*, *Organochlorines* and *PFAS*). Besides, we observe that there are some variables that could be able to compensate the missing values of the following *Covariates* variables: *hs_mbmi_None*, *hs_child_age_None*, *hs_c_height_None* and *hs_c_weight_None*, and may be also for *h_age_None*, but it could be more difficult for the variables *hs_wgtgain_None* and *e3_gac_None*.

On the other side, when we study the correlation between the *Covariates*, we obtain that there are highly correlated variables between them (see Figure A.6). In particular, the variable *hs_child_age_None* (child age at postnatal examination in years) is correlated with the variables *h_mbmi_None* (maternal pre-pregnancy body mass index in kg/m2), *hs_c_height_None* (height of the child at 6-11 years old in meters) and *hs_c_weight_None* (weight of the child at 6-11 years old in kg).

Besides, in Figures A.7, A.8, A.9, A.10, A.11 and A.12 we study how correlated are the four sources *Air Pollution*, *Metals*, *Organochlorines* and *PFAS* between them, observing that there exists some correlation, being the *Air Pollution* source the most correlated with the others (than the others between them).

Therefore, in view of the previous results and with the aim of losing the less information possible between variables, it could be interesting on breaking down the source *Covariates* in *subsources* strategically. This subdivision will be applied to both only numeric exposome data and the original exposome data set, where from the latter we will take benefit of it when we create the exposome data set with dummy binary variables (see below). Indeed, we will split the source *Covariates* on the sources *Covariates.Age*, *Covariates.Body.Measures*, *Covariates.Parents.Info* and *Covariates.Childs.Info* (see Appendix A.3.2).

Now, to continue with this study of the numeric variables, let us do a brief study of the *Covariates* variables. For instance, in Figure A.13 it is shown the boxplot of all the variables in order to see how they are distributed and for the search of outliers. There, we observe that the variables are quite centered but with different scales, and also that there is a great presence of outliers (with a total of 142 outliers). For instance, it could be also interesting to study the boxplot of each variable separately according to the binary categorical outcome variable *Asthma* in order to see if there exist differences between each class. Indeed, we observe that the majority of the outliers are concentrated on the samples with no asthma and that the variables with more differences between classes are *h_age_None*, *hs_child_age_None*, *h_mbmi_None* and *hs_c_height_None* (see Figures A.14, A.15, A.16, A.17, A.18, A.19 and A.20).

Moreover, we observe that when doing a principal component analysis we need at least five dimensions in order to have a number of principal components that explain more than the

80% of the total variation of the *Covariates* variables, and the biplot of the two first principal components show that, as expected, we can not say a lot about the two classes from them. Besides, from the biplot we also see that the variables *hs_c_weight_None*, *hs_c_height_None* and *hs_child_age_None* are much closer between them than the others, and the same happens between *e3_gac_None* and *hs_wgtgain_None* (see Figure A.21).

- Exposome data with factor variables converted to dummy binary variables

  In this case, we will first impose all factor variables to be binary and then we will convert them to dummy variables using the original exposome data once the source subdivision has been applied. In fact, for any non-binary factor, if there exists a "ruling" class in the sense that there is one class with much more samples than the others, we will classify that variable between being inside this class and not being inside it; while if all the classes are equitable, we will break it exactly on its half (see Appendix A.3.2).

Chapter 3

An incomplete source feature selection (iSFS) model

Based on [24, 25], this chapter is aimed to present the main ingredients needed to solve an optimization algorithm consisting on a unified feature learning model for heterogeneous block-wise missing (or even complete) data that performs both feature-level and source-level analysis simultaneously. Indeed, the model to be solved is the following:

$$\min_{\alpha,\beta} \frac{1}{|pf|} \sum_{m \in pf} \frac{1}{n_m} \varphi\left(\sum_{i=1}^{S} \alpha_m^i X_m^i \beta^i, y_m \right) + \lambda \Omega_2(\beta) \quad \text{subject to} \quad \Omega_1(\alpha_m) \leq 1 \quad \forall m \in pf, \tag{3.1}$$

where the subscript m denotes the matrix (or vector) restricted to the samples that contain m in their profiles and n_m is the number of rows of X_m, while the superscript i represents the data matrix (or vector) of the i-th source. For instance, here φ can be any convex loss function such as the least squares loss function or the logistic loss function.

To solve (3.1) we will first initialize β by learning an individual model on each data source and compute the optimal α via solving a constrained Lasso problem (see Section 3.2.1). Then β will be updated based on the computed α and next we will compute a new α based on the updated β via solving a regularized Lasso problem (see Section 3.2.2) and we will keep this iterative procedure until convergence of the objective function in (3.1).

At the end, in essence, we will have to deal with the regularization framework on (2.2) and its constrained form (2.3), which can be solved via gradient iteration methods.

3.1 Gradient iteration methods

On this section we present two gradient iteration methods that are aimed to solve the regularization framework (2.2) (see Section 3.1.1) and its constrained form (2.3) (see Section 3.1.2) respectively.

3.1.1 Proximal gradient iteration method

A *proximal gradient iteration method* is a forward-backward splitting method specifically tailored to optimize an objective of the form (2.2) and can be described as follows [3, 14]: at each iteration $t = 1, 2, 3, \dots$ the function $\mathcal{L}$ is linearized around the current point β^t (using its Taylor expansion) and a problem of the form

$$\min_{\beta \in \mathbb{R}^p} \mathcal{L}(\beta^t) + \nabla \mathcal{L}(\beta^t)^T (\beta - \beta^t) + \frac{L}{2} \|\beta - \beta^t\|_2^2 + \lambda \Omega(\beta) \tag{3.2}$$

is solved. In (3.2), the quadratic term (i.e. the *error term*) called *proximal term*, keeps the update in a neighborhood of the current iterate β^t where $\mathcal{L}$ is close to its linear approximation, and $L > 0$ is a parameter which should essentially be an upper bound on the Lipschitz constant of $\nabla \mathcal{L}$. Besides, by means of the inner product induced by the norm $\|\cdot\|_2$, (3.2) can be rewritten as

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \left\| \beta - \left(\beta^t - \frac{1}{L} \nabla \mathcal{L}(\beta^t) \right) \right\|_2^2 + \frac{\lambda}{L} \Omega(\beta). \tag{3.3}$$

Then, a basic proximal gradient iteration method uses the solution of problem (3.3) as the next update β^{t+1}; however, in order to find such a solution is important to compute previously a suitable value for L. Often, an upper bound on the Lipschitz constant of $\nabla \mathcal{L}$ is not known, and even if it is, it is often better to obtain a local estimate. For instance, a suitable value for L can be obtained by iteratively increasing L by a constant factor until the condition

$$\mathcal{L}(\beta_L^*) \leq \mathcal{L}(\beta^t) + \nabla \mathcal{L}(\beta^t)^T (\beta_L^* - \beta^t) + \frac{L}{2} \|\beta_L^* - \beta^t\|_2^2 \tag{3.4}$$

is met (see [1]) where β_L^* denotes the solution of (3.3).

3.1.1.1 Proximal operator

The *proximal operator*, which is denoted by $\text{Prox}_{\mu\Omega}$, was defined in [13] as the function that maps a vector $u \in \mathbb{R}^p$ to the unique solution (since $\frac{1}{2}\|\cdot\|$ is strongly convex) of

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \|u - \beta\|_2^2 + \mu \Omega(\beta).$$

This operator is clearly central to proximal gradient iteration methods due to their main step consists on computing

$$\beta^{t+1} := \text{Prox}_{\mu\Omega}(u) = \text{Prox}_{\frac{\lambda}{L}\Omega} \left(\beta^t - \frac{1}{L} \nabla \mathcal{L}(\beta^t) \right), \tag{3.5}$$

since (3.5) results on being the solution of (3.3). We will dedicate the following to compute the proximal operator for several function norms Ω that induce sparse solutions (see, for instance, [1, Ch. 3.3]):

- **ℓ_1-norm regularization (Lasso Regression [19])**

Let

$$\Omega(\beta) = \|\beta\|_1 := \sum_{j=1}^{p} |\beta_j|, \qquad \text{for } \beta = (\beta_1, \dots, \beta_p) \in \mathbb{R}^p.$$

Then, its proximal operator $\text{Prox}_{\mu\|\cdot\|_1}$ can be computed, separately in each component, as

$$\left(\text{Prox}_{\mu\|\cdot\|_1}(u)\right)_j = \text{sign}(u_j)\left(|u_j| - \mu\right)_+ = \text{sign}(u_j)\max\left(|u_j| - \mu, 0\right), \qquad \forall j = 1, \dots, p,$$

where

$$\text{sign}(x) = \begin{cases} \dfrac{x}{|x|}, & x \neq 0, \\ 0, & x = 0. \end{cases}$$

```
# Proximal operator of l1 norm
prox.operator.l1 <- function(u, mu){
  len_u <- length(u)

  # Optimal solution beta
  beta <- numeric(length = len_u)

  # Since the problem is separable, we compute
  # the optimal solution for each component
  for(j in 1:len_u)
    beta[j] <- sign(u[j])*max(abs(u[j]) - mu, 0)

  return(beta)
}
```

- **ℓ_2^2-norm regularization (Ridge Regression)**

Let

$$\Omega(\beta) = \frac{1}{2}\|\beta\|_2^2 := \frac{1}{2}\sum_{j=1}^{p} |\beta_j|^2, \qquad \text{for } \beta = (\beta_1, \dots, \beta_p) \in \mathbb{R}^p.$$

Although this regularization function does not induce sparsity, it is nonetheless widely used and it is worth mentioning its proximal operator $\text{Prox}_{\frac{\mu}{2}\|\cdot\|_2^2}$, which can be computed as

$$\text{Prox}_{\frac{\mu}{2}\|\cdot\|_2^2}(u) = \frac{1}{1+\mu}u.$$

```
# Proximal operator of l2^2 norm
prox.operator.l2 <- function(u, mu){
  # Optimal solution beta
  return(u/(1 + mu))
}
```

- **$\ell_1 + \ell_2^2$-norm regularization (Elastic-net [30])**

Let

$$\Omega(\beta) = \|\beta\|_1 + \frac{\gamma}{2}\|\beta\|_2^2, \qquad \text{for } \beta = (\beta_1, \dots, \beta_p) \in \mathbb{R}^p \text{ and } \gamma > 0.$$

Then, its proximal operator $\text{Prox}_{\|\cdot\|_1 + \frac{\gamma}{2}\|\cdot\|_2^2}$ can be computed as

$$\text{Prox}_{\mu\left(\|\cdot\|_1 + \frac{\gamma}{2}\|\cdot\|_2^2\right)}(u) = \frac{1}{1+\mu\gamma}\text{Prox}_{\mu\|\cdot\|_1}(u).$$

```
# Proximal operator of l1 + l2^2 norm
prox.operator.l1.l2 <- function(u, mu, gamma){
  # Optimal solution beta
  return(prox.operator.l2(prox.operator.l1(u, mu), mu*gamma))
}
```

- **ℓ_1/ℓ_2-norm regularization (Group Lasso [29])**

For S different groups, let

$$\Omega(\beta) := \sum_{i=1}^{S} \sqrt{p_i}\|\beta_i\|_2, \qquad \text{for } \beta = (\beta_1, \dots, \beta_S) \text{ with } \beta_i \in \mathbb{R}^{p_i}.$$

Then, its proximal operator $\text{Prox}_{\mu\Omega}$ can be computed, separately in each i-th group, as

$$\left(\text{Prox}_{\mu\Omega}(u)\right)_i = \left(1 - \frac{\sqrt{p_i}\mu}{\|u_i\|_2}\right)_+ u_i = \max\left(1 - \frac{\sqrt{p_i}\mu}{\|u_i\|_2}, 0\right) u_i, \qquad \text{for } i = 1, \dots, S.$$

```
# Proximal operator of l1/l2 norm
prox.operator.l1_l2 <- function(p, u, mu){
  if(length(u) != sum(p))
    return(u)

  # Optimal solution beta
  beta <- numeric(length = length(u))

  # Partition range
  group.init <- 1
  for(i in 1:length(p)){
    group.end <- group.init + p[i]
    group.range <- group.init:(group.end - 1)

    # Since the problem is separable, we compute the optimal
    # solution for each group
    l2.norm.u_group <- (sum(u[group.range]^2))^(1/2)
    beta[group.range] <- max((1 - sqrt(p[i])*mu/l2.norm.u_group), 0)*
                          u[group.range]

    group.init <- group.end
  }

  return(beta)
}
```

3.1.1.2 Algorithm

Here, we code (3.5) for different forms of Ω by assuming that the gradient value is known:

```
# Proximal gradient method
prox.grad.method <- function(beta, lambda, L, gradient, omega,
                             p, gamma){
  # Vector hat beta and mu
  u <- beta - gradient/L
  mu <- lambda/L

  switch (
    omega,

    # Omega being l1 norm
    "LR" = return(prox.operator.l1(u, mu)),

    # Omega being l2 norm
    "RR" = return(prox.operator.l2(u, mu)),

    # Omega being l1 + l2^2 norm
    "EN" = return(prox.operator.l1.l2(u, mu, gamma)),

    # Omega being l1/l2 norm
    "GL" = if(!is.null(p)) return(prox.operator.l1_l2(p, u, mu))
           else return(u)
  )

  return(u)
}
```

3.1.2 Norm projection iteration method

A *norm projection iteration method* is a forward-backward splitting method aimed to solve an objective of the form (2.3) whenever Ω is a norm. In particular, similar as in (3.3), the problem (2.3) reduces to the projection onto the Ω-ball

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\left\|\beta-\left(\beta^t-\frac{1}{L}\nabla\mathcal{L}(\beta^t)\right)\right\|_2^2 \qquad \text{subject to} \qquad \Omega(\beta)\le\lambda, \tag{3.6}$$

and, therefore, the problem that we have to confront is: given $\hat{\beta}\in\mathbb{R}^p$, compute

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\left\|\beta-\hat{\beta}\right\|_2^2 \qquad \text{subject to} \qquad \Omega(\beta)\le\lambda. \tag{3.7}$$

Now, in (3.7), ignoring the case $\Omega(\hat{\beta})\le\lambda$ (which has the trivial solution $\beta=\hat{\beta}$) there exists for each $\lambda>0$ a $\mu=\mu(\lambda)>0$ satisfying

$$\Omega(\mathrm{Prox}_{\mu\Omega}(\hat{\beta}))=\lambda \tag{3.8}$$

such that the optimization problem

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\left\|\beta-\hat{\beta}\right\|_2^2+\mu\Omega(\beta) \tag{3.9}$$

has the same solution as (3.7). Indeed, we have already seen in Section 3.1.1.1 that $\mathrm{Prox}_{\mu\Omega}(\hat{\beta})$ is a solution of (3.9). Hence, if we denote $\beta^*=\mathrm{Prox}_{\mu\Omega}(\hat{\beta})$, then

$$\frac{1}{2}\left\|\beta-\hat{\beta}\right\|_2^2+\mu\Omega(\beta)\ge\frac{1}{2}\left\|\beta^*-\hat{\beta}\right\|_2^2+\mu\Omega(\beta^*), \qquad \forall\beta\in\mathbb{R}^p,$$

and since we are assuming that $\Omega(\beta^*)=\lambda$,

$$\frac{1}{2}\left\|\beta-\hat{\beta}\right\|_2^2\ge\frac{1}{2}\left\|\beta^*-\hat{\beta}\right\|_2^2+\mu(\Omega(\beta^*)-\Omega(\beta))\ge\frac{1}{2}\left\|\beta^*-\hat{\beta}\right\|_2^2 \quad \text{subject to} \quad \Omega(\beta)\le\lambda,$$

so that β^* is also a solution of (3.7).

Thus, the cornerstone on solving (2.3) consists on finding a μ satisfying (3.8) and then computing

$$\beta^{t+1}=\mathrm{Prox}_{\mu\Omega}\left(\beta^t-\frac{1}{L}\nabla\mathcal{L}(\beta^t)\right) \qquad \text{whenever} \qquad \Omega\left(\beta^t-\frac{1}{L}\nabla\mathcal{L}(\beta^t)\right)>\lambda.$$

The remainder of this section is devoted to developing a method for finding such μ for different forms of Ω that induce sparse solutions (see, for instance, [21]):

- **ℓ_1-norm projection (Lasso penalty)**

Let $\Omega = \|\cdot\|_1$, so we have to find μ such that

$$\varphi(\mu) := \left\| S_\mu(\hat{\beta}) \right\|_1 = \lambda \qquad \text{with (componentwise)} \qquad S_\mu(\beta) = \operatorname{sign}(\beta)\max(|\beta| - \mu, 0),$$

where we are assuming that $\|\hat{\beta}\|_1 > \lambda$.

Let b_i, $i = 1, \ldots, p$, be the absolute values of $\hat{\beta}$ in decreasing order, and define $b_{p+1} = 0$. It is an easy computation to show that then there exists some $k \in \{1, \ldots, p\}$ such that

$$\varphi(b_k) \leq \lambda < \varphi(b_{k+1}).$$

Hence, suppose that k is given. So, it is only need to find some $0 \leq \delta < b_k - b_{k+1}$ such that

$$\lambda = \varphi(b_k - \delta) = \sum_{i=1}^{p} \max(b_i - b_k + \delta, 0) = \sum_{i=1}^{k-1}(b_i - b_k) + k\delta = \varphi(b_k) + k\delta;$$

that is,

$$\delta := \frac{\lambda - \varphi(b_k)}{k} = \frac{\lambda - \left\| S_{b_k}(\hat{\beta}) \right\|_1}{k},$$

and hence $\mu = b_k - \delta$.

```r
# Computation of the parameter mu with l1-norm
mu_computation.l1 <- function(beta, lambda){
  # Define vector b
  b <- c(abs(beta), 0)
  b <- b[order(b, decreasing = TRUE)]

  # Seeking for the index k
  k <- 2
  S.bk <- sum(abs(prox.operator.l1(beta, b[k])))
  # Do the loop until the index k is found
  while(lambda > S.bk){
    k <- k + 1
    S.bk <- sum(abs(prox.operator.l1(beta, b[k])))
  }

  k <- k - 1
  S.bk <- sum(abs(prox.operator.l1(beta, b[k])))
  return(b[k] - (lambda - S.bk)/k)
}
```

- **ℓ_2^2-norm projection (ridge penalty)**

Let $\Omega = \frac{1}{2}\|\cdot\|_2^2$, so we have to find μ such that

$$\frac{1}{2}\left\|\text{Prox}_{\frac{\mu}{2}\|\cdot\|_2^2}(\hat{\beta})\right\|_2^2 = \frac{1}{2}\left\|\frac{1}{1+\mu}\hat{\beta}\right\|_2^2 = \lambda \qquad \Longleftrightarrow \qquad \mu = \frac{1}{\sqrt{2\lambda}}\|\hat{\beta}\|_2 - 1,$$

where we are assuming that $\frac{1}{2}\|\hat{\beta}\|_2^2 > \lambda$.

```r
# Computation of the parameter mu with l2^2-norm
mu_computation.l2 <- function(beta, lambda){
  return(sqrt(sum(beta^2)/(2*lambda)) - 1)
}
```

3.1.2.1 Algorithm

Here, we code (3.6) for different forms of Ω by assuming that the gradient value is known:

```r
# Norm projection method
norm.proj.method <- function(beta, lambda, L, gradient, omega,
                             tol = 1e-3){
  # Vector hat beta
  u <- beta - gradient/L

  switch (
    omega,

    # Omega being l1 norm
    "LR" =
    if(sum(abs(u)) > lambda + tol)
      return(prox.operator.l1(u, mu_computation.l1(u, lambda))),

    # Omega being l2^2 norm
    "RR" =
    if(sum(u^2)/2 > lambda + tol)
       return(prox.operator.l2(u, mu_computation.l2(u, lambda))),
  )

  return(u)
}
```

3.1.3 Finding a solution for a suitable value of L

Recall that a suitable value of L can be obtained by iteratively increasing L by a constant factor until the condition in (3.4) is met. Further, since we are considering a gradient iteration method, we should assume that $L \geq L_{\min}$ where the parameter $L_{\min}$ is chosen as the inverse of the two-point approximation to the quasi-Newton secant equations [2]; that is,

$$L \geq L_{\min} := \frac{(\beta^t - \beta^{t-1})^T (\nabla \mathcal{L}(\beta^t) - \nabla \mathcal{L}(\beta^{t-1}))}{(\beta^t - \beta^{t-1})^T (\beta^t - \beta^{t-1})}.$$

```
# Computing L.min
L.min <- function(beta.current, beta.prev, gradient){
  diff.beta <- beta.current - beta.prev
  diff.grad.beta <- gradient(beta.current) - gradient(beta.prev)
  return(as.numeric(diff.beta%*%diff.grad.beta/
                   (diff.beta%*%diff.beta)))
}
```

Finally, the following code compute iteratively the coefficients β_L^* by using the proper method according to the framework to face up to (either regularized (2.2) or constrained (2.3)):

```
beta.suitable.L <- function(beta, lambda, function.L, gradient.L,
                            L.min, omega, optimization, L.step,
                            maxIter, tol, p = NULL, gamma = 1){
  # Compute gradient vector evaluated at beta
  gradient <- gradient.L(beta)

  # Compute objective value evaluated at beta
  objective <- function.L(beta)

  # Choose framework
  method.beta.star <- switch(
    optimization,

    "reg" = function(L){return(prox.grad.method(beta, lambda, L,
                                    gradient, omega, p, gamma))},

    "cons" = function(L){return(norm.proj.method(beta, lambda, L,
                                    gradient, omega))},
    )

  # Compute beta star from L
  L <- L.min
  beta.star <- method.beta.star(L)

  # Linearization of objective
  diff.beta <- beta.star - beta
  linear.L <- as.numeric(objective - function.L(beta.star) +
                            gradient%*%diff.beta + L/2*sum(diff.beta^2))

  iter <- 0
  while(linear.L < tol && iter < maxIter){
    # Compute next beta star from L
    L <- L*L.step
    beta.star <- method.beta.star(L)

    # Linearization of objective
    diff.beta <- beta.star - beta
    linear.L <- as.numeric(objective - function.L(beta.star) +
                            gradient%*%diff.beta + L/2*sum(diff.beta^2))

    iter <- iter + 1
  }

  return(beta.star)
}
```

3.2 iSFS model for the least square loss function

On this section, a solution is given for the model (3.1) by assuming that φ is the least square loss function (that is, $\varphi = \frac{1}{2}\|\cdot\|_2^2$) which could be adapted, with the necessary modifications, to another convex loss function φ. In this case, the objective function can be coded as follows:

```
# Objective function computation
objective.fun <- function(p, X, y, beta, alpha, pf.vec){
  # Number of sources
  S <- length(p)

  # Profiles
  profiles <- levels(pf.vec)

  # Objective function computing
  obj.func <- 0
  for(i in 1:length(profiles)){
    # Profile m
    m <- as.integer(profiles[i])

    # Profile alpha vec
    alpha.m <- alpha[[i]]

    # Block samples for the profile m
    block.samples <- getBlockSamples(pf.vec, m, S)
    X.m <- X[block.samples$samples,]

    # We update the value inside the norm
    col <- 1
    vec.sum <- numeric(length = dim(X.m)[1])
    for(j in 1:S) {
      nextcol <- col + p[j] - 1
      if(j %in% block.samples$sources)
        vec.sum <- vec.sum + alpha.m[j]*X.m[, col:nextcol]%*%
                                        beta[col:nextcol]
      col <- nextcol + 1
    }
    vec.sum <- as.vector(vec.sum) - y[block.samples$samples]

    # We update the value of the objective function
    obj.func <- obj.func + sum(vec.sum^2)/(2*dim(X.m)[1])
  }

  return(obj.func/length(profiles))
}
```

Now, before going further, let us recall that (3.1) consists on learning a consistent model (denoted with a variable β) across different source combinations, while within each combination, some weights for different sources (denoted by the variable α) are computed adaptively.

As an illustration, in Figure 3.1 we have n samples with variables taken in three different data sources and the profile vector (once converted the profiles from binary to natural numbers) is $pf = (4, 7, 3, 2)$ (so that $|pf| = 4$). Hence, the data is divided in four blocks according the availability of complete data on the sources contained on each profile, as highlighted by the red boxes. Therefore, in this particular case, the goal is to learn three models β^1, β^2 and β^3 independently for each data source as well as the weights (vectors of four components) α^1, α^2 and α^3 that combines them. Notice that, for the i-th data source, β^i remains identical while α^i_j may vary across each different group j.

On what follows, we will devote it to see how to compute the models β and the weights α for the model (3.1) when φ is the least square loss function.

3.2.1 Computing α when β is fixed

When β is fixed, the objective function of (3.1) is decoupled with respect to α_m and, for each $m \in pf$, the optimal α_m is given by the optimal solution of the following problem:

$$\min_{\alpha_m} f(\alpha_m) \qquad \text{such that} \qquad \Omega_1(\alpha_m) \le 1, \quad \alpha_m = (\alpha^1_m, \dots, \alpha^S_m) \in \mathbb{R}^S, \tag{3.10}$$

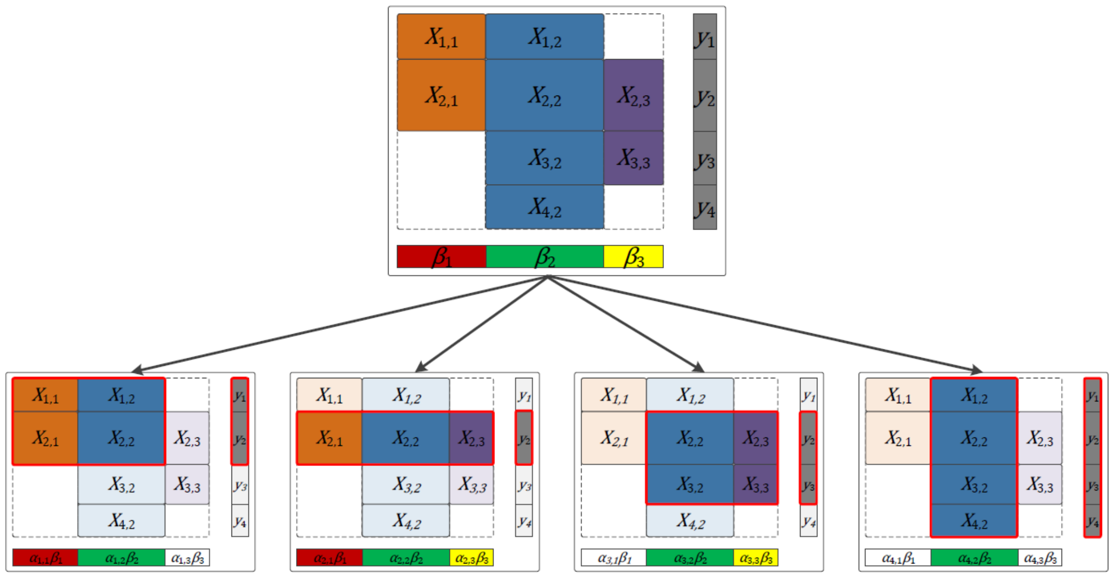


Figure 3.1: Illustration of the proposed learning model (see [25]). Notice that the missing data emerges in a block-wise way, i.e., for a sample, certain data source is either available or lost completely.

where

$$f(\alpha_m) = \frac{1}{2}\left\| \sum_{i=1}^{S} \alpha_m^i \tilde{\beta}_m^i - y_m \right\|_2^2 \qquad \text{with} \qquad \tilde{\beta}_m^i = X_m^i \beta^i \in \mathbb{R}^{n_m \times 1}.$$

```
# Compute function f
f <- function(y.m, alpha.m, tilde.beta){
  # Number of sources
  S <- dim(tilde.beta)[2]

  # Value to compute inside norm
  val <- numeric(length = length(y.m))
  for(j in 1:S)
    val <- val + alpha.m[j]*tilde.beta[,j]
  val <- val - y.m

  return(sum(val^2)/2)
}
```

Further, for each i-th data source, the gradient $\nabla f(\alpha)$ with respect each α^i can be computed as follows:

$$\nabla f(\alpha) = (\partial_1 f(\alpha), \dots, \partial_S f(\alpha)) \qquad \text{with} \qquad \partial_i f(\alpha) = \alpha^i \|\tilde{\beta}^i\|_2^2 - \langle \tilde{\beta}^i, y \rangle,$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product of two vectors.

```
# Compute alpha gradient
gradient.f <- function(y.m, alpha.m, tilde.beta){
  # Number of sources
  S <- dim(tilde.beta)[2]

  # Gradient of f
  gradient.alpha <- numeric(length = S)
  for(i in 1:S)
    gradient.alpha[i] <- alpha.m[i]*sum(tilde.beta[,i]^2) -
                         sum(tilde.beta[,i]%*%y.m)

  return(gradient.alpha)
}
```

And since

$$\|\nabla f(\alpha)-\nabla f(\tilde{\alpha})\|_2^2=\sum_{i=1}^{S}(\alpha^i-\tilde{\alpha}^i)^2\|\tilde{\beta}^i\|_2^4\leq \max\left(\|\tilde{\beta}^1\|_2,\dots,\|\tilde{\beta}^S\|_2\right)^4\|\alpha-\tilde{\alpha}\|_2^2,\quad \forall\alpha,\tilde{\alpha}\in\mathbb{R}^p,$$

we can bound the Lipschitz constant K_f of the function f as follows:

$$K_f\leq \max\left(\|\tilde{\beta}^1\|_2,\dots,\|\tilde{\beta}^S\|_2\right)^2.$$

```
# Lipschitz constant of the function f
const.Lipschitz.alpha <- function(tilde.beta){
  sum.sq <- numeric(length = dim(tilde.beta)[2])
  for(j in 1:dim(tilde.beta)[2])
    sum.sq[j] <- sum(tilde.beta[,j]^2)

  return(max(sum.sq))
}
```

Now, since we want to solve the optimization problem (3.10), we will make use of the Ω_1-norm projection iteration method (see Section 3.1.2) where we will allow Ω_1 to be either the ℓ_1-norm penalty or the ridge penalty. To do so, we first need to initialize some weights α_0:

```
# Initializing alpha0 weights uniformly
alpha.initialization <- function(pf.vec, S, keep.alpha){
  # alpha0 weights
  alpha0 <- list()

  # Profiles
  profiles <- levels(pf.vec)

  # Initialize alpha
  if(keep.alpha){
    # All alpha's set to 1/n
    for(i in 1:length(profiles))
      alpha0[[i]] <- rep(1/length(pf.vec), S)
  } else {
    # All alpha's on profile set to 1/n_m (number of samples
    # of each profile)
    for(i in 1:length(profiles)){
      # Profile m
      m <- as.integer(profiles[i])

      # Get block samples
      block.samples <- getBlockSamples(pf.vec, m, S)

      # Initialize alpha_m with 0's on the sources
      # that are not involved on the profile m
      alpha0.aux <- numeric(length = S)
      alpha0.aux[block.samples$sources]
              <- 1/length(block.samples$samples)
      alpha0[[i]] <- alpha0.aux
    }
  }

  return(alpha0)
}
```

And the Ω_1-norm projection iteration method can be coded as follows:

```
# Omega-norm projection iteration method
omega.norm.proj.method <- function(y.m, alpha0, tilde.beta, omega,
                                   L.step, maxIter, tol){
  # Function f and its gradient depending just on alpha
  func.f <- function(alpha){f(y.m, alpha, tilde.beta)}
  grad.f <- function(alpha){gradient.f(y.m, alpha, tilde.beta)}

  # First L.min value
  Lmin <- const.Lipschitz.alpha(tilde.beta)
  # Next alpha vector
  alpha <- beta.suitable.L(alpha0, 1, func.f, grad.f, 1, omega,
                           "cons", L.step, maxIter, tol)

  # Number of iterations
  iter <- 0
  # Repeat until getting solution or achieving maxIter index
  diff.func.alpha <- abs(func.f(alpha) - func.f(alpha0))
  while(diff.func.alpha > tol && iter < maxIter){
    # Next alpha vector
    alpha0 <- alpha
    alpha <- beta.suitable.L(alpha0, 1, func.f, grad.f, Lmin, omega,
                             "cons", L.step, maxIter, tol)

    # Next difference function value and iteration
    diff.func.alpha <- abs(func.f(alpha) - func.f(alpha0))
    iter <- iter + 1
  }

  return(alpha)
}
```

Finally, the code to compute α when β is fixed is the following:

```
# Computing alpha when beta is fixed
alpha.compute <- function(p, X, y, beta, alpha0, pf.vec, omega,
                          L.step, maxIter, tol){
  # Number of sources
  S <- length(p)

  alpha <- list()
  # For each profile
  for(i in 1:length(levels(pf.vec))){
    # Profile
    m <- as.integer(levels(pf.vec)[i])
    if(m == 0){
      alpha[[i]] <- rep(0, S)
      next
    }

    # Samples with current profile
    block.samples <- getBlockSamples(pf.vec, m, S)
    X.m <- X[block.samples$samples,]
    # Prediction matrix from sample
    tilde.beta <- numeric()
    col <- 1
    for(j in 1:S){
      nextCol <- col + p[j] - 1
      if(j %in% block.samples$sources)
        tilde.beta <- cbind(tilde.beta, X.m[, col:nextCol]%*%
                                            beta[col:nextCol])
      else tilde.beta <- cbind(tilde.beta, rep(0, dim(X.m)[1]))

      col <- nextCol + 1
    }

    # Computing updated alpha
    alpha[[i]] <- omega.norm.proj.method(y[block.samples$samples],
                                         alpha0[[i]], tilde.beta,
                                         omega, L.step, maxIter, tol)
  }

  return(alpha)
}
```

3.2.2 Computing β when α is fixed

When α is fixed, then (3.1) becomes an unconstrained regularization problem; that is,

$$\min_{\beta} g(\beta) + \lambda\Omega_2(\beta), \tag{3.11}$$

where

$$g(\beta) = \frac{1}{|pf|} \sum_{m\in pf} \frac{1}{2n_m} \left\| \sum_{i=1}^{S} (\alpha_m^i X_m^i)\beta^i - y_m \right\|_2^2,$$

which coincide with the objective function in (3.1).

```
# Computing function g given vector beta
g <- function(p, X, y, alpha, beta, pf.vec){
  return(objective.fun(p, X, y, beta, alpha, pf.vec))
}
```

Further, for each i-th data source, the gradient $\nabla g(\beta)$ with respect to β^i can be computed as follows:

$$\nabla g(\beta^i) = \frac{1}{|pf|} \sum_{m\in pf} \frac{1}{n_m} \chi_{\{m\,\&\,2^{S-i}\neq 0\}} \left(\alpha_m^i X_m^i\right)^T \left(\sum_{j=1}^{S} \alpha_m^j X_m^j \beta^j - y_m \right),$$

where $\chi_{\{\cdot\}}$ is the indicator function which has value 1 when the condition is satisfied and 0 otherwise, and $\left\{m \,\&\, 2^{S-i} \neq 0\right\}$ stands for whether the source i is contained (or not) on the profile m. So, the gradient $\nabla g(\beta)$ can be coded as follows:

```
# Computing gradient of function g given vector beta
gradient.g <- function(p, X, y, alpha, beta, pf.vec){
  # Number of sources
  S <- length(p)

  # Profiles
  profiles <- levels(pf.vec)

  # Gradient vector
  grad.vec <- numeric(length = length(beta))
  col.source <- 1
  for(i.source in 1:S){
    # Initialize gradient value
    gradient <- numeric(length = p[i.source])
    next.col.source <- col.source + p[i.source] - 1

    # First value to compute
    for(i in 1:length(profiles)){
      # Profile m
      m <- as.integer(profiles[i])

      # Check if the source is on this profile
      if(!as.binary(m, n = S)[i.source])
        next;

      # Profile m alpha weights
      alpha.m <- alpha[[i]]

      # Samples with current profile
      block.samples <- getBlockSamples(pf.vec, m, S)
      X.m <- X[block.samples$samples,]

      # First value to compute
      val1 <- numeric(length = dim(X.m)[1])
      col <- 1
      for(j in 1:S){
        nextcol <- col + p[j] - 1
        if(j %in% block.samples$sources)
          val1 <- val1 + alpha.m[j]*(X.m[, col:nextcol]%*%
                                       beta[col:nextcol])
        col <- nextcol + 1
      }
      val1 <- val1 - y[block.samples$samples]

      # Second value to compute
      val2 <- t(alpha.m[i.source]*X.m[,col.source:next.col.source])

      # Gradient update
      gradient <- gradient + (val2%*%val1)/dim(X.m)[1]
    }

    grad.vec[col.source:next.col.source] <- gradient
    col.source <- next.col.source + 1
  }

  return(grad.vec/length(profiles))
}
```

Now, since we want to solve the optimization problem (3.11) we will make use of the proximal gradient iteration method (see Section 3.1.1). To do so, we first initialize some models β_0 by learning them for each data source independently and following different methods. Indeed, we will use linear regression and Lasso regression models. The most important thing in Lasso models boils down to select an optimal parameter λ, which will be determined with a process of cross-validation by taking the value of λ that minimizes the mean cross-validation error.

```
# We initialize beta0 by fitting each source individually
# on the available data
beta.initialization <- function(p, X, y, beta0.comp){
  # Number of sources
  S <- length(p)

  # beta0 initialization model
  beta0.compute <- switch (
    beta0.comp,

    # Linear Model Regression
    # We use a robust one for the presence of outliers
    "LMR" = function(X, y){
      return(as.vector(rlm(y ~ . + 0, data =
                                data.frame(X))$coefficients))
    },

    # Lasso Regression
    "LR" = function(X, y){
        # Lasso (alpha = 1, lasso penalty)
        cv_lasso_model <- cv.glmnet(x = as.matrix(X), y = y, family
                                    = "gaussian", alpha = 1, intercept
                                    = F, nfolds = 5)

        # Best lambda value model
        lambda_lasso <- cv_lasso_model$lambda.min
        return(as.vector(glmnet(x = as.matrix(X), y = y, family =
                                "gaussian", alpha = 1, intercept
                                = F, lambda = lambda_lasso)$beta[,1]))
    },

    return(NULL)
  )

  # Beta coefficients
  beta.coeff <- numeric(length = dim(X)[2])
  col <- 1
  for(i in 1:S){
    nextCol <- col + p[i] - 1

    # Samples in source i with complete data
    ind.samp <- rowSums(is.na(X[, col:nextCol])) == 0
    X.complete <- X[ind.samp, col:nextCol]

    # Beta coefficient for source i
    beta.coeff[col:nextCol] <- beta0.compute(X.complete, y[ind.samp])

    col <- nextCol + 1
  }

  return(beta.coeff)
}
```

And finally, once we have the initial models β_0, we are able to compute for each step t the models β^{t+1} as in (3.5), and we will continue iterating until the objective function stops decreasing.

```
# Proximal gradient iteration method
prox.grad.iter.method <- function(p, X, y, alpha, beta0, pf.vec,
                                  lambda, omega, L.step, maxIter,
                                  tol, gamma){

  # Function g and its gradient depending just on beta
  func.g <- function(beta){g(p, X, y, alpha, beta, pf.vec)}
  grad.g <- function(beta){gradient.g(p, X, y, alpha, beta, pf.vec)}

  # Next beta vector
  # We start with L.min = 1
  beta <- beta.suitable.L(beta0, lambda, func.g, grad.g, 1,
                          omega, "reg", L.step, maxIter, tol,
                          p, gamma)
  # Number of iterations
  iter <- 0
  # Repeat until getting solution or achieving maxIter index
  diff.func.beta <- abs(func.g(beta) - func.g(beta0))
  Lmin <- 0
  while(diff.func.beta > tol && iter < maxIter){
    # L.min value
    Lmin.aux <- L.min(beta, beta0, grad.g)
    if(Lmin.aux > Lmin) Lmin <- Lmin.aux

    # Next beta vector
    beta0 <- beta
    beta <- beta.suitable.L(beta0, lambda, func.g, grad.g, Lmin,
                            omega, "reg", L.step, maxIter, tol,
                            p, gamma)

    # Next difference function value and iteration
    diff.func.beta <- abs(func.g(beta) - func.g(beta0))
    iter <- iter + 1
  }

  return(beta)
}
```

3.2.3 Algorithm of the iSFS model for the least square loss function

At this point, we know how to compute both the models β and the weights α, so we are in conditions to write down the proposed alternating algorithm for solving (3.1) with φ being the least square loss function (see Appendix B.1.1). Indeed, Algorithm 3.1 summarizes our iSFS model for block-wise missing data.

Remark 3.2.1 *On Algorithm 3.1, when all the weights α are fixed and equal to $\frac{1}{n}$ (so that its step 6 is missed) then the problem is restricted to a unified learning model for multi-source data (see [24, 25]). That happens, for instance, when the data is complete.*

Further, now we are able to make predictions of the outcome from an iSFS model (see

Algorithm 3.1 iSFS model for the least square loss function

```
1: Input: $X$, $y$, $\lambda$
2: Output: Solutions $\alpha$ and $\beta$ to (3.1) when $\varphi = \frac{1}{2}\|\cdot\|_2^2$

3: Initialize $\alpha_0$ with the function alpha.initialization of Section 3.2.1
4: Initialize $\beta_0$ with the function beta.initialization of Section 3.2.2
5: for $t = 1, 2, \dots$ do
6:     Compute $\alpha^t$ by means of the function alpha.compute of Section 3.2.1
7:     Compute $\beta^t$ by means of the function prox.grad.iter.method of Section 3.2.2
8:     if the objective function on (3.1) stops decreasing then
         return $\alpha = \alpha^t$ and $\beta = \beta^t$
9:     end if
10: end for
```

Appendix B.1.2) so that we can evaluate its performance and effectiveness, which will be done in Chapter 4.

Chapter 4

Discussion and applications of the iSFS model on simulated and exposome data

We dedicate this chapter to examine the efficacy of the proposed bi-level feature learning model by reporting its performance based on both synthetic and exposome data (see Sections 2.3.1 and 2.3.2). First, to do so, we will train the model on training data and we will make predictions on some testing data, for which we will use evaluation measures such as R square/adjusted R square, mean square error(MSE)/root mean square error(RMSE) and mean absolute error(MAE)/root mean absolute error(RMAE) (see Appendix C). Further, we will plot the predicted outcomes obtained together with the real ones.

We should mention here that we will work on different scenarios of the simulated data and the exposome data, respectively. On the former, we will separate the study according on the "grade" of correlation; while on the latter we will work with only numeric data and data where factors has been converted to binary dummy variables, applied to the four numeric outcomes of exposome data, namely *hs_zbmi_who*, *e3_bw*, *hs_correct_raven* and *hs_Gen_Tot*. Finally, we will compare those data with its corresponding block-wise missing case. Indeed, we will try to answer the following questions that araised on Chapter 2:

- How is the performance of the algorithm on Section 3.2.3 with both synthetic and exposome data?
- Which features on both synthetic and exposome complete data set are the most relevant for the model (that is, which features have non-zero values on the estimator $\hat{\beta}$)?
- How does affect the missing data on both synthetic and exposome data sets on the performance of the model?
- How does affect the data correlation on the predictions for the synthetic block-wise missing case?
- Is there any difference between the performance of the model according to the four outcomes of the exposome data?

- Is it better to work with all the numeric variables or with all the variables where the factors have been converted to binary dummy variables (both scenarios of the exposome data)?

Before going into details, we should mention that in all the models we have observed the following: the objective function tends to decrease as we increase the number of iterations on the model. So, putting more iterations for each model (and may decreasing or vanishing the tolerance value) will have as a consequence better performances, but we will pay the price of needing more computing time. Further, we will not discuss the performance of the model in [24, 25] with the model on this manuscript since the data aimed for the study is not the same that the one used there.

4.1 Simulated data

To discuss the evaluation of the iSFS model performance on simulated data, we have separated each data set in training (67%) and testing (33%) as shown in Appendix C.1.

4.1.1 Comparison on complete data

We observe on Tables C.1 and C.2, Tables C.3 and C.4, and Tables C.5 and C.6, that, as expected, the model is doing a great job on non-, low- and high-correlated data, since the adjusted R squared in all cases is very close to 1. Indeed, this is borne out with the plots on Figures C.1 and C.2, Figures C.3 and C.4, and Figures C.5 and C.6, where the predicted and the real outcomes form an almost perfect straight line.

Further, according to the adjusted R squared, we observe that the non-correlated data case is getting a better performance on both the training and testing data sets compared to the low-correlated case (though for a little difference). Besides, we observe that the high-correlated data case has the "worst" performance on both the training and testing data sets compared to the others data sets.

Moreover, for the non-correlated model we have that the variable 166 is not relevant, while for the low-correlated model all variables are relevant and for the high-correlated model the variable 172 is not relevant.

4.1.2 Comparison on incomplete data

We observe on Tables C.7 and C.8, Tables C.9 and C.10, and Tables C.11 and C.12, that the model is doing a quite good job on non-, low- and high-correlated data, since the adjusted R squared in all cases for the testing data set is greater than 0.5, having the best result for the non-correlated case with a value of 0.7. Indeed, this is corroborated with the plots on Figures C.7 and C.8, Figures C.9 and C.10, and Figures C.11 and C.12, where the predicted and the real outcomes seem to follow a line.

Further, according to the adjusted R squared, we observe that the non-correlated case has the best performance, followed (in order) by the low-correlated and the high-correlated cases.

4.1.3 Discussion on simulated data

First, we shall say that with the data generated from the theoretical model (3.1), we have obtained, as one could have expected, great results and, clearly, we have succeeded more with the complete data case than with the block-wise missing one, so we could say (at least with the data used) that the missing data affects on the performance of the model by decreasing its effectiveness, since we can observe that the values *MSE/RMSE* and *MAE/RMAE* increase in all cases for the block-wise missing data sets compared to the complete data sets.

Further, surprisingly, the non-correlated case has obtained the best results, as well as the low-correlated better results than the high-correlated.

Moreover, we have not recovered the truly sparse beta model for none of the different data used (where we have used the value 0.001 as a threshold for a component to be non-relevant). This could be caused due to the low iterations needed to obtain each model. Hence, may be with a lower tolerance or allowing the model going through the whole iterations will allow us to obtain better results.

Finally, we should point out that the time used for the computation of such models has been quiet fast.

4.2 Exposome data

To discuss the evaluation of the iSFS model performance on exposome data, we have separated each data set in training (67%) and test (33%) as shown in Appendix C.2. First, we shall mention that for the exposome data with factors converted to binary dummy variables we have not computed, for the testing data set, the adjusted R-squared due to the low number of testing samples (428 samples) compared to the number of variables (294 variables) which will always result in a negative value.

4.2.1 Comparison on complete data

4.2.1.1 Numeric variables

We observe in Tables C.13 and C.14, and Tables C.17 and C.18, that the best results are obtained for the outcomes *hs_zbmi_who* and *hs_correct_raven* with adjusted R squared greater than 0.53 for the training data while for the testing data we obtain 0.375 on *hs_zbmi_who* and 0.128 on *hs_correct_raven*. Further, in Figures C.13 and C.14, and Figures C.17 and C.18, we see how the tendency on the plots is to follow the line $y_{\text{pred}} = y_{\text{real}}$.

Nevertheless, we can not say the same for the outcomes *e3_bw* and *hs_Gen_Tot*, where the effectiveness of the model is poor (see Tables C.15 and C.16, Tables C.19 and C.20), with adjusted R squared negative on the testing data and not following at all (due to some "outliers" predicted values) the line $y_{\text{pred}} = y_{\text{real}}$ (see Figures C.15 and C.16, and Figures C.19 and C.20), having the worst performance for the outcome *hs_Gen_Tot*.

Further, for the outcome *hs_zbmi_who* we have that the non-relevant variables are *h_NO2_Log* and *h_trafload_preg_pow1over3*, while for the outcome *e3_bw* the non-relevant variables are *h_builtdens300_preg_Sqrt*, *hs_builtdens300_h_Sqrt* and *hs_builtdens300_s_Sqrt*. Moreover, for the outcomes *hs_correct_raven* and *hs_Gen_Tot*, all variables seem to be relevant.

4.2.1.2 Dummy variables

We observe in Tables C.21 and C.22, and Tables C.25 and C.26, that the best results are obtained, as in the numeric case, for the outcomes *hs_zbmi_who* and *hs_correct_raven* with adjusted R squared greater than 0.45 for the training data while for the R squared on the testing data we obtain 0.64 on *hs_zbmi_who* and 0.485 on *hs_correct_raven*. Further, in Figures C.21 and C.22, and Figures C.25 and C.26, we see how the tendency on the plots is to follow the line $y_{\text{pred}} = y_{\text{real}}$.

Nevertheless, we can not say the same for the outcomes *e3_bw* and *hs_Gen_Tot*, where the effectiveness of the model is poor (see Tables C.23 and C.24, and Tables C.27 and C.28) and not following at all (due to some "outliers" predicted values) the line $y_{\text{pred}} = y_{\text{real}}$ (see Figures C.23 and C.24, and Figures C.27 and C.28), having the worst performance for the outcome *hs_Gen_Tot*.

Further, for the outcome *hs_zbmi_who* we have that the non-relevant variables are *variable.female*, *h_landuseshan300_preg_None*, *hs_connind300_h_Log*, *hs_builtdens300_s_Sqrt* and also *variable..0.6....6.9.*, while for the outcome *e3_bw* the four variables *hs_builtdens300_h_Sqrt*, *hs_builtdens300_s_Sqrt*, *variable.0.1* and *hs_trcs_madj_Log2* are not relevant. Moreover, for the outcomes *hs_correct_raven* and *hs_Gen_Tot*, all variables seem to be relevant. In this case, we have used the value 0.05 as a threshold for a component to be non-relevant.

4.2.2 Comparison on incomplete data

4.2.2.1 Numeric variables

We observe in Tables C.29 and C.30 that the best result is obtained for the outcome *hs_zbmi_who* with adjusted R squared greater than 0.414 for the training data while for the testing data we obtain 0.118. Further, in Figures C.29 and C.30 we see how the tendency on the plots is (more or less) to follow the line $y_{\text{pred}} = y_{\text{real}}$.

Nevertheless, in this case we can not say the same for the outcomes *e3_bw*, *hs_correct_raven* and *hs_Gen_Tot*, where the effectiveness of the model is poor (see Tables C.31 and C.32, Tables C.33 and C.34, and Tables C.35 and C.36), with adjusted R squared negative on the testing data and not following at all (due to some "outliers" predicted values) the line $y_{\text{pred}} = y_{\text{real}}$ (see Figures C.31 and C.32, Figures C.33 and C.34, and Figures C.35 and C.36), having the worst performance (among those three outcomes) for the outcome *hs_Gen_Tot* and the best one for the outcome *hs_correct_raven* (with which we shall say that, a part from some points, it is not so far for the line $y_{\text{pred}} = y_{\text{real}}$).

4.2.2.2 Dummy variables

We observe in Tables C.37 and C.38 that the best result is obtained for the outcome *hs_zbmi_who* with adjusted R squared greater than 0.429 for the training data while for the testing data we obtain an R squared of 0.58. Further, in Figures C.37 and C.38 we see how the tendency on the plots is (more or less) to follow the line $y_{\text{pred}} = y_{\text{real}}$.

Nevertheless, in this case we can not say the same for the outcomes *e3_bw*, *hs_correct_raven* and *hs_Gen_Tot*, where the effectiveness of the model is poor (see Tables C.39 and C.40, Tables C.41 and C.42, and Tables C.43 and C.44), with adjusted R squared negative on the training data and not following at all (due to some "outliers" predicted values) the line $y_{\text{pred}} = y_{\text{real}}$ (see Figures C.39 and C.40, Figures C.41 and C.42, and Figures C.43 and C.44), having the worst performance (among those three outcomes) for the outcome *hs_correct_raven* and the best one for the outcome *e3_bw*.

4.2.3 Discussion on exposome data

First, we shall say that with the complete exposome data we have obtained quite good results when the outcome were either *hs_zbmi_who* or *hs_correct_raven* in both numeric and dummy variables, while for the block-wise missing data the best results have been got when the outcome is *hs_zbmi_who*. Indeed, in Section 2.3.2 we saw that the variables that could be compensated if having some missing values where those related with the *BMI*, the height and the weight, which could give us an idea why the best performance is related with the outcome *hs_zbmi_who*.

Further, as expected, we have succeeded more with the complete data case than with the block-wise missing one, so we could say that (at least with the data used) that the missing data affects on the performance of the model by decreasing its effectiveness, since we can observe that the values MSE/RMSE and MAE/RMAE increase in all cases for the block-wise missing data sets compared to the complete data sets.

Moreover, when comparing between numeric variables and dummy variables, we obtain that the best results depend strongly on the outcome and if the data is complete or block-wise missing (see Table 4.1). However, the model needs more computational time for the dummy variables than for the numeric variables, which should also be taken into account.

	Complete data	**Block-wise missing data**
hs_zbmi_who	Numeric variables	Dummy variables
e3_bw	Numeric variables	Dummy variables (for a little bit)
hs_correct_raven	Dummy variables (for a little bit)	Numeric variables
hs_Gen_Tot	Dummy variables	Numeric variables (for a little bit)

Table 4.1: Best results between numeric variables and dummy variables data sets according whether the data is complete or not and for the four numeric outcomes of exposome data.

Chapter 5

Conclusions

On this chapter, we present the conclusions of this thesis. Among them, we will also talk about the future research that can be done from this manuscript and the schedule tracking during the time that we have been working on this project.

5.1 Conclusions

When I asked to professors Ferran Reverter and Esteban Vegas whether I can work with them in a project with mathematical background but, of course, with also biostatistical basis, they present me the following issue: on many occasions the information that one can gather is not complete, since for some observations not all data sources are available (what is known as block-wise missing data) so how we could implement an integrative process with block-wise missing data based on a Lasso's type approximation that then could be applied to real omics data.

That is why in this manuscript we have studied a bi-level feature learning model motivated by the exposome data (see Section 2.3.2) and we have implemented a code that approaches for both complete and block-wise missing data (see Chapter 3). Specifically, we have introduced a unified feature learning model for complete data, which contains several classical convex models (see Section 3.1.1.1) that has been easily extended to handling the more challenging case: the block-wise missing data. Further, the effectiveness of the proposed models has been verified through both simulated data and exposome data (see Chapter 4). Therefore, at the end we have succeed in presenting an optimization regression model that given complete or block-wise missing data, we can obtain information from it in order to make predictions for similar structured data.

Finally, I would like to thank the treatment and predisposition received by my tutors, with whom I have had the opportunity to meet periodically in order to advance on this thesis in the best way together. Further, I want to say that coming from a mathematical academic line (by doing a PhD on mathematical analysis) and jumping to this computing optimization problem has been a challenging and interesting change, for which I am very grateful.

5.2 Future research

The future work's lines that have not been explored in this work (so have remained pending) and which we hope to be addressed in the near future are the following:

- Code the model in Python language and then upload it to Github.
- Generate a code for the model in Chapter 3 that deals with an iSFS model for the logistic function. Moreover, modify the model in such a way that could work with factors.
- Study deeper the model in order to decrease its computing time and increase its effectiveness. For instance, one could improve the seek of the parameter β_L^* (see Section 3.1.3) by using back-tracking line by means of, for example, the Amijo's rule [7]. Indeed, one could also apply a different L step for each component independently. Besides, we could have studied more Ω norms for the parameter α than the two proposed in Section 3.1.2.
- For the study of the current model, we could have used different parameters (tuning) and k-fold cross-validation to the sake of better results. Further, we could allowed more iterations since it has been observed that the error model decreases monotonically (at least for the data used) with each iteration. Besides, to help the study of its performance and effectiveness, we could have predicted fictional scenarios or we could have used different Ω functions (for α and β parameters, respectively) and compare between them. All in all, we could have used all the different functionalities that our model have (as, for instance, data normalization) in order to obtain the best possible combination of parameters.
- Generalize the model having also missing values (not just blocks of them) and with sources having just one variable.
- Study the model with the data used in [24, 25] (the reference papers) and compare their results with ours.
- Compare the effectiveness and performance of the model with imputation methods.

5.3 Schedule tracking

In general lines, all the objectives initially proposed in the planning of the study have been achieved. However, the part of investigating possible variants of the model either by using different models or different approaches could have been studied deeper (as we can see on Section 5.2) but the generation of the code that implements an optimization algorithm that models an integrative learning model on either complete or block-wise missing data, and its consequent evaluation, has precised more time than expected. Indeed, due to unforeseen contingencies external to the student, there are variants of the current model that were willing to be addressed and will be in a near future.

For the methodology (see Section 2) we shall mention that we have been able to give an answer for the questions that arised there, so we can affirm that it has been adequate for a thesis of this type, especially for the time we have to develop and write it.

Finally, about the scheduling, we had realized while we were on the half of this journey that before working on treating the exposome data (doing data quality control by seeing how the data is distributed using graphs) first we had to generate random and simulated block-wise missing data and to evaluate the model performance and effectiveness with that data. Also, when computing the parameters α and β of the iSFS model (see Section 3) we had to work hard in order to develop a satisfactory algorithm that compute them. In particular, we run into unexpected problems when dealing with the parameter α that, at the end, have been solved.

Chapter 6

Glossary

The purpose of this chapter is to mention the definitions of the most relevant terms and acronyms used on this thesis alphabetically arranged:

Adjusted R squared:	Correction of R squared proposed by Mordecai Ezekiel [27].
Bi-level learning:	Performs simultaneously feature-level and source-level analysis.
BiB/EDEN/INMA/ KANC/MoBa/Rhea:	UK/France/Spain/Lithuania/Norway/Greece.
BMI:	Body Mass Index.
BTEX:	Compounds of Benzene, Toluene, Ethylbenzene and Xylene.
CBCL:	Child Behavior Checklist.
GIS:	Geographic Information System.
HELIX:	Human Early-Life Exposome.
Imputation:	Assignment of a value to something by inference from the value of the products or processes to which it contributes.
iSFS model:	Incomplete Source Feature Selection.
Lasso:	Least Absolute Shrinkage and Selection Operator.
MAE/RMAE:	Mean Absolute Error/Root Mean Absolute Error.
MSE/RMSE:	Mean Square Error/Root Mean Square Error.
Multi-source analysis:	Comparison of data from multiple sources or from a single source at different times.
NO2:	Nitrogen Dioxide.
PACs:	Plural of the Catalan acronym for Continuous Assessment Test.
PM:	Particular Matter (also called particular pollution).
Profile:	Information described by a decimal integer of the binary indicator vector that specify whether a certain data source is present or not.
R squared:	Coefficient of determination.
RAVEN test:	Psychometric test that measures the level of intelligence.
Sparse model:	Model with a small number of coefficients that are non-zero.

Bibliography

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. *Optimization with Sparsity-Inducing Penalties*, volume 4. Foundations and Trends in Machine Learning, 2012. Available at https://doi.org/10.1561/2200000015.

[2] J. BARZILAI and J. M. BORWEIN. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 01 1988. Available at https://doi.org/10.1093/imanum/8.1.141.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. Available at https://doi.org/10.1137/080716542.

[4] P. Breheny and J. Huang. *Penalized methods for bi-level variable selection*, volume 2. Statistics and its interface, 2009. Available at https://dx.doi.org/10.4310/SII.2009.v2.n3.a10.

[5] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9(57):1757–1774, 2008. Available at http://jmlr.org/papers/v9/crammer08a.html.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2006. Available at https://www.researchgate.net/publication/238735054.

[7] Z. FITRIAH and S. ANAM. Modified armijo rule on gradient descent and conjugate gradient. *E-Jurnal Matematika*, 6(3):196–204, 2017. Available at https://ojs.unud.ac.id/index.php/mtk/article/view/32838.

[8] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. Available at https://doi.org/10.18637/jss.v033.i01.

[9] I. Huopaniemi, T. Suvitaival, J. Nikkilä, M. Orešič, and S. Kaski. Multivariate multi-way analysis of multi-source data. *Bioinformatics*, 26(12):i391–i398, 06 2010. Available at https://doi.org/10.1093/bioinformatics/btq174.

[10] LaTeX Team. The LaTeX Project. https://www.latex-project.org/. Accessed: 2022-06-02.

[11] L. Maitre, J. de Bont, M. Casas, O. Robinson, G. M. Aasvang, L. Agier, S. Andrušaitytė, F. Ballester, X. Basagaña, E. Borràs, C. Brochot, M. Bustamante, A. Carracedo, M. de Castro, A. Dedele, D. Donaire-Gonzalez, X. Estivill, J. Evandt, S. Fossati, L. Giorgis-Allemand, J. R Gonzalez, B. Granum, R. Grazuleviciene, K. Bjerve Gützkow, L. Småstuen Haug, C. Hernandez-Ferrer, B. Heude, J. Ibarluzea, J. Julvez, M. Karachaliou, H. C. Keun, N. Hjertager Krog, C.-H. E. Lau, V. Leventakou, S. Lyon-Caen, C. Manzano, D. Mason, R. McEachan, H. M. Meltzer, I. Petraviciene, J. Quentin, T. Roumeliotaki, E. Sabido, P.-J. Saulnier, A. P. Siskos, V. Siroux, J. Sunyer, I. Tamayo, J. Urquiza, M. Vafeiadi, D. van Gent, M. Vives-Usano, D. Waiblinger, C. Warembourg, L. Chatzi, M. Coen, P. van den Hazel, M. J. Nieuwenhuijsen, R. Slama, C. Thomsen, J. Wright, and M. Vrijheid. Human Early Life Exposome (HELIX) study: a European population-based exposome cohort. *BMJ Open*, 8(9):e021311, Sept. 2018. Available at http://dx.doi.org/10.1136/bmjopen-2017-021311.

[12] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010. Available at http://jmlr.org/papers/v11/mazumder10a.html.

[13] J. J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 255:2897–2899, 1962. Available at https://hal.archives-ouvertes.fr/hal-01867195.

[14] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140:1436–4646, 2013. Available at https://doi.org/10.1007/s10107-012-0629-5.

[15] Overleaf Team. Overleaf. https://www.overleaf.com/. Accessed: 2022-06-02.

[16] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. Available at https://www.R-project.org/.

[17] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, PBC, Boston, MA, 2022. Available at http://www.rstudio.com/.

[18] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. Available at https://www.jstor.org/stable/2346178.

[19] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. Available at https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[20] O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein. A bayesian framework for combining heterogeneous data sources for gene function prediction (in saccharomyces cerevisiae). *Proceedings of the National Academy of Sciences*, 100(14):8348–8353, 2003. Available at https://doi.org/10.1073/pnas.0832373100.

[21] E. van den Berg, M. W. Schmidt, M. P. Friedlander, and K. P. Murphy. Group sparsity via linear-time projection. *UBC - Department of Computer Science*, 2008. Available at http://www.optimization-online.org/DB_FILE/2008/07/2056.pdf.

[22] M. Vrijheid, R. Slama, O. Robinson, L. Chatzi, M. Coen, P. van den Hazel, C. Thomsen, J. Wright, T. J. Athersuch, N. Avellana, X. Basagaña, C. Brochot, L. Bucchini, M. Bustamante, A. Carracedo, M. Casas, X. Estivill, L. Fairley, D. van Gent, J. R. Gonzalez, B. Granum, R. Gražulevicˇieneˊ, K. B. Gutzkow, J. Julvez, H. C. Keun, M. Kogevinas, R. R. McEachan, H. M. Meltzer, E. Sabidó, P. E. Schwarze, V. Siroux, J. Sunyer, E. J. Want, F. Zeman, and M. J. Nieuwenhuijsen. The human early-life exposome (helix): Project rationale and design. *Environmental Health Perspectives*, 122(6):535–544, 2014. Available at https://doi.org/10.1289/ehp.1307204.

[23] S. Xiang, X. Tong, and J. Ye. Efficient sparse group feature selection via nonconvex optimization. In *International Conference on Machine Learning*, pages 284–292. PMLR, 2013. Available at https://proceedings.mlr.press/v28/xiang13.html.

[24] S. Xiang, L. Yuan, W. Fan, Y. Wang, P. M. Thompson, and J. Ye. Multi-source learning with block-wise missing data for Alzheimer's disease prediction. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 185–193, New York, NY, USA, 2013. Association for Computing Machinery. Available at https://doi.org/10.1145/2487575.2487594.

[25] S. Xiang, L. Yuan, W. Fan, Y. Wang, P. M. Thompson, and J. Ye. Bi-level multi-source learning for heterogeneous block-wise missing data. *Neuroimage*, 102:192–206, November 2014. Available at https://doi.org/10.1016/j.neuroimage.2013.08.015.

[26] Z. Xu, I. King, and M. R. Lyu. Web page classification with heterogeneous data fusion. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 1171–1172, New York, NY, USA, 2007. Association for Computing Machinery. Available at https://doi.org/10.1145/1242572.1242750.

[27] P. Yin and X. Fan. Estimating R^2 Shrinkage in Multiple Regression: A Comparison of Different Analytical Methods. *The Journal of Experimental Education*, 69(2):203–224, 2001. Available at https://doi.org/10.1080/00220970109600656.

[28] L. Yuan, Y. Wang, P. M. Thompson, V. A. Narayan, and J. Ye. Multi-source feature learning for joint analysis of incomplete multiple heterogeneous neuroimaging data. *NeuroImage*, 61(3):622–632, 2012. Available at https://doi.org/10.1016/j.neuroimage.2012.03.059.

[29] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B*, 68:49–67, 02 2006. Available at https://doi.org/10.1111/j.1467-9868.2005.00532.x.

[30] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005. Available at https://doi.org/10.1111/j.1467-9868.2005.00503.x.

Appendix A

Code and figures: methodology and materials

A.1 Software for the project development

A.1.1 R and RStudio

```
# Packages used for the development of this manuscript's code
library(ade4)
library(binaryLogic)
library(caret)
library(corrplot)
library(devtools)
library(factoextra)
library(glmnet)
library(MASS)
library(mvtnorm)
library(naniar)
```

A.2 A unified feature learning model for complete and block-wise missing multi-source data

A.2.1 Missing blocks and profiles

```
# Computing the profile vector given the dimensions p_i of each source
# A block of a source with missing data will correspond to samples
# that have any NA in that source
```

```r
get_profile <- function(p, X){
  # Samples and Sources
  n <- dim(X)[1]
  S <- length(p)

  # Profile vector
  pf.vec <- numeric(length = n)
  for(i in 1:n){
    # Profile of i-th sample
    pf <- 0
    col <- 1
    for(j in 1:S){
      nextCol <- col + p[j]
      if(!any(is.na(X[i, col:(nextCol - 1)])))
        pf <- pf + 2^(S - j)

      col <- nextCol
    }

    # Add the i-th profile to the profile vector
    pf.vec[i] <- pf
  }

  return(as.factor(pf.vec))
}
```

```r
# Group all the samples which have m as a profile together
# with those that have complete data in all the sources
# that are contained in the profile m
getBlockSamples <- function(pf.vec, m, S){
  # Get sources of the given profile
  sources.on.profile <- which(as.binary(m, n = S))

  # Set profiles
  profiles <- levels(pf.vec)

  # Add corresponding samples to the block
  samples.block <- numeric()
  for(i in 1:length(profiles)){
    profile <- as.integer(profiles[i])
    if(all(as.binary(profile, n = S)[sources.on.profile]))
      samples.block <-
        c(samples.block, which(pf.vec == profile))
  }
  # Return the block and the sources related to that block
  return(list(samples = samples.block,
              sources = sources.on.profile))
}
```

A.3 Data

A.3.1 Simulated data

```
# Number of samples
n <- 1500
# Number of sources
S <- 20

# Seed for reproducing the whole code
set.seed(123456)

# Sparsity index: number of non-zero elements of non-zero coefficients
sparsity_ind <- 3

# Dimensions of the underlying true model
p.synth <- sample(sparsity_ind:20, size = S, replace = TRUE)

# Values of the non-zero coefficients
values <- c(10, 8, 6, 4, 2, 1)

# Sparse underlying true model
beta <- c()
for(i in 1:S){
  min <- min(sparsity_ind, p.synth[i])
  coef <- c(rep(values[i], each = min),
            rep(0, each = p.synth[i] - min))
  beta <- c(beta, coef*ifelse(rbinom(p.synth[i], 1, 0.5) == 0, -1, 1))
}
beta <- c(beta, rep(0, sum(p.synth) - length(beta)))
```

```
# Noise term
eps <- rnorm(n, mean = 0, sd = 0.5)
```

- Non-correlation between variables

```
# Number of variables
num.var <- sum(p.synth)
# Mean vector equals 0
meanVec <- numeric(length = num.var)
# Standard deviation diagonal matrix
sdDiag <- diag(rep(0.5, num.var))

# Correlation and covariance matrices
corMat_nc <- diag(1, num.var)
Sigma_nc <- sdDiag%*%corMat_nc%*%sdDiag

# Non-correlation between variables
X_nc <- rmvnorm(n = n, mean = meanVec, sigma = Sigma_nc)
```

- Low-correlation between variables

```
# Correlation and covariance matrices
corMat_lc <- diag(0, num.var)
corMat_lc[lower.tri(corMat_lc, diag = FALSE)] <-
runif(num.var*(num.var - 1)/2, min = 0, max = 0.5)
corMat_lc[upper.tri(corMat_lc)] <-
          t(corMat_lc)[upper.tri(corMat_lc)]
corMat_lc <- corMat_lc%*%t(corMat_lc)
corMat_lc <- corMat_lc/(2*max(corMat_lc))
diag(corMat_lc) <- 1
Sigma_lc <- sdDiag%*%corMat_lc%*%sdDiag

# Low-correlation between variables
X_lc <- rmvnorm(n = n, mean = meanVec, sigma = Sigma_lc)
```

- High-correlation between variables

```
# Correlation and covariance matrices
corMat_hc <- diag(0, num.var)
corMat_hc[lower.tri(corMat_hc, diag = FALSE)] <-
runif(num.var*(num.var - 1)/2, min = 0.5, max = 1)
corMat_hc[upper.tri(corMat_hc)] <-
          t(corMat_hc)[upper.tri(corMat_hc)]
corMat_hc <- corMat_hc%*%t(corMat_hc)
corMat_hc <- corMat_hc/max(corMat_hc)
```

```
diag(corMat_hc) <- 1
Sigma_hc <- sdDiag%*%corMat_hc%*%sdDiag

# High-correlation between variables
X_hc <- rmvnorm(n = n, mean = meanVec, sigma = Sigma_hc)
```

```
# Convert complete data matrix to incomplete data randomly
X.NA <- function(X, p){
  S <- length(p)
  X_NA <- X
  for(i in 1:dim(X)[1]){
    num.missing.sources <- sample(1:S, 1)
    missing.sources <- sample(1:length(p), num.missing.sources)

    col <- 1
    for(j in 1:S){
      nextCol <- col + p[j] - 1
      if(j %in% missing.sources)
        X_NA[i, col:nextCol] <- NA

      col <- nextCol
    }
  }

  return(X_NA)
}

X.NA_nc <- X.NA(X_nc, p.synth)
X.NA_lc <- X.NA(X_lc, p.synth)
X.NA_hc <- X.NA(X_hc, p.synth)
```

```
# Outcome
y_nc <- eps
y_lc <- eps
y_hc <- eps

col <- 1
for(i in 1:20){
  nextCol <- col + p.synth[i] - 1
  y_nc <- y_nc + X_nc[, col:nextCol]%*%beta[col:nextCol]
  y_lc <- y_lc + X_lc[, col:nextCol]%*%beta[col:nextCol]
  y_hc <- y_hc + X_hc[, col:nextCol]%*%beta[col:nextCol]
  col <- nextCol + 1
}
```

A.3.2 Exposome data

```
# Exposome variables without ID
exposome <- exposome[,-1]
exposomeNA <- exposomeNA[,-1]

# All families except covariates and outcome variables
families <- levels(codebook$family)[-c(3,14)]

# Complete data
exposome.data <- covariates[,-1]
for(i in 1:length(families))
  exposome.data <- data.frame(exposome.data,
                   exposome[, codebook$family == families[i]])

# Incomplete data
exposomeNA.data <- covariatesNA[,-1]
for(i in 1:length(families))
  exposomeNA.data <-
  data.frame(exposomeNA.data,
             exposomeNA[, codebook$family == families[i]])

# Outcome without ID
y <- phenotype[,-1]
# g to kg
y$e3_bw <- y$e3_bw/1000

# Source of each variable
sources <- rep("0.Covariates", dim(covariates[,-1])[2])
for(i in 1:length(families))
  sources <- c(sources, rep(families[i],
                            sum(codebook$family == families[i])))
```

```
# Distribution of the missing values
vis_miss(exposomeNA.data[,1:20])
```

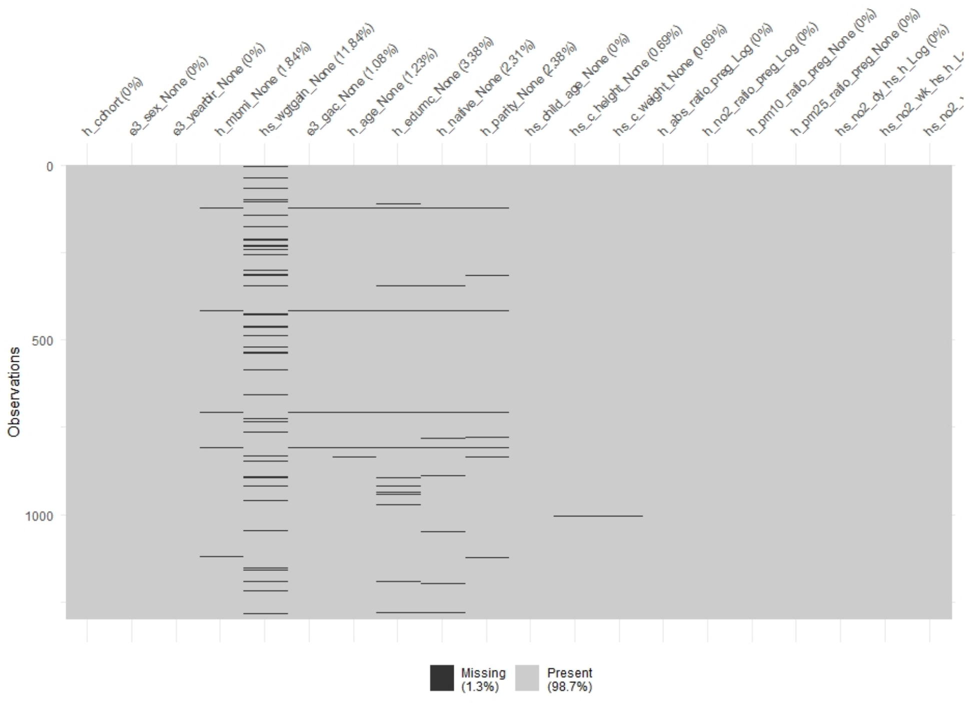


Figure A.1: Missing values pattern of the exposome data with missing data (*exposomeNA*).

```r
# Brief description of the exposome variables consisting
# on the smallest data value, the first quantile, the
# median, the third quantile, and the largest data value
# of each variable respectively
summary(exposome.data)
```

```
h_cohort e3_sex_None  e3_yearbir_None  h_mbmi_None
 1:202    female:608   2003: 55         Min.   :15.88
 2:198    male  :693   2004:107         1st Qu.:21.26
 3:224                 2005:241         Median :24.02
 4:207                 2006:256         Mean   :25.03
 5:272                 2007:250         3rd Qu.:27.34
 6:198                 2008:379         Max.   :51.42
                       2009: 13
 hs_wgtgain_None  e3_gac_None      h_age_None     h_edumc_None
 Min.   : 0.0    Min.   :28.00   Min.   :16.00   1:178
 1st Qu.: 9.0    1st Qu.:38.71   1st Qu.:27.64   2:449
 Median :12.0    Median :40.00   Median :31.00   3:674
 Mean   :13.5    Mean   :39.63   Mean   :30.80
 3rd Qu.:18.0    3rd Qu.:40.71   3rd Qu.:34.06
```

```
Max.   :55.0    Max.   :44.14   Max.   :43.51

h_native_None h_parity_None hs_child_age_None hs_c_height_None
0: 146        0:601         Min.   : 5.437    Min.   :1.054
1:  67        1:464         1st Qu.: 6.500    1st Qu.:1.209
2:1088        2:236         Median : 8.033    Median :1.280
                            Mean   : 7.976    Mean   :1.291
                            3rd Qu.: 8.920    3rd Qu.:1.365
                            Max.   :12.101    Max.   :1.685

hs_c_weight_None h_abs_ratio_preg_Log h_no2_ratio_preg_Log
Min.   :16.00    Min.   :-0.47756     Min.   :2.105
1st Qu.:22.90    1st Qu.: 0.09776     1st Qu.:2.670
Median :26.90    Median : 0.30203     Median :2.963
Mean   :28.52    Mean   : 0.39089     Mean   :3.004
3rd Qu.:32.70    3rd Qu.: 0.72516     3rd Qu.:3.298
Max.   :71.10    Max.   : 1.70921     Max.   :4.525

h_pm10_ratio_preg_None h_pm25_ratio_preg_None hs_no2_dy_hs_h_Log
Min.   : 8.066         Min.   : 6.957         Min.   :0.3797
1st Qu.:17.535         1st Qu.:13.289         1st Qu.:2.2867
Median :23.018         Median :14.879         Median :2.9618
Mean   :23.504         Mean   :15.028         Mean   :2.8307
3rd Qu.:27.677         3rd Qu.:16.999         3rd Qu.:3.4474
Max.   :47.698         Max.   :22.238         Max.   :5.1849

hs_no2_wk_hs_h_Log hs_no2_yr_hs_h_Log hs_pm10_dy_hs_h_None
Min.   :0.9523     Min.   :0.6185     Min.   :  2.916
1st Qu.:2.3313     1st Qu.:2.3800     1st Qu.: 17.818
Median :2.9806     Median :3.0238     Median : 22.899
Mean   :2.8638     Mean   :2.8975     Mean   : 26.214
3rd Qu.:3.3932     3rd Qu.:3.4085     3rd Qu.: 30.937
Max.   :4.8047     Max.   :4.4225     Max.   :157.397

hs_pm10_wk_hs_h_None hs_pm10_yr_hs_h_None hs_pm25_dy_hs_h_None
Min.   :  5.838      Min.   :11.50        Min.   : 1.518
1st Qu.: 19.142      1st Qu.:21.68        1st Qu.: 7.950
Median : 24.891      Median :24.75        Median :12.244
Mean   : 26.409      Mean   :25.10        Mean   :12.897
3rd Qu.: 32.131      3rd Qu.:31.26        3rd Qu.:16.263
Max.   :211.297      Max.   :46.82        Max.   :58.884

hs_pm25_wk_hs_h_None hs_pm25_yr_hs_h_None hs_pm25abs_dy_hs_h_Log
```

```
 Min.   : 3.139       Min.   : 4.829       Min.   :-1.78220
 1st Qu.: 9.340       1st Qu.:10.410       1st Qu.:-0.25857
 Median :12.702       Median :13.110       Median : 0.02163
 Mean   :13.153       Mean   :12.916       Mean   : 0.11514
 3rd Qu.:16.152       3rd Qu.:15.122       3rd Qu.: 0.54459
 Max.   :75.093       Max.   :21.917       Max.   : 2.26537

 hs_pm25abs_wk_hs_h_Log hs_pm25abs_yr_hs_h_Log
 Min.   :-1.03415       Min.   :-0.59670
 1st Qu.:-0.13869       1st Qu.:-0.01657
 Median : 0.04672       Median : 0.17773
 Mean   : 0.16413       Mean   : 0.18058
 3rd Qu.: 0.53700       3rd Qu.: 0.31331
 Max.   : 1.87776       Max.   : 1.36495

 h_accesslines300_preg_dic0 h_accesspoints300_preg_Log
 Min.   :0.0000             Min.   :1.270
 1st Qu.:0.0000             1st Qu.:1.963
 Median :0.0000             Median :2.879
 Mean   :0.1991             Mean   :2.670
 3rd Qu.:0.0000             3rd Qu.:3.349
 Max.   :1.0000             Max.   :4.528

 h_builtdens300_preg_Sqrt h_connind300_preg_Sqrt
 Min.   : 11.02           Min.   : 1.887
 1st Qu.:340.04           1st Qu.: 9.983
 Median :401.49           Median :12.935
 Mean   :417.06           Mean   :12.737
 3rd Qu.:502.97           3rd Qu.:15.898
 Max.   :807.57           Max.   :27.276

 h_fdensity300_preg_Log h_frichness300_preg_None
 Min.   :10.26          Min.   :0.00000
 1st Qu.:10.26          1st Qu.:0.00000
 Median :11.36          Median :0.03509
 Mean   :11.61          Mean   :0.06605
 3rd Qu.:12.83          3rd Qu.:0.12281
 Max.   :15.60          Max.   :0.42105

 h_landuseshan300_preg_None h_popdens_preg_Sqrt
 Min.   :0.0000             Min.   :  0.00
 1st Qu.:0.3408             1st Qu.: 53.79
 Median :0.4232             Median : 74.98
```

```
Mean   :0.4213             Mean   : 77.02
3rd Qu.:0.5070             3rd Qu.: 96.21
Max.   :1.0000             Max.   :261.50

h_walkability_mean_preg_None hs_accesslines300_h_dic0
Min.   :0.1000               Min.   :0.0000
1st Qu.:0.2000               1st Qu.:0.0000
Median :0.2500               Median :0.0000
Mean   :0.2674               Mean   :0.1852
3rd Qu.:0.3250               3rd Qu.:0.0000
Max.   :0.6250               Max.   :1.0000

hs_accesspoints300_h_Log hs_builtdens300_h_Sqrt hs_connind300_h_Log
Min.   :0.5771           Min.   : 20.3          Min.   :1.270
1st Qu.:1.6753           1st Qu.:300.4          1st Qu.:4.405
Median :2.7738           Median :375.5          Median :4.959
Mean   :2.4051           Mean   :381.1          Mean   :4.776
3rd Qu.:3.2846           3rd Qu.:459.1          3rd Qu.:5.364
Max.   :4.5838           Max.   :805.8          Max.   :6.617

hs_fdensity300_h_Log hs_landuseshan300_h_None hs_popdens_h_Sqrt
Min.   :10.26        Min.   :0.0000           Min.   :  1.732
1st Qu.:10.26        1st Qu.:0.3138           1st Qu.: 30.036
Median :10.96        Median :0.4028           Median : 67.405
Mean   :11.38        Mean   :0.3970           Mean   : 67.652
3rd Qu.:12.34        3rd Qu.:0.4929           3rd Qu.: 84.988
Max.   :14.98        Max.   :0.6619           Max.   :261.500

hs_walkability_mean_h_None hs_accesslines300_s_dic0
Min.   :0.100              Min.   :0.0000
1st Qu.:0.275              1st Qu.:0.0000
Median :0.300              Median :0.0000
Mean   :0.326              Mean   :0.1883
3rd Qu.:0.375              3rd Qu.:0.0000
Max.   :0.600              Max.   :1.0000

hs_accesspoints300_s_Log hs_builtdens300_s_Sqrt hs_connind300_s_Log
Min.   :0.5771           Min.   :  6.432        Min.   :1.270
1st Qu.:1.6753           1st Qu.:314.349        1st Qu.:4.528
Median :2.5225           Median :380.503        Median :4.933
Mean   :2.3902           Mean   :400.029        Mean   :4.791
3rd Qu.:3.2846           3rd Qu.:480.133        3rd Qu.:5.364
Max.   :4.0730           Max.   :805.140        Max.   :6.578
```

```
hs_fdensity300_s_Log hs_landuseshan300_s_None hs_popdens_s_Sqrt
Min.   :10.26        Min.   :0.08298          Min.   :  0.00
1st Qu.:10.26        1st Qu.:0.34004          1st Qu.: 38.56
Median :11.36        Median :0.44793          Median : 69.26
Mean   :11.56        Mean   :0.42993          Mean   : 68.10
3rd Qu.:12.57        3rd Qu.:0.53689          3rd Qu.: 84.99
Max.   :15.25        Max.   :0.72770          Max.   :210.95

h_Absorbance_Log   h_Benzene_Log      h_NO2_Log
Min.   :-0.92737   Min.   :-0.3296   Min.   :1.573
1st Qu.:-0.54273   1st Qu.: 0.3141   1st Qu.:2.979
Median :-0.26937   Median : 0.5600   Median :3.617
Mean   :-0.16919   Mean   : 0.5987   Mean   :3.833
3rd Qu.: 0.02422   3rd Qu.: 0.8437   3rd Qu.:4.576
Max.   : 3.40474   Max.   : 1.9975   Max.   :7.093

   h_PM_Log        h_TEX_Log      e3_alcpreg_yn_None
Min.   :1.549   Min.   :1.926   0:896
1st Qu.:2.069   1st Qu.:2.601   1:405
Median :2.304   Median :2.976
Mean   :2.443   Mean   :2.999
3rd Qu.:2.699   3rd Qu.:3.363
Max.   :5.236   Max.   :4.944

     h_bfdur_Ter   h_cereal_preg_Ter    h_dairy_preg_Ter
(0,10.8]   :506   (0,9]      :531     (0,17.1]   :270
(10.8,34.9]:270   (9,27.3]   :459     (17.1,27.1]:380
(34.9,Inf] :525   (27.3,Inf]:311     (27.1,Inf] :651




 h_fastfood_preg_Ter  h_fish_preg_Ter h_folic_t1_None
(0,0.25]   : 94      (0,1.9]  :343    0:606
(0.25,0.83]:535      (1.9,4.1]:490    1:695
(0.83,Inf] :672      (4.1,Inf]:468




  h_fruit_preg_Ter h_legume_preg_Ter h_meat_preg_Ter
(0,0.6]    :  6     (0,0.5]:245       (0,6.5] :427
(0.6,18.2]:922      (0.5,2]:269        (6.5,10]:387
(18.2,Inf]:373      (2,Inf]:787        (10,Inf]:487




  h_pamod_t3_None h_pavig_t3_None    h_veg_preg_Ter
None       : 42    High  : 47      (0,8.8]   :539
Often      :474    Low   :952      (8.8,16.5]:470
Sometimes :191     Medium:302      (16.5,Inf]:292
Very Often:594




hs_bakery_prod_Ter  hs_beverages_Ter  hs_break_cer_Ter
(0,2]   :345        (0,0.132]:331     (0,1.1]  :291
(2,6]   :423        (0.132,1]:454     (1.1,5.5]:521
(6,Inf]:533         (1,Inf]  :516     (5.5,Inf]:489




  hs_caff_drink_Ter      hs_dairy_Ter    hs_fastfood_Ter
(0,0.132]  :808      (0,14.6]   :359   (0,0.132]  :143
(0.132,Inf]:493      (14.6,25.6]:465   (0.132,0.5]:603
                    (25.6,Inf] :477   (0.5,Inf]  :555




hs_KIDMED_None   hs_mvpa_prd_alt_None  hs_org_food_Ter
Min.   :-3.000   Min.   :-27.76       (0,0.132]:429
1st Qu.: 2.000   1st Qu.: 23.27       (0.132,1]:396
Median : 3.000   Median : 34.71       (1,Inf]  :476
Mean   : 2.881   Mean   : 37.87
3rd Qu.: 4.000   3rd Qu.: 47.75
Max.   : 9.000   Max.   :146.75

hs_pet_cat_r2_None hs_pet_dog_r2_None hs_pet_None hs_proc_meat_Ter
0:1059             0:1108             No :807     (0,1.5]:366
1: 242             1: 193             Yes:494     (1.5,4]:471
                                                  (4,Inf]:464
   hs_readymade_Ter hs_sd_wk_None      hs_total_bread_Ter
(0,0.132]  :327     Min.   :  3.143   (0,7]      :431
(0.132,0.5]:296     1st Qu.:155.714   (7,17.5]   :381
(0.5,Inf]  :678     Median :210.000   (17.5,Inf]:489
                    Mean   :235.809
                    3rd Qu.:282.857
                    Max.   :994.286

 hs_total_cereal_Ter hs_total_fish_Ter hs_total_fruits_Ter
(0,14.1]    :418      (0,1.5]:389       (0,7]      :413
(14.1,23.6]:442      (1.5,3]:454       (7,14.1]   :407
(23.6,Inf]  :441      (3,Inf]:458       (14.1,Inf]:481
```

```
hs_total_lipids_Ter hs_total_meat_Ter hs_total_potatoes_Ter
(0,3]  :397          (0,6]  :425       (0,3]  :417
(3,7]  :403          (6,9]  :411       (3,4]  :405
(7,Inf]:501          (9,Inf]:465       (4,Inf]:479
```

```
hs_total_sweets_Ter  hs_total_veg_Ter  hs_total_yog_Ter
(0,4.1]  :344        (0,6]    :404     (0,6]    :779
(4.1,8.5]:516        (6,8.5]  :314     (6,8.5]  :308
(8.5,Inf]:441        (8.5,Inf]:583     (8.5,Inf]:214
```

```
hs_dif_hours_total_None  hs_as_c_Log2       hs_as_m_Log2
Min.   : 7.901          Min.   :-15.0124   Min.   :-38.625
1st Qu.: 9.794          1st Qu.: -4.0075   1st Qu.: -5.419
Median :10.330          Median :  0.4854   Median : -1.925
Mean   :10.296          Mean   : -0.9947   Mean   : -3.011
3rd Qu.:10.741          3rd Qu.:  1.2630   3rd Qu.:  1.007
Max.   :12.852          Max.   :  4.8227   Max.   :  6.493
```

```
  hs_cd_c_Log2       hs_cd_m_Log2      hs_co_c_Log2
 Min.   :-10.395   Min.   :-7.844   Min.   :-5.546
 1st Qu.: -4.399   1st Qu.:-2.671   1st Qu.:-2.718
 Median : -3.818   Median :-2.427   Median :-2.427
 Mean   : -3.969   Mean   :-2.179   Mean   :-2.344
 3rd Qu.: -3.393   3rd Qu.:-1.713   3rd Qu.:-2.041
 Max.   :  0.840   Max.   : 4.802   Max.   : 1.401

  hs_co_m_Log2      hs_cs_c_Log2       hs_cs_m_Log2
 Min.   :-5.184   Min.   :-1.45403   Min.   :-1.15843
 1st Qu.:-2.515   1st Qu.: 0.05658   1st Qu.: 0.07039
 Median :-2.012   Median : 0.46467   Median : 0.40054
 Mean   :-1.694   Mean   : 0.44276   Mean   : 0.48140
 3rd Qu.:-0.550   3rd Qu.: 0.80735   3rd Qu.: 0.80736
 Max.   : 2.503   Max.   : 3.06523   Max.   : 3.44626

  hs_cu_c_Log2      hs_cu_m_Log2     hs_hg_c_Log2
 Min.   : 9.079   Min.   : 9.036   Min.   :-10.8954
 1st Qu.: 9.681   1st Qu.:10.253   1st Qu.: -1.2277
 Median : 9.828   Median :10.441   Median : -0.1959
 Mean   : 9.828   Mean   :10.402   Mean   : -0.2980
 3rd Qu.: 9.966   3rd Qu.:10.541   3rd Qu.:  0.8237
 Max.   :12.123   Max.   :11.167   Max.   :  3.6554

  hs_hg_m_Log2       hs_mn_c_Log2     hs_mn_m_Log2
 Min.   :-9.0230   Min.   :1.705   Min.   :1.655
 1st Qu.:-0.3094   1st Qu.:2.836   1st Qu.:3.291
 Median : 0.5753   Median :3.119   Median :3.573
 Mean   : 0.5698   Mean   :3.128   Mean   :3.542
 3rd Qu.: 1.5705   3rd Qu.:3.392   3rd Qu.:3.807
 Max.   : 5.4429   Max.   :4.792   Max.   :5.446

  hs_mo_c_Log2        hs_mo_m_Log2       hs_pb_c_Log2
 Min.   :-9.23481   Min.   :-2.7179   Min.   :1.084
 1st Qu.:-0.76121   1st Qu.:-0.9828   1st Qu.:2.680
 Median :-0.40354   Median :-0.7322   Median :3.103
 Mean   :-0.31526   Mean   :-0.6933   Mean   :3.108
 3rd Qu.: 0.02857   3rd Qu.:-0.3978   3rd Qu.:3.485
 Max.   : 5.12101   Max.   : 6.1334   Max.   :7.735

  hs_pb_m_Log2     hs_tl_cdich_None   hs_tl_mdich_None
 Min.   :1.220   Detected  : 102    Detected  :  17
 1st Qu.:2.618   Undetected:1199    Undetected:1284
 Median :3.189
 Mean   :3.211
 3rd Qu.:3.807
 Max.   :7.547

 h_humidity_preg_None h_pressure_preg_None h_temperature_preg_None
 Min.   :55.83        Min.   : 974.9       Min.   : 3.120
 1st Qu.:70.63        1st Qu.: 980.8       1st Qu.: 8.127
 Median :77.10        Median : 983.4       Median :10.155
 Mean   :76.56        Mean   : 991.5       Mean   :11.195
 3rd Qu.:86.54        3rd Qu.:1002.3       3rd Qu.:13.798
 Max.   :90.67        Max.   :1015.5       Max.   :22.566

 hs_hum_mt_hs_h_None hs_tm_mt_hs_h_None hs_uvdvf_mt_hs_h_None
 Min.   :52.05       Min.   :-3.477     Min.   :0.007
 1st Qu.:64.99       1st Qu.: 6.761     1st Qu.:0.259
 Median :72.89       Median :12.442     Median :1.009
 Mean   :73.91       Mean   :11.611     Mean   :1.403
 3rd Qu.:82.55       3rd Qu.:16.092     3rd Qu.:2.308
 Max.   :96.14       Max.   :27.271     Max.   :5.150

 hs_hum_dy_hs_h_None hs_hum_wk_hs_h_None hs_tm_dy_hs_h_None
 Min.   : 26.19      Min.   :48.59       Min.   :-7.90
 1st Qu.: 59.15      1st Qu.:63.82       1st Qu.: 6.20
 Median : 72.27      Median :73.75       Median :12.00
 Mean   : 72.75      Mean   :74.07       Mean   :11.44
 3rd Qu.: 85.00      3rd Qu.:84.38       3rd Qu.:16.18
 Max.   :100.00      Max.   :98.62       Max.   :30.70

 hs_tm_wk_hs_h_None hs_uvdvf_dy_hs_h_None hs_uvdvf_wk_hs_h_None
 Min.   :-5.605     Min.   :0.000         Min.   :0.001429
 1st Qu.: 6.745     1st Qu.:0.220         1st Qu.:0.234286
 Median :12.375     Median :1.030         Median :1.101429
 Mean   :11.442     Mean   :1.439         Mean   :1.446599
 3rd Qu.:16.167     3rd Qu.:2.380         3rd Qu.:2.407143
 Max.   :27.688     Max.   :5.550         Max.   :5.254286

 hs_blueyn300_s_None h_blueyn300_preg_None h_greenyn300_preg_None
 0:1208              0:1194                0:321
 1:  93              1: 107                1:980
```

```
h_ndvi100_preg_None hs_greenyn300_s_None hs_blueyn300_h_None
Min.   :0.1062      0: 283               0:1184
1st Qu.:0.2488      1:1018               1: 117
Median :0.4105
Mean   :0.3917
3rd Qu.:0.5158
Max.   :0.7354

hs_greenyn300_h_None hs_ndvi100_h_None hs_ndvi100_s_None
0: 274               Min.   :0.09675   Min.   :0.09519
1:1027               1st Qu.:0.31847   1st Qu.:0.31576
                     Median :0.47907   Median :0.44998
                     Mean   :0.45053   Mean   :0.41609
                     3rd Qu.:0.57471   3rd Qu.:0.52503
                     Max.   :0.81432   Max.   :0.75681

h_lden_cat_preg_None hs_ln_cat_h_None hs_lden_cat_s_None
Min.   :33.92        1:476            1:580
1st Qu.:50.00        2:633            2:265
Median :58.63        3:104            3:299
Mean   :57.47        4: 61            4:104
3rd Qu.:64.36        5: 27            5: 37
Max.   :77.40                         6: 16

hs_dde_cadj_Log2 hs_dde_madj_Log2  hs_ddt_cadj_Log2
Min.   : 1.192   Min.   : 0.8634   Min.   :-15.4250
1st Qu.: 3.563   1st Qu.: 4.4580   1st Qu.: -1.7517
Median : 4.454   Median : 5.5719   Median : -0.4731
Mean   : 4.669   Mean   : 5.8409   Mean   : -1.5790
3rd Qu.: 5.509   3rd Qu.: 7.0023   3rd Qu.:  0.7681
Max.   :11.075   Max.   :10.8937   Max.   :  7.6305

hs_ddt_madj_Log2   hs_hcb_cadj_Log2  hs_hcb_madj_Log2
Min.   :-14.1418   Min.   :-13.136   Min.   :-9.420
1st Qu.: -0.2646   1st Qu.:  2.650   1st Qu.: 2.315
Median :  0.6778   Median :  3.050   Median : 2.797
Mean   :  0.8748   Mean   :  3.154   Mean   : 2.955
3rd Qu.:  1.5125   3rd Qu.:  3.520   3rd Qu.: 3.486
Max.   :  6.5566   Max.   :  6.461   Max.   : 7.357

hs_pcb118_cadj_Log2 hs_pcb118_madj_Log2 hs_pcb138_cadj_Log2
```

```
 Min.   :-6.9507     Min.   :-1.170      Min.   :-9.432  
 1st Qu.: 0.6038     1st Qu.: 0.627      1st Qu.: 1.744  
 Median : 1.0007     Median : 1.052      Median : 2.416  
 Mean   : 1.1023     Mean   : 1.250      Mean   : 2.402  
 3rd Qu.: 1.5596     3rd Qu.: 1.829      3rd Qu.: 3.110  
 Max.   : 4.7829     Max.   : 7.426      Max.   : 7.746  

 hs_pcb138_madj_Log2 hs_pcb153_cadj_Log2 hs_pcb153_madj_Log2
 Min.   :-10.187     Min.   :1.207       Min.   :1.110      
 1st Qu.:  1.788     1st Qu.:2.858       1st Qu.:2.852      
 Median :  2.921     Median :3.519       Median :3.854      
 Mean   :  2.868     Mean   :3.555       Mean   :3.892      
 3rd Qu.:  3.794     3rd Qu.:4.218       3rd Qu.:4.739      
 Max.   :  8.206     Max.   :7.764       Max.   :9.839      

 hs_pcb170_cadj_Log2 hs_pcb170_madj_Log2 hs_pcb180_cadj_Log2
 Min.   :-16.8417    Min.   :-2.0418     Min.   :-11.7198   
 1st Qu.: -0.8488    1st Qu.:-0.3211     1st Qu.:  0.6983   
 Median :  0.2765    Median : 0.8727     Median :  1.8340   
 Mean   : -0.3076    Mean   : 1.0875     Mean   :  1.7477   
 3rd Qu.:  1.3909    3rd Qu.: 2.2000     3rd Qu.:  3.0077   
 Max.   :  4.7832    Max.   : 7.7831     Max.   :  5.8781   

 hs_pcb180_madj_Log2 hs_sumPCBs5_cadj_Log2 hs_sumPCBs5_madj_Log2
 Min.   :-10.121     Min.   :2.182         Min.   :2.299        
 1st Qu.:  2.069     1st Qu.:3.857         1st Qu.:4.007        
 Median :  2.990     Median :4.612         Median :4.715        
 Mean   :  2.946     Mean   :4.647         Mean   :4.860        
 3rd Qu.:  4.034     3rd Qu.:5.372         3rd Qu.:5.738        
 Max.   :  9.349     Max.   :9.277         Max.   :9.341        

 hs_dep_cadj_Log2   hs_dep_madj_Log2   hs_detp_cadj_Log2
 Min.   :-12.5924   Min.   :-13.4083   Min.   :-15.4450  
 1st Qu.: -0.9973   1st Qu.:  0.9887   1st Qu.: -5.1816  
 Median :  0.9287   Median :  1.6631   Median : -3.3437  
 Mean   :  0.1606   Mean   :  1.7010   Mean   : -2.4230  
 3rd Qu.:  2.2958   3rd Qu.:  2.6659   3rd Qu.:  0.7957  
 Max.   :  9.3767   Max.   :  7.5853   Max.   :  6.2939  

 hs_detp_madj_Log2  hs_dmdtp_cdich_None hs_dmp_cadj_Log2
 Min.   :-28.3791   Detected  : 227     Min.   :-16.6419  
 1st Qu.: -3.9329   Undetected:1074     1st Qu.: -4.7344  
 Median : -0.5251                       Median : -0.2684  
 Mean   : -1.5667                         Mean   : -1.4156
 3rd Qu.:  1.0079                         3rd Qu.:  2.2472
 Max.   :  5.4700                         Max.   :  6.3794

 hs_dmp_madj_Log2  hs_dmtp_cadj_Log2  hs_dmtp_madj_Log2
 Min.   :-17.141   Min.   :-10.6455   Min.   :-15.327
 1st Qu.:  2.011   1st Qu.:  0.3311   1st Qu.:  1.072
 Median :  2.796   Median :  1.5927   Median :  2.225
 Mean   :  2.243   Mean   :  1.1332   Mean   :  1.612
 3rd Qu.:  3.756   3rd Qu.:  2.7625   3rd Qu.:  3.489
 Max.   :  8.333   Max.   :  8.6635   Max.   :  7.780

 hs_pfhxs_c_Log2   hs_pfhxs_m_Log2    hs_pfna_c_Log2
 Min.   :-8.8953   Min.   :-17.8296   Min.   :-8.1484
 1st Qu.:-2.3783   1st Qu.: -1.7277   1st Qu.:-1.7387
 Median :-1.4426   Median : -0.9284   Median :-1.0643
 Mean   :-1.5722   Mean   : -0.9841   Mean   :-1.0798
 3rd Qu.:-0.7102   3rd Qu.: -0.1648   3rd Qu.:-0.4677
 Max.   : 4.8309   Max.   :  3.7592   Max.   : 2.7178

 hs_pfna_m_Log2      hs_pfoa_c_Log2    hs_pfoa_m_Log2
 Min.   :-10.75405   Min.   :-2.2197   Min.   :-5.4760
 1st Qu.: -1.31140   1st Qu.: 0.2453   1st Qu.: 0.4107
 Median : -0.58631   Median : 0.6274   Median : 1.2007
 Mean   : -0.75352   Mean   : 0.6102   Mean   : 1.0479
 3rd Qu.:  0.09482   3rd Qu.: 0.9507   3rd Qu.: 1.7450
 Max.   :  2.56486   Max.   : 2.7352   Max.   : 4.9836

 hs_pfos_c_Log2     hs_pfos_m_Log2   hs_pfunda_c_Log2
 Min.   :-10.4131   Min.   :-1.824   Min.   :-11.784
 1st Qu.:  0.3699   1st Qu.: 1.961   1st Qu.: -5.013
 Median :  1.0274   Median : 2.649   Median : -4.078
 Mean   :  0.9700   Mean   : 2.556   Mean   : -4.246
 3rd Qu.:  1.6747   3rd Qu.: 3.213   3rd Qu.: -3.272
 Max.   :  5.0801   Max.   : 5.584   Max.   :  0.593

 hs_pfunda_m_Log2    hs_bpa_cadj_Log2 hs_bpa_madj_Log2
 Min.   :-26.21246   Min.   :-7.150   Min.   :-11.020
 1st Qu.: -3.21222   1st Qu.: 1.270   1st Qu.:  0.292
 Median : -2.47816   Median : 2.014   Median :  1.146
 Mean   : -2.65699   Mean   : 2.144   Mean   :  1.467
 3rd Qu.: -1.71446   3rd Qu.: 2.875   3rd Qu.:  2.340
 Max.   : -0.04217   Max.   : 7.833   Max.   :  6.736
```

```
hs_bupa_cadj_Log2 hs_bupa_madj_Log2 hs_etpa_cadj_Log2
Min.   :-13.940   Min.   :-15.578   Min.   :-6.0647
1st Qu.: -4.385   1st Qu.: -1.341   1st Qu.:-1.2022
Median : -3.472   Median :  1.420   Median :-0.5644
Mean   : -3.532   Mean   :  1.016   Mean   :-0.1302
3rd Qu.: -2.574   3rd Qu.:  3.603   3rd Qu.: 0.3723
Max.   :  6.597   Max.   :  8.534   Max.   :10.9895

hs_etpa_madj_Log2 hs_mepa_cadj_Log2 hs_mepa_madj_Log2
Min.   :-12.119   Min.   :-6.907    Min.   :-0.3096
1st Qu.:  1.240   1st Qu.: 1.696    1st Qu.: 5.8817
Median :  3.280   Median : 2.672    Median : 7.7170
Mean   :  3.330   Mean   : 3.394    Mean   : 7.3042
3rd Qu.:  5.127   3rd Qu.: 4.692    3rd Qu.: 8.6247
Max.   : 12.726   Max.   :14.549    Max.   :15.2601

hs_oxbe_cadj_Log2 hs_oxbe_madj_Log2  hs_prpa_cadj_Log2
Min.   :-4.1446   Min.   :-10.5100   Min.   :-12.0208
1st Qu.:-0.1665   1st Qu.:  0.7601   1st Qu.: -4.3879
Median : 1.1184   Median :  2.5546   Median : -2.2575
Mean   : 1.4523   Mean   :  3.0346   Mean   : -1.6065
3rd Qu.: 2.7929   3rd Qu.:  4.7789   3rd Qu.:  0.8151
Max.   :12.9631   Max.   : 13.6480   Max.   : 10.7801

hs_prpa_madj_Log2 hs_trcs_cadj_Log2 hs_trcs_madj_Log2
Min.   :-14.154   Min.   :-4.3599   Min.   :-4.8110
1st Qu.:  3.754   1st Qu.:-1.6413   1st Qu.: 0.5526
Median :  5.775   Median :-0.7294   Median : 2.6584
Mean   :  5.228   Mean   :-0.3519   Mean   : 3.4281
3rd Qu.:  7.073   3rd Qu.: 0.5389   3rd Qu.: 6.5909
Max.   : 13.605   Max.   : 9.2782   Max.   :10.6909

hs_mbzp_cadj_Log2 hs_mbzp_madj_Log2 hs_mecpp_cadj_Log2
Min.   :-0.5586   Min.   :-3.738    Min.   : 2.631
1st Qu.: 1.6442   1st Qu.: 1.861    1st Qu.: 4.412
Median : 2.3435   Median : 2.887    Median : 5.136
Mean   : 2.4435   Mean   : 2.978    Mean   : 5.190
3rd Qu.: 3.1093   3rd Qu.: 4.097    3rd Qu.: 5.915
Max.   : 7.1847   Max.   : 9.304    Max.   :10.628

hs_mecpp_madj_Log2 hs_mehhp_cadj_Log2 hs_mehhp_madj_Log2
Min.   : 2.427     Min.   : 1.820     Min.   :-0.4596
```

```
 1st Qu.: 4.327     1st Qu.: 3.644     1st Qu.: 3.4564  
 Median : 4.851     Median : 4.350     Median : 4.0677  
 Mean   : 5.027     Mean   : 4.398     Mean   : 4.1568  
 3rd Qu.: 5.632     3rd Qu.: 5.050     3rd Qu.: 4.7897  
 Max.   :10.411     Max.   :11.130     Max.   : 9.9176  

 hs_mehp_cadj_Log2 hs_mehp_madj_Log2 hs_meohp_cadj_Log2
 Min.   :-1.6330   Min.   :-7.469    Min.   : 1.138    
 1st Qu.: 0.8235   1st Qu.: 1.793    1st Qu.: 2.903    
 Median : 1.5741   Median : 3.057    Median : 3.633    
 Mean   : 1.6142   Mean   : 2.940    Mean   : 3.696    
 3rd Qu.: 2.3459   3rd Qu.: 3.808    3rd Qu.: 4.378    
 Max.   : 8.1407   Max.   : 8.702    Max.   :10.332    

 hs_meohp_madj_Log2 hs_mep_cadj_Log2 hs_mep_madj_Log2
 Min.   :-0.0179    Min.   : 1.748   Min.   : 3.292  
 1st Qu.: 3.1001    1st Qu.: 4.015   1st Qu.: 6.398  
 Median : 3.6836    Median : 5.054   Median : 7.776  
 Mean   : 3.7810    Mean   : 5.261   Mean   : 7.772  
 3rd Qu.: 4.4199    3rd Qu.: 6.257   3rd Qu.: 8.911  
 Max.   : 9.6122    Max.   :11.642   Max.   :14.114  

 hs_mibp_cadj_Log2 hs_mibp_madj_Log2 hs_mnbp_cadj_Log2
 Min.   :2.321     Min.   :0.9264    Min.   :1.866    
 1st Qu.:4.719     1st Qu.:4.5921    1st Qu.:3.962    
 Median :5.413     Median :5.3438    Median :4.621    
 Mean   :5.461     Mean   :5.3105    Mean   :4.676    
 3rd Qu.:6.196     3rd Qu.:5.9232    3rd Qu.:5.304    
 Max.   :9.750     Max.   :9.4609    Max.   :8.932    

 hs_mnbp_madj_Log2 hs_ohminp_cadj_Log2 hs_ohminp_madj_Log2
 Min.   :-0.7106   Min.   :-0.2821     Min.   :-11.4619   
 1st Qu.: 4.1958   1st Qu.: 1.7093     1st Qu.: -0.7237   
 Median : 4.8550   Median : 2.4143     Median : -0.2093   
 Mean   : 4.9574   Mean   : 2.5870     Mean   : -0.2990   
 3rd Qu.: 5.5687   3rd Qu.: 3.1967     3rd Qu.:  0.2665   
 Max.   :12.6539   Max.   : 9.0983     Max.   :  6.0560   

 hs_oxominp_cadj_Log2 hs_oxominp_madj_Log2 hs_sumDEHP_cadj_Log2
 Min.   :-0.9126      Min.   :-11.55154    Min.   : 2.648      
 1st Qu.: 0.8939      1st Qu.: -0.69643    1st Qu.: 5.244      
 Median : 1.4939      Median : -0.01846    Median : 6.004      
 Mean   : 1.6735      Mean   : -0.05541    Mean   : 6.049      
 3rd Qu.: 2.2830      3rd Qu.:  0.51914    3rd Qu.: 6.839
 Max.   : 9.4093      Max.   :  5.55327    Max.   :10.052

 hs_sumDEHP_madj_Log2 hs_pbde153_cadj_Log2 hs_pbde153_madj_Log2
 Min.   : 3.211       Min.   :-17.631      Min.   :-15.0030
 1st Qu.: 5.226       1st Qu.: -7.963      1st Qu.: -1.8848
 Median : 5.880       Median : -2.618      Median : -0.9487
 Mean   : 6.015       Mean   : -4.525      Mean   : -1.7406
 3rd Qu.: 6.697       3rd Qu.: -1.246      3rd Qu.: -0.0321
 Max.   :11.691       Max.   :  4.045      Max.   :  6.4338

 hs_pbde47_cadj_Log2 hs_pbde47_madj_Log2 FAS_cat_None
 Min.   :-15.357     Min.   :-11.5808    Low   :146
 1st Qu.: -2.729     1st Qu.: -1.7581    Middle:486
 Median : -2.148     Median : -0.9687    High  :669
 Mean   : -2.606     Mean   : -0.7793
 3rd Qu.: -1.535     3rd Qu.:  0.1183
 Max.   :  5.381     Max.   :  5.1183

        hs_contactfam_3cat_num_None hs_hm_pers_None
 (almost) Daily       :863          Min.   : 1.000
 Once a week          :382          1st Qu.: 4.000
 Less than once a week: 56          Median : 4.000
                                    Mean   : 4.248
                                    3rd Qu.: 5.000
                                    Max.   :10.000

           hs_participation_3cat_None e3_asmokcigd_p_None
 None                    :748          Min.   : 0.000
 1 organisation          :355          1st Qu.: 0.000
 2 or more organisations:198           Median : 0.000
                                       Mean   : 0.494
                                       3rd Qu.: 0.000
                                       Max.   :15.238

 hs_cotinine_cdich_None hs_cotinine_mcat_None   hs_globalexp2_None
 Detected  : 223        Non-smokers:759       exposure   :463
 Undetected:1078        SHS smokers:157       no exposure:838
                        Smokers    :385
 hs_smk_parents_None h_distinvnear1_preg_Log
 both    :142        Min.   :-10.022
 neither:814         1st Qu.: -3.980
 one    :345         Median : -3.002
                     Mean   : -3.153
                     3rd Qu.: -2.256
                     Max.   :  2.794

 h_trafload_preg_pow1over3 h_trafnear_preg_pow1over3
 Min.   :  0.3458          Min.   : 0.000
 1st Qu.: 33.6542          1st Qu.: 7.937
 Median : 66.6101          Median :12.119
 Mean   : 75.5390          Mean   :14.989
 3rd Qu.:113.0812          3rd Qu.:21.397
 Max.   :294.2705          Max.   :39.321

 hs_trafload_h_pow1over3 hs_trafnear_h_pow1over3 h_bro_preg_Log
 Min.   :  0.00          Min.   : 0.000          Min.   :-2.9759
 1st Qu.: 77.42          1st Qu.: 8.434          1st Qu.:-0.5009
 Median :114.87          Median :14.841          Median : 1.8701
 Mean   :112.70          Mean   :15.977          Mean   : 1.2640
 3rd Qu.:136.00          3rd Qu.:22.104          3rd Qu.: 2.7488
 Max.   :293.58          Max.   :49.348          Max.   : 4.9016

 h_clf_preg_Log    h_thm_preg_Log
 Min.   :-6.9078   Min.   :-1.600
 1st Qu.:-0.4959   1st Qu.: 1.849
 Median : 2.0776   Median : 2.912
 Mean   : 0.9645   Mean   : 2.709
 3rd Qu.: 3.1781   3rd Qu.: 3.839
 Max.   : 3.8334   Max.   : 5.031
```

```r
# Variables type without outcomes
var_indexes <- which(!(codebook$family == "Phenotype"))
var_type <- codebook$var_type[var_indexes]

# Percentages of variable's type
round(table(var_type)/length(var_type), 4)*100
```

```
var_type
 factor numeric
  25.11   74.89
```

- Exposome data without factor variables (numeric variables)

```
# Factors on exposome data
factors.exposome <- which(as.vector(sapply(exposome.data, is.factor)))

# Exposome data with only numeric variables
exposome.data.nv <- exposome.data[, -factors.exposome]
exposomeNA.data.nv <- exposomeNA.data[, -factors.exposome]

# Sources of each sample
sources.nv <- sources[-factors.exposome]

# Number of variables for each source with only numeric variables
p.nv <- as.vector(table(sources.nv))
```

```
# Sources with just one variable
one.var <- which(p.nv == 1)
sources.one.var <- c()
for(i in 1:length(one.var))
  sources.one.var <- c(sources.one.var,
                       sources.nv[sum(p.nv[1:one.var[i]])])
sources.one.var
```

```
[1] "Noise" "Social and economic capital" "Tobacco Smoke"
```

```
# Only variables to near sources
sources.nv[sources.nv == "Noise"] <- "Traffic"
sources.nv[sources.nv == "Social␣and␣economic␣capital"] <- "Lifestyle"
sources.nv[sources.nv == "Tobacco␣Smoke"] <- "Lifestyle"
new.order <- order(sources.nv)

# Exposome data
exposome.data.nv <- exposome.data.nv[,new.order]
exposomeNA.data.nv <- exposomeNA.data.nv[,new.order]

# Sources of each sample
sources.nv <- sources.nv[new.order]

# Number of variables for each source with only numeric variables
p.nv <- as.vector(table(sources.nv))
```

```
# Correlogram between covariates variables and variables with
# absolute correlation greater than 0.5
# Correlation matrix
cor.matrix <- cor(exposome.data.nv)

# Cumulative sum of number of variables for each source
cum.sum.p.nv <- cumsum(p.nv)

# Covariates indexes
covariates.var <- 1:p.nv[1]

# High.correlated sources indexes
curr.index <- 1
high.correlated.cov <- list()

# Correograms of high correlated sources
for(i in 2:length(cum.sum.p.nv)){
  next.var <- (cum.sum.p.nv[i - 1] + 1):cum.sum.p.nv[i]
  # Current correlation matrix
  cor.mat <- cor.matrix[covariates.var, next.var]

  # High correlated sources
  if(length(cor.mat[abs(cor.mat) > 0.5]) > 0){
    # Correograms
    corrplot(cor.mat, method = "circle", type = "upper",
             title = paste0("Covariates␣vs␣",
                            sources.nv[cum.sum.p.nv[i - 1] + 1]),
             tl.cex = 0.5, tl.col = "black", mar = c(0,0,1,0))

    # High.correlated sources indexes
    high.correlated.cov[[curr.index]] <- next.var
    curr.index <- curr.index + 1
  }
}
```

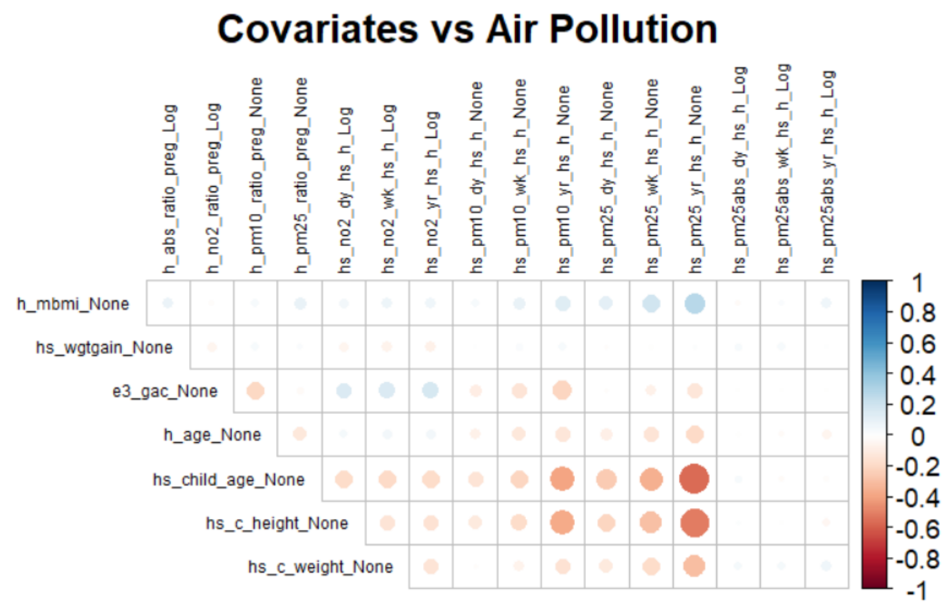


Figure A.2: Correlogram between *Covariates* variables and *Air Pollution* variables.

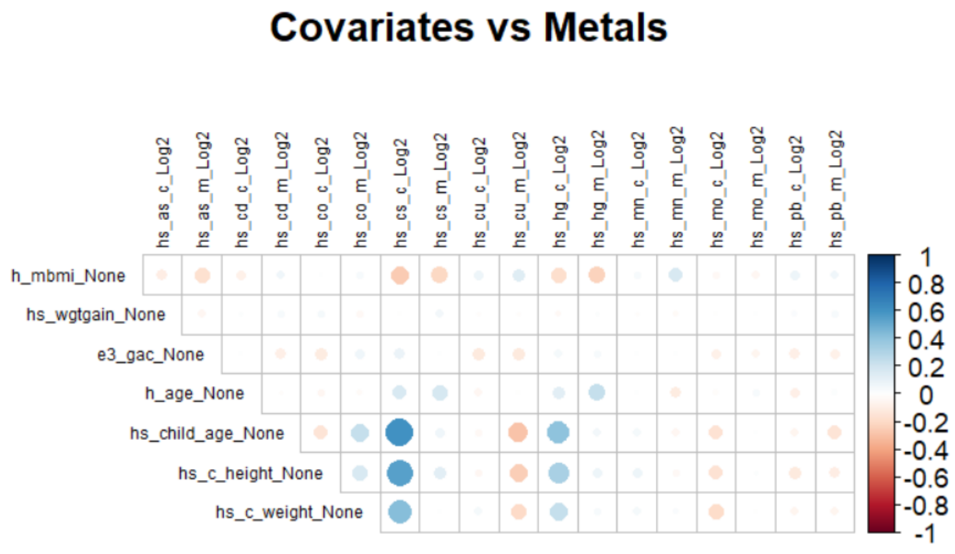


Figure A.3: Correlogram between *Covariates* variables and *Metals* variables.

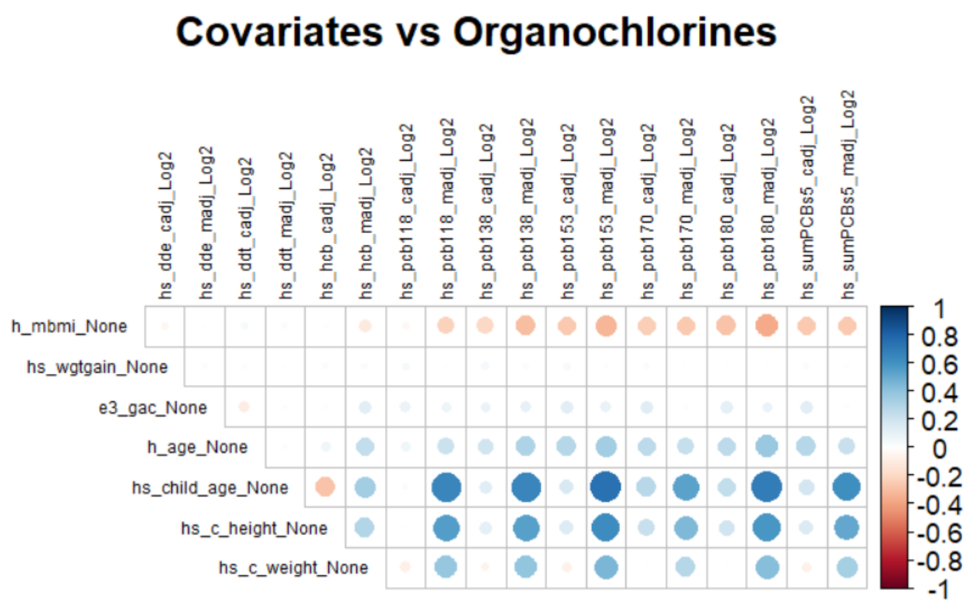


Figure A.4: Correlogram between *Covariates* variables and *Organochlorines* variables.

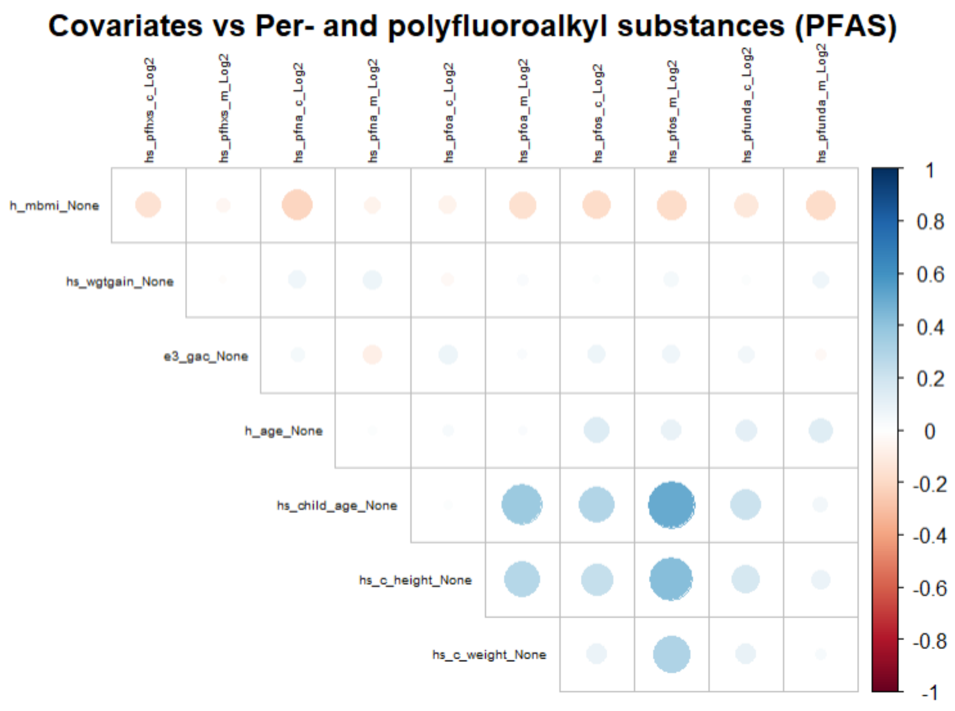


Figure A.5: Correlogram between *Covariates* variables and *PFAS* variables.

```
# Correlation matrix of covariates
cor.mat <- cor.matrix[covariates.var, covariates.var]

# Correlogram between covariates
corrplot(cor.mat, method = "circle", type = "upper",
         title = "Covariates␣vs␣Covariates",
         tl.cex = 0.5, tl.col = "black", mar = c(0,0,1,0))
```

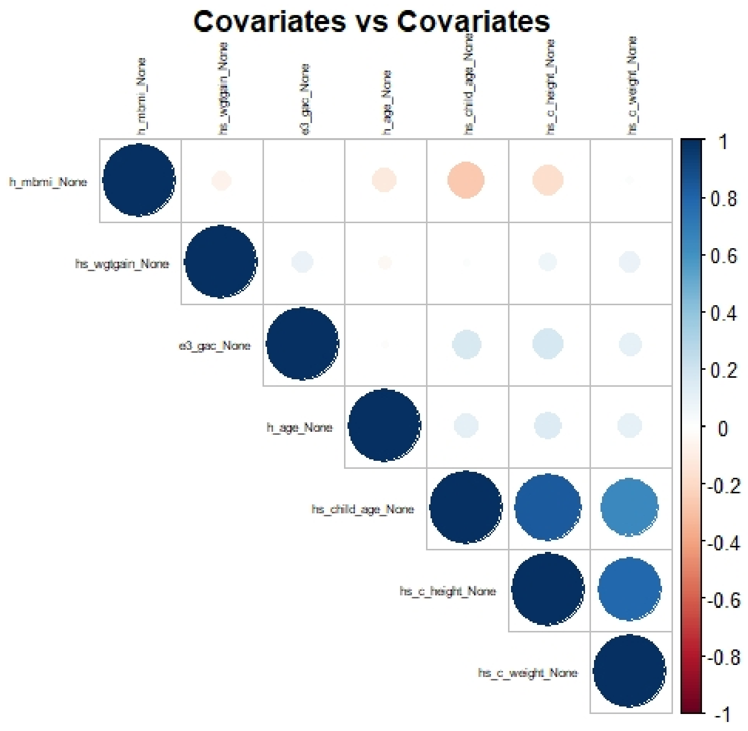


Figure A.6: Correlogram between *Covariates* variables.

```
# Correlograms betwen sources that are high correlated with covariates
for(i in 1:(length(high.correlated.cov) - 1)){
  for(j in (i + 1):length(high.correlated.cov)){
    # Current correlation matrix
    cor.mat <- cor.matrix[high.correlated.cov[[i]],
                          high.correlated.cov[[j]]]
    # Correograms
    corrplot(cor.mat, method = "circle", type = "upper",
             title = paste0(sources.nv[high.correlated.cov[[i]][1]],
                            "␣vs␣",
                            sources.nv[high.correlated.cov[[j]][1]]),
                    tl.cex = 0.5, tl.col = "black", mar = c(0,0,1,0))
  }
}
```

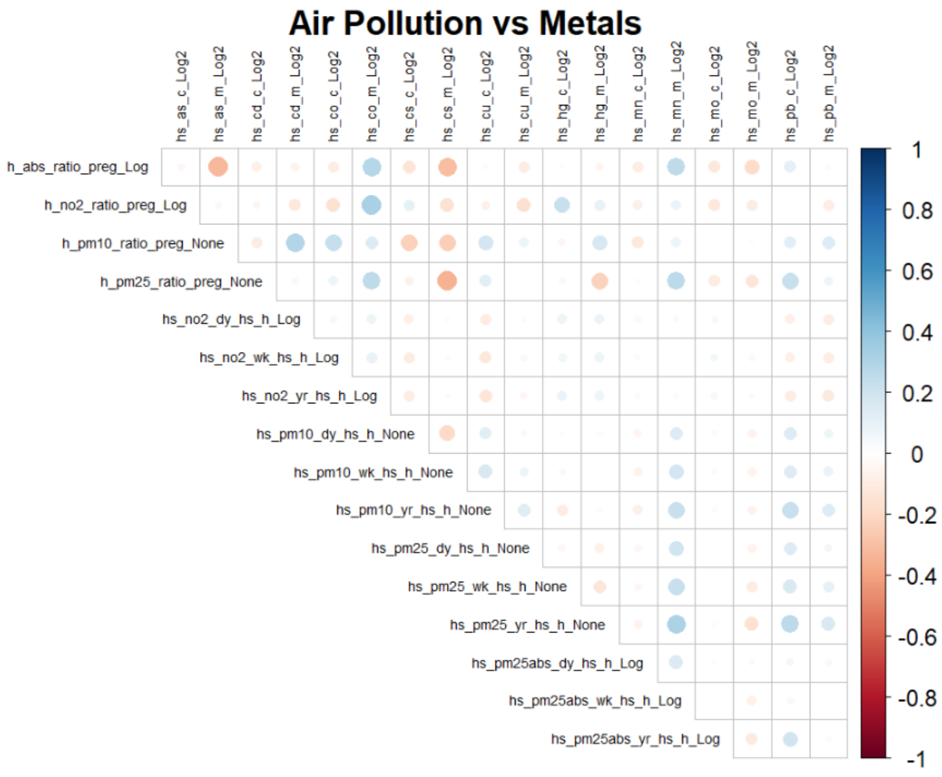


Figure A.7: Correlogram between *Air Pollution* variables and *Metals* variables.

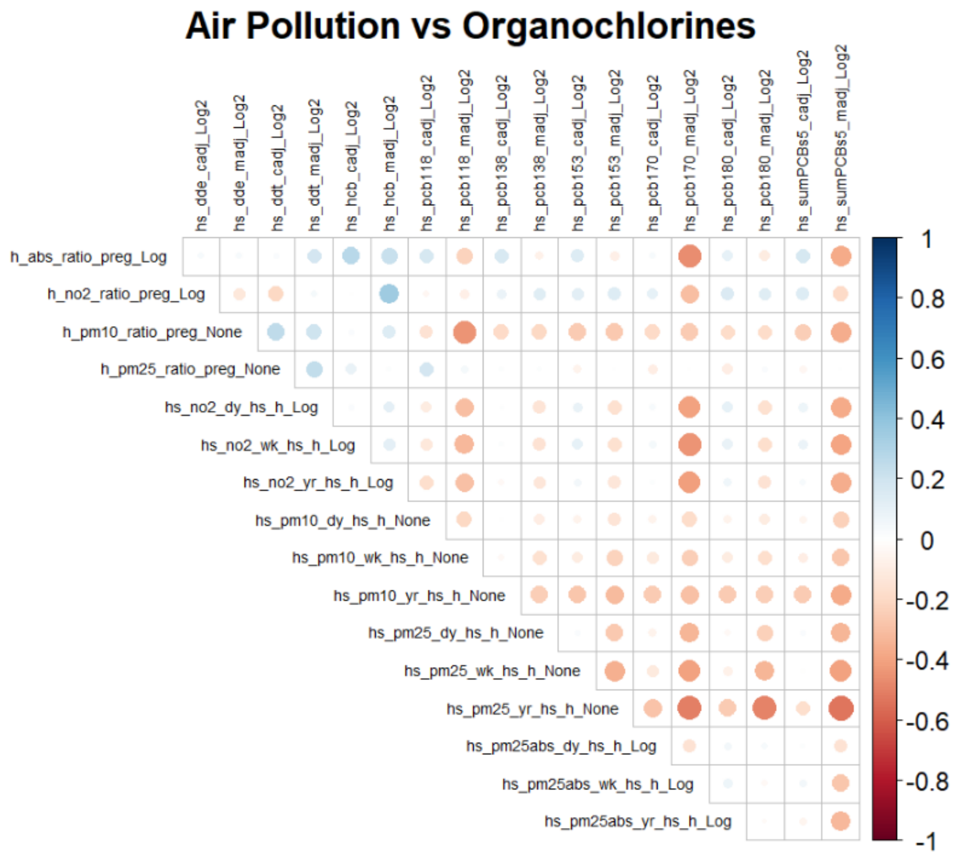


Figure A.8: Correlogram between *Air Pollution* variables and *Organochlorines* variables.

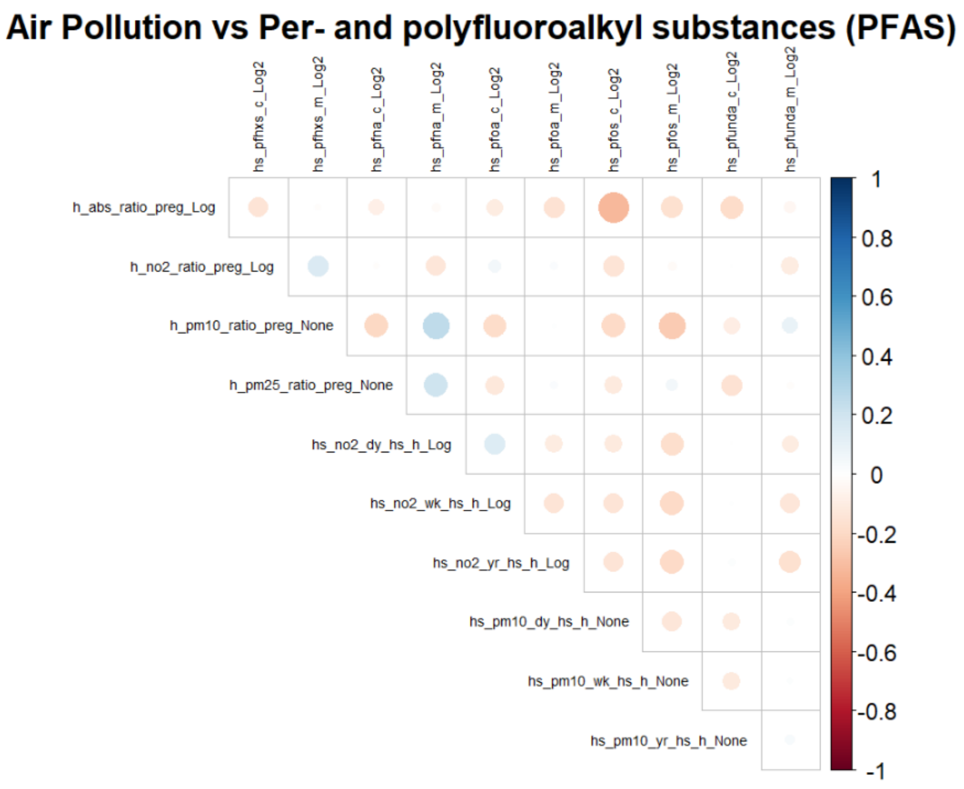


Figure A.9: Correlogram between *Air Pollution* variables and *PFAS* variables.

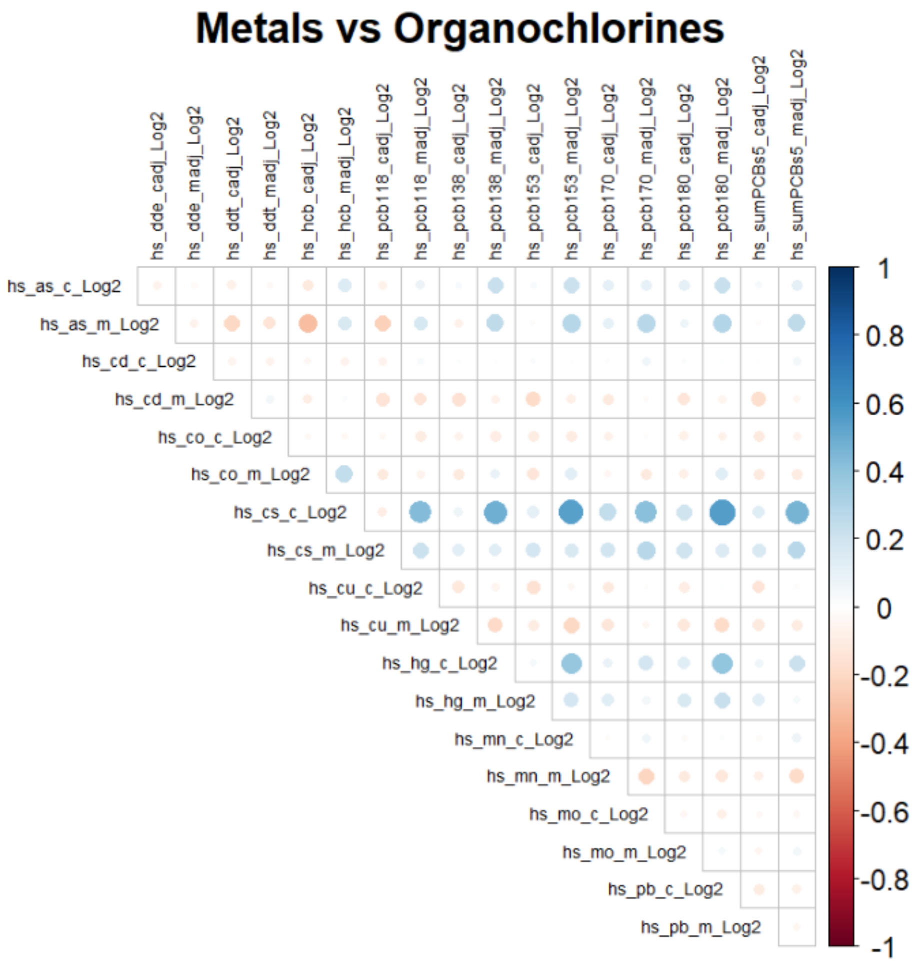


Figure A.10: Correlogram between *Metals* variables and *Organochlorines* variables.

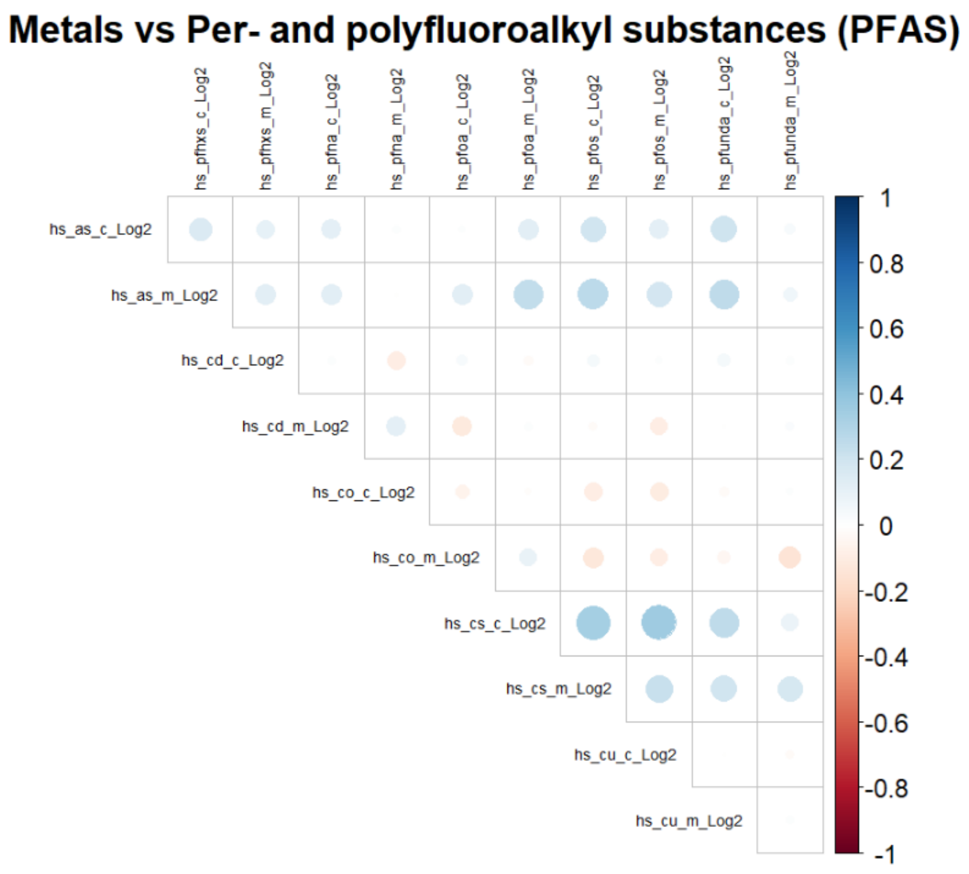


Figure A.11: Correlogram between *Metals* variables and *PFAS* variables.

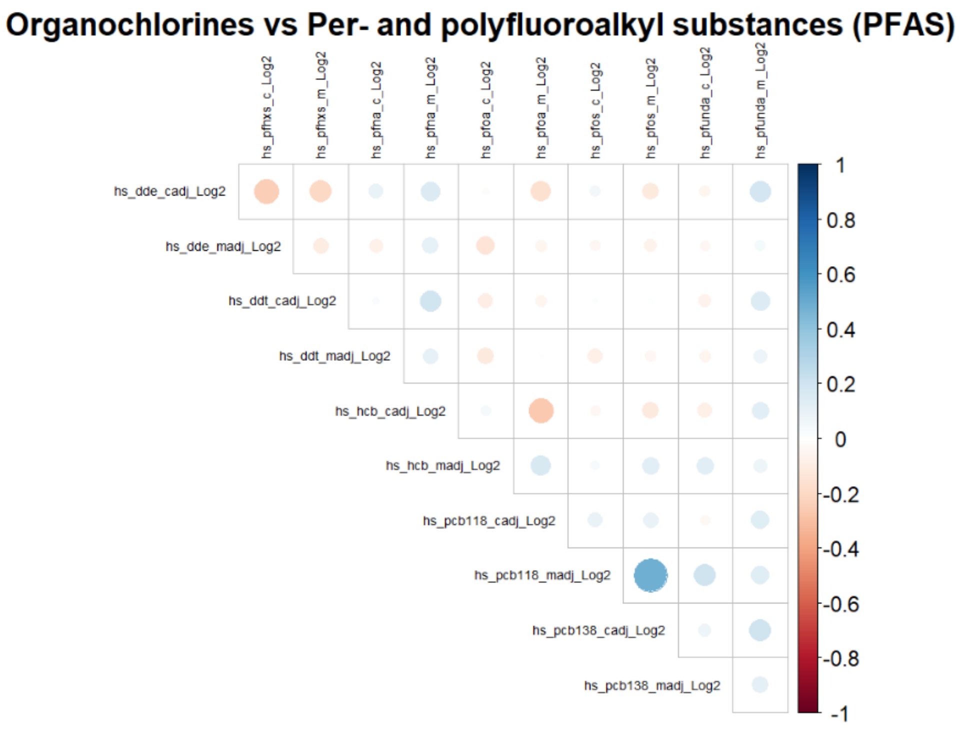


Figure A.12: Correlogram between *Organochlorines* variables and *PFAS* variables.

```
# Creating new subsources for covariates
age.cov <- c("e3_yearbir_None", "h_age_None", "e3_gac_None",
             "hs_child_age_None")
body_measures.cov <- c("h_mbmi_None", "hs_c_weight_None",
                       "hs_wgtgain_None", "hs_c_height_None")
childs.info <- c("h_native_None", "e3_sex_None")
parents.info <- c("h_cohort", "h_edumc_None", "h_parity_None")

# Dividing Covariates source into subsources for both
# numeric exposome data and the general one
colnames <- colnames(exposome.data)
colnames.nv <- colnames(exposome.data.nv)
sources[colnames %in% age.cov] <- "0.Covariates.Age"
sources.nv[colnames.nv %in% age.cov] <- "0.Covariates.Age"
sources[colnames %in% body_measures.cov]
     <- "0.Covariates.Body.Measures"
sources.nv[colnames.nv %in% body_measures.cov]
        <- "0.Covariates.Body.Measures"
sources[colnames %in% parents.info] <- "0.Covariates.Parents.Info"
sources[colnames %in% childs.info] <- "0.Covariates.Childs.Info"
```

```
# Order sources and data
order.sources <- order(sources)
order.sources.nv <- order(sources.nv)
sources <- sources[order.sources]
exposome.data <- exposome.data[, order.sources]
sources.nv <- sources.nv[order.sources.nv]
exposome.data.nv <- exposome.data.nv[, order.sources.nv]

# Number of variables for each source with only numeric variables
p.nv <- as.vector(table(sources.nv))
```

```
# Boxplot of all covariates variables
boxplot(exposome.data.nv[, covariates.var], las = 2, cex.axis = 0.5)
```

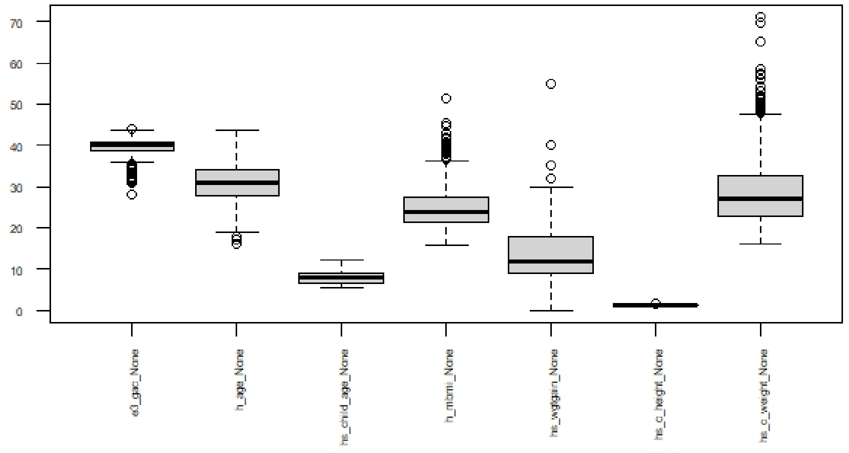


Figure A.13: Boxplot of all the *Covariates* variables.

```
# Printing outliers
outliers <- c()
covariates.var.names <- colnames(exposome.data.nv)[covariates.var]
for(i in 1:length(covariates.var)){
  out.values <-
      boxplot.stats(exposome.data.nv[, covariates.var[i]])$out
  out.samples <-
      which(exposome.data.nv[, covariates.var[i]] %in% out.values)
  if(length(out.samples) > 0){
    cat(paste0("The␣variable␣", covariates.var.names[i],
               "␣has␣the␣following␣ouliers:\n"))
    print(out.samples)
    cat("\n")

    # Outliers
    outliers <- c(outliers, out.samples)
  }
}

cat(paste0("Total␣number␣of␣outliers:␣", length(unique(outliers))))
```

```
The variable e3_gac_None has the following ouliers:
 [1]   32   62  131  167  279  335  352  383  397  425  445
 484  488  647  648  668  712  753
[19]  792  822  832  833  834  844  848  877  914  935  962
975 1098 1173 1226 1232 1281

The variable h_age_None has the following ouliers:
 [1]   78  247  273  307  345  586  594  725  851  856  962 1059 1154

The variable h_mbmi_None has the following ouliers:
 [1]   10   15   18   30   46   48   77  115  138  177  189
 203  209  225  226  255  256  285
[19]  288  297  324  406  407  410  416  461  492  504  540
569  573  574  614  615  616  626
[37]  658  705  718  726  728  751  769  864  936  940  947
973 1047 1053 1059 1074 1187 1190
[55] 1204 1275

The variable hs_wgtgain_None has the following ouliers:
[1]  225  453  530  563  721  817  917  992 1045

The variable hs_c_height_None has the following ouliers:
[1]   55  195  400  613 1285

The variable hs_c_weight_None has the following ouliers:
 [1]   12   43   79  181  285  299  407  441  453  487  608
 613  617  623  663  686  690  737
[19]  758  869  875  880  939  985  991 1020 1045 1061 1177
1182 1212 1250 1285
```

```
Total number of outliers: 142
```

```
#Asthma factor
asthma <- as.factor(y$hs_asthma)
levels(asthma) <- c("None", "Yes")

# Boxplots
for(i in 1:length(covariates.var))
  boxplot(exposome.data.nv[, covariates.var[i]] ~ asthma,
          ylab = covariates.var.names[i],
          xlab = "Asthma")
```

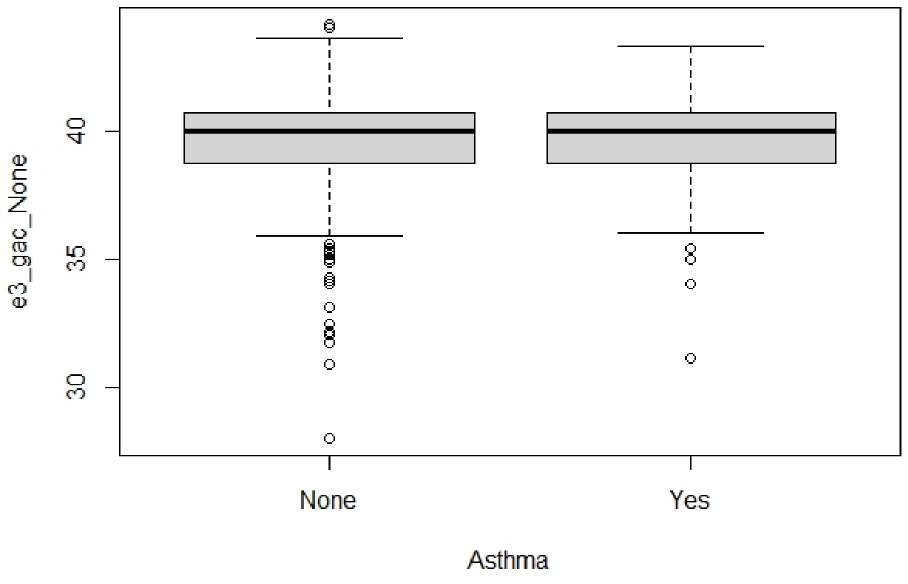


Figure A.14: Boxplot of the covariate variable *e3_gac_None* according to the factor *Asthma*.

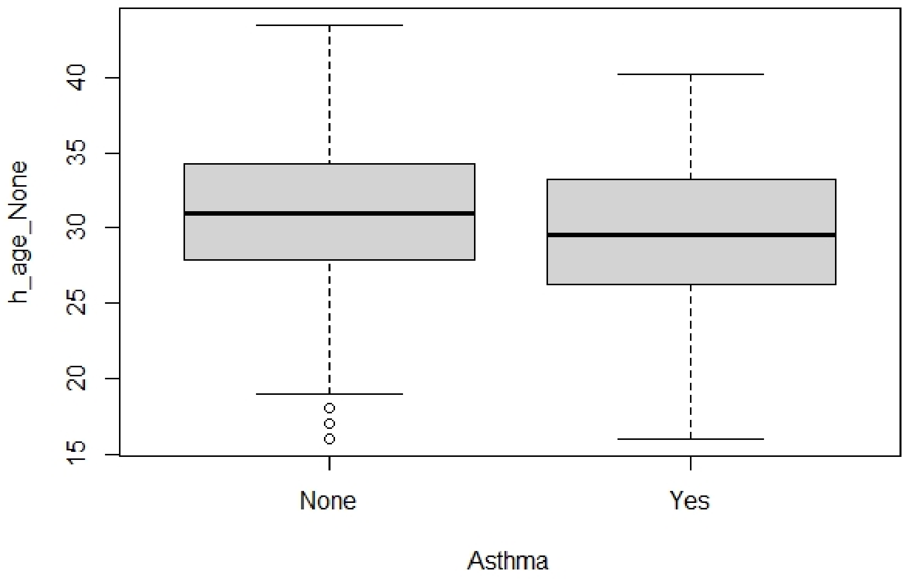


Figure A.15: Boxplot of the covariate variable *h_age_None* according to the factor *Asthma*.

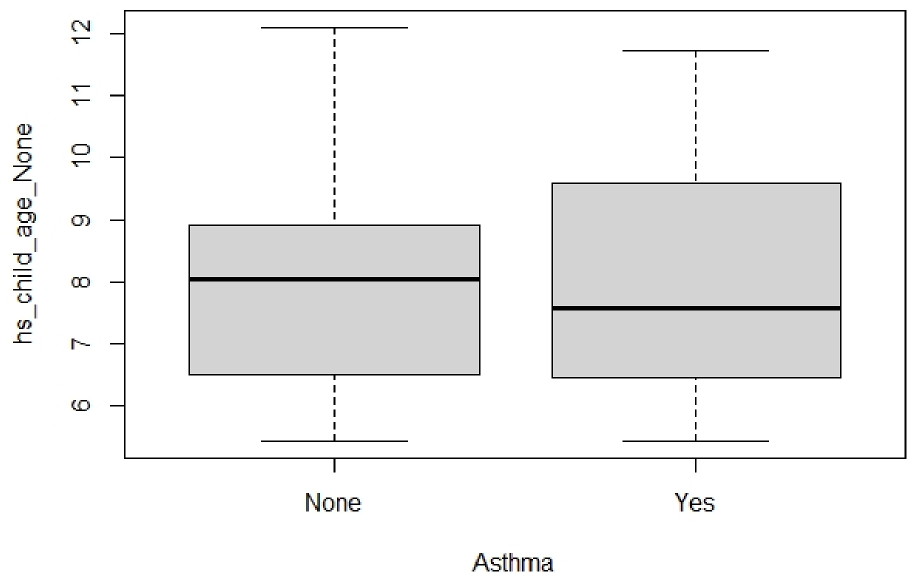


Figure A.16: Boxplot of the covariate variable *hs_child_age_None* according to the factor *Asthma*.

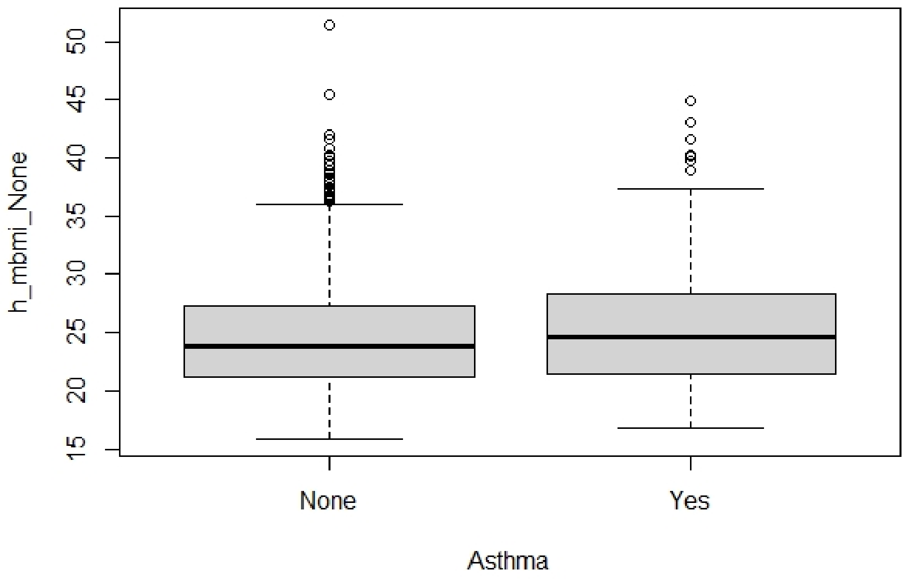


Figure A.17: Boxplot of the covariate variable *h_mbmi_None* according to the factor *Asthma*.

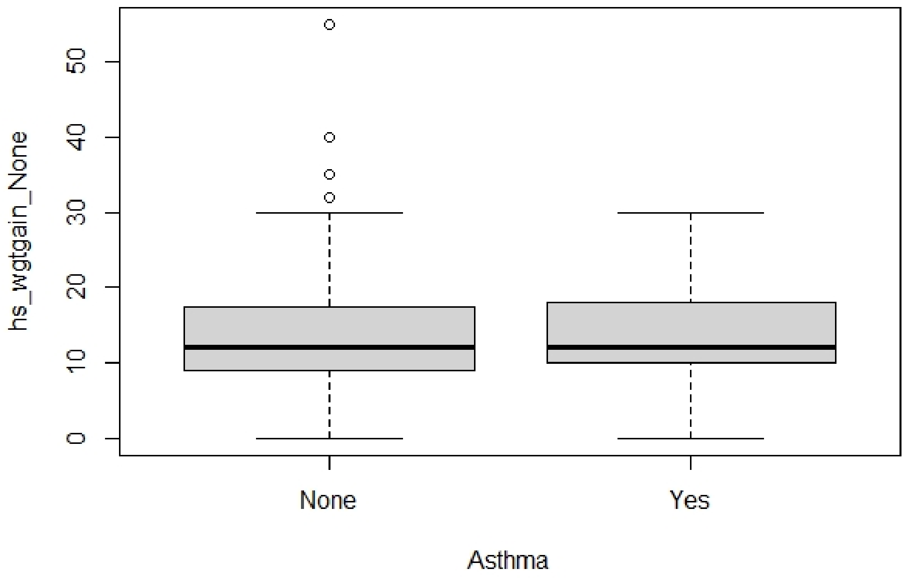


Figure A.18: Boxplot of the covariate variable *hs_wgtgain_None* according to the factor *Asthma*.

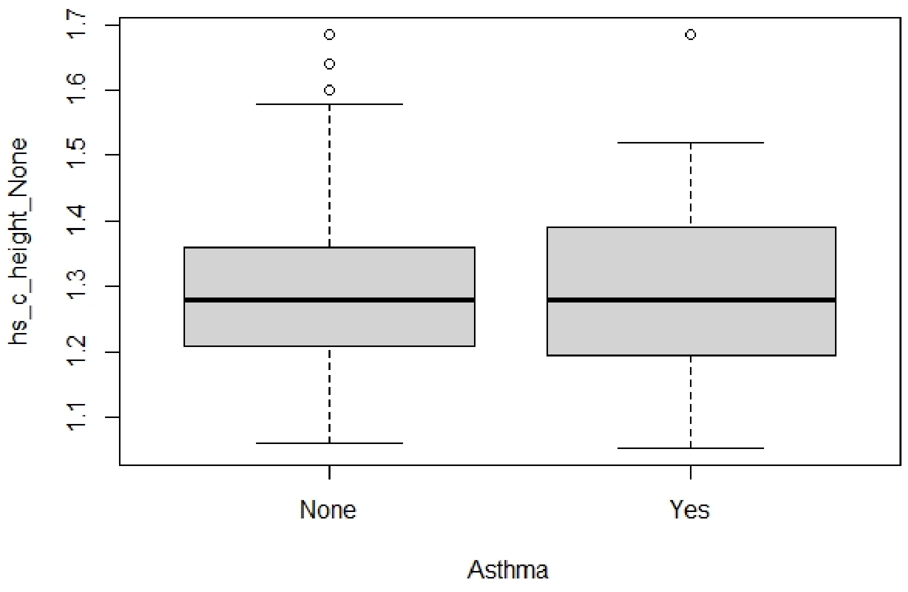


Figure A.19: Boxplot of the covariate variable *hs_c_height_None* according to the factor *Asthma*.

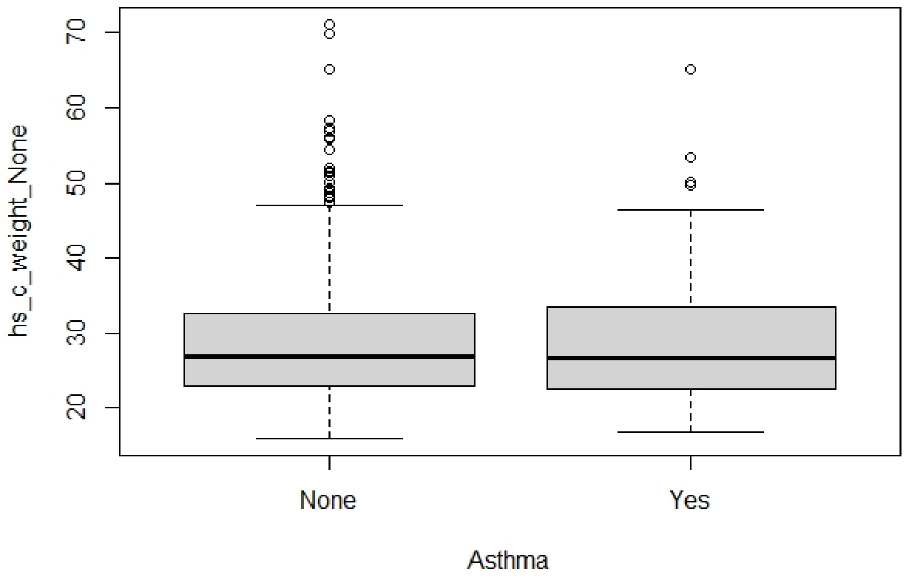


Figure A.20: Boxplot of the covariate variable *hs_c_weight_None* according to the factor *Asthma*.

```
# Principal component analysis
prin.comp <- prcomp(exposome.data.nv[, covariates.var], retx = T,
                    center = T, scale. = T)

# Percentage of variation explained for the PCA dimension
cat(paste0("PCA", 1:7, "␣␣"))
cat("\n")
cat(paste0(round(cumsum(prin.comp$sdev)/sum(prin.comp$sdev), 4)*100,
    "%"))
```

```
PCA1   PCA2   PCA3   PCA4  PCA5   PCA6  PCA7
24.97% 41.16% 56.97% 71.7% 86.19% 94.5% 100%
```

```
# Biplot two first principal components
fviz_pca_biplot(prin.comp, axes = c(1,2), xlab = "First␣Component",
                ylab = "Second␣Component", geom = c("point"),
                habillage = asthma, labelsize = 1)
```

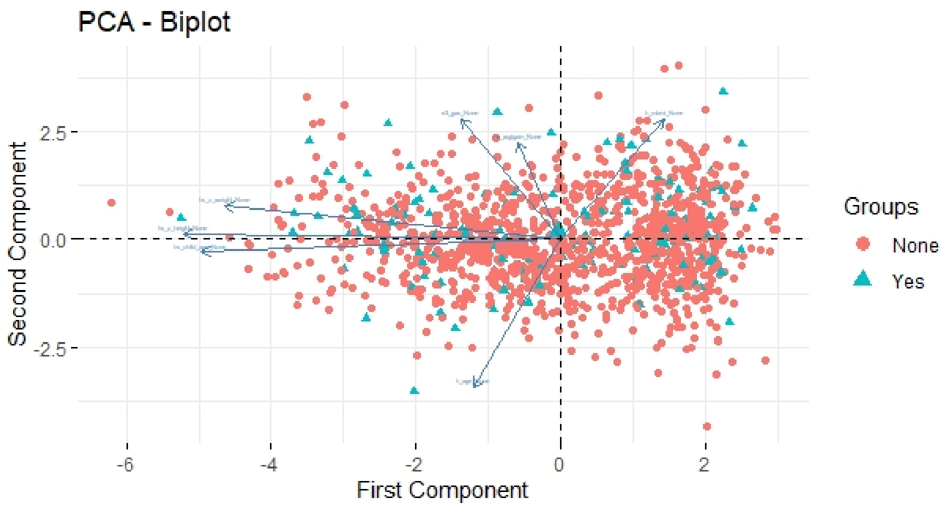


Figure A.21: Biplot of the two first principal components according to the factor *Asthma*.

- Exposome data with factor variables converted to dummy binary variables

```
# Factors on exposome data
factors.exposome <- which(as.vector(sapply(exposome.data, is.factor)))
```

```r
# Non-binary factors
non.binary.factors <- c()
for(i in 1:length(factors.exposome)){
  if(length(levels(exposome.data[, factors.exposome[i]])) > 2){
    print(table(exposome.data[, factors.exposome[i]],
                dnn = colnames(exposome.data)[factors.exposome[i]]))

    non.binary.factors <- c(non.binary.factors, factors.exposome[i])
  }
}
```

```
e3_yearbir_None
2003 2004 2005 2006 2007 2008 2009
  55  107  241  256  250  379   13
h_native_None
   0    1    2
 146   67 1088
h_cohort
  1   2   3   4   5   6
202 198 224 207 272 198
h_edumc_None
  1   2   3
178 449 674
h_parity_None
  0   1   2
601 464 236
h_bfdur_Ter
   (0,10.8] (10.8,34.9]  (34.9,Inf]
        506         270         525
h_cereal_preg_Ter
     (0,9]   (9,27.3] (27.3,Inf]
       531        459        311
h_dairy_preg_Ter
   (0,17.1] (17.1,27.1]  (27.1,Inf]
        270         380         651
h_fastfood_preg_Ter
   (0,0.25] (0.25,0.83]  (0.83,Inf]
         94         535         672
h_fish_preg_Ter
  (0,1.9] (1.9,4.1] (4.1,Inf]
      343       490       468
h_fruit_preg_Ter
   (0,0.6] (0.6,18.2] (18.2,Inf]
         6        922        373
h_legume_preg_Ter
(0,0.5] (0.5,2] (2,Inf]
    245     269     787
h_meat_preg_Ter
 (0,6.5] (6.5,10] (10,Inf]
     427      387      487
h_pamod_t3_None
      None      Often  Sometimes Very Often
        42        474        191        594
h_pavig_t3_None
  High    Low Medium
    47    952    302
h_veg_preg_Ter
   (0,8.8] (8.8,16.5] (16.5,Inf]
       539        470        292
hs_bakery_prod_Ter
  (0,2]   (2,6] (6,Inf]
    345     423     533
hs_beverages_Ter
(0,0.132] (0.132,1]   (1,Inf]
      331       454       516
hs_break_cer_Ter
  (0,1.1] (1.1,5.5] (5.5,Inf]
      291       521       489
hs_dairy_Ter
   (0,14.6] (14.6,25.6]  (25.6,Inf]
        359         465         477
hs_fastfood_Ter
  (0,0.132] (0.132,0.5]   (0.5,Inf]
        143         603         555
hs_org_food_Ter
(0,0.132] (0.132,1]   (1,Inf]
      429       396       476
hs_proc_meat_Ter
(0,1.5] (1.5,4] (4,Inf]
    366     471     464
hs_readymade_Ter
  (0,0.132] (0.132,0.5]   (0.5,Inf]
        327         296         678
hs_total_bread_Ter
     (0,7]   (7,17.5] (17.5,Inf]
       431        381        489
```

```
hs_total_cereal_Ter
   (0,14.1] (14.1,23.6]  (23.6,Inf]
        418         442         441
hs_total_fish_Ter
(0,1.5] (1.5,3] (3,Inf]
    389     454     458
hs_total_fruits_Ter
     (0,7]   (7,14.1] (14.1,Inf]
       413        407        481
hs_total_lipids_Ter
  (0,3]   (3,7] (7,Inf]
    397     403     501
hs_total_meat_Ter
  (0,6]   (6,9] (9,Inf]
    425     411     465
hs_total_potatoes_Ter
  (0,3]   (3,4] (4,Inf]
    417     405     479
hs_total_sweets_Ter
  (0,4.1] (4.1,8.5] (8.5,Inf]
      344       516       441
hs_total_veg_Ter
    (0,6]   (6,8.5] (8.5,Inf]
      404       314       583
hs_total_yog_Ter
    (0,6]   (6,8.5] (8.5,Inf]
      779       308       214
hs_ln_cat_h_None
  1   2   3   4   5
476 633 104  61  27
hs_lden_cat_s_None
  1   2   3   4   5   6
580 265 299 104  37  16
FAS_cat_None
   Low Middle   High
   146    486    669
hs_contactfam_3cat_num_None
       (almost) Daily           Once a week Less than once a week
                  863                   382                    56
hs_participation_3cat_None
                   None          1 organisation 2 or more organisations
                    748                     355                     198
hs_cotinine_mcat_None
```

```
Non-smokers SHS smokers     Smokers
        759         157         385
hs_smk_parents_None
   both neither     one
    142     814     345
```

```
# Three levels factors to binary
for(i in 1:length(non.binary.factors)){
  factor <- exposome.data[, non.binary.factors[i]]
  levels <- levels(factor)
  if(length(levels) == 3){
    sum1 <- sum(factor %in% levels[1:2])
    sum2 <- sum(factor %in% levels[2:3])
    if(sum1 < sum2){
      levels(exposome.data[, non.binary.factors[i]])[1:2] <-
        paste0(levels[1], ",␣", levels[2])
    } else {
      levels(exposome.data[, non.binary.factors[i]])[2:3] <-
        paste0(levels[2], ",␣", levels[3])
    }
  }
}
```

```
# More than three levels factors to binary
# h_cohort
levels <- levels(exposome.data$h_cohort)
levels(exposome.data$h_cohort)[levels %in% c(4, 5, 6)] <- "4,␣5,␣6"
levels(exposome.data$h_cohort)[levels %in% c(1, 2, 3)] <- "1,␣2,␣3"

# e3_yearbir_None
levels <- levels(exposome.data$e3_yearbir_None)
levels(exposome.data$e3_yearbir_None)[levels
       %in% c(2007, 2008, 2009)] <-
  "2007,␣2008,␣2009"
levels(exposome.data$e3_yearbir_None)[levels
       %in% c(2003, 2004, 2005, 2006)] <-
  "2003,␣2004,␣2005,␣2006"

# h_pamod_t3_None
levels <- levels(exposome.data$h_pamod_t3_None)
levels(exposome.data$h_pamod_t3_None)[levels %in%
       c("None", "Often", "Sometimes")] <- "Non␣Very␣Often"
```

```r
# hs_ln_cat_h_None
levels <- levels(exposome.data$hs_ln_cat_h_None)
levels(exposome.data$hs_ln_cat_h_None)[levels %in% c(1, 3, 4, 5)] <-
  "1,␣3,␣4,␣5"

# hs_lden_cat_s_None
levels <- levels(exposome.data$hs_lden_cat_s_None)
levels(exposome.data$hs_lden_cat_s_None)[levels
       %in% c(2, 3, 4, 5, 6)] <-
  "2,␣3,␣4,␣5,␣6"
```

```r
# Exposome data with factors being dummy variables
exposome.data.dv <- exposome.data
exposomeNA.data.dv <- exposomeNA.data

# Sources with factors being dummy variables
sources.dv <- sources

# Change a factor for a dummy variable in data
update.factor.to.dummy <- function(data, factor.index){
  # Factor variable
  variable <- data[, factor.index]

  dummy.variable <- acm.disjonctif(data.frame(variable))
  if(any(is.na(variable))){
    NA.samples <- which(is.na(variable))
    dummy.variable[NA.samples, ] <- rep(NA, length(dummy.variable))
  }

  if(factor.index > 1)
    data <- data.frame(data[, 1:(factor.index - 1)],
                       dummy.variable,
                       data[, (factor.index + 1):length(data)])
  else
    data <- data.frame(dummy.variable,
                       data[, (factor.index + 1):length(data)])

  return(data)
}

for(i in length(factors.exposome):1){
  # Factor to convert to dummy
  factor.exposome <- factors.exposome[i]
  # Updated sources with dummy variables
  sources.dv <-
    c(sources.dv[1:factor.exposome],
      rep(sources.dv[factor.exposome],
          length(levels(exposome.data.dv[, factor.exposome])) - 1),
      sources.dv[(factor.exposome + 1):
                   length(sources.dv)])

  # Updated exposome data with dummy variables
  exposome.data.dv <- update.factor.to.dummy(exposome.data.dv,
                                             factor.exposome)
  exposomeNA.data.dv <- update.factor.to.dummy(exposomeNA.data.dv,
                                               factor.exposome)
}

# Number of variables for each source with factors
# being dummy variable
p.dv <- as.vector(table(sources.dv))
```

Appendix B

Code: an incomplete source feature selection (iSFS) model

B.1 iSFS model for the least square loss function

B.1.1 Algorithm of the iSFS model for the least square loss function

```r
# iSFS algorithm
iSFS <- function(p, X, y, lambda, L.step = 1.5, maxIter.iSFS = 300,
                 tol.iSFS = 1e-12, omega.alpha = "LR", tol.alpha
                 = 1e-12, maxIter.alpha = 20, omega.beta = "LR",
                 beta0.comp = "LMR", tol.beta = 1e-12,
                 maxIter.beta = 20, gamma = 1, to.normalize = F,
                 beta0, alpha0){
  # Initializes the progress bar
  pb <- txtProgressBar(min = 0, # Minimum value of the progress bar
           max = maxIter.iSFS*length(lambda), # Maximum value of
                                              # the progress bar
           style = 3,    # Progress bar style
           width = 50,   # Progress bar width
           char = "=")   # Character used to create the bar

  # L.step factor definition
  L.step <- max(1.001, L.step)

  # Features
  X <- as.matrix(X)
  translation <- c()
  scale <- c()
  if(to.normalize){
    for(j in 1:dim(X)[2]){
      x <- X[, j]
      x <- x[!is.na(x)]
      min.x <- min(x)
      max.x <- max(x)
      translation <- c(translation, min.x)
      scale <- c(scale, max.x - min.x)
      X[, j] <- (X[, j] - translation[j])/scale[j]
    }
  }

  # Outcome
  if(is.factor(y))
    y <- as.numeric(as.character(y))

  # Number of sources
  S <- length(p)

  # We compute the profiles
  pf.vec <- get_profile(p, X)

  # If it is complete data, alpha weights are fixed
  keep.alpha <- length(levels(pf.vec)) == 1

  # Best alpha, beta and lambda parameters
  if(missing(alpha0))
    best.alpha <- alpha.initialization(pf.vec, S, keep.alpha)
  else if(is.list(alpha0)) best.alpha <- alpha0
       else best.alpha <- as.list(alpha0)
  if(missing(beta0))
    best.beta <- beta.initialization(p, X, y, beta0.comp)
  else if(is.list(beta0)) best.beta <- beta0
       else best.beta <- as.list(beta0)
  best.lambda <- NA

  # Best objective function value
  obj.func.best <- objective.fun(p, X, y, best.beta, best.alpha,
                               pf.vec)
  for(j in 1:length(lambda)){
    # Initial objective function value
    obj.func0 <- obj.func.best
    # We initialize alpha0 weights
    alpha0 <- best.alpha
    # We initialize beta0 models
    beta0 <- best.beta
    # If alpha is always fixed
    if(keep.alpha){
      # We compute the optimal beta
      for(k in 1:maxIter.iSFS){
        # Computing beta when alpha is fixed
        beta <- prox.grad.iter.method(p, X, y, alpha0, beta0, pf.vec,
                                      lambda[j], omega.beta, L.step,
                                      maxIter.beta, tol.beta, gamma)

        # Objective function computation
        obj.func <- objective.fun(p, X, y, beta, alpha0, pf.vec)
        # If the objective stops decreasing, we stop computing
        if(abs(obj.func - obj.func0) < tol.iSFS){
          if(obj.func < obj.func0){
            # We update the beta vector
            beta0 <- beta
            # and the objective function value
            obj.func0 <- obj.func
          }

          break;
        }

        # Otherwise, we update the beta vector
        beta0 <- beta
        # and the objective function value
        obj.func0 <- obj.func

        # Sets the progress bar to the current state
        setTxtProgressBar(pb, k + (j - 1)*maxIter.iSFS)
      }
    } else {
      # We compute the optimal alpha and beta
      for(k in 1:maxIter.iSFS){
        # Computing alpha when beta is fixed
        alpha <- alpha.compute(p, X, y, beta0, alpha0, pf.vec,
                               omega.alpha, L.step, maxIter.alpha,
                               tol.alpha)

        # Computing beta when alpha is fixed
        beta <- prox.grad.iter.method(p, X, y, alpha, beta0, pf.vec,
                                      lambda[j], omega.beta, L.step,
                                      maxIter.beta, tol.beta, gamma)

        # Objective function computation
        obj.func <- objective.fun(p, X, y, beta, alpha, pf.vec)
        # If the objective stops decreasing, we stop computing
        if(abs(obj.func - obj.func0) < tol.iSFS){
          if(obj.func < obj.func0){
            # We update both alpha and beta vectors
            beta0 <- beta
            alpha0 <- alpha
            # and the objective function value
            obj.func0 <- obj.func
          }

          break;
        }

        # Otherwise, we update both alpha and beta vectors
        beta0 <- beta
        alpha0 <- alpha
        # and the objective function value
        obj.func0 <- obj.func

        # Sets the progress bar to the current state
        setTxtProgressBar(pb, k + (j - 1)*maxIter.iSFS)
      }
    }

    # Get best parameters
    if(obj.func0 < obj.func.best){
      best.beta <- beta0
      best.alpha <- alpha0
      best.lambda <- lambda[j]
      obj.func.best <- obj.func0
    }
  }

  # Ending progress bar
  setTxtProgressBar(pb, maxIter.iSFS*length(lambda))

  # Final coefficients
  return(list(alpha = best.alpha, beta = best.beta,
              lambda = best.lambda, profile.vector = pf.vec,
              to.normalize = to.normalize, translation = translation,
              scale = scale))
}
```

B.1.2 Predictions on the iSFS algorithm

```
# Predictions of the iSFS model
predict.iSFS <- function(iSFS.model, X, p){
  # Features as matrix
  X <- as.matrix(X)
  if(iSFS.model$to.normalize)
    for(j in 1:dim(X)[2])
      X[, j] <- (X[, j] - iSFS.model$translation[j])/
                 iSFS.model$scale[j]

  # Samples and sources
  n <- dim(X)[1]
  S <- length(p)

  # Profiles of data to predict
  pf.vec.pred <- get_profile(p, X)
  pf.vec.pred <- as.numeric(levels(pf.vec.pred))[pf.vec.pred]

  # Predicted outcome
  y.pred <- numeric(length = n)
  for(i in 1:n){
    # Profile m of sample i
    m <- pf.vec.pred[i]

    # Block sample for profile
    model.profile.index <- which(levels(iSFS.model$profile.vector)
                                 == m)
    if(length(model.profile.index) == 0)
      y.pred[i] <- NA
    else {
      sources.profile <- which(as.binary(m, n = S))
      model.profile.index <- as.integer(model.profile.index[1])
      col <- 1
      for(j in 1:S){
        nextCol <- col + p[j] - 1
        if(j %in% sources.profile)
          y.pred[i] <- y.pred[i] +
            iSFS.model$alpha[[model.profile.index]][j]*
            X[i, col:nextCol]%*%iSFS.model$beta[col:nextCol]
        col <- nextCol + 1
      }
    }
  }
  return(y.pred)
}
```

Appendix C

Code, figures and tables: discussion and applications of the iSFS model on simulated and exposome data

```
# Evaluation values for the iSFS model
evaluation.model.param <- function(y.test, y.pred, n.vars = 0){
  # Convert factor to numeric
  if(is.factor(y.test))
    y.test <- as.numeric(as.character(y.test))

  # Number of samples
  n <- length(y.test)

  # Error term (y - predictions)
  error <- y.test - y.pred

  # Compute mean square error
  mean.sq.error <- sum(error^2)/n

  # Compute root mean square error
  root.mean.sq.error <- sqrt(mean.sq.error)

  # Compute mean absolute error
  mean.abs.error <- sum(abs(error))/n

  # Compute root mean absolute error
  root.mean.abs.error <- sqrt(mean.abs.error)

  # Compute R squared
  SS.res <- sum(error^2)
  mean.y <- mean(y.test)
  SS.tot <- sum((y.test - mean.y)^2)
  R.squared <- 1 - SS.res/SS.tot

  # Compute adjusted R squared
  adj.R.squared <- 1 - (SS.res*(n - 1))/(SS.tot*(n - n.vars - 1))

  # Evaluation parameters
  evaluation_param <- data.frame(mean.sq.error, root.mean.sq.error,
                                 mean.abs.error, root.mean.abs.error,
                                 R.squared, adj.R.squared)
  colnames(evaluation_param) <- c("MSE", "RMSE", "MAE", "RMAE",
                                  "R␣squared", "Adjusted␣R␣squared")

  # Table with evaluation parameters
  knitr::kable(evaluation_param, format = "simple", caption =
               "Evaluation␣values␣for␣iSFS␣model␣predictions.",
               align = rep('c', 6))

  return(evaluation_param)
}
```

C.1 Simulated data

```
# Data sets separated in training 67\% and test (33%)
# We select the indices that we will use for training
indexes_partition <- createDataPartition(y = 1:dim(X_nc)[1],
                                         p = prob_train, list = FALSE)

# Data matrix non correlation
X_nc_train <- X_nc[indexes_partition, ]
X_nc_test <- X_nc[-indexes_partition, ]
X.NA_nc_train <- X.NA_nc[indexes_partition, ]
X.NA_nc_test <- X.NA_nc[-indexes_partition, ]

# Data matrix low correlation
X_lc_train <- X_lc[indexes_partition, ]
X_lc_test <- X_lc[-indexes_partition, ]
X.NA_lc_train <- X.NA_lc[indexes_partition, ]
X.NA_lc_test <- X.NA_lc[-indexes_partition, ]

# Data matrix high correlation
```

```
X_hc_train <- X_hc[indexes_partition, ]
X_hc_test <- X_hc[-indexes_partition, ]
X.NA_hc_train <- X.NA_hc[indexes_partition, ]
X.NA_hc_test <- X.NA_hc[-indexes_partition, ]

# Outcome non correlation
y_nc_train <- y_nc[indexes_partition]
y_nc_test <- y_nc[-indexes_partition]

# Outcome low correlation
y_lc_train <- y_lc[indexes_partition]
y_lc_test <- y_lc[-indexes_partition]

# Outcome high correlation
y_hc_train <- y_hc[indexes_partition]
y_hc_test <- y_hc[-indexes_partition]
```

C.1.1 Comparison on complete data

- Non-correlated data

```
iSFS.Model_nc <- iSFS(p = p.synth, X = X_nc_train, y = y_nc_train,
                      lambda = 0.00000005, L.step = 10, maxIter.iSFS
                      = 100, maxIter.alpha = 20, maxIter.beta = 50)

y_nc.pred_train <- predict.iSFS(iSFS.Model_nc, X_nc_train, p.synth)
evaluation.model.param(y_nc_train, y_nc.pred_train, sum(p.synth))

y_nc.pred_test <- predict.iSFS(iSFS.Model_nc, X_nc_test, p.synth)
evaluation.model.param(y_nc_test, y_nc.pred_test, sum(p.synth))

plot(y_nc_train, y_nc.pred_train)
abline(a = 0, b = 1)

plot(y_nc_test, y_nc.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.2058177	0.4536713	0.3618598	0.6015478	0.9986472	0.9982907

Table C.1: Evaluation values for the model when used complete non-correlated synthetic training data.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.2821816	0.5312077	0.425397	0.6522247	0.9983073	0.9970423

Table C.2: Evaluation values for the model when used complete non-correlated synthetic testing data.

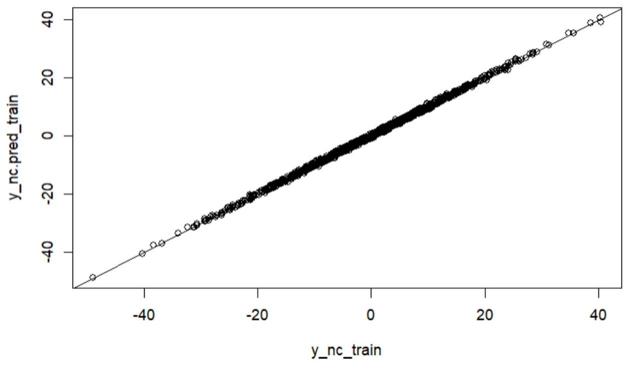


Figure C.1: Predicted training outcome vs real training outcome for complete non-correlated synthetic data.

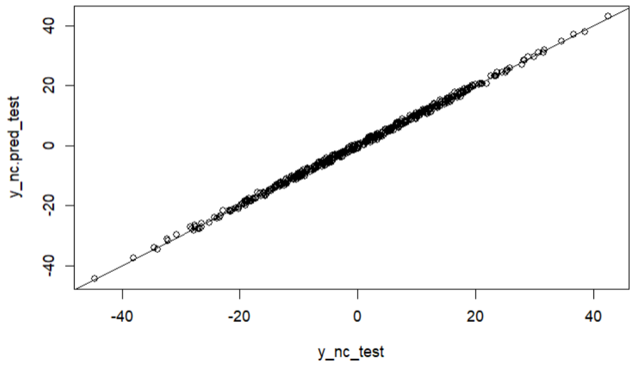


Figure C.2: Predicted testing outcome vs real testing outcome for complete non-correlated synthetic data.

```
# Non-relevant features
which(abs(iSFS.Model_nc$beta) < 0.0001)
```

```
[1] 166
```

- Low-correlated data

```
iSFS.Model_lc <- iSFS(p = p.synth, X = X_lc_train, y = y_lc_train,
                      lambda = 0.00000005, L.step = 10, maxIter.iSFS
                      = 100, maxIter.alpha = 20, maxIter.beta = 50)

y_lc.pred_train <- predict.iSFS(iSFS.Model_lc, X_lc_train, p.synth)
evaluation.model.param(y_lc_train, y_lc.pred_train, sum(p.synth))

y_lc.pred_test <- predict.iSFS(iSFS.Model_lc, X_lc_test, p.synth)
evaluation.model.param(y_lc_test, y_lc.pred_test, sum(p.synth))

plot(y_lc_train, y_lc.pred_train)
abline(a = 0, b = 1)

plot(y_lc_test, y_lc.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.2018781	0.4493085	0.3581987	0.598497	0.9980928	0.9975903

Table C.3: Evaluation values for the model when used complete low-correlated synthetic training data.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.2928369	0.5411441	0.4351202	0.6596364	0.9975627	0.9957413

Table C.4: Evaluation values for the model when used complete low-correlated synthetic testing data.

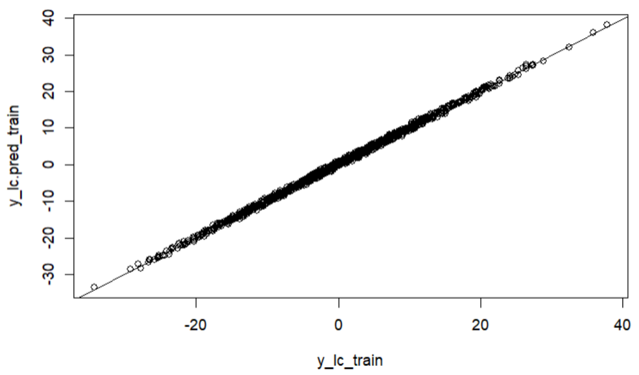


Figure C.3: Predicted training outcome vs real training outcome for complete low-correlated synthetic data.

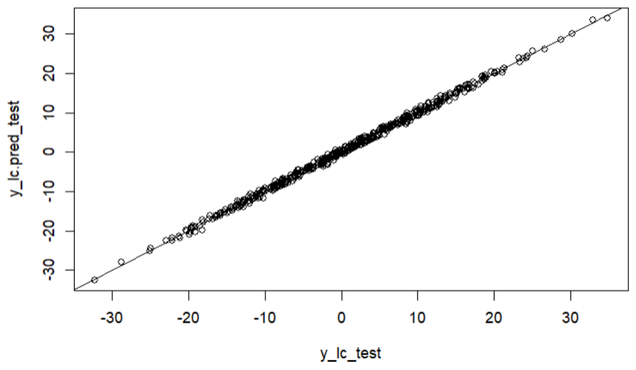


Figure C.4: Predicted testing outcome vs real testing outcome for complete low-correlated synthetic data.

```
# Non-relevant features
which(abs(iSFS.Model_lc$beta) < 0.0001)
```

```
integer(0)
```

- High-correlated data

```
iSFS.Model_hc <- iSFS(p = p.synth, X = X_hc_train, y = y_hc_train,
                      lambda = 0.00000005, L.step = 10, maxIter.iSFS
                      = 100, maxIter.alpha = 20, maxIter.beta = 50)

y_hc.pred_train <- predict.iSFS(iSFS.Model_hc, X_hc_train, p.synth)
evaluation.model.param(y_hc_train, y_hc.pred_train, sum(p.synth))

y_hc.pred_test <- predict.iSFS(iSFS.Model_hc, X_hc_test, p.synth)
evaluation.model.param(y_hc_test, y_hc.pred_test, sum(p.synth))

plot(y_hc_train, y_hc.pred_train)
abline(a = 0, b = 1)

plot(y_hc_test, y_hc.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.2060892	0.4539705	0.3575158	0.5979262	0.9907714	0.9883398

Table C.5: Evaluation values for the model when used complete high-correlated synthetic training data.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.3114668	0.5580921	0.4437904	0.6661759	0.9862592	0.9759902

Table C.6: Evaluation values for the model when used complete high-correlated synthetic testing data.

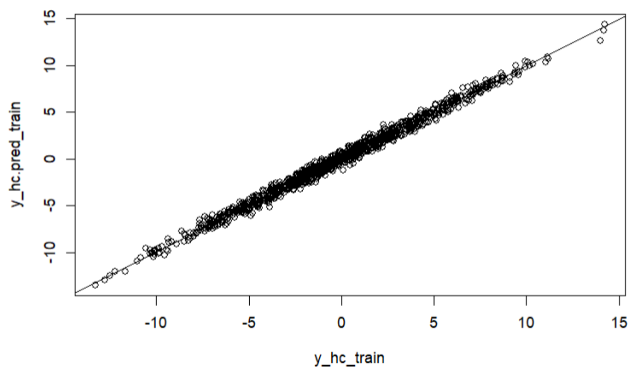


Figure C.5: Predicted training outcome vs real training outcome for complete high-correlated synthetic data.

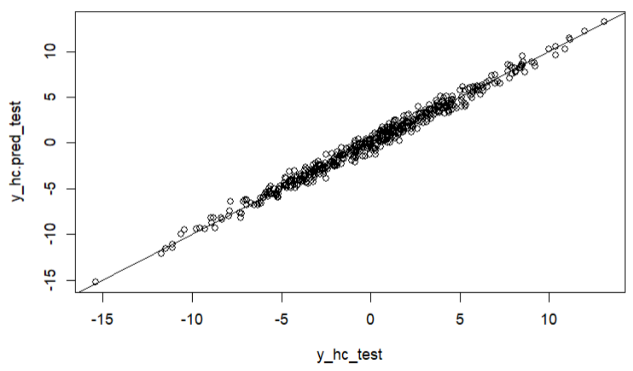


Figure C.6: Predicted testing outcome vs real testing outcome for complete high-correlated synthetic data.

```
# Non-relevant features
which(abs(iSFS.Model_hc$beta) < 0.0001)
```

```
[1] 172
```

C.1.2 Comparison on incomplete data

- Non-correlated data

```
iSFS.ModelNA_nc <- iSFS(p = p.synth, X = X.NA_nc_train,
                        y = y_nc_train, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 20,
                        maxIter.alpha = 20, maxIter.beta = 20)

yNA_nc.pred_train <- predict.iSFS(iSFS.ModelNA_nc, X.NA_nc_train,
                                  p.synth)
evaluation.model.param(y_nc_train, yNA_nc.pred_train, sum(p.synth))

yNA_nc.pred_test <- predict.iSFS(iSFS.ModelNA_nc, X.NA_nc_test,
                                 p.synth)
evaluation.model.param(y_nc_test, yNA_nc.pred_test, sum(p.synth))

plot(y_nc_train, yNA_nc.pred_train)
abline(a = 0, b = 1)

plot(y_nc_test, yNA_nc.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
15.29041	3.910295	2.863431	1.692168	0.8994971	0.8730157

Table C.7: Evaluation values for the model when used block-wise missing non-correlated synthetic training data.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
28.60194	5.348078	3.701681	1.923975	0.8284309	0.7002119

Table C.8: Evaluation values for the model when used block-wise missing non-correlated synthetic testing data.

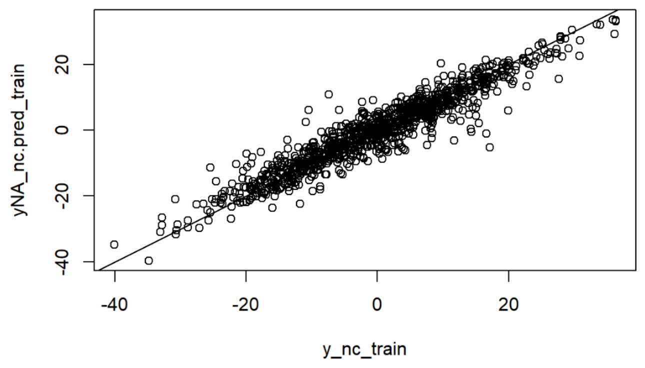


Figure C.7: Predicted training outcome vs real training outcome for block-wise missing non-correlated synthetic data.

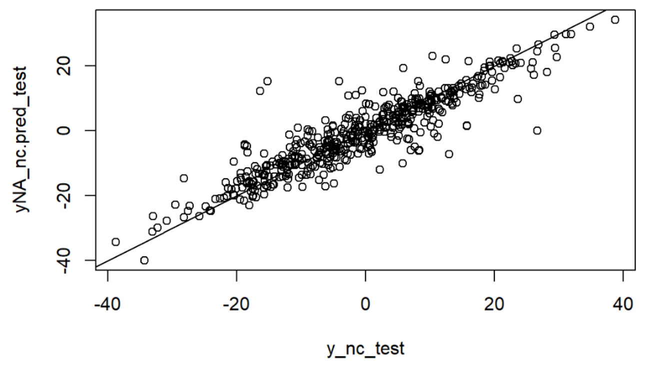


Figure C.8: Predicted testing outcome vs real testing outcome for block-wise missing non-correlated synthetic data.

- Low-correlated data

```
iSFS.ModelNA_lc <- iSFS(p = p.synth, X = X.NA_lc_train,
                        y = y_lc_train, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 20,
                        maxIter.alpha = 20, maxIter.beta = 20)

yNA_lc.pred_train <- predict.iSFS(iSFS.ModelNA_lc, X.NA_lc_train,
                                  p.synth)
evaluation.model.param(y_lc_train, yNA_lc.pred_train, sum(p.synth))

yNA_lc.pred_test <- predict.iSFS(iSFS.ModelNA_lc, X.NA_lc_test,
                                 p.synth)
evaluation.model.param(y_lc_test, yNA_lc.pred_test, sum(p.synth))
```

```
plot(y_lc_train, yNA_lc.pred_train)
abline(a = 0, b = 1)

plot(y_lc_test, yNA_lc.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
18.36427	4.285356	3.264594	1.806819	0.8265111	0.7807988

Table C.9: Evaluation values for the model when used block-wise missing low-correlated synthetic training data.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
30.2046	5.495871	3.983668	1.995913	0.7418698	0.5489612

Table C.10: Evaluation values for the model when used block-wise missing low-correlated synthetic testing data.

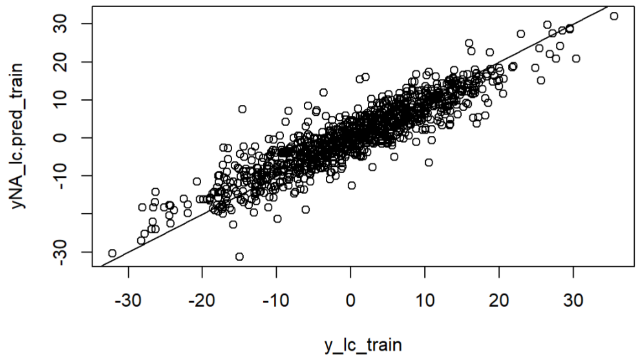


Figure C.9: Predicted training outcome vs real training outcome for block-wise missing low-correlated synthetic data.

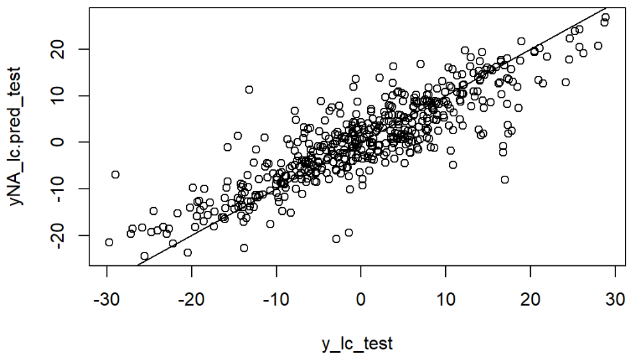


Figure C.10: Predicted testing outcome vs real testing outcome for block-wise missing low-correlated synthetic data.

- High-correlated data

```r
iSFS.ModelNA_hc <- iSFS(p = p.synth, X = X.NA_hc_train,
                        y = y_hc_train, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 20,
                        maxIter.alpha = 20,  maxIter.beta = 20)

yNA_hc.pred_train <- predict.iSFS(iSFS.ModelNA_hc, X.NA_hc_train,
                                  p.synth)
evaluation.model.param(y_hc_train, yNA_hc.pred_train, sum(p.synth))

yNA_hc.pred_test <- predict.iSFS(iSFS.ModelNA_hc, X.NA_hc_test,
                                 p.synth)
evaluation.model.param(y_hc_test, yNA_hc.pred_test, sum(p.synth))

plot(y_hc_train, yNA_hc.pred_train)
abline(a = 0, b = 1)

plot(y_hc_test, yNA_hc.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
4.758615	2.181425	1.669518	1.292098	0.7869108	0.7307644

Table C.11: Evaluation values for the model when used block-wise missing high-correlated synthetic training data.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
6.160887	2.482113	1.88701	1.373685	0.7282028	0.5250803

Table C.12: Evaluation values for the model when used block-wise missing high-correlated synthetic testing data.

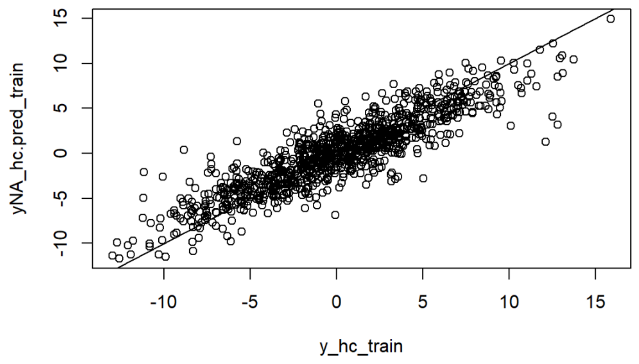


Figure C.11: Predicted training outcome vs real training outcome for block-wise missing high-correlated synthetic data.

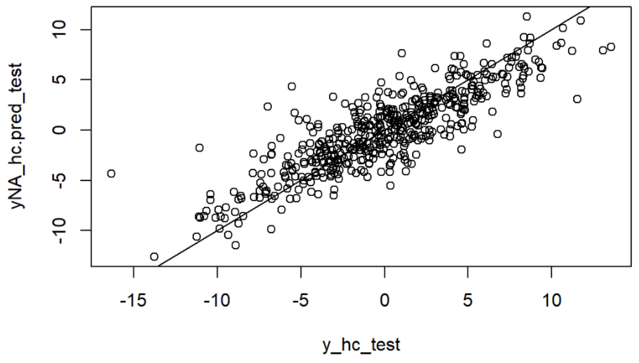


Figure C.12: Predicted testing outcome vs real testing outcome for block-wise missing high-correlated synthetic data.

C.2 Exposome data

```
# We select the indices that we will use for training
indexes_partition <-
   createDataPartition(y = 1:dim(exposome.data.nv)[1], p = prob_train,
                       list = FALSE)
```

```
# Data matrix numeric variables
exposome.data.nv_train <- exposome.data.nv[indexes_partition, ]
exposome.data.nv_test <- exposome.data.nv[-indexes_partition, ]
exposomeNA.data.nv_train <- exposomeNA.data.nv[indexes_partition, ]
exposomeNA.data.nv_test <- exposomeNA.data.nv[-indexes_partition, ]
# Data matrix dummy variables
exposome.data.dv_train <- exposome.data.dv[indexes_partition, ]
exposome.data.dv_test <- exposome.data.dv[-indexes_partition, ]
exposomeNA.data.dv_train <- exposomeNA.data.dv[indexes_partition, ]
exposomeNA.data.dv_test <- exposomeNA.data.dv[-indexes_partition, ]

# Outcome
y_train <- y[indexes_partition, ]
y_test <- y[-indexes_partition, ]
```

C.2.1 Comparison on complete data

C.2.1.1 Numeric variables

- Outcome *hs_zbmi_who*

```
iSFS.Model.nv <- iSFS(p = p.nv, X = exposome.data.nv_train,
                      y = y_train$hs_zbmi_who, lambda = 0.00000005,
                      L.step = 10, maxIter.iSFS = 100,
                      maxIter.alpha = 20, maxIter.beta = 50)

y.nv.pred_train <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_train,
                                p.nv)
evaluation.model.param(y_train$hs_zbmi_who, y.nv.pred_train,
                       sum(p.nv))

y.nv.pred_test <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_test,
                               p.nv)
evaluation.model.param(y_test$hs_zbmi_who, y.nv.pred_test,
                       sum(p.nv))

plot(y_train$hs_zbmi_who, y.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_zbmi_who, y.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.4146917	0.6439656	0.492233	0.7015932	0.7151116	0.6430708

Table C.13: Evaluation values for the model when used complete exposome (numeric variables) training data for the outcome *hs_zbmi_who*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.4891764	0.6994115	0.5407398	0.7353501	0.6325844	0.3749544

Table C.14: Evaluation values for the model when used complete exposome (numeric variables) testing data for the outcome *hs_zbmi_who*.

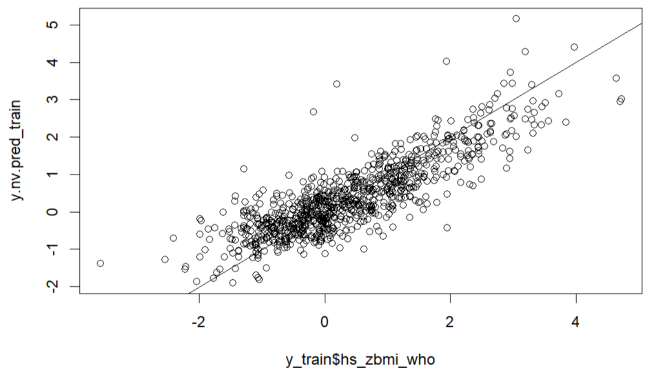


Figure C.13: Predicted training outcome vs real training outcome for complete exposome (numeric variables) data and for the outcome *hs_zbmi_who*.

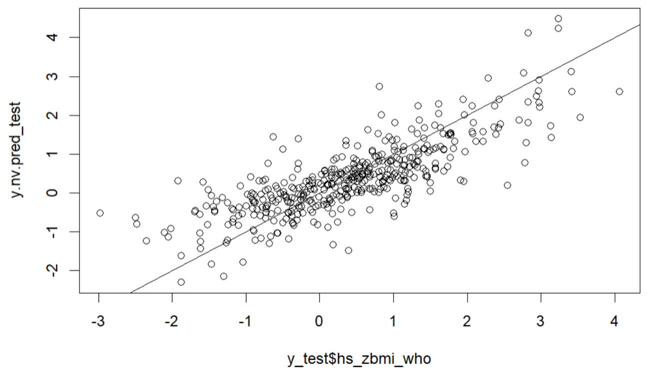


Figure C.14: Predicted testing outcome vs real testing outcome for complete exposome (numeric variables) data and for the outcome *hs_zbmi_who*.

```
# Numeric variables names
nv.colnames <- colnames(exposome.data.nv_train)

# Non-relevant features
nv.colnames[which(abs(iSFS.Model.nv$beta) < 0.05)]
```

```
[1] "h_NO2_Log"                 "h_trafload_preg_pow1over3"
```

- Outcome *e3_bw*

```
iSFS.Model.nv <- iSFS(p = p.nv, X = exposome.data.nv_train,
                      y = y_train$e3_bw, lambda = 0.00000005,
                      L.step = 10, maxIter.iSFS = 100,
                      maxIter.alpha = 20, maxIter.beta = 50)

y.nv.pred_train <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_train,
                                p.nv)
evaluation.model.param(y_train$e3_bw, y.nv.pred_train, sum(p.nv))

y.nv.pred_test <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_test,
                               p.nv)
evaluation.model.param(y_test$e3_bw, y.nv.pred_test, sum(p.nv))

plot(y_train$e3_bw, y.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$e3_bw, y.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.1470631	0.3834881	0.2967765	0.5447719	0.4360478	0.2934392

Table C.15: Evaluation values for the model when used complete exposome (numeric variables) training data for the outcome *e3_bw*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.1713531	0.4139482	0.3200768	0.5657533	0.3326442	-0.1353025

Table C.16: Evaluation values for the model when used complete exposome (numeric variables) testing data for the outcome *e3_bw*.

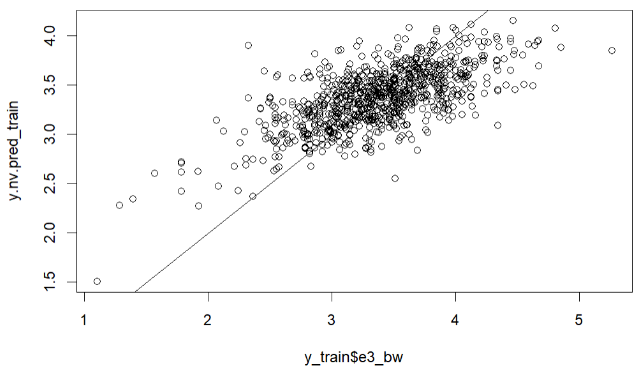


Figure C.15: Predicted training outcome vs real training outcome for complete exposome (numeric variables) data and for the outcome *e3_bw*.

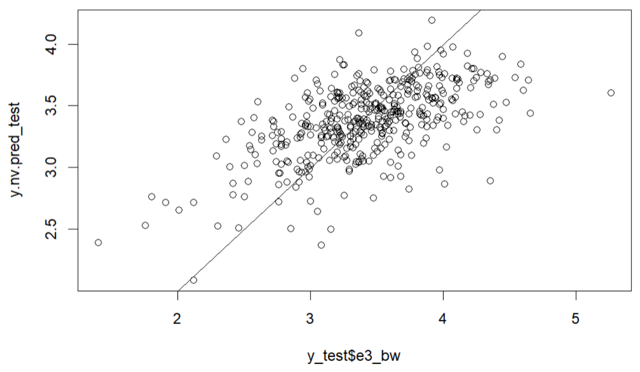


Figure C.16: Predicted testing outcome vs real testing outcome for complete exposome (numeric variables) data and for the outcome *e3_bw*.

```
# Non-relevant features
nv.colnames[which(abs(iSFS.Model.nv$beta) < 0.05)]
```

```
[1] "h_builtdens300_preg_Sqrt" "hs_builtdens300_h_Sqrt"
[3] "hs_builtdens300_s_Sqrt"
```

- Outcome *hs_correct_raven*

```
iSFS.Model.nv <- iSFS(p = p.nv, X = exposome.data.nv_train,
                      y = y_train$hs_correct_raven, lambda =
                      0.00000005, L.step = 10, maxIter.iSFS = 100,
                      maxIter.alpha = 20, maxIter.beta = 50)
```

```
y.nv.pred_train <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_train,
                                p.nv)
evaluation.model.param(y_train$hs_correct_raven, y.nv.pred_train,
                       sum(p.nv))

y.nv.pred_test <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_test,
                               p.nv)
evaluation.model.param(y_test$hs_correct_raven, y.nv.pred_test,
                       sum(p.nv))

plot(y_train$hs_correct_raven, y.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_correct_raven, y.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
16.1775	4.022126	3.152536	1.775538	0.631782	0.5386694

Table C.17: Evaluation values for the model when used complete exposome (numeric variables) training data for the outcome *hs_correct_raven*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
18.68362	4.322455	3.371796	1.836245	0.4873281	0.127845

Table C.18: Evaluation values for the model when used complete exposome (numeric variables) testing data for the outcome *hs_correct_raven*.

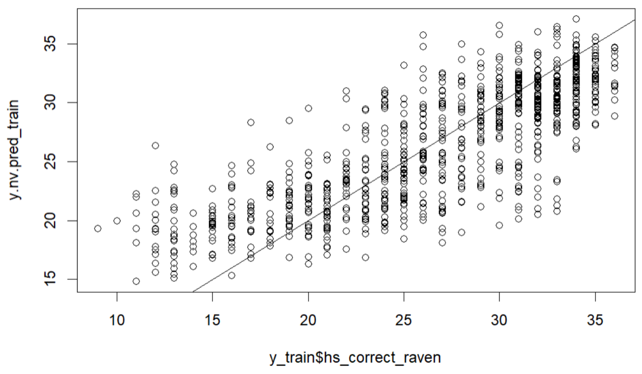


Figure C.17: Predicted training outcome vs real training outcome for complete exposome (numeric variables) data and for the outcome *hs_correct_raven*.

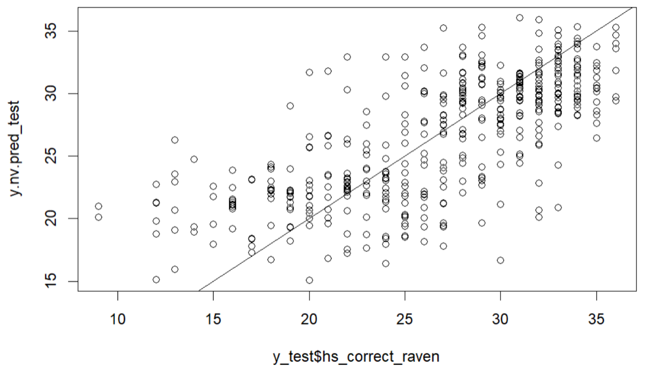


Figure C.18: Predicted testing outcome vs real testing outcome for complete exposome (numeric variables) data and for the outcome *hs_correct_raven*.

```
# Non-relevant features
nv.colnames[which(abs(iSFS.Model.nv$beta) < 0.05)]
```

```
character(0)
```

- Outcome *hs_Gen_Tot*

```
iSFS.Model.nv <- iSFS(p = p.nv, X = exposome.data.nv_train,
                      y = y_train$hs_Gen_Tot, lambda = 0.00000005,
                      L.step = 10, maxIter.iSFS = 100, maxIter.alpha
                      = 20, maxIter.beta = 50)

y.nv.pred_train <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_train,
                                p.nv)
evaluation.model.param(y_train$hs_Gen_Tot, y.nv.pred_train, sum(p.nv))

y.nv.pred_test <- predict.iSFS(iSFS.Model.nv, exposome.data.nv_test,
                               p.nv)
evaluation.model.param(y_test$hs_Gen_Tot, y.nv.pred_test, sum(p.nv))

plot(y_train$hs_Gen_Tot, y.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_Gen_Tot, y.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
247.2055	15.72277	12.0205	3.467059	0.3535896	0.1901295

Table C.19: Evaluation values for the model when used complete exposome (numeric variables) training data for the outcome *hs_Gen_Tot*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
341.1913	18.47136	14.03774	3.746698	-0.07623641	-0.8308882

Table C.20: Evaluation values for the model when used complete exposome (numeric variables) testing data for the outcome *hs_Gen_Tot*.

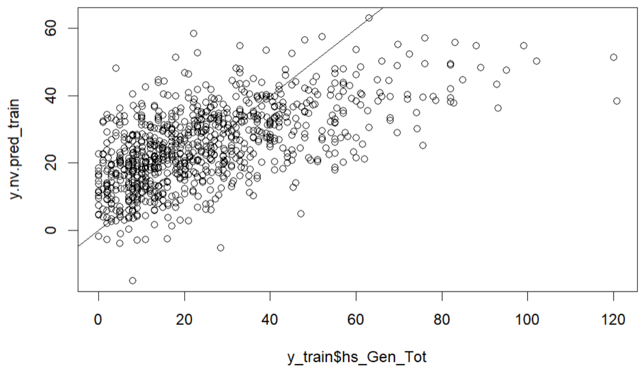


Figure C.19: Predicted training outcome vs real training outcome for complete exposome (numeric variables) data and for the outcome *hs_Gen_Tot*.

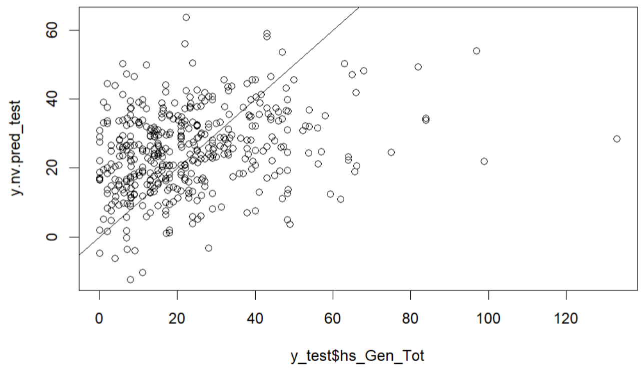


Figure C.20: Predicted testing outcome vs real testing outcome for complete exposome (numeric variables) data and for the outcome *hs_Gen_Tot*.

```
# Non-relevant features
nv.colnames[which(abs(iSFS.Model.nv$beta) < 0.05)]
```

```
character(0)
```

C.2.1.2 Dummy variables

- Outcome *hs_zbmi_who*

```
iSFS.Model.dv <- iSFS(p = p.dv, X = exposome.data.dv_train, y =
                      y_train$hs_zbmi_who, lambda = 0.00000005,
                      L.step = 10, maxIter.iSFS = 100,
                      maxIter.alpha = 20, maxIter.beta = 50,
                      beta0.comp = "LR")

y.dv.pred_train <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_train,
                                p.dv)
evaluation.model.param(y_train$hs_zbmi_who, y.dv.pred_train,
                       sum(p.dv))

y.dv.pred_test <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_test,
                               p.dv)
evaluation.model.param(y_test$hs_zbmi_who, y.dv.pred_test)

plot(y_train$hs_zbmi_who, y.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_zbmi_who, y.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.4290164	0.6549934	0.5020886	0.7085821	0.7052706	0.5553564

Table C.21: Evaluation values for the model when used complete exposome (dummy variables) training data for the outcome *hs_zbmi_who*.

MSE	RMSE	MAE	RMAE	R squared
0.479162	0.6922153	0.5320198	0.7293969	0.6401061

Table C.22: Evaluation values for the model when used complete exposome (dummy variables) testing data for the outcome *hs_zbmi_who*.

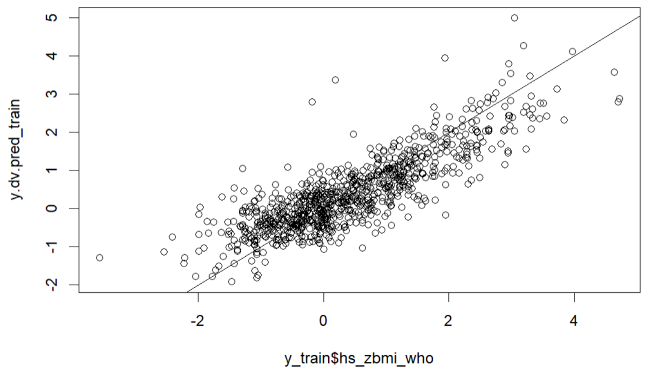


Figure C.21: Predicted training outcome vs real training outcome for complete exposome (dummy variables) data and for the outcome *hs_zbmi_who*.

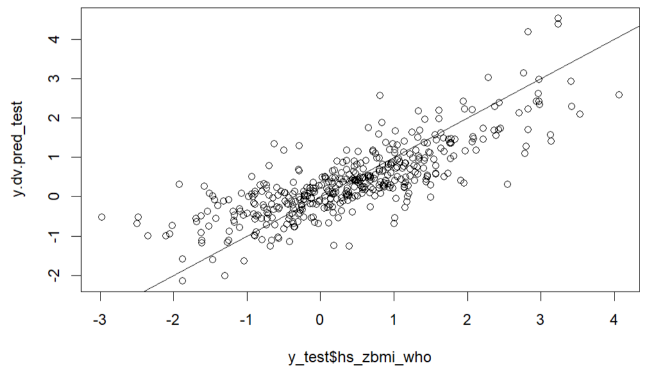


Figure C.22: Predicted testing outcome vs real testing outcome for complete exposome (dummy variables) data and for the outcome *hs_zbmi_who*.

```
# Dummy variables names
dv.colnames <- colnames(exposome.data.dv_train)

# Non-relevant features
dv.colnames[which(abs(iSFS.Model.dv$beta) < 0.05)]
```

```
[1] "variable.female"              "h_landuseshan300_preg_None"
[3] "hs_connind300_h_Log"          "hs_builtdens300_s_Sqrt"
[5] "variable..0.6....6.9."
```

- Outcome *e3_bw*

```r
iSFS.Model.dv <- iSFS(p = p.dv, X = exposome.data.dv_train, y =
                      y_train$e3_bw, lambda = 0.00000005,
                      L.step = 10, maxIter.iSFS = 100,
                      maxIter.alpha = 20, maxIter.beta = 50,
                      beta0.comp = "LR")

y.dv.pred_train <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_train,
                                p.dv)
evaluation.model.param(y_train$e3_bw, y.dv.pred_train, sum(p.dv))

y.dv.pred_test <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_test,
                               p.dv)
evaluation.model.param(y_test$e3_bw, y.dv.pred_test)

plot(y_train$e3_bw, y.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$e3_bw, y.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.1614898	0.401858	0.3120213	0.5585887	0.3807247	0.06572992

Table C.23: Evaluation values for the model when used complete exposome (dummy variables) training data for the outcome *e3_bw*.

MSE	RMSE	MAE	RMAE	R squared
0.1815342	0.4260683	0.3301896	0.5746213	0.2929926

Table C.24: Evaluation values for the model when used complete exposome (dummy variables) testing data for the outcome *e3_bw*.

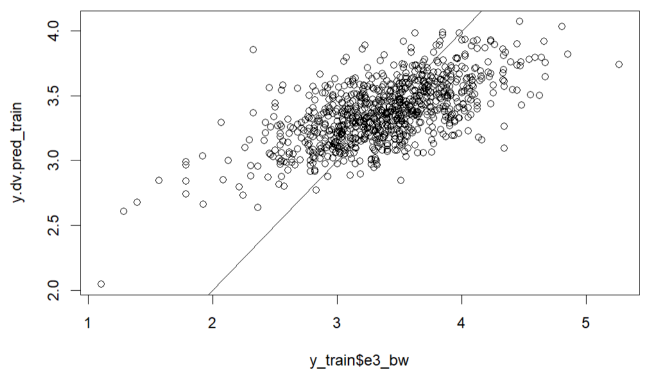


Figure C.23: Predicted training outcome vs real training outcome for complete exposome (dummy variables) data and for the outcome *e3_bw*.

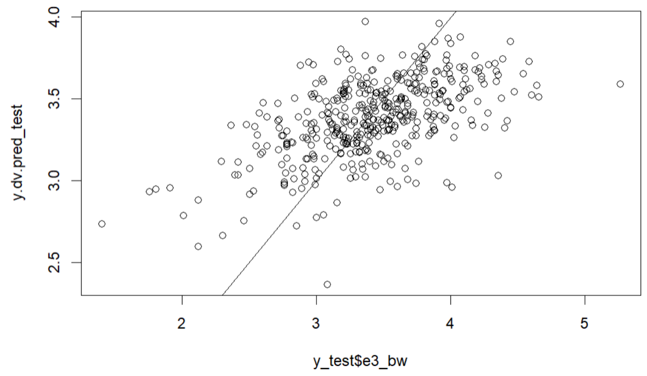


Figure C.24: Predicted testing outcome vs real testing outcome for complete exposome (dummy variables) data and for the outcome *e3_bw*.

```
# Non-relevant features
dv.colnames[which(abs(iSFS.Model.dv$beta) < 0.05)]
```

```
[1] "hs_builtdens300_h_Sqrt" "hs_builtdens300_s_Sqrt" "variable.0.1"
[4] "hs_trcs_madj_Log2"
```

- Outcome *hs_correct_raven*

```
iSFS.Model.dv <- iSFS(p = p.dv, X = exposome.data.dv_train, y =
                      y_train$hs_correct_raven, lambda = 0.00000005,
                      L.step = 10, maxIter.iSFS = 100,
                      maxIter.alpha = 20, maxIter.beta = 50,
                      beta0.comp = "LR")
```

```
y.dv.pred_train <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_train,
                                p.dv)
evaluation.model.param(y_train$hs_correct_raven, y.dv.pred_train,
                       sum(p.dv))

y.dv.pred_test <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_test,
                               p.dv)
evaluation.model.param(y_test$hs_correct_raven, y.dv.pred_test)

plot(y_train$hs_correct_raven, y.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_correct_raven, y.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
15.76673	3.970734	3.10729	1.762751	0.6411315	0.4585928

Table C.25: Evaluation values for the model when used complete exposome (dummy variables) training data for the outcome *hs_correct_raven*.

MSE	RMSE	MAE	RMAE	R squared
18.76777	4.332178	3.382309	1.839105	0.4850191

Table C.26: Evaluation values for the model when used complete exposome (dummy variables) testing data for the outcome *hs_correct_raven*.

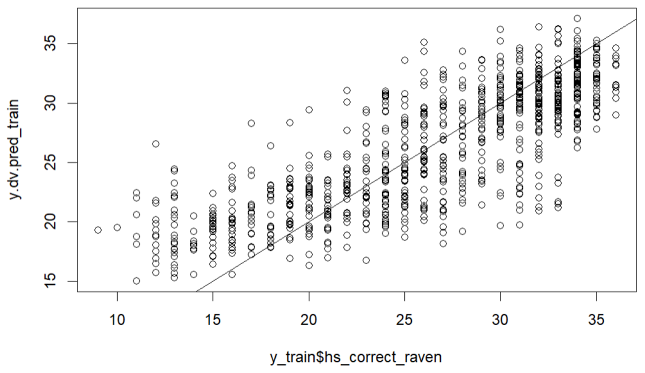


Figure C.25: Predicted training outcome vs real training outcome for complete exposome (dummy variables) data and for the outcome *hs_correct_raven*.

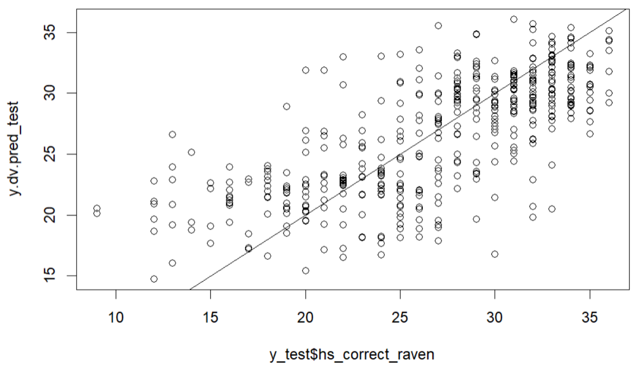


Figure C.26: Predicted testing outcome vs real testing outcome for complete exposome (dummy variables) data and for the outcome *hs_correct_raven*.

```
# Non-relevant features
dv.colnames[which(abs(iSFS.Model.dv$beta) < 0.05)]
```

```
character(0)
```

- Outcome *hs_Gen_Tot*

```
iSFS.Model.dv <- iSFS(p = p.dv, X = exposome.data.dv_train, y =
                      y_train$hs_Gen_Tot, lambda = 0.00000005,
                      L.step = 10, maxIter.iSFS = 100,
                      maxIter.alpha = 20, maxIter.beta = 50,
                      beta0.comp = "LR")

y.dv.pred_train <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_train,
                                p.dv)
evaluation.model.param(y_train$hs_Gen_Tot, y.dv.pred_train, sum(p.dv))

y.dv.pred_test <- predict.iSFS(iSFS.Model.dv, exposome.data.dv_test,
                               p.dv)
evaluation.model.param(y_test$hs_Gen_Tot, y.dv.pred_test)

plot(y_train$hs_Gen_Tot, y.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_Gen_Tot, y.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
239.9175	15.48927	11.75104	3.42798	0.3726469	0.05354336

Table C.27: Evaluation values for the model when used complete exposome (dummy variables) training data for the outcome *hs_Gen_Tot*.

MSE	RMSE	MAE	RMAE	R squared
318.5169	17.84704	13.43383	3.665219	-0.00471336

Table C.28: Evaluation values for the model when used complete exposome (dummy variables) testing data for the outcome *hs_Gen_Tot*.

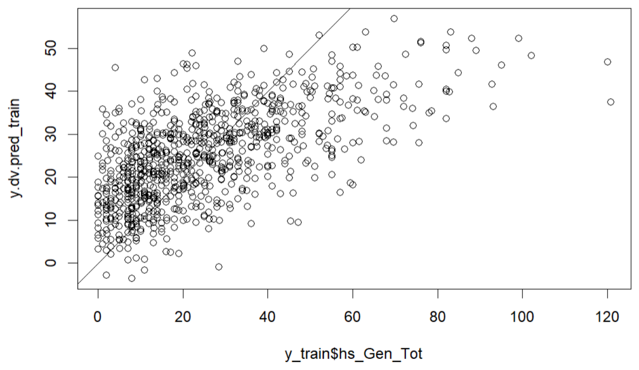


Figure C.27: Predicted training outcome vs real training outcome for complete exposome (dummy variables) data and for the outcome *hs_Gen_Tot*.

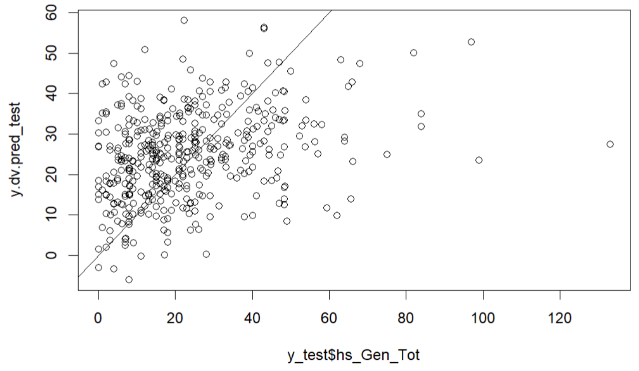


Figure C.28: Predicted testing outcome vs real testing outcome for complete exposome (dummy variables) data and for the outcome *hs_Gen_Tot*.

```
# Non-relevant features
dv.colnames[which(abs(iSFS.Model.dv$beta) < 0.05)]
```

```
character(0)
```

C.2.2 Comparison on incomplete data

C.2.2.1 Numeric variables

- Outcome *hs_zbmi_who*

```
iSFS.ModelNA.nv <- iSFS(p = p.nv, X = exposomeNA.data.nv_train,
                        y = y_train$hs_zbmi_who, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 100,
                        maxIter.alpha = 20, maxIter.beta = 50)

yNA.nv.pred_train <- predict.iSFS(iSFS.ModelNA.nv,
                                  exposomeNA.data.nv_train, p.nv)
evaluation.model.param(y_train$hs_zbmi_who, yNA.nv.pred_train,
                       sum(p.nv))

yNA.nv.pred_test <- predict.iSFS(iSFS.ModelNA.nv,
                                 exposomeNA.data.nv_test, p.nv)
evaluation.model.param(y_test$hs_zbmi_who, yNA.nv.pred_test,
                       sum(p.nv))

plot(y_train$hs_zbmi_who, yNA.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_zbmi_who, yNA.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.6807339	0.825066	0.6389482	0.7993424	0.5323436	0.4140857

Table C.29: Evaluation values for the model when used block-wise missing exposome (numeric variables) training data for the outcome *hs_zbmi_who*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.6904253	0.8309184	0.6388579	0.7992858	0.4814284	0.1178084

Table C.30: Evaluation values for the model when used block-wise missing exposome (numeric variables) testing data for the outcome *hs_zbmi_who*.

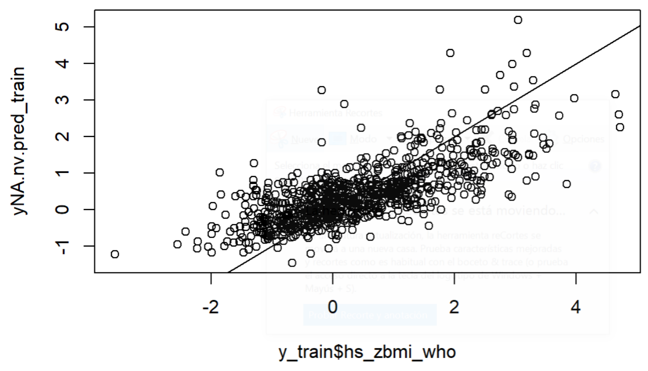


Figure C.29: Predicted training outcome vs real training outcome for block-wise missing exposome (numeric variables) data and for the outcome *hs_zbmi_who*.

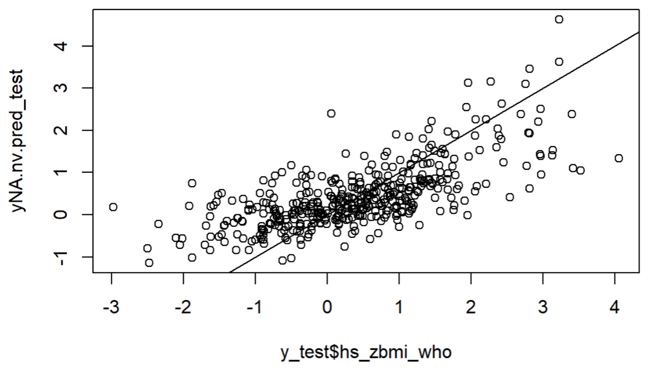


Figure C.30: Predicted testing outcome vs real testing outcome for block-wise missing exposome (numeric variables) data and for the outcome *hs_zbmi_who*.

- Outcome *e3_bw*

```
iSFS.ModelNA.nv <- iSFS(p = p.nv, X = exposomeNA.data.nv_train,
                        y = y_train$e3_bw, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 100,
                        maxIter.alpha = 20, maxIter.beta = 50)

yNA.nv.pred_train <- predict.iSFS(iSFS.ModelNA.nv,
                                  exposomeNA.data.nv_train, p.nv)
evaluation.model.param(y_train$e3_bw, yNA.nv.pred_train, sum(p.nv))

yNA.nv.pred_test <- predict.iSFS(iSFS.ModelNA.nv,
                                 exposomeNA.data.nv_test, p.nv)
evaluation.model.param(y_test$e3_bw, yNA.nv.pred_test, sum(p.nv))
```

```
plot(y_train$e3_bw, yNA.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$e3_bw, yNA.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.229331	0.4788851	0.3683425	0.6069123	0.1205699	-0.1018147

Table C.31: Evaluation values for the model when used block-wise missing exposome (numeric variables) training data for the outcome *e3_bw*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.2532475	0.503237	0.3896707	0.6242361	0.01369617	-0.6778954

Table C.32: Evaluation values for the model when used block-wise missing exposome (numeric variables) testing data for the outcome *e3_bw*.

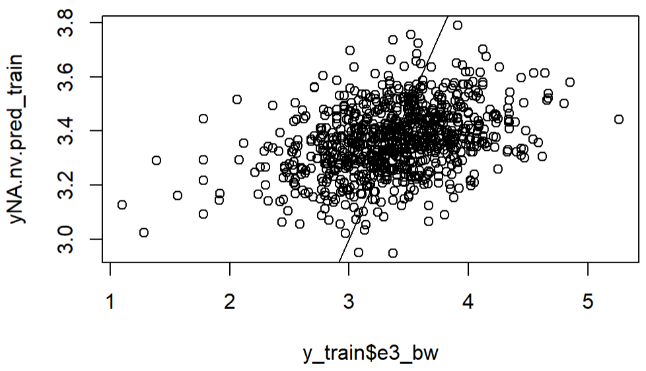


Figure C.31: Predicted training outcome vs real training outcome for block-wise missing exposome (numeric variables) data and for the outcome *e3_bw*.

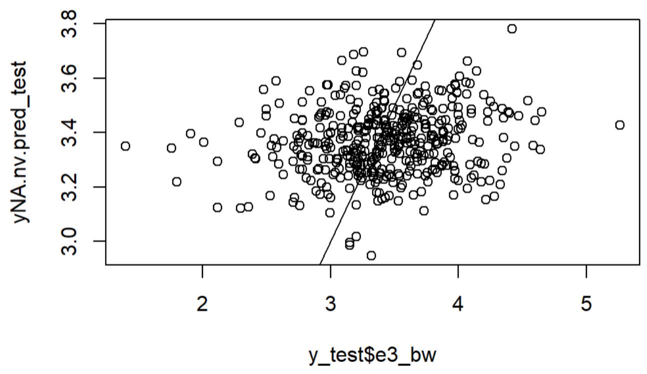


Figure C.32: Predicted testing outcome vs real testing outcome for block-wise missing exposome (numeric variables) data and for the outcome *e3_bw*.

- Outcome *hs_correct_raven*

```
iSFS.ModelNA.nv <- iSFS(p = p.nv, X = exposomeNA.data.nv_train,
                        y = y_train$hs_correct_raven,
                        lambda = 0.00000005, L.step = 10,
                        maxIter.iSFS = 100, maxIter.alpha = 20,
                        maxIter.beta = 50)

yNA.nv.pred_train <- predict.iSFS(iSFS.ModelNA.nv,
                                  exposomeNA.data.nv_train, p.nv)
evaluation.model.param(y_train$hs_correct_raven, yNA.nv.pred_train,
                       sum(p.nv))

yNA.nv.pred_test <- predict.iSFS(iSFS.ModelNA.nv,
                                 exposomeNA.data.nv_test, p.nv)
evaluation.model.param(y_test$hs_correct_raven, yNA.nv.pred_test,
                       sum(p.nv))

plot(y_train$hs_correct_raven, yNA.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_correct_raven, yNA.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	**RMSE**	**MAE**	**RMAE**	**R squared**	**Adjusted R squared**
27.79042	5.271662	4.202957	2.050111	0.3674587	0.2075057

Table C.33: Evaluation values for the model when used block-wise missing exposome (numeric variables) training data for the outcome *hs_correct_raven*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
28.23094	5.31328	4.186105	2.045997	0.2253529	-0.3178259

Table C.34: Evaluation values for the model when used block-wise missing exposome (numeric variables) testing data for the outcome *hs_correct_raven*.

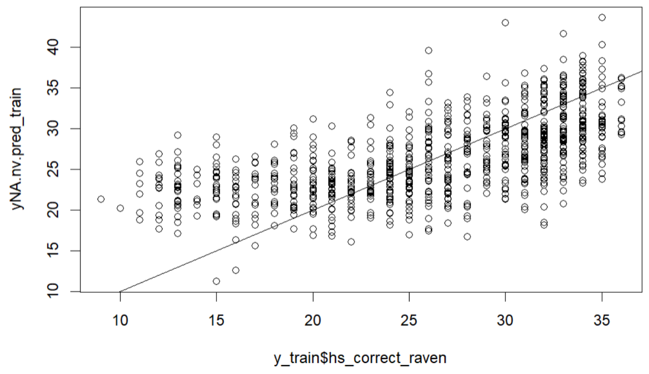


Figure C.33: Predicted training outcome vs real training outcome for block-wise missing exposome (numeric variables) data and for the outcome *hs_correct_raven*.

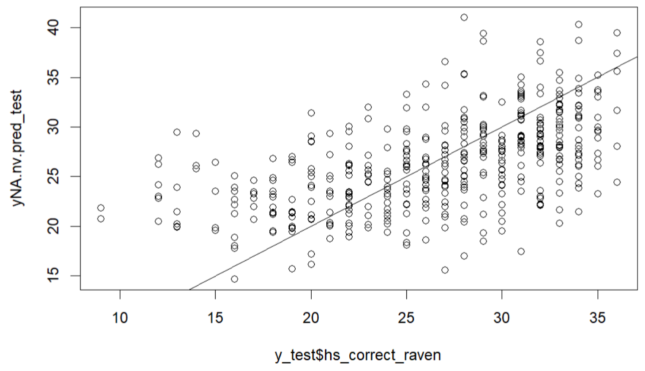


Figure C.34: Predicted testing outcome vs real testing outcome for block-wise missing exposome (numeric variables) data and for the outcome *hs_correct_raven*.

- Outcome *hs_Gen_Tot*

```
iSFS.ModelNA.nv <- iSFS(p = p.nv, X = exposomeNA.data.nv_train,
                        y = y_train$hs_Gen_Tot, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 100,
                        maxIter.alpha = 20, maxIter.beta = 50)

yNA.nv.pred_train <- predict.iSFS(iSFS.ModelNA.nv,
                                    exposomeNA.data.nv_train, p.nv)
evaluation.model.param(y_train$hs_Gen_Tot, yNA.nv.pred_train,
                       sum(p.nv))

yNA.nv.pred_test <- predict.iSFS(iSFS.ModelNA.nv,
                                   exposomeNA.data.nv_test, p.nv)
evaluation.model.param(y_test$hs_Gen_Tot, yNA.nv.pred_test,
                       sum(p.nv))

plot(y_train$hs_Gen_Tot, yNA.nv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_Gen_Tot, yNA.nv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
346.1951	18.60632	14.40539	3.795444	0.09474464	-0.1341705

Table C.35: Evaluation values for the model when used block-wise missing exposome (numeric variables) training data for the outcome *hs_Gen_Tot*.

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
339.9409	18.43749	14.36977	3.790748	-0.07229228	-0.8241785

Table C.36: Evaluation values for the model when used block-wise missing exposome (numeric variables) testing data for the outcome *hs_Gen_Tot*.

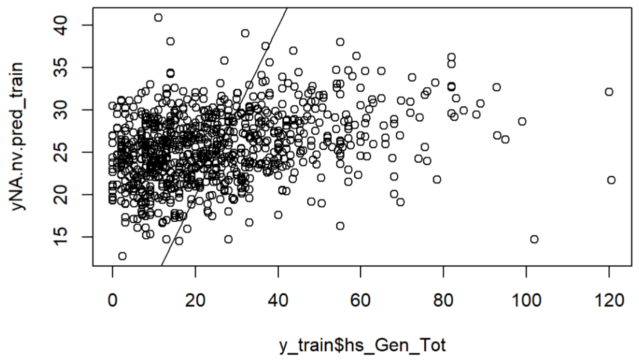


Figure C.35: Predicted training outcome vs real training outcome for block-wise missing exposome (numeric variables) data and for the outcome *hs_Gen_Tot*.

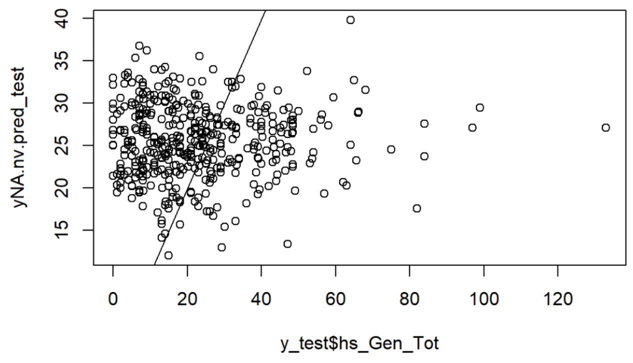


Figure C.36: Predicted testing outcome vs real testing outcome for block-wise missing exposome (numeric variables) data and for the outcome *hs_Gen_Tot*.

C.2.2.2 Dummy variables

- Outcome *hs_zbmi_who*

```
iSFS.ModelNA.dv <- iSFS(p = p.dv, X = exposomeNA.data.dv_train, y =
                        y_train$hs_zbmi_who, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 100,
                        maxIter.alpha = 20, maxIter.beta = 50,
                        beta0.comp = "LR")

yNA.dv.pred_train <- predict.iSFS(iSFS.ModelNA.dv,
                                  exposomeNA.data.dv_train,
                                  p.dv)
evaluation.model.param(y_train$hs_zbmi_who, yNA.dv.pred_train,
                       sum(p.dv))

yNA.dv.pred_test <- predict.iSFS(iSFS.ModelNA.dv,
                                 exposomeNA.data.dv_test,
                                 p.dv)
evaluation.model.param(y_test$hs_zbmi_who, yNA.dv.pred_test)

plot(y_train$hs_zbmi_who, yNA.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_zbmi_who, yNA.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.5511214	0.7423755	0.577232	0.7597579	0.6213859	0.4288036

Table C.37: Evaluation values for the model when used block-wise missing exposome (dummy variables) training data for the outcome *hs_zbmi_who*.

MSE	RMSE	MAE	RMAE	R squared
0.5596416	0.748092	0.5717857	0.7561651	0.5796587

Table C.38: Evaluation values for the model when used block-wise missing exposome (dummy variables) testing data for the outcome *hs_zbmi_who*.

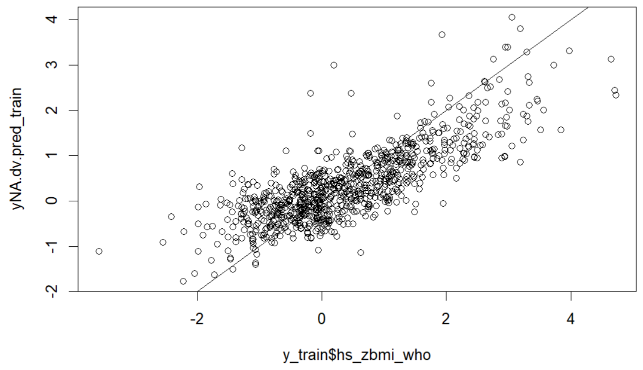


Figure C.37: Predicted training outcome vs real training outcome for block-wise missing exposome (dummy variables) data and for the outcome *hs_zbmi_who*.

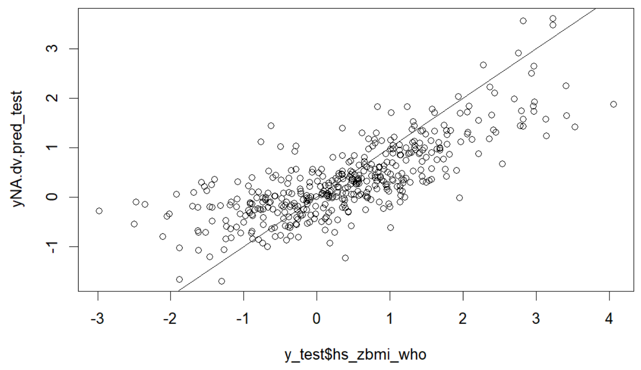


Figure C.38: Predicted testing outcome vs real testing outcome for block-wise missing exposome (dummy variables) data and for the outcome *hs_zbmi_who*.

- Outcome *e3_bw*

```
iSFS.ModelNA.dv <- iSFS(p = p.dv, X = exposomeNA.data.dv_train, y =
                        y_train$e3_bw, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 100,
                        maxIter.alpha = 20, maxIter.beta = 50,
                        beta0.comp = "LR")

yNA.dv.pred_train <- predict.iSFS(iSFS.ModelNA.dv,
                                  exposomeNA.data.dv_train,
                                  p.dv)
evaluation.model.param(y_train$e3_bw, yNA.dv.pred_train,
                       sum(p.dv))

yNA.dv.pred_test <- predict.iSFS(iSFS.ModelNA.dv,
                                 exposomeNA.data.dv_test,
                                 p.dv)
evaluation.model.param(y_test$e3_bw, yNA.dv.pred_test)

plot(y_train$e3_bw, yNA.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$e3_bw, yNA.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
0.2206533	0.4697374	0.3617135	0.6014262	0.1538469	-0.2765493

Table C.39: Evaluation values for the model when used block-wise missing exposome (dummy variables) training data for the outcome *e3_bw*.

MSE	RMSE	MAE	RMAE	R squared
0.2475389	0.4975328	0.3818376	0.6179301	0.03592912

Table C.40: Evaluation values for the model when used block-wise missing exposome (dummy variables) testing data for the outcome *e3_bw*.

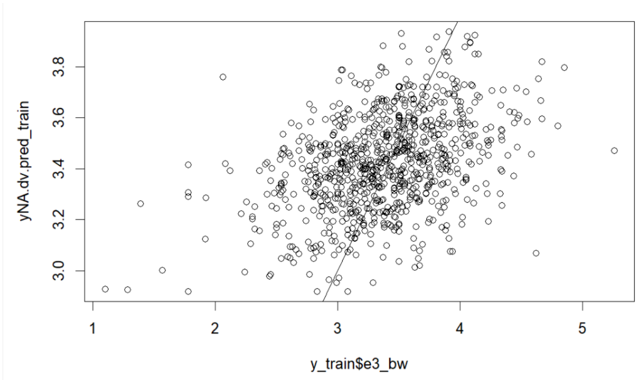


Figure C.39: Predicted training outcome vs real training outcome for block-wise missing exposome (dummy variables) data and for the outcome *e3_bw*.

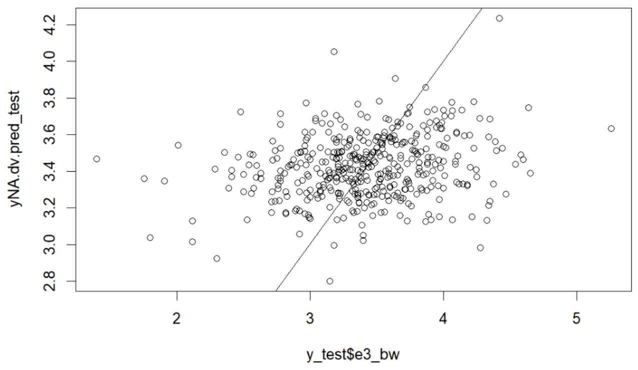


Figure C.40: Predicted testing outcome vs real testing outcome for block-wise missing exposome (dummy variables) data and for the outcome *e3_bw*.

- Outcome *hs_correct_raven*

```
iSFS.ModelNA.dv <- iSFS(p = p.dv, X = exposomeNA.data.dv_train, y =
                        y_train$hs_correct_raven, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 100,
                        maxIter.alpha = 20, maxIter.beta = 50,
                        beta0.comp = "LR")

yNA.dv.pred_train <- predict.iSFS(iSFS.ModelNA.dv,
                                  exposomeNA.data.dv_train,
                                  p.dv)
evaluation.model.param(y_train$hs_correct_raven, yNA.dv.pred_train,
                       sum(p.dv))

yNA.dv.pred_test <- predict.iSFS(iSFS.ModelNA.dv,
                                     exposomeNA.data.dv_test, p.dv)
evaluation.model.param(y_test$hs_correct_raven, yNA.dv.pred_test)

plot(y_train$hs_correct_raven, yNA.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_correct_raven, yNA.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
41.77967	6.463719	5.406547	2.325198	0.04904769	-0.4346547

Table C.41: Evaluation values for the model when used block-wise missing exposome (dummy variables) training data for the outcome *hs_correct_raven*.

MSE	RMSE	MAE	RMAE	R squared
40.75194	6.383725	5.313987	2.305209	-0.1182188

Table C.42: Evaluation values for the model when used block-wise missing exposome (dummy variables) testing data for the outcome *hs_correct_raven*.

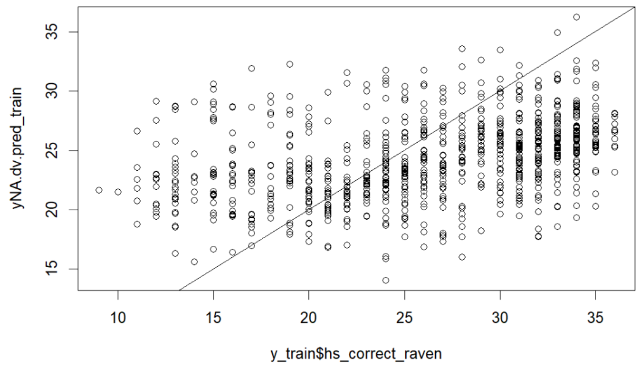


Figure C.41: Predicted training outcome vs real training outcome for block-wise missing exposome (dummy variables) data and for the outcome *hs_correct_raven*.

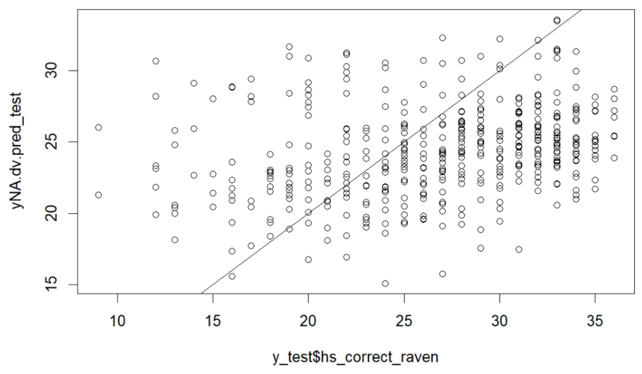


Figure C.42: Predicted testing outcome vs real testing outcome for block-wise missing exposome (dummy variables) data and for the outcome *hs_correct_raven*.

- Outcome *hs_Gen_Tot*

```
iSFS.ModelNA.dv <- iSFS(p = p.dv, X = exposomeNA.data.dv_train, y =
                        y_train$hs_Gen_Tot, lambda = 0.00000005,
                        L.step = 10, maxIter.iSFS = 100,
                        maxIter.alpha = 20, maxIter.beta = 50,
                        beta0.comp = "LR")

yNA.dv.pred_train <- predict.iSFS(iSFS.ModelNA.dv,
                                  exposomeNA.data.dv_train,
                                  p.dv)
evaluation.model.param(y_train$hs_Gen_Tot, yNA.dv.pred_train,
                       sum(p.dv))

yNA.dv.pred_test <- predict.iSFS(iSFS.ModelNA.dv,
                                 exposomeNA.data.dv_test, p.dv)
evaluation.model.param(y_test$hs_Gen_Tot, yNA.dv.pred_test)

plot(y_train$hs_Gen_Tot, yNA.dv.pred_train)
abline(a = 0, b = 1)

plot(y_test$hs_Gen_Tot, yNA.dv.pred_test)
abline(a = 0, b = 1)
```

MSE	RMSE	MAE	RMAE	R squared	Adjusted R squared
356.5169	18.88165	14.64036	3.826272	0.06775462	-0.4064325

Table C.43: Evaluation values for the model when used block-wise missing exposome (dummy variables) training data for the outcome *hs_Gen_Tot*.

MSE	RMSE	MAE	RMAE	R squared
349.911	18.70591	14.53511	3.812494	-0.1037414

Table C.44: Evaluation values for the model when used block-wise missing exposome (dummy variables) testing data for the outcome *hs_Gen_Tot*.

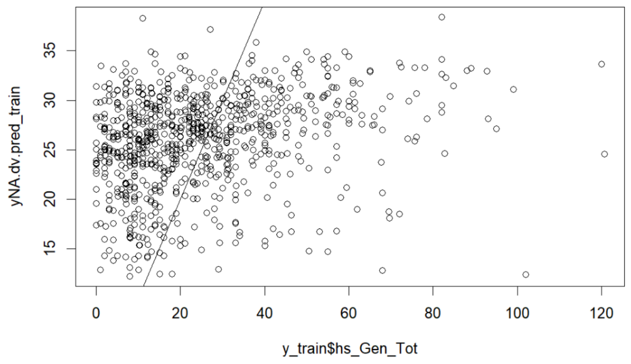


Figure C.43: Predicted training outcome vs real training outcome for block-wise missing exposome (dummy variables) data and for the outcome *hs_Gen_Tot*.

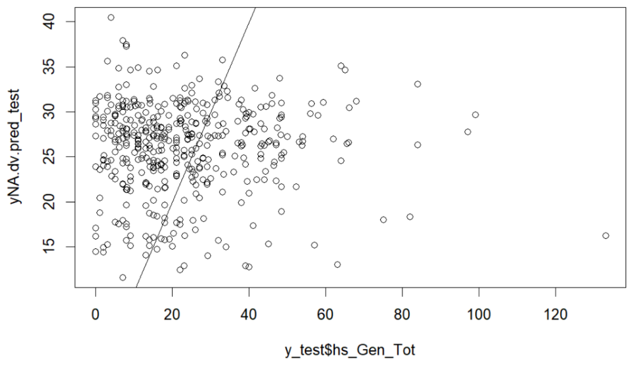


Figure C.44: Predicted testing outcome vs real testing outcome for block-wise missing exposome (dummy variables) data and for the outcome *hs_Gen_Tot*.