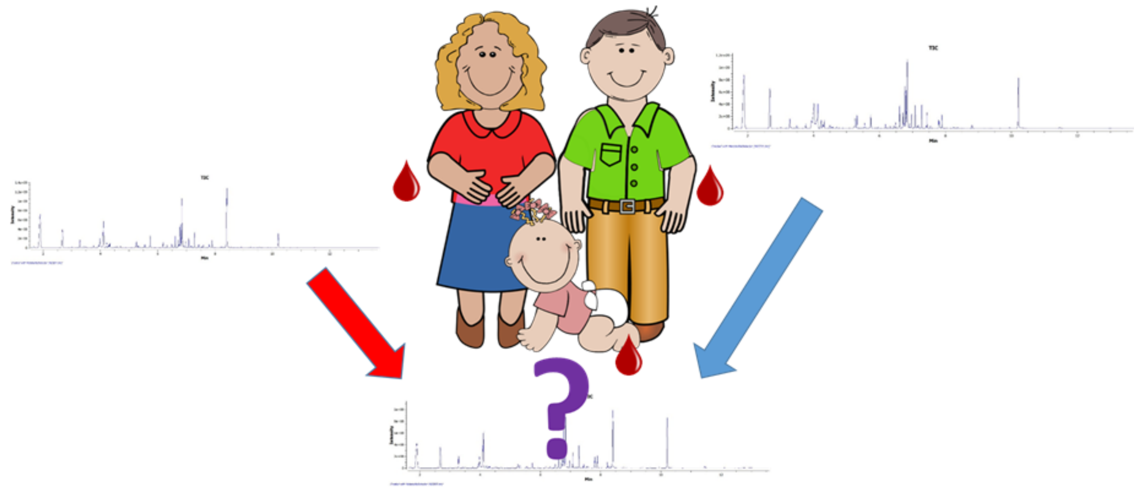
CHALMERS
UNIVERSITY OF TECHNOLOGY



Metabolic factors related to the development of allergy

A metabolomic analysis on metabolic profiles of children and their parents, venous and arterial umbilical cord blood and allergy development

Master's thesis in Biotechnology

OLLE HARTVIGSSON

Department of Biology and Biotechnology
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Metabolic factors related to the development of allergy

A metabolomic analysis on metabolic profiles of children and their parents, venous and arterial umbilical cord blood and allergy development

OLLE HARTVIGSSON

Department of Biology and Biotechnology
Division of Food and Nutrition Science
Chalmers University of Technology
Gothenburg, Sweden 2018

Metabolic factors related to the development of allergy
A metabolomic analysis on metabolic profiles of children and their parents, venous and arterial umbilical cord blood and allergy development
OLLE HARTVIGSSON

Supervisor: Malin Barman, Department of Biology and Biotechnology
Examiner and Supervisor: Alastair Ross, Department of Biology and Biotechnology

Department of Biology and Biotechnology
Division of Food and Nutrition Science
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

Metabolic factors related to the development of allergy
A metabolomic analysis on metabolic profiles of children and their parents, venous and arterial umbilical cord blood and allergy development
OLLE HARTVIGSSON
Department of Biology and Biotechnology
Chalmers University of Technology

Abstract

Allergy is an increasing problem in the western world affecting up to 25% of the population, leading to a decreased quality of life for the individuals and increased medical costs for society. Allergy is a complex disease and all the factors involved in the development of the disease are not yet established. The aim of this project was to identify differences in metabolic patterns in maternal and umbilical cord blood in relation to allergy development at one year of age. A second aim was to analyse differences in the metabolome between mothers, fathers and their newborn children and to investigate differences in metabolites in the childrens' arterial and venous umbilical cord blood. The project was performed on data from 52 families participating in the NICE-cohort using multi- and univariate statistical tools.

Using multivariate modelling it was not possible to detect global differences between allergic and non-allergic children, possibly due to only seven out of the 52 children included developing allergy by the age of one year. Univariate models which examined each metabolite separately found several metabolites, mainly amino acids and monosaccharides, differed significantly between allergic and non-allergic children.

When comparing samples from mothers, fathers and children it was found that mothers generally had higher amounts of α-tocopherol and fatty acids (mainly oleic-, linoleic- and linolenic acid) compared to paternal and infant samples, while infant samples contained high amounts of amino acids (mainly, glycylvaline, lysine, phenyalanine and tryptophan). The comparison between metabolic pattern in arterial and venous cord blood found that arterial blood contained more mono- and disaccharides (mainly deoxy galactose, glucose, sorbose and galactose) while venous blood contained more organic acids (including α-ketoglutaric acid and glutamic acid).

Due to the low number of allergic children in this ongoing, no conclusions could be drawn about a specific metabolic pattern in relation to allergy. Hence, larger studies are needed to investigate the metabolic relationship to allergy development. Differences between mothers and newborns reflects differences in their catabolic/anabolic states. Arterial and venous differences may indicate what substrates are being preferentially used by the newborn baby.

Keywords: Metabolomics, allergy, infants, family, umbilical cord, plasma, GC-MS/MS.

Acknowledgements

I would like to thank my supervisors Malin Barman and Alastair Ross for all the assistance and inputs throughout the entire project and for making this project a possibility. Carl Brunius, without you this project would not have been what it is today. Thank you so much for introducing me to R and for everything you have provided me with, not only algorithms and bug fixing, but by bringing enthusiasm and joy as well. A large thanks to Ann-Sofie Sandberg for the knowledge and the ideas that you have put into this project, your input has been greatly appreciated.

To all the researchers in the NICE-study, thank you for welcoming me into the group and for all the things I have learned from the meetings. To my office friends, thank you so much for making this year a really fun one, our companionship is something that I don't think I will ever forget.

All my friends who have listened to me when I've had problems and coming up with solutions and ideas, you guys are incredible. Our lunches together every week has been very rewarding both as I got to learn about your projects, but also for all the insane discussions we've had.

Lastly, my family, who have always supported me in doing whatever I want to do, and to not listen to anyone else in that regard. You have always been there when I've needed help through my entire education. Thank you so much for this, I would never have gotten this far without you.

Olle Hartvigsson, Gothenburg, June 2018

Contents

List of Figures

List of Tables

Glossary

bias	A systematic distortion of data dependent on treatments and sampling procedure.
bootstrapping	A way to calculate mean values less prone to outliers by random sampling multiple points in the data and averaging them together.
chorionic	Embryonic membranes in the placenta.
fitness metric	Different fitness functions that are used during model tuning.
GC-MS/MS	Gas Chromatography - tandem masspectroscopy, a sensitive analytical chemistry method that can identify compounds based on their retention time and mass fragmentation.
Immunoglobulin E	An antibody that is involved in most allergic responses.
imputation	Replacing missing values with numbers by various mathematical methods.
mast cells	A cell involved in the immune system, responsible for histamine release.
metabolite	A small molecule involved in a metabolic pathway with a molecular weight below 2000 Da.
metabolomics	The study of all metabolites in a certain sample.
overfitting	Making the models predict the outcome better than it does in reality.
Partial Least Squares	A supervised multivariate statistical method that uses training data to predict outcomes.
Principal Component Analysis	An unsupervised multivariate tool used to reduce dimensionality of data and to get an overview of it.

Random Forest	A supervised multivariate method that utilizes classification trees with imposed constraints, used for classification and regression problems.
sensitization	A process where IgE-antibodies binds to mast cells, a prerequisite for allergy development.

1
Introduction

Allergic disorders such as atopic dermatitis (eczema), atopic rhinitis (hay fever), allergic asthma and food allergies are amongst the most common diseases in the western world affecting up to 25% of the population [1]. An increasing number of people are hospitalized due to allergic reactions, affecting both the person getting hospitalized and the society due to increased medical costs [2]. It is known that heredity has an influence on the development, but environmental factors such as older siblings and pets has an impact on the risk to develop the disease as well [3]. It remains unknown why these factors would lead to an increase in allergy, and deep understanding for triggers for allergy, or why some people become allergic but not others. A recent study in mice has shown that the immune system might be stimulated already *in utero* through microbial antigens from the mother, possibly carried by antibodies, to the foetus [4]. This disputes the previous belief of the sterile womb hypothesis, which states that the intrauterine environment is sterile and that the immune system and thereby allergy development begins in the neonatal stages of life. The potential involvement of the maternal microbiome also opens up many questions and possible solutions for reduction of allergy risk in the future. Studying the flow of metabolites from the mothers to their children in relation to allergy could give an inkling to both if bacterial metabolites are transported to the foetus, but also how other molecules could potentially affect allergy development.

A metabolite is usually defined as an organic molecule involved in a metabolic pathway with a molecular weight of less than 2000 Da [5]. Metabolite concentrations in humans are a result of an interaction between genetic and environmental factors as well as food intake and medication [6] [7]. Metabolomics is a relatively new area of science which aims to identify and quantify the metabolites in a given sample. This is usually done using analytical techniques such as gas chromatography (GC), liquid chromatography (LC), nuclear magnetic resonance (NMR) and mass spectrometry (MS) in combination with extensive data analysis. The kind of sample used for a metabolomic analysis can vary, it can be everything from single cell organisms to blood, urine, faeces or tissue samples.

This project aims to identify key metabolites and metabolite patterns in newborn infants involved in allergy development at one year of age. As a secondary aim, metabolic differences between mothers, fathers and newborn infants along with differences between venous and arterial umbilical cord blood plasma will also be determined. The samples used in this thesis work are from families in the Nutritional impact on Immunological maturation during Childhood in relation to the Environment (NICE) study.

2 Theory

2.1 Umbilical cord blood

To transport nutrients to the foetus, the mothers' body delivers blood to the placenta. The parts of the umbilical cord that are in contact with the placenta are dispersed as chorionic villi [8]. The maternal blood lies in a pool around these villi (Figure 2.1). Molecules are then transported to the bloodstream of the foetus, both through passive and active transport [9]. The blood moving from the mother to the baby is transported via the umbilical cord vein while the blood that goes from the baby back to the mother is transported through the umbilical artery. Both polyunsaturated fatty acids (PUFAs) and most amino acids appears to be transported over the placenta through active transport [9] [10]. There seems to be a priority in the transportation of PUFAs, where docosahexaenoic acid (DHA) takes precedence followed by arachidonic acid, α-linolenic acid and linoleic acid [11]. Oleic acid however, appears to be transported through passive diffusion. The anionic amino acids glutamate and aspartate are not actively transported across the placenta, it is suggested that glutamate is instead converted from glutamine in the foetal liver [10]. As of now, there is little known about the metabolic relationship between arterial and venous umbilical cord blood. A study conducted in 2016 by Koh et al. studied differences in catecholamines, glucose, lactate and blood gases [12] in 57 children delivered via elective cesarean section. They found that glucose levels were higher in the venous cord blood while lactate concentrations were higher in the arterial cord blood.

2.2 Metabolic differences between parents and infant

There has been several studies that has looked into metabolite relationships between mothers and the newborn infant [14] [15] [16]. Most of these studies looked into specific metabolites that were relevant to their research question, but few analysed the whole metabolome. Few if any of these studies included the father in the investigation. Studies has shown that mothers has higher concentrations of linoleic acid, α-linolenic acid [17] and bisphenol A [18]-, while docosapentaenoic acid (DPA), DHA [17] and Vitamin D [19] concentrations were higher in cord blood. A study conducted in 1967 showed that all amino acids were higher in concentrations in venous cord blood compared to their respective mothers (almost every one statistically

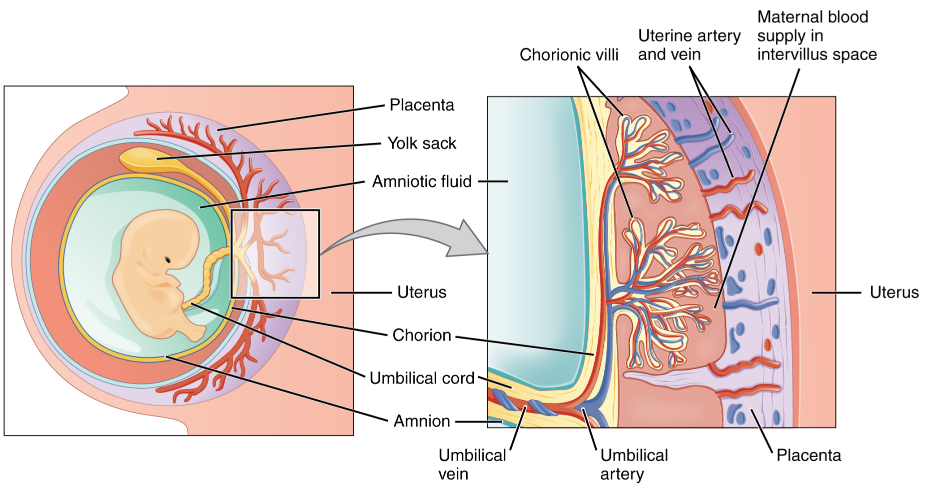


Figure 2.1: How blood is transported from the placenta to the umbilical cord. [13]

significant) [20].

2.3 Mechanisms in allergic diseases

Sensitization is a necessary prerequisite for an allergic reaction. Sensitization occurs in some individuals when an antigen (usually a protein) is introduced to the body [21]. Dendritic cells envelop the antigen and present it to Th0 cells which in turn stimulate the production of Th2 cells in atopic individuals. This process prompts B cells to start producing antigen specific Immunoglobulin E (IgE) antibodies. The IgE antibodies, although shortlived by themselves can bind to mast cells which are present in tissue and remain there for months. Why most people develop a tolerance to these antigens and others go through the sensitization process is not yet known [22].

When a sensitized individual is exposed to the antigen again, the IgE antibody binds to an epitope on the allergen which starts a chain reaction leading to vesicles in the mast cells releasing histamine and other pro-inflammatory compounds [21]. These mediate an increased permeability and attract cells and blood plasma into the tissue, ending in an inflammation at the site [21] [22]. Symptoms from an allergic reaction are commonly eczema, wheezing, hay fever and red and itching eyes.

The hygiene hypothesis states that the increased focus on cleanliness and avoidance of dirt, often with the goal of avoiding disease, is the main reason as to why western countries have seen an increase in allergy development, and is one of the most accepted theories today [23], and that early exposure to dirt may be protective due to providing stimulus for the immune system, which otherwise may get overactive. There are several indications that early exposure to microorganisms has a protective effect on allergy development [24] and several studies have shown that gut microbiota also plays a critical role in the development of allergic diseases [25] [26] [27]. Until recently it was assumed that the womb was a sterile environment and therefore unlikely to play a role in immune development - with immune protection in newborns being conferred by the immunoglobulin rich breast milk produced by mothers during the first few days post birth. Recent studies have questioned the sterile womb hypothesis by finding evidence of microbes both in the placenta and meconium [28] [29], meaning that bacterial colonization and thereby protection against allergy could be initiated already *in utero*.

2.4 Algorithms

2.4.1 Principal Component Analysis

The Principal Component Analysis (PCA) is an unsupervised method, meaning that no information about different groups is included in the analysis [30]. It works by reducing the dimensionality of the data by using covariances between the different variables to express them in fewer dimension while still retaining as much of the variance as possible [30]. In a PCA scores plot, the first component of a PCA attempts to explain as much of the variance as possible, while the following ones explains as much of the variance that previous components has not yet explained while being orthogonal to all the previous components.

When interpreting a PCA, scores (representing observations) and loadings (representing variables) are usually observed [31]. The scores show where the observations end up in the remodelled space which is made up from the components. The loadings show how the variables affect the observations. Scores and loadings which occupy the same area are usually closely correlated with each other. Conversely, scores and loadings which are on opposite sides are usually inversely correlated. A PCA offers no statistical value, but it is a useful tool to get an overview of the data, as well as for outlier identification.

2.4.2 Partial Least Squares regression and discriminant analysis

Partial Least Squares (PLS) regression is a combination of principal component regression (regression based on PCA) and ordinary least squares (OLS) regression [32]. The principal component regression aims to maximize the variance of the given data set X, while OLS aims to correlate a linear combination of X to the response variable Y. By using a combination of these two, a good prediction could be made that takes the structure of the data into account more than OLS.

Partial least squares discriminant analysis (PLS-DA) is a modification of the PLS, which is often used for classification problems (when the response variable is in form of groups), unlike the PLS regression, which is more suitable for a continuous response vector [33]. Similarly to PCA, scores and loadings are given as an easy way to interpret the results from the analysis [32]. However, these are not computed in the same way as in a PCA. The PLS-DA aims to classify the given data, by utilizing a training set to build a mathematical model. The model is then able to take new data and classify it based on the values on the input variables.

2.4.3 Random Forest

Random Forest (RF) is an ensemble algorithm frequently used for both classification and regression analysis [34]. It is a tweak of a method called bootstrap aggregation ensemble which is based on bootstrapping, which in turn is in essence a way to estimate a mean value. Bootstrapping works by inflating the sample size by randomly sampling a value, with redraw, from the data set many times, dividing them into subsamples and then take the mean of each subsample and averaging them together. The result from this procedure is a mean value that is less prone to bias from errors in the measurement. Bootstrap aggregation is a combination of a high variance decision tree classifier and bootstrapping where many random subsamples of the data are each subjected to the decision tree algorithm and then a majority vote (for classification) is cast or an average (for regression) is done from all the calculations ending in a result with less variance. The bootstrap aggregation reduces the variance significantly, but depending on the structure of the data, this may not be enough. Random forest imposes a condition to the bootstrap aggregation procedure which makes the different subsamples less correlated and thus further reducing the variance for the final result.

2.4.4 ANOVA decomposition

ANOVA decomposition is a way to remove variance in the data obtained from other sources other than that of the study question [35]. By analysing the variance for a factor, two matrices are given as a response, one which contains the variance for the factor, and one residual containing information not described by the variance of the factor (equation 2.1). By using the residual matrix, the variance from the selected factor will no longer affect the analysis. This procedure can be done in several steps (equation 2.2), using the residual matrix from the preceding decomposition to remove the influence of factors unrelated to the analysis.

$$X_{initial} \xrightarrow{\text{ANOVA1}} X_{variance} + X_{residual1} \tag{2.1}$$

$$X_{residual1} \xrightarrow{\text{ANOVA2}} X_{variance2} + X_{residual2} \tag{2.2}$$

2.4.5 Multilevel analysis

A multilevel analysis is the multivariate version of a paired t-test. It was originally designed to be used in cross-over intervention studies [36], but can also be applied to cases where samples are dependent and only small differences are expected. The analysis is conducted by subtracting the values from the "before treatment" with the values corresponding to the same person from the "after treatment" (equation 2.3) creating a new matrix with the intra-person difference [37]. Another matrix is conducted by multiplying the first matrix with -1, effectively creating a matrix with "after treatment" minus "before treatment" and then a final matrix is created by concatenating both matrices into one (equation 2.4). A Y vector is created by assigning the first half of the final matrix a 1, and the second half -1 which thereby enables classification algorithms to separate the two groups.

$$X_{before\,treatment} - X_{after\,treatment} = X_{difference} \tag{2.3}$$

$$X_{final} = \begin{bmatrix} X_{difference} \\ -X_{difference} \end{bmatrix} \tag{2.4}$$

2.4.6 The MUVR package

The MUVR package is an R package developed by Carl Brunius (Division of Food and Nutrition Science, Chalmers; freely available at `https://gitlab.com/CarlBrunius/MUVR`) which features repeated double cross-validation [38] for both PLS and RF algorithms, along with an unbiased variable selection. The package offers four different fitness metrics (which parameters the model should try to optimize), Root Mean Squared Error of Prediction (RMSEP), Area Under the Receiver Operating Characteristic (AUROC), number of misclassifications (MISS) and Balanced Error Rate (BER). MUVR also supports multilevel analysis and comes with an option to include ID. The ID can be used so that samples from the same individual, or samples that are suspected to be similar (such as family relations) will not be used to model each other, but will be taken out of the training model simultaneously. The variable selection is based on Variable Importance in Projection (VIP) values, which is calculated by how much the metabolite helps in the prediction averaged over the user defined amount of repetitions. Another feature of the algorithm is that each calculation gives three different models, referred to as 'min', 'mid', and 'max'. The max model is representative of the optimal model using the given fitness metric, while the min model is decided by being within 5% of the optimal setting, with as few variables left in the model as possible with the imposed constraint. The mid model is then calculated by the geometric mean of the first two.

3
Methods

3.1 Sample treatment and analysis

Families with planned births between March 2015 and February 2018 at Sunderby hospital in the northern part of Sweden were invited in pregnancy weeks 18-19 to participate in the NICE-study. From the approximately 600 families that accepted, 52 families were selected for metabolomics analysis based on availability of a complete set of blood samples taken at delivery (samples from the mother, father and umbilical cord venous, arterial and mixed blood). All samples were usually centrifuged, separated and aliquoted within 12 hours. However, due to weekends and holidays, some samples were stored in a -4 °C freezer for 12-65 hours. At a later stage, the samples were transported on dry ice to Chalmers University of Technology in Gothenburg and stored at -80 °C before analysis by gas chromatography-tandem mass spectrometry (GC-MS/MS). Before analysis, samples were randomised in order to minimise the effect of analytical batch and the possibility of bias due to the date of sample collection. Family groups (mother, father and baby) were randomised across four analytical batches, and the run order of samples from each family group (blood plasma from mother and father, and umbilical cord arterial, venous and mixed blood plasma) were randomised within each family group. In addition, a quality control sample (made by pooling aliquotes from all plasma samples) was injected between every 10 samples.

GC-MS/MS metabolomics was carried out using a Shimadzu TQ8030 GC-MS/MS (Shimadzu Europa GmbH, Duisberg, Germany) using a combined scanning multiple reaction monitoring (MRM) method [39]. This method performs both mass spectral scanning of all compounds leaving the GC column, detecting the fragmentation of each compound, as wel as MRM. MRM is when the mass detector selects one mass fragment ion and then further fragments it to detect a second fragment, usually resulting in detection that is highly sensitive due to low background interference. Scanning and MRM detection of up to four compounds at a time is performed in a loop taking approximately 200 ms, allowing at least 10 detection points per peak. GC-MS/MS data acquisition was performed prior to this thesis work.

After the GC-MS/MS analysis was performed, metabolite identification was done using the software Metabolite Detector [40] which deconvolutes all GC-MS peaks based on correlations between different masses and expected peak shape and tentatively identifies compounds based on retention index and matching against a mass spectral library. The targeted data was processed using a script developed by Jonsson et al. [41] which deconvolutes specific peaks in a library developed by the Swedish Metabolomics Centre in Umeå, based on retention index, mass spectral matching and peak shape of fragment ions specific for individual compounds. Both methods give relative metabolite concentrations between each sample for each metabolite. Further, MRM data were integrated using GCMS Solutions software (Shimadzu Europa GmbH).

3.2 Allergy diagnosis

Of the 52 families, five did not have any allergy record (did not show up to the diagnosis occasion), one was removed due to incomplete data, which left 46 families. Of these 46, four were diagnosed with asthma, one with eczema, one with asthma and eczema and one with food allergy for a total of 7 allergic children. The diagnoses were conducted by Dr. Anna Sandin at Sunderbyn hospital in Norrbotten County. Food allergy was diagnosed by provocation at the hospital. Atopic dermatitis was diagnosed according to criteria developed by Williams et al. 1994 [42] [43] [44]. Asthma was diagnosed by the children having at least one of the following symptoms: wheeze between infections, persistent wheeze for at least 4 weeks, wheeze during infection simultaneously with another allergic disease or three wheeze periods in connection to infectious diseases.

3.3 Analysis in SIMCA

At the start of this project, SIMCA was used for multivariate statistical analysis. Although SIMCA has a wide range of suitable tools, its use was discontinued due to insufficient validation mechanisms, as well as a lack of an unbiased variable selection approach.

As mass spectrometes have relatively high inter-sample variability due to subtle differences in sample preparation, lab conditions and the condition of the instrument, a proportion of the variation in the data will be due to analytical variation rather than biological variation. Correction for inter- and intra-batch variation is necessary. This was done through normalization vectors obtained using SIMCA (V 14.1)[45] by making a PCA on the internal standards, obtaining the eigenvalues for the first component (resulting in normalisation vectors for each injected sample) and then dividing all samples by the normalization vectors.
In SIMCA, PCA, Orthogonal-PLS (OPLS) and OPLS-DA was used to check for differences in metabolome between family members as well as trying to differentiate allergic from non allergic children. The OPLS and OPLS-DA was performed, and stepwise elimination of insignificant metabolites were removed (decided by VIP confidence interval range being both higher and lower than 0). It was later realized that this would cause overfitting, and SIMCA was replaced by R along with suitable

packages mentioned in section 3.5. Only those results obtained from the R routines will be presented in this thesis.

3.4 Second pre-treatment

To be able to move from SIMCA to R, some additional pre-treatment had to be done. All the chromatograms from the targeted analysis was visually examined, and those deemed unfit were removed from the data. Two examples of chromatograms where one was kept and one removed from the analysis is shown in Figure 3.1. Metabolites used for internal standards were removed from the dataset, along with EDTA, as this was used as an anti-coagulant in some, but not all tubes. As several metabolites were found in two or more of the untargeted, targeted and MRM scans, the replicate metabolites were removed from the less sensitive/robust analysis (MRM was considered most sensitive, followed by targeted while untargeted was considered to be least sensitive). The data cleanup was done at this point due to variables being easier to remove post analysis in SIMCA, while in R it was deemed faster to do before analysis.

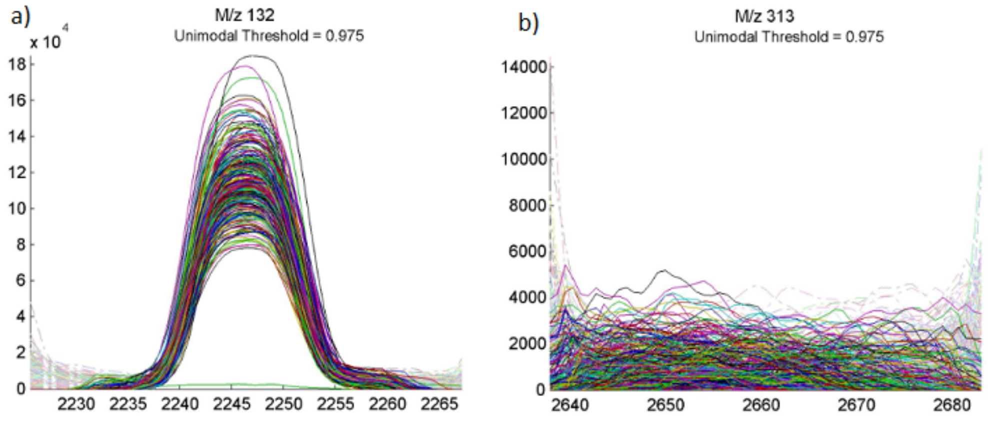


Figure 3.1: Shows two example chromatograms. Chromatogram A corresponds to octadecanoic acid-TMS, and was kept in the analysis. Chromatogram B corresponds to Cytidine-4TMS, which was not kept due to its appearance being questionable to actually represent an accurate measurement.

The data was then loaded into R [46] and as 0.137% of the data points were missing an imputation was made to get an estimation of the missing values, using the missForest algorithm [47] [48]. Unlike SIMCA, the algorithms used in R are not able to cope with missing values, and an imputation was deemed the most suitable way of dealing with these as the amount of missing points was relatively small. Following the imputation, a PCA was made to check for any outliers and obvious trends. One outlier was found, it was removed and a new PCA model was determined.

Before starting with the allergy analysis, any families which lacked allergy data (subjects not attending follow up allergy diagnosis visits) were removed from the data set. These families were included in the family metabolomics investigations.

3.5 Analysis in R

After the clean-up steps analysis began by comparing the samples from the mothers, fathers and the three different kinds of umbilical cord samples to find out what differed between them. The five samples were then compared against each other using rdCV-PLS-DA and rdCV-RF algorithms with fitness metric set to MISS. 100 permutation tests for each analysis where the dependent variable (mother, father, venous, arterial and mixed) were randomized before putting them through the same algorithms as before. The permutation tests were done to ensure that the results are not due to overfitting. Welch's t-tests were determined for all calculations comparing the fitness metric values from the real calculation to the fitness metric values of the permutations. If the null hypothesis (i.e. the fitness metric values are the same) was rejected using $\alpha = 0.05$ the test was considered significant.

There was no clear separation between arterial and venous umbilical cord samples, so a multilevel analysis was performed. An effect matrix was constructed according to equations 2.3 and 2.4. Where X is a matrix consisting of all relative metabolite concentrations for each child. The effect matrix was then run through rdCV-PLS-DA and rdCV-RF algorithms, with fitness metric set to MISS, along with permutation tests where the class of the observations in the effect matrix was randomized to 1 or -1.

As all participating families had five different kinds of samples, an ANOVA decomposition was made on the data before running it through both the rdCV-PLS and rdCV-RF algorithms again. This was done to see if the differences between the group were mainly due differences between groups or if the changes would persist when checking between the same family members. This makes it possible to check for intra-family differences, by removing variance caused by differences between families.

After the families without any allergy data were removed from the set, PLS-DA and RF was performed on each sample type individually. As the outlier removed from the PCA was a father which had an allergic child, only six samples were part of the allergy group when the fathers' metabolome was included in the comparison of allergic and non-allergic children. These analyses were conducted with balanced error rate as fitness. Balanced error rate is a fitness metric that is defined in equation 3.1.

$$BER = \frac{1}{2} * \left(\frac{Wrong\ predictions\ in\ group\ 1}{Total\ predictions\ in\ group\ 1} + \frac{Wrong\ predictions\ ingroup\ 2}{Total\ predictions\ in\ group\ 2} \right) \tag{3.1}$$

This is necessary as the groups were skewed in their distribution (7 allergic and 39 healthy). When using misclassifications as fitness, all models reported 7 misclassification, where none was considered to be allergic.

All code used to produce the results can be found in Appendix B.

4
Results

Figure 4.1a shows the first two components (out of five) of the PCA that was done on the data. An outlier can be observed at the bottom left corner of the plot, by having scores in the first component almost 5 times lower (-150 compared to -30) compared to the second lowest sample. This outlier was removed and a second PCA was made (Figure 4.1b). In both Figure 4.1a and Figure 4.1b the points are plotted for different family members (squares representing mothers, circles representing fathers and triangles representing any sample from the umbilical cord).

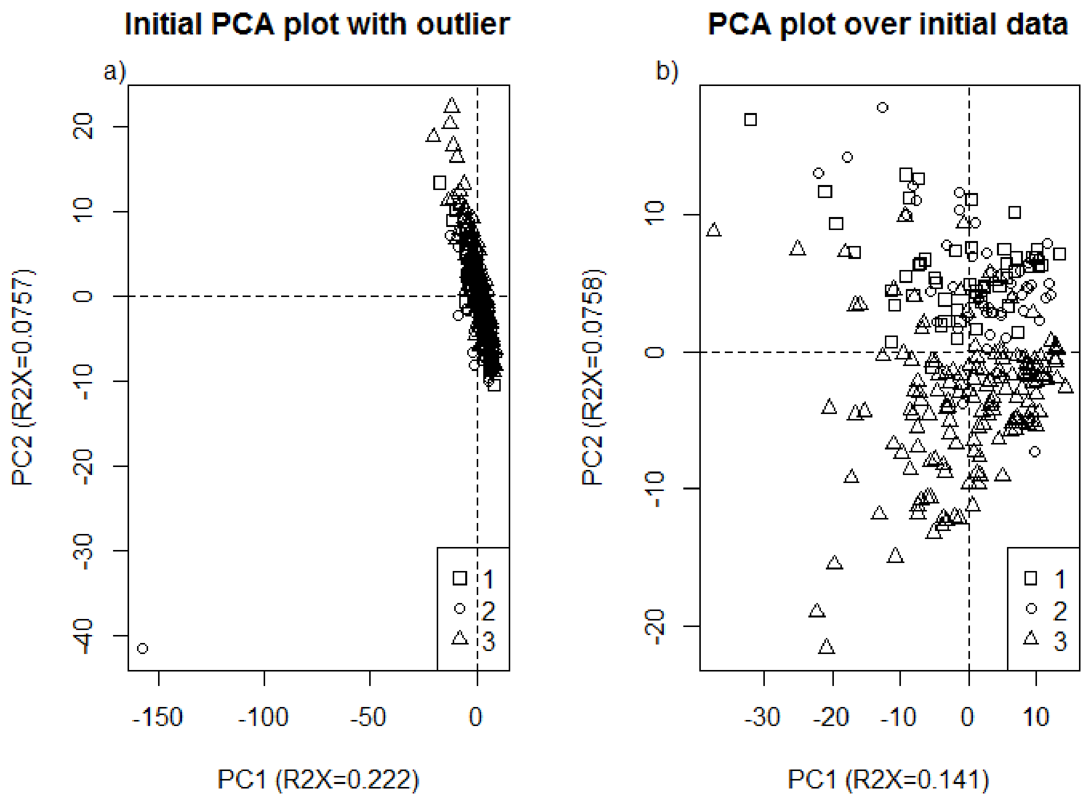


Figure 4.1: Figure 4.1a shows the first two components of the initial PCA plot before outlier removal. Figure 4.1b shows the first two components of the PCA after the outlier has been removed. Mothers are represented as squares, fathers as circles and samples from the umbilical cord are represented as triangles.

There appears to be a trend in the second component in Figure 4.1b where the umbilical cord samples have lower scores than that of the mothers and fathers. This is an indication that there are some significant metabolic differences between parents and the children. However, as mentioned in section 2.4.1 a PCA is not a statistical method, and no conclusions should be drawn from this.

The same PCA was also labelled for the four different batches. No pattern could be observed relating to batch, which is a good indication that the batch correction procedure was successful.

4.1 Difference between mothers fathers and newborn infants

4.1.1 Mothers compared to umbilical cord samples

The comparison between mothers and umbilical cord samples gives an indication on the relative flow of metabolites between mother and baby at birth. Table 4.1 includes the number of misclassifications, AUROC values, p-values (obtained with Welch's t-tests) and number of metabolites for each model for the samples from the mothers compared to the different umbilical cord samples. The number of misclassifications and AUROC values are indications of how strong a multivariate model is. AUROC is a value between 0 and 1 and the closer to one a model is, the more accurate. The p-values are obtained from number of misclassifications from the actual model, when compared to the number of misclassifications from the permutations tests. The number of metabolites in the min and max model represents how many metabolites are needed to properly model the data and how many metabolites have an influence on the model respectively.

Table 4.1: Shows the number of misclassifications, AUROC values, p-values and number of metabolites for each model in the comparisons between mothers and umbilical cord samples.

Mothers compared to Arterial umbilical cord samples						
Model:	PLS min	PLS mid	PLS max	RF min	RF mid	RF max
Misclassifications	1	1	1	1	1	1
AUROC	1.0	1.0	1.0	1.0	1.0	1.0
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of Metabolites	3	42	522	8	47	273
Mothers compared to Venous umbilical cord samples						
Misclassifications	2	1	0	1	1	0
AUROC	1.0	1.0	1.0	1.0	1.0	1.0
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of Metabolites	5	48	517	23	104	473
Mothers compared to Mixed umbilical cord samples						
Misclassifications	1	1	1	0	0	0
AUROC	1.0	1.0	1.0	1.0	1.0	1.0
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of Metabolites	11	74	505	2	33	543

In Figure 4.2 boxplots of six of the most influential metabolites in the models from the comparison between mothers and arterial umbilical cord samples are shown. Similar metabolites and concentrations were shown to separate mixed and venous samples from the mother as well. The amino acids glycylvaline, lysine and phenylalanine are significantly higher in the arterial umbilical cord samples than of that in the mothers. Fructose 1,6 disphosphate, octadecadienoic acid and α-tocopherol are higher in the mother. Further metabolites with high influence on the venous and arterial comparisons were abscisic acid, octadecatrienoic acid and octadecenoic acid which were higher in mothers and tryptophan, which were higher in the children. For the comparison between mixed umbilical cord and mothers the concentrations of ethanolamine, 1,5 diaminopentane and ornithine were higher in the mixed samples and had a high influence on the separation between the two groups. Boxplots for analyses comparing samples from the mothers to venous and mixed umbilical cord samples can be seen in Figures A.1 and A.2 in Appendix A.

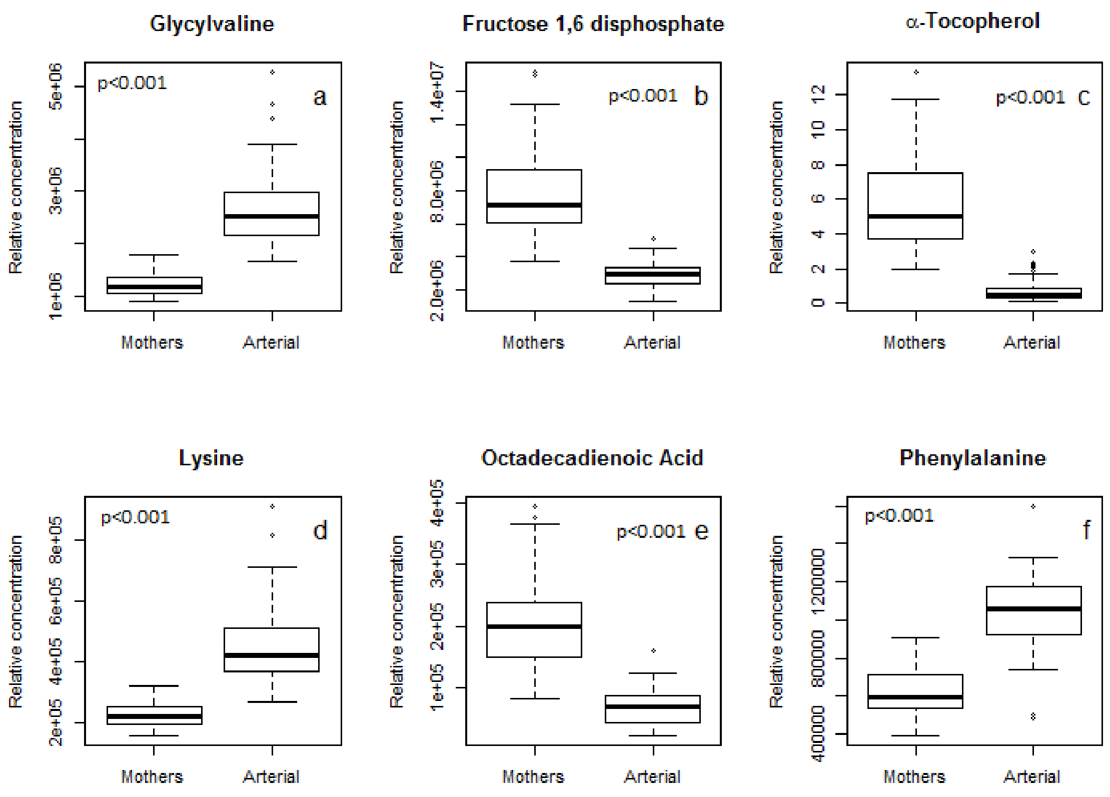


Figure 4.2: Boxplots of six of the most influential metabolites in all models for the comparison between mothers and arterial umbilical cord blood.

4.1.2 Fathers compared to umbilical cord samples

Table 4.2 shows the number of misclassifications, AUROC values, p-values from the permutation analysis and number of metabolites for each model when comparing samples from the fathers to the different kinds of umbilical cord samples.

Table 4.2: Shows the number of misclassifications, AUROC values, p-values and number of metabolites for each model in the comparisons between fathers and umbilical cord samples.

	Fathers compared to Arterial umbilical cord samples					
Model:	PLS min	PLS mid	PLS max	RF min	RF mid	RF max
Misclassifications	1	1	1	2	1	1
AUROC	1.0	0.99	0.99	1.0	1.0	1.0
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of Metabolites	4	12	37	3	8	25
	Fathers compared to Venous umbilical cord samples					
Misclassifications	3	2	2	2	2	1
AUROC	1.0	0.98	0.98	1.0	1.0	1.0
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of Metabolites	7	29	130	3	11	36
	Fathers compared to Mixed umbilical cord samples					
Misclassifications	3	2	1	1	1	1
AUROC	0.99	0.99	0.99	1.0	1.0	1.0
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of Metabolites	6	37	241	2	7	21

In Figure 4.3 boxplots of six of the most influential metabolites in the models from the comparison between fathers and venous umbilical cord samples are shown. Similar metabolites were shown to separate mixed and arterial samples from the samples from the fathers. Phosphoric acid, phosphate fragment, pyruvic acid. Nicotinic acid and malic acid were all higher in concentration in the venous samples in comparison to the samples from the fathers. Further metabolites with high influence on the arterial comparisons were melatonin, tryptophan, N-acetyl glucoseamine γ-glutamylphenylalanine. In the venous samples, allothreonine, myo-inosito and scyllo-inositol had higher concentrations while for the mixed, the concentrations of N-acetyl mannoseamine and aconitic acid were higher compared to the fathers. Boxplots for analyses comparing samples from the fathers to venous and mixed umbilical cord samples can be seen in Figures A.3 and A.4 in Appendix A.

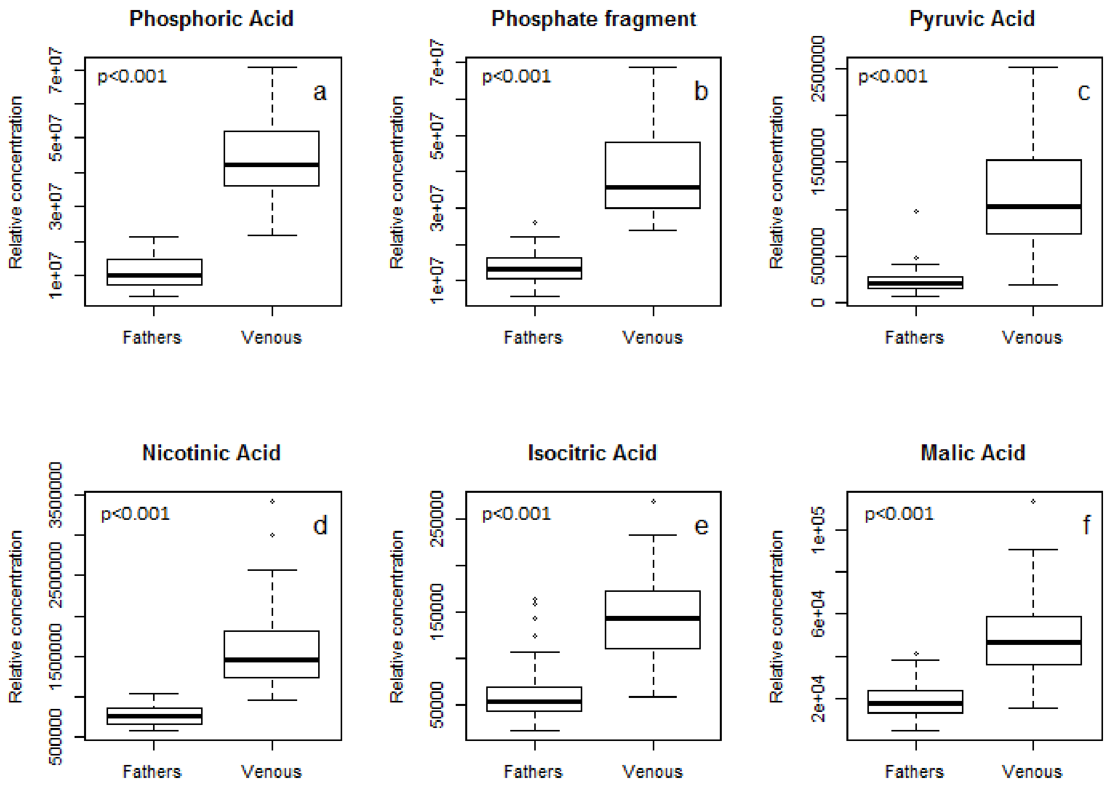


Figure 4.3: Boxplots of six of the most influential metabolites in all models for the comparison between fathers and venous umbilical cord blood.

4.1.3 Mothers compared to Fathers

In Table 4.3 the number of misclassifications, AUROC values, p-values and number of metabolites for each of the models in the comparison between mothers and fathers are shown.

Figure 4.4 shows boxplots of six of the most influential metabolites in the comparison between mothers and fathers. Higher concentrations of all of the most influential metabolites in these analyses were found in the mothers. In addition to the metabolites seen in Figure 4.4, glyceric acid 2-phosphate, melatonin, fructose 1,6-disphosphate, aconitic acid , 5-hydroxy tryptophan and citric acid also had a high impact on the models.

Table 4.3: Shows the number of misclassifications, AUROC values along with p-values from permutation tests and number of metabolites for each of the models in the analysis comparing samples from the mothers to samples from the fathers.

Model:	PLS min	PLS mid	PLS max	RF min	RF mid	RF max
Misclassifications	2	2	2	3	3	3
AUROC	0.991	0.998	0.992	0.996	0.997	0.998
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of metabolites	12	62	330	99	222	497

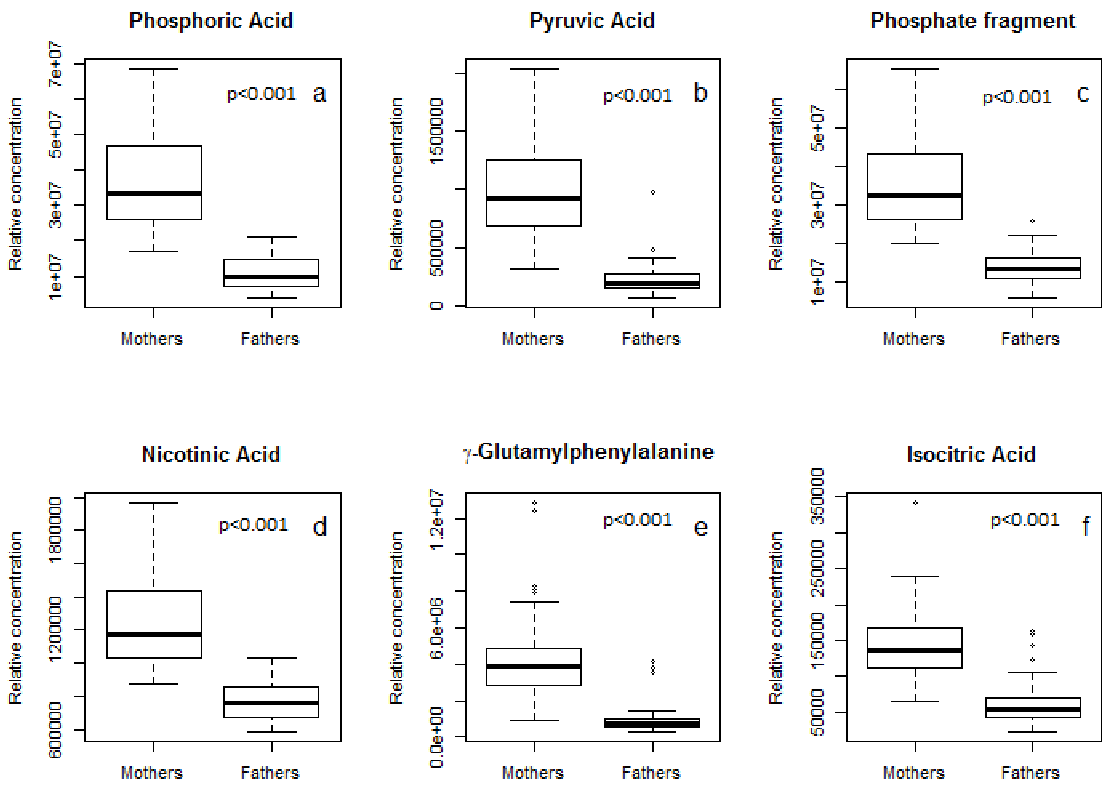


Figure 4.4: Boxplots of six of the most influential metabolites in all models for the comparison between mothers and fathers.

4.1.4 Newborn venous blood plasma compared to Newborn arterial blood plasma

Table 4.4 shows the number of misclassifications, AUROC values, p-values and number of metabolites in each of the models for the multilevel analyses of arterial and venous arterial umbilical cord blood.

Table 4.4: Shows the number of misclassifications along with p-values from permutation tests and number of metabolites for each of the models in the multilevel analysis comparing arterial and venous umbilical cord samples.

Model:	PLS min	PLS mid	PLS max	RF min	RF mid	RF max
Misclassifications	30	28	28	26	27	26
AUROC	0.801	0.797	0.770	0.818	0.822	0.820
p-value	0.013	0.0068	0.0060	0.0079	0.011	0.0078
Number of metabolites	54	74	101	12	12	15

Figure 4.5 shows six of the most influential metabolites in the multilevel analysis comparing venous and arterial umbilical cord samples. In addition to the metabolites shown in the boxplots, deoxygalactose, mannose, deoxyglucose, mannitol, idose, hypoxanthine, keto-deoxygluconate and an unknown metabolite with index "M000000 A181005-101.xxx NA 1792 PRED VAR5 ALK NA" were all found to have higher concentrations in the arterial umbilical cord blood and a significant impact on the model. Homocysteine was also found to affect the separation and its concentration was found to be higher in the venous cord blood.

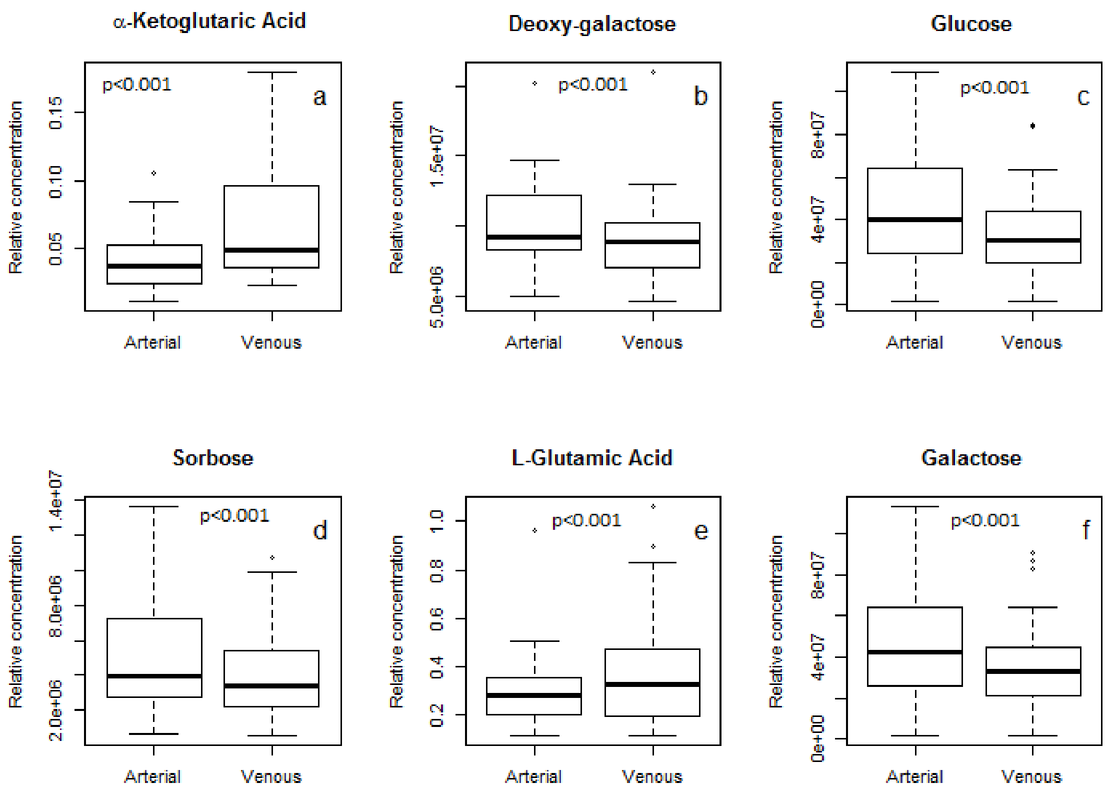


Figure 4.5: Boxplots of six of the most influential metabolites in all models for the comparison between arterial and venous umbilical cord samples. p-values are obtained from paired t-tests.

4.1.5 ANOVA decomposition on all samples

Table 4.5 shows number of misclassifications together with p-values and number of metabolites for each of the models done on the ANOVA decomposed data. Figure 4.6 shows the predictions for the mid models of both random forest and PLS-DA calculations. The plots show that neither model can separate arterial and venous umbilical cord samples.

Table 4.5: Shows the number of misclassifications values along with p-values and number of metabolites for each of the models in the analysis of the ANOVA decomposed data comparing all sample types.

Model:	PLS min	PLS mid	PLS max	RF min	RF mid	RF max
Misclassifications	35	36	37	32	32	31
p-value	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$	$<10^{-5}$
Number of metabolites	32	63	123	32	91	261

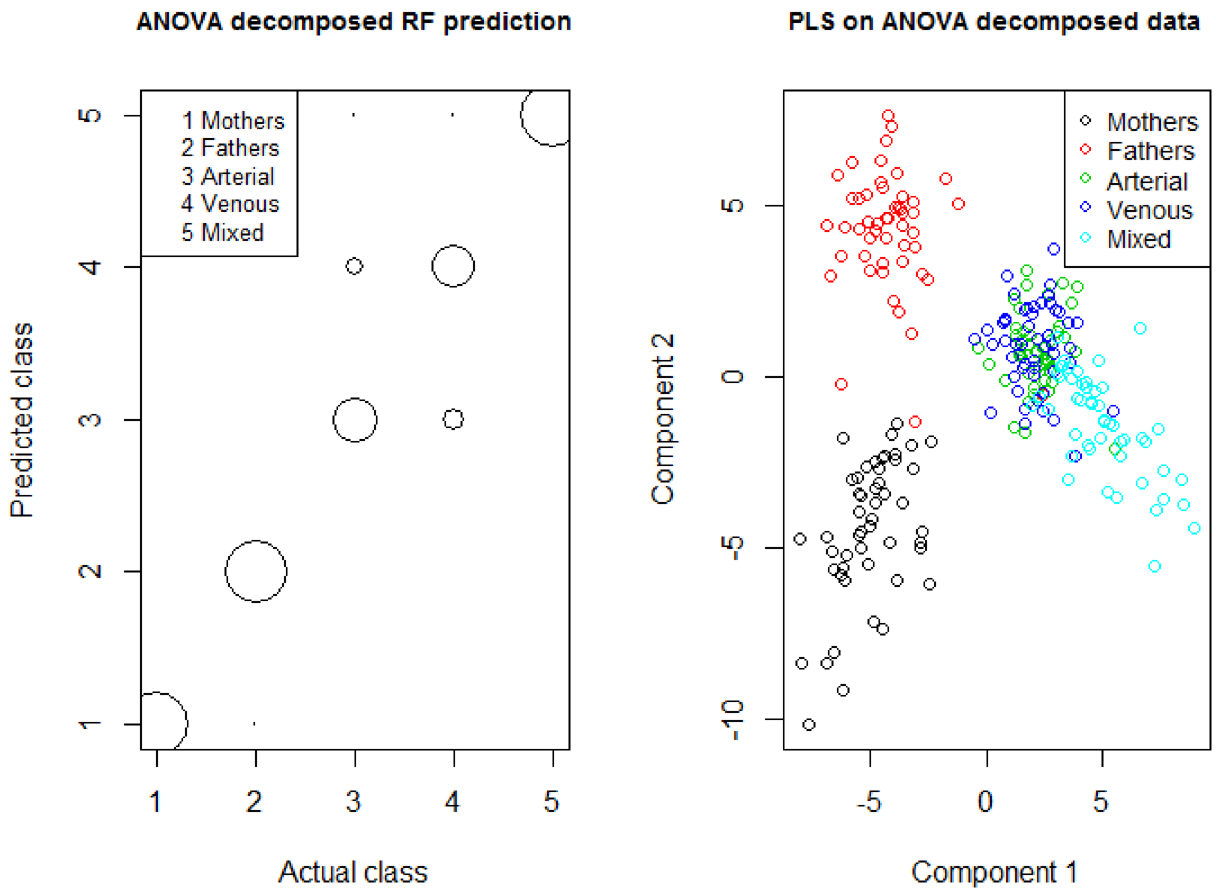


Figure 4.6: Shows RF predictions (a) and PLS predictions (b) for comparison between all samples using ANOVA decomposed data.

Figure 4.7 shows four of the most influential metabolites from the ANOVA decomposed models. In addition to these four metabolites, phosphoric acid, urea, aspartic acid, oxalic acid, 1,5-anhydro D-glucitol, α-tocopherol, N-acetyl ornithine, glycylvaline and an unknown metabolite named Unknown.sst.cgl.101 also had a high impact on the separation of the classes.

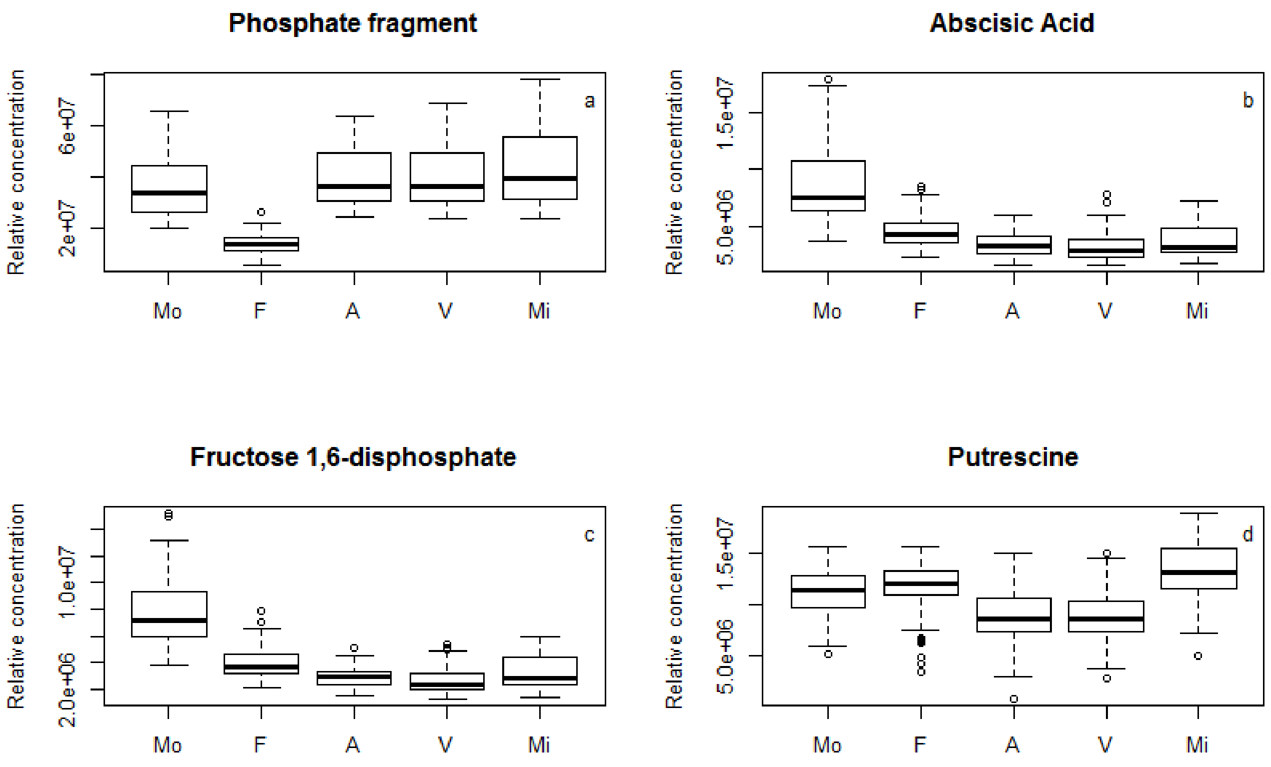


Figure 4.7: Boxplots of four of the most influential metabolites in all models for the comparison between all samples using ANOVA decomposed data.

4.2 Allergy

Neither of the models tested for allergy yielded any significance from the permutations tests (lowest p-value = 0.2). As all the different models were inadequate, no predictions of allergy development can be made from the multivariate models.

4.2.1 Mothers - Allergy

Multivariate models are based on the overall patterns in the data, and the lack of a significant model for both multivariate approaches indicates that there is no clear overall pattern in relation to allergy. However there were some significant metabolites between the mothers of allergic children and non-allergic children based on univariate statistical models. Figure 4.8 shows the six most significant metabolites that were found in the analysis for the mothers metabolome in relation to allergy. In addition to the metabolites in the figure, glutaric acid (higher in mothers with allergic children) and maltitol (higher in mothers with non-allergic children) were also found to have a significant difference between allergic and non-allergic children.

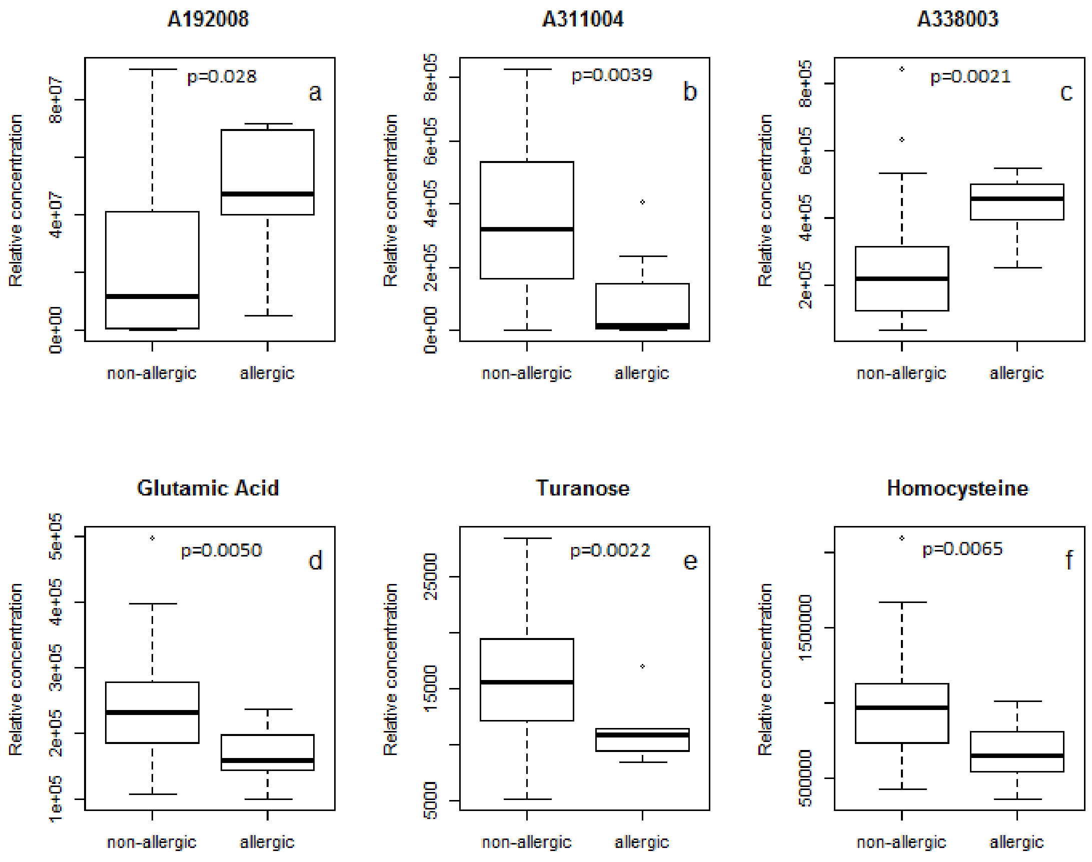


Figure 4.8: Boxplots of the six most significant metabolites in all models for the comparison of samples from mothers with allergic and non-allergic children.

4.2.2 Fathers - Allergy

Similarly to the allergy analyses using the samples from the mothers, both models were found to be inaccurate, but three significant metabolites could still be identified using the variables obtained from the models. The three significant metabolites can be found in Figure 4.9.

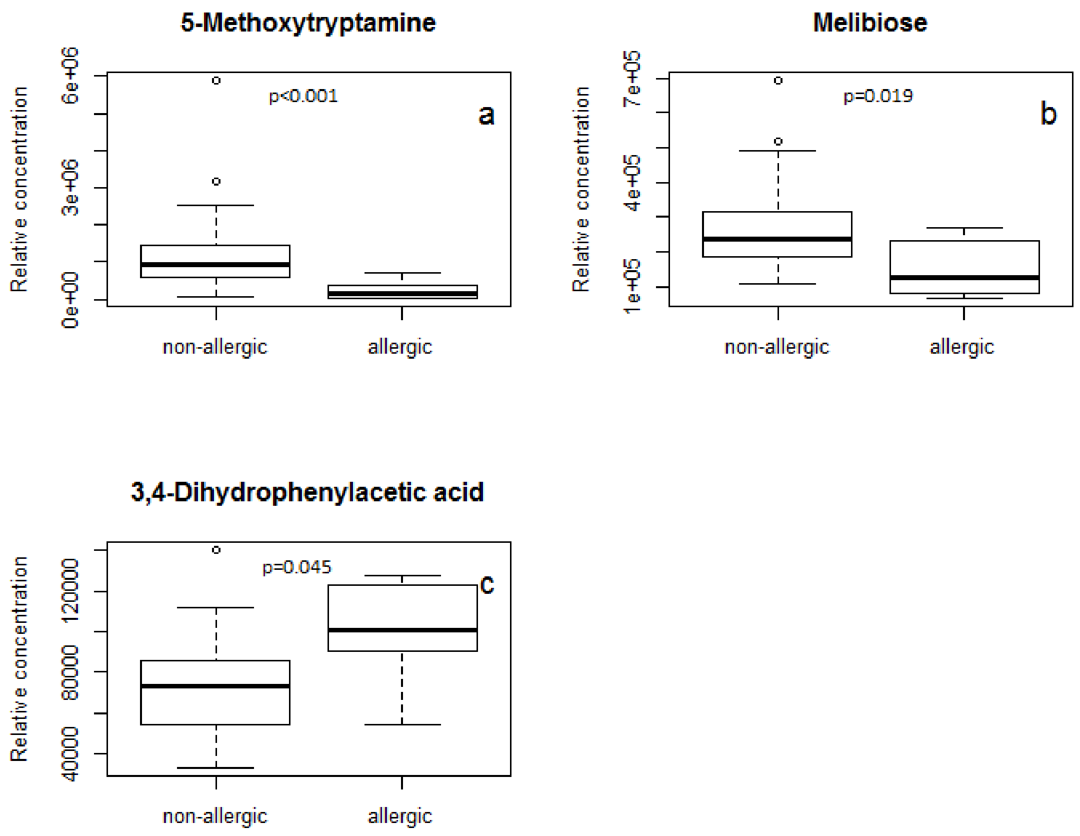


Figure 4.9: Boxplots the three significant metabolites in all models for the comparison of samples from fathers with allergic children and non allergic children.

4.2.3 Umbilical cord samples - Allergy

Neither of the models comparing arterial, venous or mixed umbilical cord blood samples for allergic compared to non-allergic children were significant based on the permutation tests.

When using univariate modelling of individual metabolites, 8 differed in arterial blood, the six most significant of these are plotted in Figure 4.10. In addition to these six, succinic acid and salicin were significantly higher in the non-allergic children. No metabolites in the models for the venous cord blood differed. In mixed umbilical cord blood, sarcosine was higher in non-allergic children (Figure 4.11) with no other differences observed. The models for the arterial samples gave 8 significant metabolites, the six most significant of these are plotted in Figure 4.10. In addition to these six, Succinic acid and Salicin were significantly higher in the non-allergic children.

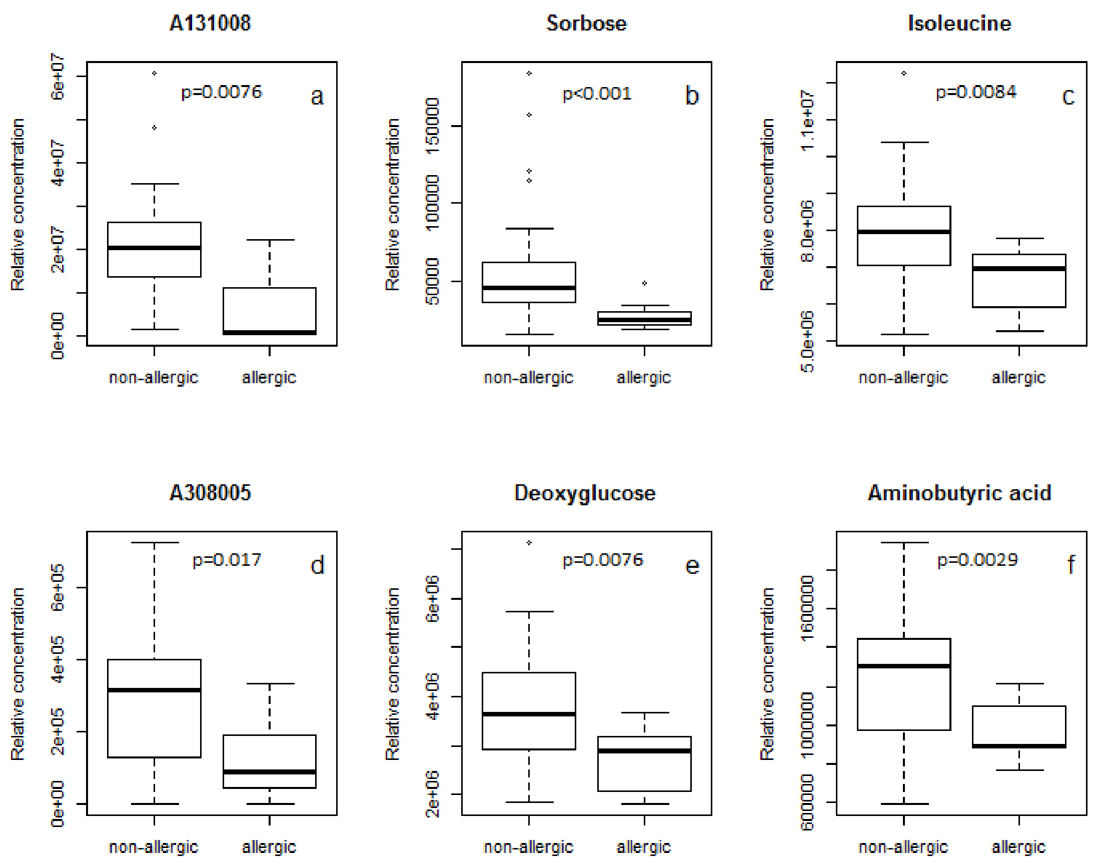


Figure 4.10: Boxplots of the six most significant metabolites in all models for the comparison between allergic and non-allergic children using arterial umbilical cord samples.

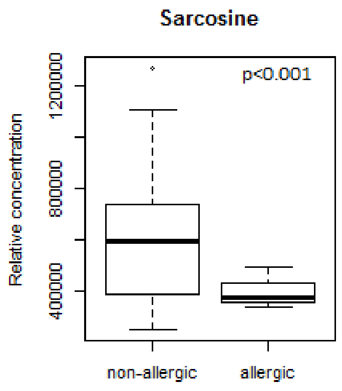


Figure 4.11: Boxplot of the only significant metabolite over all models for the comparison between allergic and non-allergic children using mixed umbilical cord samples.

5 Discussion

5.1 Difference between mothers, fathers and newborn infants

All the models comparing samples from mothers and fathers with umbilical cord samples shows p-values lower than 10^{-5} indicating that all the models are robust and not overfitted, which is otherwise a problem with multivariate models. All models in the comparison between mother and umbilical cord samples are able to well predict the corresponding classes with 0-1 misclassifications and they all have AUROC values very close to 1. When comparing the fathers to the umbilical cord samples, all models show between 1 and 3 misclassifications and AUROC values close to 1. Thus, models based on metabolomics were able to robustly distinguish between mothers, fathers and umbilical cord blood. This is not surprising given the very different conditions the blood samples were collected (mothers during birth, umbilical cord samples immediately post-birth, and fathers at some point in the days before or after the mothers gave birth. Many studies have described differences in the blood plasma metabolome between men and women [49] [50], though none to our knowledge have described the relationship between the metabolome of mother, father and baby.

A major challenge in this type of work is to unravel the differences between parents and children from inter-individual variation, which is often extremely large. The results from the ANOVA decomposition clearly shows that there are a lot of differences between the individuals, and that the differences between children is larger than the differences in venous and arterial cord blood for the same infant. This suggests that while there may be differences between arterial and venous cord blood, these differences are mainly determined by differences between individual babies or families rather than consistent biological differences that are inherent to venous and arterial cord blood.

The concentrations of several amino acids were higher in the children in comparison to the mothers. While the concentrations of several unsaturated fatty acids and α-tocopherol were higher in the mothers. This corresponds well with previous research that has been done where the concentrations of linoleic acid and α-linolenic acid were found in higher concentrations in the mothers and amino acids were found in higher concentrations in the infants [17] [20]. The higher amino acid concentrations in the umbilical cord blood samples may indicate that the child is in an anabolic state, synthesizing a lot of proteins. The higher concentrations of α-tocopherol in the mothers can most likely be explained by the higher concentrations of unsaturated fatty acid. Endogenously α-tocopherol is part of the antioxidant system to prevent unsaturated fatty acids from undergoing lipid peroxidation. As linoleic acid and linolenic acid are both actively transported over the placenta in an order of preference, the high concentrations of these in the mothers could indicate that proteins responsible for the transport are occupied with other fatty acids such as DHA[11]. Lipophilic compounds are also often transported together, so an overall higher lipid content in mothers may also lead to higher relative concentrations of α-tocopherol. Another reason for the high concentrations of the fatty acids could be that the mothers are in a catabolic state, and are utilizing the fatty acids for catabolism as giving birth to a baby involves massive energy expenditure.

The models comparing mothers to fathers appears to be of substantial significance as well, with p-values below 10^{-5} for every model, with 2 and 3 misclassifications for all of the models. The metabolites with highest influence on this model are all higher in the mothers than that of the fathers. The same trend was found in the comparison between children and fathers. This might be due to the fact that the fathers are not put through the same amount of physical stress as the mothers delivering a baby, or the children, being born. When the body is exposed to stress, it is feasible that compounds related to the stress are more likely to be needed in higher concentrations than that of a person in or close to homeostasis.

5.2 Venous compared to arterial umbilical cord blood

The models in the multilevel analysis (multivariate equivalent to paired t-tests) were all significant, although not as strong as the models comparing the different family members to each other. The number of misclassifications in a multilevel analysis might be misleading, as every person has two points. In Table 4.4 26-30 misclassifications in the model represents 13-15 childrens' venous and arterial samples being misclassified as the other. Interestingly, the arterial umbilical cord blood samples appears to have higher concentrations of monosaccharides compared to the venous, suggesting a relative outflow of monosaccharides from the infant to the mother. Metabolically this may be due to mothers using up all their glycogen deposits during labour and the birthing process, so that they are relatively depleted compared to the baby. This hypothesis could also explain why the mothers are so high in unsaturated fatty acids. As the glucose levels in the mothers are depleted, fatty acid catabolism is initiated by splitting triglycerides into free fatty acids and glycerol [51]. As the difference of saturated fatty acids were not as significant, it could be possible that there is some kind of priority in the fatty acid catabolism that is utilizing the unsaturated fatty acids secondarily to the saturated ones, thus leaving higher levels of unsaturated fatty acids in the blood. Another interesting finding in this analysis was the presence of keto-deoxygluconate and its apparent higher concentration in the arterial cord blood. Keto-deoxygluconate is of microbial origin, and is not synthesized by humans [52]. While the biological relevance of a higher amount of keto-deoxygluconate in arterial blood compared to maternal blood is unclear, it does demonstrate that there is transfer of metabolites of microbial origin between the maternal and foetal blood supply.

5.3 Allergy

Despite using balanced error rate as fitness, the models were not able to predict allergy development very well. The main reason for this is that the number of children who have developed allergy during the follow-up period is low, meaning that any model is poorly powered, especially considering the variation due to both sampling and inter-individual differences. This is further confounded by the seven children who did develop allergy having different kinds of allergic diseases, and could therefore have different metabolic patterns depending on what kind of symptom they are displaying. Due to the low number of allergic children, it was meaningless to attempt separate analyses based on the different types of allergies.

Another potential reason could be that allergy does not have that much of a correlation to the *in utero* conditions and the mothers gut microbiome and diet, however this would require data both from the mothers diets as well as data from the gut microbiome of the children to say. That some metabolites that were highly significant were still found throughout the analyses makes this a bit less likely to be the case. It is however important to stress that these investigations are highly preliminary, and that a much larger study is needed to draw any definite conclusions about the metabolic relationship of allergic diseases.

The metabolites that were found to be significant from the univariate testing suggests that there might be some kind of influence working for allergy development already *in utero*. Specifically it seems that there are both saccharides and amino acid related compounds that has an effect. Sorbose and deoxyglucose were found to have an increased concentration in the arterial umbilical cord samples of the non-allergic children. Isoleucine was also found in higher concentrations in the non-allergic childrens arterial samples while sarcosine (glycine derivative) was found to be significantly higher in the mixed umbilical cord samples of the non-allergic children. Aminobutyric acid, more commonly known as GABA, was also found in significantly higher concentrations in the non-allergic childrens' arterial samples. This is a surprising finding as GABA is an inhibitory neurotransmitter and has been suspected of being a signalling substance in asthma [53]. Isoleucine is a branch chained amino acid, which in previous studies has been associated with type 2 diabetes [54]. Branch chained amino acids are also connected to increased protein synthesis via stimulation of insulin in adults. Homocysteine, higher in mothers with non-allergic children, is often associated to bad health as high concentrations has been observed to have a connection to cardiovascular disease, diabetes and cancer development [55]. In this study homocysteine concentrations in the mothers appears to lower the risk for allergy development, indicating that having sufficiently high concentrations in the blood could be promoting health in some regards as well.

Although several metabolites were significant in the allergy investigations, only a handful of them would still be significant if a Bonferroni correction would be applied, and therefore caution about interpreting these data is needed before confirmation in follow up work.

5.4 Strengths and limitations

The largest limitation with this study is the sample size, based on a convenience sampling of available samples. More samples will be available as the NICE-cohort is completed, with over 500 children included. While not all will have suitable blood samples for the comparison of parents and children, there will be an oppotunity to increase the statistical power. Further, this study has only accounted for children who were diagnosed with allergy at one year of age. Childhood allergies develop over time and it is likely that more children in this sample will develop allergy by the time of the next follow-up at four years of age. Another way could be to artificially increase the number of observations in the allergic group, by bootstrapping up the sample size that have been used in this study. This would lead to both groups being more evenly distributed, and using number of misclassifications would give better predictions as it would not be beneficial for the model to group all allergic children with the non-allergic ones. Although it would make the algorithms work better, bootstrapping one of the groups would introduce additional bias to the models, as all the new samples are still dependent on the seven samples that are already in the model. In short, there is no good substitute for adequate statistical power.

A strength of this study is that the fathers has been included in the investigations. Fathers are rarely included in studies regarding infant health even though they could potentially have a large impact on the child. The use of unbiased feature reduction and two different types of multivariate modelling is also a strength as the chance for false positive results (i.e. due to random chance) is reduced compared to many 'conventional' approaches for multivariate modelling of metabolomics data. Another good thing is that more than one kind of algorithm has been applied to the data, leading to results that occur from both models are more statistically sound than if only one would have been used.

5.5 Future prospects

In addition to increasing the sample size it would be of interest to identify all the unknown metabolites that have been found to be of significance to get a better view of the complete picture. In all metabolomics studies using untargeted detection methods, a large number of metabolites remain unidentified due to no good matches with existing mass spectral libraries, and in future studies these unknown metabolites should be monitored and if they do prove to be important, more efforts should be made to identify them. To further investigate what occurs in the child during pregnancy, it would be of interest to check in animal models if there are significant differences in metabolites pre-birth and post-birth, to see how much the birth affects the metabolome. Without this, it is hard to get a good view on what metabolic pathways are active during the development of the foetus. It would also be of interest to investigate several stages of the pregnancy, as the child has different needs in different development stages, there could potentially be a lot of information to be gained about allergy if blood could be sampled from the foetus during different stages of its intrauterine development. As this kind of sampling is potentially lethal to the foetus, mice models could be used to give an idea on how the metabolome is altered during the different stages.

In relation to studying factors that contribute to the development of allergy it is necessary to account for suggested risk factors such as diet and gut microbiota. As these types of data also include a high number of variables, different data analysis modelling will be required compared to what has been performed here.

6
Conclusion

There are significant metabolic differences between mothers, fathers and their children. The discrepancies mainly consist of higher levels of unsaturated fatty acids in mothers, higher concentrations of amino acids in children and the fathers generally having less of these groups in their respective comparison. The difference between venous and arterial cord blood can mainly be attributed to monosaccharides and amino acids. It seems probable that during childbirth, the mother expends most of her energy reserves, leading to higher concentrations of unsaturated fatty acids in her blood as both glucose and saturated fatty acids are converted into energy. This is further supported by the low glucose levels in the venous cord blood compared to the arterial. The higher amount of amino acids in the infants might be related to the newborns being in an anabolic state, where they are trying to synthesize a lot of proteins to be able to grow.

Although there were some metabolites that were related to allergy at 1 year of age, larger sample sizes are needed, especially to enable differentiation between the different kinds of allergic diseases. This work shows that metabolomics may be a promising approach to study metabolic risk-factors for allergy development.

Bibliography

[1] Galli SJ, Tsai M, Piliponsky AM. The development of allergic inflammation. Nature. 2008;454(7203):445.

[2] Kivistö J, Protudjer J, Karjalainen J, Wickman M, Bergström A, Mattila V. Hospitalizations due to allergic reactions in Finnish and Swedish children during 1999–2011. Allergy. 2016;71(5):677–683.

[3] Julia V, Macia L, Dombrowicz D. The impact of diet on asthma and allergic diseases. Nature Reviews Immunology. 2015;15(5):308.

[4] de Agüero MG, Ganal-Vonarburg SC, Fuhrer T, Rupp S, Uchimura Y, Li H, et al. The maternal microbiota drives early postnatal innate immune development. Science. 2016;351(6279):1296–1302.

[5] Wishart DS, Tzur D, Knox C, Eisner R, Guo AC, Young N, et al. HMDB: the human metabolome database. Nucleic acids research. 2007;35(suppl_1):D521–D526.

[6] Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, et al. DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic acids research. 2006;34(suppl_1):D668–D672.

[7] Gu H, Chen H, Pan Z, Jackson AU, Talaty N, Xi B, et al. Monitoring diet effects via biofluids and their implications for metabolomics studies. Analytical chemistry. 2007;79(1):89–97.

[8] Wang Y. Vascular biology of the placenta. In: Colloquium Series on Integrated Systems Physiology: from Molecule to Function. vol. 2. Morgan & Claypool Life Sciences; 2010. p. 1–98.

[9] Haggarty P. Fatty acid supply to the human fetus. Annual review of nutrition. 2010;30:237–255.

[10] Jansson T. Amino acid transporters in the human placenta. Pediatric research. 2001;49(2):141.

[11] Haggarty P, Ashton J, Joynson M, Abramovich D, Page K. Effect of maternal polyunsaturated fatty acid concentration on transport by the human placenta. Neonatology. 1999;75(6):350–359.

[12] Koh DK, Hume R, Eisenhofer G, Watson J, Williams FL. Arterio-venous differences in cord levels of catecholamines, glucose, lactate and blood gases. Journal of perinatal medicine. 2016;44(6):695–704.

[13] OpenStax College. The Placenta; 2018. `https://commons.wikimedia.org/wiki/File:2910_The_Placenta-02.jpg`, „2910 The Placenta-02“, https://creativecommons.org/licenses/by/3.0/legalcode.

[14] Berlin I, Heilbronner C, Georgieu S, Meier C, Spreux-Varoquaux O. Newborns' cord blood plasma cotinine concentrations are similar to that of their delivering smoking mothers. Drug & Alcohol Dependence. 2010;107(2):250–252.

[15] Russell M, Carpenter M, Akhtar M, Lagattuta T, Egorin M. Imatinib mesylate and metabolite concentrations in maternal blood, umbilical cord blood, placenta and breast milk. Journal of Perinatology. 2007;27(4):241.

[16] Li LX, Chen L, Meng XZ, Chen BH, Chen SQ, Zhao Y, et al. Exposure levels of environmental endocrine disruptors in mother-newborn pairs in China and their placental transfer characteristics. PloS one. 2013;8(5):e62526.

[17] Berghaus TM, Demmelmair H, Koletzko B. Essential fatty acids and their long-chain polyunsaturated metabolites in maternal and cord plasma triglycerides during late gestation. Neonatology. 2000;77(2):96–100.

[18] Liu J, Li J, Wu Y, Zhao Y, Luo F, Li S, et al. Bisphenol A metabolites and bisphenol S in paired maternal and cord serum. Environmental science & technology. 2017;51(4):2456–2463.

[19] Wierzejska R, Jarosz M, Sawicki W, Bachanek M, Siuba-Strzelińska M. Vitamin D Concentration in Maternal and Umbilical Cord Blood by Season. International journal of environmental research and public health. 2017;14(10):1121.

[20] Lindblad B, Baldesten A. The Normal Venous Plasma Free Amino Acid Levels of Non-Pregnant Women and of Mother and Child during Delivery. Acta Paediatrica. 1967;56(1):37–48.

[21] Skypala I, Venter C. Food hypersensitivity: diagnosing and managing food allergies and intolerance. John Wiley & Sons; 2009.

[22] Mölne J, Wold A. Inflammation. Liber; 2017.

[23] Strachan DP. Family size, infection and atopy: the first decade of the'hygiene hypothesis'. Thorax. 2000;55(Suppl 1):S2.

[24] Von Mutius E, Braun-Fahrlander C, Schierl R, Riedler J, Ehlermann S, Maisch S, et al. Exposure to endotoxin or other bacterial components might protect against the development of atopy. Clinical and Experimental Allergy. 2000;30(9):1230–1234.

[25] Björkstén B, Sepp E, Julge K, Voor T, Mikelsaar M. Allergy development and the intestinal microflora during the first year of life. Journal of allergy and clinical immunology. 2001;108(4):516–520.

[26] Wopereis H, Sim K, Shaw A, Warner JO, Knol J, Kroll JS. Intestinal microbiota in infants at high risk for allergy: Effects of prebiotics and role in eczema development. Journal of Allergy and Clinical Immunology. 2017;.

[27] Azad MB, Konya T, Guttman DS, Field C, Sears M, HayGlass K, et al. Infant gut microbiota and food sensitization: associations in the first year of life. Clinical & Experimental Allergy. 2015;45(3):632–643.

[28] Aagaard K, Ma J, Antony KM, Ganu R, Petrosino J, Versalovic J. The placenta harbors a unique microbiome. Science translational medicine. 2014;6(237):237ra65–237ra65.

[29] Jiménez E, Marín ML, Martín R, Odriozola JM, Olivares M, Xaus J, et al. Is meconium from healthy newborns actually sterile? Research in microbiology. 2008;159(3):187–193.

[30] Denis DJ. Applied univariate, bivariate, and multivariate statistics. John Wiley & Sons; 2015.

[31] Van den Boogaart KG, Tolosana-Delgado R. Analyzing compositional data with R. vol. 122. Springer; 2013.

[32] de Jong S, Phatak A. Partial least squares regression. Recent advances in total least squares techniques and errors-in-variables modeling. 1997;p. 311–338.

[33] Ståhle L, Wold S. Partial least squares analysis with cross-validation for the two-class problem: A Monte Carlo study. Journal of chemometrics. 1987;1(3):185–196.

[34] Friedman J, Hastie T, Tibshirani R. The elements of statistical learning. vol. 1. Springer series in statistics New York; 2001.

[35] Smilde AK, Jansen JJ, Hoefsloot HC, Lamers RJA, Van Der Greef J, Timmerman ME. ANOVA-simultaneous component analysis (ASCA): a new tool for analyzing designed metabolomics data. Bioinformatics. 2005;21(13):3043–3048.

[36] van Velzen EJ, Westerhuis JA, van Duynhoven JP, van Dorsten FA, Hoefsloot HC, Jacobs DM, et al. Multilevel data analysis of a crossover designed human nutritional intervention study. Journal of proteome research. 2008;7(10):4483–4491.

[37] Westerhuis JA, van Velzen EJ, Hoefsloot HC, Smilde AK. Multivariate paired data analysis: multilevel PLSDA versus OPLSDA. Metabolomics. 2010;6(1):119–128.

[38] Filzmoser P, Liebmann B, Varmuza K. Repeated double cross validation. Journal of Chemometrics. 2009;23(4):160–171.

[39] Savolainen O, Sandberg A, Ross A. A Simultaneous Metabolic Profiling and Quantitative Multimetabolite Metabolomic Method for Human Plasma Using Gas-Chromatography Tandem Mass Spectrometry. Journal of proteome research. 2016;15(1):259.

[40] Hiller K, Hangebrauk J, Jäger C, Spura J, Schreiber K, Schomburg D. MetaboliteDetector: comprehensive analysis tool for targeted and nontargeted GC/MS based metabolome analysis. Analytical chemistry. 2009;81(9):3429–3439.

[41] Jonsson P, Johansson AI, Gullberg J, Trygg J, Grung B, Marklund S, et al. High-throughput data analysis for detecting and identifying differences between samples in GC/MS-based metabolomic analyses. Analytical chemistry. 2005;77(17):5635–5642.

[42] Williams H, Jburney P, Hay R, Archer C, Shipley M, Ahunter J, et al. The UK Working Party's diagnostic criteria for atopic dermatitis. British journal of dermatology. 1994;131(3):383–396.

[43] Williams H, JBURNEY P, Strachan D, Hay R. The UK Working Party's Diagnostic Criteria for Atopic Dermatitis II. Observer variation of clinical diagnosis and signs of atopic dermatitis. British journal of dermatology. 1994;131(3):397–405.

[44] Williams H, Jburney P, Pembroke A, Hay R. The UK Working Party's diagnostic criteria for atopic dermatitis. III. Independent hospital validation. British journal of dermatology. 1994;131(3):406–416.

[45] Umetrics. SIMCA. 14.1; 2015. Info@umetrics.com.

[46] R Core Team. R: A language and environment for statistical computing.; 2017. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org`.

[47] Stekhoven DJ. missForest: Nonparametric Missing Value Imputation using Random Forest; 2013. R package version 1.4.

[48] Stekhoven DJ, Buehlmann P. MissForest - non-parametric missing value imputation for mixed-type data. Bioinformatics. 2012;28(1):112–118.

[49] Krumsiek J, Mittelstrass K, Do KT, Stückler F, Ried J, Adamski J, et al. Gender-specific pathway differences in the human serum metabolome. Metabolomics. 2015;11(6):1815–1833.

[50] Bertram HC, Duus JØ, Petersen BO, Hoppe C, Larnkjær A, Schack-Nielsen L, et al. Nuclear magnetic resonance–based metabonomics reveals strong sex effect on plasma metabolism in 17-year–old Scandinavians and correlation to retrospective infant plasma parameters. Metabolism-Clinical and Experimental. 2009;58(7):1039–1045.

[51] Florkin M, Stotz EH. Pyruvate and fatty acid metabolism. vol. 18. Elsevier; 2016.

[52] Richard P, Hilditch S. D-galacturonic acid catabolism in microorganisms and its biotechnological relevance. Applied microbiology and biotechnology. 2009;82(4):597–604.

[53] Lu WY, Inman M. γ-aminobutyric acid nurtures allergic asthma. Clinical & Experimental Allergy. 2009;39(7):956–961.

[54] Suhre K, Meisinger C, Döring A, Altmaier E, Belcredi P, Gieger C, et al. Metabolic footprint of diabetes: a multiplatform metabolomics study in an epidemiological setting. PloS one. 2010;5(11):e13953.

[55] Williams KT, Schalinske KL. Homocysteine metabolism and its relation to health and disease. Biofactors. 2010;36(1):19–24.

A

Supplementary boxplots for analyses comparing different family members

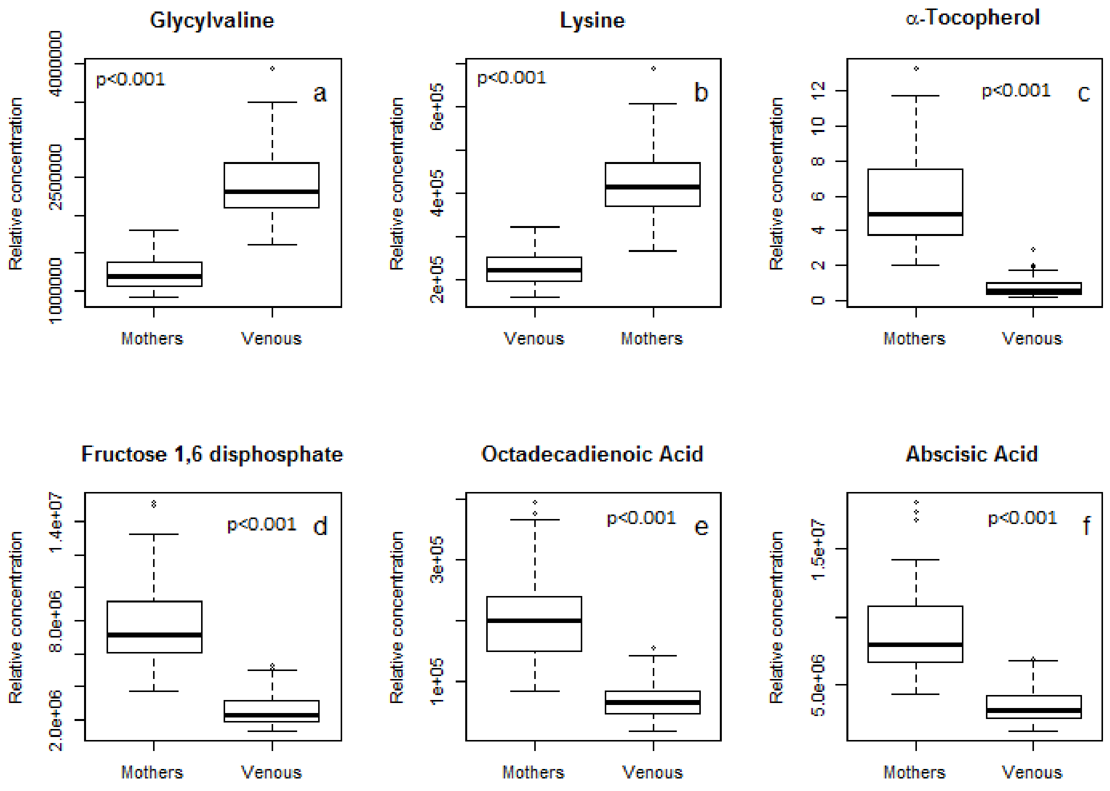


Figure A.1: Boxplots of most influential metabolites in comparison between samples from the mothers and venous umbilical cord samples.

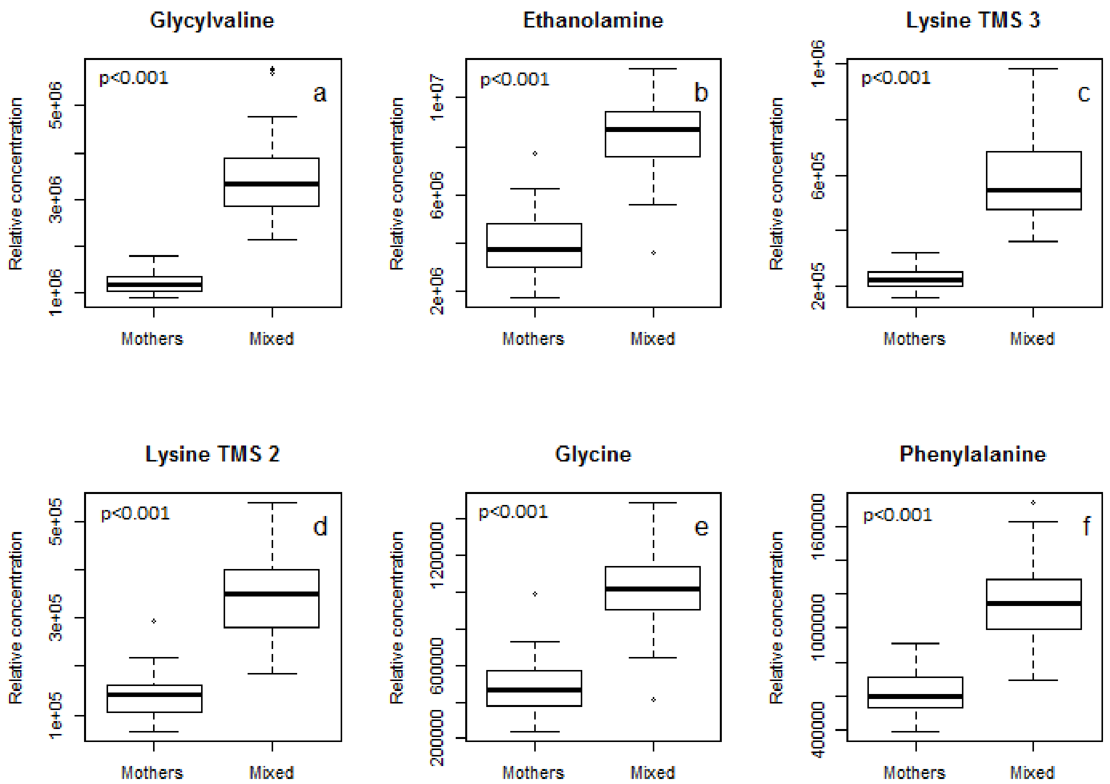


Figure A.2: Boxplots of most influential metabolites in comparison between samples from the mothers and mixed umbilical cord samples.

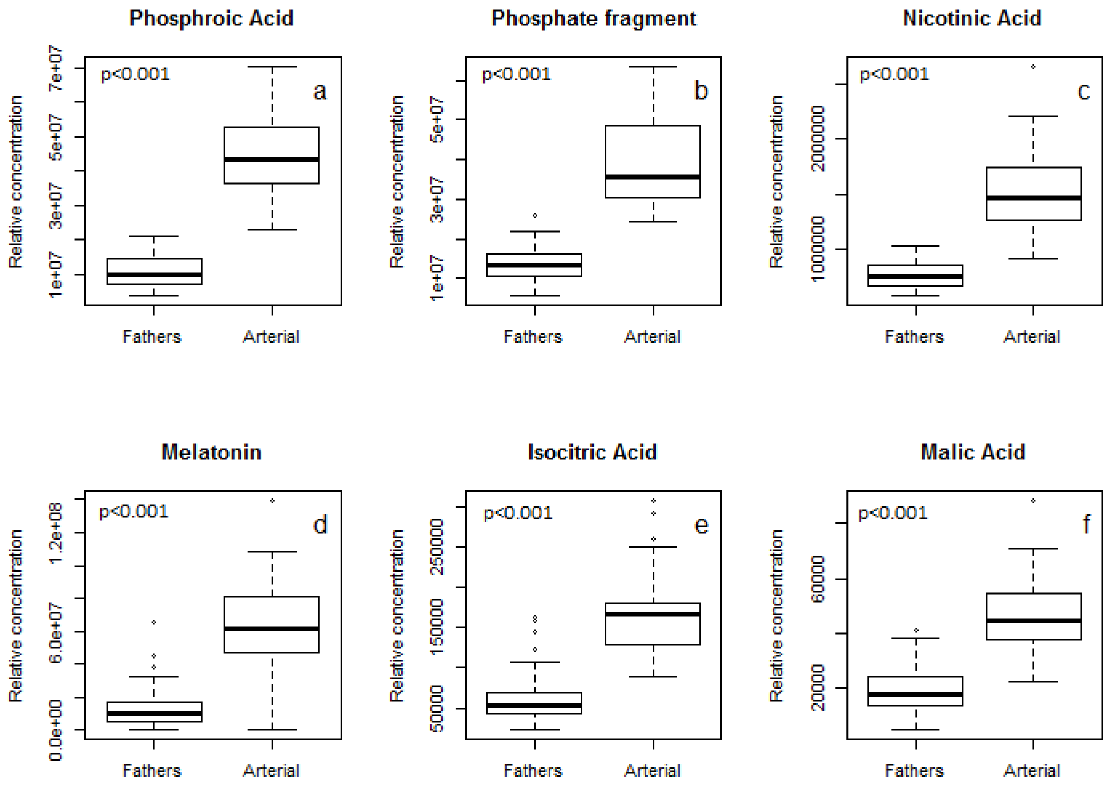


Figure A.3: Boxplots of most influential metabolites in comparison between samples from the fathers and arterial umbilical cord samples.

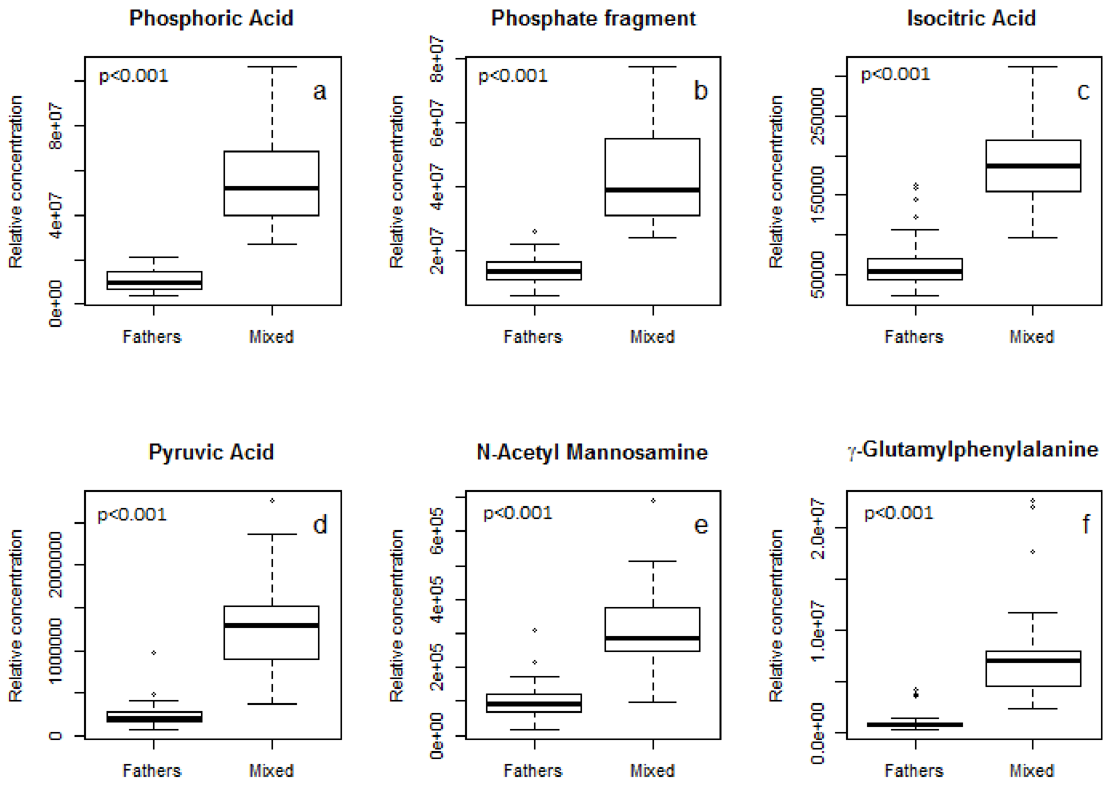


Figure A.4: Boxplots of most influential metabolites in comparison between samples from the fathers and mixed umbilical cord samples.

B
Code

```
#All code for Difference between mother father child.
#loading data and sorting them to matrices
data <- read.csv2(file="data_to_be_used3.csv",
   stringsAsFactors = F)
cmeta=c(1:12)
meta <- data[,cmeta]
Y <- data[,6]
X <- data[,-1:-12]


sum(is.na(X)) #194 missing values
dim(X) #shows dimension of X
194/(255*553) #number of missing values divided by dim(X)
#0.00137574 -> 0,137574 % missing values

#load packages
library(missForest)
library (StatTools)
library(MUVR)
library(doParallel)

#Impute missing values
XNAremoved <- missForest(X, maxiter = 10)


#PCA plots
par(mfrow=c(1,2)) #makes plots side by side

#make first PCA
pcVar1=summary(pcadata)$importance[2,] #get R2X values for
   component.
xlab=paste('PC',1,'␣(R2X=',signif(pcVar1[1],3),')',sep='') #
   defines X label name
ylab=paste('PC',2,'␣(R2X=',signif(pcVar1[2],3),')',sep='') #
   defines Y label name
```

```
plot(pcadata$x[,c(1,2)],main="Initial␣PCA␣plot␣with␣outlier"
     ,col=1,xlab=xlab,ylab=ylab,pch=meta$Family_member-1) #
    plots PCA components 1 and 2
legend('bottomright',col=1,legend=levels(factor(meta$Family_
    member)),pch=c(0,1,2)) #adds legend box to PCA plot
abline(h=0,lty=2) #Adds horizontal line at 0
abline(v=0,lty=2) #Adds vertical line at 0
mtext(c("a)","b)"), at=c(-170, 90))
#removal of outlier
Xnew=XNAremoved$ximp[-which.min(pcadata$x[,1]),]
Ynew=Y[-which.min(pcadata$x[,1])]
metanew=meta[-which.min(pcadata$x[,1]),]

#Make second PCA
pcVar=summary(pcadatanew)$importance[2,] #get R2X values for
     second plot
xlab=paste('PC',1,'␣(R2X=',signif(pcVar[1],3),')',sep='') #
    defines X label name
ylab=paste('PC',2,'␣(R2X=',signif(pcVar[2],3),')',sep='') #
    defines Y label name
plot(pcadatanew$x[,c(1,2)],main="PCA␣plot␣over␣initial␣data"
     ,col=meta$Family_member,xlab=xlab,ylab=ylab,pch=16) #
    Plots PCA components 1 and 2
legend('bottomright',col=c(1,2,3),legend=c('Mothers', '
    Fathers', 'Children'),pch=c(16)) #adds legend box to PCA
    plot
abline(h=0,lty=2) #adds horizontal line at 0
abline(v=0,lty=2) #adds vertical line at 0

#Make second PCA again checking for Batch
pcVar=summary(pcadatanew)$importance[2,] #get R2X values for
     second plot
xlab=paste('PC',1,'␣(R2X=',signif(pcVar[1],3),')',sep='') #
    defines X label name
ylab=paste('PC',2,'␣(R2X=',signif(pcVar[2],3),')',sep='') #
    defines Y label name
plot(pcadatanew$x[,c(1,2)],main="PCA␣plot␣over␣initial␣data"
     ,col=1,xlab=xlab,ylab=ylab,pch=meta$Batch) #Plots PCA
    components 1 and 2
legend('bottomright',col=1,legend=levels(factor(metanew$
    Batch)),pch=c(1,2,3,4)) #adds legend box to PCA plot
abline(h=0,lty=2) #adds horizontal line at 0
abline(v=0,lty=2) #adds vertical line at 0


#sorting data to 1-2-3
```

```
Children<- c()
Mother <- c()
Father <- c()
for (i in 1:length(metanew$Family_member)){
  if(metanew$Family_member[i]==3){
    Children <- append(Children, i)
  } else if(metanew$Family_member[i]==2){
    Father <- append(Father, i)
  } else if(metanew$Family_member[i]==1){
    Mother <- append(Mother, i)
  }
}
Ydiff3=metanew$Family_member[Children]
Ydiff2=metanew$Family_member[Father]
Ydiff1=metanew$Family_member[Mother]
Xdiff3=Xnew[Children,]
Xdiff2=Xnew[Father,]
Xdiff1=Xnew[Mother,]
metadiff3=metanew[Children,]
metadiff2=metanew[Father,]
metadiff1=metanew[Mother,]

Yaller3=metanew$any_allergies[Children]
Yaller2=metanew$any_allergies[Father]
Yaller1=metanew$any_allergies[Mother]

#Separating data into venous arterial and mixed
Arterial <- c()
Venous <- c()
Mixed <- c()
for (i in 1:length(Ydiff3)) {
  if(metadiff3$Sample[i] =="A"){
    Arterial <- append(Arterial, i)
  } else if (metadiff3$Sample[i] =="V"){
    Venous <- append (Venous, i)
  } else if (metadiff3$Sample[i] =="M"){
    Mixed <- append (Mixed, i)
  }
}

XdiffA <- Xdiff3[Arterial,]
XdiffV <- Xdiff3[Venous,]
XdiffM <- Xdiff3[Mixed,]
YdiffA <- Ydiff3[Arterial]
YdiffV <- Ydiff3[Venous]
YdiffM <- Ydiff3[Mixed]
```

```
metadiffA <- metadiff3[Arterial ,]
metadiffV <- metadiff3[Venous ,]
metadiffM <- metadiff3[Mixed ,]

YallerA <- Yaller3[Arterial]
YallerV <- Yaller3[Venous]
YallerM <- Yaller3[Mixed]

#Starting actual calculations.
nPerm=100 #100 for real investigation and 25 for "quick"
   analysis
varratio=0.9 #0.9 for real investigations and 0.75 for "
   quick" analysis
nCore=detectCores()-1 #creates a variable with amount of
   cores -1 to be used to parallel computing
#1v2

Xdiff12 <- rbind(Xdiff1, Xdiff2)
Ydiff12 <- c(Ydiff1, Ydiff2)
metadiff12 <- rbind(metadiff1, metadiff2)
permFitnessdiff12min=numeric(nPerm)
permFitnessdiff12mid=numeric(nPerm)
permFitnessdiff12max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
Modeldiff12=MUVR(X=Xdiff12, Y=Ydiff12, varRatio = varratio,
   ID=metadiff12$Family_number, nRep=2*nCore, nOuter=5,
   method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Ydiff12)
  permModeldiff12=MUVR(X=Xdiff12, Y=Yperm, varRatio=varratio
     , ID=metadiff12$Family_number, nRep=2*nCore, nOuter=5,
     method='PLS', DA=T)
  permFitnessdiff12min[p]=permModeldiff12$miss[1]
  permFitnessdiff12mid[p]=permModeldiff12$miss[2]
  permFitnessdiff12max[p]=permModeldiff12$miss[3]
}
stopCluster(cl)

#RF1-2
permFitnessdiffRF12min=numeric(nPerm)
permFitnessdiffRF12mid=numeric(nPerm)
permFitnessdiffRF12max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
```

```
ModeldiffRF12=MUVR(X=Xdiff12 ,Y=as.character(Ydiff12),
   varRatio = varratio , ID=metadiff12$Family_number, nRep=2*
   nCore, nOuter=5,method='RF', fitness = 'MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Ydiff12)
  permModeldiff12=MUVR(X=Xdiff12 , Y=as.character(Yperm),
     varRatio=varratio , ID=metadiff12$Family_number, nRep=2*
     nCore, nOuter=5,method='RF', fitness = 'MISS')
  permFitnessdiffRF12min[p]=permModeldiff12$miss[1]
  permFitnessdiffRF12mid[p]=permModeldiff12$miss[2]
  permFitnessdiffRF12max[p]=permModeldiff12$miss[3]
}
stopCluster(cl)


#RF V-1
XdiffV1 <- rbind(XdiffV, Xdiff1)
YdiffV1 <- c(YdiffV, Ydiff1)
metadiffV1 <- rbind(metadiffV, metadiff1)
permFitnessdiffRFV1min=numeric(nPerm)
permFitnessdiffRFV1mid=numeric(nPerm)
permFitnessdiffRFV1max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffRFV1=MUVR(X=XdiffV1 ,Y=as.character(YdiffV1),
   varRatio = varratio , ID=metadiffV1$Family_number, nRep=2*
   nCore, nOuter=5,method='RF',fitness = 'MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffV1)
  permModeldiffV1=MUVR(X=XdiffV1 , Y=as.character(Yperm),
     varRatio=varratio , ID=metadiffV1$Family_number, nRep=2*
     nCore, nOuter=5,method='RF', fitness = 'MISS')
  permFitnessdiffRFV1min[p]=permModeldiffV1$miss[1]
  permFitnessdiffRFV1mid[p]=permModeldiffV1$miss[2]
  permFitnessdiffRFV1max[p]=permModeldiffV1$miss[3]
}
stopCluster(cl)

#PLS v-1
permFitnessdiffV1min=numeric(nPerm)
permFitnessdiffV1mid=numeric(nPerm)
permFitnessdiffV1max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
```

```
ModeldiffV1=MUVR(X=XdiffV1, Y=YdiffV1, varRatio = varratio,
   ID=metadiffV1$Family_number, nRep=2*nCore, nOuter=5,
   method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffV1)
  permModeldiffV1=MUVR(X=XdiffV1, Y=Yperm, varRatio=varratio
      , ID=metadiffV1$Family_number, nRep=2*nCore, nOuter=5,
      method='PLS', DA=T)
  permFitnessdiffV1min[p]=permModeldiffV1$miss[1]
  permFitnessdiffV1mid[p]=permModeldiffV1$miss[2]
  permFitnessdiffV1max[p]=permModeldiffV1$miss[3]
}
stopCluster(cl)


#RF V-2
XdiffV2 <- rbind(XdiffV, Xdiff2)
YdiffV2 <- c(YdiffV, Ydiff2)
metadiffV2 <- rbind(metadiffV, metadiff2)
permFitnessdiffRFV2min=numeric(nPerm)
permFitnessdiffRFV2mid=numeric(nPerm)
permFitnessdiffRFV2max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffRFV2=MUVR(X=XdiffV2,Y=as.character(YdiffV2),
   varRatio = varratio, ID=metadiffV2$Family_number, nRep=2*
   nCore, nOuter=5,method='RF', fitness='MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffV2)
  permModeldiffV1=MUVR(X=XdiffV2, Y=as.character(Yperm),
      varRatio=varratio, ID=metadiffV2$Family_number, nRep=2*
      nCore, nOuter=5,method='RF', fitness='MISS')
  permFitnessdiffRFV2min[p]=permModeldiffV2$miss[1]
  permFitnessdiffRFV2mid[p]=permModeldiffV2$miss[2]
  permFitnessdiffRFV2max[p]=permModeldiffV2$miss[3]
}
stopCluster(cl)

#PLS v-2
permFitnessdiffV2min=numeric(nPerm)
permFitnessdiffV2mid=numeric(nPerm)
permFitnessdiffV2max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
```

```
ModeldiffV2=MUVR(X=XdiffV2, Y=YdiffV2, varRatio = varratio,
   ID=metadiffV2$Family_number, nRep=2*nCore, nOuter=5,
   method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffV2)
  permModeldiffV2=MUVR(X=XdiffV2, Y=Yperm, varRatio=varratio
      , ID=metadiffV2$Family_number, nRep=2*nCore, nOuter=5,
      method='PLS', DA=T)
  permFitnessdiffV2min[p]=permModeldiffV2$miss[1]
  permFitnessdiffV2mid[p]=permModeldiffV2$miss[2]
  permFitnessdiffV2max[p]=permModeldiffV2$miss[3]
}
stopCluster(cl)



#RF M-1
XdiffM1 <- rbind(XdiffM, Xdiff1)
YdiffM1 <- c(YdiffM, Ydiff1)
metadiffM1 <- rbind(metadiffM, metadiff1)
permFitnessdiffRFM1min=numeric(nPerm)
permFitnessdiffRFM1mid=numeric(nPerm)
permFitnessdiffRFM1max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffRFM1=MUVR(X=XdiffM1,Y=as.character(YdiffM1),
   varRatio = varratio, ID=metadiffM1$Family_number, nRep=2*
   nCore, nOuter=5,method='RF', fitness = 'MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffM1)
  permModeldiffM1=MUVR(X=XdiffM1, Y=as.character(Yperm),
      varRatio=varratio, ID=metadiffM1$Family_number, nRep=2*
      nCore, nOuter=5,method='RF', fitness = 'MISS')
  permFitnessdiffRFM1min[p]=permModeldiffM1$miss[1]
  permFitnessdiffRFM1mid[p]=permModeldiffM1$miss[2]
  permFitnessdiffRFM1max[p]=permModeldiffM1$miss[3]
}
stopCluster(cl)

#PLS M-1
permFitnessdiffM1min=numeric(nPerm)
permFitnessdiffM1mid=numeric(nPerm)
permFitnessdiffM1max=numeric(nPerm)
cl=makeCluster(nCore)
```

```
registerDoParallel(cl)
ModeldiffM1=MUVR(X=XdiffM1, Y=YdiffM1, varRatio = varratio,
   ID=metadiffM1$Family_number, nRep=2*nCore, nOuter=5,
   method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffM1)
  permModeldiffM1=MUVR(X=XdiffM1, Y=Yperm, varRatio=varratio
     , ID=metadiffM1$Family_number, nRep=2*nCore, nOuter=5,
     method='PLS', DA=T)
  permFitnessdiffM1min[p]=permModeldiffM1$miss[1]
  permFitnessdiffM1mid[p]=permModeldiffM1$miss[2]
  permFitnessdiffM1max[p]=permModeldiffM1$miss[3]
}
stopCluster(cl)



#RF M-2
XdiffM2 <- rbind(XdiffM, Xdiff2)
YdiffM2 <- c(YdiffM, Ydiff2)
metadiffM2 <- rbind(metadiffM, metadiff2)
permFitnessdiffRFM2min=numeric(nPerm)
permFitnessdiffRFM2mid=numeric(nPerm)
permFitnessdiffRFM2max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffRFM2=MUVR(X=XdiffM2,Y=as.character(YdiffM2),
   varRatio = varratio, ID=metadiffM2$Family_number, nRep=2*
   nCore, nOuter=5,method='RF', fitness = 'MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffM2)
  permModeldiffM2=MUVR(X=XdiffM2, Y=as.character(Yperm),
     varRatio=varratio, ID=metadiffM2$Family_number, nRep=2*
     nCore, nOuter=5,method='RF', fitness = 'MISS')
  permFitnessdiffRFM2min[p]=permModeldiffM2$miss[1]
  permFitnessdiffRFM2mid[p]=permModeldiffM2$miss[2]
  permFitnessdiffRFM2max[p]=permModeldiffM2$miss[3]
}
stopCluster(cl)

#PLS M-2
permFitnessdiffM2min=numeric(nPerm)
permFitnessdiffM2mid=numeric(nPerm)
permFitnessdiffM2max=numeric(nPerm)
```

```
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffM2=MUVR(X=XdiffM2, Y=YdiffM2, varRatio = varratio,
   ID=metadiffM2$Family_number, nRep=2*nCore, nOuter=5,
   method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffM2)
  permModeldiffM2=MUVR(X=XdiffM2, Y=Yperm, varRatio=varratio
      , ID=metadiffM2$Family_number, nRep=2*nCore, nOuter=5,
      method='PLS', DA=T)
  permFitnessdiffM2min[p]=permModeldiffM2$miss[1]
  permFitnessdiffM2mid[p]=permModeldiffM2$miss[2]
  permFitnessdiffM2max[p]=permModeldiffM2$miss[3]
}
stopCluster(cl)



#RF A-1
XdiffA1 <- rbind(XdiffA, Xdiff1)
YdiffA1 <- c(YdiffA, Ydiff1)
metadiffA1 <- rbind(metadiffA, metadiff1)
permFitnessdiffRFA1min=numeric(nPerm)
permFitnessdiffRFA1mid=numeric(nPerm)
permFitnessdiffRFA1max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffRFA1=MUVR(X=XdiffA1,Y=as.character(YdiffA1),
   varRatio = varratio, ID=metadiffA1$Family_number, nRep=2*
   nCore, nOuter=5,method='RF', fitness = 'MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffA1)
  permModeldiffA1=MUVR(X=XdiffA1, Y=as.character(Yperm),
      varRatio=varratio, ID=metadiffA1$Family_number, nRep=2*
      nCore, nOuter=5,method='RF', fitness = 'MISS')
  permFitnessdiffRFA1min[p]=permModeldiffA1$miss[1]
  permFitnessdiffRFA1mid[p]=permModeldiffA1$miss[2]
  permFitnessdiffRFA1max[p]=permModeldiffA1$miss[3]
}
stopCluster(cl)

#PLS A-1
permFitnessdiffA1min=numeric(nPerm)
permFitnessdiffA1mid=numeric(nPerm)
```

```
permFitnessdiffA1max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffA1=MUVR(X=XdiffA1, Y=YdiffA1, varRatio = varratio,
   ID=metadiffA1$Family_number, nRep=2*nCore, nOuter=5,
   method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffA1)
  permModeldiffA1=MUVR(X=XdiffA1, Y=Yperm, varRatio=varratio
     , ID=metadiffA1$Family_number, nRep=2*nCore, nOuter=5,
     method='PLS', DA=T)
  permFitnessdiffA1min[p]=permModeldiffA1$miss[1]
  permFitnessdiffA1mid[p]=permModeldiffA1$miss[2]
  permFitnessdiffA1max[p]=permModeldiffA1$miss[3]
}
stopCluster(cl)



#RF A-2
XdiffA2 <- rbind(XdiffA, Xdiff2)
YdiffA2 <- c(YdiffA, Ydiff2)
metadiffA2 <- rbind(metadiffA, metadiff2)
permFitnessdiffRFA2min=numeric(nPerm)
permFitnessdiffRFA2mid=numeric(nPerm)
permFitnessdiffRFA2max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffRFA2=MUVR(X=XdiffA2,Y=as.character(YdiffA2),
   varRatio = varratio, ID=metadiffA2$Family_number, nRep=2*
   nCore, nOuter=5,method='RF', fitness='MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffA2)
  permModeldiffA2=MUVR(X=XdiffA2, Y=as.character(Yperm),
     varRatio=varratio, ID=metadiffA2$Family_number, nRep=2*
     nCore, nOuter=5,method='RF', fitness='MISS')
  permFitnessdiffRFA2min[p]=permModeldiffA2$miss[1]
  permFitnessdiffRFA2mid[p]=permModeldiffA2$miss[2]
  permFitnessdiffRFA2max[p]=permModeldiffA2$miss[3]
}
stopCluster(cl)

#PLS A-2
permFitnessdiffA2min=numeric(nPerm)
```

```
permFitnessdiffA2mid=numeric(nPerm)
permFitnessdiffA2max=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ModeldiffA2=MUVR(X=XdiffA2, Y=YdiffA2, varRatio = varratio,
   ID=metadiffA2$Family_number, nRep=2*nCore, nOuter=5,
   method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YdiffA2)
  permModeldiffA2=MUVR(X=XdiffA2, Y=Yperm, varRatio=varratio
     , ID=metadiffA2$Family_number, nRep=2*nCore, nOuter=5,
     method='PLS', DA=T)
  permFitnessdiffA2min[p]=permModeldiffA2$miss[1]
  permFitnessdiffA2mid[p]=permModeldiffA2$miss[2]
  permFitnessdiffA2max[p]=permModeldiffA2$miss[3]
}
stopCluster(cl)


#Evaluate models

#Modeldiff12

#checking fitness parameters
ModeldiffRF12$auc
Modeldiff12$auc
ModeldiffRF12$miss
Modeldiff12$miss

#checking significance with permutation analysis
pPerm(Modeldiff12$miss[1], permFitnessdiff12min)
pPerm(Modeldiff12$miss[2], permFitnessdiff12mid)
pPerm(Modeldiff12$miss[3], permFitnessdiff12max)



pPerm(ModeldiffRF12$miss[1], permFitnessdiffRF12min)
pPerm(ModeldiffRF12$miss[2], permFitnessdiffRF12mid)
pPerm(ModeldiffRF12$miss[3], permFitnessdiffRF12max)

#saving metabolite information to variables
min12=getVIP(Modeldiff12, model='min')
mid12=getVIP(Modeldiff12, model='mid')
max12=getVIP(Modeldiff12, model='max')

minRF12=getVIP(ModeldiffRF12, model='min')
```

```
midRF12=getVIP(ModeldiffRF12, model='mid')
maxRF12=getVIP(ModeldiffRF12, model='max')

#checking length of matrices
nrow(min12)
nrow(mid12)
nrow(max12)
head(min12,10)
head(mid12,10)
head(max12,10)
#saving table information
t12=cbind.data.frame(head(min12$name, 10), head(min12$rank,
   10), head(mid12$name, 10), head(mid12$rank, 10), head(
   max12$name, 10), head(max12$rank, 10))

#checking length of matrices
nrow(minRF12)
nrow(midRF12)
nrow(maxRF12)
head(minRF12,10)
head(midRF12,10)
head(maxRF12,10)
#saving table information
t12RF=cbind.data.frame(head(minRF12$name, 10), head(minRF12$
   rank, 10), head(midRF12$name, 10), head(midRF12$rank, 10)
   , head(maxRF12$name, 10), head(maxRF12$rank, 10))
write.table(t12, "PLStable12", sep=";", row.names=F, quote=F
   )
write.table(t12RF, "RFtable12", sep=";", row.names=F, quote=
   F)

#checking which of the top 10 metabolite concentrations are
   highest in which sample type
#PLS-DA
mean(Xdiff1$Phosphoric_acid__3TMS_)-mean(Xdiff2$Phosphoric_
   acid__3TMS_)
mean(Xdiff1$PYRUVIC_ACID.MEOX_TMS___1__M.z89)-mean(Xdiff2$
   PYRUVIC_ACID.MEOX_TMS___1__M.z89)
mean(Xdiff1$PHOSPHATE.FRAGMENT___1__M.z299)-mean(Xdiff2$
   PHOSPHATE.FRAGMENT___1__M.z299)
mean(Xdiff1$NICOTINIC_ACID.TMS___1)-mean(Xdiff2$NICOTINIC_
   ACID.TMS___1)
mean(Xdiff1$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS___1)-
   mean(Xdiff2$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS___1)
mean(Xdiff1$ISOCITRIC_ACID.4TMS___1__M.z245)-mean(Xdiff2$
   ISOCITRIC_ACID.4TMS___1__M.z245)
```

```
mean(Xdiff1$Cic.aconitic_acid)-mean(Xdiff2$Cic.aconitic_acid
    )
mean(Xdiff1$Tryptophan___5.hydroxy.___3TMS___BP)-mean(Xdiff2$
    Tryptophan___5.hydroxy.___3TMS___BP)
mean(Xdiff1$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2____2)-mean(
    Xdiff2$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2____2)
mean(Xdiff1$Melatonin___2TMS_)-mean(Xdiff2$Melatonin___2TMS_)

#Plotting RF metabolites that are not in PLS-DA model
mean(Xdiff1$GLYCERIC_ACID.2.PHOSPHATE.4TMS____3__M.z315)-mean
    (Xdiff2$GLYCERIC_ACID.2.PHOSPHATE.4TMS____3__M.z315)

#t-tests for boxplot metabolites
t.test(Xdiff12$Phosphoric_acid___3TMS_~Ydiff12)
t.test(Xdiff12$PYRUVIC_ACID.MEOX_TMS____1__M.z89~Ydiff12)
t.test(Xdiff12$PHOSPHATE.FRAGMENT____1__M.z299~Ydiff12)
t.test(Xdiff12$NICOTINIC_ACID.TMS____1~Ydiff12)
t.test(Xdiff12$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1~
    Ydiff12)
t.test(Xdiff12$ISOCITRIC_ACID.4TMS____1__M.z245~Ydiff12)

#create boxplots for univariate analysis
par(mfrow=c(2,3))
boxplot(Xdiff12$Phosphoric_acid___3TMS_~Ydiff12, main='
    Phosphoric Acid', xaxt='n', ylab='Relative concentration'
    )
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Fathers'))
boxplot(Xdiff12$PYRUVIC_ACID.MEOX_TMS____1__M.z89~Ydiff12,
    main='Pyruvic Acid', xaxt='n', ylab='Relative 
    concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Fathers'))
boxplot(Xdiff12$PHOSPHATE.FRAGMENT____1__M.z299~Ydiff12, main
   ='Phosphate fragment', xaxt='n', ylab='Relative 
    concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Fathers'))
boxplot(Xdiff12$NICOTINIC_ACID.TMS____1~Ydiff12, main='
    Nicotinic Acid', xaxt='n', ylab='Relative concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Fathers'))
boxplot(Xdiff12$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1
   ~Ydiff12, main=expression(bold(paste(gamma, '-
    Glutamylphenylalanine', sep=' '))), xaxt='n', ylab='
    Relative concentration')
```

```
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Fathers'))
boxplot(Xdiff12$ISOCITRIC_ACID.4TMS____1__M.z245~Ydiff12,
   main='Isocitric␣Acid', xaxt='n', ylab='Relative␣
   concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Fathers'))

#A-1
#checking fitness parameters
ModeldiffRFA1$auc
ModeldiffA1$auc
ModeldiffRFA1$miss
ModeldiffA1$miss

#checking p-values for permutations tests
pPerm(ModeldiffA1$auc[1,2], permFitnessdiffA1min)
pPerm(ModeldiffA1$auc[2,2], permFitnessdiffA1mid)
pPerm(ModeldiffA1$auc[3,2], permFitnessdiffA1max)
pPerm(ModeldiffRFA1$auc[1,2], permFitnessdiffRFA1min)
pPerm(ModeldiffRFA1$auc[2,2], permFitnessdiffRFA1mid)
pPerm(ModeldiffRFA1$auc[3,2], permFitnessdiffRFA1max)

#saving variables by VIP score for table plotting
minRFA1=getVIP(ModeldiffRFA1, model='min')
midRFA1=getVIP(ModeldiffRFA1, model='mid')
maxRFA1=getVIP(ModeldiffRFA1, model='max')

minA1=getVIP(ModeldiffA1, model='min')
midA1=getVIP(ModeldiffA1, model='mid')
maxA1=getVIP(ModeldiffA1, model='max')

#checking number of metabolites for each model and saving
   variables for plotting tables
nrow(minA1)
nrow(midA1)
nrow(maxA1)

#Note - fix latex appendix tables. See if boxplots need
   configuration as well
tA1=cbind.data.frame(c(head(minA1$name, 10), rep(NA,7)), c(
   head(minA1$rank, 10),rep(NA,7)), head(midA1$name, 10),
   head(midA1$rank, 10), head(maxA1$name, 10), head(maxA1$
   rank, 10))
minA1
midA1
```

```
maxA1
nrow(minRFA1)
nrow(midRFA1)
nrow(maxRFA1)
tA1RF=cbind.data.frame(c(head(minRFA1$name, 10),rep(NA,5)),
   c(head(minRFA1$rank, 10),rep(NA,5)), c(head(midRFA1$name,
    10), rep(NA,3)), c(head(midRFA1$rank, 10), rep(NA,3)),
   head(maxRFA1$name, 10), head(maxRFA1$rank, 10))
minRFA1
midRFA1
head(maxRFA1,10)
#writing tables to csv files
write.table(tA1, "PLStableA1", sep=";", row.names=F, quote=F
   )
write.table(tA1RF, "RFtableA1", sep=";", row.names=F, quote=
   F)

#checking which of the top 10 metabolite concentrations are
   highest in which sample type
#PLS-DA
mean(XdiffA$GLYCYLVALINE.4TMS____1__M.z174)-mean(Xdiff1$
   GLYCYLVALINE.4TMS____1__M.z174)
mean(XdiffA$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2____2)-mean(
   Xdiff1$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2____2)
mean(XdiffA$alpha.Tocopherol)-mean(Xdiff1$alpha.Tocopherol)
mean(XdiffA$PHENYLALANINE_N_O.TMS____1__M.z218)-mean(Xdiff1$
   PHENYLALANINE_N_O.TMS____1__M.z218)
mean(XdiffA$TRYPTOPHAN_N_N_O.TMS____2__M.z202)-mean(Xdiff1$
   TRYPTOPHAN_N_N_O.TMS____2__M.z202)
mean(XdiffA$OCTADECADIENOIC_ACID___9_12._Z_Z_.TMS____1__M.z337
   )-mean(Xdiff1$OCTADECADIENOIC_ACID___9_12._Z_Z_.TMS____1__M
   .z337)
mean(XdiffA$LYSINE_N_N_O.TMS_3____1__M.z200)-mean(Xdiff1$
   LYSINE_N_N_O.TMS_3____1__M.z200)
mean(XdiffA$OCTADECATRIENOIC_ACID___6_9_12._Z_Z_Z_.TMS____1__M
   .z120)-mean(Xdiff1$OCTADECATRIENOIC_ACID___6_9_12._Z_Z_Z_.
   TMS____1__M.z120)
mean(XdiffA$ABSCISIC_ACID.MEOX_2TMS_2____2)-mean(Xdiff1$
   ABSCISIC_ACID.MEOX_2TMS_2____2)
mean(XdiffA$ABSCISIC_ACID.MEOX_2TMS_1____1)-mean(Xdiff1$
   ABSCISIC_ACID.MEOX_2TMS_1____1)

#metabolites found in RF models that are not in PLS-DA
mean(XdiffA$OCTADECENOIC_ACID___9._E_.TMS____1__M.z137)-mean(
   Xdiff1$OCTADECENOIC_ACID___9._E_.TMS____1__M.z137)
```

```
#t-tests for boxplot metabolites
t.test(XdiffA1$GLYCYLVALINE.4TMS____1__M.z174~YdiffA1)
t.test(XdiffA1$FRUCTOSE.1__6.DISPHOSPHATE.MEOX_7TMS_2____2~
   YdiffA1)
t.test(XdiffA1$alpha.Tocopherol~YdiffA1)
t.test(XdiffA1$LYSINE_N_N_O.TMS_3____1__M.z200~YdiffA1)
t.test(XdiffA1$OCTADECADIENOIC_ACID___9_12.__Z_Z__.TMS____1__M.
   z337~YdiffA1)
t.test(XdiffA1$PHENYLALANINE_N_O.TMS____1__M.z218~YdiffA1)


#create boxplots for univariate visualization
par(mfrow=c(2,3))
boxplot(XdiffA1$GLYCYLVALINE.4TMS____1__M.z174~YdiffA1, main=
   'Glycylvaline', xaxt='n', ylab='Relative concentration')
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Arterial'))
boxplot(XdiffA1$FRUCTOSE.1__6.DISPHOSPHATE.MEOX_7TMS_2____2~
   YdiffA1, main='Fructose 1,6 disphosphate', xaxt='n', ylab
   ='Relative concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Arterial'))
boxplot(XdiffA1$alpha.Tocopherol~YdiffA1, main=expression(
   bold(paste(alpha, '-Tocopherol', sep=''))), xaxt='n', ylab
   ='Relative concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Arterial'))
boxplot(XdiffA1$LYSINE_N_N_O.TMS_3____1__M.z200~YdiffA1, main
   ='Lysine', xaxt='n', ylab='Relative concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Arterial'))
boxplot(XdiffA1$OCTADECADIENOIC_ACID___9_12.__Z_Z__.TMS____1__M.
   z337~YdiffA1, main='Octadecadienoic Acid', xaxt='n', ylab
   ='Relative concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Arterial'))
boxplot(XdiffA1$PHENYLALANINE_N_O.TMS____1__M.z218~YdiffA1,
   main='Phenylalanine', xaxt='n', ylab='Relative 
   concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Arterial'))



#A-2
```

```
#checking fitness parameters
ModeldiffRFA2$auc
ModeldiffA2$auc
ModeldiffRFA2$miss
ModeldiffA2$miss

#checking p values with the help of permutations tests
pPerm(ModeldiffRFA2$auc[1,2], permFitnessdiffRFA2min)
pPerm(ModeldiffRFA2$auc[2,2], permFitnessdiffRFA2mid)
pPerm(ModeldiffRFA2$auc[3,2], permFitnessdiffRFA2max)

pPerm(ModeldiffA2$auc[1,2], permFitnessdiffA2min)
pPerm(ModeldiffA2$auc[2,2], permFitnessdiffA2mid)
pPerm(ModeldiffA2$auc[3,2], permFitnessdiffA2max)

#saving metabolites sorted by VIP to be able to plot tables
minRFA2=getVIP(ModeldiffRFA2, model='min')
midRFA2=getVIP(ModeldiffRFA2, model='mid')
maxRFA2=getVIP(ModeldiffRFA2, model='max')
minA2=getVIP(ModeldiffA2, model='min')
midA2=getVIP(ModeldiffA2, model='mid')
maxA2=getVIP(ModeldiffA2, model='max')

#checking number of metabolites for each model and saving
    information for tables.
nrow(minA2) #checking length of matrix
nrow(midA2)
nrow(maxA2)
tA2=cbind.data.frame(c(head(minA2$name, 10), rep(NA,7)), c(
    head(minA2$rank, 10),rep(NA,7)), head(midA2$name, 10),
    head(midA2$rank, 10), head(maxA2$name, 10), head(maxA2$
    rank, 10))
minA2$name
nrow(minRFA2)
nrow(midRFA2)
nrow(maxRFA2)
tA2RF=cbind.data.frame(c(head(minRFA2$name, 10), NA), c(head
    (minRFA2$rank, 10), NA), c(head(midRFA2$name, 10), NA), c
    (head(midRFA2$rank, 10), NA), head(maxRFA2$name, 10),
    head(maxRFA2$rank, 10))
minRFA2
midRFA2
head(maxRFA2,10)
#saving tables as .csv files
write.table(tA2, "PLStableA2", sep=";", row.names=F, quote=F
    )
```

```
write.table(tA2RF, "RFtableA2", sep=";", row.names=F, quote=
   F)

#checking which of the top 10 metabolite concentrations are
    highest in which sample type
#PLS-DA
mean(XdiffA$Phosphoric_acid__3TMS)-mean(Xdiff2$Phosphoric_
    acid__3TMS)
mean(XdiffA$PHOSPHATE.FRAGMENT___1__M.z299)-mean(Xdiff2$
   PHOSPHATE.FRAGMENT___1__M.z299)
mean(XdiffA$NICOTINIC_ACID.TMS___1)-mean(Xdiff2$NICOTINIC_
   ACID.TMS___1)
mean(XdiffA$Melatonin__2TMS)-mean(Xdiff2$Melatonin__2TMS)
mean(XdiffA$ISOCITRIC_ACID.4TMS___1__M.z245)-mean(Xdiff2$
   ISOCITRIC_ACID.4TMS___1__M.z245)
mean(XdiffA$MALIC_ACID.3TMS___1__M.z233)-mean(Xdiff2$MALIC_
   ACID.3TMS___1__M.z233)
mean(XdiffA$TRYPTOPHAN_N_N_O.TMS___2__M.z202)-mean(Xdiff2$
   TRYPTOPHAN_N_N_O.TMS___2__M.z202)
mean(XdiffA$PYRUVIC_ACID.MEOX_TMS___1__M.z89)-mean(Xdiff2$
   PYRUVIC_ACID.MEOX_TMS___1__M.z89)
mean(XdiffA$Tryptophan__5.hydroxy.__3TMS__BP)-mean(Xdiff2$
    Tryptophan__5.hydroxy.__3TMS__BP)
mean(XdiffA$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.TMS_1___2__M
    .z218)-mean(Xdiff2$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.
   TMS_1___2__M.z218)
mean(XdiffA$CYSTEINE_N_O_S.TMS___1__M.z219)-mean(Xdiff2$
   CYSTEINE_N_O_S.TMS___1__M.z219)

#metabolites in RF model not in PLS model
mean(XdiffA$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS___1 )-
   mean(Xdiff2$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS___1
   )
mean(XdiffA$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.TMS_1___2__M
    .z218 )-mean(Xdiff2$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.
   TMS_1___2__M.z218 )


#t-tests for boxplot metabolites
t.test(XdiffA2$Phosphoric_acid__3TMS_~YdiffA2)
t.test(XdiffA2$PHOSPHATE.FRAGMENT___1__M.z299~YdiffA2)
t.test(XdiffA2$NICOTINIC_ACID.TMS___1~YdiffA2)
t.test(XdiffA2$Melatonin__2TMS_~YdiffA2)
t.test(XdiffA2$ISOCITRIC_ACID.4TMS___1__M.z245~YdiffA2)
t.test(XdiffA2$MALIC_ACID.3TMS___1__M.z233~YdiffA2)
```

```
#creating boxplots
par(mfrow=c(2,3))
boxplot(XdiffA2$Phosphoric_acid___3TMS_~YdiffA2, main='
    Phosphroic␣Acid', xaxt='n', ylab='Relative␣concentration'
    )
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Arterial'))
boxplot(XdiffA2$PHOSPHATE.FRAGMENT____1__M.z299~YdiffA2, main
    ='Phosphate␣fragment', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Arterial'))
boxplot(XdiffA2$NICOTINIC_ACID.TMS____1~YdiffA2, main='
    Nicotinic␣Acid', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Arterial'))
boxplot(XdiffA2$Melatonin___2TMS_~YdiffA2, main='Melatonin',
    xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Arterial'))
boxplot(XdiffA2$ISOCITRIC_ACID.4TMS____1__M.z245~YdiffA2,
    main='Isocitric␣Acid', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Arterial'))
boxplot(XdiffA2$MALIC_ACID.3TMS____1__M.z233~YdiffA2, main='
    Malic␣Acid', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Arterial'))


#V-1
ModeldiffRFV1$auc
ModeldiffV1$auc
ModeldiffRFV1$miss
ModeldiffV1$miss

pPerm(ModeldiffRFV1$auc[1,2], permFitnessdiffRFV1min)
pPerm(ModeldiffRFV1$auc[2,2], permFitnessdiffRFV1mid)
pPerm(ModeldiffRFV1$auc[3,2], permFitnessdiffRFV1max)

pPerm(ModeldiffV1$auc[1,2], permFitnessdiffV1min)
pPerm(ModeldiffV1$auc[2,2], permFitnessdiffV1mid)
pPerm(ModeldiffV1$auc[3,2], permFitnessdiffV1max)

minRFV1=getVIP(ModeldiffRFV1, model='min')
```

```
midRFV1=getVIP(ModeldiffRFV1, model='mid')
maxRFV1=getVIP(ModeldiffRFV1, model='max')

minV1=getVIP(ModeldiffV1, model='min')
midV1=getVIP(ModeldiffV1, model='mid')
maxV1=getVIP(ModeldiffV1, model='max')

nrow(minV1) #checking length of matrix
nrow(midV1)
nrow(maxV1)
tV1=cbind.data.frame(c(head(minV1$name, 10), rep(NA,5)), c(
    head(minV1$rank, 10),rep(NA,5)), head(midV1$name, 10),
    head(midV1$rank, 10), head(maxV1$name, 10), head(maxV1$
    rank, 10))
minV1$name
nrow(minRFV1)
nrow(midRFV1)
nrow(maxRFV1)
tV1RF=cbind.data.frame(c(head(minRFV1$name, 10), rep(NA,5)),
     c(head(minRFV1$rank, 10), rep(NA,5)), c(head(midRFV1$
    name, 10), rep(NA,2)), c(head(midRFV1$rank, 10), rep(NA
    ,2)), head(maxRFV1$name, 10), head(maxRFV1$rank, 10))
head(minRFV1,10)
head(midRFV1,10)
head(maxRFV1,10)
write.table(tV1, "PLStableV1", sep=";", row.names=F, quote=F
    )
write.table(tV1RF, "RFtableV1", sep=";", row.names=F, quote=
    F)

#checking which of the top 10 metabolite concentrations are
    highest in which sample type
#PLS-DA
mean(XdiffV$alpha.Tocopherol)-mean(Xdiff1$alpha.Tocopherol)
mean(XdiffV$GLYCYLVALINE.4TMS___1__M.z174)-mean(Xdiff1$
    GLYCYLVALINE.4TMS___1__M.z174)
mean(XdiffV$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2___2)-mean(
    Xdiff1$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2___2)
mean(XdiffV$LYSINE_N_N_O.TMS_3___1__M.z200)-mean(Xdiff1$
    LYSINE_N_N_O.TMS_3___1__M.z200)
mean(XdiffV$OCTADECADIENOIC_ACID__9_12._Z_Z_.TMS___1__M.z337
    )-mean(Xdiff1$OCTADECADIENOIC_ACID__9_12._Z_Z_.TMS___1__M
    .z337)
mean(XdiffV$TRYPTOPHAN_N_N_O.TMS___2__M.z202)-mean(Xdiff1$
    TRYPTOPHAN_N_N_O.TMS___2__M.z202)
mean(XdiffV$OCTADECENOIC_ACID__9._E_.TMS___1__M.z137)-mean(
    Xdiff1$OCTADECENOIC_ACID___9._E_.TMS____1__M.z137)
mean(XdiffV$OCTADECENOIC_ACID_.9._Z_.TMS____1__M.z124)-mean(
    Xdiff1$OCTADECENOIC_ACID_.9._Z_.TMS____1__M.z124)
mean(XdiffV$ABSCISIC_ACID.MEOX_2TMS_1____1)-mean(Xdiff1$
    ABSCISIC_ACID.MEOX_2TMS_1____1)
mean(XdiffV$ABSCISIC_ACID.MEOX_2TMS_2____2)-mean(Xdiff1$
    ABSCISIC_ACID.MEOX_2TMS_2____2)
mean(XdiffV$OCTADECATRIENOIC_ACID___6_9_12._Z_Z_Z_.TMS____1__M
    .z120)-mean(Xdiff1$OCTADECATRIENOIC_ACID___6_9_12._Z_Z_Z_.
    TMS____1__M.z120)

#metabolites found in RF models that are not in PLS-DA
mean(XdiffV$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.TMS_1____2__M
    .z218)-mean(Xdiff1$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.
    TMS_1____2__M.z218)


#t-tests for boxplot metabolites
t.test(XdiffV1$GLYCYLVALINE.4TMS____1__M.z174~YdiffV1)
t.test(XdiffV1$LYSINE_N_N_O.TMS_3____1__M.z200~YdiffV1)
t.test(XdiffV1$alpha.Tocopherol~YdiffV1)
t.test(XdiffV1$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2____2~
    YdiffV1)
t.test(XdiffV1$OCTADECADIENOIC_ACID___9_12._Z_Z_.TMS____1__M.
    z337~YdiffV1)
t.test(XdiffV1$ABSCISIC_ACID.MEOX_2TMS_1____1~YdiffV1)

#Create boxplots
par(mfrow=c(2,3))
boxplot(XdiffV1$GLYCYLVALINE.4TMS____1__M.z174~YdiffV1, main=
    'Glycylvaline', xaxt='n', ylab='Relative concentration')
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Venous'))
boxplot(XdiffV1$LYSINE_N_N_O.TMS_3____1__M.z200~YdiffV1, main
    ='Lysine', xaxt='n', ylab='Relative concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Venous', 'Mothers'))
boxplot(XdiffV1$alpha.Tocopherol~YdiffV1, main=expression(
    bold(paste(alpha, '-Tocopherol', sep=''))), xaxt='n', ylab
    ='Relative concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Venous'))
boxplot(XdiffV1$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2____2~
    YdiffV1, main='Fructose 1,6 disphosphate', xaxt='n', ylab
    ='Relative concentration')
legend("topright", legend='d', bty='n', cex=1.5)
```

```
axis(side=1, at=c(1,2), labels=c('Mothers', 'Venous'))
boxplot(XdiffV1$OCTADECADIENOIC_ACID__9_12._Z_Z_.TMS___1__M.
   z337~YdiffV1, main='Octadecadienoic␣Acid', xaxt='n', ylab
   ='Relative␣concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Venous'))
boxplot(XdiffV1$ABSCISIC_ACID.MEOX_2TMS_1___1~YdiffV1, main=
   'Abscisic␣Acid', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Venous'))


#V-2
ModeldiffRFV2$auc
ModeldiffV2$auc
ModeldiffRFV2$miss
ModeldiffV2$miss


pPerm(ModeldiffRFV2$auc[1,2], permFitnessdiffRFV2min)
pPerm(ModeldiffRFV2$auc[2,2], permFitnessdiffRFV2mid)
pPerm(ModeldiffRFV2$auc[3,2], permFitnessdiffRFV2max)

pPerm(ModeldiffV2$auc[1,2], permFitnessdiffV2min)
pPerm(ModeldiffV2$auc[2,2], permFitnessdiffV2mid)
pPerm(ModeldiffV2$auc[3,2], permFitnessdiffV2max)

minRFV2=getVIP(ModeldiffRFV2, model='min')
midRFV2=getVIP(ModeldiffRFV2, model='mid')
maxRFV2=getVIP(ModeldiffRFV2, model='max')

minV2=getVIP(ModeldiffV2, model='min')
midV2=getVIP(ModeldiffV2, model='mid')
maxV2=getVIP(ModeldiffV2, model='max')

nrow(minV2) #checking length of matrix
nrow(midV2)
nrow(maxV2)
tV2=cbind.data.frame(c(head(minV2$name, 10), rep(NA,5)), c(
   head(minV2$rank, 10),rep(NA,5)), head(midV2$name, 10),
   head(midV2$rank, 10), head(maxV2$name, 10), head(maxV2$
   rank, 10))
minV2$name
nrow(minRFV2)
nrow(midRFV2)
nrow(maxRFV2)
```

```
tV2RF=cbind.data.frame(c(head(minRFV2$name, 10), NA), c(head
    (minRFV2$rank, 10), NA), c(head(midRFV2$name, 10), NA), c
    (head(midRFV2$rank, 10), NA), head(maxRFV2$name, 10),
    head(maxRFV2$rank, 10))
minRFV2
midRFV2
head(maxRFV2,10)
write.table(tV2, "PLStableV2", sep=";", row.names=F, quote=F
    )
write.table(tV2RF, "RFtableV2", sep=";", row.names=F, quote=
    F)


#checking which of the top 10 metabolite concentrations are
    highest in which sample type
#PLS-DA
mean(XdiffV$Phosphoric_acid___3TMS)-mean(Xdiff2$Phosphoric_
    acid___3TMS)
mean(XdiffV$PHOSPHATE.FRAGMENT____1__M.z299)-mean(Xdiff2$
    PHOSPHATE.FRAGMENT____1__M.z299)
mean(XdiffV$PYRUVIC_ACID.MEOX_TMS____1__M.z89)-mean(Xdiff2$
    PYRUVIC_ACID.MEOX_TMS____1__M.z89)
mean(XdiffV$NICOTINIC_ACID.TMS____1)-mean(Xdiff2$NICOTINIC_
    ACID.TMS____1)
mean(XdiffV$ISOCITRIC_ACID.4TMS____1__M.z245)-mean(Xdiff2$
    ISOCITRIC_ACID.4TMS____1__M.z245)
mean(XdiffV$MALIC_ACID.3TMS____1__M.z233)-mean(Xdiff2$MALIC_
    ACID.3TMS____1__M.z233)
mean(XdiffV$TRYPTOPHAN_N_N_O.TMS____2__M.z202)-mean(Xdiff2$
    TRYPTOPHAN_N_N_O.TMS____2__M.z202)
mean(XdiffV$ALLOTHREONINE_N_O_O.TMS____1__M.z133)-mean(Xdiff2
    $ALLOTHREONINE_N_O_O.TMS____1__M.z133)
mean(XdiffV$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.TMS_1____2__M
    .z218)-mean(Xdiff2$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.
    TMS_1____2__M.z218)
mean(XdiffV$alpha.Tocopherol)-mean(Xdiff2$alpha.Tocopherol)

#RF metabolites not top 10 in PLS-DA
mean(XdiffV$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1)-
    mean(Xdiff2$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1)
mean(XdiffV$myo.Inositol)-mean(Xdiff2$myo.Inositol)
mean(XdiffV$INOSITOL___scyllo.6TMS____1__M.z432)-mean(Xdiff2$
    INOSITOL___scyllo.6TMS____1__M.z432)
mean(XdiffV$X1_5.ANHYDRO.D.GLUCITOL.4TMS____1__M.z259)-mean(
    Xdiff2$X1_5.ANHYDRO.D.GLUCITOL.4TMS____1__M.z259)
```

```
#t-tests for boxplot metabolites
t.test(XdiffV2$Phosphoric_acid__3TMS_~YdiffV2)
t.test(XdiffV2$PHOSPHATE.FRAGMENT___1__M.z299~YdiffV2)
t.test(XdiffV2$PYRUVIC_ACID.MEOX_TMS___1__M.z89~YdiffV2)
t.test(XdiffV2$NICOTINIC_ACID.TMS___1~YdiffV2)
t.test(XdiffV2$ISOCITRIC_ACID.4TMS___1__M.z245~YdiffV2)
t.test(XdiffV2$MALIC_ACID.3TMS___1__M.z233~YdiffV2)


#creating boxplots
par(mfrow=c(2,3))
boxplot(XdiffV2$Phosphoric_acid__3TMS_~YdiffV2, main='
    Phosphoric Acid', xaxt='n', ylab='Relative concentration'
    )
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Venous'))
boxplot(XdiffV2$PHOSPHATE.FRAGMENT___1__M.z299~YdiffV2, main
    ='Phosphate fragment', xaxt='n', ylab='Relative 
    concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Venous'))
boxplot(XdiffV2$PYRUVIC_ACID.MEOX_TMS___1__M.z89~YdiffV2,
    main='Pyruvic Acid', xaxt='n', ylab='Relative 
    concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Venous'))
boxplot(XdiffV2$NICOTINIC_ACID.TMS___1~YdiffV2, main='
    Nicotinic Acid', xaxt='n', ylab='Relative concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Venous'))
boxplot(XdiffV2$ISOCITRIC_ACID.4TMS___1__M.z245~YdiffV2,
    main='Isocitric Acid', xaxt='n', ylab='Relative 
    concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Venous'))
boxplot(XdiffV2$MALIC_ACID.3TMS___1__M.z233~YdiffV2, main='
    Malic Acid', xaxt='n', ylab='Relative concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Venous'))
#M-1
ModeldiffRFM1$auc
ModeldiffM1$auc
ModeldiffRFM1$miss
ModeldiffM1$miss
```

```
pPerm(ModeldiffRFM1$auc[1,2], permFitnessdiffRFM1min)
pPerm(ModeldiffRFM1$auc[2,2], permFitnessdiffRFM1mid)
pPerm(ModeldiffRFM1$auc[3,2], permFitnessdiffRFM1max)

pPerm(ModeldiffM1$auc[1,2], permFitnessdiffM1min)
pPerm(ModeldiffM1$auc[2,2], permFitnessdiffM1mid)
pPerm(ModeldiffM1$auc[3,2], permFitnessdiffM1max)

minRFM1=getVIP(ModeldiffRFM1, model='min')
midRFM1=getVIP(ModeldiffRFM1, model='mid')
maxRFM1=getVIP(ModeldiffRFM1, model='max')

minM1=getVIP(ModeldiffM1, model='min')
midM1=getVIP(ModeldiffM1, model='mid')
maxM1=getVIP(ModeldiffM1, model='max')

nrow(minM1) #checking length of matrix
nrow(midM1)
nrow(maxM1)
tM1=cbind.data.frame(head(minM1$name, 10), head(minM1$rank,
    10), head(midM1$name, 10), head(midM1$rank, 10), head(
    maxM1$name, 10), head(maxM1$rank, 10))
minM1$name
nrow(minRFM1)
nrow(midRFM1)
nrow(maxRFM1)
tM1RF=cbind.data.frame(head(minRFM1$name, 10), head(minRFM1$
    rank, 10), head(midRFM1$name, 10), head(midRFM1$rank, 10)
    , head(maxRFM1$name, 10), head(maxRFM1$rank, 10))
minRFM1
head(midRFM1,10)
head(maxRFM1, 10)
write.table(tM1, "PLStableM1", sep=";", row.names=F, quote=F
    )
write.table(tM1RF, "RFtableM1", sep=";", row.names=F, quote=
    F)


#checking which of the top 10 metabolite concentrations are
    highest in which sample type
#PLS-DA
mean(XdiffM$GLYCYLVALINE.4TMS____1__M.z174)-mean(Xdiff1$
    GLYCYLVALINE.4TMS____1__M.z174)
mean(XdiffM$ETHANOLAMINE_N_N_O.TMS____2__M.z174)-mean(Xdiff1$
    ETHANOLAMINE_N_N_O.TMS____2__M.z174)
mean(XdiffM$LYSINE_N_N_O.TMS_3____1__M.z200)-mean(Xdiff1$
    LYSINE_N_N_O.TMS_3___1__M.z200)
mean(XdiffM$LYSINE_N_N_N_O.TMS_2___1__M.z317)-mean(Xdiff1$
    LYSINE_N_N_N_O.TMS_2___1__M.z317)
mean(XdiffM$GLYCINE_N_N_O.TMS___1__M.z248)-mean(Xdiff1$
    GLYCINE_N_N_O.TMS___1__M.z248)
mean(XdiffM$PHENYLALANINE_N_O.TMS___1__M.z218)-mean(Xdiff1$
    PHENYLALANINE_N_O.TMS___1__M.z218)
mean(XdiffM$X1_5.DIAMINOPENTANE__N_N_N_N.TMS___2__M.z100)-
    mean(Xdiff1$X1_5.DIAMINOPENTANE__N_N_N_N.TMS___2__M.z100)
mean(XdiffM$TRYPTOPHAN_N_N_O.TMS___2__M.z202)-mean(Xdiff1$
    TRYPTOPHAN_N_N_O.TMS___2__M.z202)
mean(XdiffM$alpha.Tocopherol)-mean(Xdiff1$alpha.Tocopherol)
mean(XdiffM$ORNITHINE_N_N_N_O.TMS_2___1__M.z216)-mean(Xdiff1
    $ORNITHINE_N_N_N_O.TMS_2___1__M.z216)

#RF metabolites not present in top 10 PLS-DA
mean(XdiffM$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.TMS_1___2__M
    .z218)-mean(Xdiff1$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.
    TMS_1___2__M.z218)
mean(XdiffM$OCTADECENOIC_ACID___9._E_.TMS___1__M.z137)-mean(
    Xdiff1$OCTADECENOIC_ACID___9._E_.TMS___1__M.z137)
mean(XdiffM$OCTADECADIENOIC_ACID___9_12._Z_Z_.TMS___1__M.z337
    )-mean(Xdiff1$OCTADECADIENOIC_ACID___9_12._Z_Z_.TMS___1__M
    .z337)

#t-tests for boxplot metabolites
t.test(XdiffM1$GLYCYLVALINE.4TMS___1__M.z174~YdiffM1)
t.test(XdiffM1$ETHANOLAMINE_N_N_O.TMS___2__M.z174~YdiffM1)
t.test(XdiffM1$LYSINE_N_N_O.TMS_3___1__M.z200~YdiffM1)
t.test(XdiffM1$LYSINE_N_N_N_O.TMS_2___1__M.z317~YdiffM1)
t.test(XdiffM1$GLYCINE_N_N_O.TMS___1__M.z248~YdiffM1)
t.test(XdiffM1$PHENYLALANINE_N_O.TMS___1__M.z218~YdiffM1)


#creating boxplots
par(mfrow=c(2,3))
boxplot(XdiffM1$GLYCYLVALINE.4TMS___1__M.z174~YdiffM1, main=
    'Glycylvaline', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Mixed'))
boxplot(XdiffM1$ETHANOLAMINE_N_N_O.TMS___2__M.z174~YdiffM1,
    main='Ethanolamine', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Mixed'))
boxplot(XdiffM1$LYSINE_N_N_O.TMS_3___1__M.z200~YdiffM1, main
    ='Lysine␣TMS␣3', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Mixed'))
boxplot(XdiffM1$LYSINE_N_N_N_O.TMS_2____1__M.z317~YdiffM1,
    main='Lysine␣TMS␣2', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Mixed'))
boxplot(XdiffM1$GLYCINE_N_N_O.TMS____1__M.z248~YdiffM1, main=
    'Glycine', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Mixed'))
boxplot(XdiffM1$PHENYLALANINE_N_O.TMS____1__M.z218~YdiffM1,
    main='Phenylalanine', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Mothers', 'Mixed'))

#M-2
ModeldiffRFM2$auc
ModeldiffM2$auc
ModeldiffRFM2$miss
ModeldiffM2$miss


pPerm(ModeldiffRFM2$auc[1,2], permFitnessdiffRFM2min)
pPerm(ModeldiffRFM2$auc[2,2], permFitnessdiffRFM2mid)
pPerm(ModeldiffRFM2$auc[3,2], permFitnessdiffRFM2max)

pPerm(ModeldiffM2$auc[1,2], permFitnessdiffM2min)
pPerm(ModeldiffM2$auc[2,2], permFitnessdiffM2mid)
pPerm(ModeldiffM2$auc[3,2], permFitnessdiffM2max)

minRFM2=getVIP(ModeldiffRFM2, model='min')
midRFM2=getVIP(ModeldiffRFM2, model='mid')
maxRFM2=getVIP(ModeldiffRFM2, model='max')

minM2=getVIP(ModeldiffM2, model='min')
midM2=getVIP(ModeldiffM2, model='mid')
maxM2=getVIP(ModeldiffM2, model='max')

nrow(minM2) #checking length of matrix
nrow(midM2)
nrow(maxM2)
tM2=cbind.data.frame(c(head(minM2$name, 10), rep(NA,4)), c(
    head(minM2$rank, 10),rep(NA,4)), head(midM2$name, 10),
    head(midM2$rank, 10), head(maxM2$name, 10), head(maxM2$
    rank, 10))
minM2$name
nrow(minRFM2)
nrow(midRFM2)
nrow(maxRFM2)
tM2RF=cbind.data.frame(c(head(minRFM2$name, 10), rep(NA,8)),
     c(head(minRFM2$rank, 10), rep(NA,8)), c(head(midRFM2$
    name, 10), rep(NA,3)), c(head(midRFM2$rank, 10), rep(NA
    ,3)), head(maxRFM2$name, 10), head(maxRFM2$rank, 10))
minRFM2
midRFM2
head(maxRFM2, 10)
write.table(tM2, "PLStableM2", sep=";", row.names=F, quote=F
    )
write.table(tM2RF, "RFtableM2", sep=";", row.names=F, quote=
    F)

#checking which of the top 10 metabolite concentrations are
    highest in which sample type
#PLS-DA
mean(XdiffM$ISOCITRIC_ACID.4TMS____1__M.z245)-mean(Xdiff2$
    ISOCITRIC_ACID.4TMS____1__M.z245)
mean(XdiffM$Phosphoric_acid__3TMS)-mean(Xdiff2$Phosphoric_
    acid__3TMS)
mean(XdiffM$PHOSPHATE.FRAGMENT____1__M.z299)-mean(Xdiff2$
    PHOSPHATE.FRAGMENT____1__M.z299)
mean(XdiffM$PYRUVIC_ACID.MEOX_TMS____1__M.z89)-mean(Xdiff2$
    PYRUVIC_ACID.MEOX_TMS____1__M.z89)
mean(XdiffM$N.ACETYL_MANNOSAMINE.MEOX_N_O_O_O_O.TMS_1____1__M
    .z319)-mean(Xdiff2$N.ACETYL_MANNOSAMINE.MEOX_N_O_O_O_O.
    TMS_1____1__M.z319)
mean(XdiffM$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1)-
    mean(Xdiff2$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1)
mean(XdiffM$TRYPTOPHAN_N_N_O.TMS____2__M.z202)-mean(Xdiff2$
    TRYPTOPHAN_N_N_O.TMS____2__M.z202)
mean(XdiffM$LYSINE_N_N_N_O.TMS_2____1__M.z317)-mean(Xdiff2$
    LYSINE_N_N_N_O.TMS_2____1__M.z317)
mean(XdiffM$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.TMS_1____2__M
    .z218)-mean(Xdiff2$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.
    TMS_1____2__M.z218)
mean(XdiffM$Cic.aconitic_acid)-mean(Xdiff2$Cic.aconitic_acid
    )
mean(XdiffM$NICOTINIC_ACID.TMS____1)-mean(Xdiff2$NICOTINIC_
    ACID.TMS____1)
mean(XdiffM$GLYCYLVALINE.4TMS____1__M.z174)-mean(Xdiff2$
    GLYCYLVALINE.4TMS____1__M.z174)

#RF metabolites that were not in PLS−DA top 10s
mean(XdiffM$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.TMS_2____1__M
    .z191)−mean(Xdiff2$N.ACETYL_GLUCOSAMINE.MEOX_N_O_O_O_O.
    TMS_2____1__M.z191)

#t−tests for boxplot metabolites
t.test(XdiffM2$Phosphoric_acid___3TMS_~YdiffM2)
t.test(XdiffM2$PHOSPHATE.FRAGMENT____1__M.z299~YdiffM2)
t.test(XdiffM2$ISOCITRIC_ACID.4TMS____1__M.z245~YdiffM2)
t.test(XdiffM2$PYRUVIC_ACID.MEOX_TMS____1__M.z89~YdiffM2)
t.test(XdiffM2$N.ACETYL_MANNOSAMINE.MEOX_N_O_O_O_O.TMS_1____1
    __M.z319~YdiffM2)
t.test(XdiffM2$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1~
    YdiffM2)


#create boxplots
par(mfrow=c(2,3))
boxplot(XdiffM2$Phosphoric_acid___3TMS_~YdiffM2, main='
    Phosphoric␣Acid', xaxt='n', ylab='Relative␣concentration'
    )
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Mixed'))
boxplot(XdiffM2$PHOSPHATE.FRAGMENT____1__M.z299~YdiffM2, main
    ='Phosphate␣fragment', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Mixed'))
boxplot(XdiffM2$ISOCITRIC_ACID.4TMS____1__M.z245~YdiffM2,
    main='Isocitric␣Acid', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Mixed'))
boxplot(XdiffM2$PYRUVIC_ACID.MEOX_TMS____1__M.z89~YdiffM2,
    main='Pyruvic␣Acid', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Mixed'))
boxplot(XdiffM2$N.ACETYL_MANNOSAMINE.MEOX_N_O_O_O_O.TMS_1____
    1__M.z319~YdiffM2, main='N−Acetyl␣Mannosamine', xaxt='n',
     ylab='Relative␣concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Mixed'))
boxplot(XdiffM2$GAMMA.GLUTAMYLPHENYLALANINE__N_N_O_O.TMS____1
    ~YdiffM2 , main=expression(bold(paste(gamma, '-
    Glutamylphenylalanine'))), xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Fathers', 'Mixed'))

#ANOVA decomposition
#start off by making sure only whole families are in the
    data
nrow(Xdiff1)
nrow(Xdiff2)
nrow(XdiffA)
nrow(XdiffV)
nrow(XdiffM)
metadiff1$Family_number
metadiff2$Family_number
metadiffA$Family_number
metadiffV$Family_number
metadiffM$Family_number
#All identical except for number 31 which is not in
    metadiff2 due to earlier outlier removal

#Removing 31 from every matrix.
metaAD1 <- metadiff1[-31,]
XAD1 <- Xdiff1[-31,]

metaADA <- metadiffA[-31,]
XADA <- XdiffA[-31,]

metaADV <- metadiffV[-31,]
XADV <- XdiffV[-31,]

metaADM <- metadiffM[-31,]
XADM <- XdiffM[-31,]

metaAD2 <- metadiff2
XAD2 <- Xdiff2


#rename A, M and V to numbers
metaAD1$Sample[1:nrow(metaAD1)]=1
metaAD2$Sample[1:nrow(metaAD2)]=2
metaADA$Sample[1:nrow(metaADA)]=3
metaADV$Sample[1:nrow(metaADV)]=4
metaADM$Sample[1:nrow(metaADM)]=5
#Merge matrices in same order
```

```
XAD <- as.matrix(rbind(XAD1, XAD2, XADA, XADV, XADM))
metaAD <- rbind(metaAD1, metaAD2, metaADA, metaADV, metaADM)
levels=metaAD[,c(5,2,4)] #defines the levels for which ANOVA
    decomposition will be made (Batch, Family number, Sample
   )
str(levels) #check if everything is in order

#Do AnDec analysis with proper permutations

#PLS
XAnDec=AnDec(XAD, levels, scale='yes')
XDecdata=XAnDec$factorMatrix[[length(levels)]]+XAnDec$
   residuals

permFitnessADmin=numeric(nPerm)
permFitnessADmid=numeric(nPerm)
permFitnessADmax=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ADMUVRmodel=MUVR(X=XDecdata, Y=metaAD$Sample, ID=metaAD$
   Family_number, fitness='MISS', nRep=20, nOuter=6,
   varRatio=0.9, method='PLS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(metaAD$Sample)
  permModelAD=MUVR(X=XDecdata, Y=Yperm, varRatio=0.9, ID=
     metaAD$Family_number, fitness='MISS', nRep=20, nOuter
     =6, method='PLS', DA=T)
  permFitnessADmin[p]=permModelAD$miss[1]
  permFitnessADmid[p]=permModelAD$miss[2]
  permFitnessADmax[p]=permModelAD$miss[3]
}
stopCluster(cl)

#RF
permFitnessADRFmin=numeric(nPerm)
permFitnessADRFmid=numeric(nPerm)
permFitnessADRFmax=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
ADMUVRRFmodel=MUVR(X=XDecdata, Y=metaAD$Sample, varRatio
   =0.9, ID=metaAD$Family_number, fitness='MISS', nRep=20,
   nOuter=6, method='RF')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(metaAD$Sample)
  permModelADRF=MUVR(X=XDecdata, Y=Yperm, varRatio=0.9, ID=
     metaAD$Family_number, fitness='MISS', nRep=20, nOuter
     =6, method='RF')
  permFitnessADRFmin[p]=permModelADRF$miss[1]
  permFitnessADRFmid[p]=permModelADRF$miss[2]
  permFitnessADRFmax[p]=permModelADRF$miss[3]
}
stopCluster(cl)


plotPerm(ADMUVRmodel$miss[1], permFitnessADmin)
plotPerm(ADMUVRmodel$miss[2], permFitnessADmid)
plotPerm(ADMUVRmodel$miss[3], permFitnessADmax)

plotPerm(ADMUVRRFmodel$miss[1], permFitnessADRFmin)
plotPerm(ADMUVRRFmodel$miss[2], permFitnessADRFmid)
plotPerm(ADMUVRRFmodel$miss[3], permFitnessADRFmax)

nrow(getVIP(ADMUVRmodel, model='min'))
nrow(getVIP(ADMUVRmodel, model='mid'))
nrow(getVIP(ADMUVRmodel, model='max'))

nrow(getVIP(ADMUVRRFmodel, model='min'))
nrow(getVIP(ADMUVRRFmodel, model='mid'))
nrow(getVIP(ADMUVRRFmodel, model='max'))

minAN=getVIP(ADMUVRmodel, model='min')
midAN=getVIP(ADMUVRmodel, model='mid')
maxAN=getVIP(ADMUVRmodel, model='max')

minANRF=getVIP(ADMUVRRFmodel, model='min')
midANRF=getVIP(ADMUVRRFmodel, model='mid')
maxANRF=getVIP(ADMUVRRFmodel, model='max')

tAN=cbind.data.frame(head(minAN$name, 10), head(minAN$rank,
    10), head(midAN$name, 10), head(midAN$rank, 10), head(
    maxAN$name, 10), head(maxAN$rank, 10))
tANRF=cbind.data.frame(head(minANRF$name, 10),head(minANRF$
    rank, 10), head(midANRF$name, 10), head(midANRF$rank, 10)
    , head(maxANRF$name, 10), head(maxANRF$rank, 10))
write.table(tAN, "PLStableAN", sep=";", row.names=F, quote=F
    )
write.table(tANRF, "RFtableAN", sep=";", row.names=F, quote=
    F)

#create boxplots for ANOVA decompositioned calculations
```

```
XADd=as.data.frame(XAD) #making XAD into data frame to call
    on columns using $
par(mfrow=c(2,2))
boxplot(XADd$PHOSPHATE.FRAGMENT____1__M.z299~metaAD$Sample,
    main='Phosphate␣fragment', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='a', bty='n', cex=1.0)
axis(side=1, at=c(1,2,3,4,5), labels=c('Mo', 'F', 'A', 'V',
    'Mi'))
boxplot(XADd$ABSCISIC_ACID.MEOX_2TMS_2____2~metaAD$Sample,
    main='Abscisic␣Acid', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='b', bty='n', cex=1.0)
axis(side=1, at=c(1,2,3,4,5), labels=c('Mo', 'F', 'A', 'V',
    'Mi'))
boxplot(XADd$FRUCTOSE.1_6.DISPHOSPHATE.MEOX_7TMS_2____2~
    metaAD$Sample, main='Fructose␣1,6-disphosphate', xaxt='n'
    , ylab='Relative␣concentration')
legend("topright", legend='c', bty='n', cex=1.0)
axis(side=1, at=c(1,2,3,4,5), labels=c('Mo', 'F', 'A', 'V',
    'Mi'))
boxplot(XADd$Putrescine__N.acetyl.__2TMS_~metaAD$Sample,
    main='Putrescine', xaxt='n', ylab='Relative␣concentration
    ')
legend("topright", legend='d', bty='n', cex=1.0)
axis(side=1, at=c(1,2,3,4,5), labels=c('Mo', 'F', 'A', 'V',
    'Mi'))


#ML analysis of venous and arterial
XAVEM=XADA-XADV #creating effect matrix


permFitnessAVEMmin=numeric(nPerm)
permFitnessAVEMmid=numeric(nPerm)
permFitnessAVEMmax=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
XmodelAVEM=MUVR(X=XAVEM,ML=TRUE,nRep=6,nOuter=5,varRatio
    =0.9,method='PLS', fitness='MISS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(c(-1,1), size=nrow(XAVEM), replace=TRUE)
  permModelAVEM=MUVR(X=XAVEM,ML=TRUE,Y=Yperm, varRatio=0.9,
      nRep=6, nOuter=5, method='PLS', DA=T, fitness='MISS')
  permFitnessAVEMmin[p]=permModelAVEM$miss[1]
  permFitnessAVEMmid[p]=permModelAVEM$miss[2]
  permFitnessAVEMmax[p]=permModelAVEM$miss[3]
}
stopCluster(cl)

permFitnessAVEMRFmin=numeric(nPerm)
permFitnessAVEMRFmid=numeric(nPerm)
permFitnessAVEMRFmax=numeric(nPerm)
cl=makeCluster(nCore)
registerDoParallel(cl)
XmodelAVEMRF=MUVR(X=XAVEM,ML=TRUE,nRep=6,nOuter=5,varRatio
    =0.9,method='RF', fitness='MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(c(-1,1), size=nrow(XAVEM), replace=TRUE)
  permModelAVEMRF=MUVR(X=XAVEM, Y=Yperm,ML=TRUE, varRatio
      =0.9, nRep=6, nOuter=5, method='RF', fitness='MISS')
  permFitnessAVEMRFmin[p]=permModelAVEMRF$miss[1]
  permFitnessAVEMRFmid[p]=permModelAVEMRF$miss[2]
  permFitnessAVEMRFmax[p]=permModelAVEMRF$miss[3]
}
stopCluster(cl)


#evaluate models
XmodelAVEMRF$miss
XmodelAVEM$miss
XmodelAVEMRF$auc
XmodelAVEM$auc

plotPerm(XmodelAVEMRF$miss[1], permFitnessAVEMRFmin)
plotPerm(XmodelAVEMRF$miss[2], permFitnessAVEMRFmid)
plotPerm(XmodelAVEMRF$miss[3], permFitnessAVEMRFmax)
plotPerm(XmodelAVEM$miss[1], permFitnessAVEMmin)
plotPerm(XmodelAVEM$miss[2], permFitnessAVEMmid)
plotPerm(XmodelAVEM$miss[3], permFitnessAVEMmax)

minAVEMRF=getVIP(XmodelAVEMRF, model='min')
midAVEMRF=getVIP(XmodelAVEMRF, model='mid')
maxAVEMRF=getVIP(XmodelAVEMRF, model='max')
minAVEM=getVIP(XmodelAVEM, model='min')
midAVEM=getVIP(XmodelAVEM, model='mid')
maxAVEM=getVIP(XmodelAVEM, model='max')

tAVEM=cbind.data.frame(head(minAVEM$name, 10), head(minAVEM$
    rank, 10), head(midAVEM$name, 10), head(midAVEM$rank, 10)
    , head(maxAVEM$name, 10), head(maxAVEM$rank, 10))
tAVEMRF=cbind.data.frame(head(minAVEMRF$name, 10), head(
    minAVEMRF$rank, 10), head(midAVEMRF$name, 10), head(
    midAVEMRF$rank, 10), head(maxAVEMRF$name, 10), head(
    maxAVEMRF$rank, 10))
write.table(tAVEM, "PLStableAVEM", sep=";", row.names=F,
    quote=F)
write.table(tAVEMRF, "RFtableAVEM", sep=";", row.names=F,
    quote=F)

nrow(minAVEMRF)
nrow(midAVEMRF)
nrow(maxAVEMRF)
nrow(minAVEM)
nrow(midAVEM)
nrow(maxAVEM)
head(minAVEMRF,10)
head(midAVEMRF,10)
head(maxAVEMRF,10)
head(minAVEM,10)
head(midAVEM,10)
head(maxAVEM,10)


#XAVEM is A-V, positive values -> higher in A, negative
    values -> higher in V
mean(XAVEM$alpha.ketoglutaric_acid)
mean(XAVEM$ALPHA.KETOGLUTARIC_ACID.MEOX_2TMS____1__M.z198)
mean(XAVEM$X2.DEOXY.GLUCOSE.MEOX_4TMS____2_)
mean(XAVEM$M000000_A181005.101.xxx_NA_1792_PRED_VAR5_ALK_NA)
mean(XAVEM$HYPOXANTHINE.2TMS____1__M.z280)
mean(XAVEM$X2.DEOXY.GALACTOSE.MEOX_4TMS_2____2)
mean(XAVEM$GLUCOSE.MEOX_5TMS_1____1)
mean(XAVEM$SORBOSE.MEOX_5TMS_1____2)
mean(XAVEM$L__.__.Glutamic_acid)
mean(XAVEM$GALACTOSE.MEOX_5TMS_1____1)
mean(XAVEM$GALACTOSE.MEOX_5TMS_2____1__M.z291)
mean(XAVEM$KDG_1_1MeOX_4TMS)
mean(XAVEM$Idose___1MEOX____5TMS__MP)
mean(XAVEM$X2.DEOXY.GALACTOSE.MEOX_4TMS_1____2__M.z143)
mean(XAVEM$MANNOSE.MEOX_5TMS_1____1)
mean(XAVEM$MANNITOL.6TMS____1)
mean(XAVEM$HOMOCYSTEINE__N_O_S.TMS____2__M.z246)
#making paired t-tests for EM
t.test(XADA$alpha.ketoglutaric_acid, XADV$alpha.ketoglutaric
    _acid, paired = TRUE, alternative='two.sided') #p=7.7e-07
```

```
t.test(XADA$X2.DEOXY.GALACTOSE.MEOX_4TMS_2____2, XADV$X2.
    DEOXY.GALACTOSE.MEOX_4TMS_2____2, paired = TRUE,
    alternative='two.sided') #p=1.1e-05
t.test(XADA$GLUCOSE.MEOX_5TMS_1____1, XADV$GLUCOSE.MEOX_5TMS_
    1____1, paired = TRUE, alternative='two.sided') #p=0.00033
t.test(XADA$SORBOSE.MEOX_5TMS_1____2, XADV$SORBOSE.MEOX_5TMS_
    1____2, paired = TRUE, alternative='two.sided') #p=0.00093
t.test(XADA$L_._.Glutamic_acid, XADV$L_._.Glutamic_acid,
    paired = TRUE, alternative='two.sided') #p=0.00037
t.test(XADA$GALACTOSE.MEOX_5TMS_1____1, XADV$GALACTOSE.MEOX_5
    TMS_1____1, paired = TRUE, alternative='two.sided') #p
    =0.00034



#create boxplots for A-V
XADAV=rbind(XADA, XADV)
metaADAV=rbind(metaADA, metaADV)
par(mfrow=c(2,3))
boxplot(XADAV$alpha.ketoglutaric_acid~metaADAV$Sample, main=
    expression(bold(paste(alpha, '-Ketoglutaric␣Acid', sep='␣'
    ))), xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Arterial', 'Venous'))
boxplot(XADAV$X2.DEOXY.GALACTOSE.MEOX_4TMS_2____2~metaADAV$
    Sample, main='Deoxy-galactose', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Arterial', 'Venous'))
boxplot(XADAV$GLUCOSE.MEOX_5TMS_1____1~metaADAV$Sample, main=
    'Glucose', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Arterial', 'Venous'))
boxplot(XADAV$SORBOSE.MEOX_5TMS_1____2~metaADAV$Sample, main=
    'Sorbose', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Arterial', 'Venous'))
boxplot(XADAV$L_._.Glutamic_acid~metaADAV$Sample, main='L-
    Glutamic␣Acid', xaxt='n', ylab='Relative␣concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Arterial', 'Venous'))
boxplot(XADAV$GALACTOSE.MEOX_5TMS_1____1~metaADAV$Sample,
    main='Galactose', xaxt='n', ylab='Relative␣concentration'
    )
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('Arterial', 'Venous'))
```

```
#PLS and RF on allergy
##Using data from Difference_mother_father_infant


nPerm=100 #100 for real investigation and 25 for "quick"
    analysis
varratio=0.9 #0.9 for real investigations and 0.75 for "
    quick" analysis
nCore=detectCores()-1

Xaller123 <- rbind(Xdiff1, Xdiff2, XdiffA, XdiffV, XdiffM)
Yaller123 <- c(Yaller1, Yaller2, YallerA, YallerV, YallerM)
metaaller123 <- rbind(metadiff1, metadiff2, metadiffA,
    metadiffV, metadiffM)

#removal of samples without allergy data

allergy_unavailible1 <- c()
for (i in 1:length(Yaller1)){
  if(is.na(Yaller1[i])){
    allergy_unavailible1 <- append(allergy_unavailible1, i)
  }
}
Xaller1<- Xdiff1[-allergy_unavailible1,]
Yaller1<- Yaller1[-allergy_unavailible1]
metaaller1<- metadiff1[-allergy_unavailible1,]

allergy_unavailible2 <- c()
for (i in 1:length(Yaller2)){
  if(is.na(Yaller2[i])){
    allergy_unavailible2 <- append(allergy_unavailible2, i)
  }
}
Xaller2<- Xdiff2[-allergy_unavailible2,]
Yaller2<- Yaller2[-allergy_unavailible2]
metaaller2<- metadiff2[-allergy_unavailible2,]

allergy_unavailibleA <- c()
for (i in 1:length(YallerA)){
  if(is.na(YallerA[i])){
    allergy_unavailibleA <- append(allergy_unavailibleA, i)
  }
}
XallerA<- XdiffA[-allergy_unavailibleA,]
```

```
YallerA<- YallerA[-allergy_unavailibleA]
metaallerA<- metadiffA[-allergy_unavailibleA ,]

allergy_unavailibleV <- c()
for (i in 1:length(YallerV)){
  if(is.na(YallerV[i])){
    allergy_unavailibleV <- append(allergy_unavailibleV, i)
  }
}
XallerV<- XdiffV[-allergy_unavailibleV ,]
YallerV<- YallerV[-allergy_unavailibleV]
metaallerV<- metadiffV[-allergy_unavailibleV ,]

allergy_unavailibleM <- c()
for (i in 1:length(YallerM)){
  if(is.na(YallerM[i])){
    allergy_unavailibleM <- append(allergy_unavailibleM, i)
  }
}
XallerM<- XdiffM[-allergy_unavailibleM ,]
YallerM<- YallerM[-allergy_unavailibleM]
metaallerM<- metadiffM[-allergy_unavailibleM ,]

#1vallergy

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerPLS1min=numeric(nPerm)
permFitnessallerPLS1mid=numeric(nPerm)
permFitnessallerPLS1max=numeric(nPerm)
ModelallerPLS1=MUVR(X=Xaller1,Y=Yaller1,varRatio = varratio,
    nRep=20, nOuter=5,method='PLS', fitness='MISS', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller1)
  permModelallerPLS1=MUVR(X=Xaller1, Y=Yperm, varRatio=
      varratio, ID=metaaller1$Family_number, nRep=20, nOuter
      =5,method='PLS', DA=T, fitness='MISS')
  permFitnessallerPLS1min[p]=permModelallerPLS1$miss[1]
  permFitnessallerPLS1mid[p]=permModelallerPLS1$miss[2]
  permFitnessallerPLS1max[p]=permModelallerPLS1$miss[3]
}
stopCluster(cl)

cl=makeCluster(nCore)
registerDoParallel(cl)
```

```
permFitnessallerauc1min=numeric(nPerm)
permFitnessallerauc1mid=numeric(nPerm)
permFitnessallerauc1max=numeric(nPerm)
Modelallerauc1=MUVR(X=Xaller1,Y=Yaller1,varRatio = varratio,
    nRep=20, nOuter=5,method='PLS', fitness='AUROC', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller1)
  permModelallerauc1=MUVR(X=Xaller1, Y=Yperm, varRatio=
      varratio, nRep=20, nOuter=5,method='PLS', fitness='
      AUROC', DA=T)
  permFitnessallerauc1min[p]=permModelallerauc1$auc[1,2]
  permFitnessallerauc1mid[p]=permModelallerauc1$auc[2,2]
  permFitnessallerauc1max[p]=permModelallerauc1$auc[3,2]
}
stopCluster(cl)

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerRMS1min=numeric(nPerm)
permFitnessallerRMS1mid=numeric(nPerm)
permFitnessallerRMS1max=numeric(nPerm)
ModelallerRMS1=MUVR(X=Xaller1,Y=Yaller1,varRatio = varratio,
    nRep=20, nOuter=5,method='PLS', fitness='RMSEP')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller1)
  permModelallerRMS1=MUVR(X=Xaller1, Y=Yperm, varRatio=
      varratio, nRep=20, nOuter=5,method='PLS', fitness='
      RMSEP')
  permFitnessallerRMS1min[p]=permModelallerRMS1$fitMetric$Q2
      [1]
  permFitnessallerRMS1mid[p]=permModelallerRMS1$fitMetric$Q2
      [2]
  permFitnessallerRMS1max[p]=permModelallerRMS1$fitMetric$Q2
      [3]
}
stopCluster(cl)

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerBER1min=numeric(nPerm)
permFitnessallerBER1mid=numeric(nPerm)
permFitnessallerBER1max=numeric(nPerm)
ModelallerBER1=MUVR(X=Xaller1,Y=Yaller1,varRatio = varratio,
    nRep=20, nOuter=5,method='PLS', fitness='BER', DA=T)
```

```
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller1)
  permModelallerBER1=MUVR(X=Xaller1, Y=Yperm, varRatio=
     varratio, nRep=20, nOuter=5,method='PLS', fitness='BER'
     , DA=T)
  permFitnessallerBER1min[p]=permModelallerBER1$miss[1]
  permFitnessallerBER1mid[p]=permModelallerBER1$miss[2]
  permFitnessallerBER1max[p]=permModelallerBER1$miss[3]
}
stopCluster(cl)



cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerRF1min=numeric(nPerm)
permFitnessallerRF1mid=numeric(nPerm)
permFitnessallerRF1max=numeric(nPerm)
ModelallerRF1=MUVR(X=Xaller1,Y=as.character(Yaller1),
   varRatio = varratio, nRep=20, nOuter=5,method='RF',
   fitness='MISS')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller1)
  permModelallerRF1=MUVR(X=Xaller1, Y=as.character(Yperm),
     varRatio=varratio, nRep=20, nOuter=5,method='RF',
     fitness='MISS')
  permFitnessallerRF1min[p]=permModelallerRF1$miss[1]
  permFitnessallerRF1mid[p]=permModelallerRF1$miss[2]
  permFitnessallerRF1max[p]=permModelallerRF1$miss[3]
}
stopCluster(cl)


cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerRFBER1min=numeric(nPerm)
permFitnessallerRFBER1mid=numeric(nPerm)
permFitnessallerRFBER1max=numeric(nPerm)
ModelallerRFBER1=MUVR(X=Xaller1,Y=as.character(Yaller1),
   varRatio = varratio, nRep=20, nOuter=5,method='RF',
   fitness='BER')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller1)
  permModelallerRFBER1=MUVR(X=Xaller1 , Y=as.character(Yperm)
     , varRatio=varratio , nRep=20, nOuter=5,method='RF' ,
     fitness='BER')
  permFitnessallerRFBER1min[p]=permModelallerRFBER1$miss[1]
  permFitnessallerRFBER1mid[p]=permModelallerRFBER1$miss[2]
  permFitnessallerRFBER1max[p]=permModelallerRFBER1$miss[3]
}
stopCluster(cl)



#2vallergy
cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerPLS2min=numeric(nPerm)
permFitnessallerPLS2mid=numeric(nPerm)
permFitnessallerPLS2max=numeric(nPerm)
ModelallerPLS2=MUVR(X=Xaller2 ,Y=Yaller2 ,varRatio = varratio ,
    nRep=20, nOuter=5,method='PLS', fitness='BER', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller2)
  permModelallerPLS2=MUVR(X=Xaller2 , Y=Yperm, varRatio=
     varratio , nRep=20, nOuter=5,method='PLS', fitness='BER'
     , DA=T)
  permFitnessallerPLS2min[p]=permModelallerPLS2$miss[1]
  permFitnessallerPLS2mid[p]=permModelallerPLS2$miss[2]
  permFitnessallerPLS2max[p]=permModelallerPLS2$miss[3]
}
stopCluster(cl)


cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerRF2min=numeric(nPerm)
permFitnessallerRF2mid=numeric(nPerm)
permFitnessallerRF2max=numeric(nPerm)
ModelallerRF2=MUVR(X=Xaller2 ,Y=as.character(Yaller2),
    varRatio = varratio , nRep=20, nOuter=5,method='RF',
    fitness='BER')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(Yaller2)
  permModelallerRF2=MUVR(X=Xaller2 , Y=as.character(Yperm),
     varRatio=varratio , nRep=20, nOuter=5,method='RF',
     fitness='BER')
  permFitnessallerRF2min[p]=permModelallerRF2$miss[1]
  permFitnessallerRF2mid[p]=permModelallerRF2$miss[2]
  permFitnessallerRF2max[p]=permModelallerRF2$miss[3]
}
stopCluster(cl)




#Avallergy

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerPLSAmin=numeric(nPerm)
permFitnessallerPLSAmid=numeric(nPerm)
permFitnessallerPLSAmax=numeric(nPerm)
ModelallerPLSA=MUVR(X=XallerA,Y=YallerA,varRatio = varratio,
    nRep=20, nOuter=5,method='PLS', fitness='BER', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YallerA)
  permModelallerPLSA=MUVR(X=XallerA, Y=Yperm, varRatio=
     varratio, nRep=20, nOuter=5,method='PLS' , fitness='BER
     ', DA=T)
  permFitnessallerPLSAmin[p]=permModelallerPLSA$miss[1]
  permFitnessallerPLSAmid[p]=permModelallerPLSA$miss[2]
  permFitnessallerPLSAmax[p]=permModelallerPLSA$miss[3]
}
stopCluster(cl)

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerRFAmin=numeric(nPerm)
permFitnessallerRFAmid=numeric(nPerm)
permFitnessallerRFAmax=numeric(nPerm)
ModelallerRFA=MUVR(X=XallerA,Y=as.character(YallerA),
   varRatio = varratio, nRep=20, nOuter=5,method='RF',
   fitness='BER')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YallerA)
  permModelallerRFA=MUVR(X=XallerA, Y=as.character(Yperm),
     varRatio=varratio, nRep=20, nOuter=5,method='RF' ,
     fitness='BER')
  permFitnessallerRFAmin[p]=permModelallerRFA$miss[1]
  permFitnessallerRFAmid[p]=permModelallerRFA$miss[2]
  permFitnessallerRFAmax[p]=permModelallerRFA$miss[3]
}
stopCluster(cl)




#Vvallergy

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerPLSVmin=numeric(nPerm)
permFitnessallerPLSVmid=numeric(nPerm)
permFitnessallerPLSVmax=numeric(nPerm)
ModelallerPLSV=MUVR(X=XallerV,Y=YallerV,varRatio = varratio,
    nRep=20, nOuter=5,method='PLS', fitness='BER', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YallerV)
  permModelallerPLSV=MUVR(X=XallerV, Y=Yperm, varRatio=
      varratio, nRep=20, nOuter=5,method='PLS', fitness='BER'
      , DA=T)
  permFitnessallerPLSVmin[p]=permModelallerPLSV$miss[1]
  permFitnessallerPLSVmid[p]=permModelallerPLSV$miss[2]
  permFitnessallerPLSVmax[p]=permModelallerPLSV$miss[3]
}
stopCluster(cl)

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerRFVmin=numeric(nPerm)
permFitnessallerRFVmid=numeric(nPerm)
permFitnessallerRFVmax=numeric(nPerm)
ModelallerRFV=MUVR(X=XallerV,Y=as.character(YallerV),
    varRatio = varratio, nRep=20, nOuter=5,method='RF',
    fitness='BER')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YallerV)
  permModelallerRFV=MUVR(X=XallerV, Y=as.character(Yperm),
      varRatio=varratio, nRep=20, nOuter=5,method='RF',
      fitness='BER')
  permFitnessallerRFVmin[p]=permModelallerRFV$miss[1]
  permFitnessallerRFVmid[p]=permModelallerRFV$miss[2]
  permFitnessallerRFVmax[p]=permModelallerRFV$miss[3]
```

```
}
stopCluster(cl)



#Mvallergy
cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerPLSMmin=numeric(nPerm)
permFitnessallerPLSMmid=numeric(nPerm)
permFitnessallerPLSMmax=numeric(nPerm)
ModelallerPLSM=MUVR(X=XallerM,Y=YallerM,varRatio = varratio,
    nRep=20, nOuter=5,method='PLS', fitness='BER', DA=T)
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YallerM)
  permModelallerPLSM=MUVR(X=XallerM, Y=Yperm, varRatio=
     varratio, nRep=20, nOuter=5,method='PLS', fitness='BER'
     , DA=T)
  permFitnessallerPLSMmin[p]=permModelallerPLSM$miss[1]
  permFitnessallerPLSMmid[p]=permModelallerPLSM$miss[2]
  permFitnessallerPLSMmax[p]=permModelallerPLSM$miss[3]
}
stopCluster(cl)

cl=makeCluster(nCore)
registerDoParallel(cl)
permFitnessallerRFMmin=numeric(nPerm)
permFitnessallerRFMmid=numeric(nPerm)
permFitnessallerRFMmax=numeric(nPerm)
ModelallerRFM=MUVR(X=XallerM,Y=as.character(YallerM),
   varRatio = varratio, nRep=20, nOuter=5,method='RF',
   fitness='BER')
for (p in 1:nPerm) {
  cat('\nPermutation',p,'of', nPerm)
  Yperm=sample(YallerM)
  permModelallerRFM=MUVR(X=XallerM, Y=as.character(Yperm),
     varRatio=varratio, nRep=20, nOuter=5,method='RF',
     fitness='BER')
  permFitnessallerRFMmin[p]=permModelallerRFM$miss[1]
  permFitnessallerRFMmid[p]=permModelallerRFM$miss[2]
  permFitnessallerRFMmax[p]=permModelallerRFM$miss[3]
}
stopCluster(cl)
```

```
#Evalute models
# 1 - allergy
ModelallerPLS1$auc
ModelallerPLS1$nComp
ModelallerPLS1$miss
ModelallerBER1$auc
ModelallerBER1$ncomp
ModelallerBER1$miss
ModelallerRF1$miss
ModelallerRF1$auc
ModelallerRFBER1$miss
ModelallerRFBER1$auc
ModelallerRF1$yPred$min
#check for gaussian shape of permutations
plotPerm(ModelallerBER1$miss[1], permFitnessallerBER1min)
plotPerm(ModelallerBER1$miss[2], permFitnessallerBER1mid)
plotPerm(ModelallerBER1$miss[3], permFitnessallerBER1max)
plotPerm(ModelallerRFBER1$miss[1], permFitnessallerRFBER1min
    )
plotPerm(ModelallerRFBER1$miss[2], permFitnessallerRFBER1mid
    )
plotPerm(ModelallerRFBER1$miss[3], permFitnessallerRFBER1max
    )
#store metabolites to variables
PLSaller1min=getVIP(ModelallerPLS1, model='min')
PLSaller1mid=getVIP(ModelallerPLS1, model='mid')
PLSaller1max=getVIP(ModelallerPLS1, model='max')
RFaller1min=getVIP(ModelallerRF1, model='min')
RFaller1mid=getVIP(ModelallerRF1, model='mid')
RFaller1max=getVIP(ModelallerRF1, model='max')

#check number of metabolites in each model
nrow(PLSaller1min)
nrow(PLSaller1mid)
nrow(PLSaller1max)
nrow(RFaller1min)
nrow(RFaller1mid)
nrow(RFaller1max)

head(PLSaller1min,10)
head(PLSaller1mid,10)
head(PLSaller1max,10)
RFaller1min
RFaller1mid
RFaller1max
```

```
#store top 10 metabolites for each model to for table
taller1PLS=cbind.data.frame(head(PLSaller1min$name, 10),
    head(PLSaller1min$rank, 10), head(PLSaller1mid$name, 10),
     head(PLSaller1mid$rank, 10), head(PLSaller1max$name, 10)
    , head(PLSaller1max$rank, 10))
taller1RF=cbind.data.frame(c(head(RFaller1min$name, 10),0),
    c(head(RFaller1min$rank, 10),0), c(head(RFaller1mid$name,
     10),0), c(head(RFaller1mid$rank, 10),0), head(
    RFaller1max$name, 10), head(RFaller1max$rank, 10))

write.table(taller1PLS, "PLStablealler1", sep=";", row.names
    =F, quote=F)
write.table(taller1RF, "RFtablealler1", sep=";", row.names=F
    , quote=F)

#check mean values to see if concentration higher in
    allergic or non-allergic
#creating two new matrices containing data on allergic and
    non-allergic respectively
Moaller=which(metaaller1$any_allergies==1)
XMoaller=Xaller1[Moaller,]
Xmonoaller=Xaller1[-Moaller,]

#substracting aller from noaller
mean(Xmonoaller$GLUTARIC_ACID.2TMS____1__M.z261)-mean(
    XMoaller$GLUTARIC_ACID.2TMS____1__M.z261)
mean(Xmonoaller$GLYCERIC_ACID.3TMS____3)-mean(XMoaller$
    GLYCERIC_ACID.3TMS____3)
mean(Xmonoaller$HOMOCYSTEINE__N_O_S.TMS____2__M.z246)-mean(
    XMoaller$HOMOCYSTEINE__N_O_S.TMS____2__M.z246)
mean(Xmonoaller$Calystegine_B2__4TMS_)-mean(XMoaller$
    Calystegine_B2__4TMS_)
mean(Xmonoaller$Unknown.sst.cgl.023)-mean(XMoaller$Unknown.
    sst.cgl.023)
mean(Xmonoaller$Nivalenol__4TMS_)-mean(XMoaller$Nivalenol__4
    TMS_)
mean(Xmonoaller$GLUTAMIC_ACID_N_O_O.TMS____1__M.z247)-mean(
    XMoaller$GLUTAMIC_ACID_N_O_O.TMS____1__M.z247)
mean(Xmonoaller$TURANOSE_MEOX_8TMS_1____1__M.z307)-mean(
    XMoaller$TURANOSE_MEOX_8TMS_1____1__M.z307)
mean(Xmonoaller$GAMMA.GLUTAMYLPHENYLALANINE__N_O_O.TMS____1__
    M.z230)-mean(XMoaller$GAMMA.GLUTAMYLPHENYLALANINE__N_O_O.
    TMS____1__M.z230)
mean(Xmonoaller$MALTITOL.9TMS____1)-mean(XMoaller$MALTITOL.9
    TMS____1)
mean(Xmonoaller$N.ACETYL_MANNOSAMINE.MEOX_N_O_O_O_O.TMS_2____
    1__M.z149)-mean(XMoaller$N.ACETYL_MANNOSAMINE.MEOX_N_O_O_
    O_O.TMS_2____1__M.z149)
mean(Xmonoaller$MELIBIOSE.MEOX_8TMS_1____1)-mean(XMoaller$
    MELIBIOSE.MEOX_8TMS_1____1)
mean(Xmonoaller$M000000_A192008.101.xxx_NA_1918_52_PRED_VAR5
    _ALK_NA)-mean(XMoaller$M000000_A192008.101.xxx_NA_1918_52
    _PRED_VAR5_ALK_NA)
mean(Xmonoaller$M000000_A311004.101.xxx_NA_3093_84_PRED_VAR5
    _ALK_NA)-mean(XMoaller$M000000_A311004.101.xxx_NA_3093_84
    _PRED_VAR5_ALK_NA)
mean(Xmonoaller$M000000_A338003.101.xxx_NA_3372_53_PRED_VAR5
    _ALK_NA)-mean(XMoaller$M000000_A338003.101.xxx_NA_3372_53
    _PRED_VAR5_ALK_NA)


#t-tests for RF metabolites
t.test(Xaller1$N.ACETYL_MANNOSAMINE.MEOX_N_O_O_O_O.TMS_2____1
    __M.z149~metaaller1$any_allergies, alternative='two.sided
    ') #p = 0.1737
t.test(Xaller1$MELIBIOSE.MEOX_8TMS_1____1~metaaller1$any_
    allergies, alternative='two.sided') #p = 0.1717
t.test(Xaller1$M000000_A192008.101.xxx_NA_1918_52_PRED_VAR5_
    ALK_NA~metaaller1$any_allergies, alternative='two.sided')
     #p = 0.02844
t.test(Xaller1$M000000_A311004.101.xxx_NA_3093_84_PRED_VAR5_
    ALK_NA~metaaller1$any_allergies, alternative='two.sided')
     #p = 0.003852
t.test(Xaller1$M000000_A338003.101.xxx_NA_3372_53_PRED_VAR5_
    ALK_NA~metaaller1$any_allergies, alternative='two.sided')
     #p = 0.002143

#t-tests for PLS-DA metabolites
t.test(Xaller1$GLUTARIC_ACID.2TMS____1__M.z261~metaaller1$any
    _allergies, alternative='two.sided')  #p = 0.02009
t.test(Xaller1$GLYCERIC_ACID.3TMS____3~metaaller1$any_
    allergies, alternative='two.sided') #p = 0.1898
t.test(Xaller1$HOMOCYSTEINE__N_O_S.TMS____2__M.z246~
    metaaller1$any_allergies, alternative='two.sided') #p =
    0.006478
t.test(Xaller1$Calystegine_B2___4TMS_~metaaller1$any_
    allergies, alternative='two.sided') #p = 0.2184
t.test(Xaller1$Unknown.sst.cgl.023~metaaller1$any_allergies,
     alternative='two.sided') #p = 0.3821
t.test(Xaller1$Nivalenol___4TMS_~metaaller1$any_allergies,
    alternative='two.sided') #p = 0.2084
t.test(Xaller1$GLUTAMIC_ACID_N_O_O.TMS____1__M.z247~
    metaaller1$any_allergies, alternative='two.sided') #p =
    0.005013
t.test(Xaller1$TURANOSE_MEOX_8TMS_1____1__M.z307~metaaller1$
    any_allergies, alternative='two.sided') #p = 0.002177
t.test(Xaller1$GAMMA.GLUTAMYLPHENYLALANINE__N_O_O.TMS____1__M
    .z230~metaaller1$any_allergies, alternative='two.sided')
    #p = 0.3801
t.test(Xaller1$MALTITOL.9TMS___1~metaaller1$any_allergies,
    alternative='two.sided') #p = 0.02877


#plotting 6 of 8 significant metabolites
par(mfrow=c(2,3))
boxplot(Xaller1$M000000_A192008.101.xxx_NA_1918_52_PRED_VAR5
   _ALK_NA~metaaller1$any_allergies, main='A192008', xaxt='n
    ', ylab='Relative␣concentration')
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(Xaller1$M000000_A311004.101.xxx_NA_3093_84_PRED_VAR5
   _ALK_NA~metaaller1$any_allergies, main='A311004', xaxt='n
    ', ylab='Relative␣concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(Xaller1$M000000_A338003.101.xxx_NA_3372_53_PRED_VAR5
   _ALK_NA~metaaller1$any_allergies, main='A338003', xaxt='n
    ', ylab='Relative␣concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(Xaller1$GLUTAMIC_ACID_N_O_O.TMS____1__M.z247~
    metaaller1$any_allergies, main='Glutamic␣Acid', xaxt='n',
     ylab='Relative␣concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(Xaller1$TURANOSE_MEOX_8TMS_1____1__M.z307~metaaller1$
    any_allergies, main='Turanose', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(Xaller1$HOMOCYSTEINE__N_O_S.TMS____2__M.z246~
    metaaller1$any_allergies, main='Homocysteine', xaxt='n',
    ylab='Relative␣concentration')
```

```
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
   )



#2 - allergy
ModelallerPLS2$auc
ModelallerPLS2$nComp
ModelallerPLS2$miss
ModelallerRF2$miss
ModelallerRF2$auc


PLSaller2min=getVIP(ModelallerPLS2, model='min')
PLSaller2mid=getVIP(ModelallerPLS2, model='mid')
PLSaller2max=getVIP(ModelallerPLS2, model='max')
RFaller2min=getVIP(ModelallerRF2, model='min')
RFaller2mid=getVIP(ModelallerRF2, model='mid')
RFaller2max=getVIP(ModelallerRF2, model='max')

#check number of metabolites in each model
nrow(PLSaller2min)
nrow(PLSaller2mid)
nrow(PLSaller2max)
nrow(RFaller2min)
nrow(RFaller2mid)
nrow(RFaller2max)
RFaller2min
RFaller2mid
RFaller2max
head(PLSaller2min,10)
head(PLSaller2mid,10)
head(PLSaller2max,10)

taller2PLS=cbind.data.frame(head(PLSaller2min$name, 10),
   head(PLSaller2min$rank, 10), head(PLSaller2mid$name, 10),
    head(PLSaller2mid$rank, 10), head(PLSaller2max$name, 10)
   , head(PLSaller2max$rank, 10))
taller2RF=cbind.data.frame(head(RFaller2min$name, 10), head(
   RFaller2min$rank, 10), head(RFaller2mid$name, 10), head(
   RFaller2mid$rank, 10), head(RFaller2max$name, 10), head(
   RFaller2max$rank, 10))

write.table(taller2PLS, "PLStablealler2", sep=";", row.names
   =F, quote=F)
```

```
write.table(taller2RF, "RFtablealler2", sep=";", row.names=F
    , quote=F)

#check mean values to see if allergic or non allergic
    children have highest concentration of metabolite
Faaller=which(metaaller2$any_allergies==1)
XFaaller=Xaller2[Faaller,]
Xfanoaller=Xaller2[-Faaller,]

mean(Xfanoaller$Tryptamine___5.methoxy.___2TMS___MP)-mean(
    XFaaller$Tryptamine___5.methoxy.___2TMS___MP)
mean(Xfanoaller$INDOLE.3.ACETIC_ACID.2TMS____1)-mean(XFaaller
    $INDOLE.3.ACETIC_ACID.2TMS____1)
mean(Xfanoaller$GLUCO.GULO.HEPTOSE.MEOX_6TMS_1____1__M.z333)-
    mean(XFaaller$GLUCO.GULO.HEPTOSE.MEOX_6TMS_1____1__M.z333)
mean(Xfanoaller$M000000_A147005.101.xxx_NA_1462_03_PRED_VAR5
    _ALK_NA)-mean(XFaaller$M000000_A147005.101.xxx_NA_1462_03
    _PRED_VAR5_ALK_NA)
mean(Xfanoaller$MELIBIOSE.MEOX_8TMS_1____1)-mean(XFaaller$
    MELIBIOSE.MEOX_8TMS_1____1)
mean(Xfanoaller$M000000_A348003.101.xxx_NA_3477_88_PRED_VAR5
    _ALK_NA)-mean(XFaaller$M000000_A348003.101.xxx_NA_3477_88
    _PRED_VAR5_ALK_NA)
mean(Xfanoaller$X2.AMINOBUTYRIC_ACID___O_N.TMS____2)-mean(
    XFaaller$X2.AMINOBUTYRIC_ACID___O_N.TMS____2)
mean(Xfanoaller$M000000_A192017.101.xxx_NA_1916_51_PRED_VAR5
    _ALK_NA)-mean(XFaaller$M000000_A192017.101.xxx_NA_1916_51
    _PRED_VAR5_ALK_NA)
mean(Xfanoaller$M000000_A355001.101.xxx_NA_3550_16_PRED_VAR5
    _ALK_NA)-mean(XFaaller$M000000_A355001.101.xxx_NA_3550_16
    _PRED_VAR5_ALK_NA)
mean(Xfanoaller$METHIONINE___N_O.TMS____1___M.z176)-mean(
    XFaaller$METHIONINE___N_O.TMS____1___M.z176)
mean(Xfanoaller$M000000_A171005.101.xxx_NA_1692_54_PRED_VAR5
    _ALK_NA)-mean(XFaaller$M000000_A171005.101.xxx_NA_1692_54
    _PRED_VAR5_ALK_NA)
mean(Xfanoaller$DL.Serine)-mean(XFaaller$DL.Serine)
mean(Xfanoaller$X3_4.DIHYDROXYPHENYLACETIC_ACID.3TMS____3___M.
    z281)-mean(XFaaller$X3_4.DIHYDROXYPHENYLACETIC_ACID.3TMS_
    ___3___M.z281)
mean(Xfanoaller$OCTADECENOIC_ACID__.9.__Z__.TMS____1___M.z124)-
    mean(XFaaller$OCTADECENOIC_ACID__.9.__Z__.TMS____1___M.z124)
mean(Xfanoaller$OCTADECATRIENOIC_ACID___6_9__12.__Z_Z_Z__.TMS____
    1___M.z120)-mean(XFaaller$OCTADECATRIENOIC_ACID___6_9__12.__Z
    _Z_Z__.TMS____1___M.z120)
mean(Xfanoaller$Phenylacetaldehyde___1MEOX___BP)-mean(XFaaller
    $Phenylacetaldehyde___1MEOX___BP)

#t-tests for RF metabolites
t.test(Xaller2$Tryptamine___5.methoxy.___2TMS___MP~metaaller2$
    any_allergies, alternative='two.sided') #p = 2.98*10^-5
t.test(Xaller2$INDOLE.3.ACETIC_ACID.2TMS____1~metaaller2$any_
    allergies, alternative='two.sided') #p = 0.0866
t.test(Xaller2$GLUCO.GULO.HEPTOSE.MEOX_6TMS_1____1__M.z333~
    metaaller2$any_allergies, alternative='two.sided') # p =
    0.4155
t.test(Xaller2$M000000_A147005.101.xxx_NA_1462_03_PRED_VAR5_
    ALK_NA~metaaller2$any_allergies, alternative='two.sided')
     # p = 0.2575
t.test(Xaller2$MELIBIOSE.MEOX_8TMS_1____1~metaaller2$any_
    allergies, alternative='two.sided') # p = 0.01899

#t-tests for PLS-DA metabolites
t.test(Xaller2$M000000_A348003.101.xxx_NA_3477_88_PRED_VAR5_
    ALK_NA~metaaller2$any_allergies, alternative='two.sided')
     #p = 0.3247
t.test(Xaller2$X2.AMINOBUTYRIC_ACID___O_N.TMS____2~metaaller2$
    any_allergies, alternative='two.sided') #p = 0.06304
t.test(Xaller2$M000000_A192017.101.xxx_NA_1916_51_PRED_VAR5_
    ALK_NA~metaaller2$any_allergies, alternative='two.sided')
     #p =0.3787
t.test(Xaller2$M000000_A355001.101.xxx_NA_3550_16_PRED_VAR5_
    ALK_NA~metaaller2$any_allergies, alternative='two.sided')
     #p =0.3959
t.test(Xaller2$METHIONINE___N_O.TMS____1___M.z176~metaaller2$
    any_allergies, alternative='two.sided') #p = 0.2019
t.test(Xaller2$M000000_A171005.101.xxx_NA_1692_54_PRED_VAR5_
    ALK_NA~metaaller2$any_allergies, alternative='two.sided')
     #p =0.3781
t.test(Xaller2$DL.Serine~metaaller2$any_allergies,
    alternative='two.sided') #p = 0.08956
t.test(Xaller2$X3_4.DIHYDROXYPHENYLACETIC_ACID.3TMS____3___M.
    z281~metaaller2$any_allergies, alternative='two.sided') #
    p = 0.04533
t.test(Xaller2$OCTADECENOIC_ACID__.9._Z__.TMS____1___M.z124~
    metaaller2$any_allergies, alternative='two.sided') #p =
    0.3081
t.test(Xaller2$OCTADECATRIENOIC_ACID___6_9_12._Z_Z_Z__.TMS____1
    ___M.z120~metaaller2$any_allergies, alternative='two.sided
    ') #p = 0.2737
t.test(Xaller2$Phenylacetaldehyde___1MEOX___BP~metaaller2$any_
    allergies, alternative='two.sided') #p = 0.05777
```

```
#boxplots of significant metabolites
par(mfrow=c(2,2))
boxplot(Xaller2$Tryptamine___5.methoxy.___2TMS___MP~metaaller2$
   any_allergies, main='5-Methoxytryptamine', xaxt='n', ylab
   ='Relative␣concentration')
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
   )
boxplot(Xaller2$MELIBIOSE.MEOX_8TMS_1____1~metaaller2$any_
   allergies, main='Melibiose', xaxt='n', ylab='Relative␣
   concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
   )
boxplot(Xaller2$X3_4.DIHYDROXYPHENYLACETIC_ACID.3TMS____3___M.
   z281~metaaller2$any_allergies, main='3,4-
   Dihydrophenylacetic␣acid', xaxt='n', ylab='Relative␣
   concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
   )


par(mfrow=c(1,2))
size <- sapply(1:length(ModelallerRF2$yPred[,2]), function(i
   ) { sum(ModelallerRF2$yPred[,2]==ModelallerRF2$yPred[,2][
   i] & Yaller2==Yaller2[i]) })
plot(Yaller2, ModelallerRF2$yPred[,2], main="RF␣allergy␣
   predictions␣for␣Fathers", cex.main=0.57,  cex=size, pch
   =1, xlab='Allergy␣status', ylab='Predicted␣allergy')
legend("topleft", legend=c("2␣Fathers", "3␣Mixed"), cex
   =0.85, bty='n')
axis(1, at=c(2,3))
mtext(c("a)","b)"), at=c(1.9, 3.5))

plot(ModelallerPLS2$Fit$plsFitMid$variates$X[,1],
   ModelallerPLS2$Fit$plsFitMid$variates$X[,2], col=Yaller2
   +1, pch=16, xlab="Component␣1", ylab='Component␣2')
legend("topleft", legend=c('non-allergic', 'allergic'), col=
   unique(YallerA+1),pch=16   ,bty='n')
title(main="PLS-DA␣of␣samples␣from␣Father␣-␣allergy", cex.
   main=1)
```

```
#A - allergy
ModelallerPLSA$miss
ModelallerPLSA$auc
ModelallerRFA$miss
ModelallerRFA$auc
permFitnessallerRFAmid


PLSallerAmin=getVIP(ModelallerPLSA, model='min')
PLSallerAmid=getVIP(ModelallerPLSA, model='mid')
PLSallerAmax=getVIP(ModelallerPLSA, model='max')
RFallerAmin=getVIP(ModelallerRFA, model='min')
RFallerAmid=getVIP(ModelallerRFA, model='mid')
RFallerAmax=getVIP(ModelallerRFA, model='max')

#check number of metabolites in each model
nrow(PLSallerAmin)
nrow(PLSallerAmid)
nrow(PLSallerAmax)
nrow(RFallerAmin)
nrow(RFallerAmid)
nrow(RFallerAmax)
RFallerAmin
RFallerAmid
RFallerAmax
head(PLSallerAmin,10)
head(PLSallerAmid,10)
head(PLSallerAmax,10)


tallerAPLS=cbind.data.frame(head(PLSallerAmin$name, 10),
   head(PLSallerAmin$rank, 10), head(PLSallerAmid$name, 10),
    head(PLSallerAmid$rank, 10), head(PLSallerAmax$name, 10)
   , head(PLSallerAmax$rank, 10))
tallerARF=cbind.data.frame(head(RFallerAmin$name, 10), head(
   RFallerAmin$rank, 10), head(RFallerAmid$name, 10), head(
   RFallerAmid$rank, 10), head(RFallerAmax$name, 10), head(
   RFallerAmax$rank, 10))

write.table(tallerAPLS, "PLStableallerA", sep=";", row.names
   =F, quote=F)
write.table(tallerARF, "RFtableallerA", sep=";", row.names=F
   , quote=F)

#check mean values to see if allergic or non allergic
   children have highest concentration of metabolite
```

```
Aaller=which(metaallerA$any_allergies==1)
XAaller=XallerA[Aaller,]
XAnoaller=XallerA[-Aaller,]

mean(XAnoaller$M000000_A131008.101.xxx_NA_1306_75_TRUE_VAR5_
    ALK_NA)-mean(XAaller$M000000_A131008.101.xxx_NA_1306_75_
    TRUE_VAR5_ALK_NA)
mean(XAnoaller$SORBOSE.MEOX_5TMS_2____1__M.z306)-mean(XAaller
    $SORBOSE.MEOX_5TMS_2____1__M.z306)
mean(XAnoaller$ISOLEUCINE__N_O.TMS____2)-mean(XAaller$
    ISOLEUCINE__N_O.TMS____2)
mean(XAnoaller$Threose____1MEOX____3TMS__BP)-mean(XAaller$
    Threose____1MEOX____3TMS__BP)
mean(XAnoaller$No_match._262_30_QC1_NICE_Batch1_Rerun_
    05052017_6_228)-mean(XAaller$No_match._262_30_QC1_NICE_
    Batch1_Rerun_05052017_6_228)
mean(XAnoaller$Ferulic_acid___cis.___2TMS_)-mean(XAaller$
    Ferulic_acid___cis.___2TMS_)
mean(XAnoaller$SALICIN.5TMS____1__M.z272)-mean(XAaller$
    SALICIN.5TMS____1__M.z272)
mean(XAnoaller$SUCCINIC_ACID_2TMS____1__M.z151)-mean(XAaller$
    SUCCINIC_ACID_2TMS____1__M.z151)
mean(XAnoaller$Glucose___2.amino.2.deoxy.___4TMS__MP)-mean(
    XAaller$Glucose___2.amino.2.deoxy.___4TMS__MP)
mean(XAnoaller$X2.DEOXY.GLUCOSE.MEOX_4TMS____2)-mean(XAaller$
    X2.DEOXY.GLUCOSE.MEOX_4TMS____2)
mean(XAnoaller$X4.AMINOBUTYRIC_ACID__N_N_O.TMS____1)-mean(
    XAaller$X4.AMINOBUTYRIC_ACID__N_N_O.TMS____1)
mean(XAnoaller$Progesterone___11beta.hydroxy.___2MEOX____1TMS__
    BP)-mean(XAaller$Progesterone___11beta.hydroxy.___2MEOX____1
    TMS__BP)


#t-tests of RF metabolites
t.test(XallerA$M000000_A131008.101.xxx_NA_1306_75_TRUE_VAR5_
    ALK_NA~metaallerA$any_allergies, alternative='two.sided')
     #p = 0.007551
t.test(XallerA$SORBOSE.MEOX_5TMS_2____1__M.z306~metaallerA$
    any_allergies, alternative='two.sided') #p = 0.0002498

#t-tests of PLS-DA metabolites

t.test(XallerA$M000000_A371001.101.xxx_NA_3717_2_PRED_VAR5_
    ALK_NA~metaallerA$any_allergies, alternative='two.sided')
     #p = 0.129
t.test(XallerA$ISOLEUCINE__N_O.TMS____2~metaallerA$any_
    allergies, alternative='two.sided') #p = 0.008397
t.test(XallerA$M000000_A308005.101.xxx_NA_3079_45_PRED_VAR5_
   ALK_NA~metaallerA$any_allergies, alternative='two.sided')
    #p = 0.01723
t.test(XallerA$Threose____1MEOX____3TMS__BP~metaallerA$any_
    allergies, alternative='two.sided') #p = 0.02202
t.test(XallerA$No_match._262_30_QC1_NICE_Batch1_Rerun_
    05052017_6_228~metaallerA$any_allergies, alternative='two
    .sided') #p = 0.3269
t.test(XallerA$Ferulic_acid___cis.___2TMS_~metaallerA$any_
    allergies, alternative='two.sided') #p = 0.3649
t.test(XallerA$SALICIN.5TMS____1__M.z272~metaallerA$any_
    allergies, alternative='two.sided') #p = 0.02848
t.test(XallerA$SUCCINIC_ACID_2TMS____1__M.z151~metaallerA$any
   _allergies, alternative='two.sided') #p = 0.03979
t.test(XallerA$Glucose___2.amino.2.deoxy.___4TMS__MP~
    metaallerA$any_allergies, alternative='two.sided') #p =
    0.06911
t.test(XallerA$X2.DEOXY.GLUCOSE.MEOX_4TMS____2~metaallerA$any
   _allergies, alternative='two.sided') #p = 0.007593
t.test(XallerA$X4.AMINOBUTYRIC_ACID___N_N_O.TMS____1~
    metaallerA$any_allergies, alternative='two.sided') #p =
    0.002942
t.test(XallerA$Progesterone___11beta.hydroxy.___2MEOX____1TMS___
   BP~metaallerA$any_allergies, alternative='two.sided') #p
   = 0.07646

#boxplots of 6 significant metabolites (out of 8)
par(mfrow=c(2,3))
boxplot(XallerA$M000000_A131008.101.xxx_NA_1306_75_TRUE_VAR5
   _ALK_NA~metaallerA$any_allergies, main='A131008', xaxt='n
   ', ylab='Relative␣concentration')
legend("topright", legend='a', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(XallerA$SORBOSE.MEOX_5TMS_2____1__M.z306~metaallerA$
   any_allergies, main='Sorbose', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='b', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(XallerA$ISOLEUCINE___N_O.TMS____2~metaallerA$any_
    allergies, main='Isoleucine', xaxt='n', ylab='Relative␣
    concentration')
legend("topright", legend='c', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(XallerA$M000000_A308005.101.xxx_NA_3079_45_PRED_VAR5
    _ALK_NA~metaallerA$any_allergies, main='A308005', xaxt='n
    ', ylab='Relative␣concentration')
legend("topright", legend='d', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(XallerA$X2.DEOXY.GLUCOSE.MEOX_4TMS____2~metaallerA$
    any_allergies, main='Deoxyglucose', xaxt='n', ylab='
    Relative␣concentration')
legend("topright", legend='e', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )
boxplot(XallerA$X4.AMINOBUTYRIC_ACID__N_N_O.TMS____1~
    metaallerA$any_allergies, main='Aminobutyric␣acid', xaxt=
    'n', ylab='Relative␣concentration')
legend("topright", legend='f', bty='n', cex=1.5)
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )




plot(ModelallerPLSA$Fit$plsFitMid$variates$X[,1],
    ModelallerPLSA$Fit$plsFitMid$variates$X[,2], col=YallerA
    +1, pch=16, xlab="Component␣1", ylab='Component␣2')
legend("topleft", legend=c('non-allergic', 'allergic'), col=
    unique(YallerA+1),pch=16    ,bty='n')
title(main="PLS-DA␣of␣samples␣from␣Arterial␣-␣allergy", cex.
    main=1)

pPerm(ModelallerPLSA$auc[2,2], permFitnessallerPLSAmid)

#V - allergy
ModelallerPLSV$miss
ModelallerPLSV$auc
ModelallerRFV$miss
ModelallerRFV$auc
permFitnessallerRFVmid


PLSallerVmin=getVIP(ModelallerPLSV, model='min')
PLSallerVmid=getVIP(ModelallerPLSV, model='mid')
PLSallerVmax=getVIP(ModelallerPLSV, model='max')
RFallerVmin=getVIP(ModelallerRFV, model='min')
RFallerVmid=getVIP(ModelallerRFV, model='mid')
```

```
RFallerVmax=getVIP(ModelallerRFV, model='max')

#check number of metabolites in each model
nrow(PLSallerVmin)
nrow(PLSallerVmid)
nrow(PLSallerVmax)
nrow(RFallerVmin)
nrow(RFallerVmid)
nrow(RFallerVmax)
RFallerVmin
RFallerVmid
RFallerVmax
head(PLSallerVmin,10)
head(PLSallerVmid,10)
head(PLSallerVmax,10)

tallerVPLS=cbind.data.frame(head(PLSallerVmin$name, 10),
   head(PLSallerVmin$rank, 10), head(PLSallerVmid$name, 10),
    head(PLSallerVmid$rank, 10), head(PLSallerVmax$name, 10)
   , head(PLSallerVmax$rank, 10))
tallerVRF=cbind.data.frame(head(RFallerVmin$name, 10), head(
   RFallerVmin$rank, 10), head(RFallerVmid$name, 10), head(
   RFallerVmid$rank, 10), head(RFallerVmax$name, 10), head(
   RFallerVmax$rank, 10))

write.table(tallerVPLS, "PLStableallerV", sep=";", row.names
   =F, quote=F)
write.table(tallerVRF, "RFtableallerV", sep=";", row.names=F
   , quote=F)



#check mean values to see if allergic or non allergic
   children have highest concentration of metabolite
Valler=which(metaallerV$any_allergies==1)
XValler=XallerV[Valler,]
XVnoaller=XallerV[-Valler,]

mean(XVnoaller$MALTOSE.MEOX_8TMS_1____1__M.z217)-mean(XValler
   $MALTOSE.MEOX_8TMS_1____1__M.z217)
mean(XVnoaller$M000000_A275014.101.xxx_NA_2740_81_PRED_VAR5_
   ALK_NA)-mean(XValler$M000000_A275014.101.xxx_NA_2740_81_
   PRED_VAR5_ALK_NA)
mean(XVnoaller$Nonacosane___n)-mean(XValler$Nonacosane___n)
mean(XVnoaller$GLYCOLIC_ACID.2TMS____1__M.z66)-mean(XValler$
   GLYCOLIC_ACID.2TMS____1__M.z66)
```

```
mean(XVnoaller$Cholesterol.5beta_6beta.epoxide___1TMS_)-mean(
    XValler$Cholesterol.5beta_6beta.epoxide___1TMS_)
mean(XVnoaller$Phenylpyruvic_acid___1MEOX____1TMS__MP)-mean(
    XValler$Phenylpyruvic_acid___1MEOX____1TMS__MP)
mean(XVnoaller$M000000_A148003.101.xxx_NA_1464_64_PRED_VAR5_
    ALK_NA)-mean(XValler$M000000_A148003.101.xxx_NA_1464_64_
    PRED_VAR5_ALK_NA)
mean(XVnoaller$M000000_A196022.101.xxx_NA_1959_01_PRED_VAR5_
    ALK_NA)-mean(XValler$M000000_A196022.101.xxx_NA_1959_01_
    PRED_VAR5_ALK_NA)
mean(XVnoaller$X1_2_3.Propanetriol___1._4.hydroxy.3.
    methoxyphenyl_.___4TMS_)-mean(XValler$X1_2_3.Propanetriol_
    _1._4.hydroxy.3.methoxyphenyl_.___4TMS_)
mean(XVnoaller$M000000_A322001.101.xxx_NA_3204_72_PRED_VAR5_
    ALK_NA)-mean(XValler$M000000_A322001.101.xxx_NA_3204_72_
    PRED_VAR5_ALK_NA)
mean(XVnoaller$Unknown.bth.pae.013)-mean(XValler$Unknown.bth
    .pae.013)
mean(XVnoaller$MALTOTRIOSE.MEOX_11TMS_2____2__M.z361)-mean(
    XValler$MALTOTRIOSE.MEOX_11TMS_2____2__M.z361)



#t-tests of RF metabolites
t.test(XallerV$MALTOSE.MEOX_8TMS_1____1__M.z217~metaallerV$
    any_allergies, alternative='two.sided') #p = 0.9205
t.test(XallerV$M000000_A275014.101.xxx_NA_2740_81_PRED_VAR5_
    ALK_NA~metaallerV$any_allergies, alternative='two.sided')
     #p = 0.06712

#t-tests of PLS-DA metabolites
t.test(XallerV$Nonacosane__n~metaallerV$any_allergies,
    alternative='two.sided')#p = 0.1861
t.test(XallerV$GLYCOLIC_ACID.2TMS____1__M.z66~metaallerV$any_
    allergies, alternative='two.sided')#p = 0.1021
t.test(XallerV$Cholesterol.5beta_6beta.epoxide___1TMS_~
    metaallerV$any_allergies, alternative='two.sided')#p =
    0.05189
t.test(XallerV$Phenylpyruvic_acid___1MEOX____1TMS__MP~
    metaallerV$any_allergies, alternative='two.sided')#p =
    0.2616
t.test(XallerV$M000000_A148003.101.xxx_NA_1464_64_PRED_VAR5_
    ALK_NA~metaallerV$any_allergies, alternative='two.sided')
    #p = 0.4806
t.test(XallerV$M000000_A196022.101.xxx_NA_1959_01_PRED_VAR5_
    ALK_NA~metaallerV$any_allergies, alternative='two.sided')
    #p = 0.291
t.test(XallerV$X1_2_3.Propanetriol___1._4.hydroxy.3.
    methoxyphenyl_.___4TMS_~metaallerV$any_allergies,
    alternative='two.sided')#p = 0.3573
t.test(XallerV$M000000_A322001.101.xxx_NA_3204_72_PRED_VAR5_
    ALK_NA~metaallerV$any_allergies, alternative='two.sided')
    #p = 0.08813
t.test(XallerV$Unknown.bth.pae.013~metaallerV$any_allergies,
     alternative='two.sided')#p = 0.3283
t.test(XallerV$MALTOTRIOSE.MEOX_11TMS_2____2__M.z361~
    metaallerV$any_allergies, alternative='two.sided')#p =
    0.06282

#no significant metabolites found.


plot(ModelallerPLSV$Fit$plsFitMid$variates$X[,1],
    ModelallerPLSV$Fit$plsFitMid$variates$X[,2], col=YallerV
    +1, pch=16, xlab="Component␣1", ylab='Component␣2')
legend("topleft", legend=c('non-allergic', 'allergic'), col=
    unique(YallerV+1),pch=16    ,bty='n')
title(main="PLS-DA␣of␣samples␣from␣Venous␣-␣allergy", cex.
    main=1)


#M- allergy
ModelallerPLSM$miss
ModelallerPLSM$auc
ModelallerRFM$miss
ModelallerRFM$auc
permFitnessallerRFMmid


PLSallerMmin=getVIP(ModelallerPLSM, model='min')
PLSallerMmid=getVIP(ModelallerPLSM, model='mid')
PLSallerMmax=getVIP(ModelallerPLSM, model='max')
RFallerMmin=getVIP(ModelallerRFM, model='min')
RFallerMmid=getVIP(ModelallerRFM, model='mid')
RFallerMmax=getVIP(ModelallerRFM, model='max')

#check number of metabolites in each model
nrow(PLSallerMmin)
nrow(PLSallerMmid)
nrow(PLSallerMmax)
nrow(RFallerMmin)
nrow(RFallerMmid)
nrow(RFallerMmax)
RFallerMmin
RFallerMmid
RFallerMmax
head(PLSallerMmin,10)
head(PLSallerMmid,10)
head(PLSallerMmax,10)

tallerMPLS=cbind.data.frame(head(PLSallerMmin$name, 10),
   head(PLSallerMmin$rank, 10), head(PLSallerMmid$name, 10),
    head(PLSallerMmid$rank, 10), head(PLSallerMmax$name, 10)
   , head(PLSallerMmax$rank, 10))
tallerMRF=cbind.data.frame(head(RFallerMmin$name, 10), head(
   RFallerMmin$rank, 10), head(RFallerMmid$name, 10), head(
   RFallerMmid$rank, 10), head(RFallerMmax$name, 10), head(
   RFallerMmax$rank, 10))

write.table(tallerMPLS, "PLStableallerM", sep=";", row.names
   =F, quote=F)
write.table(tallerMRF, "RFtableallerM", sep=";", row.names=F
   , quote=F)


#check mean values to see if allergic or non allergic
   children have highest concentration of metabolite
Maller=which(metaallerM$any_allergies==1)
XMaller=XallerM[Maller,]
XMnoaller=XallerM[-Maller,]

mean(XMnoaller$SORBOSE.MEOX_5TMS_2____1__M.z306)-mean(XMaller
   $SORBOSE.MEOX_5TMS_2____1__M.z306)
mean(XMnoaller$GLYCERIC_ACID.3TMS____3)-mean(XMaller$GLYCERIC
   _ACID.3TMS____3)
mean(XMnoaller$Heptadecan.1.ol__n.___1TMS_)-mean(XMaller$
   Heptadecan.1.ol__n.___1TMS_)
mean(XMnoaller$M000000_A202006.101.xxx_NA_2017_1_PRED_VAR5_
   ALK_NA)-mean(XMaller$M000000_A202006.101.xxx_NA_2017_1_
   PRED_VAR5_ALK_NA)
mean(XMnoaller$Heptanoic_acid__n.___1TMS_)-mean(XMaller$
   Heptanoic_acid__n.___1TMS_)
mean(XMnoaller$Unknown.bth.pae.011)-mean(XMaller$Unknown.bth
   .pae.011)
mean(XMnoaller$M000000_A209004.101.xxx_NA_2078_56_PRED_VAR5_
   ALK_NA)-mean(XMaller$M000000_A209004.101.xxx_NA_2078_56_
   PRED_VAR5_ALK_NA)
mean(XMnoaller$Unknown.sst.cgl.104)-mean(XMaller$Unknown.sst
    .cgl.104)
mean(XMnoaller$GLUCARIC_ACID_1_4_LACTONE.4TMS____1)-mean(
    XMaller$GLUCARIC_ACID_1_4_LACTONE.4TMS____1)
mean(XMnoaller$M000000_A145005.101.xxx_NA_1436_41_PRED_VAR5_
    ALK_NA)-mean(XMaller$M000000_A145005.101.xxx_NA_1436_41_
    PRED_VAR5_ALK_NA)
mean(XMnoaller$D223156)-mean(XMaller$D223156)
mean(XMnoaller$M000000_A238002.101.xxx_NA_2366_44_PRED_VAR5_
    ALK_NA)-mean(XMaller$M000000_A238002.101.xxx_NA_2366_44_
    PRED_VAR5_ALK_NA)
mean(XMnoaller$SARCOSINE.2TMS____1)-mean(XMaller$SARCOSINE.2
    TMS____1)

#t-tests of RF metabolites
t.test(XallerM$SORBOSE.MEOX_5TMS_2____1__M.z306~metaallerM$
    any_allergies, alternative='two.sided') #p = 0.4961
t.test(XallerM$GLYCERIC_ACID.3TMS____3~metaallerM$any_
    allergies, alternative='two.sided') #p = 0.0688

#t-tests of PLS-DA metabolites
t.test(XallerM$Heptadecan.1.ol___n.___1TMS_~metaallerM$any_
    allergies, alternative='two.sided') #p = 0.1331
t.test(XallerM$M000000_A202006.101.xxx_NA_2017_1_PRED_VAR5_
    ALK_NA~metaallerM$any_allergies, alternative='two.sided')
     #p = 0.3415
t.test(XallerM$Heptanoic_acid___n.___1TMS_~metaallerM$any_
    allergies, alternative='two.sided') #p = 0.3703
t.test(XallerM$Unknown.bth.pae.011~metaallerM$any_allergies,
     alternative='two.sided') #p = 0.1788
t.test(XallerM$M000000_A209004.101.xxx_NA_2078_56_PRED_VAR5_
    ALK_NA~metaallerM$any_allergies, alternative='two.sided')
     #p = 0.4059
t.test(XallerM$Unknown.sst.cgl.104~metaallerM$any_allergies,
     alternative='two.sided') #p = 0.3912
t.test(XallerM$GLUCARIC_ACID_1_4_LACTONE.4TMS____1~metaallerM
    $any_allergies, alternative='two.sided') #p = 0.4429
t.test(XallerM$M000000_A145005.101.xxx_NA_1436_41_PRED_VAR5_
    ALK_NA~metaallerM$any_allergies, alternative='two.sided')
     #p = 0.419
t.test(XallerM$D223156~metaallerM$any_allergies, alternative
    ='two.sided') #p = 0.4018
t.test(XallerM$M000000_A238002.101.xxx_NA_2366_44_PRED_VAR5_
    ALK_NA~metaallerM$any_allergies, alternative='two.sided')
     #p = 0.42
t.test(XallerM$SARCOSINE.2TMS____1~metaallerM$any_allergies,
    alternative='two.sided') #p =3.382*10^-5
```

```
par(mfrow=c(2,3))
boxplot(XallerM$SARCOSINE.2TMS____1~metaallerM$any_allergies,
     main='Sarcosine', xaxt='n', ylab='Relative concentration
    ')
axis(side=1, at=c(1,2), labels=c('non-allergic', 'allergic')
    )


plot(ModelallerPLSM$Fit$plsFitMid$variates$X[,1],
    ModelallerPLSM$Fit$plsFitMid$variates$X[,2], col=YallerM
    +1, pch=16, xlab="Component 1", ylab='Component 2')
legend("topleft", legend=c('non-allergic', 'allergic'), col=
    unique(YallerM+1),pch=16    ,bty='n')
title(main="PLS-DA of samples from Mixed - allergy", cex.
    main=1)
```