Variable importance measures for random forests

Cindy Boon

Supervisor: Dr. N. Parolya

Msc Applied Mathematics
Faculty Electrical Engineering, Computer Science and Mathematics
TU Delft

Abstract

Measuring variable importance is often a difficult task: among others models can be complex and covariates can interact with each other and can be correlated. This study focuses on two questions: First, what should be the theoretical measure of variable importance under a given data-generating model? And second, what are the best estimates of these theoretical measures? Two theoretical measures and some corresponding estimates are presented of which one is the well-known random forests variable importance measure (Breiman (2001a)). A simulation study is done for both linear and nonlinear models to find out what are the best estimates of variable importance measures for given data-generating models. Most measures struggle when covariates are correlated, but make an improvement in performance when the number of split variables is tuned.

Contents

Chapter 1

Introduction

Random forests are very popular methods for statistical prediction in this age of big data and complex model structures. The method is among others well known for being able to deal with large p and small n problems and for its way of measuring variable importance. The importance of variables can be measured in tons of different ways and different kinds of data sets ask for different ways of measuring. An ultimate goal would be to find a measure that can be applied to many different kinds of data sets.

Although random forests are widely used for prediction and measuring variable importance, little is known about theoretical properties. Several theoretical results are out there of which some will be presented in this work. The main focus of this work is experiments that are based on curiosities that arose from theory.

The objective of this research is to investigate the behaviour of variable importance measures for random forests on a range of data sets with numerical response: data sets with different amounts of observations and dimensions, with either correlated or uncorrelated covariates, with various distributions of the covariates and with either linear or nonlinear underlying models.

The outline of this work is as follows: Chapter 2 lays down a theoretical framework about statistical prediction, tree-based methods and consistency of random forests. Chapter 3 considers the questions: What should be the measure of variable importance under a given data-generating model, which measure gives us the most correct picture and what is the optimal measure? Various theoretical measures of variable importance will be presented and a study of best estimates of these measures will be done. The theoretical difference between those measures is considered and the influence of correlated covariates on these measures of variable importance. In chapter 4 are presented the results of experiments on the behaviour of variable importance measures based on simulated data sets. Finally, there are conclusions and a discussion about possible future work.

This project is a cooperation between TU Delft and RIVM, the Dutch National Institute for Public Health and Environment.

Chapter 2

Theoretical framework

2.1 Statistical prediction

The goal of statistical prediction is to solve problems by discovering patterns in available data in order to make valuable predictions. We distinguish two kinds of goals: the underlying goal and the ultimate goal.

Oftentimes the ultimate goal is to make predictions of a response Y with the knowledge of corresponding covariates $\mathbf{X}$, but the ultimate goal could also be to measure the importance of covariates. The underlying goal is to estimate or 'learn' or simply approximate an unknown mechanism that is connecting one or more input variables or covariates $\mathbf{X}$ to their output variable or response Y by some function $\Pi : \mathscr{X} \to \mathscr{Y}$, a predictor, where $\mathscr{X}$ is used for the space of input values and $\mathscr{Y}$ for the space of output values. In order to construct the predictor a training data set $\mathscr{S}_n = ((\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n)))$ with observations is used that include both $\mathbf{X}_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,p})$ and Y_i for $i = 1, \ldots, n$ with p the number of covariates and n the number of observations in the training data set (**?**). For statistical prediction normally it is assumed that all pairs $(\mathbf{X}_i, Y_i)$ in $\mathscr{S}_n$ and the 'new cases' $(\mathbf{X}, Y)$, of which one only knows $\mathbf{X}$ and Y is to be predicted, are independent and identically distributed (iid). This is useful for theoretical reasons, but is often not completely realistic in practice. Even when this assumption is not met, there are prediction methods that can be applied and work well more generally.

Data in $\mathbf{X}$ and Y can be numerical or nominal, which is also called categorical. Numerical data can be continuous or discrete, nominal data are discrete with no numerical relationship between the different categories. When the output variable Y is numerical, the prediction problem is called a regression problem. When Y is nominal, the prediction problem is called a classification problem.

Then a prediction study consists of roughly the following steps: with at least a rough objective in mind one investigates the available data and possibly changes the objective to a more realistic one. One looks into different methods to apply to the data and with those methods constructs predictors. One looks into ways to assess the quality of different predictors, sets theoretical goals and tries to go about these goals empirically. If one is not satisfied with the quality of the predictor(s) in terms of accuracy, computational efficiency, interpretability, etc., one possibly researches and tries to fit other methods to the data and assesses the quality of the new

predictor(s). If one is satisfied with one of the obtained predictors, one can make predictions.

The following sections will be on methods for constructing predictors and assessing a predictor.

2.1.1 Methods for constructing predictors

There are many methods out there. Which one to use depends on the objective and data set that one has. A first distinction that can be made among the methods is parametric and nonparametric methods. This will be explained by means of an article from Breiman called 'Two cultures' (Breiman (2001a)).

Breiman distinguishes two approaches to discover the unknown mechanism that is connecting $\mathbf{X}$ to Y. The first approach is to try to model the unknown mechanism and thereby assume that the data are generated by a given stochastic data model. The second approach is to not try to model the unknown mechanism but use an algorithmic model instead. These two approaches for modeling are also referred to as parametric and nonparametric modeling, respectively. For parametric modeling it is necessary to have at least some knowledge of the conditional distribution of Y given $\mathbf{X}$ since one has to come up with an appropriate model for the data. This is easiest when the training data set is small and when the dimension p of $\mathbf{X}$ is small.

With the evolution of the computer a lot more information has become available. Data sets can now be larger than they have ever been and so can be the dimension p of $\mathbf{X}$. One can imagine that the larger p is, the more difficult it gets to guess the relationship or function Π that connects $\mathbf{X}$ to Y. Parametric modeling can be favored in experimental research when the relationship that one tries to model is rather simple, but something else is needed when p is large and n might even be small. A possible solution could be to work with a suboptimal predictor or try to reduce the number of covariates p. Nonparametric methods can perform well when the training data set is large, p is not small and little is known of the conditional distribution of Y given $\mathbf{X}$. Nonparametric modeling can be favored when one has complex observational data. Examples of parametric methods are linear and quadratic regression, linear and quadratic discriminant analysis and logistic regression. Examples of nonparametric methods are the k-nearest neighbor algorithm and tree-based methods like random forests.

A lot has been said about these two different approaches. To be able to make a good choice and be aware of the dangers, Breiman first argues that the field of statistical prediction should cherish the use of parametric models, but should embrace the use of nonparametric models, too. In other words, Breiman is open to both kinds of methods. In his article (Breiman (2001b)) he mainly points at three matters to consider: the multiplicity of good models, the conflict between simplicity and accuracy, and (the curse of) dimensionality.

Multiplicity of good models: There is often a multitude of different predictors giving about the same minimum error rate, while at the same time they can be distant in terms of the form of the model. The conclusions, like variable importance, can be very different among these different models. Breiman argues that more complex models can be needed to overcome the multiplicity problem.

The conflict between simplicity and accuracy: Breiman argues for accuracy over interpretability, which does often result in the use of more complex and nonparametric methods. Hand (2006) argues that the use of complex models often only leads to small gains. With the

move from a simple model to a complex model one often throws away a lot of predictive power, and also, the small gains can even be misleading since the apparent problem that is being modeled does not have to match the real problem, also known as overfitting. Hand (2006) is pro investigating new and complex models, but warns against the use of complex models.

Curse of dimensionality: This has for a long time been seen as a curse and the solution then was to reduce dimensionality. But reducing dimensionality also reduces the amount of information available for prediction. Recent work has shown that dimensionality can also be a blessing Breiman (2001b).

These three matters are good to be aware of and will result in different choices for different objectives and different data sets.

2.1.2 Assessing the quality of a predictor

If the true response is Y and the output of the predictor is $\hat{Y}$, then most of the time these are not the same. The closer they are to each other, the better the quality of the predictor.

The prediction error can be measured as

$$e\left(\Pi_{\mathscr{S}_n}\right) \coloneqq \mathbb{E}[L(Y,\hat{Y}|\mathscr{S}_n] \equiv \mathbb{E}[L(Y,\Pi_{\mathscr{S}_n}(\mathbf{X}))|\mathscr{S}_n], \tag{2.1}$$

where L is some loss function on Y and $\hat{Y} = \Pi_{\mathscr{S}_n}$ is a predictor based on a random training set $\mathscr{S}_n$. There are different loss functions and different optimal predictors of Y based on $\mathbf{X}$ for numerical and categorical Y (Ferreira et al. (2019)).

First theoretical measures of accuracy will be presented and then estimates will be studied.

Measures of accuracy

Three well known measures of accuracy based on three different loss functions are considered: The Mean Square(d) Error (MSE) and Mean Absolute Error (MAE) for regression problems and numerical Y and then Probability of Misclassification (PMC) for classification problems and nominal Y.

Numerical Y: Two loss functions are presented and four measures of error: MSE, MAE, R^2 and PEV where the last two are related to each other and to the first one. The first loss function for numerical Y considered is $L(Y,\hat{Y}) = \left(Y-\hat{Y}\right)^2$. This loss function defines the mean square(d) error (MSE) of $\Pi_{\mathscr{S}_n}$:

$$\mathrm{MSE}\left(\Pi_{\mathscr{S}_n}\right) \coloneqq \mathbb{E}\left[\left(Y-\Pi_{\mathscr{S}_n}\right)^2|\mathscr{S}_n\right]. \tag{2.2}$$

The best predictor of Y that is a function of $\mathbf{X}$ in the sense of MSE is $\Pi(\mathbf{X}) \coloneqq \mathbb{E}[Y|\mathbf{X}]$. This conditional mean and theoretical value is what one is aiming for to estimate as good as possible. This is proven at the end of this section in Proof 1 and 2. The corresponding $\mathrm{MSE}(\Pi(\mathbf{X})) \equiv \mathbb{E}\left[(Y-\Pi(\mathbf{X}))^2\right] = \mathbb{E}\left[\epsilon^2\right] = \mathrm{Var}(\epsilon) = \sigma^2$ is the lower bound for the mean squared error of any other predictor of Y.

The second loss function for numerical Y that is considered is $L(Y,\hat{Y}) = |Y - \hat{Y}|$. This loss function defines the mean absolute error (MAE) of $\Pi_{\mathscr{S}_n}$:

$$\text{Mae}\left(\Pi_{\mathscr{S}_n}\right) \coloneqq \mathbb{E}\left[|Y - \Pi_{\mathscr{S}_n}||\mathscr{S}_n\right]. \tag{2.3}$$

The best predictor of Y that is a function of $\mathbf{X}$ in the sense of MAE is $\Pi(\mathbf{X}) \coloneqq \text{med}[\mathbf{X}]$. This conditional median and theoretical value is what one is aiming for to estimate as good as possible. This is proven at the end of this section in Proof 3.

The third loss function for numerical Y that is considered is $L(Y,\hat{Y}) = Y - \hat{Y}$. This loss function defines the bias of $\Pi_{\mathscr{S}_n}$:

$$\text{Bias}\left(\Pi_{\mathscr{S}_n}\right) \coloneqq \mathbb{E}\left[Y - \Pi_{\mathscr{S}_n}|\mathscr{S}_n\right]. \tag{2.4}$$

Related to the MSE is the proportion of explained variance (PEV) of $\Pi_{\mathscr{S}_n}$, which is a 'standardized' version of MSE:

$$\text{Pev}\left(\Pi_{\mathscr{S}_n}\right) \coloneqq 1 - \frac{\text{MSE}\left(\Pi_{\mathscr{S}_n}\right)}{\text{Var}(Y)} = 1 - \frac{\text{MSE}\left(\Pi_{\mathscr{S}_n}\right)}{\text{MSE}\left(\Pi^{(0)}\right)} \tag{2.5}$$

with $\Pi^{(0)}(\mathbf{X} = \bar{Y}$ (Ferreira (2019)).

Categorical Y: The loss function $L\left(Y,\hat{Y}\right) = 1 - \delta_{Y,\hat{Y}}$, where $\delta_{Y,\hat{Y}}$ is Kronecker's delta, is also called 0-1 loss. This loss function defines the Probability of Misclassification (PMC) of $\Pi_{\mathscr{S}_n}$:

$$\text{Pmc}(\Pi_{\mathscr{S}_n}) = \mathbb{E}\left[1 - \delta_{Y,\hat{Y}}|\mathscr{S}_n\right] = \mathbb{P}(Y \neq \hat{Y}|\mathscr{S}_n). \tag{2.6}$$

The best predictor of Y that is a function of $\mathbf{X}$ in the sense of PMC is $\Pi(\mathbf{X}) \coloneqq \text{mode}(g_{\mathbf{X}})$. This conditional mode and theoretical value is what one is aiming for to estimate as good as possible. This is proven at the end of this section in Proof 4. Predicting Y by the conditional mode is also called the Bayes rule. Also, $\text{Pmc}(\hat{Y}) = \mathbb{P}(\hat{Y} = 0|Y = 1, \mathscr{S}_n) + \mathbb{P}(\hat{Y} = 1|Y = 0, \mathscr{S}_n) = \text{Fnr} + \text{Fpr}$ with Fnr is the false negative rate and Fpr is the false positive rate.

The sensitivity (Sen) and specificity (Spe) are $\text{Sen} = 1 - \text{Fnr}$ and $\text{Spe} = 1 - \text{Fpr}$ and

$$\text{Pev}(\Pi_{\mathscr{S}_n}) \coloneqq 1 - \frac{\text{Pmc}(\Pi_{\mathscr{S}_n})}{\text{Pmc}(\max \pi_j)} = 1 - \frac{\mathbb{P}(Y \neq \Pi_{\mathscr{S}_n}(\mathbf{X})|\mathscr{S}_n)}{1 - \max \pi_j} \tag{2.7}$$

If one desires a different combination of sensitivity and specificity, an ROC analysis can be performed.

Statement and Proof 1

If X is a random variable with $\mathbb{E}[X^2] < \infty$, then the best constant predictor of X in the sense of MSE is $\mu \coloneqq \mathbb{E}[X]$.

Proof:
Using $\mathbb{E}[X - \mu] = 0$, we have

$$\begin{aligned}
\text{MSE}(c) \equiv \mathbb{E}\left[(X - c)^2\right] &= \mathbb{E}\left[(X - \mu + \mu - c)^2\right] \\
&= \mathbb{E}\left[(X - \mu)^2 + 2(X - \mu)(\mu - c) + (\mu - c)^2\right] \\
&= E\left[(X - \mu)^2\right] + 2(\mu - c)\mathbb{E}[X - \mu] + (\mu - c)^2 \\
&= \text{MSE}(\mu) + (\mu - c)^2 \geq \text{MSE}(\mu)
\end{aligned}$$

with equality if and only if $c = \mu$.

Statement and Proof 2

Suppose that (X, Y) is a random vector such that $\mathbb{E}[Y^2] < \infty$. Then the best predictor of Y that is a function of X in the sense of MSE is $\Pi(X) \coloneqq \mathbb{E}[Y|X]$.

Proof:
Using that by the law of total expectation

$$\begin{aligned}\mathbb{E}\left[(Y - \Pi(X))\left(\Pi(X) - \Pi'(X)\right)\right] &= \mathbb{E}\left[\mathbb{E}\left[(Y - \Pi(X))(\Pi(X) - \Pi'(X))\right]|X\right] \\ &= \mathbb{E}\left[E[Y - \Pi(X)|X](\Pi(X) - \Pi'(X))\right] \\ &= \mathbb{E}\left[E[Y|X - \Pi(X)](\Pi(X) - \Pi'(X))\right] \\ &= \mathbb{E}\left[0 \cdot (\Pi(X) - \Pi'(X))\right] \\ &= 0,\end{aligned}$$

we have

$$\begin{aligned}\text{MSE}\left(\Pi'(X)\right) &\equiv \mathbb{E}[(Y - \Pi'(X))^2] = \mathbb{E}[(Y - \Pi(X) + \Pi(X) - \Pi'(X))^2] \\ &= \mathbb{E}[(Y - \Pi(X))^2 + 2(Y - \Pi(X))(\Pi(X) - \Pi'(X)) + (\Pi(X) - \Pi'(X))^2] \\ &= \mathbb{E}[(Y - \Pi(X))^2] + 2\mathbb{E}[(Y - \Pi(X))(\Pi(X) - \Pi'(X))] + \mathbb{E}[(\Pi(X) - \Pi'(X))^2 \\ &= \text{MSE}(\Pi(X)) + \mathbb{E}[(\Pi(X) - \Pi'(X))^2] \\ &\geq \text{MSE}(\Pi(X))\end{aligned}$$

with equality if and only if $\Pi(X) = \Pi'(X)$.

Statement and Proof 3

Suppose X is a random variable with a continuous, strictly positive density function and such that $\mathbb{E}[|X|] < \infty$, then the best constant predictor of X in the sense of MAE is med$[X]$. That is, the constant $c = \text{med}[X]$ is such that $\mathbb{E}[|X - c|] \leq \mathbb{E}[|X - d|]$ for any other constant d.

Proof: We have

$$\begin{aligned}\psi(c) \coloneqq \mathbb{E}|X - c| &\equiv \int_{-\infty}^{\infty} |x - c| f(x)dx \\ &= \int_{-\infty}^{c} (c - x)f(x)dx + \int_{c}^{\infty} (x - c)f(x)dx \\ &= c\int_{-\infty}^{c} f(x)dx - \int_{-\infty}^{c} xf(x)dx + \int_{c}^{\infty} xf(x)dx - c\int_{c}^{\infty} f(x)dx \\ &= cF(c) - \int_{-\infty}^{c} xf(x)dx + \int_{c}^{\infty} xf(x)dx - c1 - F(c) \\ &= cF(c) - g(c) + h(c) - c1 - F(c),\end{aligned}$$

where, as usual, we write F for the distribution function of f, which as we know satisfies

$$F(y) = \int_{-\infty}^{y} f(x)dx, 1 - F(y) = \int_{y}^{\infty} f(x)dx,$$

and

$$g(c) = \int_{-\infty}^{c} xf(x)dx, 1 - F(y) = \int_{c}^{\infty} xf(x)dx.$$

Of course, since f is continuous $g'(c) = cf(c)$ by the fundamental theorem of calculus. Also, $h'(c) = -cf(c)$ by the fundamental theorem of calculus and the fact that $g(c') = g(c) = \int_c^{c'} xf(x)dx$:

$$\begin{aligned} h'(c) &= \frac{d}{dc}\int_c^{\infty} xf(x)dx \\ &= \frac{d}{dc}\int_c^{c'} xf(x)dx + \frac{d}{dc}\int_{c'}^{\infty} xf(x)dx \\ &= \frac{d}{dc}(G(c') - G(c)) \\ &= -cf(c). \end{aligned}$$

Thus

$$\begin{aligned} \psi'(c) &= F(c) + cf(c) - g'(c) + h'(c) - 1 - F(c) + cf(c) \\ &= F(c) + cf(c) - cf(c) - cf(c) - 1 - F(c) + cf(c) \\ &= F(c) - 1 - F(c) \end{aligned}$$

so $\psi'(c) = 0 \iff F(c) = 1 - F(c)$ and c is the median of F; in other words, $\int_{-\infty}^{c} f(x)dx = \int_c^{\infty} f(x)dx$: the area under $f(x)$ on $(-\infty, c]$ has the same size as the area under $f(x)$ on $[c, \infty)$.

Statement and Proof 4

Let (X, Y) be a random vector with $f_x(y) \coloneqq \mathbb{P}(Y = y|X = x) > 0$ if and only if $y = 1, 2, \ldots, K$, for some integer $K \geq 2$ (K is a constant, so it is independent of x) and such that $f_x(y_1) \neq f_x(y_2)$ if $y_1 \neq y_2$. The best predictor of Y in the sense of probability of misclassification (pmc) - also called misclassification error - that

(a) is a constant, is $c = \text{mode}[Y]$.
(b) is a function of X, is $\Pi(X) \coloneqq \text{mode}[g_X]$.

Proof (a):

$$\mathbb{P}(Y = c) = \mathbb{E}(\mathbb{1}_{Y=c}) > \mathbb{E}(\mathbb{1}_{Y=d}) = \mathbb{P}(Y = d) \tag{2.8}$$

with

$$\mathbb{1}_{Y=y} = \begin{cases} 1, & \text{if } Y = y. \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

.

Proof (b):
By definition, if $X = x$, then $g_X(\Pi(x)) \geq g_X(\Pi'(x))$ for any other Π', so

$$\begin{aligned}
\mathbb{P}(Y = \Pi(X)) &= \sum_X \mathbb{P}(Y = \Pi(X), X = x) \\
&= \sum_X \mathbb{P}(Y = \Pi(x) | X = x) \mathbb{P}(X = x) \\
&= \sum_X g_X(\Pi(x)) \mathbb{P}(X = x) \\
&\geq \sum_X g_X(\Pi'(x)) \mathbb{P}(X = x) \\
&= \sum_X \mathbb{P}(Y = \Pi'(X), X = x) \\
&= \mathbb{P}(Y = \Pi'(X)).
\end{aligned}$$

Estimates of measures of accuracy

For assessing the quality of the predictor, different observations are needed than the model is fitted on. There are different possibilities for doing this.

One can take a portion of the training data set $\mathscr{S}_n$ and make it test data. This is computationally efficient, but only useful when $\mathscr{S}_n$ is large enough so that the corresponding predictor, say Π, is hardly affected because of it or if $\Pi_{\mathscr{S}_n}$ is stable in the sense that if we remove a random observation the predictor does not change.

A second option is to randomly split the data set into two, construct a predictor with one half of the data set, assess the quality of that predictor with the other half. Then randomly split the data set into two again and repeat the whole process a number of times. And finally average over the successive estimates.

The last option here considered is to randomly split the data set into k sets, construct the predictor with all but one of the k sets and assess the quality of the predictor with the one set that is left. Then repeat this for every set in k and average over the successive estimates. This method is called k-fold cross-validation for any k and is called leave-one-out cross-validation (LOOCV) for the special case $k = n$.

For MSE(Π) the LOOCV estimator based on $\mathscr{S}$ is

$$\text{mse}(\Pi) \coloneqq \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \Pi_{\mathscr{S}_i}(\mathbf{X})\right)^2, \tag{2.10}$$

where $\mathscr{S}_i$ denotes the sample $\mathscr{S}$ without its i-th element, $(\mathbf{X}_i, Y_i)$, and

$$\Pi_{\mathscr{S}_i}$$

is the predictor constructed with $\mathscr{S}_i$. This estimator of MSE(Π) gets closer to being unbiased when $n \to \infty$ or when $\Pi_{\mathscr{S}_n}$ is stable. The same principle holds for Mae(Π), Pev(Π) and Bias(Π) for numerical Y. Their LOOCV estimates are:

$$\text{mae}(\Pi) \coloneqq \frac{1}{n} \sum_{i=1} \left|Y_i - \Pi_{\mathscr{S}_i}(\mathbf{X})\right|,$$

$$\text{pev}(\Pi) \coloneqq 1 - \frac{\text{mse}(\Pi)}{\text{mse}(\Pi^{(0)})} \quad \text{and}$$

$$\text{bias}(\Pi) \coloneqq \frac{1}{n} \sum_{i=1} \left(Y_i - \Pi_{\mathscr{I}_i}(\mathbf{X}) \right).$$

In this study the focus is on regression problems and the loss function used is $L(Y, \hat{Y}) = \left(Y - \hat{Y}\right)^2$ and corresponding MSE.

2.1.3 Illustration

During the course of this study a dataset for illustrations is used on the compressive strength of concrete from Yeh (2006). This is a dataset of 1030 observations or blocks of concrete. The response is the compressive strength of the block and there are eight features.

One of the most standard approaches for prediction is to fit a linear model to the data with the method of least squares. This method minimizes the sum of the squares of the residuals. The underlying model that the method of least squares assumes is $Y = X\beta$ which results in the optimal parameter $\hat{\beta} = (X^T X)^{-1} X^T Y$ for which $X^T X$ is invertible If X has full rank as can be seen in statement 5.

Figure 2.1.1 shows the predictive performance of the linear model fitted to the data.

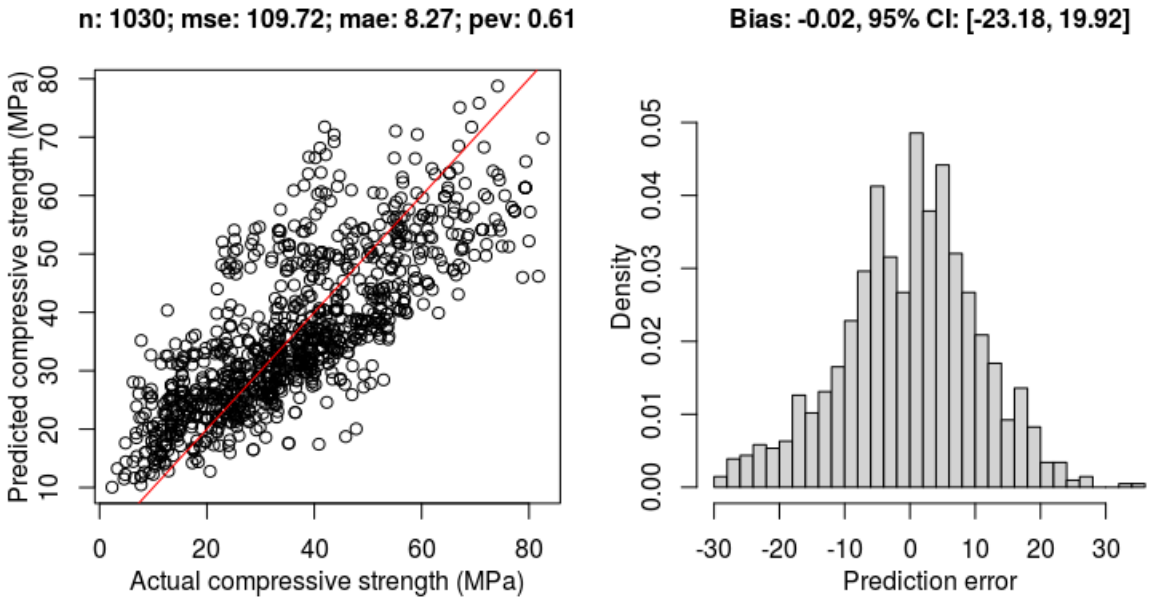

Figure 2.1.1: Predictive performance of linear model fitted to the data.

Statement and Proof 5

Let X be an $n \times p$ matrix of rank p, i.e. with its p columns linearly independent; think of the design matrix in a regression model. Then $X^T X$ is invertible (i.e. its columns are linearly independent).

Proof: By assumption, the columns of X are linearly independent, which is to say that $Xv = 0 \implies v = 0$ for every p-dimensional vector v. We know that the $p \times p$ matrix $X^T X$ is invertible if and only if its columns are linearly independent. If $X^T X$ were not invertible then there would be a vector $u \neq 0$ such that $X^T X u = 0$. But then it would follow that $u^T X^T X u = (Xu)^T (Xu) = |Xu|^2 = 0$, and hence $Xu = 0$, contradicting our assumption.

2.2 Tree-based methods

Tree-based methods are a nonparametric class of prediction methods based on the algorithmic structure of cutting the feature space into regions which can be presented as the growth of a tree (Friedman et al. (2001)). The fundament of every tree-based method is the construction of a singular decision tree where a single tree on its own can be a predictor. The concept of a tree predictor arose in the mid-1980s when Breiman was faced with nonstandard data that could not be treated by standard methods like linear regression, stepwise linear regression and discriminant analysis. At the same time the machine learning movement arose, trying to deal with the nonstandard data and complex prediction problems and trying to attain better predictive accuracy for these complex problems like speech recognition, image recognition, nonlinear time series prediction, handwriting recognition and prediction in financial markets.

A single tree as a predictor can be useful: they are easy to interpret and can be powerful and reasonably accurate, especially when the training set used to construct the tree is large. However, a downside of the tree predictor is that a slight perturbation of the training set can result in quite a different new tree compared to the original one, while the two trees have similar test set errors as mentioned as the multiplicity problem in Breiman (2001b). In short, trees can be good predictors, but instability and inaccuracy can be problematic. Fortunately, there are ways to improve upon single tree predictors. Some familiar methods are bagging, boosting, arching and additive logistic regression. One particular improvement and very popular method is the random forest algorithm, a combination between bagging and an additional randomization processes, which is our motivation to study tree-based methods.

We do consider single trees first to get a thorough understanding of the fundaments of the random forest algorithm. Then after we dive into the random forest algorithm and variable importance measures.

2.2.1 Decision trees

Instead of making an assumption about the data, tree-based methods take a nonparametric approach to estimate theoretical values like $\mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ and $\mathbb{P}(Y = y|\mathbf{X} = \mathbf{x})$, where a regression tree and classification tree are needed, respectively. Tree-based methods cut feature spaces into

partitions and grow trees. The root node of the tree resembles all of the feature space and every cut in the feature space resembles the splitting of a node in the tree. With every split of a node, there are two new nodes, and with every cut of a partition, the partition is split into two smaller partitions. Both the new nodes and the smaller partitions can be split again. This process continues until a certain stopping criterion stops the process. Then the resulting partition is given by the terminal nodes or leaves of the tree.

To visualize this for a regression problem, we take a dataset on the compressive strength of concrete from Yeh (2006). This is a dataset of 1030 observations or 1030 blocks of concrete. The response is the compressive strength of the block and there are eight features of which we take two: the age of the block counted in days and the cement to water ratio (c.to.w.ratio). Figure 2.2.1 shows a partition of the feature space with eight regions and the predicted value in each region. Figure 2.2.2 shows the corresponding tree with eight leaf nodes and matching predicted values. The tree has a little yes- and no-sign to show that any observation satisfying the condition at any split is assigned to the left and others to the right branch.

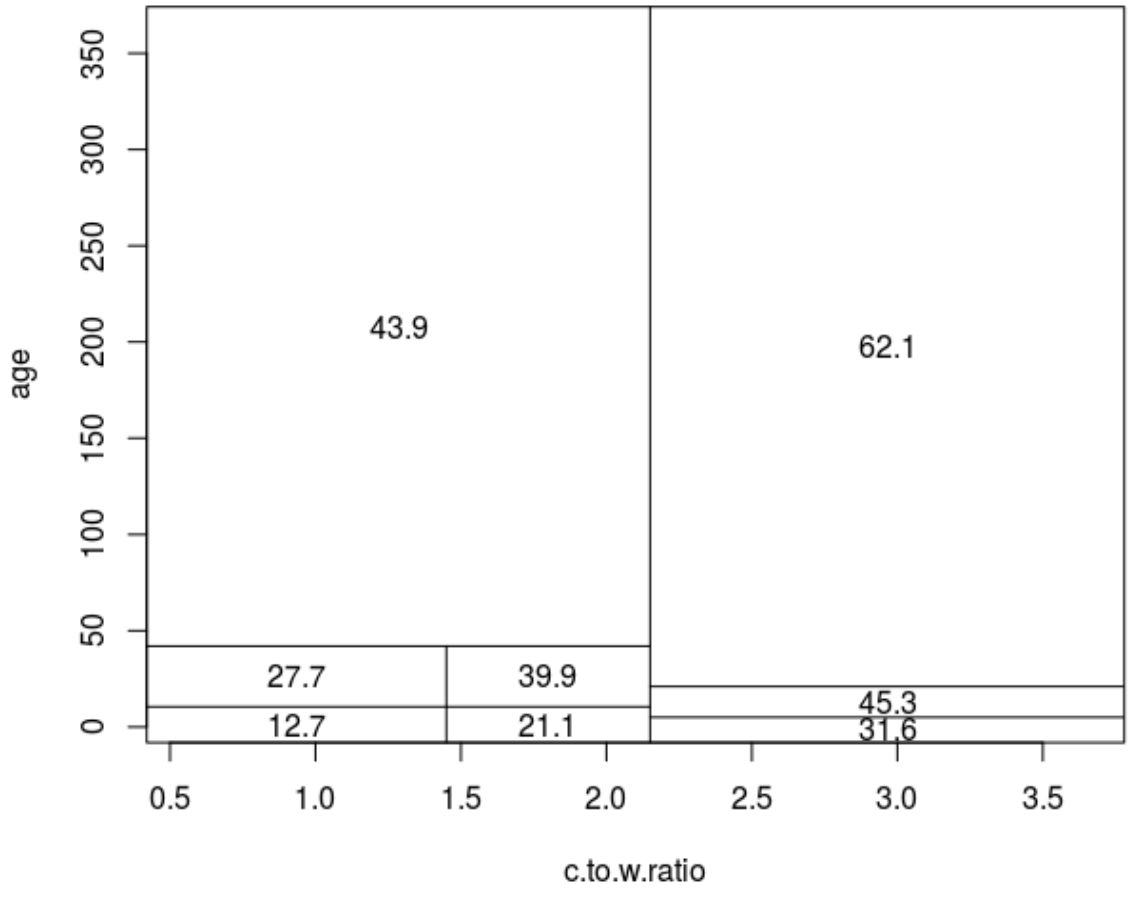

Figure 2.2.1: Partition of the feature space for a data set on the compressive strength of concrete.

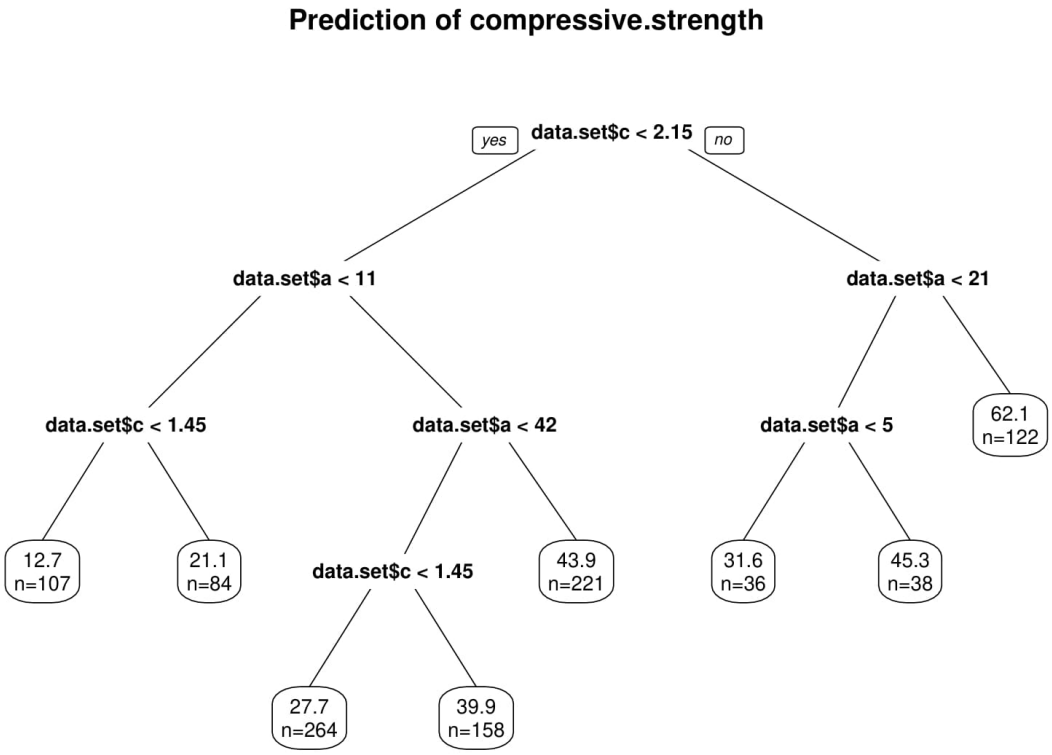

Figure 2.2.2: Tree based a data set on the compressive strength of concrete.

Construction of a tree predictor

The construction of the tree predictor basically consists of the split criterion and the stopping criterion with pruning as a possible extra step to apply when the process of growing the tree has stopped.

How the response is modeled depends on the split criterion we adopt. If the response is modelled as, say, a constant c_m, then $\Pi(x) = \sum_{m=1}^{M} c_m I_{x \in R_m}$ with M the number of leaf nodes and R_m the region corresponding to leaf node m with $m = 1, \ldots, M$. The usual choice of split criterion for regression problems is the minimization of the MSE. As shown in Section 2.1 the optimal value of c_m is the average of y_i in region R_m: $\hat{c}_m = \text{ave}(y_i | x_i \in R_m)$ (Friedman et al. (2001)).

Finding the best binary partition in terms of MSE is generally computationally infeasible. An approach that is feasible and most used to construct trees is a one-step lookahead or greedy approach: We start with all of the data and consider a split variable j and a split point s. We take any random split point s which splits any partition in two regions and call one of the regions $R_1(j, s) = \{X | X_j \leq s\}$ and the other region $R_2(j, s) = \{X | X_j > s\}$. Then we find the minimum values of $\min \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2$ and $\min \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2$ which are $\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s))$ and $\hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s))$ and add them up. We calculate the sum of these two values for any variable and any possible split point that changes what observations are in what region and choose the split point that gives the smallest sum. The split will be made

in the middle of two consecutive data points. In other words we solve

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

It is possible to continue this process until every leaf consists of one observation, but this might lead to overfitting. On the other hand, we want to grow the tree large enough so it captures the most important structures. There is different approaches to tuning this tree size and going about this trade-off of minimizing variance and bias. The preferred strategy according to Friedman et al. (2001) is to grow a large tree until some small number nodesize is reached in the leaves and to, after growing the tree, prune the tree with cost-complexity pruning first introduced by Breiman and Ihaka (1984). This strategy is not in all cases an optimal strategy as we show in the following illustration.

An example of a situation where there is improvement possible over cost-complexity pruning is when we apply both complexity pruning and an optimization algorithm to the dataset on the compressive strength of concrete. This time we use all features, but only the first 300 of the 1030 observations for the sake of computation time and these results are very similar to the results that follow from using all observations. The performance of the trees is obtained with leave-one-out cross-validation (LOOCV). The left graph in Figure 2.2.3 shows the performance of the tree predictor with default value of nodesize. The right graph shows the performance of a tree with optimized nodesize.

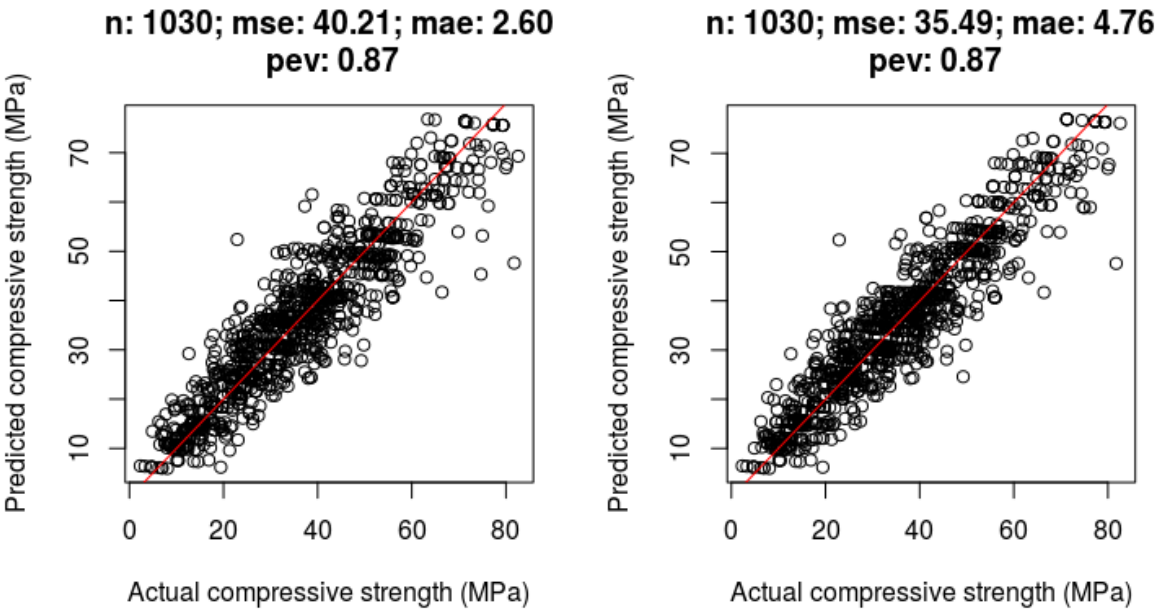

Figure 2.2.3: Performance of tree with default value of nodesize vs optimized value of nodesize.

Performance of a tree predictor

The performance of a tree predictor can be obtained with leave-one-out cross-validation (LOOCV). In Figure 2.2.4 a tree can be seen with stopping criterium: nodesize ≥ 30. The parameter node-

size is not optimized, but this tree predictor already performs better than fitting the linear model in 2.1 as can be seen in Figure 2.2.5.

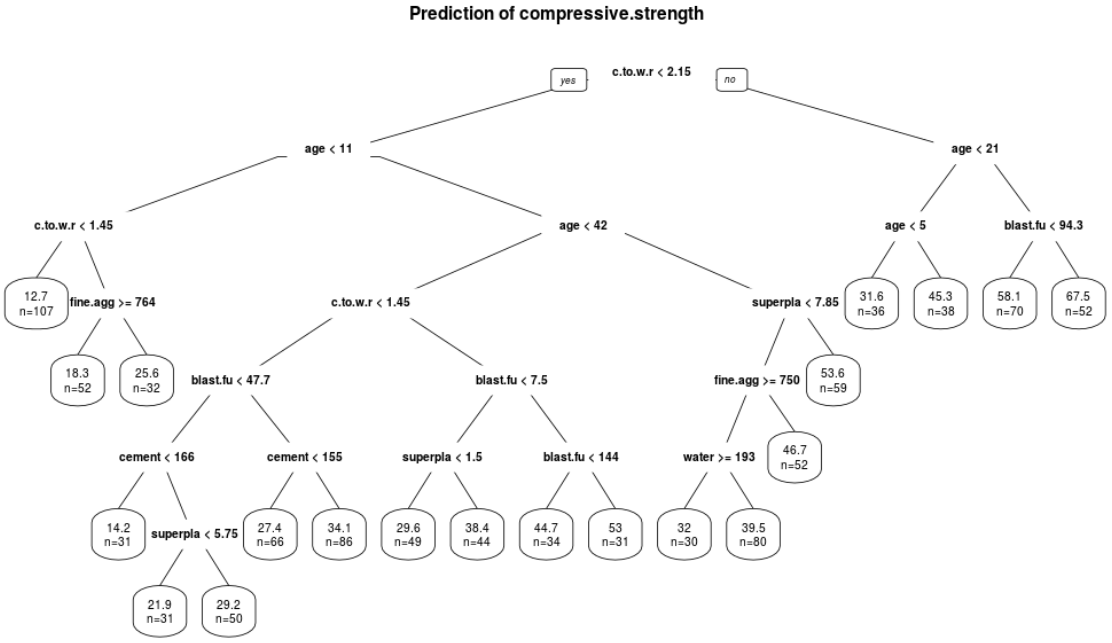

Figure 2.2.4: Tree with nodesize ≥ 30.

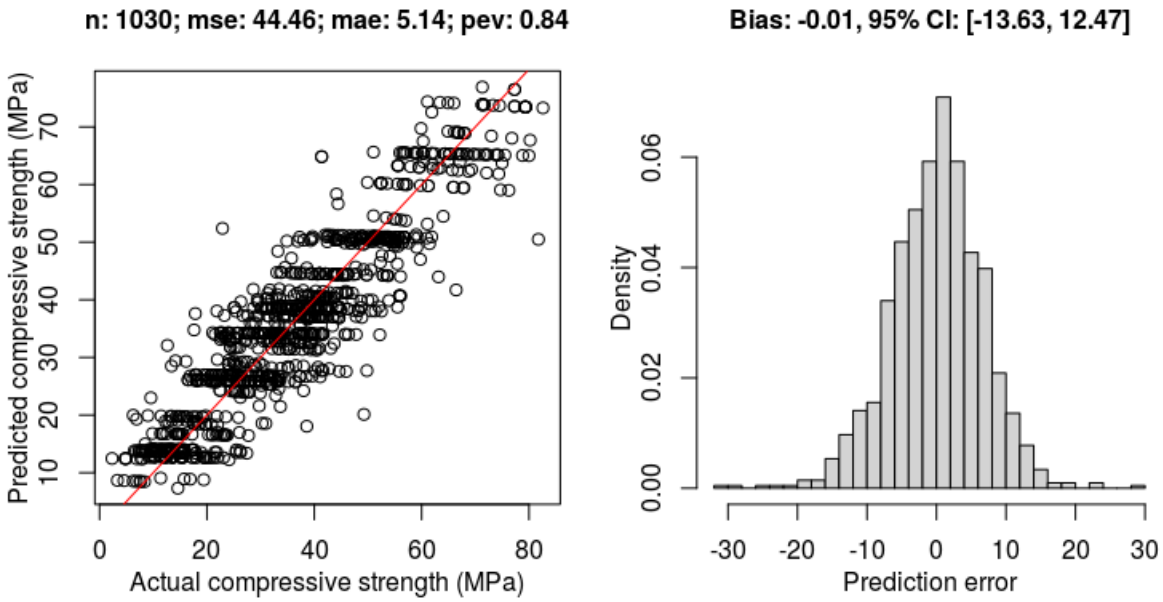

Figure 2.2.5: Performance of tree with nodesize ≥ 30.

Limitations

As already discussed in the introduction: trees are limited and they can be easily improved upon. Trees have difficulty capturing additive structures because of their binary tree structure, they lack smoothness because they are step functions and they have low bias but high variance mainly because of the hierarchical nature of the process. Trees can be improved upon by among others random forests.

2.2.2 Random forest predictor

To overcome the limitations just mentioned and improve on the accuracy of tree predictors, ensemble learning methods, that grow many base learners and aggregate them to predict, have been designed. The random forest algorithm is such a method and grows many randomized decision trees and aggregates them to predict, which reduces the variance and correlation between the sampled trees. This idea of random forests is written down by Leo Breiman in Breiman (2001a) where both bagging (Breiman (1996)) and the CART-split (Breiman and Ihaka (1984)) criterion play critical roles. Random forests are there in many different forms. In this study when we mention random forests it refers to the original algorithm in Breiman (2001a).

Random forests have become a very successful method in many scientific fields, because of their good predictive performance for a wide range of regression and classification problems: They are known for being able to handle large-scale problems with a large number of both observations and features as well as for problems where the number of features is much larger than the number of observations. They are known for being able to deal with complex data structures like interactions and even highly correlated features. They have only few parameters to tune and they return measures of variable importance. Random forests are among others used for air quality prediction, chemoinformatics, ecology, 3D object recognition and bioinformatics (Biau and Scornet (2016)).

One major drawback of random forests that is mentioned by some is their lack of interpretability. Where trees are so easy to interpret, random forests are not due to its element of randomization. Some prefer interpretability over accuracy and others do not as we mentioned in section 2.1 on statistical prediction.

One way to get some insight into the algorithm is through the importance of variables which can be measured in different ways and aims to quantify the influence of each feature on the predictive performance. The two measures that are implemented in the randomForest package are the Mean Decrease Impurity (MDI) or Gini measure and the Mean Decrease Accuracy (MDA) or permutation measure. The MDA will be elaborated on in Chapter 3. We chose to not elaborate on the Gini measure, because most of the recent literature was negative about it. However, Nembrini et al. (2018) write about the revival of the Gini importance, so this could be an interesting topic for future research. It is definitely possible to do experiments with simulations and try to discover patterns and make hypothesis with variable importance measures, but all of this is not theoretically justified.

Much intertwined with the drawback of interpretability is the small theoretical understanding of random forests which is also due to its element of randomization. Most of its practical performance can not yet be explained theoretically due to the complex data-dependent mechanisms:

both bagging and the CART-split criterion are difficult to analyze with rigorous mathematics. Theoretical analysis has been done, but only on simplified versions of the original procedure which we elaborate on in Section 2.3.

Construction of a random forest

Random forests have the single tree predictor as its base learner. The process consists of (Biau and Scornet (2016)):

1. Sample fractions from the data (bootstrapping or subsampling);
2. Grow randomized decision trees on each fraction of the data (and do not prune);
3. Aggregate the tree predictors.

The randomized decision trees are random in two processes:

- Subsampling fractions from the original dataset (with or without replacement), one for the construction of every tree. Breiman in his original algorithm randomly sampled exactly n times from n points with replacement which is called bootstrapping (Efron (1982));
- Subsampling the features (without replacement) at every split point of every tree. At each split point only the sampled features are taken into account for making the split.

The structure of the random forest algorithm is the same for regression and classification problems, but besides the split-criterion, that also differed for regression and classification trees, also the aggregation process differs: For regression the final prediction is the average over the predictions of every single tree. For classification a majority class vote is used.

Most articles mention that there are three parameters that can be tuned: the number of sampled data points for each tree (a_n), the number of possible directions for splitting at each node of each tree ($mtry$) and the number of examples in each cell below which the cell is not split ($nodesize$). Some see the number of trees (M) also as a tuning parameter, so it is mentioned here too.

The (default) values of these parameters are:

- $mtry \in 1, \ldots, p$ and default value $mtry = \lfloor p/3 \rfloor$ for regression and $mtry = \lfloor \sqrt{p} \rfloor$ for classification;
- $nodesize \in 1, \ldots, a_n$ and default value $nodesize = 5$ for regression and $nodesize = 1$ for classification;
- $a_n \in 1, \ldots, n$ and default value $a_n = n$ for both regression and classification;
- $M \in 1, \ldots, \infty$ and default value is $M = 500$ for both regression and classification.

If $mtry = p$ and the samples are drawn with replacement, the subsampling step is called bagging.

The random forest regression predictor is $F(x) := \frac{1}{M} \sum_{m=1}^{M} T_m(x)$ with T_m the mth random forest tree.

Variance reduction

Trees are known for their low bias and high variance. The idea of random forests is to reduce the variance of its prediction by the randomization processes.

The element that random forests adds to bagging is the subsampling of covariates. The size of the correlation of pairs of bagged trees limits the benefits of averaging. The following theory demonstrates this: Every tree has variance σ^2. Then the average variance over all trees M if their outputs are i.i.d is $\frac{1}{M}\sigma^2$. But if trees are only identically distributed (i.d.) with positive pairwise correlation ρ, the variance of the average is $\rho \cdot \sigma^2 + \frac{(1-\rho)}{M} \cdot \sigma^2$ (Friedman et al. (2001)). The second term disappears as M increases, but the first remains and causes the limitation. The subsampling of $mtry$ covariates reduces the correlation between any pair of trees and reduces the variance of the average. However, there is a downside to increasing $mtry$ which is an increase of bias.

Performance of a random forest predictor

Instead of using k-fold cross-validation for the error estimation of the predictor, it is possible to perform cross-validation during the process of the algorithm. The difference between the sampled data for a single tree and the whole training set is new data for that particular tree and can be used for making predictions with that particular tree and calculating the corresponding prediction errors. The non-sampled data for a single tree is called the Out-Of-Bag (OOB) set for that particular tree and the corresponding error is called the OOB error.

Figure 2.2.6 shows the performance of the random forest predictor for the dataset on the compressive strength of concrete, which is better than the single tree predictor of the previous section.

Variable importance

Figure 2.2.7 shows a variable importance plot of the MDA or permutation importance.

2.3 Consistency of random forests

Although random forests are used a lot in practice and a lot is discovered about its practicalities, little is known about its consistency and other mathematical properties. The goal is to prove that the random forest predictor Π_n is L^2 consistent, i.e. $\mathbb{E}[\Pi_n(\mathbf{X}) - \Pi(\mathbf{X})]^2 \to 0$ as $n \to \infty$. The difficulty of reaching this goal for the random forest predictor lies in the complexity of simultaneously analyzing the twofold randomization process, which consists of both bagging (Breiman, 1996) and the classification and regression tree (CART)-split criterion (Breiman et al., 1984). Various theoretical studies are performed on simplified models that omit the bagging, CART-split criterion and/or the data dependency: some studies concentrate on isolated parts of the procedure, some ignore the bagging step, some replace the CART-split criterion by a more elementary cut protocol and most studies focus on data-independent procedures. In the study of Scornet, Biau and Vert (2015) a step is made to a more realistic representation of

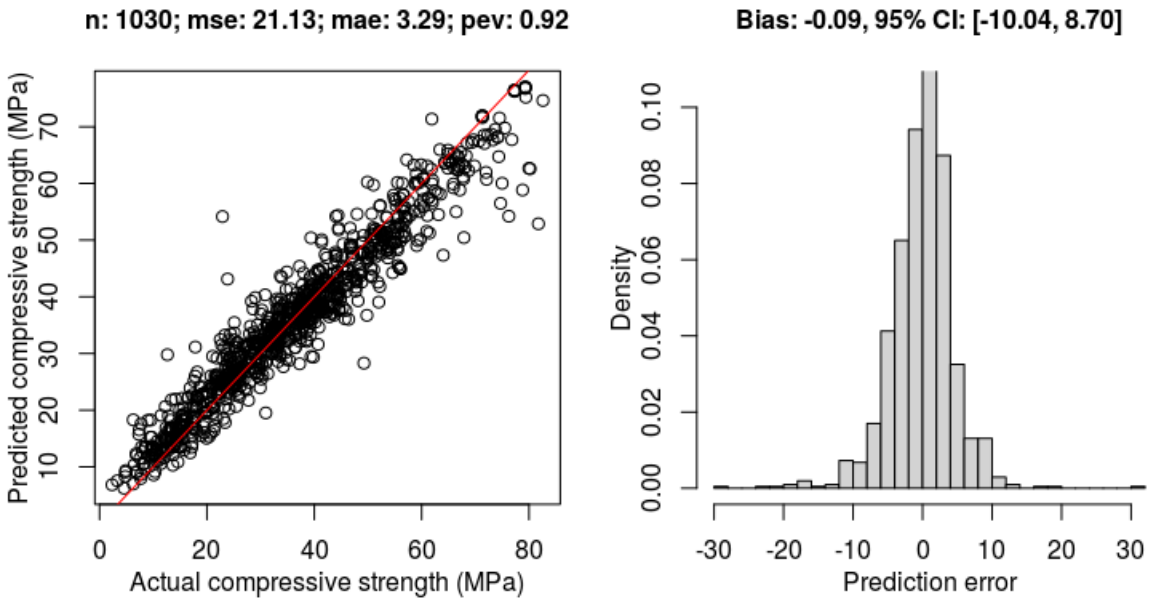

Figure 2.2.6: Performance of RF.

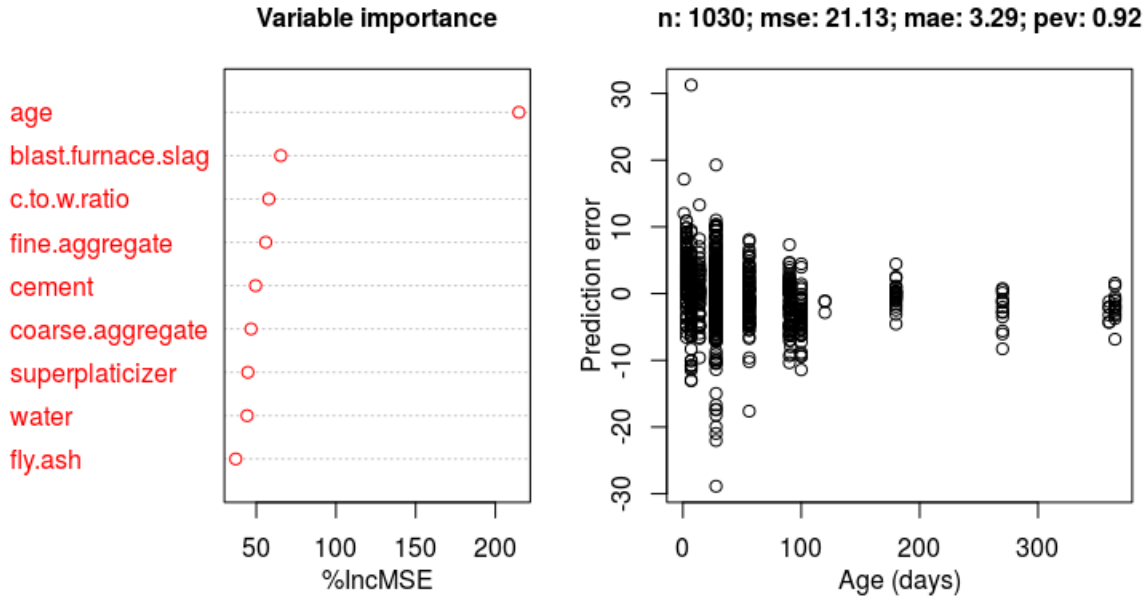

Figure 2.2.7: Variable importance.

Breiman's original procedure of random forests. The context in their study Scornet et al. (2015) is an additive regression model satisfying the following properties: (H1) The response Y follows $Y = \sum_{j=1}^{p} \Pi_j(\mathbf{X}^{(j)}) + \epsilon$, where $\mathbf{X} = (\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(p)})$ is uniformly distributed over $[0,1]^p$, ϵ is an independent centered Gaussian noise with finite variance $\sigma^2 > 0$ and each component Π_j is continuous.

These models, which decompose the regression function as a sum of univariate functions, are flexible and easy to interpret.

The results of this study are the first consistency results for Breiman's original procedure, including a data-dependent model and including the twofold randomization process. Bootstrapping is replaced by subsampling, but this difference is harmless.

We will give a description of the approach of this study to attain L^2 consistency for random forests, mention the assumptions that are made along the way and give a clear overview of the assumptions and downsides of the assumptions at the end.

2.3.1 Approximation and estimation error

In order to attain L^2 consistency it is necessary to control the approximation and estimation error of a method. Proposition 2 offers a good control of the approximation error:

PROPOSITION 2. Assume that (H1) holds. Then, for all $\rho, \xi > 0$, there exists $N \in \mathbb{N}^*$ such that, for all $n > N$, $\mathbf{P}[\Delta(\Pi, A_n(\mathbf{X}, \Theta)) \leq \xi] \geq 1 - \rho$, where the variation of Π within each cell A is $\Delta(\Pi, A) = \sup_{\mathbf{x},\mathbf{x}' \in A} |\Pi(\mathbf{x}) - \Pi(\mathbf{x}')|$ and $A_n(\mathbf{X}, \Theta)$ is the cell of a tree built with random parameter Θ that contains the point $\mathbf{X}$.

This states that, provided n is large enough, the variation of Π within a cell of a random tree is small, thereby forcing the approximation error of the forest to asymptotically approach zero. For controlling the estimation error in this study, two different regimes are considered. The first regime is when $t_n < a_n$ (the trees are not fully grown) and the second regime is when $t_n = a_n$ (the trees are fully grown). For the first regime the key to controlling the estimation error is t_n and for the second regime it is the subsampling rate a_n/n. For consistency in the first regime when $t_n < a_n$ the assumptions are (H1) and control over the depth of the trees via the parameter t_n:

THEOREM 1. Assume that (H1) is satisfied. Then, provided $a_n \to \infty$, $t_n \to \infty$ and $t_n(\log a_n)^9/a_n \to 0$ (t_n tends to infinity more slowly than a_n), random forests are consistent.

Consistency in the second regime is more complicated and needs an assumption and some notation: $Z_i = \mathbb{I}_{\mathbf{X} \leftrightarrow \mathbf{X}_i, \Theta}$ is the indicator that $\mathbf{X}_i$ falls into the same cell as $\mathbf{X}$ in the random tree designed with $\mathscr{S}_n$ and the random parameter Θ. Similarly, $Z'_j = \mathbb{I}_{\mathbf{X} \leftrightarrow \mathbf{X}_i, \Theta'}$ where Θ' is an independent copy of Θ. Accordingly,

$$\psi_{i,j}(Y_i, Y_j) = \mathbb{E}[Z_i Z'_j | \mathbf{X}, \Theta, \Theta', \mathbf{X}_1, \ldots, \mathbf{X}_n, Y_i, Y_j] \tag{2.11}$$

and

$$\psi_{i,j} = \mathbb{E}[Z_i Z'_j | \mathbf{X}, \Theta, \Theta', \mathbf{X}_1, \ldots, \mathbf{X}_n]. \tag{2.12}$$

Finally, for any random variable W_1, W_2, Z is $\mathrm{Corr}(W_1, W_2|Z)$ the conditional correlation coefficient (whenever it exists).

The assumption is
(H2) Let $Z_{i,j} = (Z_i, Z'_j)$. Then one of the following two conditions holds:
(H2.1) One has

$$\lim_{n \to \infty} (\log a_n)^{2p-2} (\log n)^2 \mathbb{E}[\max_{i,j,i \neq j} |\psi_{i,j}(Y_i, Y_j) - \psi_{i,j}|]^2 = 0. \tag{2.13}$$

(H2.2) There exists a constant $C > 0$ and a sequence $(\gamma_n)_n \to 0$ such that, almost surely,

$$\max_{l_1,l_2=0,1} \frac{|\mathrm{Corr}(Y_i - \Pi(\mathbf{X}_i), \mathbb{I}_{Z_{i,j}=(l_1,l_2)}|\mathbf{X}_i, \mathbf{X}_j, Y_j)|}{\mathbb{P}^{1/2}[Z_{i,j} = (l_1, l_2)|\mathbf{X}_i, \mathbf{X}_j, Y_j]} \leq \gamma_n \tag{2.14}$$

and

$$\max_{l_1=0,1} \frac{|\mathrm{Corr}((Y_i - \Pi(\mathbf{X}_i))^2, \mathbb{I}_{Z_i=l_1}|\mathbf{X}_i)|}{\mathbb{P}^{1/2}[Z_i = l_1|\mathbf{X}_i]} \leq C. \tag{2.15}$$

(H2.1) means that the influence of two Y-values on the probability of connection of two couples of random points tends to zero as $n \to \infty$. Assumption (H2.2) holds whenever the correlation between the noise and the probability of connection of two couples of random points vanishes quickly enough, as $n \to \infty$.

For consistency in the second regime when $t_n = a_n$ the assumptions are (H1), (H2) and control over the subsampling rate a_n/n:

THEOREM 2. Assume that (H1) and (H2) are satisfied, and let $t_n = a_n$. Then, provided $a_n \to \infty$, $t_n \to \infty$ and $a_n \log n/n \to 0$, random forests are consistent.

2.3.2 Assumptions

The assumptions that are made in this study are (H1) for the whole study and large enough n is needed to satisfy Proposition 2 and control the approximation error. To control the estimation error when $t_n < a_n$ it is assumed that t_n tends to infinity more slowly than a_n and when $t_n = a_n$ then it is assumed that (H2) holds and that the subsampling rate a_n/n is $\mathrm{o}(1/\log n)$. If high-dimensionality is modeled by letting $p = p_n \to \infty$, then assumption (H2.1) may be too restrictive since the term $(\log a_n)^{2p-2}$ will diverge at a fast rate. When a partition is independent of the Y_is, then the correlations in (H2.2) are zero, but when the partitions depend on the whole of $\mathscr{S}_n$, it is not known if (H2.2) is satisfied.

Chapter 3

Variable importance measures

3.1 Introduction

Measuring variable importance (VI) is most of the time a difficult task. Unless the relationship between covariates and response is very simple and can be accurately grasped in a (linear) model, it is a tricky study to perform. A first idea to measure VI could be to calculate the correlations between the response and each of the covariates. However,correlations between the covariates can mess with these scores. Also, interactions between covariates makes measuring variable importance less straightforward.

This brings us to the question: What should the measure of variable importance be under the data-generating model? And what are the best estimates of measures of VI based on predictors constructed from data.

It is important that these predictors constructed from data are quite accurate. Only then can they be approximately reliable sources for informative measures of VI. If the predictive performance of a predictor is not quite good, then measures of VI most probably can not capture the VI of the covariates.

There are both theoretical and empirical aims. The theoretical aim is, if possible, to figure out which measure gives the most correct picture, what measure is optimal. The empirical aim is to find the best estimates of the theoretical measures and to find empirical measures that are computationally efficient. The next sections will give a deeper understanding of these two aims.

3.2 Two theoretical measures of VI

Both measures of VI here presented are based on an increase in mean loss due to some change in the test set. The first measure is called Leave-One-Covariate-Out (LOCO) by Lei et al. (2017). It measures the importance of the k–th variable by means of the theoretical quantity

$$I_k(\Pi) = \mathbb{E}\left[L\left(Y, \Pi^{(k)}(\hat{\mathbf{X}})\right)\right] - \mathbb{E}\left[L\left(Y, \Pi(\mathbf{X})\right)\right] \tag{3.1}$$

where Π is the optimal predictor based on $\mathbf{X} = (X_1, \ldots, X_p)$, L is some loss function and $\Pi^{(k)}(\hat{\mathbf{X}})$ is the optimal predictor based on all the variables in $\mathbf{X} = (X_1, \ldots, X_p)$ except X_k.

This measure we think is the most universal provided the $\Pi(\mathbf{X})$ and $\Pi^{(k)}(\check{\mathbf{X}})$ are the optimal predictors and thus may be seen as providing the true variable importance.

The second measure by Zhu et al. (2012) defines the importance of the k–th variable by means of the theoretical quantity

$$\tilde{I}_k(\Pi) = \mathbb{E}\left[L\left(Y, \Pi(\tilde{\mathbf{X}}^{(k)})\right)\right] - \mathbb{E}\left[L\left(Y, \Pi(\mathbf{X})\right)\right] \tag{3.2}$$

where Π is the optimal predictor based on $\mathbf{X} = (X_1, \ldots, X_p)$ and $\Pi(\tilde{\mathbf{X}}^{(k)})$ is the optimal predictor based on $\tilde{\mathbf{X}} \coloneqq (X_1, \ldots, X_{k-1}, \tilde{X}_k, X_{k+1}, \ldots, X_p)$ where $\tilde{X}_k$ has the same distribution as X_k but is independent of the other X_is.

Although we believe the first measure is most universal, also the second measure is considered for two reasons: the second measure could perform better and the first measure is computationally very inefficient. The latter will be elaborated on in the next section.

3.3 Estimators of the two theoretical measures

Again, suppose one has a data set $\mathscr{S}_n = \{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n)\}$. In this section empirical counterparts of the theoretical measures $I_k(\Pi)$ and $\tilde{I}_k(\Pi)$ are compared to each other with Π denoting the random forest and $L(s,t) = (s-t)^2$ which corresponds to the so-called mean square error criterion.

One can estimate the theoretical $I_k(\Pi)$:

$$\begin{aligned}
I_k(\Pi) &= \mathbb{E}\left[L\left(Y, \Pi^{(k)}(\check{\mathbf{X}}^{(k)})\right)\right] - \mathbb{E}\left[L\left(Y, \Pi(\mathbf{X})\right)\right] \\
&= \mathbb{E}\left[\left(Y - \Pi^{(k)}(\check{\mathbf{X}}^{(k)})\right)^2\right] - \mathbb{E}\left[\left(Y - \Pi(\mathbf{X})\right)^2\right] \\
&= \mathrm{MSE}\left(\Pi^{(k)}(\check{\mathbf{X}}^{(k)})\right) - \mathrm{MSE}(\Pi(\mathbf{X})) \\
&\approx \mathrm{mse}\left(\Pi^{(k)}(\check{\mathbf{X}}^{(k)})\right) - \mathrm{mse}(\Pi(\mathbf{X})) \\
&= \check{\mathrm{mse}}_k - \mathrm{mse} \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - F_i^{(k)}(\check{\mathbf{X}}_i^{(k)})\right)^2 - \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - F_i(\mathbf{X}_i)\right)^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \frac{1}{N_i}\sum_{j:T_j \in \mathscr{T}_i} T_j^{(i,k)}(\check{\mathbf{X}}_i^{(k)})\right)^2 - \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \frac{1}{N_i}\sum_{j:T_j \in \mathscr{T}_i} T_j^{(i)}(\mathbf{X}_i)\right)^2 \\
&= i_k(\Pi),
\end{aligned}$$

where

$$\mathrm{mse} \coloneqq \mathrm{mse}(F(\mathbf{X}) \coloneqq \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - F_i(\mathbf{X}_i)\right)^2,$$

$$\check{\text{mse}}_k \coloneqq \text{mse}\left(F^{(k)}(\check{\mathbf{X}}^{(k)})\right) \coloneqq \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - F_i^{(k)}(\check{\mathbf{X}}_i^{(k)})\right)^2,$$

$\mathscr{T}_i = T_1^{(i)}, T_2^{(i)}, \ldots, T_{N_i}^{(i)}$ is the set of trees that are constructed without $(\mathbf{X}_i, Y_i)$, N is the total number of trees and N_i the number of trees in $\mathscr{T}_i$ which is about $N/3$.

This most universal measure is highly computationally inefficient as already said. This will be explained after two estimates of $\tilde{I}_k(\Pi)$ are presented.

One can estimate the theoretical $\tilde{I}_k(\Pi)$ (T-VIM) with Breiman's permutation importance measure (Breiman (2001a)):

$$\begin{aligned}
\tilde{I}_k(\Pi) &\approx \frac{1}{N}\sum_{j=1}^{N}\left(\tilde{\text{mse}}_k^{(j)} - \text{mse}^{(j)}\right) \\
&= \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{n_j}\sum_{i=1}^{n_j}\left(Y_i^{(j)} - T_j(\tilde{\mathbf{X}}_i^{(j,k)})\right)^2 - \frac{1}{n_j}\sum_{i=1}^{n_j}\left(Y_i^{(j)} - T_j(\mathbf{X}_i^{(j)})\right)^2\right) \\
&= \tilde{i}_k(\Pi)
\end{aligned}$$

where

$$\text{mse}^{(j)} \coloneqq \text{mse}\left(T_j(\mathbf{X}^{(j)})\right) \coloneqq \frac{1}{n_j}\sum_{i=1}^{n_j}\left(Y_i^{(j)} - T_j(\mathbf{X}_i^{(j)})\right)^2,$$

$$\tilde{\text{mse}}_k^{(j)} \coloneqq \text{mse}\left(T_j(\tilde{\mathbf{X}}^{(j,k)})\right) \coloneqq \frac{1}{n_j}\sum_{i=1}^{n_j}\left(Y_i^{(j)} - T_j(\tilde{\mathbf{X}}_i^{(j,k)})\right)^2,$$

the T_js are tree predictors based on $\mathbf{X} = (X_1, \ldots, X_p)$, $\tilde{\mathbf{X}}^{(k)} \coloneqq (X_1, \ldots, X_{k-1}, \tilde{X}_k, X_{k+1}, \ldots, X_p)$ where $\tilde{X}_k$ is a permutation of X_k and $n_j \approx n/3$ is the number of trees in the set $s_j = (\mathbf{X_1}^{(j)}, Y_1^{(j)}), (\mathbf{X_2}^{(j)}, Y_2^{(j)}), \ldots, (\mathbf{X_{n_j}}^{(j)}, Y_{n_j}^{(j)})$ with observations that are not used for the construction of T_j.

The theoretical $\tilde{I}_k(\Pi)$ (F-VIM) can also be estimated differently:

$$\begin{aligned}
\hat{I}_k(\Pi) &= \mathbb{E}\left[L\left(Y, \Pi(\hat{\mathbf{X}}^{(k)})\right)\right] - \mathbb{E}\left[L\left(Y, \Pi(\mathbf{X})\right)\right] \\
&= \mathbb{E}\left[\left(Y - \Pi(\hat{\mathbf{X}}^{(k)})\right)^2\right] - \mathbb{E}\left[\left(Y - \Pi(\mathbf{X})\right)^2\right] \\
&= \text{MSE}\left(\Pi(\hat{\mathbf{X}}^{(k)})\right) - \text{MSE}(\Pi(\mathbf{X})) \\
&\approx \text{mse}\left(\Pi(\hat{\mathbf{X}}^{(k)})\right) - \text{mse}(\Pi(\mathbf{X})) \\
&= \hat{\text{mse}}_k - \text{mse} \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - F_i(\hat{\mathbf{X}}_i^{(k)})\right)^2 - \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - F_i(\mathbf{X}_i)\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \frac{1}{N_i}\sum_{j:T_j\in\mathscr{T}_i} T_j^{(i)}(\hat{\mathbf{X}}_i^{(k)})\right)^2 - \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \frac{1}{N_i}\sum_{j:T_j\in\mathscr{T}_i} T_j^{(i)}(\mathbf{X}_i)\right)^2 \\
&= \hat{i}_k(\Pi)
\end{aligned}$$

where

$$\text{mse} \coloneqq \text{mse}(F(\mathbf{X}) \coloneqq \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - F_i(\mathbf{X}_i)\right)^2,$$

$$\hat{\text{mse}}_k \coloneqq \text{mse}\left(\Pi(\hat{\mathbf{X}}^{(k)})\right) \coloneqq \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - F_i(\hat{\mathbf{X}}_i^{(k)})\right)^2,$$

$\hat{\mathbf{X}}^{(k)} \coloneqq (X_1, \ldots, X_{k-1}, \hat{X}_k, X_{k+1}, \ldots, X_p)$ and $\hat{X}_k$ has the same distribution as X^k but is independent of the other X_is.

Both $\tilde{i}_k(\Pi)$ and $\hat{i}_k(\Pi)$ are computationally more efficient than $i_k(\Pi)$ because of the different processes that are involved. Both $\tilde{i}_k(\Pi)$ and $\hat{i}_k(\Pi)$ permute values of a particular covariate in the test data set and compute the increase of error on the permuted test set compared to the error on the original test set. Instead of permuting a covariate, $i_k(\Pi)$ removes all values of one covariate from the test set. Then it is necessary to contruct a whole new predictor, because with the removal of a covariate the size of the data set changes. For $\tilde{i}_k(\Pi)$ and $\hat{i}_k(\Pi)$ only one predictor is constructed, the value of covariates are permuted one by one and the size of the training data set stays the same. The computation time of both $\tilde{i}_k(\Pi)$ and $i_k(\Pi)$ are presented in Table 3.3.

n	**p**	**ntree**	$\tilde{i}_k(\Pi)$	$i_k(\Pi)$	**Time to run** $i_k(\Pi)$ **/ Time to run** $\tilde{i}_k(\Pi)$
100	10	500	0.08 secs	0.63 secs	≈ 8
//	200	//	0.82 secs	2.62 mins	≈ 192
//	500	//	1.94 secs	15.76 mins	≈ 487
//	10	1000	0.16 secs	0.63 secs	≈ 4
//	200	//	1.72 secs	2.75 mins	≈ 96
//	500	//	4.04 secs	16.55 mins	≈ 246
500	10	500	0.65 secs	5.27 secs	≈ 8
//	200	//	8.32 secs	23.03 mins	≈ 166
//	500	//	17.98 secs	2.38 hours	≈ 477

Table 3.3.1: Computation time of $\tilde{i}_k(\Pi)$ and $\hat{i}_k(\Pi)$.

An important subject for the next section and chapter is the difference between T-VIM and F-VIM, both theoretically and experimentally.

3.4 Theoretical difference between T-VIM and F-VIM

T-VIM and F-VIM are consistent since permuting the values of one variable at a time corresponds to drawing randomly from its distribution. However, the two measures do not correspond exactly to each other because of computational reasons. The important difference lies in the order of calculations. For T-VIM the performance is calculated tree by tree and all these values are averaged over all trees. For F-VIM the MSE is calculated after two complete random forests are grown: the original forest and a permuted one.

As presented in the previous section, T-VIM and F-VIM can be expressed as the estimates:

$$\tilde{i}_k(\Pi) = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{n_j}\sum_{i=1}^{n_j}\left(Y_i^{(j)} - T_j(\tilde{\mathbf{X}}_i^{(j,k)})\right)^2 - \frac{1}{n_j}\sum_{i=1}^{n_j}\left(Y_i^{(j)} - T_j(\mathbf{X}_i^{(j)})\right)^2\right)$$

and

$$\hat{i}_k(\Pi) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \frac{1}{N_i}\sum_{j:T_j\in\mathscr{T}_i} T_j^{(i)}(\hat{\mathbf{X}}_i^{(k)})\right)^2 - \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \frac{1}{N_i}\sum_{j:T_j\in\mathscr{T}_i} T_j^{(i)}(\mathbf{X}_i)\right)^2$$

The performance of a single tree T_j for $j = 1, \ldots, N$ is obtained with the use of the observations $s_j = (\mathbf{X}_1^{(j)}, Y_1^{(j)}), (\mathbf{X}_2^{(j)}, Y_2^{(j)}), \ldots, (\mathbf{X}_{n_j}^{(j)}, Y_{n_j}^{(j)})$ with $n_j \approx n/3$ of $S_n = (\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n)$ that are, because of bagging, not used to train T_j. The performance of a forest $F_i(\mathbf{X}_i)$ is obtained observation by observation, where all input variables of each observation go through the subset of trees $\mathscr{T}_i = T_1^{(i)}, T_2^{(i)}, \ldots, T_{N_i}^{(i)}$ with $N_i \approx N/3$ in whose construction that observation is not involved.

3.5 Consistency of T-VIM

In Section 2.3 the consistency of the random forest algorithm was discussed: Scornet et al. (2015) showed that Breiman's random forest algorithm is consistent in the context of the additive regression model $= \sum_{j=1}^{p} \Pi_j(\mathbf{X}^{(j)}) + \epsilon$ and with the assumption of independent covariates. The consistency of T-VIM is a different notion and still an open problem according to Gregorutti et al. (2017), especially when covariates are not independent. This study focuses on the performances of the measures when the data is correlated, because this is often what data sets representing reality look like. The main contribution on the consistency of T-VIM is due to Zhu et al. (2012), but like Scornet et al. (2015) also Zhu et al. (2012) assumes independent covariates.

3.6 The effects of correlated covariates

Experimentally one can show that the more correlation between covariates, the more measures are unstable as can be seen in Section 4.2. According to Bühlmann et al. (2013), LASSO has a tendency to select one representative from a group of correlated covariates. For random forests research about the impact of correlated predictors is among others done by Tolosi and Lengauer (2006). There seems to be no consensus on the subject according to Gregorutti et al. (2017), because of a lack of theoretical foundation.

Intuitively, when a covariate is permuted this is done to destroy the relationship between that covariate and Y. When covariates are correlated the permutation also destroys the relationship among the correlated covariates. A higher MSE can then either imply that the permuted covariate is important or that the permuted covariate is dependent on another covariate. Strobl et al. (2007) have come up with conditional importance measures to avoid this problem of not knowing if the permuted covariate is important, dependent on another covariate or both. This method is computationally demanding and therefore difficult to use for large p.

This is why this study does not focus on the conditional importance measures, but focuses on the more efficient T-VIM and F-VIM and aims to find out how they behave in order to find out when they are useful and perform good enough.

3.7 Suggestions

The following two suggestions from the literature will be investigated with a simulation study:

Suggestion 1 by Gregorutti et al. (2017): Tree ensemble variable importance measures are more likely to rank unimportant covariates as important covariates when covariates are correlated.

Suggestion 2 by Genuer et al. (2010): Larger values of $mtry$ increases values of variable importance for important covariates. When $mtry$ is smaller, correlated covariates could score higher, as fewer potentially stronger competing covariates are present at each split Strobl et al. (2007).

Chapter 4

Simulation study

In this chapter the results of several experiments on variable importance measures for regression problems will be presented. The two main actors of this chapter are T-VIM and F-VIM that are described in the previous chapter.

The goal of the simulation study is to discover for what kind of data what measure of VI works best: T-VIM, F-VIM or a classical benchmark.

The following two suggestions from the literature will be investigated:

Suggestion 1 by Gregorutti et al. (2017): Tree ensemble variable importance measures are more likely to rank unimportant covariates as important covariates when covariates are correlated. For positive correlations, the importance of the two correlated covariates decreases when the correlation increases. The value of the prediction error after permutation is then close to the value of the prediction error without permutation and the importance is small.

Suggestion 2 by Genuer et al. (2010): Larger values of $mtry$ (the number of covariates selected at each split) increases value of variable importance for important covariates. When $mtry$ is smaller, correlated covariates could score higher, as fewer potentially stronger competing covariates are present at each split Strobl et al. (2007).

In the experiments here considered, the data sets vary in

- The number of observations n and (noise) covariates p;

The underlying models vary in

- The mean $\mathbb{E}[Y|X]$ which can be either linear or nonlinear;
- The values of the coefficients.

The random forest can vary in parameters. In this study different values for $mtry$ are considered and the effects on and behavior of the measures are studied.

A simulation study is needed, because it is necessary to know the truth about the importance of the variables or at least it is necessary to be able to estimate it. The estimation process, called

sensitivity analysis, will be explained first in this chapter. Then several classical benchmarks are presented and discussed that will be used to show the relative performance and accuracy of the F-VIM and T-VIM in the results. The data are presented and the results: First linear models and then the experiments with nonlinear models.

4.1 Preparation

4.1.1 Data

In the next section on results, various data sets are generated from various models of which some are linear and some are nonlinear. For a linear model the mean $\mu(x)$ is linear in x and for a nonlinear model the mean $\mu(x)$ is nonlinear in x. All covariates $\mathbf{X}_i = (X_{i1}, \ldots, X_{ip})$ have a multinormal distribution $X \sim N_p(U, \Sigma)$ with $U = (\mu_1, \ldots, \mu_p)'$ and

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho_{1,2}\sigma_1\sigma_2 & \cdots & \rho_{1,p}\sigma_1\sigma_p \\ \rho_{2,1}\sigma_2\sigma_1 & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho_{p-1,p}\sigma_{p-1}\sigma_p \\ \rho_{p,1}\sigma_p\sigma_1 & \cdots & \rho_{p,p-1}\sigma_p\sigma_{p-1} & \sigma_p^2 \end{pmatrix}$$

The error $\epsilon_i \sim N(0, 1)$ is independent of the covariates. In the legend of the figures one can see the number of observations n, the number of covariates p and the number of nonzero coefficients s. In the caption one can see the MSE of the predictor, whether there are pairwise correlated variables and the vector of coefficients β.

4.1.2 Classical benchmark

T-VIM and F-VIM are compared to each other and to a classical benchmark. A classical benchmark that is well known for good performance for linear models is LASSO.

4.2 Results

4.2.1 Linear models

In this section several results of experiments are shown. The first results show what happens when the values of observations n and covariates p are varied. Then the results are presented of the introduction of correlated covariates. In all linear models the coefficients vector $\beta = (1, 2, 3, 1, 2, 3)$ for $p = 6$ and $\beta = (1, 2, 3, 1, 2, 3, 0, \ldots, 0)$ for $p > 6$. For all simulations the random forest algorithm is ran 50 times and the results are averaged. This gives a more stable result.

Independent covariates: sensitivity of measures to n and p

Parameters *ntree* and *mtry* are set to their default values ($ntree = 500$ and $mtry = p/3$ for regression). All boxplots are based on 50 runs of the RF algorithm. In all experiments, the first six values of coefficient vector β are nonzero. If there are more than six values in β or more than six covariates, then these coefficients have value 0. When the amount of noise variables is large, the graphs are cut off after the 16th variable.

Six figures are presented that have no pairwise correlation among the covariates. The first three figures have a large number of observations ($n = 500$), the last three have a smaller number of observations ($n = 100$). The first three and last three figures vary in number of covariates (p): $p = 6$, $p = 200$ and $p = 500$.

For all experiments done for linear models, LASSO outperforms the other measures. The performance can be seen in 4.2.1 and all of the performances are similar.

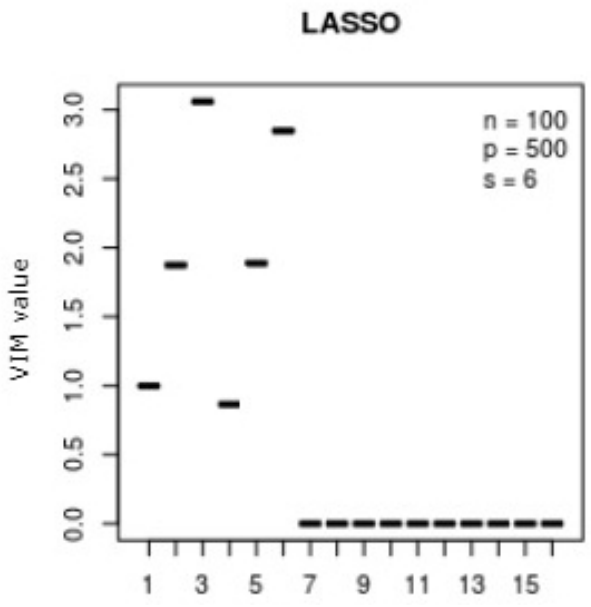

Figure 4.2.1: Performance of VIM of LASSO.

In Figure 4.2.2 $n = 500$, $p = 6$ and $s = 6$. It stands out that the variability of the F-VIM is larger than the variability of the other measures. Also, both T-VIM and F-VIM rank variable 6 higher than variable 3 while their values are the same.

In the second experiment in Figure 4.2.3 both T-VIM and F-VIM now rank variable 3 and 6 almost equal and both can't distinguish variable 1 and 4 from noise variables 7 and 8.

Where first variable 6 was ranked higher by both T-VIM and F-VIM, now both measures rank variable 3 highest in Figure 4.2.4 and again both measures can't distinguish variable 1 and 4 from the noise variables.

With $n = 100$ there is more variability of values for both T-VIM and F-VIM as can be seen in Figure 4.2.5. In this experiment it stands out that T-VIM ranks variable 3 on average higher than 6 and F-VIM ranks variables 6 on average higher than variable 3.

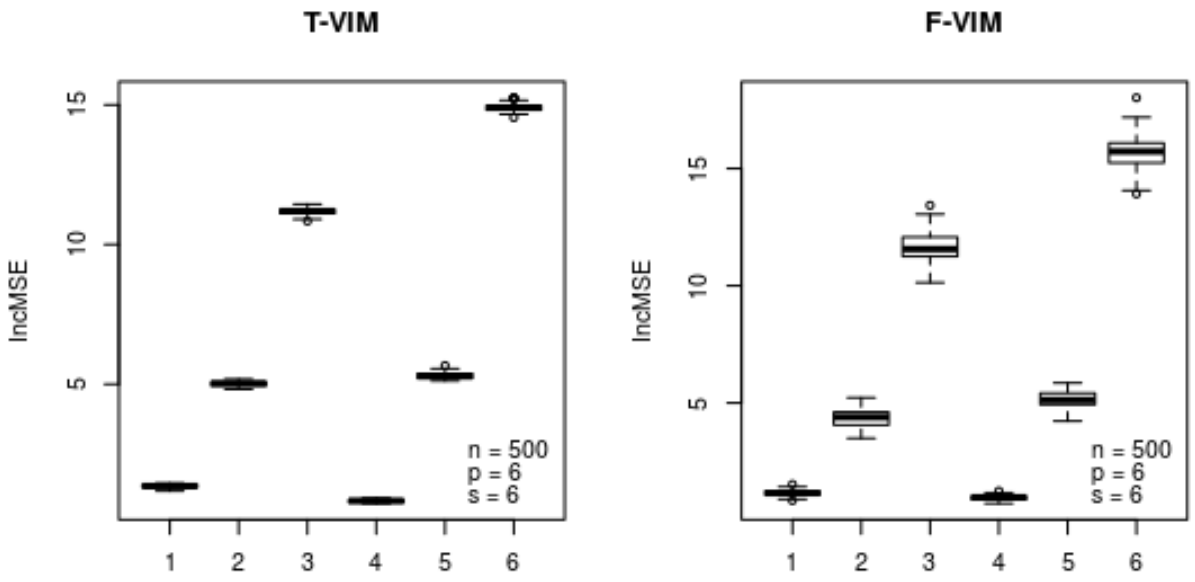

Figure 4.2.2: VIMs with mse ≈ 4.5.

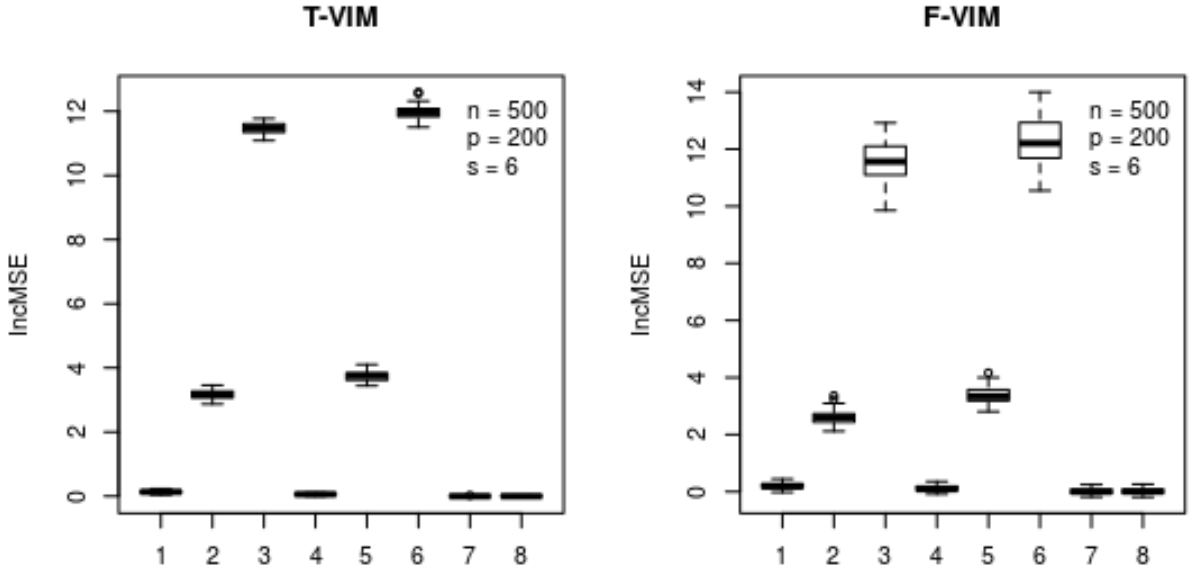

Figure 4.2.3: VIMs with mse ≈ 8.3.

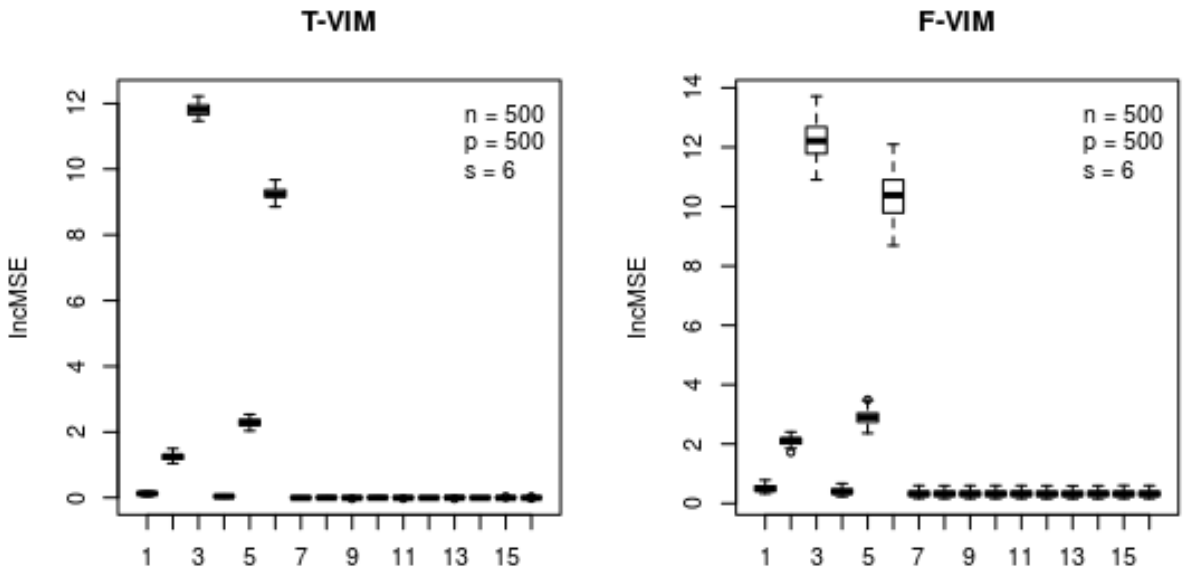

Figure 4.2.4: VIMs with mse ≈ 9.4.

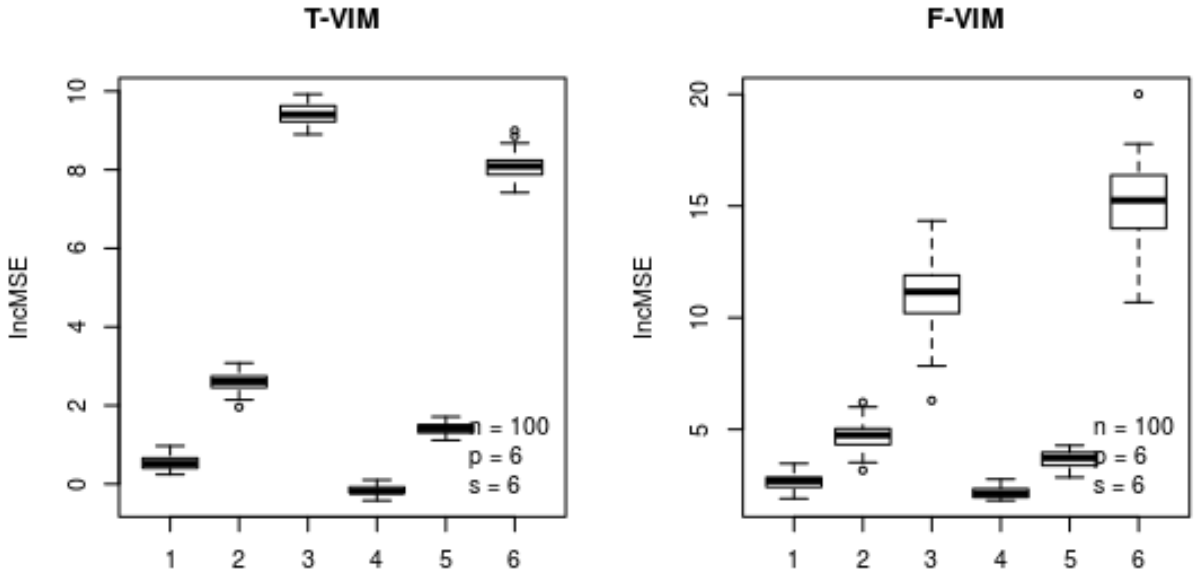

Figure 4.2.5: VIMs with mse ≈ 8.1.

For $p = 200$ in Figure 4.2.6 it stands out that F-VIM has a larger variability of values but has similar mean values as when $p = 6$. T-VIM ranks variable 6 a lot higher for $p = 200$ than for $p = 6$.

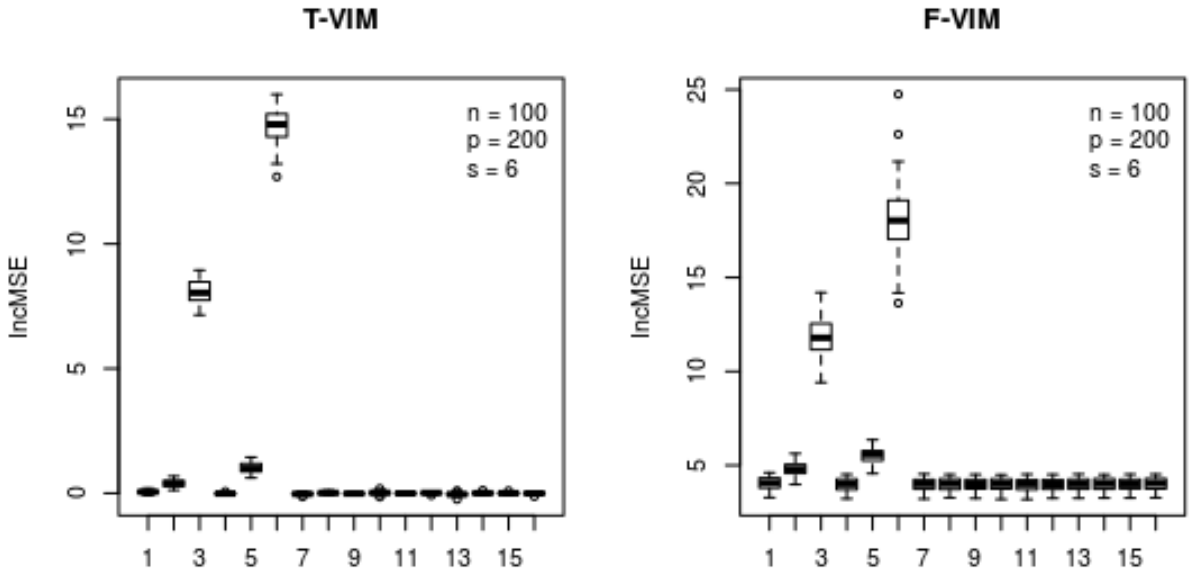

Figure 4.2.6: VIMs with mse ≈ 14.6.

For $p = 500$ in Figure 4.2.7 the MSE decreased compared to $p = 6$ and $p = 200$. Both T-VIM and F-VIM have a larger variability of values and rank variable 3 highest and variable 6 second and quite low.

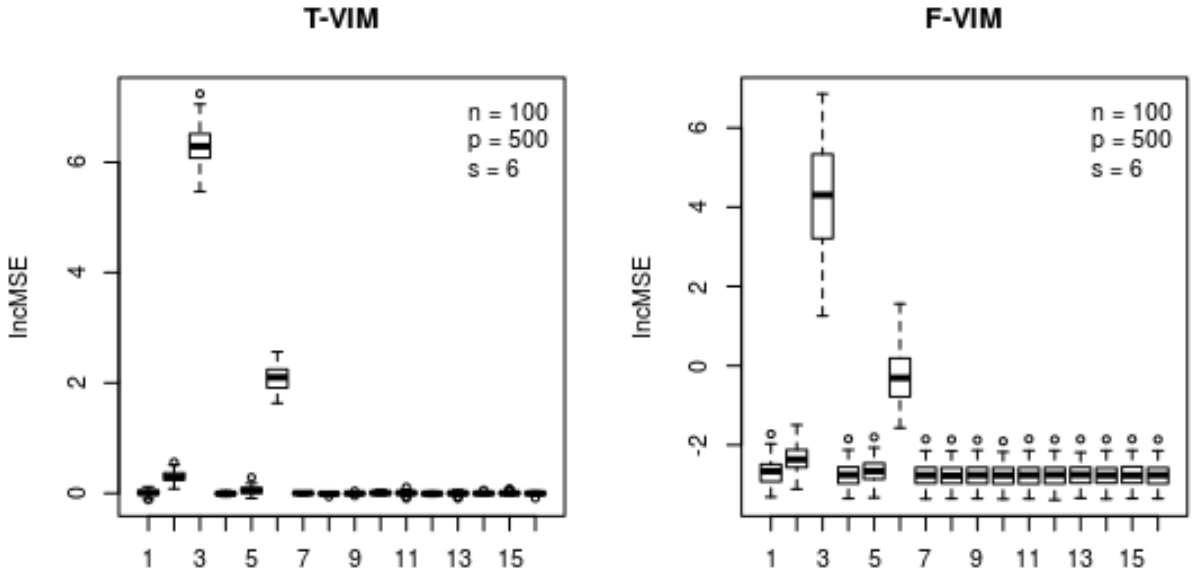

Figure 4.2.7: VIMs with mse ≈ 18.4.

Correlated covariates: sensitivity of measures to n and p

The following six graphs are results of experiments with pairwise correlated covariates X_1, X_2 and X_3 with a correlation coefficient of $\rho = 0.9$.

In Figure 4.2.8 the effects of the correlation can be clearly seen for T-VIM and F-VIM. For T-VIM all pairwise correlated variables have a much higher increase of MSE and the ranking of variable 1 is very close to variable 6. F-VIM gives a much higher value to variable 1 and 2, but all other variables have about the same value as when there was no pairwise correlation. It stands out that F-VIM still picks up the high value of variable 6 and T-VIM does not.

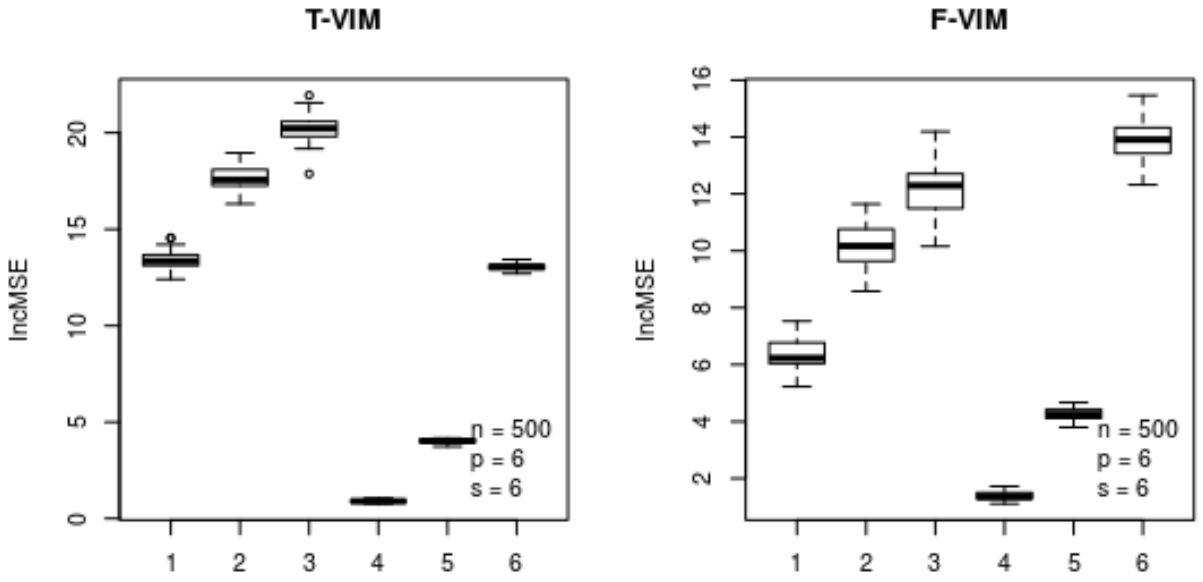

Figure 4.2.8: VIMs with mse ≈ 4.

For $p = 200$ in Figure 4.2.9 covariate 1 and 6 have the same score for T-VIM while for F-VIM covariate 2 and 6 have the same score. T-VIM performs similar to when $p = 6$ and F-VIM performs worse with ranking covariate 6.

And for $p = 500$ in Figure 4.2.10 T-VIM performs in scoring covariates 4-6 all lower than covariates 1-3. F-VIM performs similar to when $p = 200$.

For $n = 100$ and $p = 6$ in Figure 4.2.11 T-VIM mixes up the order of covariates 1-3 and scores covariates 4-6 lower than covariates 1-3. F-VIM again gives covariate 6 a proper score.

For $p = 200$ in Figure 4.2.12 T-VIM has got the order of covariate 1-3 right and F-VIM scores covariate 6 lower than for $p = 6$.

For $p = 500$ in Figure 4.2.13 both T-VIM and F-VIM can not distinguish covariates 4-6 from noise covariates.

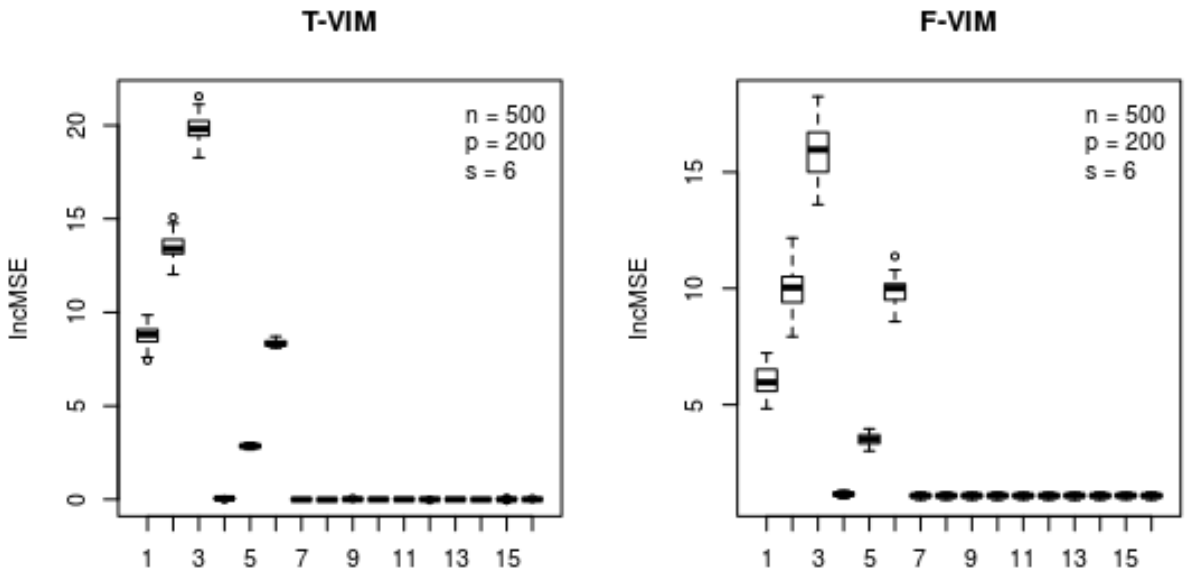

Figure 4.2.9: VIMs with mse ≈ 6.7.

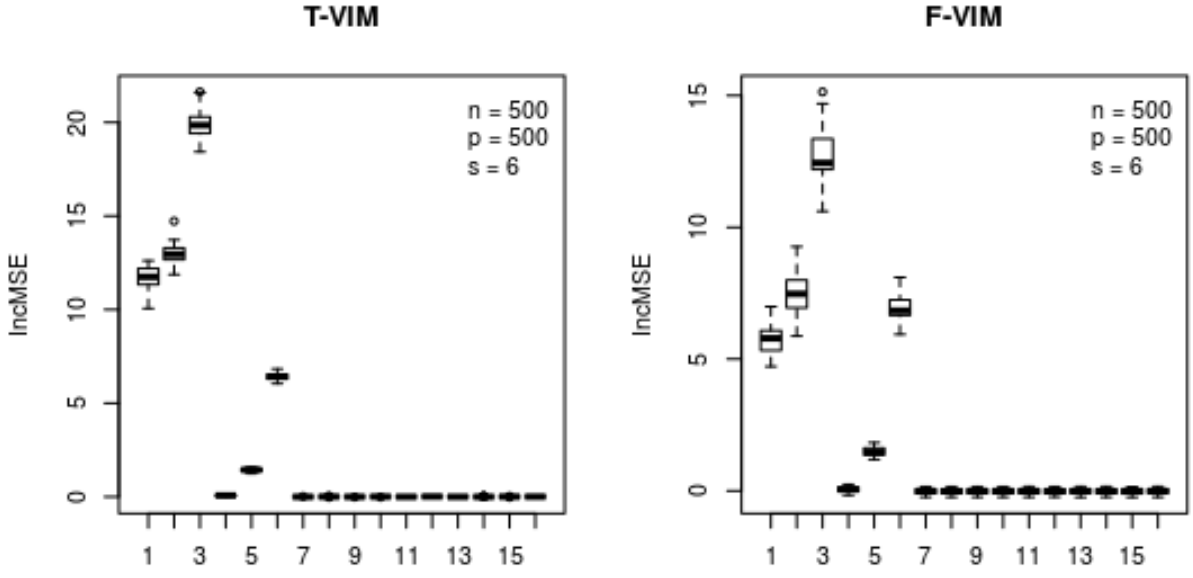

Figure 4.2.10: VIMs with mse ≈ 7.8.

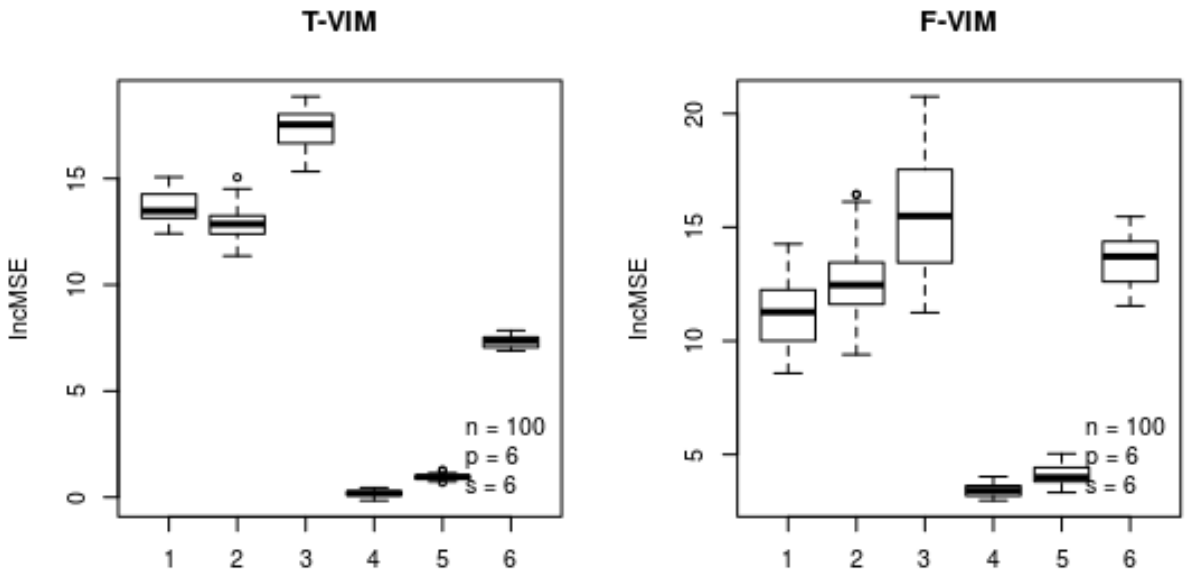

Figure 4.2.11: VIMs with mse ≈ 7.6.

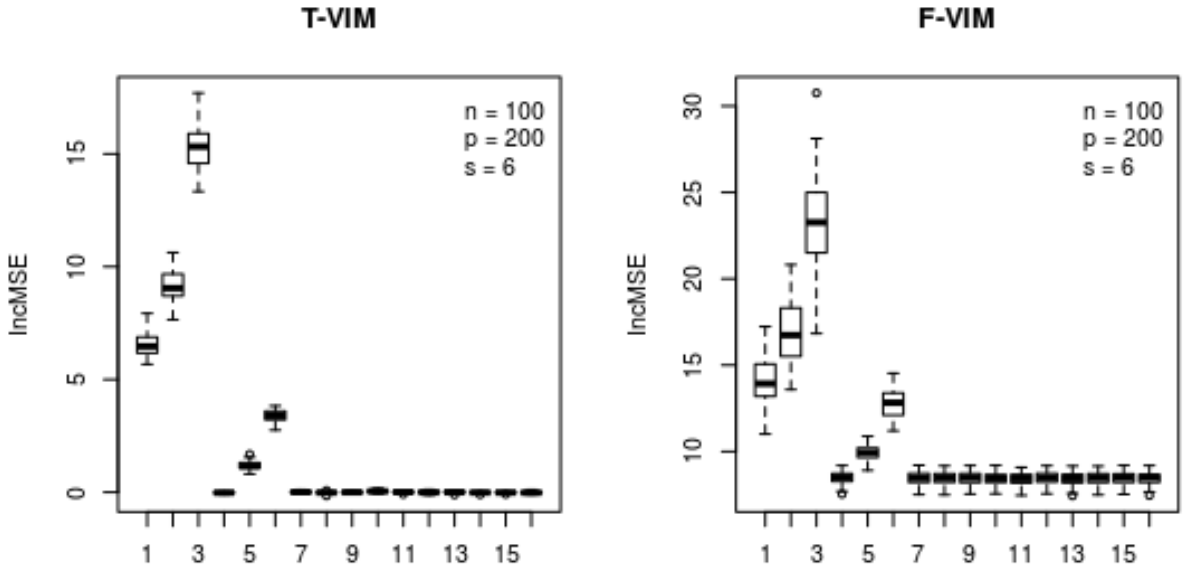

Figure 4.2.12: VIMs with mse ≈ 14.2.

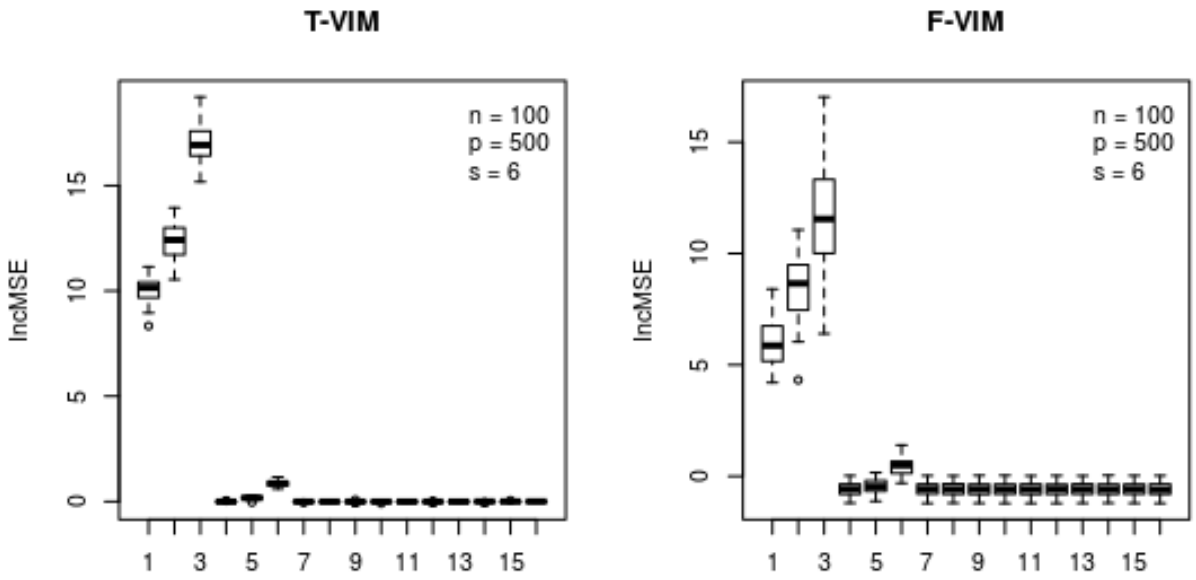

Figure 4.2.13: VIMs with mse ≈ 14.6.

Sensitivity of measures to *mtry*

The parameters of random forests can be tuned and one of them is *mtry*. With the tuning of *mtry* most of the time *mtry* is larger than its default value. The performance of the measures for different values of p and for default and optimized values of *mtry* are presented in Figure 4.2.14-Figure 4.2.19.

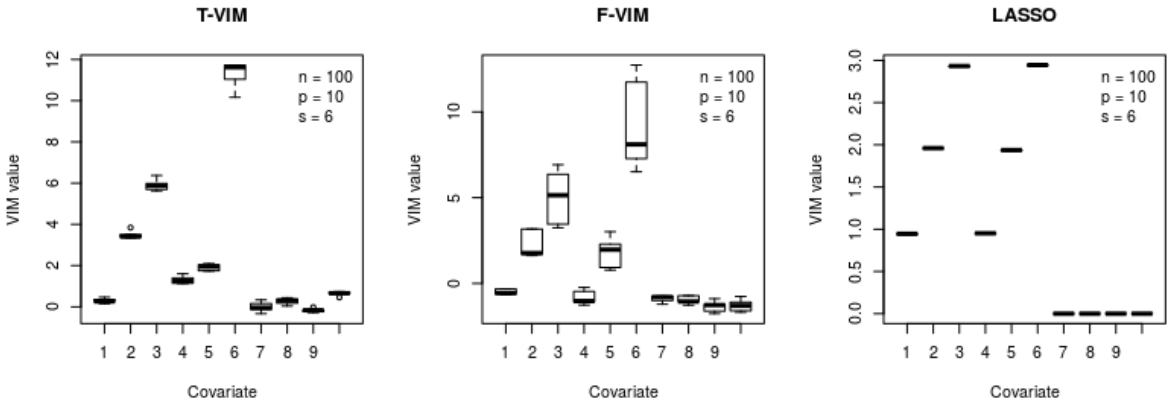

Figure 4.2.14: Independent covariates and value of *mtry* is default value: mse ≈ 10.5

Variability of measures

The variability of measures is attained by running the algorithm 50 times for different distributions of $\mathbf{X}$. The performance of the measures is presented in Figure 4.2.20-4.2.23.

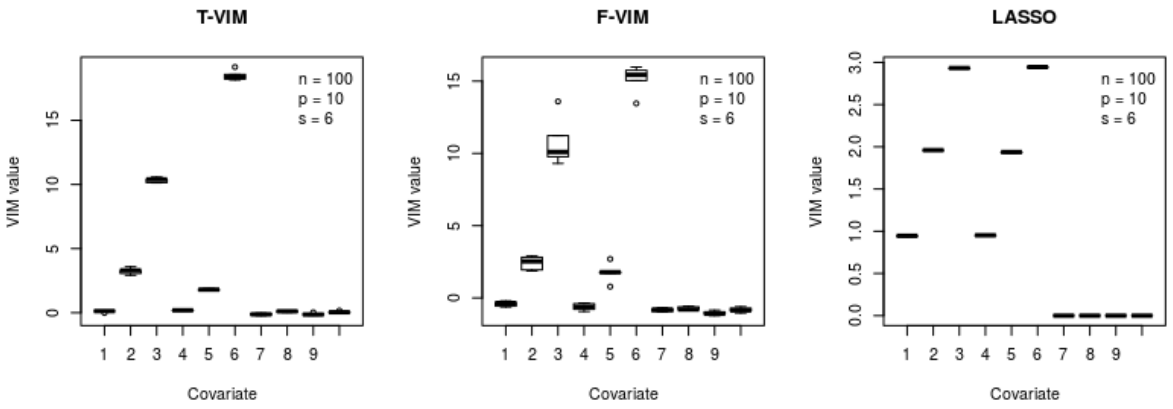

Figure 4.2.15: Independent covariates and value of *mtry* is optimized: mse $\approx$ 8.5

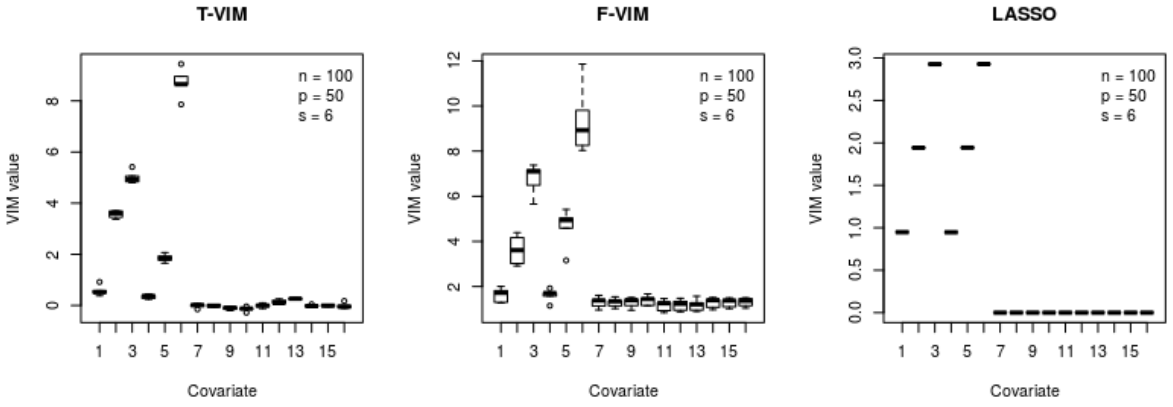

Figure 4.2.16: Independent covariates and value of *mtry* is default value: mse $\approx$ 14.6

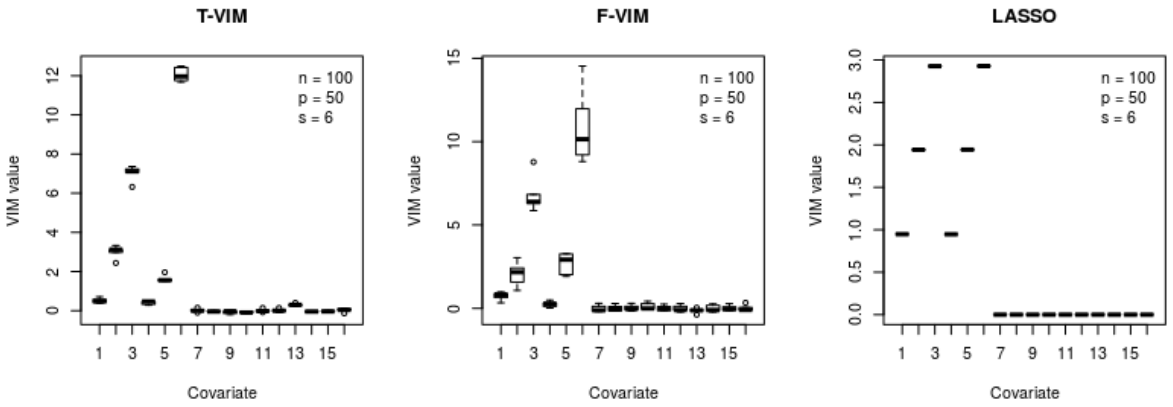

Figure 4.2.17: Independent covariates and value of *mtry* is optimized: mse $\approx$ 13.5

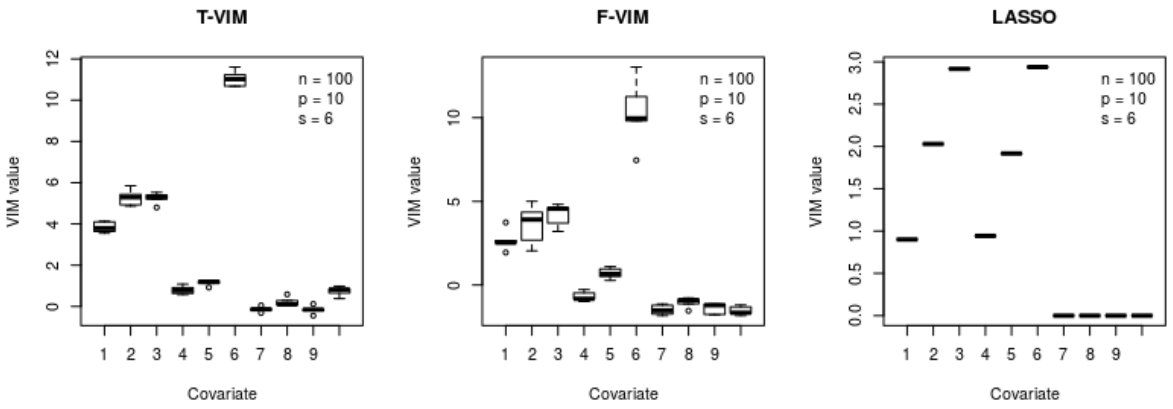

Figure 4.2.18: Correlated covariates and value of *mtry* is default value: mse ≈ 11

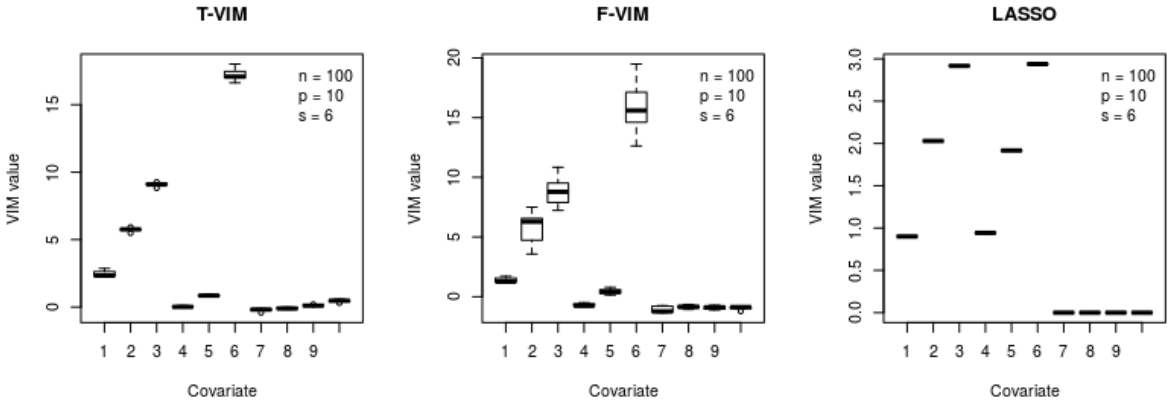

Figure 4.2.19: Correlated covariates and value of *mtry* is optimized: mse ≈ 9.4

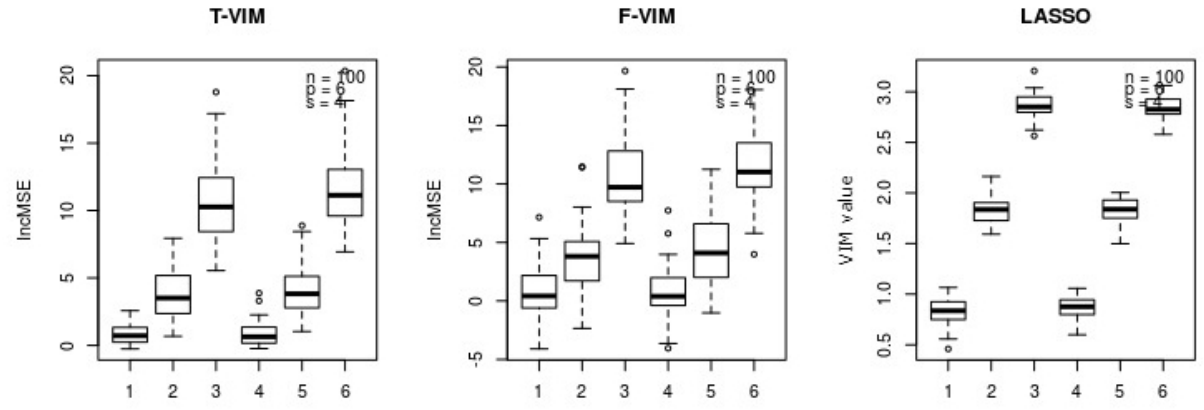

Figure 4.2.20: Variability of measures with independent covariates and small amount of observations.

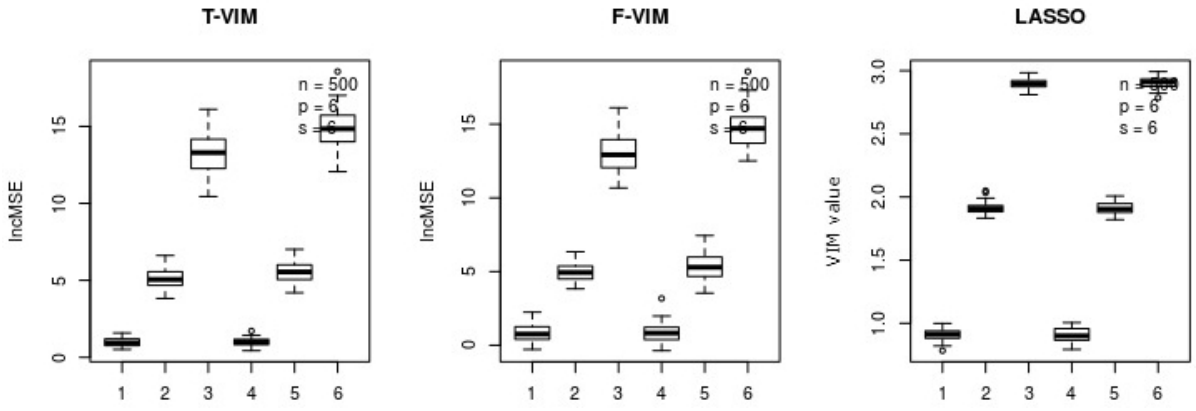

Figure 4.2.21: Variability of measures with independent covariates and large amount of observations.

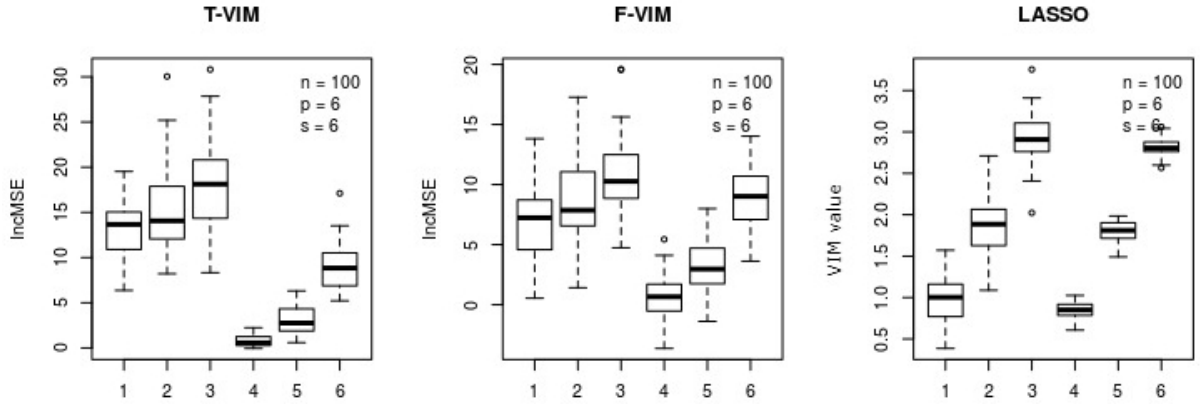

Figure 4.2.22: Variability of measures with correlated covariates and small amount of observations.

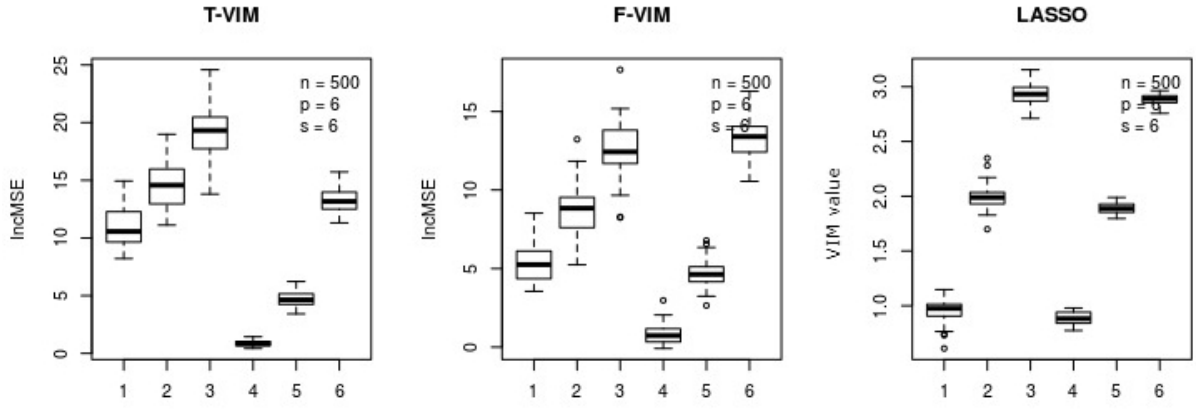

Figure 4.2.23: Variability of measures with correlated covariates and large amount of observations.

4.2.2 Nonlinear models

In this section experiments are done with five nonlinear models (Fang et al. (2004)):

$$y_i = \frac{x_1 \cdot e^{x_2}}{1 + x_3^2} + \epsilon_i$$

$$y_i = x_1 \cdot x_2 \cdot x_3 + \epsilon_i$$

$$y_i = \frac{x_1 + x_2}{e^{x_3}} + \epsilon_i$$

$$y_i = x_1 \cdot \ln(0.5 + x_2^2 + x_2^2) + \epsilon_i$$

$$y_i = e^{(1-x_1)} \cdot \ln(1 + (x_2 \cdot x_3^2) + \epsilon_i$$

These five models are picked in order to research what is the difference between different measures for a wide range of different underlying models. 'True' coefficients are estimated with global partial derivatives on data sets with $n = 10000$ observations.

The first nonlinear model is

$$y_i = \frac{x_1 \cdot e^{x_2}}{1 + x_3^2} + \epsilon_i.$$

In Figure 4.2.24 T-VIM, F-VIM and LASSO have a good performance. LASSO performs best and assigns value 0 to covariate 2 and 3. T-VIM has least variability amongst the RF runs and F-VIM has the largest variability. Remarkable is that the VI values for F-VIM are a lot larger than for T-VIM.

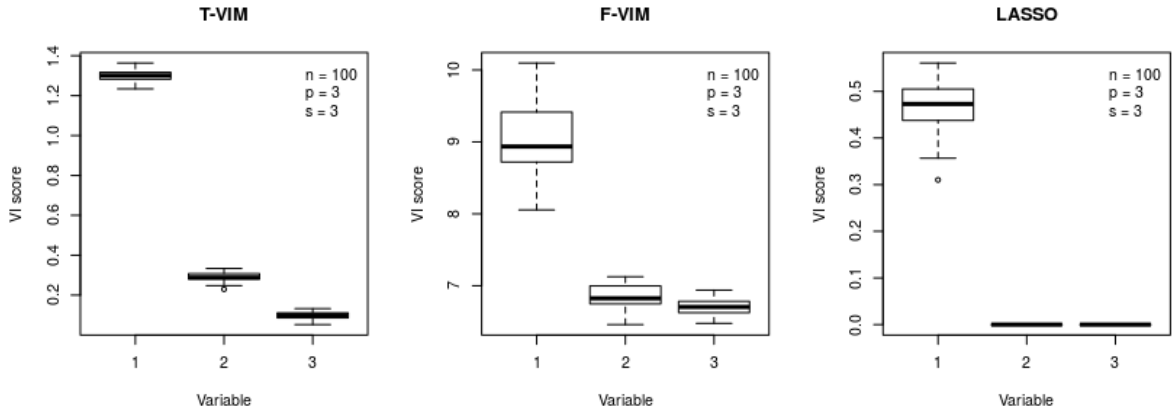

Figure 4.2.24: VIMs for $\beta \approx (1.1, 0, 0)$, mse ≈ 0.5, $U = 0$, $diag(\Sigma) \approx 1$ and no correlated covariates.

For $p = 6$ in Figure 4.2.25 the results are similar: T-VIM has the smallest variability for covariate 1, LASSO is correct in assigning value 0 to covariate 2-6. T-VIM has a bit more trouble assigning noise variables value zero than F-VIM has.

In Figure 4.2.26 pairwise correlation of the covariates is added. LASSO can not grasp the values of covariates 2 and 3. T-VIM and F-VIM have a similar performance.

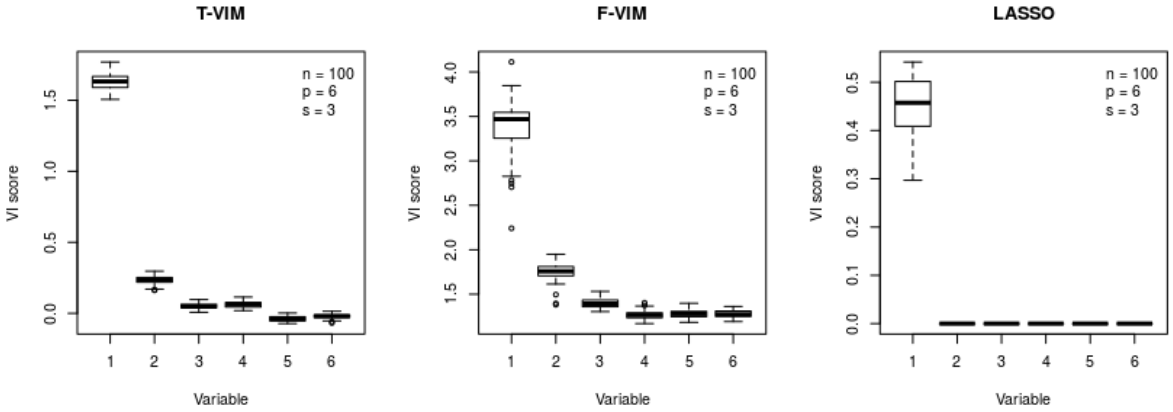

Figure 4.2.25: VIMS for $\beta \approx (1.1, 0, \ldots, 0)$, mse ≈ 1, $U = 0$, $diag(\Sigma) \approx 1$ and no correlated covariates.

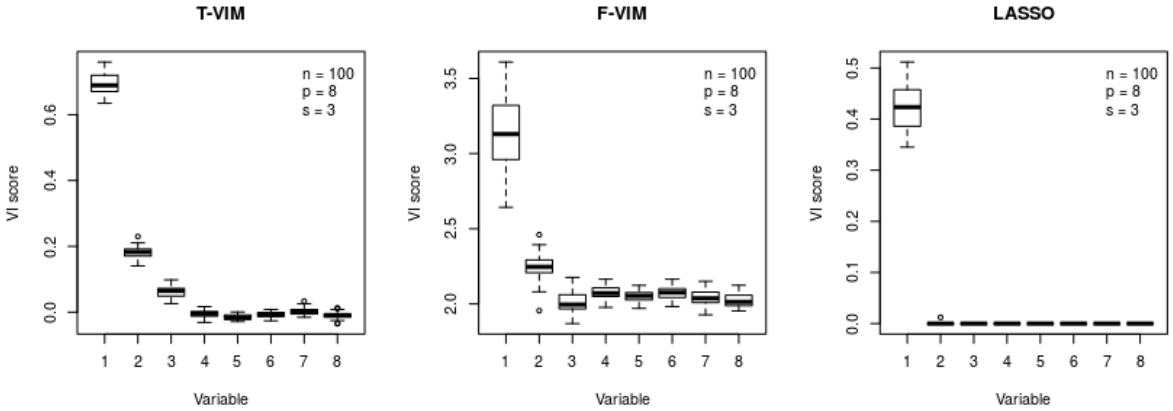

Figure 4.2.26: VIMs for $\beta \approx (1, 0.4, -0.3, 0, \ldots, 0)$, mse ≈ 0.6, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = 0.5$.

In Figure 4.2.27 U and Σ are less standard: $U = (1, 2, 2)$ and $diag(\Sigma) = (1.5, 1.05, 3.1)$. All measures capture the most important variable. Only T-VIM and F-VIM also capture variable 2 and 3.

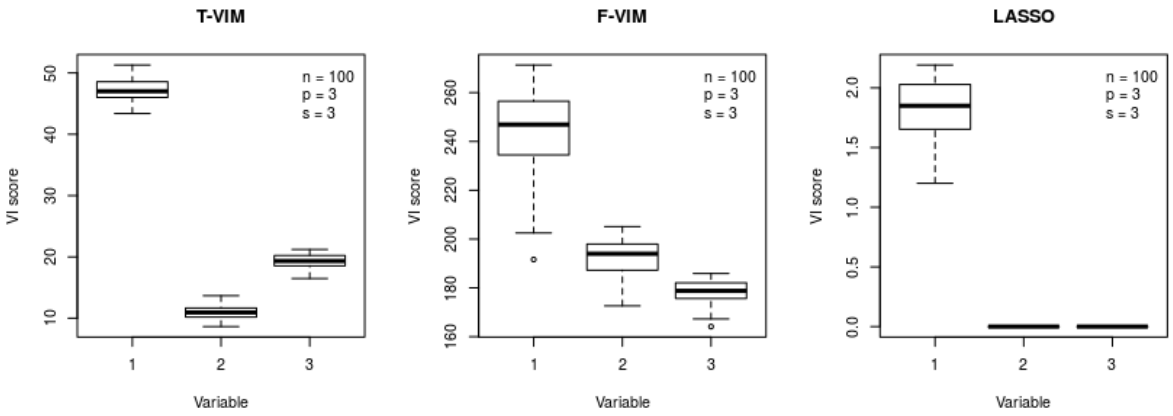

Figure 4.2.27: VIMs for $\beta \approx (4.5, 0.9, -0.9)$, mse ≈ 27.8, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and X_1, X_2 and X_3 are pairwise correlated with $cov(X_1, X_2) = -0.5$,$cov(X_2, X_3) = -0.35$ and $cov(X_1, X_3) = 0.8$.

The second nonlinear model is

$$y_i = x_1 \cdot x_2 \cdot x_3 + \epsilon_i.$$

In Figure 4.2.28 all measures capture the three nonzero coefficients. F-VIM shows a large variability for the RF runs and both T-VIM and F-VIM have difficulty assigning value zero to the noise covariates.

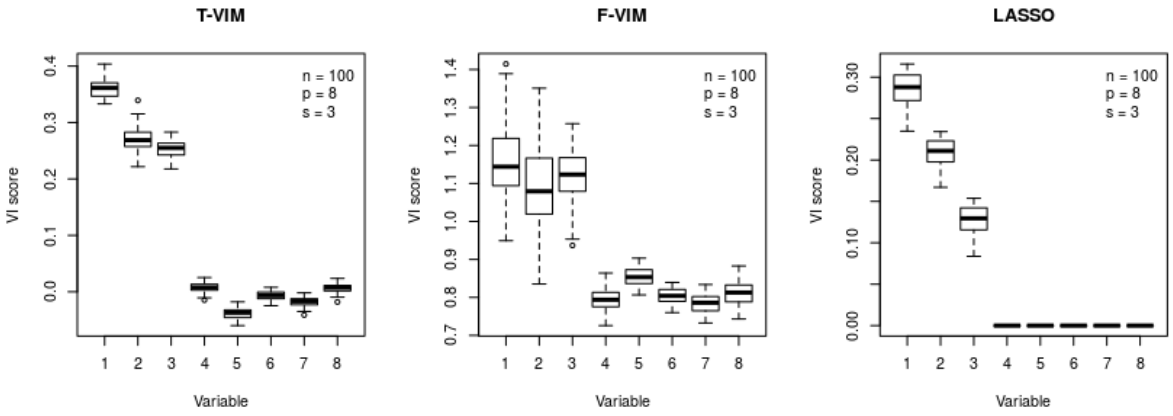

Figure 4.2.28: VIMs for $\beta \approx (0.5, 0.5, 0.5, 0, \ldots, 0)$, mse ≈ 0.6, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = 0.5$.

In Figure 4.2.29 only X_1 and X_2 are pairwise correlated. LASSO struggles with the interaction terms. F-VIM performs best and T-VIM seems more sensitive to the correlated covariates X_1 and X_2.

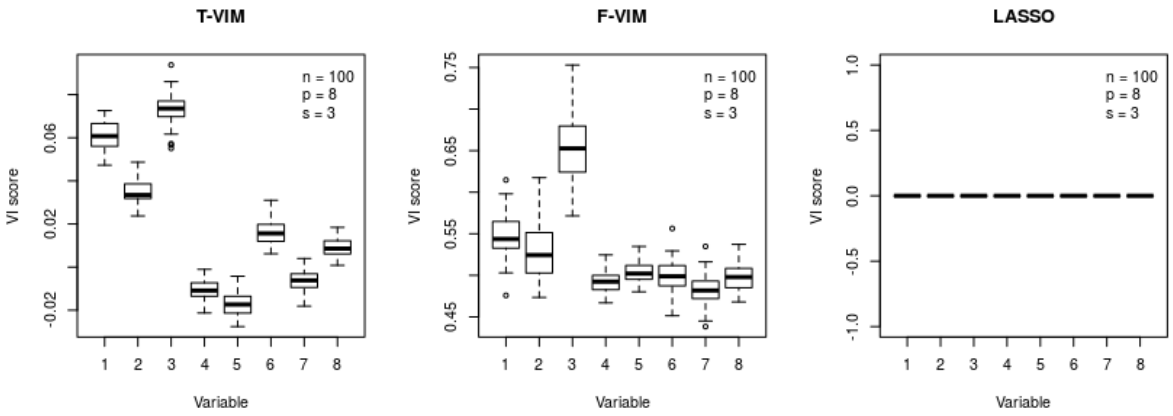

Figure 4.2.29: VIMs for $\beta \approx (0, 0, 0.5, 0, \ldots, 0)$, mse ≈ 0.4, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = (0.5, 0, 0)$.

In Figure 4.2.30 U and Σ are less standard: $U = (1, 2, 2)$ and $diag(\Sigma) = (1.5, 1.05, 3.1)$. Both T-VIM and F-VIM have a good performance and LASSO has not.

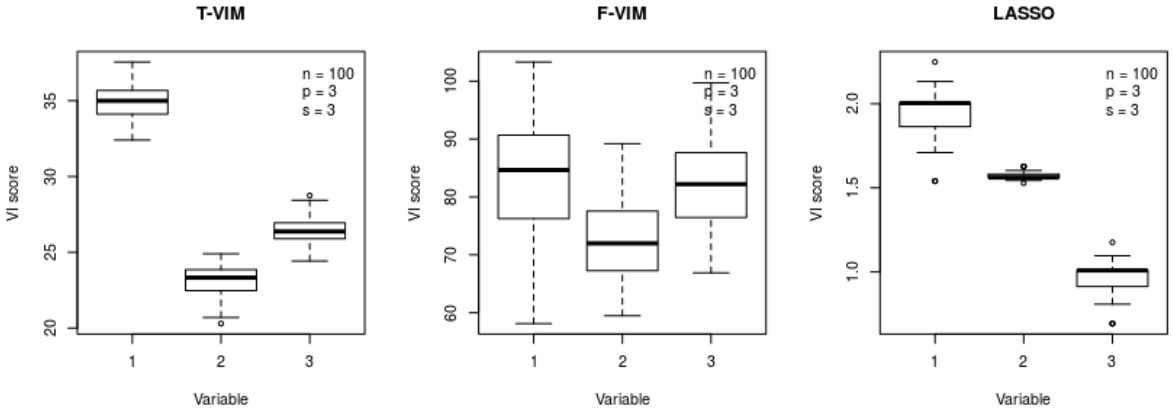

Figure 4.2.30: VIMs for $\beta \approx (4.4, 1.2, 2.5)$, mse ≈ 13.1, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and X_1, X_2 and X_3 are pairwise correlated with $cov(X_1, X_2) = 0.5$,$cov(X_2, X_3) = 0.35$ and $cov(X_1, X_3) = -0.8$.

In Figure 4.2.31 all measures have difficulty capturing the right ranking.

The third model is

$$y_i = \frac{x_1 + x_2}{e^{x_3}} + \epsilon_i.$$

In Figure 4.2.32 all measures capture the two most important variables. Both T-VIM and F-VIM have difficulty assigning value zero to noise covariates.

In Figure 4.2.33 F-VIM captures the most important variable. T-VIM seems to be sensitive to the correlated covariates. LASSO has a bad performance.

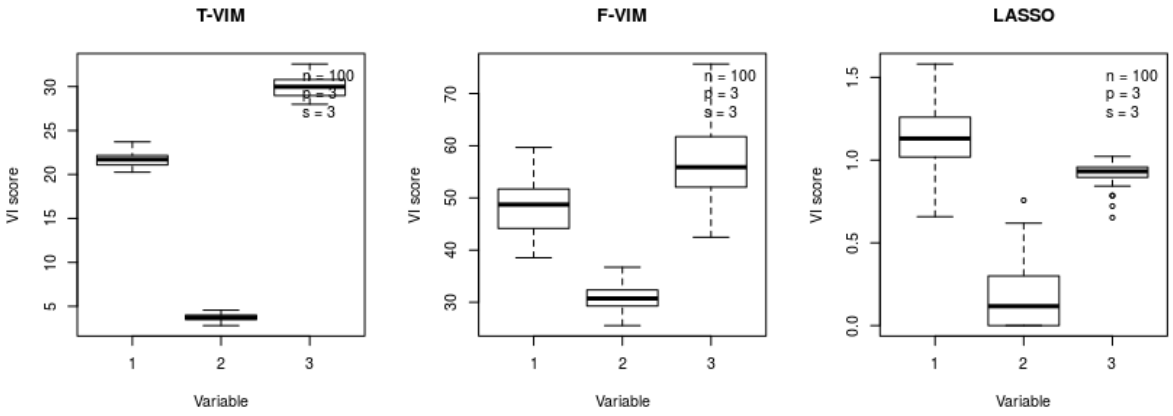

Figure 4.2.31: VIMs for $\beta \approx (3.6, 2.8, 1.5)$, mse ≈ 12.1, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and X_1, X_2 and X_3 are pairwise correlated with $cov(X_1, X_2) = -0.5$,$cov(X_2, X_3) = -0.35$ and $cov(X_1, X_3) = 0.8$.

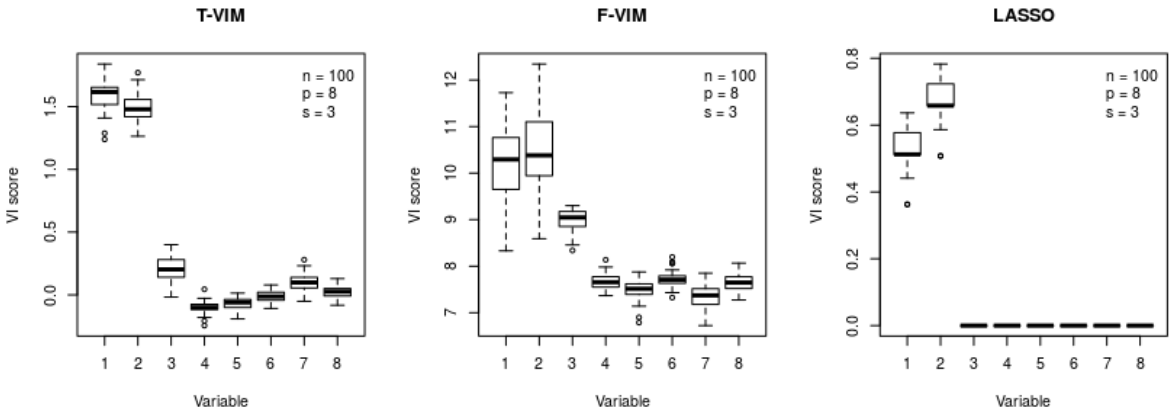

Figure 4.2.32: VIMs for $\beta \approx (1.7, 1.7, 0.1, 0, \ldots, 0)$, mse ≈ 4, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and no correlation.

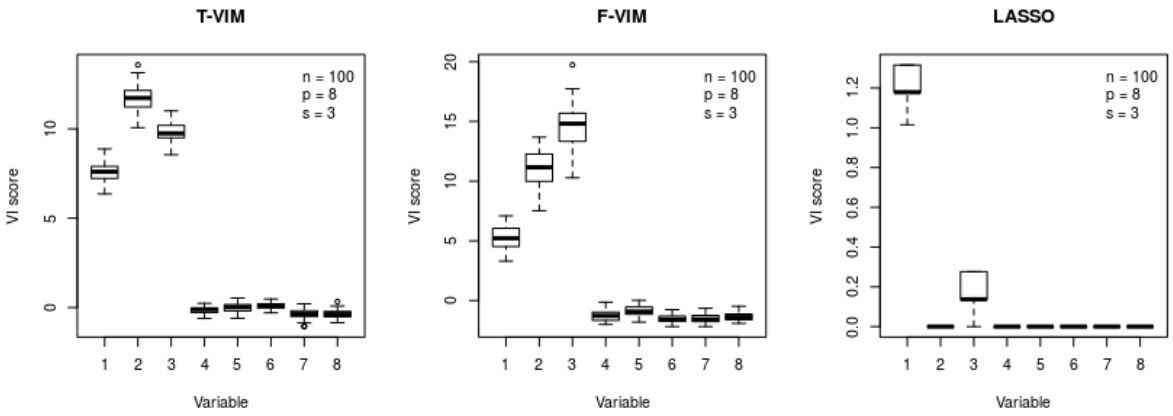

Figure 4.2.33: VIMs for $\beta \approx (1.7, 1.7, 3.2, 0, \ldots, 0)$, mse ≈ 9.1, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = 0.9$.

In Figure 4.2.34 only X_1 and X_2 are correlated. T-VIM performs best and LASSO worst.

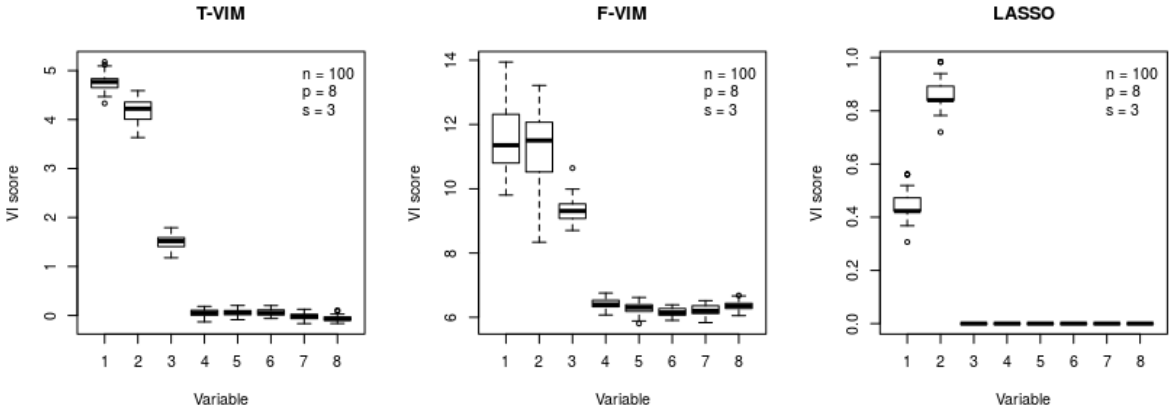

Figure 4.2.34: VIMs for $\beta \approx (1.7, 1.7, 0.1, 0, \ldots, 0)$, mse ≈ 3, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = (0.9, 0, 0)$.

In Figure 4.2.35 U and Σ are less standard: $U = (1, 2, 2)$ and $diag(\Sigma) = (1.5, 1.05, 3.1)$. This experiment gives an insight in how the measures behave with a negative coefficient.

Figure 4.2.36 is quite similar but all measures just struggle a lit bit more.

Figure 4.2.37 is again quite similar with different correlation coefficients.

The fourth model is

$$y_i = x_1 \cdot \ln(0.5 + x_2^2 + x_2^2) + \epsilon_i.$$

In Figure 4.2.38 all measures perform well. LASSO has the largest variability.

In Figure 4.2.39 the first three covariates are correlated. All measures perform well.

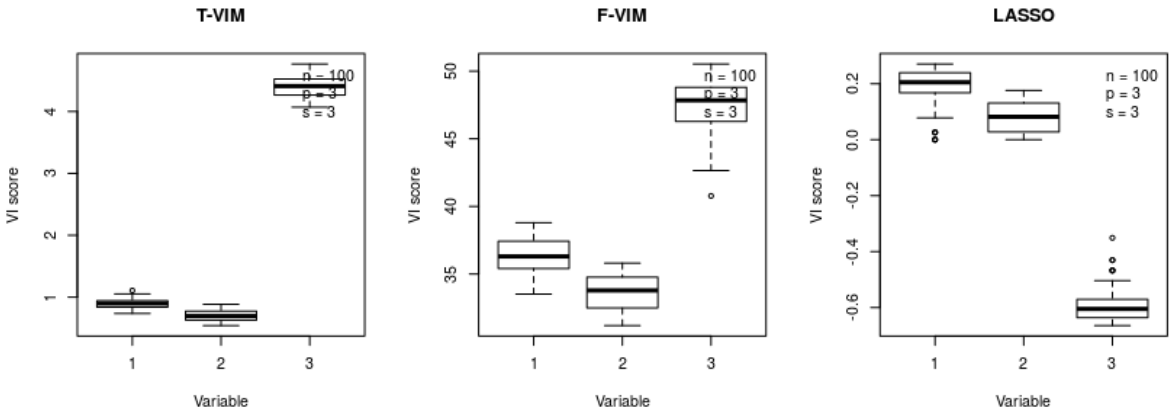

Figure 4.2.35: VIMs for $\beta \approx (0.6, 0.6, -1.8)$, mse ≈ 2, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and no correlation.

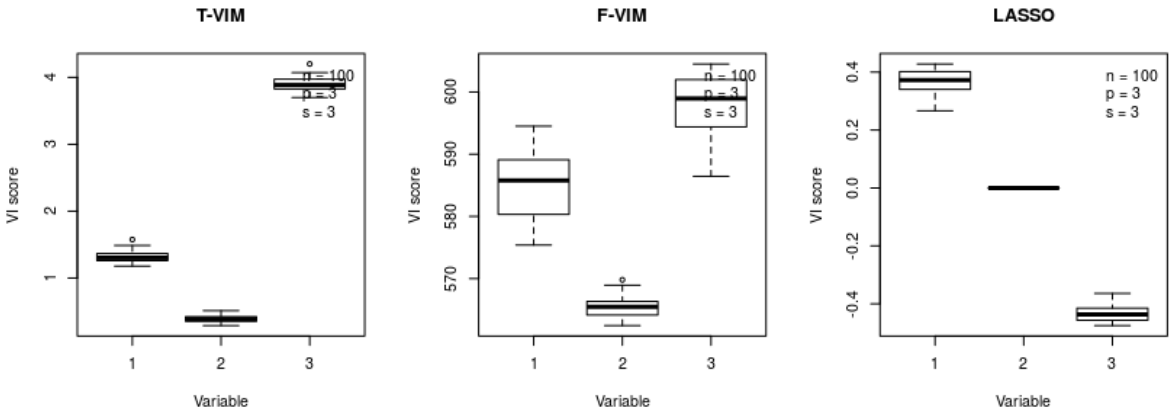

Figure 4.2.36: VIMs for $\beta \approx (0.6, 0.6, -2.1)$, mse ≈ 0.4, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and X_1, X_2 and X_3 are pairwise correlated with $cov(X_1, X_2) = 0.5$,$cov(X_2, X_3) = 0.35$ and $cov(X_1, X_3) = -0.8$.

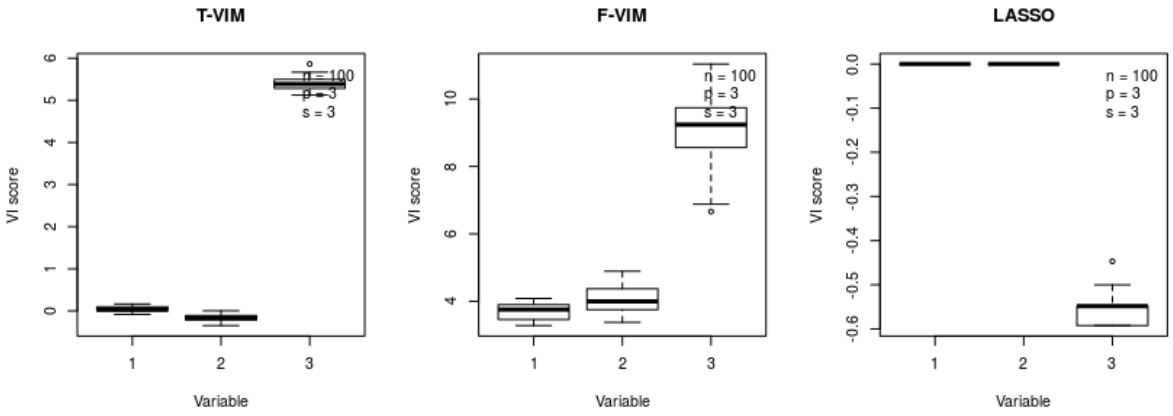

Figure 4.2.37: VIMs for $\beta \approx (0.7, 0.7, -1.6)$, mse ≈ 2.6, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and X_1, X_2 and X_3 are pairwise correlated with $cov(X_1, X_2) = -0.5$,$cov(X_2, X_3) = -0.35$ and $cov(X_1, X_3) = 0.8$.

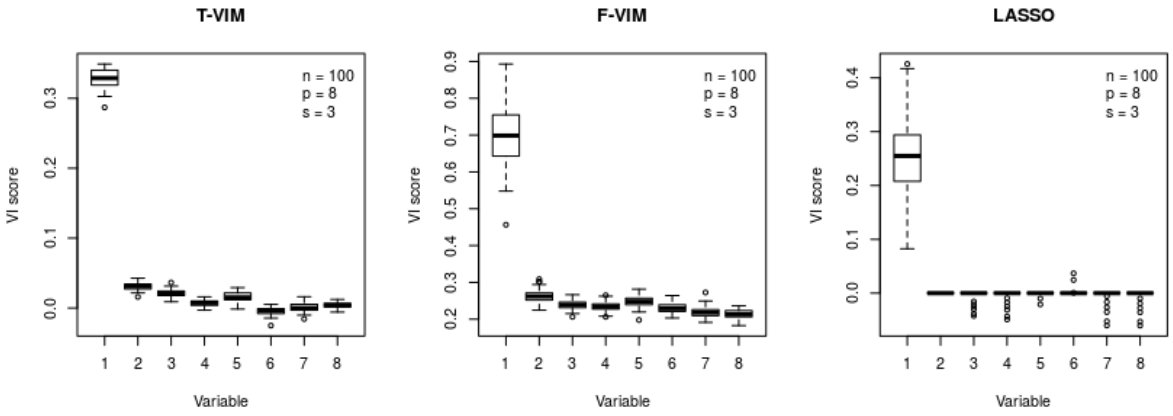

Figure 4.2.38: VIMs for $\beta \approx (0.7, 0, \ldots, 0)$, mse ≈ 0.4, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and no correlation.

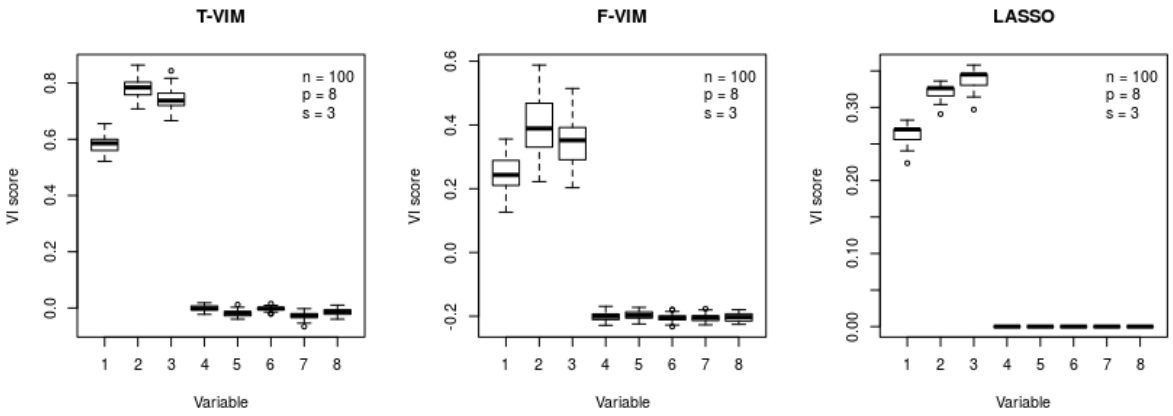

Figure 4.2.39: VIMs for $\beta \approx (0.5, 0.5, 0.5, 0, \ldots, 0)$, mse ≈ 0.4, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = 0.9$.

In Figure 4.2.40 U and Σ are less standard: $U = (1, 2, 2)$ and $diag(\Sigma) = (1.5, 1.05, 3.1)$. All measures perform well.

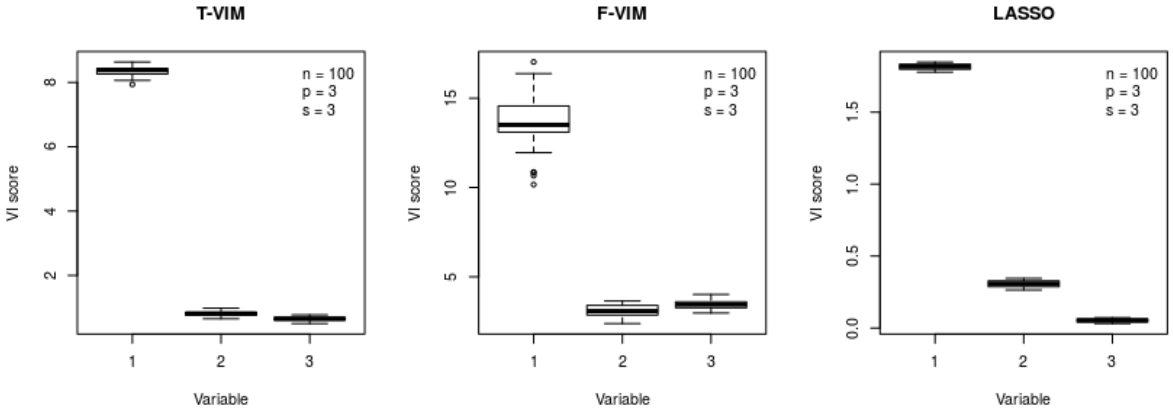

Figure 4.2.40: VIMs for $\beta \approx (2.2, 0.4, 0.3)$, mse ≈ 1.2, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and no correlation.

The measures also perform well in the presence of correlated covariates as can be seen in Figure 4.2.41.

The fifth model is

$$y_i = e^{(1-x_1)} \cdot \ln(1 + (x_2 \cdot x_3^2) + \epsilon_i.$$

In Figure 4.2.42 LASSO performs worst. All measures struggle capturing the true ranking. F-VIM assigns positive values to zero coefficients of covariates 2-8. It does assign covariate 1 as the most important one, which T-VIM does not.

The MSE is very high in Figure 4.2.43. Both 4.2.43 as well as 4.2.44 can not distinguish among the correlated covariates.

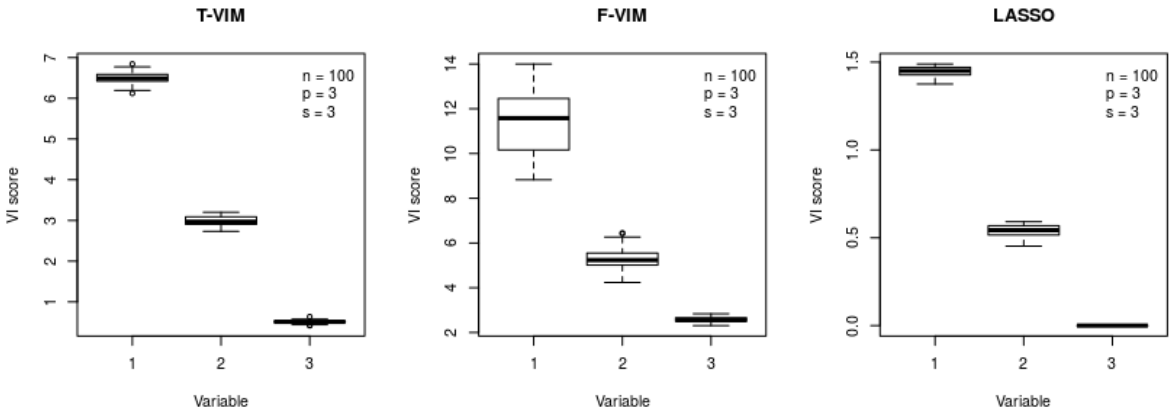

Figure 4.2.41: VIMs for $\beta \approx (2.2, 0.6, 0.2)$, mse ≈ 1.1, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and X_1, X_2 and X_3 are pairwise correlated with $cov(X_1, X_2) = 0.5$,$cov(X_2, X_3) = 0.35$ and $cov(X_1, X_3) = -0.8$.

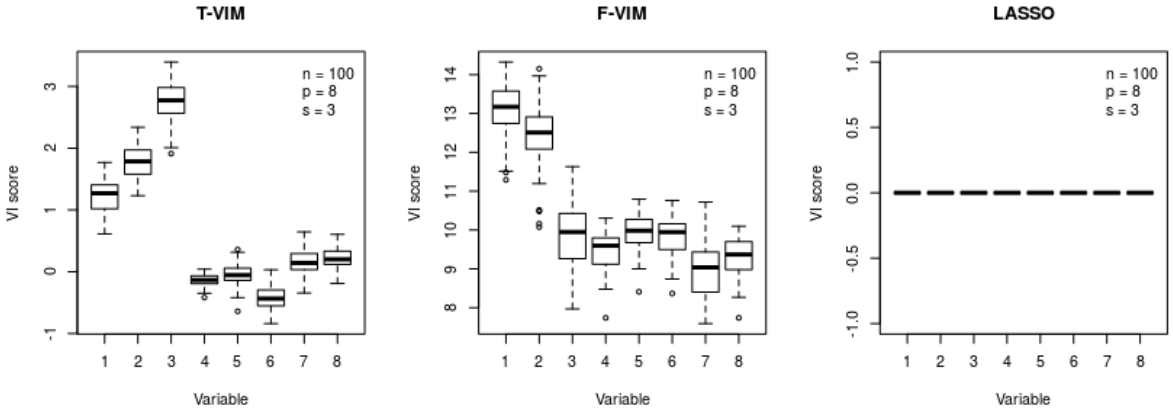

Figure 4.2.42: VIMs for $\beta \approx (-1.9, 0, -0.10, \ldots, 0)$, mse ≈ 10.2, $U = 0$, $diag(\Sigma) \approx 1$ and no correlation.

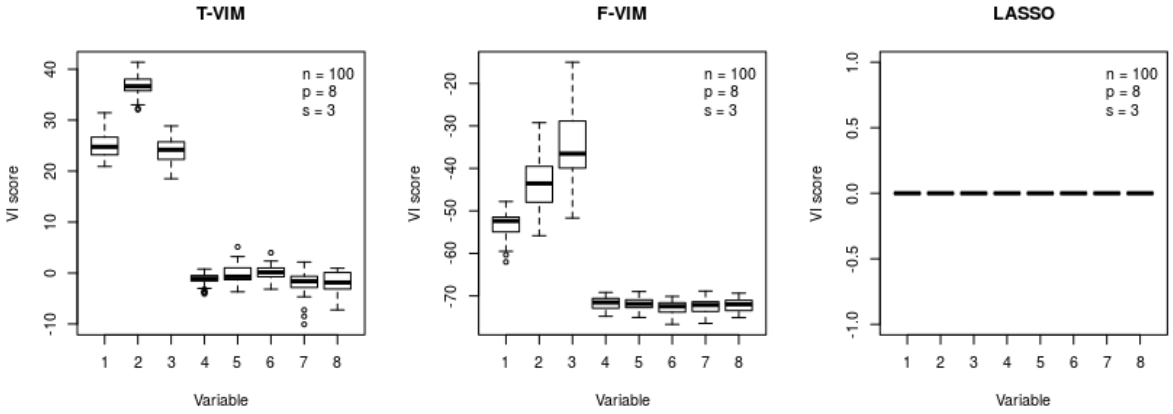

Figure 4.2.43: VIMs for $\beta \approx (-5.2, -2.5, -2.4, 0, \ldots, 0)$, mse ≈ 92.9, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = 0.9$.

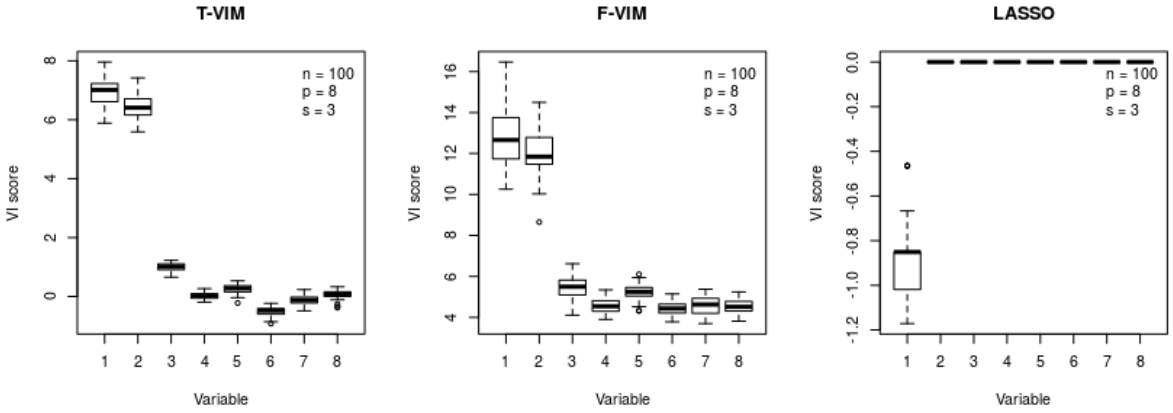

Figure 4.2.44: VIMs for $\beta \approx (-2.8, -1.8, 0, \ldots, 0)$, mse ≈ 7.4, $U = 0$, $diag(\Sigma) \approx 1$ and X_1, X_2 and X_3 are pairwise correlated with $\rho = (0.9, 0, 0)$.

In Figure 4.2.45 U and Σ are less standard: $U = (1, 2, 2)$ and $diag(\Sigma) = (1.5, 1.05, 3.1)$. T-VIM and F-VIM have a good performance. LASSO has not.

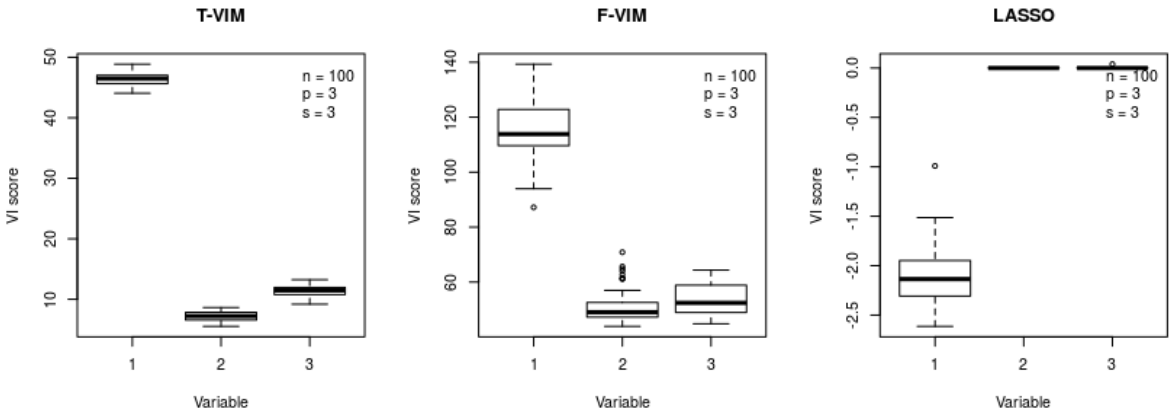

Figure 4.2.45: VIMs for $\beta \approx (-5.3, 1.8, 1.3)$, mse ≈ 24.9, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and no correlation.

In Figure 4.2.46 all measures have difficulty capturing the true ranking.

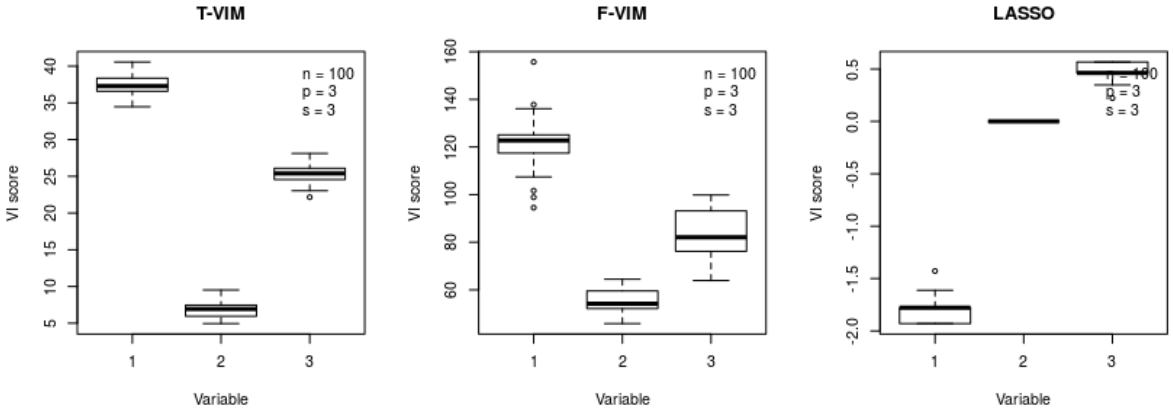

Figure 4.2.46: VIMs for $\beta \approx (-5.5, 2, 1.2)$, mse ≈ 24.2, $U = (1, 2, 2)$, $diag(\Sigma) = (1.5, 1.05, 3.1)$ and X_1, X_2 and X_3 are pairwise correlated with $cov(X_1, X_2) = 0.5$,$cov(X_2, X_3) = 0.35$ and $cov(X_1, X_3) = -0.8$.

4.2.3 Results

For the generated linear models with and without pairwise correlated covariates:

- LASSO performs always well and better for linear models than T-VIM and F-VIM under the assumed data-generating models with and without pairwise correlated covariates.

For the generated linear models without pairwise correlated covariates:

- F-VIM has a larger variability over the RF runs;
- The larger p, the larger the difference between the most important variables and the variables that follow;
- When n decreases there is more variability over RF runs for both T-VIM and F-VIM.

For the generated linear models with pairwise correlated covariates X_1, X_2 and X_3:

- Correlated variables get a higher score than when they were uncorrelated: for T-VIM this is more extreme than for F-VIM when p is small and the larger p gets the worse F-VIM performs and for $n = 100$ this trend is more extreme than for $n = 500$.

For the generated nonlinear models:

- LASSO sometimes captures the most important variables and not the less important ones and sometimes fails to perform;
- Both T-VIM and F-VIM sometimes have trouble assigning value zero to noise covariates;
- T-VIM seems more sensitive to correlated covariates and assigns them higher values than it should;
- When covariates are correlated it is more difficult for both T-VIM and F-VIM to distinguish their ranking.

4.2.4 Evaluation of suggestions

Suggestion 1 by Gregorutti et al. (2017): Tree ensemble variable importance measures are more likely to rank unimportant covariates as important covariates when covariates are correlated.

Evaluation of suggestion 1: This is clearly visible in among others Figure 4.2.8 and Figure 4.2.9 where covariates 1-3 score high compared to covariates 4-6. T-VIM is more sensitive to this tendency than F-VIM. This must be because T-VIM counts the increase of MSE tree by tree, while F-VIM grows complete forest and then calculates the increase of MSE. More theoretical study is needed to confirm this.

Suggestion 2 by Genuer et al. (2010): Larger values of m increase values of variable importance for important covariates. When m is smaller, correlated covariates could score higher, as fewer potentially stronger competing covariates are present at each split Strobl et al. (2007).

Evaluation of suggestion 2: This is clearly visible for both T-VIM and F-VIM in Figure 4.2.18 and Figure 4.2.19.

Chapter 5

Conclusions, discussion and acknowledgement

5.1 Conclusions

The objective of this research was to investigate the behaviour of variable importance measures for random forests on a range of data sets with numerical Y: data sets with either $n > p$ or $p > n$, with either pairwise correlated or uncorrelated covariates, with various distributions of the covariates and with either linear or nonlinear underlying models.

The first questions considered was: What should the measure of variable importance be under the data-generating model? According to us it should be Leave-One-Covariate-Out (LOCO), but because of the computational inefficiency of LOCO a permutation measure was considered. This permutation measure was estimated by two empirical measures: T-VIM and F-VIM. Experiments were done for different linear and nonlinear models with different values for n and p and correlated and independent covariates.

This study can confirm the two suggestions:

1. Tree ensemble variable importance measures are more likely to rank unimportant covariates as important covariates when covariates are correlated. And T-VIM is more sensitive to this tendency than F-VIM.
2. Larger values of m increase values of variable importance for important covariates.

This shows that it is very important to tune *mtry*, especially in the presence of pairwise correlated covariates. And when *mtry* is tuned, the performance of the measures might be good enough.

5.2 Discussion

Measuring variable importance is a difficult task, especially for nonlinear models and in the presence of pairwise correlated covariates. Future research can be done on a wider range of models and distributions for X. And of course, it is desirable that there will be more theoretical justifications for the consistency of the random forest algorithm, the consistency of the measures of variable importance based on both independent and pairwise correlated covariates and the difference between T-VIM and F-VIM. Further research can be done on the LOCO measure. This measure could be used with a smaller amount of the covariates that are somehow selected with another method.

5.3 Acknowledgement

A special word of thanks goes out to my supervisors José de Sousa Jorge Ferreira from RIVM, who guided me the first part of my graduation project, and Nestor Parolya.

Bibliography

Biau, G. and Scornet, E. A random forest guided tour. *Test*, 25:197–227, 2016.

Breiman, L. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

Breiman, L. Random forests. *Machine Learning*, 45:5–32, 2001a.

Breiman, L. Statistical modeling: The two cultures. *Statistical Science*, 16:199–231, 2001b.

Breiman, L. and Ihaka, R. Nonlinear discrimant analysis via scaling and ace. *Technical report*, 1984.

Bühlmann, P., Rütimann, P., van de Geer, S., and Zhang, C.-H. Correlated variables in regression: clustering and sparse estimation. *J. Stat. Plan. Inference*, 143:1835–1858, 2013.

Efron, B. *The jackknife, the bootstrap and other sampling plans*, volume 38. CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, 1982.

Fang, S., Gertner, G.Z., and Anderson, A.A. Estimation of sensitivity coefficients of nonlinear model input parameters which have a multinormal distribution. *Comput. Phys. Comm.*, 157: 9–16, 2004.

Ferreira, J. Exercises on statistical prediction. 2019.

Ferreira, J., Nicolaie, A., and Wong, A. Statistical prediction. 2019.

Friedman, J., Hastie, Trevor, and Tibshirani, R. *The elements of statistical learning*. Springer series in statistics, New York, 2001.

Genuer, R., Poggi, J.-M., and Tuleau-Malot, C. Variable selection using random forests. *Pattern Recognition Letters*, 31:2225–2236, 2010.

Gregorutti, B., Michel, B., and Saint-Pierre, P. Correlation and variable importance in random forests. *Statistics and Computing*, 27(3):659–678, 2017.

Hand, David J. Classifier technology and the illusion of progress. *Statistical Science*, 21(1):1–15, 2006.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R.J., and Wasserman, L. Distribution-free predictive inference for regression. *Journal of the American Statistical Association 113*, 523:1094–1111, 2017.

Nembrini, S., König, I.R., and Wright, M.N. The revival of the gini importance? *Bioinformatics*, 2018. URL `https://arxiv.org/abs/1804.03515`.

Scornet, E., Biau, G., and Vert, J.-P. Consistency of random forests. *The Annals of Statistics*, 43:1716–1741, 2015.

Strobl, C., Boulesteix, A.-L., Zeileis, A., and Hothorn, T. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, 8(1):25, 2007.

Tolosi, L. and Lengauer, T. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27:1–33, 2006.

Yeh, I-Cheng. Analysis of strength of concrete using design of experiments and neural networks. *Journal of Materials in Civil Engineering, 18(4): 597-604*, 2006. URL `https://archive.ics.uci.edu/ml/datasets/concrete+compressive+strength`.

Zhu, R., Zeng, D., and Kosorok, M.R. Reinforcement learning trees. *Technical report*, 2012.

Appendix: R code

```r
library(mvtnorm)
library(randomForest)
library(Matrix)
library(glmnet)
library(plyr)
library(splines)
library(tuneRanger)
library(mlr)
library(MASS)

#------------------ Data: Compressive strength of concrete ----------------------

library(readr)
concrete <- read_csv("Data/concrete.csv")
data.set <- as.data.frame(concrete)

names(data.set)[1] <- "cement"
names(data.set)[2] <- "blast.furnace.slag"
names(data.set)[3] <- "fly.ash"
names(data.set)[4] <- "water"
names(data.set)[5] <- "superplaticizer"
names(data.set)[6] <- "coarse.aggregate"
names(data.set)[7] <- "fine.aggregate"
names(data.set)[8] <- "age"
names(data.set)[9] <- "compressive.strength"

ratio <- vector(length=nrow(data.set))
for(i in (1:nrow(data.set))){
  ratio[i] <- data.set$cement[i]/data.set$water[i]
}
ratio <- round(ratio,1)

data.set$c.to.w.ratio <- ratio
data.set <- data.set[, c(1:8,10,9)]

#------------------ Partition of tree ----------------------

library(tree)
library(rpart)
library(rpart.plot)

cs <- data.set$compressive.strength
age <- data.set$age
c.to.w.ratio <- data.set$c.to.w.ratio

cem.tr <- tree(cs ~ age + c.to.w.ratio, data.set)
cem.tr
summary(cem.tr)
cem.tr1 <- snip.tree(cem.tr, nodes = c(20,21,11,7,6))
cem.tr1
summary(cem.tr1)
par(pty = "s")
plot(data.set[, 8],data.set[, 9],
     xlab="age", ylab="c.to.w.ratio")
```

```
partition.tree(cem.tr)

CP <- 0.001
fit.tree <- rpart(data.set$compressive.strength ~ data.set$age +
data.set$c.to.w.ratio,data=data.set, control=rpart.control(minsplit=10),cp=CP)
good.name <- names(data.set)[ncol(data.set)]
prp(fit.tree,faclen = 0, cex = 0.8, extra = 1, main=paste("Prediction of ",
good.name,sep="")
,digits=3)

#------------------ Optimizing nodesize for tree predictor ----------------------

CP <- 0.00000
good.name <- names(data.set)[ncol(data.set)]

### Default value of nodesize ###

predictions <- NULL
for(i in (1:nrow(data.set))){
  data.set.i <- data.set[-i,]
  fit.tree.i <- rpart(as.formula(paste(names(data.set)[ncol(data.set)],"~.",sep="")),
  data=data.set.i,cp=CP)
  prediction.i <- predict(fit.tree.i,type="vector",newdata=data.set[i,-ncol(data.set),
  drop=FALSE])
  predictions <- c(predictions,prediction.i)
}
pred.er <- predictions - data.set[,ncol(data.set)]
mse.plot <- mean(pred.er^2)

explanation1 <- paste("n: ",nrow(data.set),"; mse: ",
formatC(mse.plot,digits=2,format="f"),"; mae: ",formatC(mae,digits=2,format="f"),
                      "\n pev: ",formatC(pev,digits=2,format="f"),sep="")
plot(data.set[,ncol(data.set)], predictions, xlab="Actual compressive strength (MPa)",
     ylab="Predicted compressive strength (MPa)",
     main=explanation1); abline(a=0,b=1,col="red")

### Optimizing nodesize ###

mse <- NULL
for (j in 1:10){
  predictions <- NULL
  for(i in (1:nrow(data.set))){
    data.set.i <- data.set[-i,]
    fit.tree.i <- rpart(as.formula(paste(names(data.set)[ncol(data.set)],"~.",sep="")),
    data=data.set.i,control=list(minbucket=j),cp=CP)
    prediction.i <- predict(fit.tree.i,type="vector",newdata=data.set[i,-ncol(data.set),
    drop=FALSE])
    predictions <- c(predictions,prediction.i)
  }
  pred.er.j <- predictions - data.set[,ncol(data.set)]
  mse[j] <- mean(pred.er.j^2)
  print(j)
}

mse.min <- min(mse)
nodesize <- which.min(mse)
mae <- mean(abs(pred.er))
pred.0 <- CV.lm(data.set,0)
mse.0 <- mean(pred.0$prediction.error^2)
pev <- 1-mse.min/mse.0

predictions.opt <- NULL
for(i in (1:nrow(data.set))){
  data.set.i <- data.set[-i,]
  fit.tree.i <- rpart(as.formula(paste(names(data.set)[ncol(data.set)],"~.",sep="")),
  data=data.set.i,control=list(minbucket=nodesize),cp=CP)
  prediction.i <- predict(fit.tree.i,type="vector",newdata=data.set[i,-ncol(data.set),
  drop=FALSE])
  predictions.opt <- c(predictions.opt,prediction.i)
}
```

```
pred.er <- predictions.opt - data.set[,ncol(data.set)]
mse.plot <- mean(pred.er^2)

explanation1 <- paste("n: ",nrow(data.set),"; mse: ",
formatC(mse.plot,digits=2,format="f"),"; mae: ",formatC(mae,digits=2,format="f"),
                        "\n pev: ",formatC(pev,digits=2,format="f"),sep="")
plot(data.set[,ncol(data.set)], predictions.opt, xlab="Actual compressive strength (MPa)",
     ylab="Predicted compressive strength (MPa)",
     main=explanation1); abline(a=0,b=1,col="red")

#----------- Set parameters ----------------------------

set.seed(10)

n <- 200 # Numbers of observations
p <- 8 # Numbers of features
s <- 3 # Number of relevant features

n.sim <- 50 # Number of RF simulations
sigma <- 1 # Marginal error standard deviation
bval <- 1 # Magnitude of nonzero coefficients

par(mfrow=c(1,3))

#-------------- Set measures on or off ------------------------

breiman.0 <- 1
permu.0 <- 0
lasso.0 <- 0
glm.0 <- 0
glmnet.0 <- 0
glmboost.0 <- 0
gbm.0 <- 0
nn.0 <- 0
gini.0 <- 0
check.0 <-1

#-------------- Pairwise correlation ------------------

### NO CORRELATION

rho <- 0
U <- rep(0,times=p)
# U <- c(1,2,2)
a <- 1
b <- 1.0001
var <- seq(a,b,((b-a)/(p-1)))
# var <- c(1.5,1.05,3.1)
cov <- rho*sqrt(var)*sqrt(var)
Sigma <- matrix(cov,p,p)
diag(Sigma) <- var

### EQUAL CORRELATION FOR ALL COVARIATES

# rho <- 0.5
# mu <- rep(0,times=p)
# var <- 0.5
# cov <- rho*sqrt(var)*sqrt(var)
# Sigma <- matrix(cov,p,p) # covariance matrix
# diag(Sigma) <- var

### CORRELATION AMONG SOME COVARIATES

# rho.1 <- 0.9
# rho.2 <- 0
# rho.3 <- 0
#
# U <- rep(0,p)
# U <- c(1,2,2)

```

```
# a <- 1
# b <- 1.0001
# var <- seq(a,b,((b-a)/(p-1)))
# var <- c(1.5,1.05,3.1)

# Sigma <- matrix(cov.1,p,p)
# Sigma[1:3,1:3] <- cov.2
# Sigma <- matrix(0,p,p)
# diag(Sigma) <- var
# a <- which(Sigma==var[1])
# a.place <- floor(a/p)+1
# b <- which(Sigma==var[2])
# b.place <- floor(b/p)+1
# c <- which(Sigma==var[3])
# c.place <- floor(c/p)
# if(b.place==c.place){
#   c.place <- c.place+1
# }

# Sigma[a.place,b.place] <- rho.1*sqrt(var[1])*sqrt(var[2])
# Sigma[b.place,a.place] <- rho.1*sqrt(var[1])*sqrt(var[2])
# Sigma[b.place,c.place] <- rho.2*sqrt(var[3])*sqrt(var[2])
# Sigma[c.place,b.place] <- rho.2*sqrt(var[3])*sqrt(var[2])
# Sigma[a.place,c.place] <- rho.3*sqrt(var[1])*sqrt(var[3])
# Sigma[c.place,a.place] <- rho.3*sqrt(var[1])*sqrt(var[3])

# cov.1 <- -0.5
# cov.2 <- -0.35
# cov.3 <- 0.8
# Sigma[a.place,b.place] <- cov.1
# Sigma[b.place,a.place] <- cov.1
# Sigma[b.place,c.place] <- cov.2
# Sigma[c.place,b.place] <- cov.2
# Sigma[a.place,c.place] <- cov.3
# Sigma[c.place,a.place] <- cov.3

#-------------- Generate x matrix -------------------

L <- t(chol(Sigma))
xx <- matrix(0,n,p)
for (i in 1:n){
  R <- rnorm(p)
  xx[i,] <- U + L%*%R
}
#-------------- Generate error vector -------------

e <- rnorm(n,0,0.1) # normal

#-------------- Error variance  ---------

sx <- rep(1,n) # sigma.type is constant
sigma.vec <- sigma*sx

#-------------- Mean function = linear ------------

# beta <- rep(0,p)
# beta[1:s] <- sample(c(1,2)*bval,s,replace=T)
# MU <- x %*% beta

# beta <- rep(0,p)
# beta.vector <- c(1,2,3,1,2,3)
# vector <- rep(0, times=(p-6))
# beta <- c(beta.vector,vector)
# MU <- x %*% beta

MU <- x[,1]+1.5*x[,2]+2*x[,3] +2.5*x[,4] +3*x[,5]+3.5*x[,6]
# MU <- 2.5*x[,1]+2*x[,2]+1.5*x[,3]+x[,4]

#-------------- Mean function = nonlinear ------------

```

```
# MU <- (x[,1]*exp(x[,2]))/(1+x[,3]^2)
# MU <- x[,1]*x[,2]*x[,3]
# MU <- (x[,1]+x[,2])/exp(x[,3])
# MU <- x[,1]*log(0.5+x[,2]^2+x[,3]^2)
# MU <- exp(1-x[,1])*log(1+(x[,2]*x[,3])^2)

#--------------- Construct y ----------------

y <- MU + e*sigma.vec

#-------------- Data ---------------------

data <- as.data.frame(cbind(x,y))
names(data) <- c(1:p,"Y")
train <- data[1:(nrow(data)/2),]
test <- data[(nrow(data)/2+1):nrow(data),]
X.train <- train[,1:(ncol(data)-1)]
Y.train <- train[,ncol(data)]
X.test <- test[,1:(ncol(data)-1)]
Y.test <- test[,ncol(data)]

# --------------- Tuning mtry met caret ------------------------------

control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
set.seed(7)
metric <- "RMSE"
tunegrid <- expand.grid(.mtry=c(1:p))
rf_gridsearch <- train(as.formula(paste(names(train)[ncol(train)],"~.", sep="")),
data=train, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)
print(rf_gridsearch)
plot(rf_gridsearch)
mtry.tune <- as.numeric(rf_gridsearch$bestTune)

#---------- Empty vectors and matrices -----------------

I.k <- matrix(NA,p,n.sim)
I.gini.k <- matrix(NA,p,n.sim)
I.k.matrix <- matrix(NA,p,n.sim)
I.gini.k.matrix  <- matrix(NA,p,n.sim)
I.tilde.k.matrix <- matrix(NA,p,n.sim)
I.check.k.matrix <- matrix(NA,p,n.sim)
mse.vec <- NULL
mse <- NULL
coefs.lasso.matrix <- matrix(NA,p,n.sim)
coefs.glm.boost.matrix <- matrix(NA,p,n.sim)
coefs.glm.net.matrix <- matrix(NA,p,n.sim)
coefs.gbm.matrix <- matrix(NA,p,n.sim)
coefs.nn.matrix <- matrix(NA,p,n.sim)
coefs.gbm.matrix <- matrix(NA,p,n.sim)
coefs.xgb.matrix <- matrix(NA,p,n.sim)
coefs.glm.matrix <- matrix(NA,p,n.sim)

#-------------- Algorithm for Breimans measure, permutation measure and LASSO -----------

for(i in 1:n.sim){
  predictor <- randomForest(y=Y.train,x=X.train,ntree=500,importance=TRUE,scale=FALSE)
#  predictor <- randomForest(y=Y.train,x=X.train,ntree=500,importance=TRUE,scale=FALSE,
mtry=mtry.tune)
  I.k.matrix[,i] <- predictor$importance[,1]
  I.gini.k.matrix[,i] <- predictor$importance[,2]

  Pi <- predictor$predicted
  pred.er <- Y.train-Pi
  mse.vec[i] <- mean(pred.er^2)

  I.tilde.k <- NULL
  for(j in 1:ncol(X.test)){
    X.test.j <- X.test
    X.test.j[,j] <- sample(X.test.j[,j],nrow(X.test),replace=TRUE)
    Y.pred.j <- predict(predictor,newdata=X.test.j,type="response")
```

```
335       pred.er.j <- Y.test-Y.pred.j
336       mse.j <- mean(pred.er.j^2)
337       I.tilde.k[j] <- mse.j-mse.vec[i]
338     }
339     I.tilde.k.matrix[,i] <- I.tilde.k
340
341     #---------- LASSO --------------------
342
343     if(lasso.0 == 1){
344       names(X.train) <- c(paste("X",(1:p),sep="."))
345       X.train.sparse <- sparse.model.matrix(~.,X.train)
346       glmmod <- glmnet(x=X.train.sparse, y=Y.train, alpha=1, family="gaussian")
347       cv.glmmod <- cv.glmnet(x=X.train.sparse, y=Y.train, alpha=1, family="gaussian")
348       lambda <- cv.glmmod$lambda.1se
349       coefs <- as.matrix(coef(cv.glmmod))
350       # rownames(coefs) <- c(-1:p)
351       coefs.lasso.matrix[,i] <- coefs[3:(p+2)]
352       names(X.train) <- c(1:p)
353     }
354
355     # ---------- GLM ----------------
356
357     if(glm.0 == 1){
358       glm.1 <- train(Y~.,data = train,method = "glm", trControl=trainControl(method="cv"))
359       coef.glm <- varImp(glm.1)
360       coefs.glm <- as.matrix(coef.glm$importance)
361       coefs.glm.matrix[,i] <- coefs.glm[1:p]
362     }
363
364     # ---------- GLMNET ----------------
365
366     if(glmnet.0 == 1){
367       glm.net <- train(Y~.,data = train,method = "glmnet", trControl=trainControl(method="cv"))
368       coef.glm.net <- varImp(glm.net)
369       coefs.glm.net <- as.matrix(coef.glm.net$importance)
370       coefs.glm.net.matrix[,i] <- coefs.glm.net[1:p]
371     }
372
373     # ---------- GLMBOOST ----------------
374
375     if(glmboost.0 == 1){
376       glm.boost <- train(Y~.,data = train,method = "glmboost", trControl=trainControl(method="cv"))
377       coef.glm.boost <- varImp(glm.boost)
378       coefs.glm.boost <- as.matrix(coef.glm.boost$importance)
379       coefs.glm.boost.matrix[,i] <- coefs.glm.boost[1:p]
380     }
381
382     # ---------- GBM ----------------
383
384     if(gbm.0 == 1){
385       gbm.1 <- train(Y~.,data = train,method = "gbm", trControl=trainControl(method="cv"))
386       coefs.gbm <- summary(gbm.1,plotit=FALSE)
387       coefs.gbm <- coefs.gbm[order(coefs.gbm[,1],decreasing=FALSE),]
388       coefs.gbm <- coefs.gbm[,2]
389       coefs.gbm.matrix[,i] <- coefs.gbm[1:p]
390     }
391
392     # ---------- NN ----------------
393
394     if(nn.0 == 1){
395       nn.1 <- train(Y~.,data = train,method = "nnet", trControl=trainControl(method="cv"))
396       coef.nn <- varImp(nn.1)
397       coefs.nn <- as.matrix(coef.nn$importance)
398       coefs.nn.matrix[,i] <- coefs.nn[1:p]
399     }
400
401     # --------- LOCO -------------
402
403     if(check.0 == 1){
404       I.check.k <- NULL
```

```
405       for(j in 1:ncol(X.train)){
406         predictor.j <- randomForest(y=Y.train,x=X.train[,-i],ntree=500)
407         pred.er.j <- Y.train-predictor.j$predicted
408         mse.j <- mean(pred.er.j^2)
409         I.check.k[j] <- mse.j-mse.vec[i]
410         print(j)
411       }
412       I.check.k.matrix[,i] <- I.tilde.k
413     }
414
415     print(i)
416   }
417
418   mse <- mean(mse.vec)
419
420   #------------------ Plot ----------------------
421
422   if(breiman.0 == 1){
423     if(nrow(I.k.matrix) <= 10){
424       boxplot(t(I.k.matrix),ylab="VIM value", xlab= "Covariate", main = "T-VIM")
425       legend("topright",
426              legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
427              bty = "n",
428              text.col = "black",
429              horiz = F,
430              cex=1, pt.cex = 1,
431              inset = 0.01)
432     } else{
433       boxplot(t(I.k.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate", main = "T-VIM")
434       legend("topright",
435              legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
436              bty = "n",
437              text.col = "black",
438              horiz = F,
439              cex=1, pt.cex = 1,
440              inset = 0.01)
441     }
442   }
443
444   if(permu.0 == 1){
445     if(nrow(I.tilde.k.matrix) <= 10){
446       boxplot(t(I.tilde.k.matrix),ylab="VIM value", xlab= "Covariate", main = "F-VIM")
447       legend("topright",
448              legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
449              bty = "n",
450              text.col = "black",
451              horiz = F,
452              cex=1, pt.cex = 1,
453              inset = 0.01)
454     } else{
455       boxplot(t(I.tilde.k.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
456       main = "F-VIM")
457       legend("topright",
458              legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
459              bty = "n",
460              text.col = "black",
461              horiz = F,
462              cex=1, pt.cex = 1,
463              inset = 0.01)
464     }
465   }
466
467   if(lasso.0 == 1){
468     if(nrow(coefs.lasso.matrix) <= 10){
469       boxplot(t(coefs.lasso.matrix),ylab="VIM value", xlab= "Covariate", main = "LASSO")
470       legend("topright",
471              legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
472              bty = "n",
473              text.col = "black",
474              horiz = F,
```

```
475             cex=1, pt.cex = 1,
476             inset = 0.01)
477   } else{
478     boxplot(t(coefs.lasso.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
479     main = "LASSO")
480     legend("topright",
481             legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
482             bty = "n",
483             text.col = "black",
484             horiz = F,
485             cex=1, pt.cex = 1,
486             inset = 0.01)
487   }
488 }
489
490 if(glm.0 == 1){
491   if(nrow(coefs.glm.matrix) <= 10){
492     boxplot(t(coefs.glm.matrix),ylab="VIM value", xlab= "Covariate", main = "GLM")
493     legend("topright",
494             legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
495             bty = "n",
496             text.col = "black",
497             horiz = F,
498             cex=1, pt.cex = 1,
499             inset = 0.01)
500   } else{
501     boxplot(t(coefs.glm.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
502     main = "GLM")
503     legend("topright",
504             legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
505             bty = "n",
506             text.col = "black",
507             horiz = F,
508             cex=1, pt.cex = 1,
509             inset = 0.01)
510   }
511 }
512
513 if(glmnet.0 == 1){
514   if(nrow(coefs.glm.net.matrix) <= 10){
515     boxplot(t(coefs.glm.net.matrix),ylab="VIM value", xlab= "Covariate", main = "GLMNET")
516     legend("topright",
517             legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
518             bty = "n",
519             text.col = "black",
520             horiz = F,
521             cex=1, pt.cex = 1,
522             inset = 0.01)
523   } else{
524     boxplot(t(coefs.glm.net.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
525     main = "GLMNET")
526     legend("topright",
527             legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
528             bty = "n",
529             text.col = "black",
530             horiz = F,
531             cex=1, pt.cex = 1,
532             inset = 0.01)
533   }
534 }
535
536 if(glmboost.0 == 1){
537   if(nrow(coefs.glm.boost.matrix) <= 10){
538     boxplot(t(coefs.glm.boost.matrix),ylab="VIM value", xlab= "Covariate",
539     main = "GLMBOOST")
540     legend("topright",
541             legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
542             bty = "n",
543             text.col = "black",
544             horiz = F,
```

```
            cex=1, pt.cex = 1,
            inset = 0.01)
  } else{
    boxplot(t(coefs.glm.boost.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
    main = "GLMBOOST")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  }
}

if(gbm.0 == 1){
  if(nrow(coefs.gbm.matrix) <= 10){
    boxplot(t(coefs.gbm.matrix),ylab="VIM value", xlab= "Covariate", main = "GBM")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  } else{
    boxplot(t(coefs.gbm.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
    main = "GBM")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  }
}

if(nn.0 == 1){
  if(nrow(coefs.nn.matrix) <= 10){
    boxplot(t(coefs.nn.matrix),ylab="VIM value", xlab= "Covariate", main = "NNET")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  } else{
    boxplot(t(coefs.nn.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
    main = "NNET")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  }
}

if(gini.0 ==1){
  if(nrow(I.gini.k.matrix) <= 10){
    boxplot(t(I.gini.k.matrix),ylab="VIM value", xlab= "Covariate", main = "GINI")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
```

```
            inset = 0.01)
  } else{
    boxplot(t(I.gini.k.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
    main = "GINI")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  }
}

if(check.0 ==1){
  if(nrow(I.check.k.matrix) <= 10){
    boxplot(t(I.check.k.matrix),ylab="VIM value", xlab= "Covariate", main = "LOCO")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  } else{
    boxplot(t(I.check.k.matrix[1:(s+10),]),ylab="VIM value", xlab= "Covariate",
    main = "LOCO")
    legend("topright",
            legend = c(paste("n =",(nrow(X.train))),paste("p =",p),paste("s =",s)),
            bty = "n",
            text.col = "black",
            horiz = F,
            cex=1, pt.cex = 1,
            inset = 0.01)
  }
}

print(beta)
print(round(mse,1))

#------------------ Global partial derivatives ----------------------

PD.vec.mod1.1 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod1.1[i] <- (exp(x[i,2])) / (1+x[i,3]^2)
}

PD.vec.mod1.2 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod1.2[i] <- (x[i,1]*exp(x[i,2])) / (1+x[i,3]^2)
}

PD.vec.mod1.3 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod1.3[i] <- (-2*x[i,1]*x[i,3]*exp(x[i,2])) / ((1+x[i,3]^2)^2)
}

round(mean(PD.vec.mod1.1),1)
round(mean(PD.vec.mod1.2),1)
round(mean(PD.vec.mod1.3),1)

------------------------------

PD.vec.mod2.1 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod2.1[i] <- x[i,2]*x[i,3]
}

PD.vec.mod2.2 <- NULL
for(i in 1:nrow(x)){
```

```
  PD.vec.mod2.2[i] <- x[i,1]*x[i,3]
}

PD.vec.mod2.3 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod2.3[i] <- x[i,1]*x[i,2]
}

round(mean(PD.vec.mod2.1),1)
round(mean(PD.vec.mod2.2),1)
round(mean(PD.vec.mod2.3),1)

-------------------------------

PD.vec.mod3.1 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod3.1[i] <- 1/(exp(x[i,3]))
}

PD.vec.mod3.2 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod3.2[i] <- 1/(exp(x[i,3]))
}

PD.vec.mod3.3 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod3.3[i] <- (-(x[i,1]+x[i,2])) / (exp(x[i,3]))
}

round(mean(PD.vec.mod3.1),1)
round(mean(PD.vec.mod3.2),1)
round(mean(PD.vec.mod3.3),1)

-------------------------------

PD.vec.mod4.1 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod4.1[i] <- log(0.5+x[i,2]^2+x[i,3]^2)
}

PD.vec.mod4.2 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod4.2[i] <- (2*x[i,1]*x[i,2]) / (0.5+x[i,2]^2+x[i,3]^2)
}

PD.vec.mod4.3 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod4.3[i] <- (2*x[i,1]*x[i,3]) / (0.5+x[i,2]^2+x[i,3]^2)
}

round(mean(PD.vec.mod4.1),1)
round(mean(PD.vec.mod4.2),1)
round(mean(PD.vec.mod4.3),1)

-------------------------------

PD.vec.mod5.1 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod5.1[i] <- -exp(1-x[i,1])*log(1+(x[i,2]*x[i,3])^2)
}

PD.vec.mod5.2 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod5.2[i] <- (2*x[i,2]*x[i,3]^2*exp(1-x[i,1])) / (1+(x[i,2]*x[i,3])^2)
}

PD.vec.mod5.3 <- NULL
for(i in 1:nrow(x)){
  PD.vec.mod5.3[i] <- (2*x[i,3]*x[i,2]^2*exp(1-x[i,1])) / (1+(x[i,2]*x[i,3])^2)
}
```

```
755
756  round(mean(PD.vec.mod5.1),1)
757  round(mean(PD.vec.mod5.2),1)
758  round(mean(PD.vec.mod5.3),1)
```