Tinius Petter Mellbye

Bachelor's thesis

A situation-specific solution to Freedman's paradox

Lasso on multiple imputed data sets as an alternative to p-value based variable selection and effect size reporting.

May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Bachelor's thesis

2021

Tinius Petter Mellbye

A situation-specific solution to Freedman's paradox

Lasso on multiple imputed data sets as an alternative to p-value based variable selection and effect size reporting.

Bachelor’s thesis
May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Abstract Some research uses p-values to a large extent for variable selection and the reporting of variables' explanatory power. However, this way of performing explanatory modelling has been shown to overestimate the parameter estimates. The purpose of this thesis is to present an alternative and improved way of performing explanatory modelling with multiple imputed data sets. The thesis introduces lasso, the multivariate imputation by chained equation procedure, cross-validation and the bootstrap; as well as how these are combined in a scheme to produce reliable parameter estimates and corresponding confidence intervals. The results obtained show a considerable overestimation of the parameter estimates from the p-value based approach and conclude that the proposed method is preferable.

Keywords Missing data · Multivariate Imputation by Chained Equations · Variable selection · p-values · Least absolute shrinkage and selection operator (Lasso) · Cross-validation · Bootstrap · Parameter estimation

Sammendrag Noen forskningsprosjekter baserer seg i stor grad på p-verdier for variabelvalg og rapportering av styrken på variablers relasjon til responsvariabelen. Det er vist at denne måten å utføre forklarende modellering fører til overestimering av parameterestimatenes størrelse. Hensikten med denne oppgaven er å presentere en alternativ og forbedret måte å utføre forklarende modellering med flere imputerte datasett. Oppgaven introduserer lasso, MICE prosedyren, kryssvalidering og bootstrap metoden, samt hvordan disse kombineres i en algoritme for å produsere pålitelige parameterestimater og tilhørende konfidensintervall. Resultatene viser en betydelig overestimering av parameterestimatene ved bruk av den p-verdi baserte metoden og konkluderer med at den foreslåtte metoden er å foretrekke.

Nøkkelord Manglende data · MICE prosedyren · Variabelvalg · p-verdier · Lasso · Kryssvalidering · Bootstrap metoden · Parameter estimering

Contents

1 Introduction

When research is performed there are multiple steps in the process from an idea to publication. Firstly, the goal of the study is determined. Secondly, the study is designed and data is collected. Thirdly, one preprocess and explore the data. Fourth, on selects variables. Fifth, the methods/models which will be used are determined. Sixth they are evaluated and validated before one is selected. Lastly, the model is applied and results reported. The workflow above is typically a way of performing research and should be considered a flow rather than a set of disjoint tasks (Shmueli, 2010). In the process above there are many decisions that need to be made. How to handle missing data (Little and Rubin, 2002) is among the first one encounters. How to handle this depends on the problem and the structure of the data. Some methods are faster but result in a reduction in performance, while others are more theoretically supported and produce better results, but can be more complicated, computationally expensive and limit which methods may be applied. After the data preprocessing one typically has to decide which variables to include in the analysis. There are several methods one may use for variable selection. Traditionally, p-values have been among the most used, but other, more modern, methods have become increasingly popular. For selecting the model which will be used, whether the goal is of an explanatory or predictive nature, the data structure, amongst other factors, will have an influence. For explaining relations between variables and the response, a simple model, like a linear regression model or decision tree, is generally preferable over a black-box model, but for high prediction accuracy, it is often the other way around. For simpler models like a linear model, which do not have any hyperparameters, the evaluation of the model is often performed indirectly by evaluating to which degree the assumptions of the model are satisfied. For more complex models with hyperparameters, the validation of the models is more direct. These models are often applied in a predictive modelling setting and the methods for evaluation and validation of the models are in most cases aiming to reflect the predictive performance of the models, typically with cross-validation (Stone, 1974). When fitting the model and reporting the results there are again many methods one can use depending on the goal and which model has been applied. To report the association between variables and the response, p-values have been extensively used when linear models are fitted. With the great expansion of more advanced statistical methods, we need other ways of explaining relations in the data. The fairly new branch in statistics called Explainable AI (XAI) aims to fill the void between advanced methods and explainability and has produced methods for simpler models like linear regression models (see Molnar (2020) for an introduction).

In this thesis, I will investigate some elements of the Schmutz et al. (2017) article, mostly related to variable selection and result reporting. The goal of Schmutz et al. (2017) is an explanatory analysis to identify ways to promote physical activity (PA) and decrease sedentary behaviour (SB) among preschool children. They collected data from several Swiss children in form of self-reporting and measurements. As is often the case, the data set had missing values. The missing data was handled by performing the multiple imputations by chained equations (mice) procedure (Buuren and Groothuis-Oudshoorn, 2011). The data was collected from multiple childcare centres and during the exploratory data analysis, a linear mixed model was found to be appropriate. The p-values from the linear mixed effect model including the full set of 35 covariates was used to select a subset of the variables before a second linear mixed model was fitted to the selected variables. Lastly, the p-values from the second model were reported as measures of the strength of the relationships between the variables and the responses, in addition to performing an R^2 decomposition of an ordinary linear regression model by using the LMG method (Gold et al., 1980).

Schmutz et al. (2017) solely relied on p-values for variable selection and reporting. However, this is a sub-optimal approach as when p-value based model selection has been performed and we refit the model to the subset of variables, the lastly obtained p-values tend to overestimate the significance of the variables (Freedman, 1983; Hubbard and Lindsay, 2008; Nahm, 2017). Moreover, one should consolidate other methods as well. This thesis will consider the lasso (Tibshirani, 1996) as an alternative to using p-values for variable selection specifically for the imputed data sets used in Schmutz et al. (2017) and compare the findings. The literature from combining coefficients from linear models fitted to imputed data sets is extensive and the software is well developed, for instance, the mice package (Buuren and Groothuis-Oudshoorn, 2011) in R (R Core Team, 2021).

Yet, other slightly more complex methods have little supporting software, which to some degree is due to the lack of research on how to combine the models from each of the imputed data sets. In this thesis, I will present a way of performing lasso on multiple imputed data sets and how to combine the results to presentable findings. I will introduce the multiple imputation procedure, more specifically the mice procedure, the linear regression model, the lasso in general and how it alters the linear regression model specifically. I will also argue for why other methods, like the lasso, should be applied instead of *p*-values for variable selection. Moreover, as standard errors for the coefficients are not readily available when performing lasso, I will describe how the bootstrap (Efron, 1979) is used to get an estimate of the standard errors. An algorithm for how these methods are combined to produce the desired result is also presented. In addition, a comparison of the results I obtained and the results obtained by following the procedure used in Schmutz et al. (2017), ending with a discussion of potential improvements.

2 Background & Theory

2.1 Missing data

The data set used in Schmutz et al. (2017) had missing values which were handled by using multiple imputation, more specifically the Multivariate Imputation by Chained Equations (MICE) procedure (see Buuren and Groothuis-Oudshoorn, 2011). Missing data can arise in most situations and fields of research. How to handle missing data depends on multiple factors, where the most prominent is the type of missing data. There are mainly three types of missing data, namely

- MCAR – Missing completely at random,
- MAR – Missing at random,
- MNAR – Missing not at random.

The first type is the least complicated, though not necessarily realistic. In that case, the missing data have no relationship with any of the observed or unobserved values in the data. Performing analysis on the data does not introduce any bias. The second type, missing at random, means that the distribution of the missing data only depends on the observed data and not the unobserved. Lastly, if the data is neither MCAR nor MAR, it is defined as MNAR, missing not at random (also called non-ignorable missing data). In this case, whether a data point will be missing depends on the missing data. To learn about missing data see for example Little and Rubin (2002).

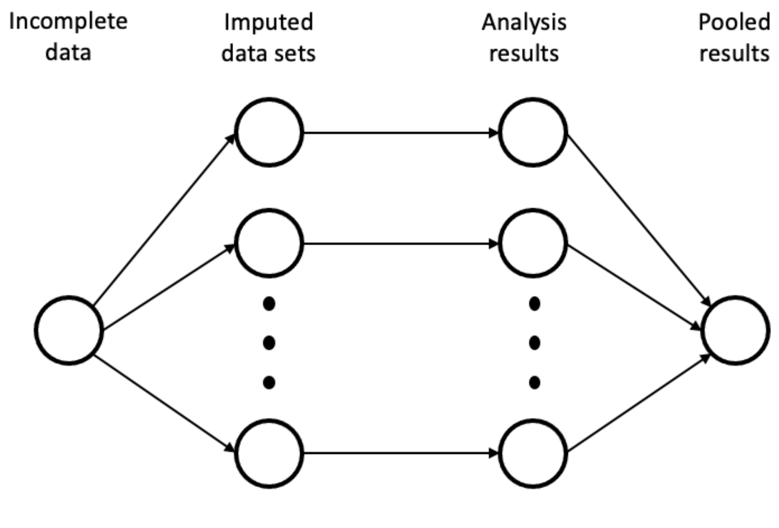


Figure 1: The main steps when using multiple imputation.

Figure 1 illustrates the main steps when doing multiple imputation. Firstly, one imputes the missing values into M separate data sets. Then one analyses the imputed data sets separately, possibly by fitting a model. Lastly, the results from the analysis are pooled together.

2.1.1 Multiple imputation

How the imputation step is done depends on the setting, data structure, e.g. the amount of data, the type of missing data, and so on. Schmutz et al. (2017) used the MICE procedure to produce 40 imputed data sets, which are the data sets used in this thesis.

For the sake of explanation consider the data $\mathbf{D}$ to have n observations and p variables, where $q \leq p$ variables have missing values, and a response vector $\mathbf{Y}$. Here we will assume, without loss of generality, that $\mathbf{D}$ has the q variables with missing values as the first q columns in the data. $\mathbf{D}_{-j}$ is the $n \times p$ data with the j^{th} variable removed. Furthermore, Ω_j is the set of indices for observations with missing values in variable j and $\mathcal{P}$ is a set of $m \cdot q$ prediction models, where M is the number of imputed data sets. The model applied to the j^{th} variable in the m^{th} data set is denoted as $\mathcal{P}_{m,j}$.

Algorithm 1 MICE algorithm

```
    Input:
        data $\mathbf{D}$
 1: initialise missing values in $\mathbf{D}$ with e.g. column mean or mode
 2: $\mathbf{D}^{(0)} = \mathbf{D}$
 3: for $m$ from 1 to $M$ do
 4:     $\mathbf{D}^* = \mathbf{D}^{(0)}$
 5:     while stopping criterion not met do
 6:         for $j$ from 1 to $q$ do
 7:             impute new values into $\mathbf{D}^*$ at $\Omega_j$ by using $\mathcal{P}_{m,j}$ trained on $\mathbf{D}_{-j}$
        $\mathbf{D}^{(m)} = \mathbf{D}^*$
 8: return $\mathbf{D}^{(m)}$ $\forall\, m \in \{1, \dots, M\}$
```

2.1.2 Model fitting

After the M imputed data sets are constructed one continues with the model fitting and analysis. Schmutz et al. (2017) used a random intercept linear mixed model (LMM), with a variable called *childcare* as random intercept, representing which childcare centre each child belonged to. The model was fitted to all the 40 data sets. In this thesis, an ordinary linear model will be used, as the childcare variable was not available. When one aims to fit a linear regression model to the data $\{y_i, x_{i1}, \dots, x_{in}\}_{i=1}^{n}$, it is assumed that

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}\,,$$

where

$$\begin{aligned}
\mathbf{Y} &= \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{bmatrix}, \\
\boldsymbol{\beta} &= \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix},
\end{aligned} \tag{1}$$

with $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. One seeks to find an estimate $\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ of $\mathbf{Y}$ which minimises the squared Euclidean distance between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ (which is equivalent to the minimum of the Euclidean distance). As $\mathbf{X}$ is considered to be fixed, the estimation is done by finding $\hat{\boldsymbol{\beta}}$ solving the optimisation problem

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\operatorname{argmin}} \mathcal{F}(\boldsymbol{\beta})$$

where

$$\mathcal{F}(\boldsymbol{\beta}) = ||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||_2^2 = (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \, .$$

In other words, we want to find $\hat{\boldsymbol{\beta}}$ solving

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^{p+1}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \, .$$

2.1.3 Pooling results

The results are pooled according to Rubin's rule first outlined by Rubin in 1987. Little and Rubin (2002, section 5.4) explain multiple imputation and how to pool the results. Let $\hat{\boldsymbol{\beta}}_m$ and $\mathbf{W}_m$, for $m \in \{1, \dots, M\}$, be the parameter estimates and the associated variances for the m^{th} imputed data set. Then the parameter estimates are combined across the M data sets by

$$\bar{\boldsymbol{\beta}}_M = \frac{1}{M} \sum_{m=1}^{M} \hat{\boldsymbol{\beta}}_m \, .$$

Furthermore, the variance of the estimates has two terms, the average of the within-imputation variance

$$\bar{\mathbf{W}}_M = \frac{1}{M} \sum_{m=1}^{M} \mathbf{W}_m \, ,$$

and the between-imputation variance

$$\mathbf{B}_M = \frac{1}{M-1} \sum_{m=1}^{M} (\hat{\boldsymbol{\beta}}_m - \bar{\boldsymbol{\beta}}_M)^T (\hat{\boldsymbol{\beta}}_m - \bar{\boldsymbol{\beta}}_M) \, .$$

The total variance is

$$\mathbf{V}_M = \bar{\mathbf{W}}_M + \frac{M+1}{M} \mathbf{B}_M \, ,$$

2.2 Variable selection

Variable selection, also known as feature selection, attribute selection or variable subset selection, is the process of selecting a subset of the observed variables. In explanatory modelling, one seeks to select the variables that explain the underlying process which generates the response/variable of interest. A used method is to pick variables with p-values smaller than a predetermined significance level α (often 0.05).

2.2.1 p-values

p-values are understood to be the *probability of observing what we have observed or something "more extreme" given that the null hypothesis is true.* In scientific publications, p-values are sometimes used to determine whether or not some variables are significant, which is also the case for Schmutz et al. (2017). Though, relying solely on p-values does not fully justify disregarding certain variables and include others in your model. A discussion of the validity of research findings, particularly based on p-values, can be found in Ioannidis (2005) and some p-value disclaimers can be found in Nahm (2017), for example. There are multiple underlying mechanisms that may lead to wrong conclusions. Freedman (1983) illustrate how selecting the most significant variables from a model, before refitting the model to the subset, will generally lead to overestimating variables' explanatory power and significance. In Schmutz et al. (2017) they firstly fit a linear mixed model, select 13 variables that had $p < 0.1$ and refitted the model to the 13 selected variables. Then the results from the last model were reported, where eight of the variables are considered significant when $\alpha = 0.1$. One can compare table 2 and 3 in the article to see that the p-values differ between the two models. No form of theoretical justification or argument for why they reported the results in this way is disclosed.

It is generally recommended that one consults other methods of variable selection to scrutinise the results obtained. There are multiple methods for variable selection and the scope of this thesis do not include all of them. An overview of some methods can be found in Hastie et al. (2009). The thesis will look into one method and how it can be implemented to handle the case of multiple imputed data sets.

2.2.2 Lasso

The least absolute shrinkage and selection operator, often called lasso, was first discovered by Santosa and Symes in 1986, though it was rediscovered in 1996 by Tibshirani who popularised it and is generally given the credit for the discovery. The lasso was coined as an intermediary between stepwise selection (Efroymson, 1960) and Ridge regression (Hoerl and Kennard, 1970). It sought to improve on the potential variability from the stepwise selection method following from it being a discrete method, and that ridge does not perform variable selection.

The lasso is widely used when performing predictive modelling, but the built-in variable selection method makes it suited for explanatory modelling as well. In predictive modelling, where prediction accuracy is the main goal, the lasso is often used as a regularisation technique to reduce overfitting, with the nice side effect of performing model selection. As an example, the unconstrained least squares model is unbiased but has a large variance. By shrinking the variables, we reduce the variance significantly and increase the bias slightly leading to enhanced predictive performance, a consequence of the *bias-variance trade-off* (Hastie et al., 2009, section 2.9 and 3.4).

On the other hand, in explanatory modelling, the variable selection aspect is generally the most desired property while the reduction in overfitting, and thus enhanced predictive performance, is considered a nice side effect. Yet, it is argued that also in explanatory modelling the predictive performance is relevant to assess. For a comprehensive discussion of explanatory versus predictive modelling see Shmueli (2010). As we saw above, when doing linear regression we want to find $\hat{\beta}$ solving

$$\min_{\hat{\boldsymbol{\beta}} \in \mathbb{R}^{p+1}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \,.$$

When we do lasso, we add the constraint that

$$\sum_{j=1}^{p} |\boldsymbol{\beta}_j| \leq t \,.$$

Note that we do not put a constraint on $\boldsymbol{\beta}_0$, the intercept. Thus we end up with

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\operatorname{argmin}} \big((\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda ||\boldsymbol{\beta}_{-0}||_1 \big) \,,$$

where $\boldsymbol{\beta}_{-0}$ denotes the $\boldsymbol{\beta}$ vector with the element corresponding to the intercept removed, λ is a tuning parameter which determines how much increasing the values of $\boldsymbol{\beta}_{-0}$ should be penalised and $|| \cdot ||_1$ denotes the ℓ_1-norm. Methods for solving such problems numerically can be found in Nocedal and Wright (2006).

2.3 Cross Validation

In any model fitting procedure (including linear (mixed) models) it is desired to assess the quality of the model, measured for instance by MSE in the regression setting, in order to decide on which model to use before testing the model selected. It is generally easy to attain an estimate of the training error. However, the training error is not a reliable measure of the performance of the models on new data. When we have a lot of data, it is acceptable to simply split the data into training, validation and test sets. In this case, the models are trained on the training set, validated and selected based on the performance on the validation set. Lastly, the model performance is then assessed on the test set. In the case of not having an abundance of data, it is not affordable to set aside a validation set as one needs that data for training the models sufficiently. Not enough data relative to the complexity of the model is a source for overfitting. In this setting, one can use cross-validation to assess the performance of the model to select the best one. The following algorithm provides an overview of the method. $\mathbf{D}_{-k}^{train}$ is the training data leaving out the k^{th} fold and $\mathbf{D}_{k}^{train}$ is the k^{th} fold of the training data.

Algorithm 2 Cross Validation (CV)

```
    Input:
        training data $\mathbf{D}^{train}$
        a set of models $\mathcal{M}$
 1: Split $\mathbf{D}^{train}$ into $K$ folds
 2: for model in $\mathcal{M}$ do
 3:     for $k$ from 1 to $K$ do
 4:         Train model on $\mathbf{D}_{-k}^{train}$
 5:         Predict on $\mathbf{D}_{k}^{train}$ using the trained model
 6:         Store performance of the model
 7:     Average the $K$ performances of the model
 8: Select the model with best average performance over the $K$ folds
```

One could argue that it is only the inner loop that is the actual CV procedure, but Algorithm 2 shows how it is generally used. Note that if you only have one model of interest, ϕ, then $\mathcal{M} = \{\phi\}$ and one is left with only the inner loop for one model.

2.4 Bootstrap

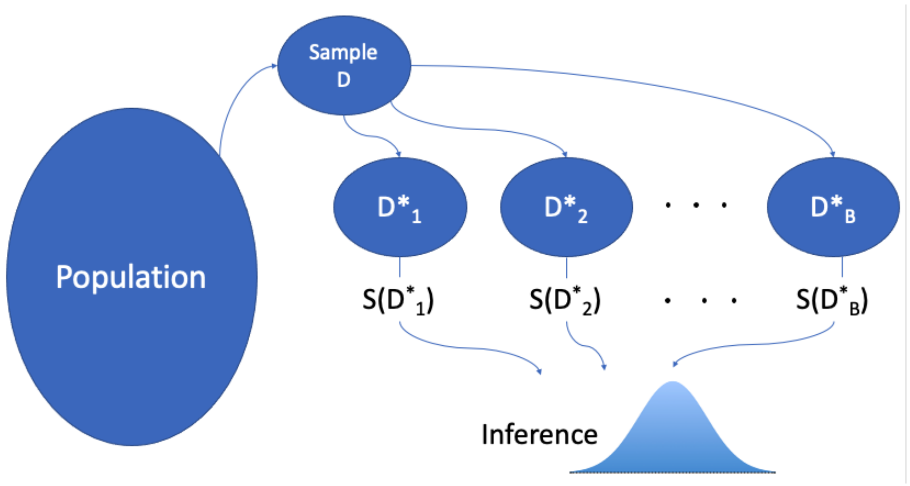


Figure 2: An overview of the bootstrap scheme, where $S(\mathbf{D}_i^*)$ is the estimate of interest from the i^{th} bootstrap sample.

For the lasso, discussed in subsubsection 2.2.2, there is no consensus concerning how one should calculate the standard errors of the parameter estimates and it is even less clear how one would do it when multiple imputation is combined with lasso. Instead, one can estimate them using the bootstrap. The bootstrap is a way to bypass problems of calculating parameter properties by rather estimating the properties from resampling. It was first presented by Efron in 1979.

The bootstrap procedure works as illustrated in Figure 2, by taking n, the number of observations, samples with replacement from the original sample. One does this B times to get B bootstrap data sets. Then we estimate the parameter of interest on all the B resampled data sets and end up with B estimates of the parameter. One can then draw inference around these estimates by calculating for example the variance of the parameter.

3 Methods

3.1 The data

The imputed data used in this thesis is mostly the same as the data that was used in Schmutz et al. (2017) and was kindly provided by the authors of the article. The data is sampled from 394 Swiss children aged two to six years, coming from 84 childcare centres located in 5 different cantons. The cantons comprise approximately 50% of the Swiss population and belong to two sociocultural areas of Switzerland, a French and a German part.

The variables were selected in advance based on previous research and theory and grouped into five types: (i) biological and demographic; (ii) psychological, cognitive and emotional; (iii) behavioural; (iv) social and cultural; and (v) environmental. A description of the 35 variables can be found in Additional file 1 in Schmutz et al. (2017). The collected data had missing values, hence Schmutz et al. performed multiple imputation by chained equations (Buuren and Groothuis-Oudshoorn, 2011). Therefore, the data used in this thesis consist of 40 data sets with imputed values. They performed the analysis with three responses: total PA (TPA), moderate-to-vigorous PA (MVPA) and SB. This thesis only considers TPA as the response, but the analysis performed could be altered to have either MVPA or SB as the response.

3.2 Statistical methods

The method for attaining the results in the thesis is described in Algorithm 3 and the notation is consistent with the notation in Section 2. There are three main procedures in the scheme: First, the optimal λ is selected; second, the coefficients and between-variances are calculated by fitting the model to all the data in the M data sets, using the selected λ and finally, apply the bootstrap to estimate the within-variance of the coefficients and calculate the total variance. $\mathcal{M}_\lambda$ is the linear model with lasso and shrinkage coefficient λ.

In the first procedure, we select the λ by applying cross-validation to a grid of λ values, $\mathbf{\Lambda}$. For each λ in $\mathbf{\Lambda}$ we fit linear models, shrunken according to the lasso with shrinkage parameter set to λ, to each $\mathbf{D}_{-k}^{(m)}$ for $m \in \{1, \dots, M\}$ before predicting on $\mathbf{D}_k^{(m)}$ (following the notation from subsection 2.3, where (m) indicate that m^{th} imputed data set). We then calculate the MSE for each of the M predictions and take the mean of the M MSEs. This is done K times before we calculate the cross-validation error (mean of the mean MSEs). Then we find the lambda that produces the lowest cross-validation error.

In Algorithm 2 the hierarchy of the loops are

```
iterate over models
  do cross validation on the current model.
```

In the first procedure of Algorithm 3 it might look like the hierarchy is the other way around, because we fit M models inside the CV loop, but given λ, the models are deterministic in all the M cases. Hence, we follow the CV algorithm as described above.

Algorithm 3 Full scheme

```
    Input:
        $M$ standardised imputed data sets $\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(M)}$
 1: procedure SELECTING $\lambda$
 2:     Initialise $\mathbf{\Lambda}$                                  ▷ a grid of $\lambda$s
 3:     Initialise cv_results                ▷ a matrix to store the combined validation
 4:     Initialise $K$                 ▷ The number of folds in $K$-fold cross validation
 5:
 6:     for $\lambda$ in $\mathbf{\Lambda}$ do
 7:         Initialise cv_error                ▷ a vector to store the cv error for $\lambda$
 8:
 9:         for k from 1 to K do
10:             Initialise validation_errors          ▷ vector to store validation errors
11:
12:             for $m$ from 1 to $M$ do
13:                 Fit $\mathcal{M}_\lambda$ to $\mathbf{D}_{-k}^{(m)}$
14:                 Predict on $\mathbf{D}_{k}^{(m)}$
15:                 Store CV error in validation_errors                  ▷ MSE is used
16:             Calculate mean CV errors
17:             Store mean CV errors in cv_error
18:         Store cv_error in cv_results
19:     Combine the CV errors across the $K$ iterations of the CV          ▷ mean is used
20:     Set $\lambda_{\text{best}}$ to the $\lambda$ producing the lowest combined CV error
21:
22: procedure MODEL FITTING
23:     Refit $\mathcal{M}_{\lambda_{\text{best}}}$ to the $M$ data sets
24:     Calculate $\bar{\boldsymbol{\beta}}_M$                  ▷ the pooled parameter estimates
25:     Calculate $B_M$                            ▷ the between imputation variance
26:
27: procedure BOOTSTRAPPED WITHIN-VARIANCE CALCULATION
28:     for $m$ from 1 to $M$ do
29:         Bootstrap $\mathbf{D}^{(m)}$ and calculate coefficients using $\lambda_{best}$
30:         Calculate $\mathbf{W}_m$              ▷ the coefficients' within-imputation variance
31:     Calculate $\bar{\boldsymbol{W}}_{\boldsymbol{M}}$             ▷ the average of the within-imputation variance
32:
33: Calculate $\mathbf{V}_M$                        ▷ the total variance of the coefficients
```

When fitting the linear model we do not have to tune any parameters and the variance of the coefficients are readily available. Thus we can discard the first procedure and the bootstrap part in Algorithm 3. We end up simply having to fit the linear model to each of the M data sets and directly calculate the $\bar{\mathbf{W}}_M$ and $\mathbf{B}_M$ to get $\mathbf{V}_M$.

Moreover, before applying Algorithm 3 it is advisable to decide on whether explanatory or predictive modelling is the main goal. When predictive modelling is of interest, a test data set should be set aside and we only use a subset of the data for training in contrast to when we do explanatory modelling. At least this is the case when one sticks to the traditional way of doing statistical modelling, but as argued in Shmueli (2010) one might want to evaluate predictive performance also when doing explanatory modelling. In Schmutz et al. (2017) the main interest lies in explanatory modelling, therefore both an analysis of the full and a reduced training data set containing 70% of the data are performed in this thesis.

	Linear model		Lasso	
	β	95% CI	β	95% CI
Intercept	-0.199	$(-0.385, -0.014)$	-0.092	$(-0.549, 0.364)$
Sex	0.107	$(0.033, 0.181)$	0.086	$(-0.069, 0.241)$
Age	0.274	$(0.199, 0.349)$	0.247	$(-0.045, 0.538)$
Birthweight	0.074	$(-0.004, 0.153)$	0.043	$(-0.114, 0.2)$
Cronic health condition	-0.231	$(-0.512, 0.051)$	-0.084	$(-0.366, 0.199)$
BMI	0.141	$(-0.032, 0.313)$	0.033	$(-0.807, 0.873)$
Gross motor skill	0.117	$(0.037, 0.197)$	0.087	$(-0.049, 0.223)$
Sibling	-0.254	$(-0.414, -0.095)$	-0.106	$(-0.332, 0.121)$
Parental BMI	-	-	0	$(-0.242, 0.242)$
SES	-	-	-0.002	$(-0.098, 0.094)$
Family structure	0.391	$(0.126, 0.657)$	0.198	$(-0.147, 0.543)$
Self-regulation	-	-	-0.002	$(-0.171, 0.167)$
Psychological difficulties	-	-	0.001	$(-0.091, 0.093)$
Emotionality temperament	-0.067	$(-0.154, 0.02)$	-0.004	$(-0.082, 0.075)$
Activity temperament	0.213	$(0.133, 0.293)$	0.185	$(0.057, 0.313)$
Shyness temperament	-	-	-0.032	$(-0.133, 0.069)$
Parenting stress	0.088	$(0.005, 0.172)$	0.03	$(-0.106, 0.167)$
Cognitive performance	-	-	0	$(-0.099, 0.099)$
Sleep duration	-	-	0	$(-0.136, 0.136)$
Play frequency	-	-	0	$(-0.167, 0.167)$
Parental sedentary behavior	-	-	0	$(-0.12, 0.119)$
Parental sports club membership	0.254	$(0.074, 0.435)$	0.084	$(-0.178, 0.346)$
pa_cat_comb21	-	-	0.011	$(-0.261, 0.283)$
Transport to childcare	-	-	0	$(-0.329, 0.329)$
Parental involvement in child physical activity	-	-	0.009	$(-0.204, 0.221)$
Parental tobacco use	-	-	0	$(-0.247, 0.247)$
Parental alcohol consumption	-0.396	$(-0.74, -0.053)$	-0.212	$(-0.865, 0.44)$
Time outdoors	0.122	$(0.045, 0.199)$	0.081	$(-0.138, 0.299)$
Fixed toys	0.105	$(0.021, 0.189)$	0.082	$(-0.196, 0.36)$
Portable toys	0.048	$(-0.036, 0.132)$	0.025	$(-0.139, 0.189)$
Days at childcare	-	-	-0.008	$(-0.156, 0.14)$
Living area per person	-0.08	$(-0.161, 0.001)$	-0.009	$(-0.147, 0.129)$
Neighborhood safety	-0.096	$(-0.18, -0.012)$	-0.059	$(-0.178, 0.061)$
Dog	-	-	-0.037	$(-0.406, 0.332)$
Season	0.26	$(0.088, 0.433)$	0.122	$(0.095, 0.148)$
Region	-	-	0	$(-0.224, 0.224)$

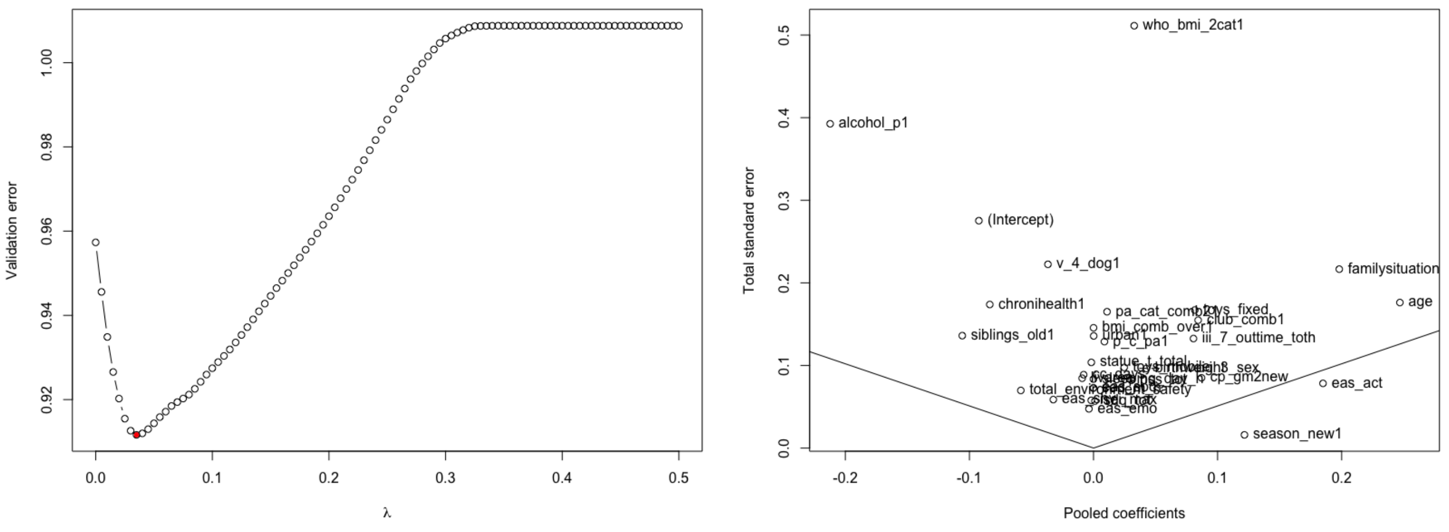
Table 1: The table presents the variables' coefficients and a 95% confidence interval.

4 Results

From Table 1, we see that if we follow the procedure used in Schmutz et al. (2017) the absolute value of the estimates are considerably larger than when we apply the lasso method presented in the thesis. Hence, the results confirm that using p-value based variable selection, before refitting the model to the subset of variables, overestimates parameter estimates. Additionally, we observe from the narrow confidence intervals, which indicate a too small uncertainty, that the p-value based approach is too confident about the estimates.

In addition, from Table 1 we see that *activity temperament* and *season* are the only two variables with 95% confidence intervals not containing 0 when we apply lasso in the full data set case. Furthermore, the two above-mentioned covariates are considered significant in the final linear model. Moreover, all the variables shrunken to 0 by the lasso were not considered significant at $\alpha = 0.1$ in the first linear model and those that are not set to 0 have confidence interval centred close to 0. Additionally, the R^2 decomposition of the ordinary linear model with *TPA* as response, using the LMG method (Gold et al., 1980), attribute the largest portion of R^2 to *age* and second largest to *activity temperament* (Schmutz et al., 2017).

In the case of using the full data set, the mean cross-validation errors for each λ in $\mathbf{\Lambda}$ is plotted in Figure 3a, where the one resulting in the lowest mean MSE is marked red. Note that when λ just above 0.3, all the coefficients are set to zero and we only have an intercept. The coefficients and estimated standard errors are illustrated in Figure 3b. The figure show that only the variables *Activity temperament* and *Season* are below the $\alpha = 0.05$ threshold.

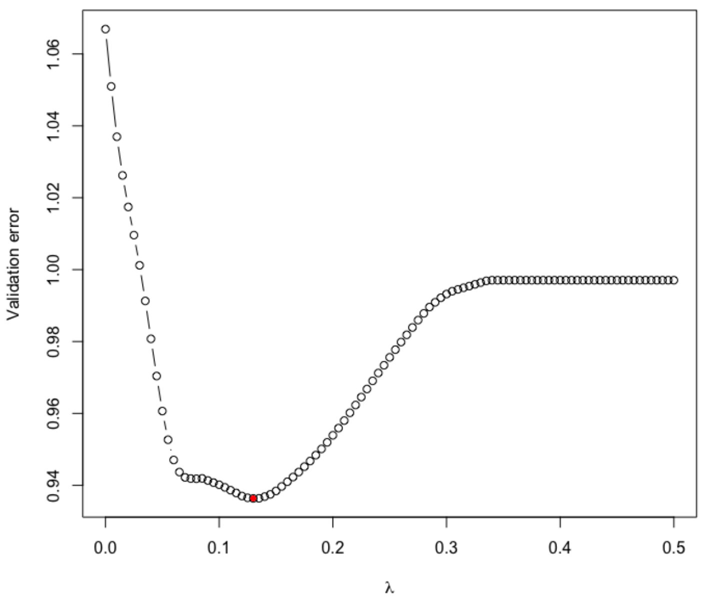


(a) A plot of λ versus the mean cross validation error.

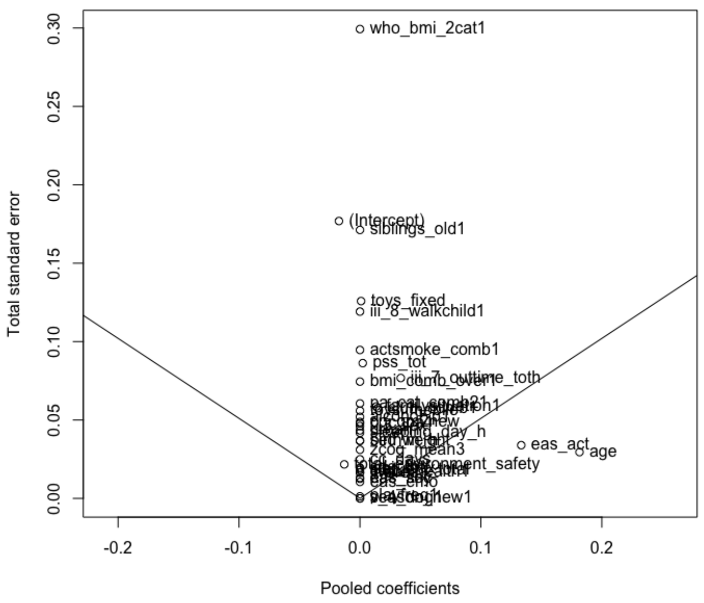
(b) The pooled coefficients against the total standard errors.

Figure 3: Plots of the full data set case. The red dot in (a) marks λ_{best} which is the λ used in (b). The lines in (b) are given by $y = |x|/1.96$ indicating the threshold for significance at $\alpha = 0.05$.

Performing model fitting on a subset of the data generally produces somewhat different results, which is also the case here. When using a training data set consisting of 70% of the data, we get the lasso path illustrated in Figure 4a. The selected shrinkage parameter, λ_{best}, is somewhat larger than when the full data set is used and therefore the variables are more severely shrunken. Additionally, the intercept model (all coefficients are set to zero) has a lower error than the full model where non of the variables are shrunken. In other words, predicting the average of all observations perform better, in terms of mean cross-validation error, than the full model. The coefficients and estimated standard errors for λ_{best} using the training set is illustrated in Figure 4b. Many of the variables are set to zero and now *Activity temperament* and *Age* are the two variables below the $\alpha = 0.05$ threshold. By setting aside 30% of the data to a test set one experience a reduction in the power



(a) A plot of λ versus the mean cross validation error.



(b) The pooled coefficients against the total standard errors.

Figure 4: Plots of the reduced data set case. The red dot in (a) marks λ_{best} which is the λ used in (b). The lines in (b) are given by $y = |x|/1.96$ indicating the threshold for significance at $\alpha = 0.05$.

of the model. This reduction can be seen by comparing Figure 3a and Figure 4a. In the former, the validation error of the full model ($\lambda = 0$) is lower than that of the latter. This is expected as overfitting can be thought of as a function of the model complexity and amount of data. A too complex model on too little data overfits, while a non-complex model on a large data set does generally not. In both cases we observe overfitting, but the model that has fewer instances to train on overfits more.

5 Discussion

In this thesis p-value based variable selection and parameter estimation are compared to an implementation of a lasso based scheme for performing explanatory analysis on multiple imputed data sets. When using the p-value based method we observed a considerable overestimation of the parameter estimates.

p-values are historically important and still have a place in statistical modelling, but one should be aware of the shortcoming, what information the p-values provide and when not to use p-values. The results showed that other methods should be applied instead of p-value based methods for variable selection and determining relations between variables. The lasso is one of many such methods. We have seen that variables are overestimated in terms of significance when p-values are used to evaluate the significance of the relationship between variables and the response of the analysis. The use, some might say overuse, of p-values has multiple reasons: Firstly, the knowledge and experience with other methods might be limited, leading researchers to stick to what they know, typically p-values. Secondly, the motivation and need for being published might consciously or subconsciously motivate researchers to use methods that output findings and relations between variables. Overestimating variables' effect size with p-values would increase the chance of the paper being published and it might therefore be the method of choice for some.

When applying Algorithm 3, whether one uses the full data set or a subset of the data in form of a training set, changes the results, as we saw above. The reduction in power from setting aside a test set affects the choice of λ_{best}. When we use less data, the validation error for models with relatively small λs increase, but the models with larger λs are not as affected. This leads to selecting a larger λ as λ_{best}. To illustrate this, one can think of a heavy ball lying on a ribbon, where the ribbon is the lasso path and the ball is the location of λ_{best}. If you lift the left side of the ribbon while the right side remains still, the ball will roll to the right before it settles, similarly, λ_{best} rolls to the right in the plot before it settles at a larger value, in general.

The increase in λ_{best} has a very clear effect on the coefficients. When we use the training data we end up selecting a larger λ_{best}, shrinking the coefficients more and setting more of them to zero in comparison to the case of the full data. This is illustrated by Figure 3b and Figure 4b for the cases of using the full data and training data, respectively.

When constructing Algorithm 3 I chose to find a global λ_{best} as I initially believed it was the only logical possibility. However, the hierarchy of the loops in the first procedure could have been flipped. We could rather have used cross-validation to find the best shrinkage parameter for a linear model on each of the M data sets. There would be one main advantage with that approach. The time needed for conducting the first procedure in Algorithm 3 would be reduced. The reason for this is that many of the functions that perform lasso, for instance, glmnet from the *glmnet* package which I used, do cross-validation to assess the performance of λs on an appropriate $\mathbf{\Lambda}$ automatically. Therefore, when I applied the function glmnet function in Algorithm 3 I did the alternative method on each element in $\mathbf{\Lambda}$ and did CV on each of those. Hence, by selecting one λ for each of the M data sets, one would expect the time of the first procedure to be reduced by a factor of $\frac{1}{K \cdot |\mathbf{\Lambda}|}$, where $|\mathbf{\Lambda}|$ is the number of elements in $\mathbf{\Lambda}$. As an example, assume that the original procedure takes eight minutes and 24 seconds when 5-fold CV is performed and $|\mathbf{\Lambda}| = 101$, then the time would be reduced to approximately one second. Although one loses some transparency, because a model with one global λ is more transparent than a model with M λs, I would argue that the advantage of the reduction in time outweighs this. I have not evaluated the difference in performance as this is outside the scope of this thesis, but future work should include this evaluation. I do not believe the difference would be large, because when we average the models from the M imputed data sets in both cases the information from each imputed data set would be incorporated in the final pooled model. Therefore, I would recommend the alternative approach for personal implementations.

References

Buuren, Stef van and Karin Groothuis-Oudshoorn (2011). 'mice: Multivariate Imputation by Chained Equations in R'. In: *Journal of Statistical Software* 45. URL: https://www.jstatsoft.org/article/view/v045i03.

Efron, Bradley (1979). 'Bootstrap Methods: Another Look at the Jackknife'. In: *The Annals of Statistics* 7 (1). URL: https://www.jstor.org/stable/2958830?seq=1#metadata_info_tab_contents.

Efroymson, M. A. (1960). 'Multiple Regression Analysis,' in: *Mathematical Methods for Digital Computers.*

Freedman, David A (1983). 'A Note on Screening Regression Equations.' In: *The American Statistician* 37 (2). URL: https://www.jstor.org/stable/2685877?seq=1#metadata_info_tab_contents.

Gold, Ruth Z., Richard Harold Lindeman and Peter Francis Merenda (1980). *Introduction to Bivariate and Multivariate Analysis.* Scott Foresman & Co. ISBN: 9780673150998.

Hastie, Trevor, Robert Tibshirani and Jerome Friedman (2009). *The Elements of Statistical Learning.* 2nd ed. Springer Series in Statistics. Springer Science+Business Media. ISBN: 9780387848570.

Hoerl, Arthur E. and Robert W. Kennard (1970). 'Ridge Regression: Biased Estimation for Nonorthogonal Problems'. In: *Technometrics* 12 (1). URL: https://www.jstor.org/stable/1267351?seq=1#metadata_info_tab_contents.

Hubbard, Raymond and R. Murray Lindsay (2008). 'Why P Values Are Not a Useful Measure of Evidence in Statistical Significance Testing'. In: *Theory and Psycology* 18 (1). URL: https://www.researchgate.net/publication/240281208_Why_P_Values_Are_Not_a_Useful_Measure_of_Evidence_in_Statistical_Significance_Testing.

Ioannidis, John P. A. (2005). 'Why Most Published Research Findings Are False'. In: URL: https://doi.org/10.1371/journal.pmed.0020124.

Little, Roderick J. and Donald B. Rubin (2002). *Statistical Analysis with Missing Data.* Wiley Series in Probability and Statistics. John Wiley & Sons. ISBN: 0471183865.

Molnar, Christoph (2020). *Interpretable Machine Learning.* lulu.com. ISBN: 9780244768522.

Nahm, Francis Sahngun (2017). 'What the P values really tell us'. In: *The Korean Journal of Pain.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5665734/.

Nocedal, Jorge and Stephen J. Wright (2006). *Numerical Optimization.* Springer Series in Operations Research. Springer Science+Business Media. ISBN: 9780387303031.

R Core Team (2021). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Santosa, Fadil and William W. Symes (1986). 'Linear Inversion of Band-Limited Reflection Seismograms'. In: *Journal on Scientific and Statistical Computing* 7 (4), pp. 1307–1330. URL: https://epubs.siam.org/doi/10.1137/0907087.

Schmutz, Einat A. et al. (2017). 'Correlates of preschool children's objectively measured physical activity and sedentary behavior: a cross-sectional analysis of the SPLASHY study'. In: *International Journal of Behavioral Nutrition and Physical Activity* 14. URL: https://ijbnpa.biomedcentral.com/articles/10.1186/s12966-016-0456-9#citeas.

Shmueli, Galit (2010). 'To Explain or to Predict?' In: *Statist. Sci.* 25 (3). URL: https://projecteuclid.org/journals/statistical-science/volume-25/issue-3/To-Explain-or-to-Predict/10.1214/10-STS330.full.

Stone, Mervyn (1974). 'Cross-Validatory Choice and Assessment of Statistical Predictions'. In: *Journal of the Royal Statistical Society* 36 (2). URL: https://www.jstor.org/stable/2984809?seq=1#metadata_info_tab_contents.

Tibshirani, Robert (1996). 'Regression Shrinkage and Selection via the Lasso'. In: *Journal of the Royal Statistical Society* 58. URL: https://www.jstor.org/stable/2346178.

Appendix

A Code – Full scheme

In the code below, df_train can either be the full data or a subset depending on whether you are interested in prediction or not.

```r
# Start by seeing that the data is correct
head(df_train)

###### I split the 40 data sets into 40 elements of a list ######

# Here we will split the training data set into the respective 40 sets and make a
↪  list of them
n = dim(df_train)[1]/40
df_list = list(df_train[0:n, ])
for (i in 1:39){
  df_list[[i+1]] = df_train[(n*i + 1):(n*(i+1)), ]
}


###### Now lasso on the 40 data sets with the same lambda over a grid of lambdas
↪  and select the best using CV ######

# Split the M data sets into K folds, with equal split for all data sets
# (each individual is in the same place in all 40 data sets)

# Set seed for reproducability
set.seed(135246)
# We shuffle the observation to make sure the folds are split randomly
shuffle = sample(1:n, n)
# Make a grid of lambdas, test which interval of lambdas are appropriate
lambda_grid = c(seq(from = 0, to = 0.5, 0.005))


# K gives the type of CV
K = 5
# A list to store values for the different models
cv_results = matrix(nrow = K, ncol = length(lambda_grid))

# K-fold crossvalidation to choose the approriate lambda
for (k in 1:K){
  # Create fold k from the shuffled values
  fold_k = shuffle[((k-1)*n/K + 1):(k*n/K)]

  cv_error = c()
  for (lambda in lambda_grid){
    # Now we do lasso for each data set on the specific lambda and store the
    ↪  coefficients

    validation_errors = c()
    for (df in df_list){

      x = model.matrix(tpa15_a1_cpm ~., df[-fold_k, ])
      y = df[-fold_k, ]$tpa15_a1_cpm
      fit_lasso = glmnet(x, y, alpha = 1) # alpha = 1 gives lasso and is the
      ↪  default of glmnet()
      lasso_coefs = coef(fit_lasso, s = lambda)

      ##### validate her on the k-th fold #####
      x = df[fold_k, -dim(df)[2]] # remove the response
      y = df[fold_k, ]$tpa15_a1_cpm
      # We convert x into a matrix and add 1s on a new first column because we
      ↪  have an intercept
      x = cbind(rep(1, nrow(x)),data.matrix(x))
      y_hat = x %*% lasso_coefs[-2]
      # Append the validation error/MSE to the vector
      validation_errors = c(validation_errors, sqrt(apply((y_hat - y)^2, 2,
      ↪  mean)))
    }
    # Combine the validation error
    combined_mse = mean(validation_errors^2)

    cv_error = c(cv_error, combined_mse)
    # To see the progress of the algorithm
    print(lambda)
  }
  # To see the progress of the algorithm
  print(k)
  cv_results[k, ] = cv_error
}



##### Combine errors for the lambdas across folds and select the best lambda

# Make a data frame to use lambdas as column names
val_error_df = data.frame(cv_results)
colnames(val_error_df) = lambda_grid

# Mean of the errors across the folds
comb_val_err = apply(val_error_df, 2, mean)
# Select the optimum lambda
lambda_best = lambda_grid[which.min(comb_val_err)]
best_cv_err = min(comb_val_err)

##### plot the optimum lambda #####
plot(lambda_grid, comb_val_err, type = 'b', ylab = "Validation error",
        xlab = bquote(lambda))
points(lambda_best, best_cv_err, pch = 20, col = 'red')


##### Fit the model using the best lambda #####

# Initialise a matrix to store coefficients
coef_matrix = matrix(nrow = length(df_list), ncol = dim(df_list[[1]])[2])

i = 0
# Now we do lasso for each data set on lambda_best and store the coefficients
validation_errors = c()
for (df in df_list){
  i = i + 1

  x = model.matrix(tpa15_a1_cpm~., df)
  y = df$tpa15_a1_cpm
  fit_lasso = glmnet(x, y, alpha = 1)
  lasso_coefs = coef(fit_lasso, s = lambda_best)


  # Append the coefficients to the i-th row. We remove the second element, which
  ↪  is a second intercept always equal 0
  coef_matrix[i, ] = t(lasso_coefs[-2])
}

##### Combine using Rubin's rule #####

# average the coefficients from the 40 data sets
pooled_coefs = apply(coef_matrix, 2, mean)
# Get the between-standard-error for the coefficients
between_var = apply(coef_matrix, 2, var)


###### Bootstrap to get the within-imputed-variance ######
# We make a bootstrap sample for each data frame in df_list

# The number of bootstrap samples to estimate coefficients
boot_length = 100

# Matrix to store values
within_var_matrix = matrix(nrow = length(df_list), ncol = length(pooled_coefs))

i = 0
for (df in df_list){
  i = i + 1
  df_temp = df # This is done to be able change the data frame in the developing
  ↪  phase, without changing df directly

  # A matrix for storing coefficients
  boot_coef_matrix = matrix(nrow = boot_length, ncol = length(pooled_coefs))

  for (boot_iteration in 1:boot_length){
    # We sample bootstrap indices
    boot_indices = sample(dim(df_temp)[2], dim(df_temp)[2], replace = TRUE)
    # Make the bootstrap sample
    boot_sample = df_temp[boot_indices, ]
    # Make a model matrix using the bootstrap samples
    x = model.matrix(tpa15_a1_cpm~., boot_sample)
    # y is the responses in the bootstrap sample
    y = boot_sample$tpa15_a1_cpm
    # do lasso on x and y
    fit_lasso = glmnet(x, y, alpha = 1)
    # store coefficient values in the matrix
    boot_coef_matrix[boot_iteration, ] = as.vector(coef(fit_lasso,
            s = lambda_best))[-2]
  }
  # Store the within_variance estimate
  within_var_matrix[i, ] = apply(boot_coef_matrix, 2, var)
}
# The mean of the bootstrap variances
within_var = apply(within_var_matrix, 2, mean)

##### Calculate the total variance/std for the coefficients #####
```

```
# Calculate total variance
tot_var = within_var + between_var + between_var/(length(df_list))
# Calculate standard errors
pooled_se = sqrt(tot_var)
# Combine the results
result = data.frame(cbind(pooled_coefs, pooled_se))
# Add rownames, but remove the second element which is an intercept always equal
↪  to 0
rownames(result) = rownames(lasso_coefs)[-2]
# Print the result
result
```

B Code – Linear model

This is the code producing the results in the *Linear model* column in Table 1.

```
# Set seed for reproducability
set.seed(135)

# Matrices to store values in
coef_matrix = matrix(nrow = ncol(df_list[[1]]), ncol = length(df_list))
coef_var_matrix = matrix(nrow = ncol(df_list[[1]]), ncol = length(df_list))

i = 0
for (df in df_list){
  i = i + 1
  # Fit a linear model
  my_lm = lm(tpa15_a1_cpm~., data = df)
  # Extract the coefficents
  lm_coefficients = my_lm$coefficients
  # Store coefficients in i-th column
  coef_matrix[, i] = lm_coefficients

  # Extract standard error and elementwise square them to get the variance
  coef_var = summary(my_lm)$coefficients[, 2]^2
  # Store variances in i-th column
  coef_var_matrix[, i] = coef_var
}

######## Combine using Rubin's rule ##########

# average the coefficients from the 40 data sets
pooled_coefs = apply(coef_matrix, 1, mean)
# Get the between-standard-error for the coefficients
between_var = apply(coef_matrix, 1, var)
# Get the within variance
within_var = apply(coef_var_matrix, 1, mean)


##### Calculate the total variance/std for the coefficients #####

# total variance = within_var + between_var * corrected_degrees_of_freedom
tot_var = within_var + between_var + between_var/(length(df_list))
# get pooled standard error
pooled_se = sqrt(tot_var)

result = data.frame(cbind(pooled_coefs, pooled_se))
```

```
# Add rownames, but remove the second element which is an intercept always equal
↪  to 0
rownames(result) = names(lm_coefficients)

# calculate confidence intervals
ci_95_low = round(result$pooled_coefs - qnorm(0.95)*result$pooled_se, 3)
ci_95_high = round(result$pooled_coefs + qnorm(0.95)*result$pooled_se, 3)
# combine the two into one input
ci_95 = glue("({ci_95_low}, {ci_95_high})")

# Add the confidence interval to the results data frame
result$ci_95 = ci_95

# Get everything in the right shape
table = data.frame(matrix(c(round(result[, 1], 3), result[, 3]), ncol = 2))
rownames(table) = rownames(result)
# print table of results
table


##### Perform analysis on a subset of the variables #####
# Here we use the variables that are significant i.e. those variables
# with confidence intervals not containing 0

# Set seed for reproducability
set.seed(13523532)
variables = c("i_3_sex", "age", "birthweight", "cp_gm2new", "familysituation",
              "eas_act", "club_comb", "alcohol_p", "iii_7_outtime_toth",
              "toys_fixed", "total_environment_safety", "season_new")


formula = paste("tpa15_a1_cpm ~ ", paste(variables, collapse=" + "), sep = "")

# We make some matrices to store coefficients and their respective variances
coef_matrix = matrix(nrow = length(variables)+1, ncol = length(df_list))
coef_var_matrix = matrix(nrow = length(variables)+1, ncol = length(df_list))
i = 0
for (df in df_list){
  i = i + 1
  my_lm = lm(formula, data = df)
  lm_coefficients = my_lm$coefficients
  coef_matrix[, i] = lm_coefficients

  coef_var = summary(my_lm)$coefficients[, 2]^2
  coef_var_matrix[, i] = coef_var
}
coef_matrix


######## Combine using Rubin's rule ##########
# What are the criterias for using Rubin's rule? Normal distribution of
↪  variables?

# average the coefficients from the 40 data sets
pooled_coefs = apply(coef_matrix, 1, mean)
# Get the between-standard-error for the coefficients
between_var = apply(coef_matrix, 1, var)
# Get the within variance
```

```r
within_var = apply(coef_var_matrix, 1, mean)


##### Calculate the total variance/std for the coefficients #####

tot_var = within_var + between_var + between_var/(length(df_list)) # within_var +
↪  between_var * corrected_degrees_of_freedom
pooled_se = sqrt(tot_var)


result = data.frame(cbind(pooled_coefs, pooled_se))
# Add rownames, but remove the second element which is an intercept always equal
↪  to 0
rownames(result) = names(lm_coefficients)

# calculate confidence intervals
ci_95_low = round(result$pooled_coefs - qnorm(0.95)*result$pooled_se, 3)
ci_95_high = round(result$pooled_coefs + qnorm(0.95)*result$pooled_se, 3)
# combine the two into one input
ci_95 = glue("({ci_95_low}, {ci_95_high})")

result$ci_95 = ci_95

# Get everything in the right shape
table = data.frame(matrix(c(round(result[, 1], 3), result[, 3]), ncol = 2))

rownames(table) = rownames(result)
table


##### Perform analysis on a subset of the variables #####
# Here we use the variables that are significant i.e. those variables
# with confidence intervals not containing 0

# Set seed for reproducability
set.seed(13523532)

# Here we extract the variables that was significant at alpha = 0.1
variables = c("i_3_sex", "age", "birthweight", "chronihealth", "who_bmi_2cat",
              "cp_gm2new", "siblings_old", "familysituation", "eas_emo",
              "eas_act", "pss_tot", "club_comb", "alcohol_p",
              "iii_7_outtime_toth","toys_fixed", "toys_mobile", "livarea",
              "total_environment_safety", "season_new")


formula = paste("tpa15_a1_cpm ~ ", paste(variables, collapse=" + "), sep = "")

# We make some matrices to store coefficients and their respective variances
coef_matrix = matrix(nrow = length(variables)+1, ncol = length(df_list))
coef_var_matrix = matrix(nrow = length(variables)+1, ncol = length(df_list))
i = 0
for (df in df_list){
  i = i + 1
  my_lm = lm(formula, data = df)
  lm_coefficients = my_lm$coefficients
  coef_matrix[, i] = lm_coefficients

  coef_var = summary(my_lm)$coefficients[, 2]^2
  coef_var_matrix[, i] = coef_var
}
coef_matrix


######## Combine using Rubin's rule ##########
# What are the criterias for using Rubin's rule? Normal distribution of
↪  variables?

# average the coefficients from the 40 data sets
pooled_coefs = apply(coef_matrix, 1, mean)
# Get the between-standard-error for the coefficients
between_var = apply(coef_matrix, 1, var)
# Get the within variance
within_var = apply(coef_var_matrix, 1, mean)


##### Calculate the total variance/std for the coefficients #####

tot_var = within_var + between_var + between_var/(length(df_list)) # within_var +
↪  between_var * corrected_degrees_of_freedom
pooled_se = sqrt(tot_var)


result = data.frame(cbind(pooled_coefs, pooled_se))
# Add rownames, but remove the second element which is an intercept always equal
↪  to 0
rownames(result) = names(lm_coefficients)

# calculate confidence intervals
ci_95_low = round(result$pooled_coefs - qnorm(0.95)*result$pooled_se, 3)
ci_95_high = round(result$pooled_coefs + qnorm(0.95)*result$pooled_se, 3)
# combine the two into one input
ci_95 = glue("({ci_95_low}, {ci_95_high})")

result$ci_95 = ci_95

# Get everything in the right shape
table = data.frame(matrix(c(round(result[, 1], 3), result[, 3]), ncol = 2))

rownames(table) = rownames(result)
table
```