Technische Universität München

Department of Matematics

Identification of Directly Imputable Missing Data Patterns Using R-vine Copulas and Application to Multiple Imputation

Master Thesis

by

Alexander Sakuth

Supervisor:	Prof. Claudia Czado, Ph.D.
Advisor:	Dominik Müller, Prof. Claudia Czado, Ph.D.
Submission date:	September 29, 2016

I hereby declare that this thesis is my own work and that no other sources have been used except those clearly indicated and referenced.

Garching, September the 29th, 2016

Abstract

In this master thesis we first give an introduction to missing data theory and some single and multiple imputation methods in Chapter 1 and 2. After introducing copulas in Chapter 3 with a focus on pair copula constructions and R-vine distributions, we develop an own imputation method in Chapter 4, based on R-vine copulas. The advantages of R-vine copulas are the high flexibility of R-vines and the separation of marginal distributions and dependence structure modeling. This might make copulas interesting in the imputation context as it is crucial to identify and then reproduce the dependence structure of the data. In the course of developing an own imputation algorithm, we derive a theory to determine for which conditional distributions of an R-vine a closed form can be derived analytically using only available bivariate copulas. Each conditional distribution gives us a solution to impute one corresponding missing data pattern. As not for all conditional distributions a closed form can be derived, we also present an algorithm to do a special R-vine fit with restrictions, such that a conditional distribution of choice can be derived from this R-vine. Finally, in Chapter 5 we test our own imputation method in several scenarios against other methods, evaluating how well they capture the dependence structure of the data. Especially in the tail dependence of clearly not multivariate normal distributed data our method shows some advantages.

Zusammenfassung

In dieser Masterarbeit geben wir zu Beginn eine Einführung zur Theorie zu fehlenden Daten und einigen einfachen und multiplen Imputationsmethoden in den Kapiteln 1 und 2. Nach einer Einführung zu Copulas mit Hauptaugenmerk auf Paar-Copula Konstruktionen und R-vines in Kapitel 3, entwickeln wir in Kapitel 4 eine eigene Imputationsmethode, die auf R-vines basiert. Die Vorzüge von R-vine Copulas sind die hohe Flexibilität von R-vines und die Trennung der Modellierung der marginalen Randverteilungen von der Modellierung der Abhängigkeitsstruktur. Dies könnte Copulas im Zusammenhang mit Imputation interessant machen, da es ein entscheidender Punkt ist die Abhängigkeiststruktur eines Datensatzes zu identifizieren und dann zu reproduzieren. Im Rahmen der Entwicklung einer eigenen Imputationsmethode, leiten wir die Theorie dazu her welche bedingten Verteilungen in geschlossener Form analytisch hergeleitet werden können bei einem R-vine, wenn man nur die vorhandenen bivariaten Copulas nutzen möchte. Mit jeder bedingten Verteilungsfunktion haben wir eine Lösung für ein zugehöriges Schemata fehlender Daten gefunden. Da wir nicht für alle bedingten Verteilungen eine geschlossene Form herleiten können, präsentieren wir einen Algorithmus für eine spezielle R-vine Anpassung mit Restriktionen, so dass eine ausgewählte bedingte Verteilung von diesm R-vine hergeleitet werden kann. Abschließend testen wir unsere eigene Imputationmethode in Kapitel 5 in verschiedenen Szenarien im Vergleich zu anderen Methoden und werten dabei aus, wie gut die Abhängigkeitsstruktur abgebildet wird. Besonders bei der Abhängigkeit an den Rändern der Verteilung von deutlich nicht normalverteilten Daten zeigt unsere Methode Vorteile.

Contents

1 Missing Data **1**
- 1.1 Missing data pattern . . . 1
- 1.2 Mechanisms of missing data . . . 2
  - 1.2.1 Missing Completely At Random (MCAR) . . . 4
  - 1.2.2 Missing At Random (MAR) . . . 5
  - 1.2.3 Missing Not At Random (MNAR) . . . 7

2 Imputation Methods **8**
- 2.1 Deletion methods . . . 8
  - 2.1.1 Listwise deletion . . . 9
  - 2.1.2 Pairwise deletion . . . 9
- 2.2 Single Imputation (SI) . . . 9
  - 2.2.1 Mean imputation . . . 10
  - 2.2.2 Simple random imputation . . . 11
  - 2.2.3 Regression imputation . . . 11
  - 2.2.4 Stochastic regression imputation . . . 14
  - 2.2.5 k-nearest neighbors . . . 16
  - 2.2.6 Summary . . . 17
- 2.3 Multiple Imputation (MI) . . . 17
  - 2.3.1 General MI process . . . 17
  - 2.3.2 Joint Modeling (JM) . . . 19
  - 2.3.3 Fully Conditional Specification (FCS) . . . 21
  - 2.3.4 CoImp - a copula imputation method . . . 22
- 2.4 Computational resources for single and multiple imputation . . . 23

3 Copulas **25**
- 3.1 Copulas . . . 25
- 3.2 Dependence Measures . . . 27
- 3.3 Pair copula constructions . . . 29
- 3.4 Regular vines (R-vines) . . . 32
- 3.5 Simulating regular vine copula distributions . . . 37

4 Vine Copula Imputation **38**
- 4.1 Ad hoc imputable missing data patterns . . . 38
  - 4.1.1 R-vine matrices of diagonal form . . . 39

4.1.2 Recursive decomposition of R-vines . . . 40
4.1.3 Imputation for a single missing variable . . . 42
4.1.4 Imputation for two missing variables . . . 46
4.1.5 Imputation for general missing data patterns . . . 48
4.1.6 Identifying redundant solutions . . . 51
4.1.7 Summary . . . 55
4.2 Non ad hoc imputable missing data patterns . . . 57
4.2.1 Adding one variable to an existing R-vine . . . 57
4.2.2 Adding several variables to an existing R-vine . . . 60
4.3 A general vine copula imputation method . . . 63

5 Simulation Study Based on Abalone Data 67
5.1 The abalone data . . . 67
5.2 Transformation of the margins . . . 69
5.3 Fitting an R-vine to the data . . . 74
5.4 Creating MCAR data . . . 76
5.5 Test setup and software specifications . . . 77
5.5.1 Data setup . . . 77
5.5.2 Software specifications . . . 77
5.5.3 Evaluation . . . 79
5.6 Simulations and results . . . 80
5.6.1 Checks . . . 80
5.6.2 Benchmarking with knn and CoImp . . . 88
5.6.3 Benchmarking with multivariate normal JM and FCS . . . 92

6 Conclusion 109

A Bivariate copula family distributions 111
A.1 Independence copula . . . 111
A.2 Gaussian copula . . . 111
A.3 Student t copula . . . 111
A.4 Clayton copula . . . 111
A.5 Gumbel copula . . . 112
A.6 Frank copula . . . 112
A.7 Remark to rotated copulas . . . 112

B Presentation of the VineCopulaImpute R-package 113

C Additional Simulation Study Results for a Smaller Sample Size 122

Chapter 1

Missing Data

The presence of missing data, or missing values, in a data set is a very common occurrence in real world data and can be seen frequently especially in social and medical studies as well as in political data. Causes can be various. In the following, a few examples are named. It is likely in surveys that interviewed persons are not willing to state their income. In longitudinal studies it is often seen that people drop out over time. The cause might be a move to another town. Other studies are even designed to have missing data. The participants are given only some questionnaires from the full set of questionnaires. This is at first glance bad for later analysis. However having people fill out all questionnaires might come along with high costs and reduces the willingness to go through all the questions what might result in nonrespondents. Last but not least also sloppy handling of the data collection or mistakes in measurements can cause missing data. Various other reasons might be given as example.

1.1 Missing data pattern

In this context we introduce the term *missing data pattern*. The missing data pattern solely describes the location of missing data. Note that it does not give any explanations or relationships between missing and observed data.

Figure 1.1 gives a visualization of a general missing data pattern structure. In this exemplary data with four variables and 100 observations some values were randomly deleted. Each row in the figure represents a combination of missingness for the four variables. Light gray indicates the variable is observed, i.e. not missing, whereas dark gray indicates a missing value. On the top is a bar plot showing how often missingness is observed in each single variable in total. In our example, all variables have missing data. Variable 3 has the most missing values and Variable 2 least. The barplot on the right is showing how often each combination is observed. In our example, read the third row as following: in 3 cases Variable 1 and Variable 4 are missing at the same time while Variables 2 and 3 are observed. In 27 cases we had no missing data at all according to the last row.

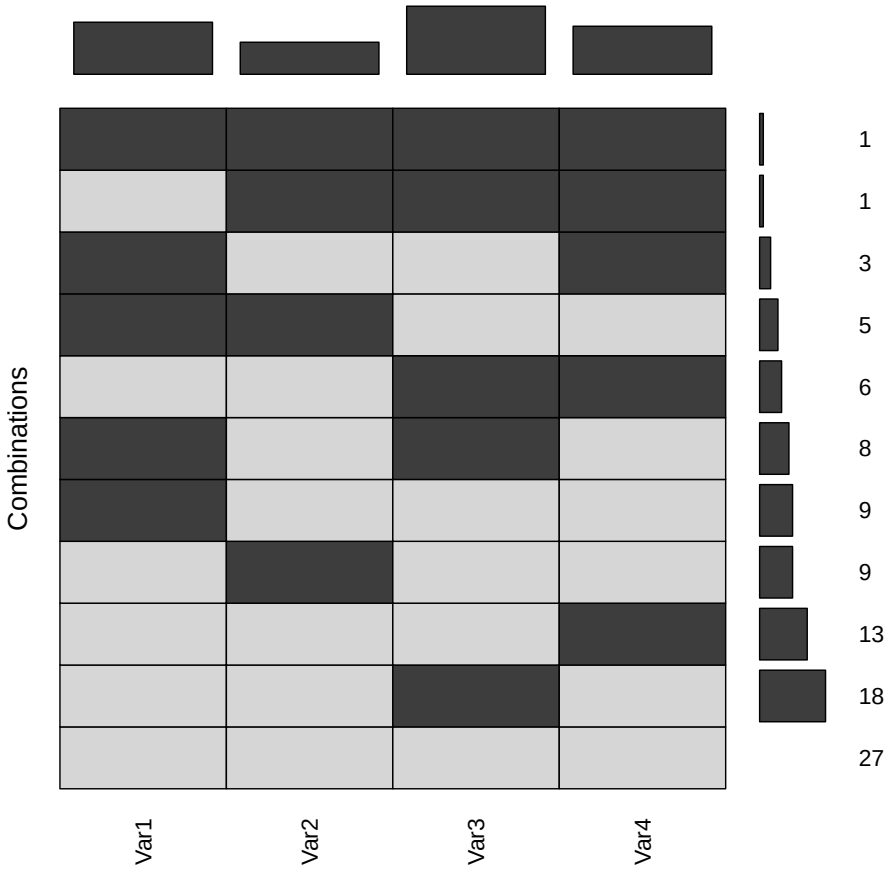
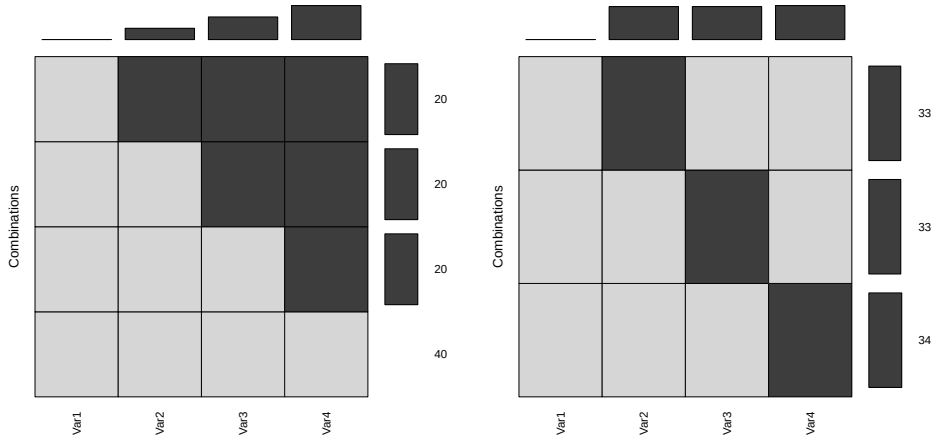


Figure 1.1: General missing data pattern structure for a four dimensional dataset.

Recall that a missing data pattern only describes the location of missing data. Nevertheless certain missing data pattern structures can help us to draw some conclusions about the origin of missingness. We will again consider the introductory examples of missing data in longitudinal studies due to dropout and the planned missing data design. A typical missing data pattern structure for these cases might look like as illustrated in Figure 1.2.

1.2 Mechanisms of missing data

The term missing data mechanism is about the reasons of missingness and about how the observed data might be related to the probability of missingness. Before we can make a proper description, we first need some notation to develop a general framework.
At first, we assume that a variable set $\boldsymbol{Y}$ is hypothetically complete. Then $\boldsymbol{Y}$ can be decomposed in two parts, namely $\boldsymbol{Y}_{obs}$ and $\boldsymbol{Y}_{mis}$. $\boldsymbol{Y}_{obs}$ denotes the part of $\boldsymbol{Y}$ that is observed, while $\boldsymbol{Y}_{mis}$ denotes the part that has missing values.



(a) (b)

Figure 1.2: Left: Typical longitudinal missing data pattern structure,
Right: Typical missing data pattern structure for planned missing data design.

Rubin (1976) classified the mechanisms of missing data and the main literature still refers to the three mechanisms introduced by him. These are named *Missing at Random*, *Missing Completely at Random* and *Missing Not at Random*.
The idea of Rubin was to introduce a random variable R indicating the missingness of a data point. R has binary values and a probability distribution. If a data point is observed, R equals 1, while if a data point is missing, R equals 0.
To introduce the earlier named mechanism introduced by Rubin, we follow the notation of Carpenter and Kenward (2013).
In a data set with d variables and n individuals

$$\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{id}), \quad i = 1, \ldots, n.$$

Let $\boldsymbol{Y}_{i,obs}$ and $\boldsymbol{Y}_{i,mis}$ be the subsets of variables of the observed or missing data respectively for all individuals $i = 1, \ldots, n$. Then

$$\boldsymbol{R}_i := (\mathbb{1}_{\{Y_{i1} \in \boldsymbol{Y}_{i,obs}\}}, \ldots, \mathbb{1}_{\{Y_{id} \in \boldsymbol{Y}_{i,obs}\}}), \quad i = 1, \ldots, n.$$

Example 1.1 *Assume a hypothetical full data set of 5 individuals measuring 3 variables is given by*

$$\boldsymbol{Y} = \begin{pmatrix} 2 & 10 & 27 \\ 4 & 18 & 22 \\ 3 & 11 & 22 \\ 8 & 19 & 28 \\ 3 & 17 & 16 \end{pmatrix},$$

but there is missing data and we only observe

$$\hat{\boldsymbol{Y}} = \begin{pmatrix} NA & 10 & 27 \\ 4 & 18 & NA \\ 3 & NA & 22 \\ NA & 19 & 28 \\ NA & NA & 16 \end{pmatrix},$$

where NA denotes a missing value. Then

$$\begin{aligned} \boldsymbol{Y}_1 &= (Y_{11}, Y_{12}, Y_{13}), \\ \boldsymbol{Y}_{1,obs} &= \{Y_{12}, Y_{13}\}, \\ \boldsymbol{Y}_{1,mis} &= \{Y_{11}\}, \\ \boldsymbol{R}_1 &= (0, 1, 1). \end{aligned}$$

Now we can define the mechanism of missing data as

$$P(\boldsymbol{R}_i|\boldsymbol{Y}_i). \tag{1.1}$$

Despite the definition of $\boldsymbol{R}_i$ as an indicator of observability it gives us all the information about the missingness and is therefore called mechanism of "missing" data not least because of historical reasons and common practice.

1.2.1 Missing Completely At Random (MCAR)

$$P(\boldsymbol{R}_i|\boldsymbol{Y}_i) = P(\boldsymbol{R}_i) \tag{1.2}$$

The probability of missing data is uncorrelated to observed and missing data. It is intuitively totally random like a coin flip. The *marginplot()* function from the *VIM*-package, see Templ et al. (2015), visualizes how missingness occurs on two variables in a data set. An example for two variables where data was deleted totally random can be seen in Figure 1.3. The plot shows an enhanced scatterplot for the two variables *length* and *diameter* from the *abalone data set* which we will use for a case study later. For more details on the data set see Chapter 5. The light gray dots show a scatterplot of the observations for which both variables are observed, while the dark gray dots show the observations of one variable while the other is missing. With the same gray scale the boxplots on each side summarize the distributions of the observed and incomplete data respectively. The gray numbers show how many values in total are missing for each variable. Is it the case that data is missing in both variables at the same time, the total number of these cases is shown in black font at the lower left corner. In our example, the variable *length* is missing in 400 cases and the variable *diameter* is missing in 250 cases, both variables are simultaneously missing in 103 cases. Unsurprisingly, we cannot see a very special shape in Figure 1.3 or a significant difference in boxplots as data was deleted completely at random. We took just a random sample without replacement to create the missing data. As the missingness is unrelated to data, the observed data is representative of the population. MCAR is the easiest mechanism to handle, but very seldom in reality. The mechanism presented next is more general.

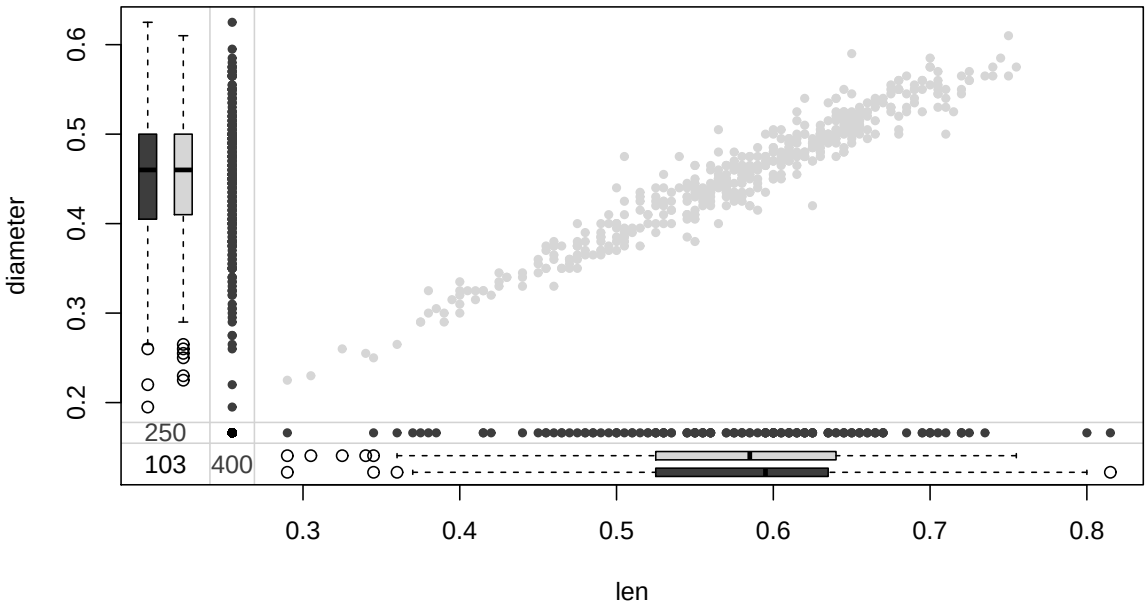


Figure 1.3: Marginal plot of two variables with MCAR mechanism.

1.2.2 Missing At Random (MAR)

$$P(\boldsymbol{R}_i|\boldsymbol{Y}_i) = P(\boldsymbol{R}_i|\boldsymbol{Y}_{i,obs}) \tag{1.3}$$

For data which is MAR the probability of missingness is dependent on the observed data only and independent of missing data. In contrast to MCAR, the probability of observing a variable is dependent on the values of the other variables. Graham (2009) therefore suggested it would be better to call this mechanism not MAR but "conditionally missing at random" to prevent misunderstandings. One example would be that people with higher income are more likely to leave out the question for their income in a survey. As long as you ask for other variables as the job position, the car they drive and suitable others, you could see that people with fancier cars and better job position are more likely to have missing values for the income variable. So the missingness depends on the observed variables. A more intuitive and straight forward example would be a fisher who keeps track of his catches. He measures the length and the weight. But as he is not allowed to catch fishes below some length to protect the population, all measures for the weight of the too short fishes are missing as he has to throw back these in the water immediately after measuring the length.

Figure 1.4 shows again a marginal plot of two variables, this time data was not deleted completely random but a missing value in *length* is more likely the higher the value of *diameter* is. In detail, data was deleted from the complete data set as following:

1. Delete data from the variable *diameter* randomly by drawing a sample without replacement.

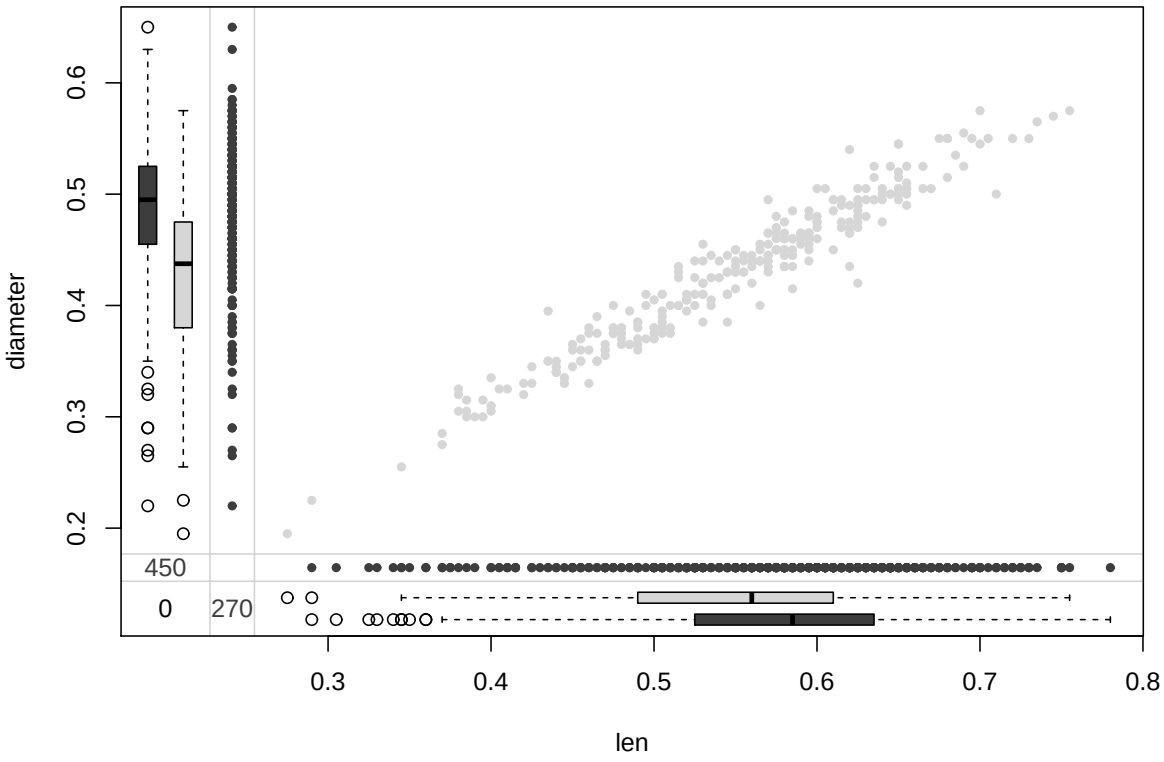


Figure 1.4: Marginal plot of two variables with MAR mechanism.

2. Determine the $(0.25, 0.5, 0.75)$ quantiles of the remaining diameter observations and denote them by (q_1, q_2, q_3).

3. Each observation x from the variable *length* is deleted if

$$\begin{cases} x \leq q_1 & \text{, with } p = 0.2 \\ q_1 < x \leq q_2 & \text{, with } p = 0.4 \\ q_2 < x \leq q_3 & \text{, with } p = 0.6 \\ x > q_3 & \text{, with } p = 0.8 \end{cases}$$

This time in Figure 1.4 contrary to Figure 1.3 the boxplots next to the vertical axis differ clearly. Obviously by our construction of missingness it is more likely to have missing data in *length* the higher the values of the *diameter* are. Also the difference in the horizontal boxplot can be understood as a consequence of the construction. By deleting random values from diameter one could expect rather similar shapes. However as a result of the high correlation and the more frequent deletion of higher values in length the shapes differ. For a better understanding, note that it is not necessarily important to delete the data from diameter in the first step. Doing the same construction on the full diameter variable

and delete observations randomly from it at the very last can still produce MAR data. Even though the missingness in length depends on the quantiles which might differ for the full and the observed-only data, the fact that diameter values are deleted randomly implies that the remaining data is still representative for the whole dataset.
How the relationship between missingness and observed values work exactly is fortunately not important as MCAR and MAR mechanisms are called ignorable. Note that this does not mean the missing data mechanism, i.e. whether MCAR or MAR data is underlying, should be ignored. It is only ignorable in the sense of how exactly the missingness was caused for later analysis. In the next introduced mechanism this will be not the case.

1.2.3 Missing Not At Random (MNAR)

$$P(\boldsymbol{R}_i|\boldsymbol{Y}_i) \neq P(\boldsymbol{R}_i|\boldsymbol{Y}_{i,obs}) \tag{1.4}$$

Missing values in this mechanism are caused by Y_{mis} even after taking all other variables into account. One example would be a data set only storing the income and the favorite football team of a person. Assuming again that persons with higher income are more likely to leave out the question for their income and additionally assume that the favorite football team is uncorrelated to the income, the missingness in income is not explainable by the observed data, i.e. the favorite football team. The missingness in income is dependent on income itself or a latent variable. MNAR requires more complex approaches, as the mechanism is called non ignorable. Whether a data set is MAR or MNAR is hard to say, as the information needed for the distinction depends on the missing values. As Graham (2009) says, the best way to find out if data is MNAR is to follow up and measure again. Unfortunately, this is not always possible. There exist several statistical tests to distinguish between MCAR and MAR models, see for example Little (1988).

The three different mechanisms of missing data imply different requirements for the handling of the missing data. The next chapter about imputation methods will give us an overview about simple and more elaborate techniques to handle missing data.

Chapter 2

Imputation Methods

In the last chapter we shortly introduced some missing data theory. The problem with missing data is that nearly all analytical methods and software packages were designed for complete data. First we want to show how missing data was and is dealt with in past up till now. We will start with some methods that were used early and have some severe weaknesses. Still we want to present them not only to give a short historical overview but more important to show the weaknesses these often intuitive methods have and why there is a need for more specialized techniques. Last but not least these methods are still implemented in many software packages and one will come across them. After introducing the simple methods, we will move forward to the multiple imputation (MI) methods. Imputation is the process of complementing the not complete data with values. In contrast to single imputation, in multiple imputation the complementation is done several times. However before we go into more detail later we start with the most intuitive and still wide spread approach, namely deletion methods.
We will use an exemplary data with 2 variables from the *abalone data*, namely length and diameter of female abalone, see Chapter 5 for a more detailed explanation of the data, to show the concepts with missingness only occurring in the length variable for simplicity. We will point out how a concept works with missingness in more than one variable or in higher dimensions if necessary. Usually imputation is a tool to achieve a higher purpose. For example one wants to impute a data set to perform a regression analysis on the imputed data set. We distinguish between the imputation model and the analysis model. The parameters of interest in the analysis model might be slopes in a regression, their standard deviations, correlations, means, or others. We will present here the imputation models and show the influence on the parameters of interest in the analysis of the imputed data.

2.1 Deletion methods

Deletion methods are still standard options in software packages and often immediate reaction when dealing with missing data. These methods require MCAR to achieve unbiased parameter estimates and even in the MCAR scenario the statistical power is seriously reduced due to the wasteful dealing with data.

2.1.1 Listwise deletion

Listwise deletion also referred to as casewise deletion or complete case analysis excludes all observations with at least one missing value in any variable, so that the remaining data set only contains complete data, if there exist any. As mentioned already this method requires MCAR and can produce biased parameters when this assumption is violated.
Still it is popular since one needs no software packages or special methods to continue with the data analysis. Especially beginners might just not be aware of the fact that listwise deletion can have sincere impact on later analysis.

Example 2.1 (abalone data) *The data consists of 1306 observations on 2 variables. The variables length and diameter contain the measurements of the abalone length and diameter respectively for female abalone.*
Assuming missing data for all lengths of an abalone with a diameter less than $0.45mm$ *distorts the estimates, see Figure 2.1. In the observed data the mean of lengths of a female abalone is* $0.58mm$*, while the mean of the complete data is* $0.63mm$*. This difference is due to the positive correlation of both variables, so that after deleting lower diameter values the complete cases are not representative for the whole and contain higher length values. Note that this data is MAR.*

2.1.2 Pairwise deletion

Pairwise deletion, also referred to as available-case analysis follows the idea to use as many data as available and does an analysis by analysis approach. The prime example application would be the calculation of a sample correlation matrix. Using only the complete subsets for all combinations of two variables a problem arises due to the different sample base size that can cause a non positive definite correlation matrix, see also Enders (2010) Section 2.2.4 for further reading. Even though the idea to consider as many data as possible in every step is a clear improvement, the problem with pairwise deletion is once again that it requires MCAR data and can produce distorted parameters if the assumption does not hold.

2.2 Single Imputation (SI)

The term single imputation comes from the fact that these approaches generate a single replacement value for each missing data point. Multiple imputation methods in contrast create several copies of the incomplete data set and impute each data set with different plausible estimates of the missing values. The advantage of single imputation is that it yields a complete data set and uses data that deletion methods would waste. Still we want to name some cons for single imputation methods. They can produce biased estimates and underestimate standard errors. Intuitively, standard errors should be higher for missing values and imputed values should not be treated as real data as they contain more uncertainty.

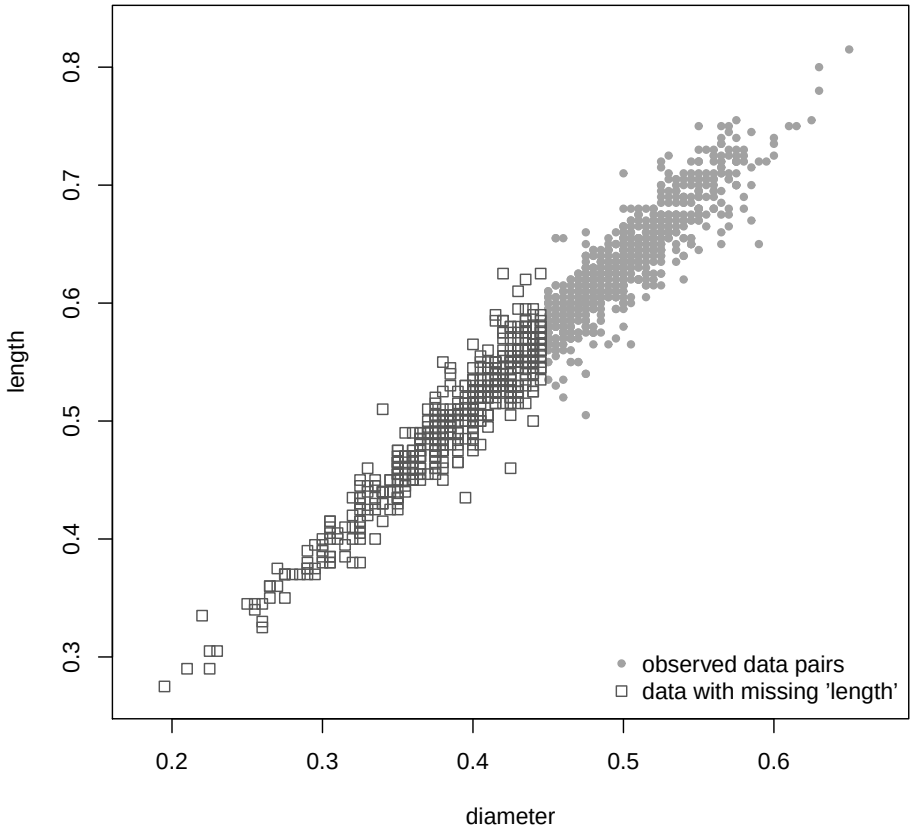


Figure 2.1: Illustration of *abalone data* and data deletion in Example 2.1.

2.2.1 Mean imputation

The first imputation method we want to introduce works with the idea to simply replace all missing observations with one single value, the arithmetic mean of the observed values. Also other values like the median might be reasonable. Unfortunately, this method comes with serious limitations even in the MCAR case.

Example 2.1 (Continued) *For illustration we will again look at the abalone data but this time we deleted 500 length observations randomly. The mean of the remaining observations for the length variable is* $0.58mm$ *and all missing observations were imputed with this value. One can clearly see in Figure 2.2 that the imputed values totally miss the inherent positive correlation of the data. The correlation of both variables in the full data is* 0.97 *while it is only* 0.74 *after the imputation. Also the standard deviation and variance are much too low in the imputed data set as no variation is implemented in the imputation method.*

It should be clear that replacing all missing values with a single value is a rather bad approach. So the next method we will introduce works differently.

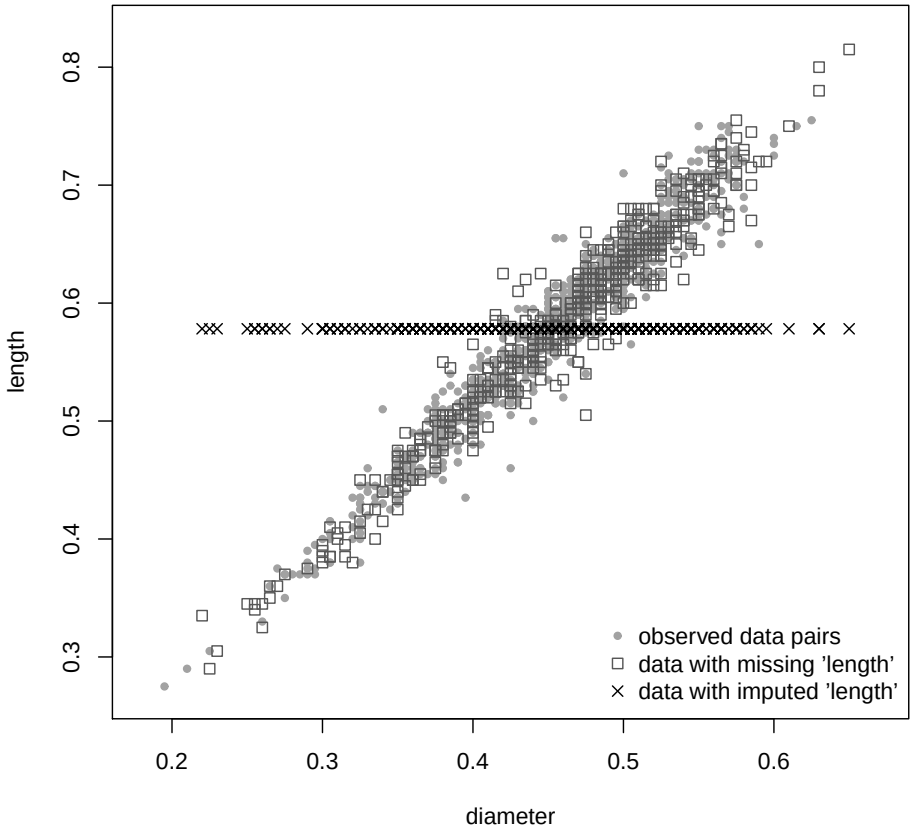


Figure 2.2: Mean imputation for the length in the *abalone data* in a MCAR scenario.

2.2.2 Simple random imputation

This time we will not use a single value for the imputation of all missing data, but randomly sample from the observed values with replacement to obtain a full data set.

Example 2.1 (Continued) *Again we will have a look at the abalone data and delete 500 length observations randomly to impute these now by the random sample method. Contrary to the mean imputation now there is variation incorporated as we can see in Figure 2.3, however still the positive correlation of the full data which is* 0.97 *is missed clearly and is even worse than in the mean imputation at* 0.59*.*

Again it is clear that this method does not yield unbiased estimates even in the MCAR case and we need some concept that picks up on the inherent correlation. This will be adressed in the next section.

2.2.3 Regression imputation

In the previous method we did not use any correlation between the variables while imputing. This was crucial as we saw in the *abalone data* and Example 2.1 so far. In this

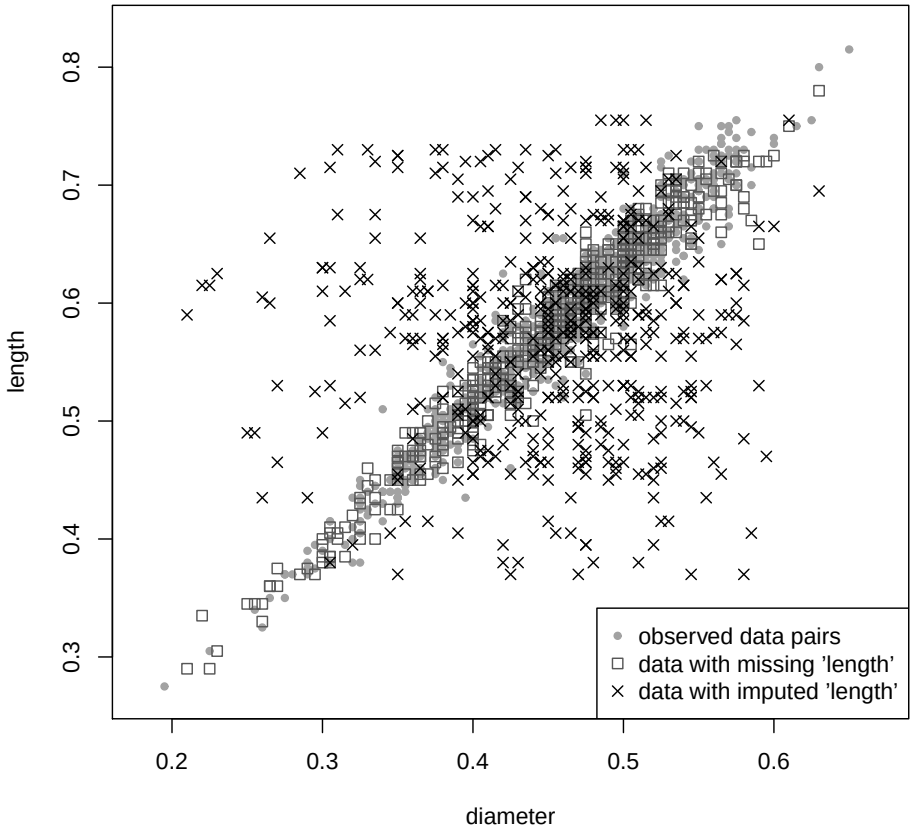


Figure 2.3: Simple random imputation for the length in the *abalone data* in a MCAR scenario.

data set it can be seen that the longer an abalone was, the greater was its diameter. To incorporate such an effect we will now use a linear regression model to impute the missing values.

Example 2.1 (Continued) *We will once more use the abalone data to outline the regression method. However now we delete the data in such a manner that it is MAR. This is achieved by deleting* 70% *of all length observations with a diameter of more than* 0.45*mm, and only* 30% *of the observations with a diameter of at most* 0.45*mm. Now the missing observations are imputed in the following way:*

1. *Fit the linear regression model*

$$X_{i,length} = \beta_0 + \beta_1 \cdot X_{i,diameter} + \epsilon_i, \ i = 1, \ldots, n \tag{2.1}$$

for the n complete cases, where $\boldsymbol{X}_{length}$ *and* $\boldsymbol{X}_{diameter}$ *denote the data vectors of both variables respectively while* β_0 *and* β_1 *denote intercept and slope of the regression and* ϵ_i *is the normally distributed residual error with mean* 0 *and standard deviation* σ*.*

2. *Evaluate the regression equation for all missing cases and impute the incomplete cases with the predicted scores. Here we assume that diameter is observed and missingness only occurs in length.*

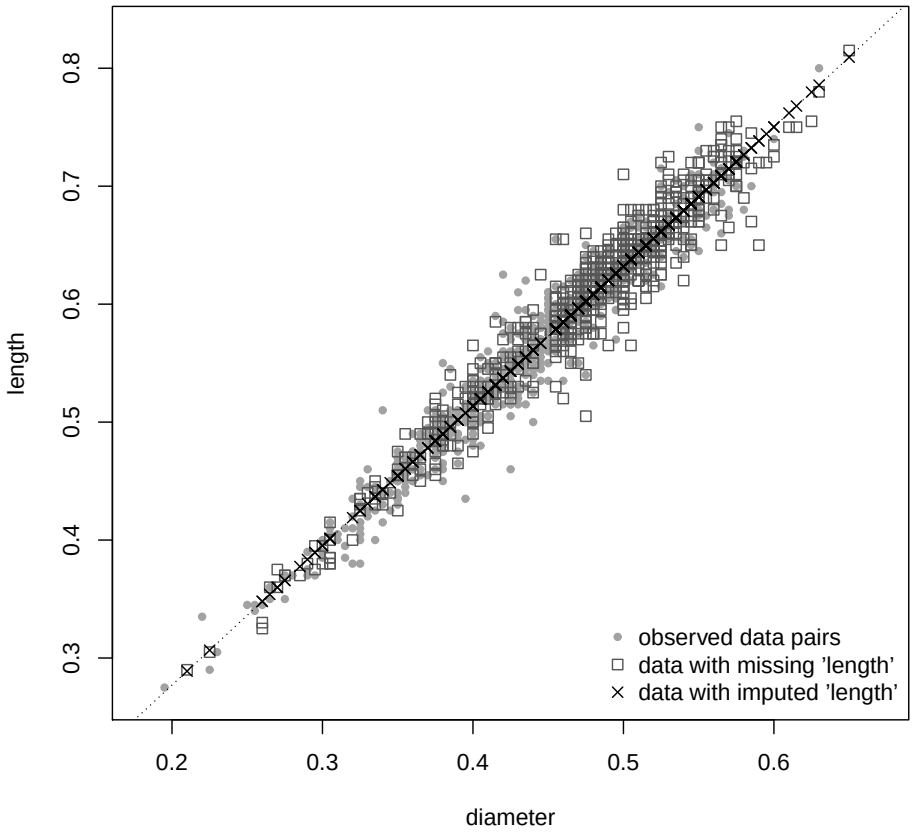


Figure 2.4: Regression imputation for the length in the *abalone data* in a MAR scenario.

In Figure 2.4 we can see that the inherent correlation is picked up. Unfortunately the fact that all imputed values lie directly on the regression line intensifies the positive correlation and still gives biased estimates of the correlation parameter. Compare the full data correlation of 0.97 *to* 0.99 *in the imputed data set. Again as in the mean imputation we miss variation.*

In a multivariate data set things become more complex and we have many regression equations. Let X, Y and Z be three variables, all with some missing values. Ignoring the complete cases and data rows which are totally missing, Figure 2.5 shows the possible missing data patterns. Then we have to fit regression models as follows:

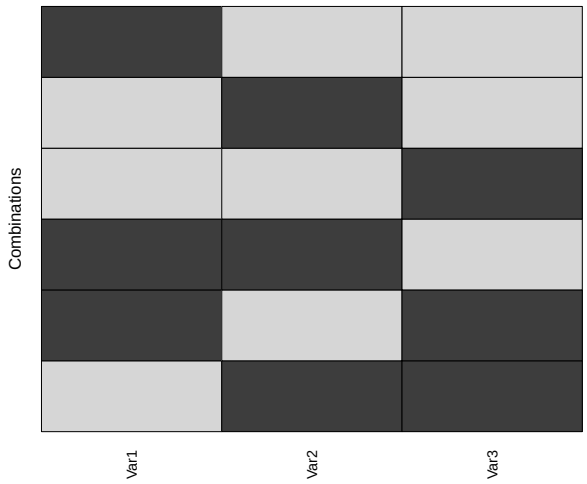


Figure 2.5: Missing data patterns for 3 variables.

$$\begin{aligned}
X \text{ missing: } & X \sim Y + Z \\
Y \text{ missing: } & Y \sim X + Z \\
Z \text{ missing: } & Z \sim X + Y \\
X \text{ and } Y \text{ missing: } & X \sim Z \\
& Y \sim Z \\
X \text{ and } Z \text{ missing: } & X \sim Y \\
& Z \sim Y \\
Y \text{ and } Z \text{ missing: } & Y \sim X \\
& Z \sim X
\end{aligned} \tag{2.2}$$

The number of variables is directly related to the number of regression problems. Fortunately there are special methods like the sweep operator implemented in many software packages that solve all single regression equations in short time, see Goodnight (1978). Still with this method we miss the variation around the regression line. Note that the regression imputation is also known as conditional mean imputation. The on hand solution is to add residual errors to the imputed values what will be discussed in the next section.

2.2.4 Stochastic regression imputation

The idea of stochastic regression imputation is to base on the regression imputation and restore the lost variation around the regression line by adding normally distributed residual errors. Note that this is the only single imputation method that generates unbiased estimates in the MAR case, according to Enders (2010) p.46.

Example 2.1 (Continued) *We use exactly the same incomplete data as for the regression imputation. Figure 2.6 shows the imputed data set. The correlation of the imputed data is now* 0.97*, exactly as in the full data. The mean for the length is* 0.58 *in the imputed data and* 0.58 *in the full data.*

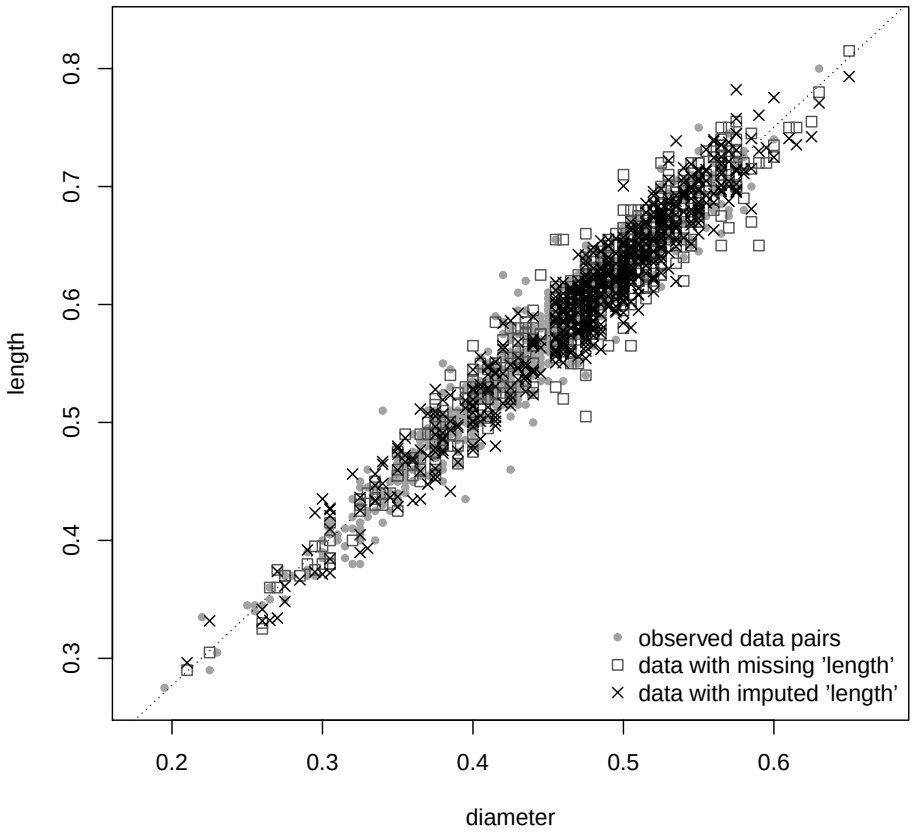


Figure 2.6: Stochastic regression imputation for the length in the *abalone data* in a MAR scenario.

The multivariate case builds up on the regression imputation as well. This means in a three variable data example one would still have to solve the equation system (2.2). However the residual error terms are now multivariate normal distributed if more than one variable is missing.

So we have now a solution that gives us unbiased estimates in the MAR case. Unfortunately this does not mean we are done. Standard errors are still too low. It should be intuitively clear that the errors and uncertainty in an imputed data is higher than in a complete data. So far we treated the imputed data as real data and missed the variability that enters through the imputation. How this issue can be handled is addressed in the section about multivariate imputation which we will start directly before we conclude

this section about single imputation methods with the presentation of a nonparametric imputation method, the k-nearest neighbors imputation.

2.2.5 k-nearest neighbors

Whereas the stochastic linear regression imputation is a parametric method, an easy to understand nonparametric imputation method is the *k-nearest neighbors (knn)* imputation. The basic idea is that, if something is similar in some respects, it is likely to be similar in other respects.
In its origin knn is a classification system. Assume the simple case that we have measurements of the maximum daily temperature for the period of one year from a fixed location. Now we are given the maximum daily temperature of an unknown day from the same place and we want to classify from which season it is. The knn classification method suggests to look for the data points that are closest to our new point, and see from which season they are. Assume the five nearest daily maximum temperature measurements are three observations from summer, one from spring and another one from autumn. Then it is likely that the new point is also from summer as this was the most often observed class in the nearest neighbors. The knn is only one classification system and other classification methods use for example random forests, as introduced by Breiman (2001), or the propensity score, as described in Essama-Nssah (2006). Even a neural network based classification can be used for imputation, see Ennett et al. (2008).

One application of the knn classification method is imputation, see also Jonsson and Wohlin (2004). Assume we have given an incomplete d-dimensional data set and let y_a be an observation with missing values for some attributes $\{X_m\}_{m\in M}$ and observed attributes $\{X_l\}_{l\in L}$, with $M, L \subsetneq \{1,\dots,d\}$, $M \cap L = \emptyset$ and $M \cup L = \{1,\dots,d\}$. Then the knn method imputes the missing value in X_{m_1} for an $m_1 \in M$ by looking at all cases which are complete on X_{m_1} as well as on all of the observed attributes $\{X_l\}_{l\in L}$. We call these cases *comparable*. Among these comparable cases we now try to find the nearest neighbors.

The goal of finding nearest neighbors can be realized by identifying the comparable observations which minimize some similarity measure, like any distance metric. One very popular example used in practice is the Euclidean metric. For a comparable observation y_b the Euclidean distance is $||y_a - y_b||_2 := \sqrt{\sum_{l\in L}(y_{a_l} - y_{b_l})^2}$, where y_{a_l} is the value of attribute X_l of observation y_a. A weakness of the Euclidean metric is that if variables are not scaled, the variable with the largest range has most weight. The Gower distance overcomes this weakness. Gower (1971) suggested to normalize the Manhattan metric, i.e. the absolute distances in an attribute $|y_{a_l} - y_{b_l}|$, by the range of the attribute r_l on all comparable observations including the observation we want to impute. Then the distance measures for every attribute are between 0 and 1. For a comparable observation y_b the Gower distance is $||y_a - y_b||_g := \sum_{l\in L} \frac{|y_{a_l} - y_{b_l}|}{r_l}$. A variation of the Gower distance is also used in the *kNN* function of the *VIM* R-package, see Templ et al. (2015).

Once the k nearest neighbors $y_a^1,\dots,y_a^k$ of y_a are identified, the attribute of interest X_{m_1} can be imputed for example using the arithmetic average $\sum_{i=1}^k y_{a_{m_1}}^i$. A weighted average

that gives more weight to neighbors with smaller distance is also possible. For a discrete variable, the median might be a good solution.
The choice of k should be a tradeoff of large enough to give reliable results and small enough, so that the neighbors are in fact near. Duda et al. (2001) suggest to choose $k = \sqrt{N}$, where N is the number of comparable cases.
The knn imputation method clearly is a better alternative than imputing all missing values by the mean of a variable, but it is also computationally time consuming, since many distances have to be calculated. The method is deterministic and will always give the same results, as there is no random element. Furthermore all imputed values are limited in the range of the observed values. The imputed values therefore might result in lighter tails and less variation. If there is no close neighbor at all, the imputed value is likely to be a bad estimate of the real value.

2.2.6 Summary

So far we presented the most commonly used and intuitive single imputation methods. Obviously there are far more, but none that is outstanding. Some are specific for special scenarios as longitudinal data and it might make sense to look up more details for the special missing data one wants to impute. Besides the k-nearest neighbors there are several other nonparametric approaches. All these methods have the same problems of biased parameter estimates or too low variation. So we will now introduce multiple imputation methods.

2.3 Multiple Imputation (MI)

2.3.1 General MI process

Multiple Imputation (MI) is a data based approach. This means unlike in a model based approach the imputation of the data is separated from the analysis. An example for a model based approach would be the Full Information Maximum Likelihood (FIML) for missing data, that will not be discussed in this thesis. For more details on FIML imputation see Dong and Peng (2013).
More precisely, there are three phases in the MI process, namely the imputation phase, the analysis phase and the pooling phase. In the imputation phase, $m > 1$ copies of the incomplete data set are created and missing values are imputed using the preferred imputation method. Afterwards in the analysis phase the analysis, e.g. a regression analysis, is run as for a complete data set and the parameter of interest q is estimated for each imputed data set, yielding the estimates $\hat{q}_1, \ldots, \hat{q}_m$. Of course this can be extended to a set of parameters of interest. Finally, in the pooling phase all estimates $\hat{q}_1, \ldots, \hat{q}_m$ are pooled to a single point estimate for the parameter of interest with the respective standard error. Since we replace the missing values in the imputation phase with various reasonable values it is possible to factor in the uncertainty that arises from the missingness of the values. So far in the SI methods we treated the imputed data set as real data, although it should be clear that the imputed values are only one possible realization and therefore the parameter estimate should factor in this additional uncertainty.

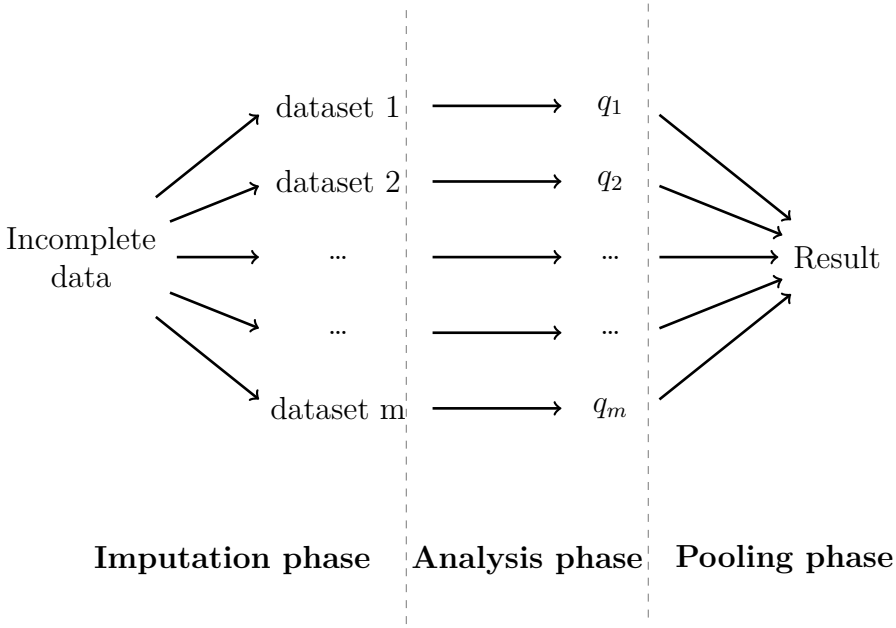


Figure 2.7: The three phases in the multiple imputation procedure.

Rubin (1987) formulated rules, today known as *Rubin's rules*, for the pooling which incorporate the additional uncertainty associated with the imputation. The m sample estimates $\hat{q}_1, \ldots, \hat{q}_m$ for the parameter of interest q are pooled to a point estimate $\bar{q}$ by just building the arithmetic average

$$\bar{q} := \frac{1}{m} \sum_{i=1}^{m} \hat{q}_i. \tag{2.3}$$

The variance consists of several components.

1. $$Var_W := \frac{1}{m} \sum_{i=1}^{m} \hat{se}_i^2, \tag{2.4}$$

   the so called sample within-imputation variance. It is the arithmetic average of all the squared sample standard errors of $\hat{q}_1, \ldots, \hat{q}_m$.

2. $$Var_B := \frac{1}{m-1} \sum_{i=1}^{m} (\hat{q}_i - \bar{q})^2, \tag{2.5}$$

   the so called sample between-imputation variance. This component accounts for the fact that missing values are no real data and add noise. Here is the main difference to SI methods.

The total sample variance then is

$$Var_T := Var_W + Var_B + \frac{Var_B}{m}. \tag{2.6}$$

The last term originates from the fact that the point estimate $\bar{q}$ from (2.3) also has a sampling error depending on m. However, as $m \to \infty$ this term distinguishes.
Rubin's rules require that $\hat{q}$ is (asymptotically) normally distributed around the population parameter q. This assumption is fulfilled for the mean, the standard error, regression coefficients and others, but for example not for a correlation coefficient. In case the assumption is not fulfilled, the pooling procedure needs to be adjusted as following:

1. Use Fisher's z transformation, see for example van Buuren (2012) p. 156, or any other suitable normalizing transformation for $\hat{q}_1, \ldots, \hat{q}_m$.
2. Do the pooling for the transformed $\hat{q}_1^*, \ldots, \hat{q}_m^*$ by Rubin's rules.
3. Do the back transformation, i.e. inverse of Fisher's z transformation.

Using the MI procedure with three phases has not only the advantage that we can account for the between-imputation variance. It is also possible to separate the imputation totally from the analysis. This allows for auxiliary variables in the imputation model that are not part of the subsequent analysis model. This can yield better results than having the restriction to use the analysis model variables only.
As the analysis and pooling phases are independent of imputation method used before, we will discuss several different imputation methods in the next sections. We will discuss the *Joint Modeling* and the *Fully Conditional Specification* and conclude the chapter about imputation methods with a copula based method called *CoImp*, which might only be understood after reading Chapter 3 about copulas first.

2.3.2 Joint Modeling (JM)

In the Joint Modeling (JM) imputation method we assume the variables have a joint distribution P with parameter set $\boldsymbol{\theta}$.

$$\boldsymbol{Y} \sim P(\boldsymbol{Y}|\boldsymbol{\theta}) \tag{2.7}$$

P can be any multivariate distribution, but the most studied case in literature is the multivariate normal model and often authors implicitly mean JM with a multivariate normal distribution when they talk about multiple imputation. The idea is to impute the missing values by taking random draws from the distribution $P(\boldsymbol{Y}_{mis}|\boldsymbol{Y}_{obs}, \boldsymbol{\theta})$. However in general $\boldsymbol{\theta}$ is unknown. Tanner and Wong (1987) gave a solution to this problem by introducing the Data Augmentation (DA) algorithm. The DA is referred to as a Bayesian version of the Expectation-Maximization (EM) algorithm and is a Markov chain Monte Carlo (MCMC) method. MCMC and EM algorithm in the DA background are explained in more detail in Tanner and Wong (2010). The DA is a two-step iterative procedure alternating between an imputation step (I-step) and a posterior step (P-step) in which the parameter estimates are updated.

Definition 2.1 (I-step and P-step in the DA algorithm)
I-step: Draw $\boldsymbol{Y}^{*(t)} \sim P(\boldsymbol{Y}_{mis}|\boldsymbol{Y}_{obs}, \boldsymbol{\theta})$,
P-step: Draw $\boldsymbol{\theta}^{*(t)} \sim P(\boldsymbol{\theta}|\boldsymbol{Y}_{obs}, \boldsymbol{Y}^{*(t)})$,
where $\boldsymbol{Y}^{*(t)}$ *is the imputed data from iteration* t *and* $\boldsymbol{\theta}^{*(t)}$ *the parameter estimate from iteration* t.

In words, the I-step imputes the missing data by taking random draws from the conditional distribution depending on the parameters from the prior P-step and the observed data. The P-step randomly draws a new parameter set from the respective posterior distributions. The result of this procedure is a Markov Chain $\{(\boldsymbol{Y}^{*(t)}, \boldsymbol{\theta}^{*(t)}),\ t = 1, 2, \dots\}$ with stationary distribution $f(\boldsymbol{\theta}, \boldsymbol{Y}_{mis}|\boldsymbol{Y}_{obs})$. Consequently we have to wait till the Markov Chain converges and should only take the imputed values after an appropriate burn-in period.
A best practice for the starting values of $\boldsymbol{\theta}$ is to take the estimates obtained by the EM algorithm.

Example 2.2 (Multivariate Normal DA)
In case of a d-dimensional multivariate normal model we have two parameters and specify $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. *Then,*

$$\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

The I-step depends on the observed data and the estimated mean vector and covariance matrix from the prior P-step. This step can be realized as the stochastic regression method described in Section 2.2.4. Adding a random error term to the conditional mean is basically the same as taking random draws from the conditional distribution. Schafer (1997) shows in Section 5.4.2 that the posterior distributions for the parameters in the P-step for the multivariate normal distribution are

$$P(\boldsymbol{\Sigma}|\hat{\boldsymbol{\mu}}, \boldsymbol{Y}^*, \boldsymbol{Y}_{obs}) \sim \mathcal{W}_{d-1}^{-1}(\hat{\boldsymbol{\Lambda}})$$
$$P(\boldsymbol{\mu}|\boldsymbol{\Sigma}^*, \boldsymbol{Y}^*, \boldsymbol{Y}_{obs}) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}, d^{-1}\boldsymbol{\Sigma}^*),$$

where $\hat{\boldsymbol{\mu}}$ *is the sample mean vector computed from the imputed data set of the prior I-step.* $\boldsymbol{Y}^*$ *is the imputed data from the prior I-step,* $\hat{\boldsymbol{\Lambda}}$ *is the sample sum of squares and cross product matrix* $Y^{*\prime}Y^*$, $\mathcal{W}_{d-1}^{-1}$ *is the inverse Wishart distribution function with* $d-1$ *degrees of freedom and* $\boldsymbol{\Sigma}^*$ *the random draw from the posterior distribution in the first line.*
A more detailed description of the algorithm that shows necessary computations step by step for the normal JM using DA can be found in van Buuren (2012) p. 106.
It seems not critical if the normality assumption is violated as several empirical studies, for example Demirtas et al. (2008), suggest.

Since the DA is a Markov Chain procedure it depends only on the information from the prior iteration and we can see the random draws from the distributions in the I-step and P-step are mutually dependent and therefore imputed values from consecutive iterations are dependent. Now there are two possible ways to handle this problem. Either we start a new Markov Chain for each imputation and take the values after an appropriate burn-in period, or we take the imputed values from one DA at regular intervals. An appropriate interval can be determined for example by checking several graphical diagnostics, like time series plots, worst linear function plots and autocorrelation function plots, see Enders (2010) Sections 7.9 and 7.10 for more. The interval length k should be of such kind that imputed values at iteration t and iterations at $t + k$ are independent. Starting a new Markov Chain for every imputation can fasten the procedure through parallel computing. The number of necessary imputations for JM with a multivariate normal model is discussed for example in Graham et al. (2007). They suggest that $40-100$ imputations might

be needed to ensure a certain level of statistical power even if the missing percentage is high.

2.3.3 Fully Conditional Specification (FCS)

Fully Conditional Specification (FCS), also known under various other names as "multiple imputation by chained equations (mice)" or "sequential regression", avoids to specify a joint distribution $P(\boldsymbol{Y}|\boldsymbol{\theta})$. Sometimes no standard multivariate distribution might fit the special data structure, or it is desirable to avoid the joint distribution if some variables are discrete or dichotomous and not continuous. Instead a more flexible variable-by-variable approach is used. Contrary to JM, FCS specifies the multivariate distribution through specifying a set of conditional distributions $P(\boldsymbol{Y}_j|\boldsymbol{Y}_{-j}, \theta_j)$ for each variable $j = 1, \ldots, d$. As JM, FCS is also an iterative MCMC method and is drawing randomly from posterior distributions. To achieve one imputed data set, the following iteration is performed repeatedly.

Definition 2.2 (Iterative step in the FCS)
The t-th iteration in the FCS method cycles through all variables Y_j for $j = 1 \ldots, d$ and, according to van Buuren et al. (2006), uses the Gibbs sampler to randomly draw

$$\begin{aligned}
\boldsymbol{\theta}_1^{*(t)} &\sim P(\boldsymbol{\theta}_1|\boldsymbol{Y}_{1,obs}, \boldsymbol{Y}_2^{*(t-1)}, \ldots, \boldsymbol{Y}_d^{*(t-1)}) \\
\boldsymbol{Y}_1^{*(t)} &\sim P(\boldsymbol{Y}_{1,mis}|\boldsymbol{Y}_{1,obs}, \boldsymbol{Y}_2^{*(t-1)}, \ldots, \boldsymbol{Y}_d^{*(t-1)}, \boldsymbol{\theta}_1^{*(t)}) \\
&\vdots \\
\boldsymbol{\theta}_d^{*(t)} &\sim P(\boldsymbol{\theta}_d|\boldsymbol{Y}_{d,obs}, \boldsymbol{Y}_1^{*(t)}, \ldots, \boldsymbol{Y}_{d-1}^{*(t)}) \\
\boldsymbol{Y}_d^{*(t)} &\sim P(\boldsymbol{Y}_{d,mis}|\boldsymbol{Y}_{d,obs}, \boldsymbol{Y}_1^{*(t)}, \ldots, \boldsymbol{Y}_{d-1}^{*(t)}, \boldsymbol{\theta}_d^{*(t)}),
\end{aligned}$$

where $\boldsymbol{Y}_j^{*(t-1)}$ is the imputed data vector for the variable j from step $t-1$.

The univariate imputation step can again be performed by using a preferred method. By default, the *mice* R-package, see van Buuren et al. (2015), uses a *predictive mean matching (pmm)* imputation method. Pmm is for example explained in Vink et al. (2014). In short, pmm imputes the missing vales with the values from the nearest neighbors, in the sense of expected values. This means, after fitting a model for the missing value conditional on the observed variables, the observations which are closest based on the expected, or predicted, value will be donors. The stochastic regression can also be used for the imputation step instead.
Van Buuren et al. (2006) suggest to use random draws from the observed data to obtain starting values for the iteration.
Following the described iterative procedure theoretically yields a draw from the joint distribution. However the joint distribution for the conditional distributions may not exist in theory. This issue is called incompatibility. Compatibility is a necessary requirement of the Gibbs sampler. If this requirement is not fulfilled the procedure may not converge. The convergence in the FCS is faster as in the DA and only $5 - 20$ iterations are needed,

according to van Buuren (2012) p.113. The convergence can be monitored by looking at the means, standard deviations and pairwise correlations of the variables, which should stay stable after some iterations. The speed of the convergence is directly and negatively influenced by high missing data rates and high correlations.
As each conditional distribution has to be specified, a lot of work and modeling effort comes along with the newly gained flexibility. Due to the fact that the joint distribution does not always exist theoretically, there is no sufficient theory and as van Buuren et al. (2006) say: *"little is known about the quality of the resulting imputations"*. Still it might be reasonable not to use the theoretically more grounded JM due to the possible increase in flexibility. In fact, FCS is a standard method in practice.

2.3.4 CoImp - a copula imputation method

The last imputation method we want to introduce is the *CoImp (CopulaImputation)* method from Di Lascio, F. Marta L. et al. (2014), also implemented in R, see Di Lascio, Francesca M. L. and Giannerini (2014). This imputation method uses a copula model and the reader should have knowledge about copulas or first read Chapter 3 of this thesis. The advantage of copula models is the potential to model the margins and the dependence structure separately. As we saw in Section 2.2 about single imputation methods, it is essential to model the dependence structure to achieve unbiased results for a parameter of interest as a regression slope and only the stochastic regression imputation was able to factor in a linear correlation. The CoImp imputation method is basically a two step approach.
Suppose we have given an incomplete data set X on d variables. In the first step the marginal densities $f_1, \dots, f_d$, as well as the multivariate copula model $c_{1:d}$ are estimated by the semiparametric two-step inference for margins (IFM) for complete cases. In the IFM the margins are estimated without any assumptions by means of the local log-likelihood function and the dependence parameter θ of the copula model $c_{1:d}$ is estimated by the pseudo-maximum log-likelihood method for rank transformed available complete observations. For more details see Di Lascio, F. Marta L. et al. (2014).
In the second step the conditional density functions are derived by using the copula model and Baye's rule. For a single missing variable, we define $-\boldsymbol{j} := \{1, \dots, d\} \backslash j$, $\boldsymbol{x_{-j}} := \{x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_d\}$, $u_i := F_i(x_i)$ and $\boldsymbol{u_{-j}} := \{u_1, \dots, u_{j-1}, u_{j+1}, \dots, u_d\}$ for $j \in \{1, \dots, d\}$. Then

$$f_{j|-\boldsymbol{j}}(x_j|\boldsymbol{x_{-j}}) = \frac{f_{1:d}(x_1, \dots, x_d)}{f_{-\boldsymbol{j}}(\boldsymbol{x_{-j}})} = \frac{c_{1:d}(u_1, \dots, u_d)}{c_{-\boldsymbol{j}}(\boldsymbol{u_{-j}})} f_j(x_j) = c_{j|-\boldsymbol{j}}(u_j|\boldsymbol{u_{-j}}) f_j x_j. \tag{2.8}$$

Equation (2.8) can be extended for a general number of conditioned variables k. Set $\boldsymbol{j}_{1:k} := \{j_1, \dots, j_k\}$ and $-\boldsymbol{j}_{1:k} := \{1, \dots, d\} \backslash \boldsymbol{j}_{1:k}$, as well as $\boldsymbol{x}_{-\boldsymbol{j}_{1:k}} := \{x_1, \dots, x_d\} \backslash \{x_{j_1}, \dots, x_{j_k}\}$ and $\boldsymbol{u}_{-\boldsymbol{j}_{1:k}} := \{u_1, \dots, u_d\} \backslash \{u_{j_1}, \dots, u_{j_k}\}$, for $j_1, \dots, j_k \in \{1, \dots, d\}$. It holds that

$$f_{\boldsymbol{j}_{1:k}|-\boldsymbol{j}_{1:k}}(x_{j_1}, \dots, x_{j_k}|\boldsymbol{x}_{-\boldsymbol{j}_{1:k}}) = c_{\boldsymbol{j}_{1:k}|-\boldsymbol{j}_{1:k}}(u_{j_1}, \dots, u_{j_k}|\boldsymbol{u}_{-\boldsymbol{j}_{1:k}}) f_{j_1}(x_{j_1}) \cdots f_{j_k}(x_{j_k}). \tag{2.9}$$

The missing values are then imputed by drawing from the suitable conditional density function (2.9). As it might not be feasible to derive the conditional density functions

analytically, a Hit or Miss Monte Carlo method is used, for details see Di Lascio, F. Marta L. et al. (2015).
The corresponding R-package, see Di Lascio, Francesca M. L. and Giannerini (2014), so far includes the Gaussian copula, the Frank copula, the Clayton copula and the Gumbel copula for the copula model. Di Lascio, F. Marta L. et al. (2015) show for these copula models how the conditional density functions are derived analytically for the bivariate case.
In contrast to the Joint Modeling or the Fully Conditional Specification there are no iterations changing the joint density function and the method relies on the first fit of the copula model on available data. Despite of this obvious disadvantage, copulas may have a high potential in the imputation of missing values as it is possible to separate the modeling of the dependence structure and the margins. Identifying the dependence structure and reproducing it for the incomplete cases might be the most important step in imputation. After we give a detailed introduction to copulas and vine copulas in the next chapter, we will introduce our own vine copula based imputation method.

2.4 Computational resources for single and multiple imputation

The methods presented in this chapter are all implemented in R. Of course sometimes more than once or not in the pure form as it was presented here. Adjustments and specifications are various. Table 2.1 includes some basic R functions which are helpful to perform the described single imputation methods, whereas Table 2.2 shows a possible choice of packages with their respective functions for the multiple imputation. Later on these packages will be used for the implementation of a benchmarking with our own method.

Method	Package	Function	Description
deletion	stats	complete.cases(…)	Returns all observations of a dataset, which are complete.
mean	base	mean(…)	Computes the mean of a vector.
	Hmisc	impute(…,fun=mean)	Imputes the missing values with the mean of each variable.
bootstrap	base	sample(…,replace=TRUE)	Draws a random sample from a vector with replacement.
regression	stats	lm(…)	Performs a linear regression analysis.
stoch. regr.	stats	rnorm(…)	Generates realizations of a univariate normal distribution.
knn	VIM	kNN(…)	Imputes a data set with the knn method.

Table 2.1: Useful R functions to implement the SI techniques described in Section 2.2.

Method	Package	Function	Description
JM	norm	prelim.norm(…)	Preliminary preparation sorting, centering and scaling the data.
	norm	em.norm(…)	Calculates starting vales using the EM algorithm.
	norm	da.norm(…)	Performs the DA.
FCS	mice	mice(…)	Performs the imputation using FCS.
CoImp	CoImp	CoImp(…)	Performs the CoImp imputation method.

Table 2.2: R-packages to implement the MI techniques described in Section 2.3.

Chapter 3

Copulas

In this chapter we will introduce the theory of copulas. We will focus on the presentation of the results, proofs will be left out or a reference will be given. For an extensive general introduction to copulas see Nelsen (2006). Our focus is just on introducing the minimal framework for the next chapters, especially on bivariate pair copula constructions, dependence measures and R-vines.

3.1 Copulas

First we start with some notation we will use throughout this chapter.

Definition 3.1 (Marginal, joint and conditional distributions)
Suppose we have given d random variables $\boldsymbol{X} = (X_1, \ldots, X_d)$ *then we denote the*

- *marginal density and distribution with* $f_i(x_i),\ F_i(x_i),\ i = 1, \ldots, d,$
- *joint density and distribution with* $f_{1:d}(x_1, \ldots, x_d)$ *and* $F_{1:d}(x_1, \ldots, x_d)$,
- *conditional density and distribution of* X_i *given* X_j, $i \neq j$, *with* $f_{i|j}(x_i|x_j)$ *and* $F_{i|j}(x_i|x_j)$.

A copula can be very simply introduced with the following definition.

Definition 3.2 (Copula)
A multivariate distribution function defined on $[0,1]^d$, *with uniformly distributed marginals, is called a copula. The d-dimensional copula will be denoted by* $C_{1:d}$ *and the corresponding density, which can be derived by partial differentiation, will be denoted by* $c_{1:d}$. *More precisely* $c_{1:d}(u_1, \ldots, u_d) := \frac{\partial^d}{\partial u_1 \ldots \partial u_d} C_{1:d}(u_1, \ldots, u_d)$, *for all* $u_1, \ldots, u_d \in [0,1]$.

For a given data $\boldsymbol{X}$, with known joint distribution function $F_{1:d}$, the uniformly distributed marginals as required for a copula can be obtained by doing the so called probability integral transform.

Definition 3.3 (Probability integral transform)
The transformation $u := F(x)$ is called the probability integral transform (PIT). If $X \sim F$, then $U := F(X)$ is uniformly distributed.

So from now on we will have an additional variable scale.

Definition 3.4 (Two variables scales)

- ***x-scale:*** *The original scale $(X_1, \ldots, X_d)$, with density $f_{1:d}(x_1, \ldots, x_d)$.*
- ***u-scale:*** *The copula scale $(U_1, \ldots, U_d)$, where $U_i := F_i(X_i)$, for $i = 1, \ldots, d$, and the copula density is $c_{1:d}(u_1, \ldots, u_d)$.*

The most central and fundamental result for copulas is Sklar's theorem. It allows to separate the handling and modeling of the one dimensional marginals from the multivariate dependence structure.

Theorem 3.5 (Sklar's Theorem)
Let $F_{1:d}$ be a be a d-dimensional joint distribution of a d-dimensional random vector $\boldsymbol{X}$ with marginal distribution functions F_i, $i = 1, \ldots, d$, then there exists a d-dimensional copula $C_{1:d}$ with density $c_{1:d}$, such that

$$\begin{aligned} F_{1:d}(x_1, \ldots, x_d) &= C_{1:d}(F_1(x_1), \ldots, F_d(x_d)), \\ f_{1:d}(x_1, \ldots, x_d) &= c_{1:d}(F_1(x_1), \ldots, F_d(x_d)) f_1(x_1) \cdots f_d(x_d). \end{aligned}$$

For absolutely continuous distributions the copula is unique.
Also the inverse holds. The copula corresponding to a multivariate distribution is given by

$$\begin{aligned} C_{1:d}(u_1, \ldots, u_d) &= F_{1:d}(F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d)), \\ c_{1:d}(u_1, \ldots, u_d) &= \frac{f_{1:d}(F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d))}{f_1(F_1^{-1}(u_1)) \cdots f_d(F_d^{-1}(u_d))}. \end{aligned}$$

A proof of Sklar's theorem can be found in Nelsen (2006).

The conditional densities and distribution functions of bivariate distributions can also be expressed in terms of their copulas.

Lemma 3.6 (Conditional densities and distribution functions of bivariate distributions in terms of their copulas)
Using the same notation as in Sklar's theorem for the bivariate case, the conditional density and distribution function of X_1 given X_2 can be expressed as

$$\begin{aligned} f_{1|2}(x_1|x_2) &= c_{12}(F_1(x_1), F_2(x_2)) f_1(x_1), \\ F_{1|2}(x_1|x_2) &= \frac{\partial}{\partial u_2} C_{12}(F_1(x_1), u_2)|_{u_2 = F_2(x_2)} \\ &= \frac{\partial}{\partial F_2(x_2)} C_{12}(F_1(x_1), F_2(x_2)). \end{aligned}$$

Furthermore

$$C_{1|2}(u_1|u_2) = \frac{\partial}{\partial u_2} C_{12}(u_1, u_2),$$
$$C_{1|2}(F_1(x_1)|F_2(x_2)) = \frac{\partial}{\partial u_2} C_{12}(F_1(x_1), u_2)|_{u_2=F_2(x_2)},$$

and for the inverse function holds

$$F_{1|2}^{-1}(u_1|x_2) = F_1^{-1}(C_{1|2}^{-1}(u_1|F_2(x_2))).$$

With this result we have already enough basics to simulate from a bivariate copula. However at first we introduce the h functions of bivariate copulas to simplify notation.

Definition 3.7 (h functions of bivariate copulas)
The two h functions of a bivariate copula are defined as

$$h_{1|2}(u_1|u_2) := \frac{\partial}{\partial u_2} C_{12}(u_1, u_2),$$
$$h_{2|1}(u_2|u_1) := \frac{\partial}{\partial u_1} C_{12}(u_1, u_2),\ u_1, u_2 \in [0, 1].$$

Theorem 3.8 (Simulation from a bivariate copula)
If we simulate for $i = 1, \ldots, n$

1. *u_{i1} from a uniform distribution on $[0, 1]$ and*
2. *u_{i2} from $C_{1|2}(\cdot|u_i 1)$ given by the appropriate h function,*

we obtain an i.i.d. sample from the bivariate copula C_{12}.

As already mentioned, central for copulas is the separation of the marginals and the multivariate dependence structure, so a tool to measure the dependence is necessary.

3.2 Dependence Measures

The most popular dependence measure is Pearson's correlation coefficient. However it is not invariant to monotone transformation of the margins. The PIT is such an monotone transformation and gives reason not to use Pearson's correlation coefficient. Instead we will use Kendall's tau which is rank based.

Definition 3.9 (Kendall's tau)
Let (X_{11}, X_{12}) and (X_{21}, X_{22}) be independent and identically distributed copies of the random vector (X_1, X_2). Then Kendall's tau is defined as the difference of the probability of concordance and the probability of discordance.

$$\tau_{1,2} := \tau(X_1, X_2) = P((X_{11} - X_{21})(X_{12} - X_{22}) > 0) - P((X_{11} - X_{21})(X_{12} - X_{22}) < 0)$$

An estimate of Kendall's tau for a realization of the random vector (X_1, X_2) is given by

$$\hat{\tau}_{1,2} := \frac{N_c - N_d}{\binom{N}{2}}$$

where N_d and N_c denote the number of discordant and concordant pairs respectively while N is the number of all observations, so that $\binom{N}{2}$ is the total number of distinct pairs.

Kendall's tau can also be expressed in terms of the copula.

Theorem 3.10 (Kendall's τ expressed in terms of the copula)
For a random vector (X_1, X_2) with bivariate distribution copula C_{12}, the empirical Kendall's tau of (X_1, X_2) can be expressed as

$$\tau_{1,2} = 4 \int_{[0,1]^2} C_{12}(u_1, u_2) dC_{1,2}(u_1, u_2) - 1.$$

So far we only looked at the Kendall's tau dependence measure. However one measure might not be enough. In case that two copulas yield similar results regarding Kendall's tau value, we should look at another measure. One scalar measure is the tail dependence coefficient, which measures the probability of mutual occurrence of extremely high or low values of two variables.

Definition 3.11 (Upper/lower tail dependence)
For a bivariate copula C the upper/lower tail dependence coefficients are defined as

$$\lambda_{upper} := \lim_{t \nearrow 1} P(X_2 > F_2^{-1} | X_1 > F_1^{-1}(t)) = \lim_{t \nearrow 1} \frac{1 - 2t + C(t,t)}{1 - t},$$

$$\lambda_{lower} := \lim_{t \searrow 0} P(X_2 \leq F_2^{-1} | X_1 \leq F_1^{-1}(t)) = \lim_{t \searrow 0} \frac{C(t,t)}{t}.$$

For many copula families the tail dependence coefficients can be derived analytically, if they exist. If $\lambda_{upper/lower} = 0$, we say that the copula has no upper/lower tail dependence. Since the tail dependence coefficients are defined as limits, it is not possible to derive sample versions. Krupskii and Joe (2014) suggest to use another tail-weighted measure of dependence.

Definition 3.12 (Alternative upper/lower tail dependence measure)
For $(U_1, U_2) \sim C$, with C being a bivariate copula, define the tail dependence measures

$$\begin{aligned}
\varrho_L(a,p) =& Cor\left[a\left(1 - \frac{U_1}{p}\right), a\left(1 - \frac{U_2}{p}\right) \middle| U_1 < p,\ U_2 < p \right], \\
\varrho_U(a,p) =& Cor\left[a\left(1 - \frac{1 - U_1}{p}\right), a\left(1 - \frac{1 - U_2}{p}\right) \middle| 1 - U_1 < p,\ 1 - U_2 < p \right],
\end{aligned}$$

where $p \leq 0.5$ is a truncation level, and $a(\cdot)$ is a weighting function mapping from $[0,1] \to (0, \infty)$, which is monotone increasing continuous, with $a(0) = 0$.

The properties of $a(\cdot)$ ensure that more weight is on the tails and the maximum (minimum) is obtained only when perfect comonotonic (countermonotonic) dependence is in the tails. For the calculation of these tail dependence measures we do not need any limits as they are defined as the correlation of transformed variables. As a consequence it is straight forward to derive sample versions.

Definition 3.13 (Empirical upper/lower tail dependence measure)
For an i.i.d. sample $(X_{i1}, X_{i2})_{i\in 1:n}$ from the bivariate copula distribution $C_{1,2}$, define the empirical upper and lower tail dependence measures

$$\begin{aligned}\hat{\varrho}_L(a,p) =& \widehat{Cor}\left[a\left(1-\frac{R_{i1}}{p}\right), a\left(1-\frac{R_{i2}}{p}\right)\middle| R_{i1}<p,\ R_{i2}<p\right],\\ \hat{\varrho}_U(a,p) =& \widehat{Cor}\left[a\left(1-\frac{1-R_{i1}}{p}\right), a\left(1-\frac{1-R_{i2}}{p}\right)\middle| 1-R_{i1}<p,\ 1-R_{i2}<p\right],\end{aligned}$$

with $R_{ij} := (rank(X_{ij})-0.5)/n$, for $j = 1, 2$, and $\widehat{Cor}(\cdot,\cdot)$ denoting the sample correlation coefficient.

The rank transformation ensures that the marginal distributions are uniform.

Krupskii and Joe (2014) suggest to choose the power function $a(u) = u^6$. With this choice of a power function for $a(\cdot)$, $\hat{\varrho}_L(a,p)$ and $\hat{\varrho}_U(a,p)$ are asymptotically normal and using the 6th power maximizes $\Delta/\sigma(\hat{\Delta})$ among the power functions. Here $\Delta := \Delta(a,p,C_1,C_2) := \varrho_L(a,p,C_1) - \varrho_L(a,p,C_2)$, $\hat{\Delta} := \hat{\Delta}(a,p,C_1,C_2) := \hat{\varrho}_L(a,p,C_1) - \hat{\varrho}_L(a,p,C_2)$ and $\sigma(\hat{\Delta})$ denotes its standard error. So this choice ensures that for two copulas with a similar Kendall's tau, the difference in the tail dependence measure is large, while the variance of the estimate is small. Further a good choice for p is 0.5, as other choices produce higher standard errors.

3.3 Pair copula constructions

In Appendix A we present some of the most popular bivariate copula families. There are of course multivariate extensions of many copula families, but we will focus on constructing multivariate copulas with bivariate building blocks using conditioning, as in Aas et al. (2009). In three dimensions we can decompose the joint density for example as

$$f(x_1,x_2,x_3) = f_{3|12}(x_3|x_1,x_2)f_{2|1}(x_2|x_1)f_1(x_1). \tag{3.1}$$

From this decomposition it follows, using Sklar's theorem and Lemma 3.6, that we can express the joint density in terms of bivariate pair copulas.

Definition 3.14 (Pair copula decomposition in three dimensions)
A three dimensional joint density can be decomposed in terms of pair copulas as

$$\begin{aligned}f(x_1,x_2,x_3) =& c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2) \times c_{23}(F_2(x_2), F_3(x_3)) \\ &\times c_{12}(F_1(x_1), F_2(x_2)) f_3(x_3) f_2(x_2) f_1(x_1)\end{aligned} \tag{3.2}$$

where $c_{13;2}(\cdot,\cdot;x_2)$ denotes the copula density associated with the conditional distribution of (X_1, X_3) given $X_2 = x_2$.

Note that this decomposition is not unique, as (3.1) is not a unique decomposition. Another pair copula decomposition of a three dimensional joint density is given by

$$\begin{aligned} f(x_1,x_2,x_3) =& c_{23;1}(F_{2|1}(x_2|x_1), F_{3|1}(x_3|x_1); x_1) \times c_{12}(F_1(x_1), F_2(x_2)) \\ & \times c_{13}(F_1(x_1), F_3(x_3)) f_3(x_3) f_2(x_2) f_1(x_1). \end{aligned}$$

In Definition 3.14 $c_{13;2}(\cdot,\cdot;x_2)$ depends on x_2. For simplification and better application we assume throughout this thesis that this does not hold. This is called the simplifying assumption.

Definition 3.15 (Simplifying assumption in the tree dimension)
In three dimensions the simplifying assumption holds, when

$$c_{13;2}(\cdot,\cdot;x_2) = c_{13;2}(\cdot,\cdot) \text{ for all } x_2.$$

Assuming the simplifying assumption, the copula corresponding to (X_1, X_3) given $X_2 = x_2$ can be expressed as

$$C_{13;2}(u_1,u_3|x_2) = F_{13|2}(F^{-1}_{1|2}(u_1|x_2), F^{-1}_{3|2}(u_3|x_2)).$$

The corresponding density is given then by

$$c_{13;2}(u_1,u_3|x_2) = \frac{c_{123}(F_1(F^{-1}_{1|2}(u_1|x_2)), F_2(x_2), F_3(F^{-1}_{3|2}(u_3|x_2)))}{c_{12}(F_1(F^{-1}_{1|2}(u_1|x_2)), F_2(x_2) c_{23}(F_2(x_2), F_3(F^{-1}_{3|2}(u_3|x_2)))}.$$

For the copula corresponding to (U_1, U_3) given $U_2 = u_2$ it holds

$$c^u_{132}(u_1,u_3|u_2) = c_{13;2}(u_1,u_3|x_2) \text{ for } u_2 := F_2(x_2).$$

Another application of the simplifying assumption, is the construction of a multivariate joint density using (parametric) bivariate copulas. We present how this works in three dimensions.

Definition 3.16 (Pair copula construction of a joint density in three dimensions)
Given the parametric bivariate copulas $c_{13;2}(\cdot,\cdot;\theta_{13;2})$, $c_{12}(\cdot,\cdot;\theta_{12})$ and $c_{23}(\cdot,\cdot;\theta_{23})$, a parametric pair copula construction in three dimensions gives the following three dimensional density with parameter vector $\boldsymbol{\theta} = (\theta_{13;2}, \theta_{23}, \theta_{12})$

$$\begin{aligned} f(x_1,x_2,x_3;\boldsymbol{\theta}) =& c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); \theta_{13;2}) \\ & \times c_{23}(F_2(x_2), F_3(x_3); \theta_{23}) \times c_{12}(F_1(x_1), F_2(x_2); \theta_{12}) \\ & \times f_3(x_3) f_2(x_2) f_1(x_1) \end{aligned} \tag{3.3}$$

In the same way this works for a copula density.

Definition 3.17 (Pair copula construction of a three dimensional parametric copula)
A copula family in three dimensions can be constructed as

$$\begin{aligned} c(u_1, u_2, u_3; \theta_{13;2}, \theta_{23}, \theta_{12}) :=& c_{13;2}(C_{1|2}(u_1|u_2), C_{3|2}(u_3|u_2); \theta_{13;2}) \\ & \times c_{23}(u_2, u_3; \theta_{23}) \times c_{12}(u_1, u_2; \theta_{12}) \end{aligned} \tag{3.4}$$

where $C_{1|2}(\cdot|u_2)$ and $C_{3|2}(\cdot|u_2)$ are the conditional distribution functions of U_1 given $U_2 = u_2$ and U_3 given $U_2 = u_2$ respectively.

So we saw how decompositions and constructions of joint densities work in the three dimensional case. In d dimensions one possible decomposition satisfies

$$f_{1:d}(x_1, \ldots, x_d) = f_{d|1:d-1}(x_d|x_1, \ldots, x_{d-1}) \cdots f_{2|1}(x_2|x_1) f_1(x_1). \tag{3.5}$$

However this decomposition is again not unique as in the three dimensional case. In fact the number of possible decompositions increases with the dimension d.
For higher dimensions we need some notation.

Definition 3.18 (Bivariate associated and conditional copulas)
For a set of random variables $(X_1, \ldots, X_d)$ the following notation will be used.

- *Let $D \subsetneq \{1, \ldots, d\}$, with $i, j \notin D$. Then $C_{ij;D}(\cdot, \cdot; \boldsymbol{x}_D)$ denotes the bivariate conditional distribution of (X_i, X_j) given that $\boldsymbol{X}_D = \boldsymbol{x}_D$.*
- *The bivariate conditional distribution of (U_i, U_j) given $\boldsymbol{U}_D = \boldsymbol{u}_D$ is denoted by $C_{ij;D}(\cdot, \cdot; \boldsymbol{u}_D)$.*
- *For distinct indices $i, j, i_1, \ldots, i_k$ with $i < j$ and $i_1 < \cdots < i_k$ we use the abbreviation*

$$c_{i,j;\boldsymbol{i}_{1:k}} := c_{i,j;i_1,\ldots,i_k}(F_{i|\boldsymbol{i}_{1:k}}(x_i|x_{i_1}, \ldots, x_{i_k}), F_{j|\boldsymbol{i}_{1:k}}(x_j|x_{i_1}, \ldots, x_{i_k}); x_{i_1}, \ldots, x_{i_k}) \tag{3.6}$$

Now we are able to extend the theory to construct multivariate densities by pair copulas. As we saw already in three dimensions, different decompositions yield different pair copula constructions of a joint density. To be able to analyze these different decomposition structures in higher dimensions, for example to find the best fitting pair copula construction for a data set, we need some framework and notation. This is introduced in the next section about regular (R-)vine copulas.
However before we conclude this section we will present one important result, shown by Joe (1996), which is useful for the analytical determination of conditional distribution functions.

Theorem 3.19 (Recursion for conditional distribution functions)
Let F be the absolutely continuous joint distribution of a random variable X and a random vector $\boldsymbol{Y}$. Let $Y_j \in \boldsymbol{Y}$ and define $\boldsymbol{Y}_{-j} := \boldsymbol{Y} \backslash Y_j$ as the subset of $\boldsymbol{Y}$ with Y_j removed. Then $F_{X|\boldsymbol{Y}}$, the conditional distribution of X given $\boldsymbol{Y} = \boldsymbol{y}$, satisfies the following recursion

$$F_{X|\boldsymbol{Y}}(\cdot|\boldsymbol{y}) = \frac{\partial C_{X,Y_j|\boldsymbol{Y}_{-j}}(F_{X|-\boldsymbol{Y}_j}(x|\boldsymbol{y}_{-j}), F_{Y_j|-\boldsymbol{Y}_j}(y|\boldsymbol{y}_{-j}))}{\partial F_{Y_j|\boldsymbol{Y}_{-j}}(y_j|\boldsymbol{y}_{-j})} \tag{3.7}$$

where $C_{X,Y_j|\boldsymbol{Y}_{-j}}(\cdot, \cdot; \boldsymbol{y}_{-j})$ is the copula corresponding to (X, Y_j) given $\boldsymbol{Y}_{-j} = \boldsymbol{y}_{-j}$.

3.4 Regular vines (R-vines)

In this section a little basic knowledge on graphs is required, especially about trees. For an extensive introduction on graph theory see for example Gross and Yellen (2006) Chapter 1-4. With this basics we will start introduce R-vine distributions, following the notation, definitions and results in Czado and Stoeber (2012).

Definition 3.20 (Regular vine tree sequence)
$\mathcal{V} = (T_1, \dots, T_{d-1})$ is a regular vine tree sequence on d elements if,

1. *T_j is connected, for $j = \{1, \dots, d-1\}$.*
2. *T_1 is a tree with nodes $N_1 = \{1, \dots, d\}$ and a set of edges E_1.*
3. *T_j is a tree with nodes $N_j = E_{j-1}$ and edges E_j, for $j = \{2, \dots, d-1\}$.*
4. *For $j = 2, \dots, d-1$ and $\{a, b\} \in E_j$ it must hold that $|a \cap b| = 1$. This is called the proximity condition.*

The proximity condition implies that if there is an edge between a and b in T_j, for $j \in \{2, \dots, d-1\}$, then a and b as edges in T_{j-1} must both be connected to a common node.

Example 3.1 (4-dimensional R-vine)
Figure 3.1 gives an illustration of one possible 4-dimensional R-vine tree sequence.

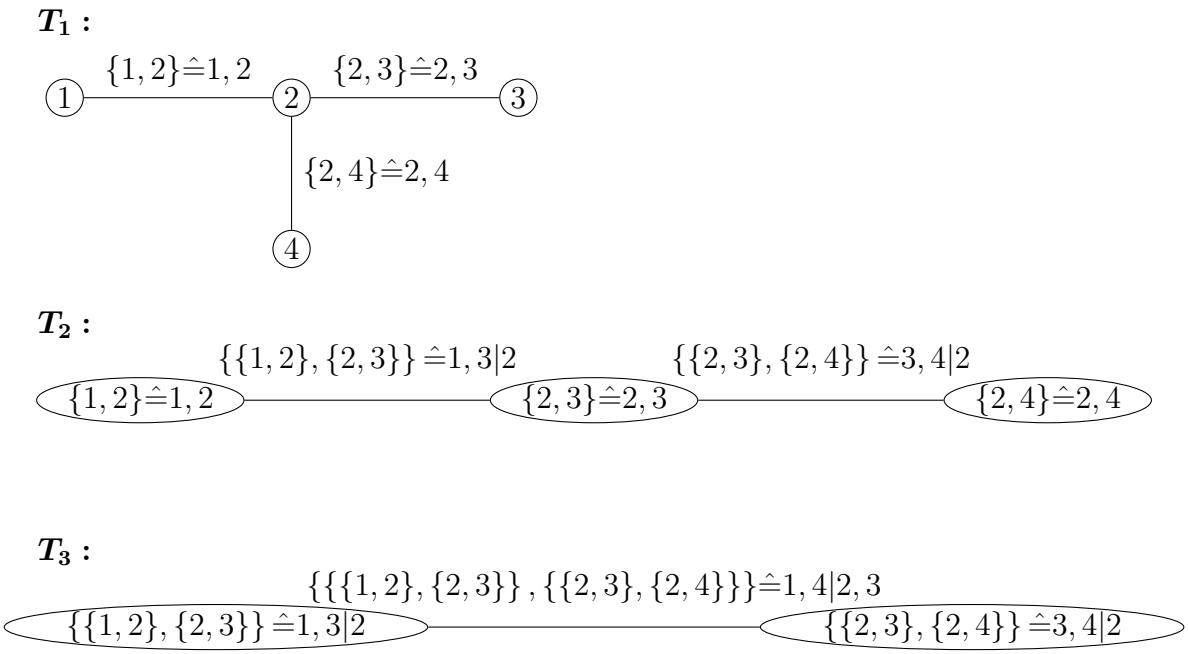


Figure 3.1: 4 dimensional R-vine tree sequence.

For a shorter notation and the assignment of copulas to edges, we will introduce some new terms and notation.

Definition 3.21 (Complete union and conditioned sets)
For $e \in E_i$ define define the complete union A_e of e as

$$A_e := \{j \in N_1 | \exists e_1 \in E_1, \dots, e_{i-1} \in E_{i-1} : j \in e_1 \in \dots \in e_{i-1} \in e\}.$$

The conditioning set of an edge $e = \{a, b\}$ is

$$D_e := A_a \cap A_b$$

and the conditioned sets are given by

$$C_{e,a} := A_a \setminus D_e \quad \text{and} \quad C_{e,b} := A_b \setminus D_e,$$
$$C_e := C_{e,a} \cup C_{e,b}$$

Example 3.1 (Continued)
For the edge $e = \{\{1, 2\}, \{2, 3\}\}$ in Figure 3.1, $A_e = \{1, 2, 3\}$. $A_a = \{1, 2\}$ and $A_b = \{2, 3\}$, therefore $D_e = \{2\}$. Finally $C_e = \{1, 3\}$.

With this definitions on hand we can now define an R-vine distribution, introducing a stochastic link between the graph theory based R-vine tree sequence and copulas.

Definition 3.22 (Regular vine distribution)
A d-dimensional joint distribution $F_{1:d}$ on the random variables $X_1, \dots, X_d$ is called regular vine distribution, if there exists a tuple $(\mathcal{F}, \mathcal{V}, \mathcal{B})$, such that

1. *$\mathcal{F} = (F_1, \dots, F_d)$ is a vector of continuous invertible marginal distribution functions, representing the marginal distribution functions of $X_i, i = 1, \dots, d$.*
2. *$\mathcal{V}$ is an R-vine tree sequence on d elements.*
3. *$\mathcal{B} = \{B_e | i = 1, \dots, d-1; e \in E_i\}$ is a set of copulas, where each B_e is a symmetric bivariate copula with density and E_i are the edge sets of the corresponding tree T_i in the R-vine tree sequence.*
4. *For each $e \in E_i, i = 1, \dots, d-1, e = \{a, b\}$, B_e is the corresponding copula to the conditional distribution of $X_{C_{e,a}}$ and $X_{C_{e,b}}$ given $\boldsymbol{X}_{D_e}$. $B_e(.,.)$ does not depend on $\boldsymbol{x}_{D_e}$.*

The copula B_e corresponding to edge e will be denoted by $C_{C_{e,a},C_{e,b};D_e}$ and the corresponding density by $c_{C_{e,a},C_{e,b};D_e}$ respectively.

Example 3.1 (Continued)
The copula B_e for the edge $e = \{\{1, 2\}, \{2, 3\}\}$ in Figure 3.1 is the corresponding copula for the conditional bivariate distribution of (X_1, X_3) given $X_2 = x_2$ and it holds

$$F_{1,3|2}(x_1, x_3|x_2) = B_e(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2)).$$

Fortunately, there is a result which ensures the existence of such an R-vine distribution.

Theorem 3.23 (Existence of a regular vine distribution)
Let $(\mathcal{F}, \mathcal{V}, \mathcal{B})$ have properties 1.-3. of Definition 3.22. Then there is a unique distribution with density

$$f_{1,\dots,d} = f_1 \cdots f_d \cdot \prod_{i=1}^{d-1} \prod_{e \in E_i} c_{C_{e,a}C_{e,b}|D_e}(F_{C_{e,a}|D_e}, F_{C_{e,b}|D_e}) \tag{3.8}$$

such that for each $e \in E_i$, $i = 1, \dots, d-1$, with $e = \{a, b\}$, we have for the distribution function of $X_{C_{e,a}}$ and $X_{C_{e,b}}$ given $\mathbf{X}_{D_e}$

$$F(x_{C_{e,a}}, x_{C_{e,b}}|\mathbf{x}_{D_e}) = B_e(F(x_{C_{e,a}}|\mathbf{x}_{D_e}), F(x_{C_{e,b}}|\mathbf{x}_{D_e})), \tag{3.9}$$

and the marginal distributions are given by $F_i(x_i)$, for $i = 1, \dots, d$.

A proof of this theorem is given in Bedford and Cooke (2002).

Now we know about the existence of R-vine distributions, the definition of R-vine copulas can be done. We solely specify the marginal distributions and say, that a regular vine copula is a regular vine distribution, where all margins are uniformly distributed on $[0, 1]$.

Definition 3.24 (Log-likelihood of an R-vine)
The log-likelihood of a d-dimensional R-vine for some observations $\boldsymbol{u} = (\boldsymbol{u}_1', \dots, \boldsymbol{u}_n')'$ is given by

$$l(\boldsymbol{\theta}, \boldsymbol{u}) = \sum_{i=1}^{n} \sum_{l=1}^{d-1} \sum_{e \in E_l} \ln[c_{j(e),k(e)|D(e)}(F(u_{i,j(e)}|\boldsymbol{u}_{i,D(e)}), F(u_{i,k(e)}|\boldsymbol{u}_{i,D(e)})|\boldsymbol{\theta}_{j(e),k(e)|D(e)})],$$

where E_l is the edge set of tree l, for $l = 1, \dots, d-1$, $\boldsymbol{u}_i = (u_{i,1}, \dots, u_{i,d})' \in [0, 1]^d$, $i = 1, \dots, n$ and $c_{j(e),k(e)|D(e)}$ is denoting the bivariate copula corresponding to an edge e with parameters $\boldsymbol{\theta}_{j(e),k(e)|D(e)}$.

We saw already in Figure 3.1 how an R-vine tree structure can be illustrated. Another representation, suitable for computations and computer handling, is the R-vine matrix. We will follow here the notation as in Dißmann et al. (2013).

Definition 3.25 (Regular Vine Matrix)
Let M be a lower triangular matrix with entries $m_{i,j}, i \geq j$. Each entry $m_{i,j}$ is allowed to take integer values from 1 to d. M is called a regular vine matrix if for $i = 1, \dots, d-1$ and for all $k = i+1, \dots, d-1$ there is an j in $i+1, \dots, d-1$ satisfying

$$\begin{aligned} \{m_{k,i}, \{m_{k+1,i}, \dots, m_{d,i}\}\} &= \{m_{j,j}, \{m_{k,j}, \dots, m_{d,j}\}\} \text{ or} \\ \{m_{k,i}, \{m_{k+1,i}, \dots, m_{d,i}\}\} &= \{m_{k,j}, \{m_{k+1,j}, \dots, m_{d,j}, m_{j,j}\}\} \end{aligned} \tag{3.10}$$

A R-vine matrix defined as above has following properties, see Dißmann et al. (2013).

Property 3.26 (Properties of an R-vine matrix)

1. *Every column contains all the entries that a column to the right contains.*
2. *There is a new entry on the diagonal in every column.*
3. *All elements in a column are different.*
4. *Deleting the first row and column from a d-dimensional R-vine matrix gives a* $(d-1)$*-dimensional R-vine matrix.*

Between the graphical illustration and the R-vine matrix representation is a relation, that can be used to derive one from the other. In the following, we will present two algorithms. The first shows how to find the tree sequence given the matrix, and the second algorithm does the derivation vice versa.

Algorithm 3.27 (Construction of a tree sequence from an R-vine matrix M**)**
Input : *A* $d-$*dimensional regular vine matrix* M
Output: *The corresponding regular vine tree sequence*

$N_1 := \{1, \dots, d\}$
$E_2 := \{\}, \dots, E_{d-1} := \{\}$
$E_1 := \{m_{d-1,d-1}, m_{d,d-1}\}$
for $i = d-2, \dots, 1$ **do**
  $e_1^i := \{m_{i,i}, m_{d,i}\}$
  $E_1 := E_1 \cup \{e_1^i\}$
  for $k = 1, \dots, -(i-d)-1$ **do**
    Select $a_k \in E_k$ *with* $A_{a_k} = \{m_{d,i}, \dots, m_{d-k,i}\}$
    $e_{k+1}^i := \{e_k^i, a_k\}$
    $E_{k+1} := E_{k+1} \cup e_{k+1}^i$
  end
end
$\mathcal{V} := (T_1 := (N_1, E_1), T_2 := (E_1, E_2), \dots, T_{d-1} := (E_{d-2}, E_{d-1}))$

Algorithm 3.28 (Computing a regular vine matrix for a regular vine tree sequence $\mathcal{V}$)

Input : *A regular vine tree sequence* $\mathcal{V} = (T_1, \dots, T_{d-1})$
Output: *The corresponding* $d-$*dimensional regular vine matrix* M

$\mathcal{X} := \{\}$
for $i = d, \dots, 3$ **do**
  Choose $x, \tilde{x}, D$ *with* $x,\ \tilde{x} \notin \mathcal{X}$ *and* $|D| = i - 2$ *such that there is an edge* e *with* $C_e = \{x, \tilde{x}\}, D_e = D$
  $m_{d+1-i,d+1-i} := x,\ m_{d+2-i,d+1-i} := \tilde{x}$
  for $k = i - 2, \dots, 1$ **do**
    Choose $\hat{x}$ *such that there is an edge* e *with* $C_e = \{x, \hat{x}\}$ *and* $|D_e| = k - 1$
    $m_{d+1-k,d+1-i} := \hat{x}$
  end
  $\mathcal{X} := \mathcal{X} \cup \{x\}$
end
Choose $x, \tilde{x} \in \{1, \dots, d\} \backslash \mathcal{X}$
$m_{d-1,d-1} := x,\ m_{d,d-1} := \tilde{x},\ m_{d,,d} := \tilde{x}$
$M := (m_{k,i} | i = 1, \dots, d,\ i \leq d + 1 - k)$

Example 3.1 (Continued)
A possible R-vine matrix for the tree sequence in Figure 3.1 is given by

$$\begin{pmatrix} 1 & & & \\ 4 & 4 & & \\ 3 & 3 & 3 & \\ 2 & 2 & 2 & 2 \end{pmatrix}.$$

The first column can be read as:
There is an edge $e_1 = 1, 4|2, 3$ *in* $\boldsymbol{T_3}$*, an edge* $e_2 = 1, 3|2$ *in* $\boldsymbol{T_2}$ *and an edge* $e_2 = 1, 2$ *in* $\boldsymbol{T_1}$.

Now that R-vine copulas and their representations were introduced, we can use them as our main tool for the development of an own imputation algorithm in the next chapter, after we conclude this introduction with some remarks on R-vine fitting and simulation. When it comes to fitting an R-vine distribution to a data set, Dißmann et al. (2013) provided an algorithm that finds the tree structure, copula families and parameters, based on Kendall's tau dependence measure. We will not explain this algorithm here in detail and refer to the publication for more details.
More crucial for our imputation method will be the simulation from R-vine copulas, especially conditional simulation. This topic is addressed in the next section.

3.5 Simulating regular vine copula distributions

In general sampling from a multivariate joint distribution can be realized using the *Inverse Transform Sampling*.

Theorem 3.29 (Inverse Transform Sampling)
To obtain a random vector $(X_1, \dots, X_d)$ *with joint distribution function* $F_{1:d}$ *and marginal distribution functions* $F_1, \dots, F_d$*, do*

0. *Simulate* $U_1, \dots, U_d \sim \mathcal{U}[0,1]$ *iid*
1. *Set* $X_1 = F_1^{-1}(U_1)$
2. *Set* $X_2 = F_{2|1}^{-1}(U_2|X_1)$
3. *Set* $X_3 = F_{3|12}^{-1}(U_3|X_1, X_2)$

$\vdots$

m. *Set* $X_d = F_{d|1,\dots,d-1}^{-1}(U_d|X_1, \dots, X_{d-1})$

where $F_{2|1}$*,* $F_{3|12}$ *to* $F_{d|1,\dots,d-1}$ *correspond to the respective conditional distribution functions. Then* $(X_1, \dots, X_d)$ *has joint distribution function* $F_{1:d}$ *with marginal distribution functions* $F_1, \dots, F_d$*.*

As a copula is a multivariate distribution, this works in the same way for copulas.
Using Theorem 3.29 together with 3.19 and the h-functions from 3.7, Dißmann (2010) developed an algorithm to simulate from a general R-vine distribution. The details can be seen in the thesis and will not be presented here. For us it is sufficient that such an algorithm exists and can be used.
In the context of imputation, conditional simulation is important. For observations in a d-dimensional data set, where the variables$X_1, \dots, X_k$ are observed and the variables $X_{k+1}, \dots, X_d$ are missing, we need to simulate from $F_{k+1:d|1:k}$. Theorem 3.29 can be easily adjusted for this case. In Steps 1 to k we just fill in the given values, i.e. $X_1 = x_1, \dots, X_k = x_k$. Then we proceed as before.
Also the algorithm of Dißmann (2010) for the simulation from an R-vine can be adjusted to incorporate observed values. However not every conditional distribution $F_{k+1:d|1:k}$ can be derived in a closed form. Which conditional distribution can be derived in a closed form will be worked out in the next chapter.

Chapter 4

Vine Copula Imputation

4.1 Ad hoc imputable missing data patterns

Suppose we have given a specific d-dimensional R-vine distribution including the marginal distributions, bivariate copulas and parameter estimates of the copulas. Then the question is which variables can be missing, so that we can simulate them using the given R-vine distribution together with the observed variables.
Throughout this chapter we will stick to a 5-dimensional example:

Example 4.1 (5-dimensional R-vine)
Suppose we have given the following R-vine tree structure

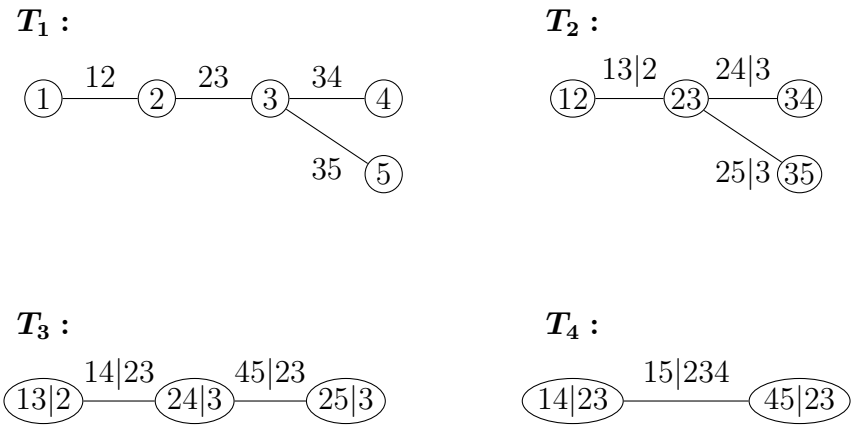


Figure 4.1: 5 dimensional R-vine tree structure.

with corresponding R-vine matrix

$$M = \begin{pmatrix} 5 & & & & \\ 1 & 1 & & & \\ 4 & 4 & 4 & & \\ 2 & 3 & 2 & 2 & \\ 3 & 2 & 3 & 3 & 3 \end{pmatrix} \tag{4.1}$$

and all bivariate copulas and parameter estimates $c_{15;234}$, $c_{14;23}$, $c_{45;23}$, $c_{13;2}$, $c_{24;3}$, $c_{25;3}$, c_{12}, c_{23}, c_{34} *and* c_{15}.
Then the joint density can be derived from Theorem 3.23 as

$$f_{1:5}(x_1,x_2,x_3,x_4,x_5) = c_{15;234}\cdot c_{14;23}\cdot c_{45;23}\cdot c_{13;2}\cdot c_{24;3}\cdot c_{25;3} \cdot c_{12}\cdot c_{23}\cdot c_{34}\cdot c_{15}\cdot f_1(x_1)\cdot f_2(x_2)\cdot f_3(x_3)\cdot f_4(x_4)\cdot f_5(x_5). \tag{4.2}$$

4.1.1 R-vine matrices of diagonal form

Lemma 4.1 (Diagonal form of R-vine matrix)
For every d-dimensional R-vine distribution, the corresponding R-vine matrix can be stated in the form that the diagonal $d = (v_1, v_2, \ldots, v_n)\prime$, *and the lower diagonal* $ld = (v_2, \ldots, v_n)\prime$. *The remaining elements are determined by the R-vine tree structure. We say that the R-vine matrix of this form has diagonal form. After a possible permutation/relabeling of the variables it looks like:*

$$\begin{pmatrix}
d & & & & & & & \\
d-1 & d-1 & & & & & & \\
* & d-2 & d-2 & & & & & \\
* & * & \ddots & \ddots & & & & \\
 & & & & \ddots & \ddots & & \\
* & & & & * & 2 & 2 & \\
* & * & & & * & * & 1 & 1
\end{pmatrix}$$

Proof: W.l.o.g. we found the first column of the R-vine matrix to be $(v_1, v_2, \ldots, v_d)\prime$ by applying Algorithm 3.28. Using the notation from the algorithm, this means in the first iteration $x = v_1$ and $\tilde{x} = v_2$. From this we can conclude the existence of the edges $(v_1, v_2|v_3, \ldots, v_d)$ and $(v_1, v_3|v_4, \ldots, v_d)$. Using the proximity condition backwards together with Property 3.26, that the diagonal element v_1 does not appear in any further column on the right, and the fact that the R-vine matrix is a lower diagonal matrix implies that there exists also an edge $(v_2, v_*|\{v_3, \ldots, v_d\}\backslash v_*)$. This means in the next step of Algorithm 3.28 we can choose $x := v_2$ and $\tilde{x} := v_*$. So we showed that v_2 is the diagonal element in the second column of the R-vine matrix. Repeating the argumentation now for the other columns from left to the right yields the claim. □

Corollary 4.2 *For every d-dimensional R-vine distribution, there are exactly two corresponding R-vine matrices* M^1, M^2 *of diagonal form. Furthermore* $M^1_{1,1} = M^2_{2,1}$ *and* $M^2_{1,1} = M^1_{2,1}$.

Proof: There are two alternatives for the $(1,1)$-element of the R-vine matrix according to Algorithm 3.28. The rest of the first column and the other columns are then determined by the R-vine tree structure applying Algorithm 3.28 and the restriction that the R-vine matrix shall be of diagonal form. □

Example 4.1 (Continued)
The R-vine matrix of diagonal form

$$M' = \begin{pmatrix} 1 & & & & \\ 5 & 5 & & & \\ 4 & 4 & 4 & & \\ 3 & 2 & 2 & 2 & \\ 2 & 3 & 3 & 3 & 3 \end{pmatrix}$$

is describing the same R-vine tree structure as the R-vine matrix M in (4.1). As M is also of diagonal form, M and M' are the two only R-vine matrices of diagonal form for this specific structure according to Corollary 4.2
The R-vine matrix

$$M'' = \begin{pmatrix} 5 & & & & \\ 1 & 4 & & & \\ 4 & 1 & 3 & & \\ 2 & 2 & 1 & 2 & \\ 3 & 3 & 2 & 1 & 1 \end{pmatrix}$$

is not of diagonal form, but is also describing the same R-vine tree structure.

An R imputation of an algorithm to bring a given R-vine matrix to diagonal form with unchanged first column can be found in the *RVineImpute* package, which was developed in the course of this thesis and presented in Appendix B.

4.1.2 Recursive decomposition of R-vines

Theorem 4.3
The joint density f for a given $d-$dimensional R-vine with R-vine matrix M can always be decomposed in the following way:

$$\begin{aligned} f_{m_{1,1}:m_{d1}}(x_{m_{1,1}}, \ldots, x_{m_{d,1}}) = {} & f_{m_{1,1}|m_{2,1}:m_{d,1}}(x_{m_{1,1}}|x_{m_{2,1}}, \ldots, x_{m_{d,1}}) \\ & \cdot f_{m_{2,2}|m_{3,2}:m_{d,2}}(x_{m_{2,2}}|x_{m_{3,2}}, \ldots, x_{m_{d,2}}) \cdot \ldots \\ & \cdot f_{m_{d-1,d-1}|m_{d,d-1}}(x_{m_{d-1,d-1}}|x_{m_{d,d-1}}) \cdot f_{m_{d,d}}(x_{m_{d,d}}) \end{aligned} \tag{4.3}$$

and the components of the above decomposition can be expressed in form of the given bivariate copulas as

$$f_{m_{ii}|m_{i+1,i}:m_{d,i}}(x_{m_{i,i}}|x_{m_{i+1,i}}, \ldots, x_{m_{d,i}}) = \left(\prod_{j=1}^{d-1-i} c_{m_{i,i},m_{i+j,i};m_{i+j+1,i}:m_{d,i}} \right) \cdot c_{m_{i,i},m_{d,i}} \cdot f_{m_{i,i}}(x_{m_{i,i}}) \tag{4.4}$$

for $i = 1, \ldots, d-2$, where we used abbreviation $c_{i,j;i_1:i_k}$ as in Definition 3.18.
Additionally

$$f_{m_{d-1,d-1}|m_{d,d-1}}(x_{m_{d-1,d-1}}|x_{m_{d,d-1}}) = c_{m_{d-1,d-1},m_{d,d-1}} \cdot f_{m_{d-1,d-1}|m_{d,d-1}}(x_{m_{d-1,d-1}}). \tag{4.5}$$

Proof: The decomposition (4.3) can always be done in this way as it is assured that there is always a new entry on each diagonal as stated in Property 3.26. Equation (4.4) for the components is a direct consequence of Lemma 3.6. By the special construction it is assured that we only use bivariate copulas already given by the R-vine distribution. □

In a general decomposition of the joint density

$$f_{1:d}(x_1,\dots,x_d) = f_{i|-\boldsymbol{i}}(x_i|\{x_1,\dots,x_d\}\backslash x_i)\cdot f_{-\boldsymbol{i}}(\{x_1,\dots,x_d\}\backslash x_i),\ i\in\{1,\dots,d\} \tag{4.6}$$

for an R-vine distribution, with $-\boldsymbol{i} := \{1,\dots,d\}\backslash i$, the first part of the right side $f(x_i|\{x_1,\dots,x_d\}\backslash x_i)$ can only be expressed in terms of the given bivariate copulas if X_i is either $m_{1,1}$ or $m_{2,1}$, with M being the respective R-vine matrix. For all other choices of X_i it is not possible to find a closed form using only the given bivariate copulas.

Example 4.1 (Continued) *Applying Theorem 4.3 to our R-vine yields*

$$\begin{aligned} f_{1:5}(x_1,x_2,x_3,x_4,x_5) =& f_{5|1:4}(x_5|x_1,x_2,x_3,x_4)\ f_{1|2:4}(x_1|x_2,x_3,x_4)\ f_{4|2,3}(x_4|x_2,x_3) \\ & f_{2|3}(x_2|x_3)\ f_3(x_3) \end{aligned} \tag{4.7}$$

and all components in (4.7) can be expressed in terms of the given copulas.

$$f_{5|1:4}(x_5|x_1,x_2,x_3,x_4) = c_{15;234}\ c_{45;23}\ c_{25;3}\ c_{35}\ f_5(x_5) \tag{4.8}$$
$$f_{1|2:4}(x_1|x_2,x_3,x_4) = c_{14;23}\ c_{13;2}\ c_{12}\ f_1(x_1) \tag{4.9}$$
$$f_{4|2,3}(x_4|x_2,x_3) = c_{24;3}\ c_{34}\ f_4(x_4) \tag{4.10}$$
$$f_{2,3}(x_2|x_3) = c_{23}\ f_2(x_2) \tag{4.11}$$
$$f_3(x_3) = f_3(x_3) \tag{4.12}$$

As we substitute (4.8) to (4.12) in (4.7), we get again the formula for the joint density of our R-vine distribution

$$\begin{aligned} f_{1:5}(x_1,x_2,x_3,x_4,x_5) =& c_{15;234}\cdot c_{14;23}\cdot c_{45;23}\cdot c_{13;2}\cdot c_{24;3}\cdot c_{25;3}\cdot c_{12}\cdot c_{23} \\ & \cdot c_{34}\cdot c_{15}\cdot f_1(x_1)\cdot f_2(x_2)\cdot f_3(x_3)\cdot f_4(x_4)\cdot f_5(x_5). \end{aligned} \tag{4.13}$$

If we would have used M' to derive the decomposition, we end up with

$$\begin{aligned} f_{1:5}(x_1,x_2,x_3,x_4,x_5) =& f_{1|2:5}(x_1|x_2,x_3,x_4,x_5)\ f_{5|2:4}(x_5|x_2,x_3,x_4)\ f_{4|2,3}(x_4|x_2,x_3) \\ & f_{2|3}(x_2|x_3)\ f_3(x_3), \end{aligned} \tag{4.14}$$

with

$$f_{1|2:5}(x_1|x_2,x_3,x_4,x_5) = c_{15;234}\ c_{14;23}\ c_{13;2}\ c_{12}\ f_1(x_1) \tag{4.15}$$

and

$$f_{5|2:4}(x_5|x_2,x_3,x_4) = c_{45;23}\ c_{25;3}\ c_{35}\ f_5(x_5). \tag{4.16}$$

If we want to derive a closed form for the conditional density of a variable, that is neither $m_{1,1}$ nor $m_{2,1}$ of the corresponding R-vine matrix M, given the others, then it is not

possible to do this using only the available bivariate copulas.
Suppose variable X_4 is missing and we observe (X_1, X_2, X_3, X_5). For the density we have following possible decompositions:

$$\begin{aligned} f_{4|1,2,3,5}(x_4|x_1,x_2,x_3,x_5) &= c_{14;235}\ f_{4|2,3,5}(x_4|x_2,x_3,x_5) && \text{or} \\ &= c_{24;135}\ f_{4|1,3,5}(x_4|x_1,x_3,x_5) && \text{or} \\ &= c_{34;125}\ f_{4|1,2,5}(x_4|x_1,x_2,x_5) && \text{or} \\ &= c_{45;123}\ f_{4|1,2,3}(x_4|x_1,x_2,x_3), \end{aligned}$$

but none of these decompositions uses an available bivariate copula.
We come to the same result for the two other possibilities $(X_2|X_1, X_3, X_4, X_5)$ and $(X_3|X_1, X_2, X_4, X_5)$.

Next we want to clear the question: given the R-vine structure and all bivariate copulas with corresponding parameters, which variables can be missing that we can simulate the missing ones given the observed ones by using a closed form conditional distribution? We will start in the next section with the simplification of a single missing variable.

4.1.3 Imputation for a single missing variable

The general approach we will use to impute missing variables will be the inverse transform sampling from Theorem 3.29. Assuming that only one variable is missing, means we only have to do step d of the inverse transform sampling.

Definition 4.4 (Case I and Case II classification of the missing data patterns with one missing variable)
Classify the missing data patterns with one missing variable for a given R-vine distribution as

- *Case I:*
  The missing variable is not in the conditioning set of any bivariate copula of the vine.
- *Case II:*
  The missing variable is in the conditioning set of at least one bivariate copula of the vine.

Lemma 4.5
The following conditions for a variable X in an R-vine with R-vine matrix M of diagonal form are equivalent:

i. The variable X is not in the conditioning set of any bivariate copula of the R-vine.

ii. In every tree of the R-vine tree structure the variable X is part of the conditioned set of a node which is a leaf.

iii. The variable X is either $m_{1,1}$ or $m_{2,1}$.

Proof:
i.→iii.: We know there exists only one edge in T_{d-1}, namely $(m_{1,1}, m_{2,1}|m_{3,1}, \dots, m_{d,1})$. As X is not in the conditioning set of any bivariate copula, this means X is not in the conditioning set of this edge. So X is either $m_{1,1}$ or $m_{2,1}$.
iii.→i.: With Lemma 4.1 we now the first two columns are of the form $(m_{1,1}, m_{2,1}, \dots)'$ and $(0, m_{2,1}, \dots)'$. Property 3.26 tells us that both $m_{1,1}$ and $m_{2,1}$ do not appear in any column further to the right. Applying Algorithm 3.27 to find the tree sequence it is clear that $m_{1,1}$ and $m_{2,1}$ are never in the conditioning set of any edge and therefore not in the conditioning set of any bivariate copula.
iii.→ii.: The specialty of a leaf in a tree is that it has only one edge. Let X be $m_{1,1}$ or $m_{2,1}$, then we already know that X is not in any conditioning set of any edge or node, as nodes are the edges of the prior step. Furthermore with help of Lemma 4.1 and Algorithm 3.27 it is easy to see that in every tree X is in the conditioned set of exactly one edge. And as edges in T_j are nodes in T_{j+1} conclude that X is part of the conditioned set of exactly one edge and one node in in every tree. Clearly the edge must be at the node. Now we only need to show that the node is a leaf. If there would be another edge at the node for which X is in the conditioned set, then X must be in the conditioned or conditioning set of this edge, however this produces a contradiction.
ii.→iii.: If X is in every tree in the conditioned set of a leaf, then it is in the conditioned set of a leaf in T_{d-1}. Set $v_1 := m_{1,1},\ v_2 := m_{2,1},\ v_3 := m_{3,1}$ and $v_4 := m_{3,2}$. In T_{d-1} there are only two nodes, which are also leafs, namely $(v_1, v_3|D_1)$ and $(v_2, v_4|D_2)$, where D_1, D_2 are the respective conditioning sets, connected by the edge $(v_1, v_2|v_3, \dots, v_d)$. Note that it could be the case that $v_3 = v_4$. So $v_1, \dots, v_4$ are in the conditioned sets of the leafs in T_{d-1}. If X is v_1 or v_2 the claim holds. However if X is v_3 or v_4 one can check, just by trying all possible combinations, that it is not possible for X to be in the conditioned set of a leaf in T_{d-2}. □
Lemma 4.5 iii. implies that there are always two missing data patterns in Case I.

Example 4.1 (Continued)
By using Lemma 4.5 iii., we see that $m_{1,1} = 5$ and $m_{2,1} = 1$.

$$\begin{pmatrix} \boxed{5} & & & & \\ \boxed{1} & 1 & & & \\ 4 & 4 & 4 & & \\ 2 & 3 & 2 & 2 & \\ 3 & 2 & 3 & 3 & 3 \end{pmatrix}$$

Consequently we study Case I and II for our example:
Case I: "Variable 1 or 5 is missing while all others are given."
Case II: "Variable 2,3 or 4 is missing while all others are given."

Now assume in a given $d-$dimensional R-vine we want to find a closed form for $F_{i|-\boldsymbol{i}}(x_i|\{x_1, \dots, x_d\}\backslash x_i)$ only using the given bivariate copulas, as we observed that X_i

has missing values and all other variables are always observed. Applying the recursion formula from Theorem 3.19 and using its notation ($X := X_i$), we can easily see that it is only possible to choose a Y_j, as used in the theorem, in the way that the resulting copula is given from the R-vine distribution, if the missing data pattern $(X_i|\{X_1, \dots, X_d\}\backslash X_i)$ is in Case I. Iteratively applying Theorem 3.19 in such a way that only given copulas are used we can find the desired closed form.

Example 4.1 (Continued)
We start with the Case I missing data pattern, where variable 5 is missing while all other variables are observed. We have already shown the closed form for the density $f_{5|1:4}(x_5|x_1, x_2, x_3, x_4)$ in (4.8), namely

$$f_{5|1:4}(x_5|x_1, x_2, x_3, x_4) = c_{15;234}\ c_{45;23}\ c_{25;3}\ c_{35}\ f_5(x_5)$$

The conditional distribution function $F_{5|1:4}(x_5|x_1, x_2, x_3, x_4)$ can be derived by repeatedly using Theorem 3.19.

$$F_{5|1:4}(x_5|x_1, x_2, x_3, x_4) = C_{5|1;234}(F_{5|2:4}(x_5|x_2, x_3, x_4), F_{1|2:4}(x_1|x_2, x_3, x_4)) \tag{4.17}$$
$$F_{5|2:4}(x_5|x_2, x_3, x_4) = C_{5|4;23}(F_{5|2,3}(x_5|x_2, x_3), F_{4|2,3}(x_4|x_2, x_3)) \tag{4.18}$$
$$F_{1|2:4}(x_1|x_2, x_3, x_4) = C_{1|4;23}(F_{1|2,3}(x_1|x_2, x_3), F_{4|2,3}(x_4|x_2, x_3)) \tag{4.19}$$
$$F_{5|2,3}(x_5|x_2, x_3) = C_{5|2;3}(F_{5|3}(x_5|x_3), F_{2|3}(x_2|x_3)) \tag{4.20}$$
$$F_{4|2,3}(x_4|x_2, x_3) = C_{4|2;3}(F_{4|3}(x_4|x_3), F_{2|3}(x_2|x_3)) \tag{4.21}$$
$$F_{1|2,3}(x_1|x_2, x_3) = C_{1|3;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2)) \tag{4.22}$$
$$F_{5|3}(x_5|x_3) = C_{5|3}(F_5(x_5), F_3(x_3)) \tag{4.23}$$
$$F_{2|3}(x_2|x_3) = C_{2|3}(F_2(x_2), F_3(x_3)) \tag{4.24}$$
$$F_{4|3}(x_4|x_3) = C_{4|3}(F_4(x_4), F_3(x_3)) \tag{4.25}$$
$$F_{1|2}(x_1|x_2) = C_{1|2}(F_1(x_1), F_2(x_2)) \tag{4.26}$$
$$F_{3|2}(x_3|x_2) = C_{3|2}(F_3(x_3), F_2(x_2)) \tag{4.27}$$

Putting all together we end up with a closed form for the conditional distribution function using only copulas from the R-vine structure and the marginal distributions:

$$\begin{aligned} F_{5|1:4}(x_5|x_1, x_2, x_3, x_4) = {} & C_{5|1;234}(C_{5|4;23}(C_{5|2;3}(C_{5|3}(F_5(x_5), F_3(x_3)), C_{2|3}(\\ & F_2(x_2), F_3(x_3))), C_{4|2;3}(C_{4|3}(F_4(x_4), F_3(x_3)), C_{2|3}(\\ & F_2(x_2), F_3(x_3)))), C_{1|4;23}(C_{1|3;2}(C_{1|2}(F_1(x_1), F_2(x_2)), \\ & C_{3|2}(F_3(x_3), F_2(x_2))), C_{4|2;3}(C_{4|3}(F_4(x_4), F_3(x_3)), \\ & C_{2|3}(F_2(x_2), F_3(x_3))))) \end{aligned} \tag{4.28}$$

Otherwise if we would have chosen the other element of Case I, namely $(X_1|X_2, X_3, X_4, X_5)$, likewise as before we can also derive the closed form for the conditional distribution:

$$\begin{aligned} F_{1|2:5}(x_1|x_2, x_3, x_4, x_5) = {} & C_{1|5;234}(C_{1|4;23}(C_{1|3;2}(C_{1|2}(F_1(x_1), F_2(x_2)), C_{3|2}(F_3(x_3), \\ & F_2(x_2))), C_{4|2;3}(C_{4|3}(F_4(x_4), F_3(x_3)), C_{2|3}(F_2(x_2), F_3(x_3)))), \\ & C_{5|4;23}(C_{5|2;3}(C_{5|3}(F_5(x_5), F_3(x_3)), C_{2|3}(F_2(x_2), F_3(x_3))), \\ & C_{4|2;3}(C_{4|3}(F_4(x_4), F_3(x_3)), C_{2|3}(F_2(x_2), F_3(x_3))))) \end{aligned} \tag{4.29}$$

Once we have found a closed form for the conditional distribution $F_{i|-\boldsymbol{i}}(x_i|\{x_1,\dots,x_d\}\backslash x_i)$ using only given bivariate copulas the only thing missing before we can perform step m of the inverse transform sampling is to invert it. However this appears to be easy with help of Lemma 3.6.

Example 4.1 (Continued)
Repeatedly using Lemma 3.6 yields:

$$F^{-1}_{5|1234}(u_5|x_1,x_2,x_3,x_4) = F^{-1}_{5|234}(C_{5|1;234}(u_5|F_{1|2:4}(x_1|x_2,x_3,x_4))|x_2,x_3,x_4) \tag{4.30}$$
$$F^{-1}_{5|234}(u_5|x_2,x_3,x_4) = F^{-1}_{5|23}(C_{5|4;23}(u_5|F_{4|2,3}(x_4|x_2,x_3))|x_2,x_3) \tag{4.31}$$
$$F^{-1}_{5|23}(u_5|x_2,x_3) = F^{-1}_{5|3}(C_{5|2;3}(u_5|F_{2,3}(x_2|,x_3))|x_3) \tag{4.32}$$
$$F^{-1}_{5|3}(u_5|x_3) = F^{-1}_{5}(C_{5|3}(u_5|F_3(x_3))) \tag{4.33}$$

Now we can put all together for a closed form of $F^{-1}_{5|1234}(u_5|x_1,x_2,x_3,x_4)$ using only the bivariate copulas of the R-vine structure and the marginal distributions by using (4.17) to (4.33).

$$\begin{aligned} F^{-1}_{5|1234}(u_5|x_1,x_2,x_3,x_4) = F^{-1}_5(&C_{5|3}(C_{5|2;3}(C_{5|4;23}(C_{5|1;234}(u_5|C_{1|4;23}(\\ &C_{1|3;2}(C_{1|2}(F_1(x_1),F_2(x_2)),C_{3|2}(F_3(x_3), \\ &F_2(x_2))),C_{4|2;3}(C_{4|3}(F_4(x_4),F_3(x_3)),C_{2|3}(\\ &F_2(x_2),F_3(x_3)))))|C_{4|2;3}(C_{4|3}(F_4(x_4),F_3(x_3)), \\ &C_{2|3}(F_2(x_2),F_3(x_3))))|C_{2|3}(F_2(x_2),F_3(x_3)))|F_3(x_3))) \end{aligned} \tag{4.34}$$

Building the inverse likewise as before for $F(x_1|x_2,x_3,x_4,x_5)$ yields:

$$\begin{aligned} F^{-1}_{1|2345}(u_1|x_2,x_3,x_4,x_5) = F^{-1}_1(&C_{1|2}(C_{1|3;2}(C_{1|4;23}(C_{1|5;234}(u_1|C_{5|4;23}(\\ &C_{5|2;3}(C_{5|3}(F_5(x_5),F_3(x_3)),C_{2|3}(F_2(x_2),F_3(x_3))),C_{4|2;3}(\\ &C_{4|3}(F_4(x_4),F_3(x_3)),C_{2|3}(F_2(x_2),F_3(x_3)))))|C_{4|2;3}(\\ &C_{4|3}(F_4(x_4),F_3(x_3)),C_{2|3}(F_2(x_2),F_3(x_3))))|C_{3|2}(F_3(x_3), \\ &F_2(x_2)))|F_2(x_2))) \end{aligned} \tag{4.35}$$

Therefore, we showed we can simulate $(X_5|X_1,X_2,X_3,X_4)$ and $(X_1|X_2,X_3,X_4,X_5)$.

So, we can simulate a missing variable given all others for the missing data patterns in Case I, but not if the missing data pattern is part of Case II.

Notation 4.6 (Alternative a, alternative b)
For a given R-vine matrix M we will call the variable stored in $m_{1,1}$ alternative a, and the variable stored in $m_{2,1}$ will be called alternative b.

Using this notation we can summarize that we found out we can simulate the alternative *a* variable given all variables below in column 1 of the corresponding R-vine matrix M and if we choose the alternative *b* variable to be missing we can simulate it given all other variables below in the first column of the R-vine matrix together with alternative *a*. These are exactly the two missing data patterns of Case I.

4.1.4 Imputation for two missing variables

We proceed with missing data patterns in which two variables are missing. This corresponds to a start in step $d-1$ of the inverse transform sampling. For example in a 5−dimensional R-vine we could first simulate $(X_1|X_2, X_3, X_4)$ and then $(X_5|X_1, X_2, X_3, X_4)$. Now it is again the question for which missing data patterns with two missing variables it is possible to do the inverse transform sampling by using the available bivariate copulas to derive a closed form for the conditional distributions. With this question in mind observe the following:

Theorem 4.7 (Reduced R-vine distributions)
Given an R-vine distribution with corresponding R-vine matrix M of diagonal form,

1. *removing each node and edge with the variable alternative 'a', results in a reduced R-vine distribution with corresponding R-vine matrix M^a, that can be obtained by deleting the first row and column of M.*

2. *removing each node and edge with the variable alternative 'b', results in a reduced R-vine distribution with corresponding R-vine matrix M^b, that can be obtained by deleting the second row and column of M.*

Proof: 1. Property 3.26 already states that deleting the first row and column again results in a reduced R-vine, and the right elements were deleted as all connections of the variable alternative 'a' with the other variables are stored in the first column of M.
2. Let M' be the second R-vine matrix of diagonal form according to Corollary 4.2. Note that since M and M' are both of diagonal form, the first column of M' is $(m_{2,1}, m_{1,1}, m_{3,2}, \ldots, m_{d,2})\prime$ and the second column is $(0, m_{1,1}, m_{3,1}, \ldots, m_{d,1})\prime$. Then observe that deleting the second row and column of M and bringing it to diagonal form afterwards, if necessary, is the same as deleting the first row and column of M'. □

The reduced R-vine matrix M^b is not necessarily of diagonal form but can again be brought to it in such a way that the first column is unchanged.

Example 4.1 (Continued)
Removing each node with Variable 5 (alternative a) and all connected edges in the R-vine tree sequence results in the 4 dimensional R-vine tree sequence illustrated in Figure 4.2 with reduced R-vine matrix

$$\begin{pmatrix} \cancel{5} & & & & \\ 1 & 1 & & & \\ 4 & 4 & 4 & & \\ 2 & 3 & 2 & 2 & \\ 3 & 2 & 3 & 3 & 3 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ 4 & 4 & & \\ 3 & 2 & 2 & \\ 2 & 3 & 3 & 3 \end{pmatrix} = M^a. \tag{4.36}$$

Doing the same with Variable 1 (alternative b) results in the 4 dimensional R-vine structure illustrated in Figure 4.3. The corresponding R-vine matrix M^b, which is by coincidence

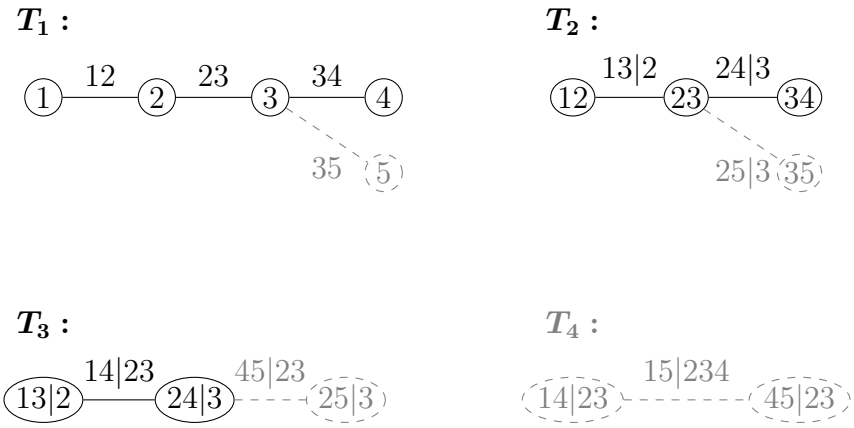


Figure 4.2: Reduced 4 dimensional R-vine tree structure derived from the 5 dimensional R-vine tree structure by deleting all nodes and edges with Variable 5.

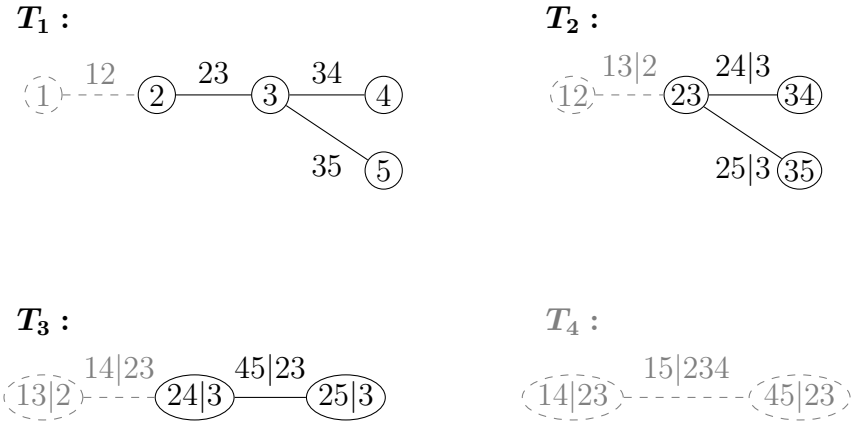


Figure 4.3: Graphical illustration of the reduced 4 dimensional R-vine structure derived from the 5 dimensional R-vine by deleting all nodes and edges with Variable 1.

already of diagonal form, is given by

$$\begin{pmatrix} 5 & & & & \\ \cancel{1} & \cancel{1} & & & \\ 4 & 4 & 4 & & \\ 2 & 3 & 2 & 2 & \\ 3 & 2 & 3 & 3 & 3 \end{pmatrix} = \begin{pmatrix} 5 & & & & \\ 4 & 4 & & & \\ 2 & 2 & 2 & & \\ 3 & 3 & 3 & 3 & \end{pmatrix} = M^b. \tag{4.37}$$

For both of the reduced R-vine matrices M^a and M^b from Theorem 4.7 we already have the corresponding bivariate copulas from the full R-vine distribution and do not need to estimate them again. However now we will see that the problem shows up to be recursive with the reduced R-vine distributions. After we solved step d of the inverse transform sampling and reduced the R-vine distributions, we are now in step $d-1$ of the inverse

transform sampling. As we have determined the R-vine matrices and copulas of the reduced R-vine distributions already, we can solve this step similar as before.

Example 4.1 (Continued)
With the choice of alternative a first, we choose that Variable 5 is missing, and we get two new alternatives in the reduced R-vine distribution. Alternative a in M^a: Variable 1 is missing, and alternative b in M^a: Variable 4 is missing. As these are $m^a_{1,1}$ and $m^a_{2,1}$ of the reduced matrix M^a.

$$M^a = \begin{pmatrix} \boxed{1} & & & \\ \boxed{4} & 4 & & \\ 3 & 2 & 2 & \\ 2 & 3 & 3 & 3 \end{pmatrix}$$

If we first choose alternative b which means Variable 1 is missing, we get also two new alternatives for which additional variable can be missing so we can derive all conditional distributions to simulate them. Alternative a in M^b: Variable 5 is missing, and alternative b in M^b: Variable 4 is missing. As these are $m^b_{1,1}$ and $m^b_{2,1}$ of the reduced matrix M^b.

$$M^b = \begin{pmatrix} \boxed{5} & & & \\ \boxed{4} & 4 & & \\ 2 & 2 & 2 & \\ 3 & 3 & 3 & 3 \end{pmatrix}$$

We end up with the pairs in Table 4.1.

1st choice (variable)	2nd choice (variable)	Simulate in reverse order
a (5)	a (1)	$(X_1 \mid X_2, X_3, X_4)$ then $(X_5 \mid X_1, X_2, X_3, X_4)$
a (5)	b (4)	$(X_4 \mid X_1, X_2, X_3)$ then $(X_5 \mid X_1, X_2, X_3, X_4)$
b (1)	a (5)	$(X_5 \mid X_2, X_3, X_4)$ then $(X_1 \mid X_2, X_3, X_4, X_5)$
b (1)	b (4)	$(X_4 \mid X_2, X_3, X_5)$ then $(X_1 \mid X_2, X_3, X_4, X_5)$

Table 4.1: Variable pairs in Example 4.1 which are allowed to be missing so that we are able to simulate them directly, using the given R-vine structure and inverse transform sampling.

4.1.5 Imputation for general missing data patterns

After we observed the recursive character of the problem in the last section, we can now determine for which general missing data patterns it suffices to use the inverse transform

sampling with the available bivariate copulas for the imputation.
A possible combination of missing variables can be obtained by:

Algorithm 4.8 (Finding a missing data pattern we can impute directly)
Input: *An R-vine matrix M of diagonal form.*
Output: *A missing data pattern, that we can impute by using the inverse transform sampling with the available bivariate copulas from the R-vine distribution.*

1. *Start to choose from the full R-vine matrix alternative a or b ($m_{1,1}$ or $m_{2,1}$).*
2. *Reduce the R-vine matrix by either deleting the first row and column if the last choice was alternative a, or by deleting the second row and column if the last choice was alternative b.*
3. *Stop if the reduced matrix has only 1 element.*
4. *Bring reduced R-vine matrix to diagonal form with unchanged first column if necessary.*
5. *Stop or choose between alternatives a and b in the reduced matrix of diagonal form and go again to step 2.*

With Algorithm 4.8 we can find every missing data pattern we can impute directly using the bivariate copulas from given R-vine. However only one at a time. To have a better overview of all the missing data patterns, that we can find with Algorithm 4.8, we will now introduce a tree that represents all possible outcomes.

Definition 4.9 (Choice-tree)
For a d-dimensional R-vine matrix M of diagonal form construct a tree as following:

1. *Set a parent node '0' in level 0 of the tree.*
2. *The two children of node '0' in level 1 are alternative a and alternative b. The edges are named after the alternatives.*
3. (a) *For a node v in level $j = 1, \dots, d-2$ of the tree, let $c \in \{a, b\}^j$ be the vector of the names of the edges that are part of the path from '0' to v in the order of appearance.*
   (b) *For k from 1 to j, reduce the R-vine matrix by deleting the first row and column if $c_k =$ "a", else by deleting the second row and column and bringing the R-vine matrix again to diagonal form if necessary.*
   (c) *The two children of v are alternative a and alternative b of the reduced R-vine matrix. The edges are named after the alternatives.*

This tree will be called choice-tree.

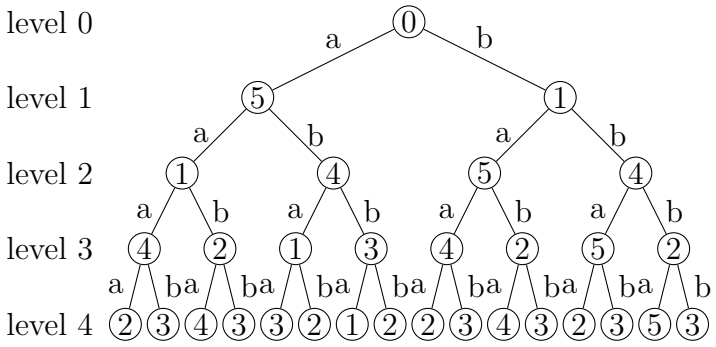


Figure 4.4: Choice-tree for Example 4.1.

Definition 4.10 (Choice-path, variable-path)
A path in the choice-tree that starts at the parent node '0' and uses every single edge not more often than once will be called choice-path.
A choice-path will be represented with a vector of names of the edges in the choice-path. The order in this vector represents the order the edges are visited in the choice-path.
The vector of names of the nodes a choice-path c visits after node '0' will be called variable-path derived from choice-path c. The order in the variable-path represents the order the nodes are visited in the choice-path c.
Let $\mathbf{V}$ *be the set of all variables contained in a variable-path derived from choice-path c, then we say the choice-path c covers the variables in* $\mathbf{V}$*.*

Every variable-path can be interpreted as a missing data pattern where we can simulate the missing variables given the observed ones using the inverse transform sampling with the available bivariate copulas.

Theorem 4.11 (Ad-hoc imputable missing data patterns)
Let M be the corresponding R-vine matrix for a given d-dimensional R-vine with joint distribution function $F_{1:d}$*. For a missing data pattern with* $n < d$ *missing variables, define the set of all observed variables as* I_1*. Let* $v = (v_1, \ldots, v_n)$ *be a variable path in the choice-tree derived from M that contains all the missing variables. Then simulate for every incomplete data row with this missing data pattern*

1. v_n *from* $F^{-1}_{v_n|I_1}$, *and set* $I_2 := I_1 \cup \{v_n\}$.
2. v_{n-1} *from* $F^{-1}_{v_{n-1}|I_2}$, *and set* $I_3 := I_2 \cup \{v_{n-1}\}$.

$\ldots$ $\ldots$ $\ldots$

n. v_1 *from* $F^{-1}_{v_1|I_n}$.

The imputed values together with the observed ones have joint distribution function $F_{1:d}$*. All conditional distributions used, can be expressed in terms of given pair copulas from the R-vine distribution.*

Example 4.1 (Continued)
For the variable-path $(5, 1, 2, 4)$ *derived from the choice-path* (a, a, b, a)*, simulate*

$$X_4 \text{ from } F^{-1}_{4|3}$$
$$X_2 \text{ from } F^{-1}_{2|3,4}$$
$$X_1 \text{ from } F^{-1}_{1|2,3,4}$$
$$X_5 \text{ from } F^{-1}_{5|1,2,3,4}$$

For another variable-path $(1, 4)$ *derived from the choice-path* (b, b)*,* $I_1 = \{2, 3, 5\}$ *and we can simulate the variables again in the reverse order of the path:*

$$X_4 \text{ from } F^{-1}_{4|2,3,5}$$
$$X_1 \text{ from } F^{-1}_{1|2,3,4,5}$$

The total number of variable-paths in a choice-tree for a d-dimensional R-vine is $2^d - 2$, but observe that several variable-paths in Figure 4.4 can lead to a solution to the same missing data pattern. So do $(1, 5)$ and $(5, 1)$ both lead to a possibility to simulate the missing variables 1 and 5 given the observed variables 2,3 and 4. Only the order in the inverse transform sampling is changed. At the first view it is hard to identify all possible missing data patterns as we have so many choice-paths that cover the same variables. The next section will address how to identify those redundant solutions.

4.1.6 Identifying redundant solutions

When we are interested in for which missing data patterns we can simulate the missing variables given the observed ones via the estimated R-vine distribution and inverse transform sampling, we usually do not care in which order we simulate the missing variables. Therefore observe the following.

Property 4.12
Being in Step 5. of the i*-th iteration of Algorithm 4.8 there are two choices. If*

- *alternative a is chosen, then alternative b becomes alternative a in iteration* $i + 1$*.*
- *alternative b is chosen, then alternative a remains alternative a in the iteration* $i+1$*.*

This result follows directly from the way we build the reduced matrix. The principle for the first part of the above property is the fact that the R-vine matrix has diagonal form. The consequence of this result is that it is now possible to determine which variable-paths have the same set of variables covered by different choice-paths.

Lemma 4.13 (Redundant choice-paths)
All variables, covered by a choice-path c *of the form* $c = (***, b, a, ***)$ *with a 'b' followed by an 'a', are also covered by another choice-path* $\hat{c} = (* * *, a, a, * * *)$ *of same length. Both paths coincide at the start up till the pattern occurs, but may have a different end.*

Proof: The idea here is to use Property 4.12 in the form that the choice of alternative 'b' does not change alternative 'a'. So it is possible to choose at first 'a', and alternative 'b' remains an alternative, namely alternative 'a', in the next step and will be chosen afterwards. Doing this change, we have chosen exactly the same variables in both paths. Therefore the reduced R-vine distributions coincide and in the remaining steps the two choice paths have exactly the same alternatives. Then $\hat{c}$ can cover the same variables as c. The variables just appear in different order in the two choice-paths. □

Example 4.1 (Continued)

1. *The choice-path* (b, b, a) *covers the same variables as the choice-path* (a, a, a)*, namely* $1, 4$ *and* 5*.*

2. *The choice-path* (a, b, b, a) *covers the same variables as the choice-path* (a, a, a, b)*, namely* $1, 3, 4$ *and* 5*.*

Theorem 4.14 (Ordered choice-paths)
Every choice-path in the choice-tree for a d-dimensional R-vine matrix having an 'a' following a 'b' covers the same variables as another choice-path of the form: k times 'a', followed by l times 'b', where $k = 0, \ldots, d-1$ *and* $l = 0, \ldots, d-1-k$*. A choice-path of this form will be called ordered, otherwise unordered.*

Proof: Use Lemma 4.13 repeatedly until the choice-path is of desired form. □

Furthermore note that every ordered choice-path covers a different set of variables due to its special structure.

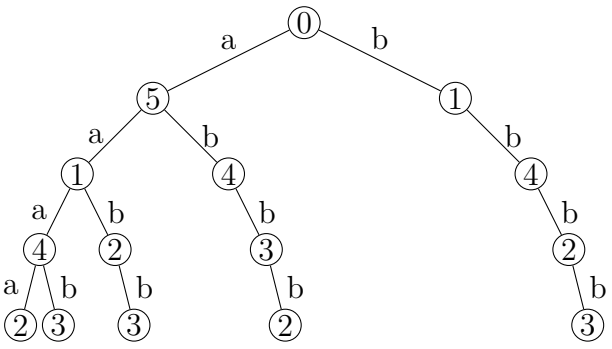


Figure 4.5: Choice-tree for Example 4.1 where all unordered choice-paths were removed.

Example 4.1 (Continued)
Now we can easily read out of Figure 4.5 all missing data patterns we can simulate with the inverse transform sampling and available bivariate copulas.

Finally we can now formulate the outline of a general algorithm that identifies all missing data patterns where we can simulate the missing variables directly via the R-vine structure and inverse transform sampling. Additionally this algorithm will give us a possible simulation order.

1 missing:	(1), (5)
2 missing:	(1,4), (1,5), (4,5)
3 missing:	(1,2,4), (1,2,5), (1,4,5), (3,4,5)
4 missing:	(1,2,3,4), (1,2,3,5), (1,2,4,5), (1,3,4,5), (2,3,4,5)

Table 4.2: Combinations of missing variables we can simulate given the others in Example 4.1.

Algorithm 4.15
Input: *A $d \times d$ dimensional R-vine matrix.*
Output: *A list of all unique variable combinations for which the variables can be simulated via the R-vine structure and inverse transform sampling.*

1. *Bring R-vine matrix to diagonal form.*
2. *Find all choice-paths of the form k times 'a', followed by l times 'b', where $k = 0, \dots, d-1$ and $l = 0, \dots, d-1-k$.*
3. *Translate all choice-paths to variable-paths.*

How to simulate the variables contained in the variable-paths from step 3. was already shown in Theorem 4.11.
An R implementation of Algorithm 4.15 can be found in the developed *VineCopulaImpute* package, which is presented in Appendix B.
When we want to translate the choice-paths to variable-paths we pick an element in the matrix, reduce the matrix, pick a new element, reduce again and so on. It is not necessary to bring the reduced matrices again to diagonal form due to the special choice-paths in Step 2. as it either is already of diagonal form after repeatedly picking alternative a, or if we pick consecutively alternative b we only pick elements in the first column of the reduced matrix, that we have obtained after reducing for the last alternative a, which is unchanged in the diagonal form. Here it is again an advantage that we only have to take choice-paths into consideration where no 'a' follows a 'b'. In fact it is not even necessary to really reduce the matrix. If we choose alternative a we pick the diagonal element in the next column and if we choose alternative b for the first time we pick the element under the diagonal element of the next column. For every further choice of alternative b we pick the element under the last element picked.

Example 4.1 (Continued)
The choice-path (a, a, b, b) gives the variable-path $(5, 1, 2, 3)$, as can be checked in the matrix:

$$\begin{pmatrix} \mathbf{5} & & & & \\ 1 & \mathbf{1} & & & \\ 4 & 4 & 4 & & \\ 2 & 3 & \mathbf{2} & 2 & \\ 3 & 2 & \mathbf{3} & 3 & 3 \end{pmatrix}$$

The number of choice-paths in Step 2. of Algorithm 4.15 is $\sum_{i=2}^{d} i = \frac{d(d+1)}{2} - 1$, and is e.g. 14 for $d = 5$. This sum is easy to interpret as there are

2 missing data patterns with 1 missing variable,
3 missing data patterns with 2 missing variables,
4 missing data patterns with 3 missing variables,
...
d missing data patterns with d-1 missing variables,

where we can simulate the missing variables given the observed ones using the R-vine structure and inverse transform sampling.

Example 4.1 (Continued)
Applying Algorithm 4.15 to our 5 dimensional example leads to the following choice-paths and variable-paths:

	choice-path	*variable-path*
1.	*(a)*	*(5)*
2.	*(a,a)*	*(5,1)*
3.	*(a,a,a)*	*(5,1,4)*
4.	*(a,a,a,a)*	*(5,1,4,2)*
5.	*(a,a,a,b)*	*(5,1,4,3)*
6.	*(a,a,b)*	*(5,1,2)*
7.	*(a,a,b,b)*	*(5,1,2,3)*
8.	*(a,b)*	*(5,4)*
9.	*(a,b,b)*	*(5,4,3)*
10.	*(a,b,b,b)*	*(5,4,3,2)*
11.	*(b)*	*(1)*
12.	*(b,b)*	*(1,4)*
13.	*(b,b,b)*	*(1,4,2)*
14.	*(b,b,b,b)*	*(1,4,2,3)*

Before we summarize our results we will show a direct consequence that is very useful for the conditional simulation.

Corollary 4.16 (Special diagonal form for a missing data pattern)
Suppose we have given a variable path $(v_1, \ldots, v_m)$, $m \leq d$, for a choice tree of an d-dimensional R-vine with diagonal R-vine matrix M. Then there exists a second R-vine matrix M' with diagonal $(v_1, \ldots, v_m, *, \ldots, *)'$, describing the same R-vine tree structure.

Proof: The variable path $(v_1, \ldots, v_m)$ corresponds to a choice path of the form k times a, followed by l times b, as in Theorem 4.14. The first k elements are already on the

diagonal in M. The $(k+1)$-th column looks like $(m_{k+1,k+1}, v_{k+1}, \ldots, v_m, *\ldots, *)'$. According to Algorithm 3.28, another possible R-vine matrix has v_{k+1} on the diagonal and $m_{k+1,k+1}$ beneath in the $(k+1)$-th column and the $(k+2)$-th column can then be $(m_{k+1,k+1}, v_{k+2}, \ldots, v_m, *\ldots, *)'$. This argumentation can be used repeatedly until the R-vine matrix is of desired form. □

This Corollary simplifies conditional simulation from an R-vine. If we want to simulate the variables from a variable-path, the only thing we need to be possible to do is to conditionally simulate Variable $m_{1,1}$ of the R-vine matrix M.

Example 4.1 (Continued)
Using Algorithm 4.15, we derived that $(5,4,3)$ *is a variable path. A R-vine matrix of the form as in Corollary 4.16 is*

$$M^d = \begin{pmatrix} 5 & & & & \\ 1 & 4 & & & \\ 4 & 1 & 3 & & \\ 2 & 2 & 1 & 2 & \\ 3 & 3 & 2 & 1 & 1 \end{pmatrix}.$$

Then if we are able to simulate the $(1,1)$*-element of a general R-vine matrix conditioning on all other variables, the procedure is as following:*

1. *Simulate Variable 3 conditioning on Variables 1 and 2 from the reduced R-vine, with R-vine matrix* $\begin{pmatrix} 3 & & \\ 1 & 2 & \\ 2 & 1 & 1 \end{pmatrix}$.
2. *Simulate Variable 4 conditioning on Variables 1, 2 and 3 from the reduced R-vine, with R-vine matrix* $\begin{pmatrix} 4 & & & \\ 1 & 3 & & \\ 2 & 1 & 2 & \\ 3 & 2 & 1 & 1 \end{pmatrix}$.
3. *Simulate Variable 5 conditioning on the other variables from the full R-vine, with R-vine matrix* M^d.

4.1.7 Summary

So we found the missing data patterns where we can impute the missing variables via the R-vine structure and conditional distribution, however it is the question how large is the proportion of these missing data patterns compared to the number of all possible missing data patterns. There are in total $2^d - 2$ missing data patterns if we neglect the two missing data patterns where no variable or all variables are missing. And we saw that

we can impute the missing variables for a given R-vine in $\frac{d(d+1)}{2} - 1$ missing data patterns via inverse transform sampling which does mean that the gap is increasing exponentially. Table 4.3 shows the proportion for dimensions $d = 2, \ldots, 10$.

d	imputable mdp's	total number of mdp's	proportion
2	2	2	100%
3	5	6	83%
4	9	14	64%
5	14	30	47%
6	20	62	32%
7	27	126	21%
8	35	254	14%
9	44	510	9%
10	54	1022	5%

Table 4.3: Proportion of missing data patterns (mdp's) that we can impute by using the R-vine structure and inverse transform sampling of all possible missing data patterns in dimensions $d = 2, \ldots, 10$.

We can even say a bit more about the number of missing data patterns that we can impute in a fix dimension d, summarized in Table 4.4. Observe here that we can always impute all possible missing data patterns in the case with all but one variable missing, but for all other cases we cannot impute all possible missing data patterns with our method. For all

number of missing variables	imputable mdp's	total # of mdp's
1	2	d
2	3	$\binom{d}{2}$
3	4	$\binom{d}{3}$
$\vdots$	$\vdots$	$\vdots$
$d-3$	$d-2$	$\binom{d}{3}$
$d-2$	$d-1$	$\binom{d}{2}$
$d-1$	d	d

Table 4.4: Imputable missing data patterns (mdp's) compared to total number of mpd's in dimension d, broken down by number of missing variables.

the missing data patterns that can not be imputed 'ad hoc' we need a different approach. The next part of this chapter will deal with this issue.

4.2 Non ad hoc imputable missing data patterns

So far we identified the missing data patterns which we can solve directly using the inverse transform sampling together with closed forms for conditional distribution functions, given a specific R-vine distribution. This distribution can be gained by fitting the complete part of the data. In the course of the derivation of this theory, we saw that we can only impute two missing data patterns with one missing variable given the others. Now, we will present a possible solution how to solve the missing data patterns that can not be imputed ad hoc from closed forms of conditional distribution functions.
Assume we have given an incomplete data set with variables $X_1, \dots, X_d$ and we want to impute the missing data patterns where only X_k is missing and all other variables are given, for a $k \in \{1, \dots, d\}$. Fitting an R-vine to the complete part of the data, we obtain an R-vine Matrix M. Additionally, assume now that X_k is neither $m_{1,1}$ nor $m_{2,1}$. From the prior section we know that it is not possible to find a closed form for

$$F^{-1}_{k|-\boldsymbol{k}}(X_k|X_1, \dots, X_{k-1}, X_{k+1}, \dots, X_d) \tag{4.38}$$

using the pair copulas from the given R-vine distribution.
In the following, we will derive a possible solution that ensures we can find a closed form for (4.38) if X_k is not in $m_{1,1}$ or $m_{2,1}$, still acknowledging the dependence structure as far as possible. Later, we will expand this theory to general missing data patterns.
The proposed approach for a missing data pattern with one missing variable is to fit an R-vine to the reduced data set $\boldsymbol{X}_{-\boldsymbol{k}} := \{X_1, \dots, X_{k-1}, X_{k+1}, \dots, X_d\}$ and then "add" X_k to the R-vine in such a manner that we can find a closed form for (4.38).

4.2.1 Adding one variable to an existing R-vine

Let M be the R-vine matrix for a subset $\boldsymbol{X}_{-\boldsymbol{k}}$ of a d-dimensional data set. Now we want to add X_k to the R-vine. Denote the R-vine matrix of the d-dimensional R-vine including X_k as M^{add}. As our goal is to find a closed form for (4.38), we already know that X_k must be $m^{add}_{1,1}$ or $m^{add}_{2,1}$. W.l.o.g. set

$$m^{add}_{1,1} := X_k. \tag{4.39}$$

This has the advantage that it is ensured X_k is not occurring in any column further to the right, according to Property 3.26, which also states that deleting the first row and column of an d-dimensional R-vine matrix gives a $(d-1)$-dimensional R-vine matrix for a reduced data set. As M^{add} shall be an extension of M we can write

$$M^{add} = \begin{pmatrix} X_k & \\ * & \\ * & M \\ * & \end{pmatrix}. \tag{4.40}$$

Note that this is only one possible form of M^{add} and it is necessary to choose as in (4.39) to get this form. The form as in (4.40) has the great advantage that everything is filled out except the first column. With this shortcut the problem of adding a variable to the

tree structure reduces to specifying the first column of M^{add} in (4.40).
As we already know from Lemma 4.5 X_k has to be part of the conditioned set of a leaf in every tree of the R-vine tree structure and never part of a conditioning set of any node. Additionally, as X_k is occurring only in the first column we know that there is exactly one node in every tree where X_k is in the conditioned set. This means we have to add one leaf in every tree of the R-vine tree structure of the $(d-1)$-dimensional R-vine and X_k shall be in the conditioned set of this leaf. Of course the properties of an R-vine matrix, as the proximity condition, have to be fulfilled.
In the first tree $\boldsymbol{T}_1$ we have $(d-1)$ possibilities where we can add a leaf without violating the proximity condition and we need some criterion to choose among them.
Dißmann et al. (2013) present an algorithm that selects the structure, copula families and parameters for an R-vine. The algorithm that is also the groundwork for the *RVineStructureSelect()* function of the *VineCopula* R-package, see Schepsmeier et al. (2015), selects the structure in one tree depending on the fitted copulas of the lower trees. In $\boldsymbol{T_1}$ for all variable pairs (i,j), $j \neq k \in \{1,\dots,d\}$, the empirical Kendall's tau $\hat{\tau}_{i,j}$ is computed and the spanning tree which maximizes the sum of these correlation coefficients is selected. Afterwards the copulas for the edges are fitted. In all higher trees the algorithm works similar using the empirical conditional Kendall's tau $\hat{\tau}_{i,j|D}$, computed for all possible edges $(i,j|D)$ with conditioning set D, not violating the proximity condition. Again the spanning tree which maximizes the sum of these (conditional) correlation coefficients is selected and the corresponding copulas are fitted. This sequential approach ensures that the strongest pairwise dependencies are modeled initially.
The empirical (conditional) Kendall's tau computations in these steps work as following: As already mentioned, in the first tree we just use the estimate from Definition 3.9 to calculate the empirical unconditional Kendall's tau $\hat{\tau}_{i,j}$ for all variable pairs (i,j), $i,j \in \{1,\dots,d\}$ with $i \neq j$. Here we are using the u-data $(u_1,\dots,u_d)$ of all variables. After finding the maximum spanning tree with these weights for the edges and fitting the corresponding copulas we calculate pseudo observations using the h-functions for bivariate copulas from Definition 3.7. For each copula $C_{k,l}$, $\{k,l\}$ being an edge in $\boldsymbol{T_1}$, we obtain pseudo observations $u_{k|l} := h_{k|l}(u_k|u_l)$ and $u_{l|k} := h_{l|k}(u_l|u_k)$, where $h_{k|l}$ and $h_{l|k}$ are the respective h-functions of the copula.
In the second and all higher trees we work with the pseudo observations from the prior tree. Suppose these are given. Then in tree $\boldsymbol{T_n}$, with $1 < n < d$, we have pseudo observations from the prior tree of the form $u_{i|D}$, with $i \in \{1,\dots,d\}$, $D \subset \{1,\dots,d\}\backslash i$ and $|D| = n-1$. Then the empirical conditional Kendall's tau $\hat{\tau}_{k,l|D}$ is defined as the empirical unconditional Kendall's tau for the pseudo observations $u_{k|D}$ and $u_{l|D}$. The pseudo observations for the next tree can then be calculated from the fitted copula $C_{k,l|D}$ and its h-functions from Definition 3.7. However this time we are using not the original u-scale data, but the pseudo observations $u_{k|D}$ and $u_{l|D}$ from the last tree as arguments.
Adopting the basic idea of the algorithm of Dißmann et al. (2013), we also use the empirical (conditional) Kendall's tau as criterion for an algorithm to add a variable to an existing R-vine.

Algorithm 4.17 (Adding a variable to an R-vine)

Input : *A d-dimensional data set $\boldsymbol{X}$ with an R-vine distribution for the variables $X_1, \ldots, X_{d-1}$ and corresponding R-vine matrix M.*

Output: *An extension of the input R-vine distribution for all variables in $\boldsymbol{X}$ with corresponding R-vine matrix M^{add}, so that it is possible to find a closed form for the conditional distribution of $(X_d|X_1, \ldots, X_{d-1})$.*

$m^{add}_{i,j} := m_{i-1,j-1}$, *for* $i, j \in \{2, \ldots, d\}$
$m^{add}_{1,1} := d$
$I_1 := \{1, \ldots, d-1\},\ D_1 := \emptyset$
$x_1 := \arg\max_{i \in I_1} \hat{\tau}_{i,d}$
Fit the copula $C_{x_1,d}$ *and set* $m^{add}_{d,1} := x_1$
for $k = 2, \ldots, d-1$ **do**
  $I_k := I_{k-1} \backslash x_{k-1},\ D_k := D_{k-1} \cup x_{k-1}$
  $J_k := \{j \in I_k : (j, d|D_k)$ *is a valid edge in* $\boldsymbol{T_k}$ *fulfilling the proximity condtion*$\}$
  $x_k := \arg\max_{j \in J_k} \hat{\tau}_{j,d|D_k}$
  Fit the copula $C_{x_k,d|D_k}$ *and set* $m^{add}_{d-k+1,1} := x_k$
end

The steps in Algorithm 4.17 which refer to a pair copula fit include the selection of the copula family as well as the parameter estimation.
If there is only one edge $(j, d|D_k)$ fulfilling the proximity condition, of course the calculation of the empirical conditional Kendall's tau can be skipped. For example for $k = d-1$ there is always only one possibility to construct the last tree.
By means of a possible permutation of the variables, the newly added variable has not necessarily to be X_d.

Example 4.1 (Continued)
We will stick again to the example from the beginning of the chapter and assume there is a 6-th variable X_6. According to (4.40) *the R-vine matrix has the following form:*

$$M^{add} = \begin{pmatrix} 6 & & & & & \\ * & 5 & & & & \\ * & 1 & 1 & & & \\ * & 4 & 4 & 4 & & \\ * & 2 & 3 & 2 & 2 & \\ * & 3 & 2 & 3 & 3 & 3 \end{pmatrix}$$

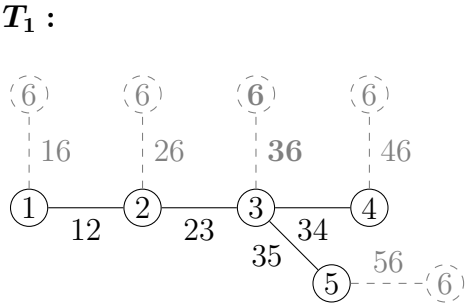
This is also the form we get by applying Algorithm 4.17 up to the second row. In the following we will show how the algorithm works for this example. The focus is on the structure selection and not on the copula fitting.
As it is possible to add variable 6 *to every node of $\boldsymbol{T_1}$ we at first calculate the empirical Kendall's tau for all variable pairs $\hat{\tau}_{i,6}$, $i = 1, \ldots, 5$. Then we add the edge $(i, 6)$ which has the greatest empirical Kendall's tau. In Figure 4.6 (a) all possibilities to add variable 6 are shown in gray with dashed edges. Assume that the greatest empirical Kendall's tau*

was observed for $\hat{\tau}_{3,6}$*. So we add the edge* $(3, 6)$*. This is indicated in Figure 4.6 (a) by bold font. So we can update the R-vine matrix as in Figure 4.6 (b). Once the corresponding copula was fitted, we have to decide where to connect the node* (36) *in* $\boldsymbol{T_2}$*.*
This time it is important at first to identify the possibilities we have, that do not violate the proximity condition. We end up with the alternatives to connect (36) *with* (23)*,* (34) *or* (35)*, also illustrated in Figure 4.6 (c). Observe that these are exactly the edges connecting to node* (3) *in* $\boldsymbol{T_1}$*, the node we chose to connect our new variable to. This time we need to calculate empirical conditional Kendall's tau for the possible edges* $(26|3)$*,* $(46|3)$ *and* $(56|3)$*. Assume we find the maximum of these three is* $\hat{\tau}_{26|3}$*, therefore we connect* (36) *with* (23) *and update the matrix as in Figure 4.6 (d).*
(23) *is connected to all edges in* $\boldsymbol{T_2}$*, so it is possible to connect* $(26|3)$ *with each node in* $\boldsymbol{T_3}$*, in line with the proximity condition. Assume that* $\hat{\tau}_{16|23} \geq \hat{\tau}_{46|23}, \hat{\tau}_{56|23}$*, so we add the edge* $(16|23)$ *to connect the nodes* $(26|3)$ *and* $(13|2)$*, as indicated in Figure 4.6 (e). The R-vine matrix is updated as in Figure 4.6 (f).*
In the next tree $\boldsymbol{T_4}$ *we only have one possibility to add the new node fulfilling the proximity condition, see Figure 4.6 (g). This is due to the fact, that we connected* $(26|3)$ *in the prior tree to a leaf. Here we can skip calculations for the empirical Kendall's tau and can directly update the matrix as in Figure 4.6 (h).*
In the last step the calculations can always be skipped as there is only one variable left. Figure 4.6 (i) and (j) show the construction of the last tree and the full R-vine matrix.

Algorithm 4.17 gives us an R-vine, of such kind that we are able to impute a missing data pattern with one missing variable by using the closed form for the inverse conditional distribution (4.38).
In the next section we will see how to extend the theory of adding variables to an existing R-vine for a general missing data pattern. The aim is again to do it in such a way that it is possible afterwards to use closed form conditional distributions to impute the missing variables by using the inverse transform sampling.

4.2.2 Adding several variables to an existing R-vine

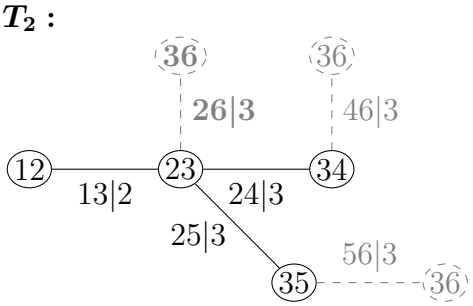
In the last section we saw how to add one variable X_d to an R-vine fitted on $(X_1, \ldots, X_{d-1})$, in such way that we are able to find a closed form for the conditional distribution function for $(X_d|X_1, \ldots, X_{d-1})$ and therefore can impute the respective missing data pattern. Now we extend this theory to a general missing data pattern. Assume again that we have a data set $(X_1, \ldots, X_d)$, but only $(X_1, \ldots, X_k)$ are observed, while $(X_{k+1}, \ldots, X_d)$ are missing. Given an R-vine distribution for $(X_1, \ldots, X_k)$, we can add the missing variables sequentially one by one to the existing R-vine with help of Algorithm 4.17. By doing so we already know that we can impute the missing variables afterwards from Section 4.1, as the procedure of adding a variable in the upper left corner of the R-vine matrix, can be seen as counterpart to deleting the first row and column. We only have to clarify which variable we start with each time.newpage



(a) Options to add (6) in $\boldsymbol{T_1}$

$$M^{add} = \begin{pmatrix} 6 & & & & & \\ * & 5 & & & & \\ * & 1 & 1 & & & \\ * & 4 & 4 & 4 & & \\ * & 2 & 3 & 2 & 2 & \\ 3 & 3 & 2 & 3 & 3 & 3 \end{pmatrix}$$

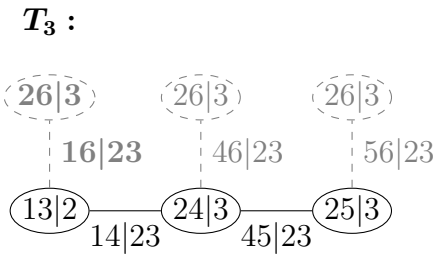
(b) Updated R-vine matrix after extending $\boldsymbol{T_1}$



(c) Options to add (36) in $\boldsymbol{T_2}$

$$M^{add} = \begin{pmatrix} 6 & & & & & \\ * & 5 & & & & \\ * & 1 & 1 & & & \\ * & 4 & 4 & 4 & & \\ 2 & 2 & 3 & 2 & 2 & \\ 3 & 3 & 2 & 3 & 3 & 3 \end{pmatrix}$$

(d) Updated R-vine matrix after extending $\boldsymbol{T_2}$



(e) Options to add (26|3) in $\boldsymbol{T_3}$

$$M^{add} = \begin{pmatrix} 6 & & & & & \\ * & 5 & & & & \\ * & 1 & 1 & & & \\ 1 & 4 & 4 & 4 & & \\ 2 & 2 & 3 & 2 & 2 & \\ 3 & 3 & 2 & 3 & 3 & 3 \end{pmatrix}$$

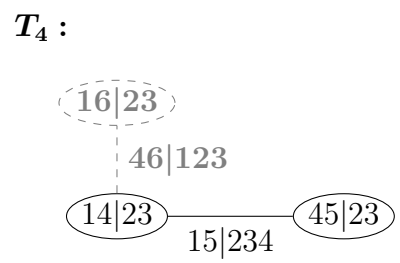
(f) Updated R-vine matrix after extending $\boldsymbol{T_3}$

Figure 4.6: Left: Illustration of the options to extend the R-vine structure adopting Algorithm 4.17 for Example 4.1. Solid/black: basic R-vine, dashed/gray: options to add node/edge, bold/gray: chosen option. Right: History of R-vine matrix updates.

Definition 4.18 (Start criterion for the sequential use of Algorithm 4.17) *Given a data set $(X_1, \dots, X_d)$ and an R-vine for the variables $(X_1, \dots, X_k)$, $k < d$, set $I := \{1, \dots, k\}$ and $I^* := \{k+1, \dots, d\}$. Then the following variable $\hat{i}$ will be added at first to the R-vine using Algorithm 4.17:*

$$\hat{i} := \arg\max_{i^* \in I^*} (\max_{i \in I} \hat{\tau}_{i,i^*}) \tag{4.41}$$

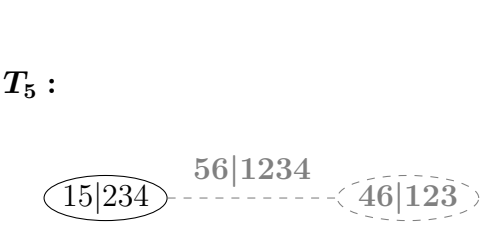
The criterion defined assures that the variable which has the highest Kendall's tau with one of the observed variables in $\boldsymbol{T_1}$ will be added at first. After adding one variable, the same criterion will be used on the extended R-vine and so forth.
Since we found a way to get a general missing data pattern imputable, by adding the



(g) Options to add (16|23) in $\boldsymbol{T_4}$

$$M^{add} = \begin{pmatrix} 6 & & & & & \\ * & 5 & & & & \\ 4 & 1 & 1 & & & \\ 1 & 4 & 4 & 4 & & \\ 2 & 2 & 3 & 2 & 2 & \\ 3 & 3 & 2 & 3 & 3 & 3 \end{pmatrix}$$

(h) Updated R-vine matrix after extending $\boldsymbol{T_4}$



(i) New tree $\boldsymbol{T_5}$

$$M^{add} = \begin{pmatrix} 6 & & & & & \\ 5 & 5 & & & & \\ 4 & 1 & 1 & & & \\ 1 & 4 & 4 & 4 & & \\ 2 & 2 & 3 & 2 & 2 & \\ 3 & 3 & 2 & 3 & 3 & 3 \end{pmatrix}$$

(j) Complete extended R-vine matrix

Figure 4.6: (Continued)

missing variables recursively to an R-vine fitted on the observed variables, we can now formulate a general imputation algorithm in the next section.

4.3 A general vine copula imputation method

The imputation algorithm we propose in this chapter will put to use the theory derived so far.

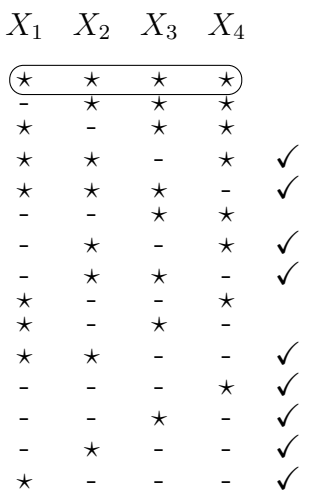
Algorithm 4.19 (Vine Copula Imputation)
Input : *A d-dimensional incomplete data set* $\boldsymbol{X}$.
Output: *An imputed data set for* $\boldsymbol{X}$.

1 *Fit an R-vine on the complete rows of* $\boldsymbol{X}$.
2 *Identify the ad hoc imputable missing data patterns applying Algorithm 4.15 and impute them.*
3 **while** *not all missing data patterns imputed* **do**
4 *Sort the data rows by missing data patterns and number of missing variables.*
5 *Pick the first missing data pattern that is not imputed and fit an R-vine to the observed variables, using all rows that are complete on the subset of the observed variables, including rows which were already imputed.*
6 *Add the missing variables to the R-vine sequentially with help of Algorithm 4.17 and the start criterion from Definition 4.18, using the complete cases on all variables, including already imputed observations.*
7 *Identify the ad hoc imputable missing data patterns of the extended R-vine applying Algorithm 4.15 and impute them.*
8 **end**

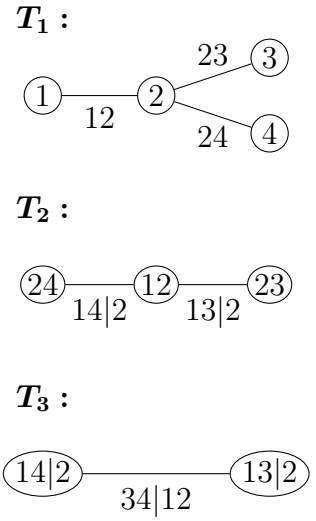
The first time the sorting in Row 4 is done, ensures that we do pick missing data patterns with less missing variables first. Doing so the fit on the observed variables can only be better than otherwise, as we have less restrictions in the structure selection. The repeated sorting is not really necessary but keeps the data clearly arranged.
One could argue, whether there should be a criterion to choose among the missing data patterns with the same number of missing variables, instead of just picking the first, but this topic is not addressed here to simplify matters.
Another approach, replacing the steps in Row 6 and 7, would be to add the variables one by one, however to impute the newly added variable directly from a reduced R-vine and do a new fit of the R-vine, before adding the next variable and so forth. This might be a better fit for the missing data pattern, but comes along with many R-vine structure selections, which are computationally intensive.

Example 4.2
Assume we have given a 4*-dimensional incomplete data set in which all missing data patterns occur, besides the one where all variables are missing. The possible missing data patterns are shown in Figure 4.7 (a). the observed variables are illustrated with '⋆', while missing values are represented by '-'. The framed row highlights the complete case, where all variables are given.*
Algorithm 4.19 starts with the fit of an R-vine on the complete cases. The R-vine tree structure and matrix obtained are shown in Figure 4.7 (b) and (c) respectively. Afterwards we identify the observed data patterns which are ad hoc imputable for this R-vine with help of Algorithm 4.15: $(3|124), (4|123), (13|24), (14|23), (34|12), (123|4), (124|3), (134|2)$

and $(234|1)$*. These missing data patterns, which are imputed subsequently, are also highlighted with check marks in Figure 4.7 (a).*
After we sorted the data again, the missing data patterns are illustrated in Figure 4.7 (d). This time the imputed values from the prior step are represented by '+'. The dashed frame shows the new complete cases. Only the first row was given from the beginning, all others result from the prior step. The first missing data pattern that we could not solve so far is the one, where only variable 1 is missing while the other variables are given. So we fit an R-vine to the complete data for the missing variables 2,3 and 4, see solid frame in Figure 4.7 (d). The R-vine structure is represented in Figure 4.7 (e) with solid/black lines/font and the R-vine matrix is the matrix M in Figure 4.7 (f). Then we add variable 1, by using Algorithm 4.17. The extended R-vine structure is represented by the dashed/gray extension in Figure 4.7 (e), and the extended R-vine matrix is given by M^{add} in Figure 4.7 (f). Unfortunately the obtained R-vine, we fitted to find a solution for the missing data pattern $(1|234)$*, does not give a solution to any of the 4 other missing data pattern left. So we impute the one missing data pattern also marked with an check mark in Figure 4.7 (d) and continue with the next missing data pattern.*
It is the missing data pattern, where only Variable 2 is missing, and the data looks as in Figure 4.7 (g) by now. Again we fit an R-vine to the complete data for the observed variables $(1, 3, 4)$ *and then extend it by adding Variable 2. The (extended) R-vines are represented in Figure 4.7 (h) and (i). Then we impute the ad hoc imputable missing data patterns for this R-vine, including three of the four remaining cases, tagged again with a check mark in Figure 4.7 (g).*
Afterwards the data is fully imputed, besides for the missing data pattern where the Variables 1 and 2 are missing, while 3 and 4 are observed. This time we have to use Algorithm 4.17 sequentially. However at first we fit the R-vine on the observed Variables 3 and 4. Note that in fact we only fit a pair copula here. Assume the start criterion from Definition 4.18 tells us to add Variable 2 at first. The obtained R-vine structure is shown by the gray dashed extension in Figure 4.7 (k) and represented by M_1^{add} *in Figure 4.7 (l). Then variable 1 is added to this R-vine, represented by the black dashed extension in Figure 4.7 (k) and* M_2^{add} *in Figure 4.7 (l). Finally this R-vine makes it possible to impute the last missing data pattern.*



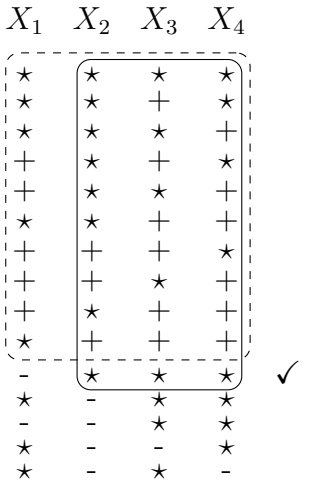
(a) Missing data patterns at beginning



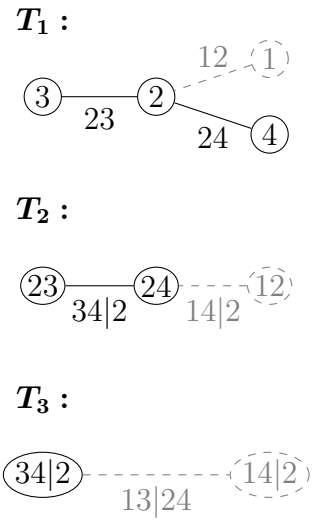
(b) Structure of R-vine fitted to complete cases in (a)

$$M = \begin{pmatrix} 3 & & & \\ 4 & 4 & & \\ 1 & 1 & 1 & \\ 2 & 2 & 2 & 2 \end{pmatrix}$$

(c) R-vine matrix for (b)



(d) Missing data patterns after imputation of ad hoc imputable missing data patterns of R-vine (b)



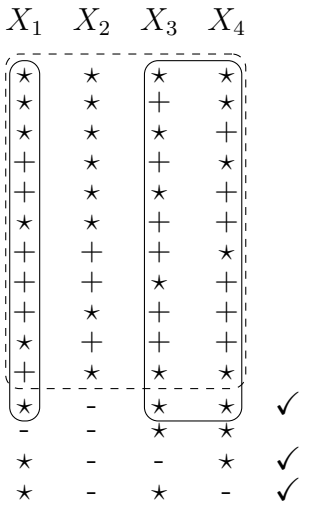
(e) Structure of R-vine fitted to complete cases in (d) (solid) and then extended (dashed)

$$M = \begin{pmatrix} 3 & & & \\ 4 & 4 & & \\ 2 & 2 & 2 & \end{pmatrix}$$

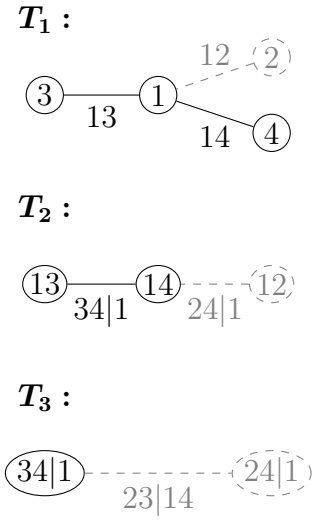
$$M^{add} = \begin{pmatrix} 1 & & & \\ 3 & 3 & & \\ 4 & 4 & 4 & \\ 2 & 2 & 2 & 2 \end{pmatrix}$$

(f) Respective R-vine matrices for (e)

Figure 4.7: Applying Algorithm 4.19 to Example 4.2.
Left: Missing data patterns and the stepwise imputation. '⋆' observed, '-' missing, '+' imputed. dashed frame: imputed data so far, solid frame: data used for R-vine fit, '✓' ad hoc imputable missing data pattern for the full R-vine in Middle.
Middle: Structure of R-vine fitted to data. solid: solid frame Left, dashed: extended.
Right: R-vine matrices for the R-vines in Middle.



(g) Missing data patterns after imputation of ad hoc imputable missing data patterns of R-vine (e)

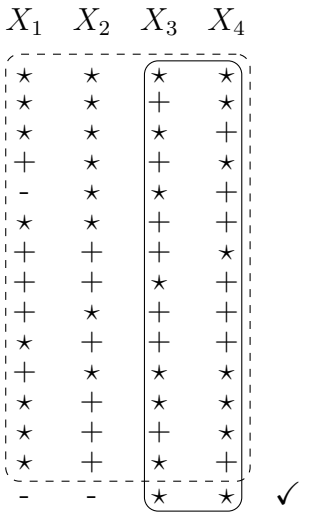


(h) Structure of R-vine fitted to complete cases in (g) (solid) and then extended (dashed)

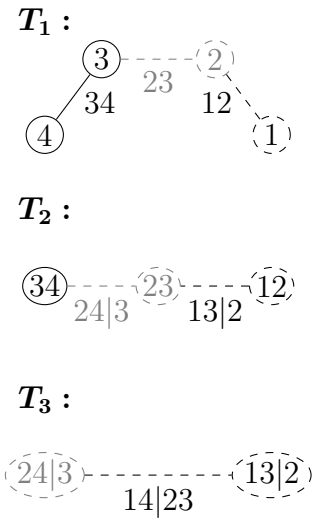
$$M = \begin{pmatrix} 3 & & \\ 4 & 4 & \\ 1 & 1 & 1 \end{pmatrix}$$

$$M^{add} = \begin{pmatrix} 2 & & & \\ 3 & 3 & & \\ 4 & 4 & 4 & \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

(i) Respective R-vine matrices for (h)



(j) Missing data patterns after imputation of ad hoc imputable missing data patterns of R-vine (h)



(k) Structure of R-vine fitted to complete cases in (j) (solid) and then extended (dashed) twice - first extension gray, second black

$$M = \begin{pmatrix} 4 & \\ 3 & 3 \end{pmatrix}$$

$$M_1^{add} = \begin{pmatrix} 2 & & \\ 4 & 4 & \\ 3 & 3 & 3 \end{pmatrix}$$

$$M_2^{add} = \begin{pmatrix} 1 & & & \\ 4 & 2 & & \\ 3 & 4 & 4 & \\ 2 & 3 & 3 & 3 \end{pmatrix}$$

(l) Respective R-vine matrices for (k)

Figure 4.7: (Continued)

Chapter 5

Simulation Study Based on Abalone Data

In our simulation study we will use the abalone data from Lichman (2013) as foundation. Then we transform it to the u-scale, fit an R-vine to the transformed data set and simulate from this R-vine. Afterwards we will artificially delete some data, apply several imputation methods and finally evaluate how well they did by comparing imputed and complete data before missingness was introduced using different measures. First we will proceed introducing the data set.

5.1 The abalone data

Data set description:

> "Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.
>
> From the original data examples with missing values were removed (the majority having the predicted value missing), and the ranges of the continuous values have been scaled for use with an ANN [artificial neural network](by dividing by 200)."

In Table 5.1 all collected variables are described. We will focus on female abalone, as we built our imputation method for continuous variables and cannot handle the nominal variable *sex*. Another choice would be possible. After deleting row 2052 of the original data due to a non reasonable outlier in the variable *height*, we further reduce the 1306 observations to 1000 for a better interpretability of missing percentages, by sampling 1000 indexes from 1 to 1306 without replacement in R with *set.seed(101)*. Table 5.2 shows a summary of the remaining eight variables with 1000 observations each. Notice here that the variable *height* has remarkably few unique values (37) and therefore is kind of discrete

as the number of rings is. As we want to focus on continuous variables, or appropriate quasi-continuous ones, we drop the variables *height* and *rings*.

Name	Data Type	Meas.	Description
Sex	nominal		M, F, and I (infant)
Length	continuous	mm	Longest shell measurement
Diameter	continuous	mm	perpendicular to length
Height	continuous	mm	with meat in shell
Whole weight	continuous	grams	whole abalone
Shucked weight	continuous	grams	weight of meat
Viscera weight	continuous	grams	gut weight (after bleeding)
Shell weight	continuous	grams	after being dried
Rings	integer		+1.5 gives the age in years

Table 5.1: Abalone data set variables.

	len	diameter	height	whole.wt	shucked.wt	viscera.wt	shell.wt	rings
Min.:	0.275	0.195	0.015	0.080	0.031	0.021	0.025	5
1st Qu.:	0.525	0.410	0.140	0.741	0.298	0.159	0.215	9
Median:	0.590	0.465	0.160	1.032	0.438	0.224	0.295	10
Mean:	0.579	0.455	0.157	1.047	0.446	0.231	0.303	11.13
3rd Qu.:	0.640	0.500	0.175	1.310	0.566	0.296	0.380	12
Max.:	0.815	0.650	0.250	2.657	1.488	0.590	1.005	29
#Unique:	91	80	37	865	718	547	416	22

Table 5.2: Summary of abalone data for females with minimum, 1st quartile, median, mean, 3rd quartile, maximum and the number of unique values per variable.

In Figure 5.1, the *pairs.panels()* from the *psych* R-package, see Revelle (2015), was used to show pairwise empirical Kendall's tau coefficients in the upper triangle, pairwise scatterplots in the lower triangle, and histograms on the diagonal for all variables. The gray points in the scatter plots indicate the mean of both variables.

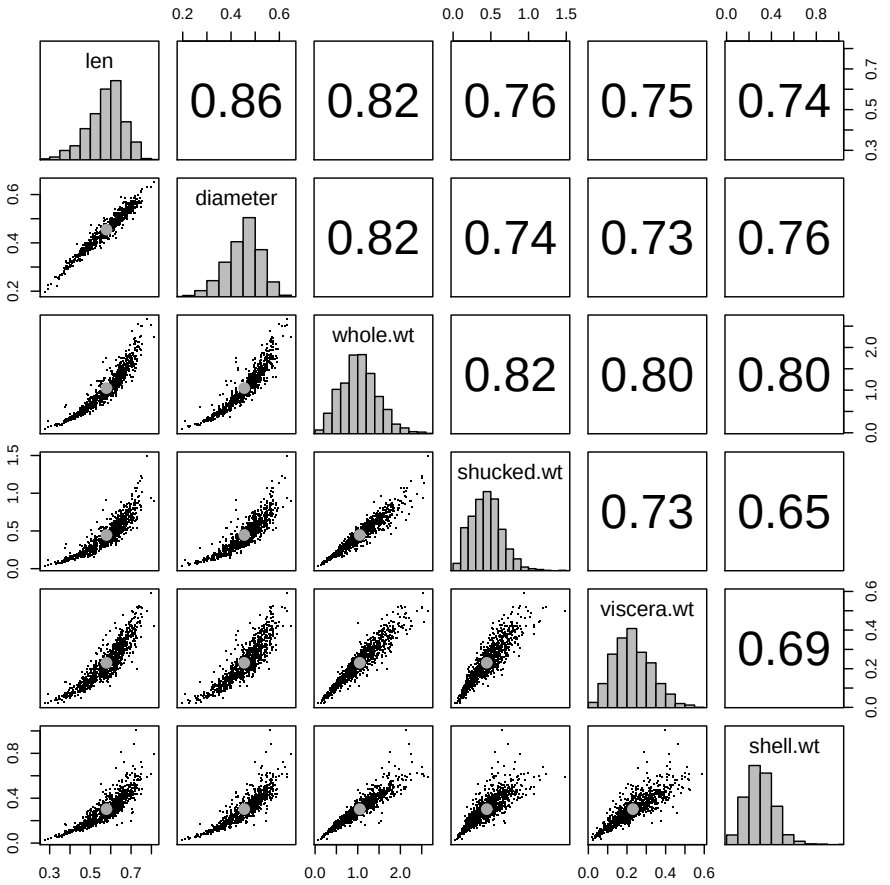


Figure 5.1: Scatter plots, histograms and correlations by Kendall's tau.

5.2 Transformation of the margins

With the goal in mind to fit a vine structure to our data, it is necessary to have uniformly distributed margins first. Factoring in the shape of the marginal distributions, we are not satisfied with the Normal Gaussian fit and will use a more general distribution family. Barndorff-Nielsen (1977) introduced the Generalized Hyperbolic distribution. The Generalized Hyperbolic distribution is defined as normal mean-variance-mixture with a Generalized Inverse Gaussian mixing distribution in the following way, see McNeil et al. (2005) p.77 ff. for the more general multivariate definition.

Definition 5.1 (Univariate Generalized Hyperbolic distribution)
A random variable X has a Generalized Hyperbolic distribution if

$$X = \mu + \gamma W + \sigma\sqrt{W}Z \tag{5.1}$$

where

1. $Z \sim \mathcal{N}(0,1)$ *is standard normal distributed random variable,*

2. $\mu, \gamma, \sigma \in \mathbb{R}$ *and*

3. $W \geq 0$*, random variable independent of* Z*, with* $W \sim \mathcal{N}^-(\lambda, \chi, \psi)$

where $\mathcal{N}^-$ *denotes the Generalized Inverse Gaussian distribution.*

Barndorff-Nielsen and Halgreen (1977) showed that the Generalized Hyperbolic distribution has infinitie divisiblity.

Definition 5.2 (Univariate Generalized Inverse Gaussian distribution)
A random variable $W \sim \mathcal{N}^-(\lambda, \chi, \psi)$ *has the probability density function*

$$f_W(w) = \left(\frac{\psi}{\chi}\right)^{\frac{\lambda}{2}} \frac{w^{\lambda-1}}{2K_\lambda(\sqrt{\chi\psi})} exp\left\{-\frac{1}{2}\left(\frac{\chi}{w} + \psi w\right)\right\}, \tag{5.2}$$

with the parameters satisfying

$$\begin{aligned} &\chi > 0,\ \psi \geq 0 \quad &&if\ \lambda < 0,\\ &\chi > 0,\ \psi > 0 \quad &&if\ \lambda = 0,\\ &\chi \geq 0,\ \psi > 0 \quad &&if\ \lambda > 0, \end{aligned}$$

and $K_\lambda(\cdot)$ *denoting a modified Bessel function of third kind.*
The n*-th moment of* W *has the form*

$$E[W^n] = \left(\frac{\chi}{\psi}\right)^{\frac{n}{2}} \frac{K_{\lambda+n}(\sqrt{\chi\psi})}{K_\lambda(\sqrt{\chi\psi})} \tag{5.3}$$

Definition 5.3 (Modified Bessel function of third kind)
The modified Bessel function of third kind is defined as

$$K_\lambda(x) = \frac{1}{2}\int_0^\infty w^{\lambda-1} exp\left\{-\frac{1}{2}x(w + w^{-1})\right\} dw,\ x > 0. \tag{5.4}$$

Then it follows directly from the construction in Definition 5.1 that:

1. $E[X] = \mu + \gamma E[W]$
2. $Var[X] = \gamma^2 Var[W] + \sigma^2 E[W]$
3. $(X|W = w) \sim \mathcal{N}(\mu + \gamma w, \sigma^2 w)$

From the last property we can derive the closed form of the density of the GH distribution as

$$\begin{aligned} f_X(x) &= \int_0^\infty f_{X|W}(x|w) f_W(w) dw \qquad (5.5)\\ &= \dots \\ &= \frac{\left(\sqrt{\frac{\psi}{\chi}}\right)^\lambda \left(\psi + \frac{\gamma^2}{\sigma^2}\right)^{\frac{1}{2}-\lambda}}{\sqrt{2\pi}\sigma K_\lambda(\sqrt{\chi\psi})} \times \frac{K_{\lambda-\frac{1}{2}}\left(\sqrt{(\chi + q(x)^2)\left(\psi + \frac{\gamma^2}{\sigma^2}\right)}\right) exp\left\{\frac{q(x)\gamma}{\sigma}\right\}}{\left(\sqrt{(\chi + q(x)^2)\left(\psi + \frac{\gamma^2}{\sigma^2}\right)}\right)^{\frac{1}{2}-\lambda}}, \end{aligned}$$

with $q(x) = \frac{x-\mu}{\sigma}$.

This GH distribution family includes many special cases as the Normal Inverse Gaussian ($\lambda = -\frac{1}{2}$), Variance Gamma ($\chi = 0, \lambda > 0$), Generalized Hyperbolic Student-t ($\psi = 0, \lambda < 0$) and Hyperbolic ($\lambda = \frac{d+1}{2}$) distributions.

The *ghyp* R-package, see Luethi and Breyman (2013), which we use for fitting internally, works with the $(\lambda, \chi, \psi, \mu, \sigma, \gamma)$ parametrization, but does the fitting with an alternative $(\chi, \bar{\alpha}, \mu, \sigma, \gamma)$ parametrization. Also another parametrization $(\lambda, \alpha, \delta, \beta, \mu)$ is wide spread. Both alternatives are also implemented in the *ghyp*-package and it is easy to get from the $(\lambda, \chi, \psi, \mu, \sigma, \gamma)$ parametrization to the other ones by:

1. $(\lambda, \chi, \psi, \mu, \sigma, \gamma) \rightarrow (\chi, \bar{\alpha}, \mu, \sigma, \gamma)$:
   $\bar{\alpha} = \sqrt{\chi\psi}$

2. $(\lambda, \chi, \psi, \mu, \sigma, \gamma) \rightarrow (\lambda, \alpha, \delta, \beta, \mu)$:
   $\alpha = \frac{1}{\sigma}\sqrt{\psi + \frac{\gamma^2}{\sigma^2}},\ \beta = \frac{\gamma}{\sigma^2},\ \delta = \sqrt{\chi\sigma^2}$

Fitting the abalone data using the *fit.ghypuv()* function of the *ghyp*-package in R, we get the following parameters shown in Table 5.3.

	len	diameter	whole.wt	shucked.wt	viscera.wt	shell.wt
λ	5.156	6.445	17.579	9.041	14.404	12.184
$\bar{\alpha}$	4.223	2.020	0.110	0.575	11.961	0.545
μ	0.719	0.566	-0.741	-0.162	-0.161	-0.125
σ	0.069	0.058	0.089	0.011	0.039	0.028
γ	-0.140	-0.111	1.788	0.607	0.391	0.428

Table 5.3: Parameters of the fitted GH distribution for all variables from the female abalone data.

In Figure 5.2 one can see histograms of the original data with the densities of fitted generalized hyperbolic distributions and qq-plots of sample quantiles vs. theoretical, indicating that the Generalized Hyperbolic distribution is an appropriate choice.

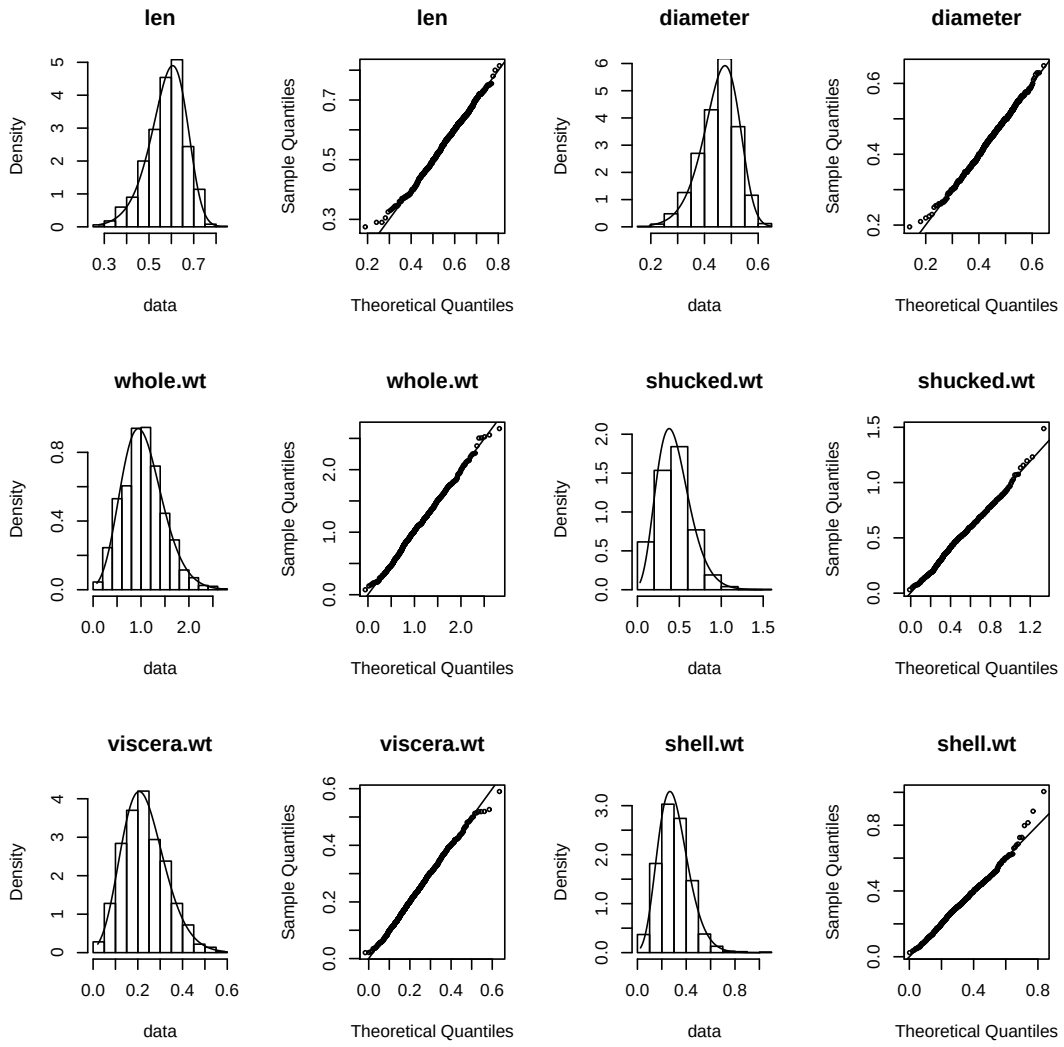


Figure 5.2: Repeating from left to right: Histogram of original data with density of fitted generalized hyperbolic distribution, qq-plot of sample quantiles vs. theoretical.

In the next step we transform the variables with the probability integral transform, i.e. $u_i := \hat{F}_i(x)$, using the *pghyp()* function of the same package, where $\hat{F}_i$ is the estimated marginal distribution function for Variable i. Figure 5.3 shows scatter plots, histograms and contour plots of the margins after the probability integral transform of the fitted GH distributions.

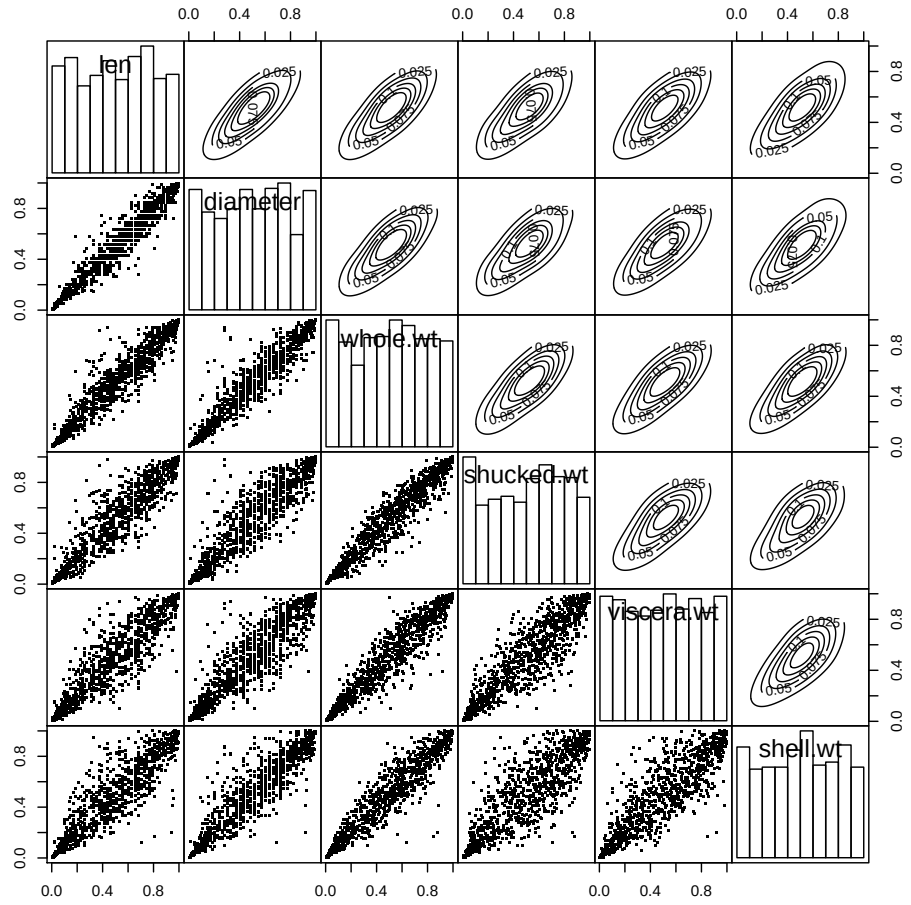


Figure 5.3: Scatter plots, histograms and contour plots of the uniform transformed margins.

The back transformation of the data at the end can be realized easily by the quantile function *qghyp()*, which is also included in the *ghyp* package.

5.3 Fitting an R-vine to the data

Since we determined how to transform our data to the u-scale we can now fit an R-vine using the *RVineStructureSelect()* function from the *VineCopula* R-package. The structure is illustrated in Figure 5.4. The variables are mapped to the numbers 1 to 6 in the order *length*, *diameter*, *whole weight*, *shucked weight*, *viscera weight* and *shell weight*. Note that we only allowed for the following copula families: Independence, Gaussian, Student t, Frank, Clayton, Gumbel as well as the rotations for the latter two. For more details on the copula families see Appendix A or Nelsen (2006).

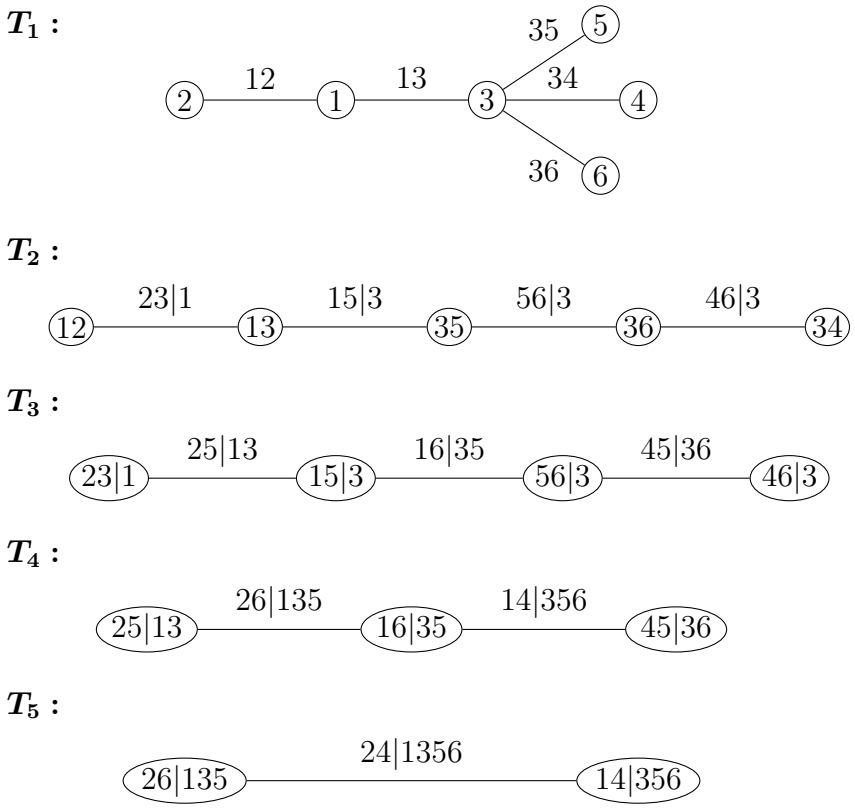


Figure 5.4: 6 dimensional R-vine tree structure for the *abalone* data.

In Table 5.4 all families, parameters, Kendall's taus and tail dependence coefficients are summarized. The log-likelihood of the fit is 6951.34, while the AIC is -13778.35.

Tree	Edge	Family	Paramter1	Parameter2	Tau	UTD	LTD
1	1,2	180° Gumbel	6.44	-	0.84	-	0.89
	3,1	180° Gumbel	5.32	-	0.81	-	0.86
	3,4	180° Gumbel	5.68	-	0.82	-	0.87
	3,5	180° Gumbel	4.96	-	0.80	-	0.85
	6,3	180° Gumbel	4.85	-	0.79	-	0.85
2	3,2;1	Student-t	0.39	5.17	0.26	0.15	0.15
	5,1;3	180° Gumbel	1.05	-	0.05	-	0.06
	6,4;3	Frank	-4.99	-	-0.46	-	-
	6,5;3	270° Gumbel	-1.16	-	-0.14	-	-
3	5,2;3,1	270° Gumbel	-1.04	-	-0.03	-	-
	6,1;5,3	Clayton	0.11	-	0.05	-	0.00
	5,4;6,3	Student-t	-0.38	4.66	-0.25	0.01	0.01
4	6,2;5,3,1	Frank	1.53	-	0.17	-	-
	4,1;6,5,3	Student-t	0.23	12.71	0.14	0.01	0.01
5	4,2;6,5,3,1	Clayton	0.07	-	0.03	-	0.00

Table 5.4: Summary for the R-vine fit for the *abalone* data set with copula families, parameters, Kendall's taus, upper and lower tail dependence for each edge

5.4 Creating MCAR data

After the R-vine was fitted to the real data, we can simulate data sets from it. However as we need incomplete data sets, we have to specify a way to create missing values first. The following algorithm based on the *MCAR()* function of the *CoImp* R-package, see Di Lascio, Francesca M. L. and Giannerini (2014), creates an MCAR data set based on a full data set.

Algorithm 5.4 *(Creating MCAR data)*
Input:

- *data: a data set with n.var columns (variables) and n.obs rows (observations)*
- *perc.miss: a percentage between* 0 *and* 1
- *max.var.miss: an integer* $<$ *n.var*

Output:

- *data.incomplete: a data set with* (1$-$*perc.miss)*% *complete cases, for all incomplete cases at most max.var.miss variables are missing*

1. *Choose randomly without replacement* ***which rows*** *shall contain incomplete data by creating a vector*

```
Rows.incompl<-sample(1:n.obs,n.obs*perc.miss)
n.rows.incompl<-length(Rows.incompl)
```

2. *For each incomplete row choose randomly* ***how many variables*** *shall be missing (at least* 1*, at most max.var.miss) by creating a second vector of the same length as in the prior step*

```
N.var.miss<-sample(1:max.var.miss,n.rows.incompl,replace=TRUE)
```

3. *After determining how many variables shall be missing in each row in the prior step, we now choose randomly without replacement* ***which variables*** *shall be missing in each incomplete row and set them to* NA

```
for (i in 1:n.rows.incompl) {
positions<-sample(1:n.var, N.var.miss[i])
data[Rows.incompl[i],positions]<-NA
}
```

Using Algorithm 5.4 to create MCAR data sets from simulated complete data sets, we are now able to test various imputation techniques including our own vine imputation method. However at first we describe the general setup how we simulate data sets. We will also introduce a notation to keep track of all the different complete, incomplete and imputed data sets. Additionally we have to specify the functions and parameter specifications, which we use for the implementation.

5.5 Test setup and software specifications

In this section we will describe the general procedure how we create our test cases and imputations, also specifying the R-functions with their parameters for the implementation and the measures for the evaluation.

5.5.1 Data setup

Let's denote by D the real complete abalone data set of 1000 observations on 6 variables after transforming each variable to the u-scale and by RV the R-vine fitted to it in Section 5.3. The empirical estimate of a quantity θ derived from D will be denoted by $\hat{\theta}$.
Then in all our test scenarios we will simulate from this RV K data sets $D_1, \dots, D_K$ of same length. The respective empirical estimates of θ derived from these data sets will be denoted by $\hat{\theta}_1, \dots, \hat{\theta}_K$.
So far the data sets created are complete and in the next step we introduce the missingness. Using an appropriate method, some values in each data set D_k, $k = 1, \dots, K$ are deleted. The resulting data sets are then denoted by D_k^M respectively.
Now we can use an imputation method I to impute each incomplete data set D_k^M. As we want to use multiple imputation, we perform the imputation L times. The data sets created in this way, are denoted by $D_k^{I(l)}$, where $l = 1, \dots, L$, and the empirical estimates for θ derived from each data set $D_k^{I(l)}$ are denoted by $\hat{\theta}_k^{I(l)}$.
Finally we also introduce an estimate $\hat{\theta}_k^I$ for each $k \in \{1, \dots, K\}$ that pools the L empirical estimates $\hat{\theta}_k^{I(l)}$, $l = 1, \dots, L$ to a single value. Examples for pooling can be the mean or the median.
The procedure described so far is also illustrated in Figure 5.5 and the notation introduced will be used throughout the remaining part of the chapter.

5.5.2 Software specifications

Here we shortly specify which R-packages and functions we use for the implementation. If not indicated otherwise the parameter defaults were used. All methods will use the u-scale data as input.

For the implementation of our own vine copula imputation method, we will mainly use the *RVineAdhocImpute()* and *RVineImpute()* functions from the *VineCopulaImpute* package, which is part of this thesis. R-vines are fitted using the *RVineSeqEst()*, the *RVineCopSelect()* and the *RVineStructureSelect()* functions from the *VineCopula* package. In each

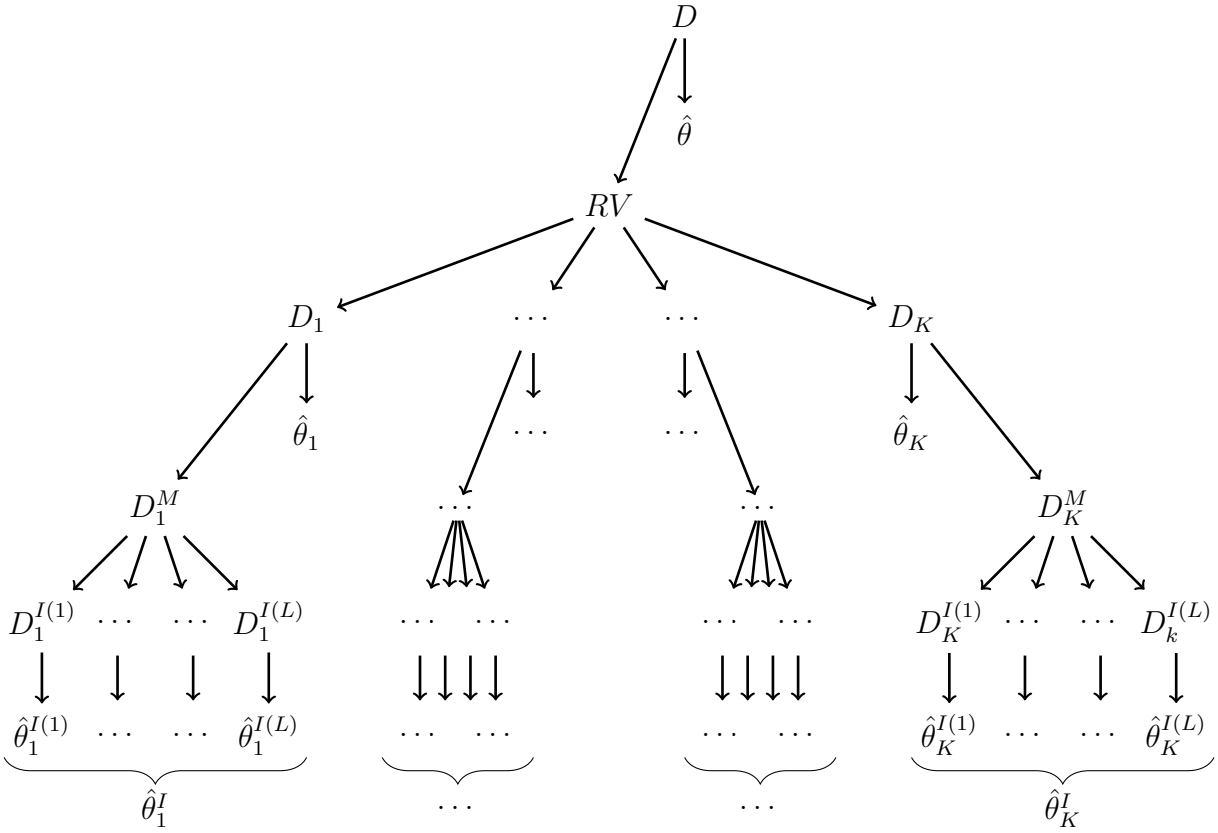


Figure 5.5: Illustration of how complete data sets D_k, incomplete data sets D_k^M and imputed data sets $D_k^{I(l)}$, $k = 1, \ldots, K$, $l = 1, \ldots, L$, were created in the simulation study originating from the R-vine RV fitted to the real data D. From these data sets, the quantities $\hat{\theta}$, $\hat{\theta}_k$, $\hat{\theta}_k^{I(l)}$ are determined, as well as a summarizing quantity $\hat{\theta}_k^I$.

step where copula families have to be determined, we choose among the Independence, Gaussian, Student t, Frank, Clayton or Gumbel copula as well as among the rotations for the latter two.

For the multivariate normal Joint Modeling we use the *norm* R-package with the functions *prelim.norm()*, *em.norm()* and *da.norm(·, steps=500)*.

The Fully Conditional Specification is implemented using the *mice* R-package and the corresponding *mice(·,maxit=20)* function.

The *CoImp(·,n.marg=6,model = list(normalCopula(0.5, dim=6,dispstr="ex"),claytonCopula(10, dim=6)))* function from the *CoImp* package is used for the CoImp method.

Last for the k nearest neighbor method, we use the *kNN(·, numFun=weightedMean)* function from the *VIM* package together with the *weightedMean()* function from the *laeken* package as argument.

5.5.3 Evaluation

Our goal is to determine how well an imputation method works. Therefore we measure the effect on some statistic θ when using different imputation methods $I_1, \ldots, I_P$. Using the notation from Section 5.5.1, in our test setup we can derive the empirical statistics $\hat{\theta}$, $\hat{\theta}_k$, $\hat{\theta}_k^{I_p(l)}$ and the pooled $\hat{\theta}_k^{I_p}$ for $p = 1, \ldots, P$. The method of choice for the pooling will be the median, as it is more robust to outliers than the mean.
The test statistics we choose for comparison are the pairwise Kendall's tau coefficients and the pairwise alternative upper/lower tail dependence coefficients from Defintion 3.12. These are both bivariate measures and empirical versions are given in Definition 3.9 and Definition 3.13. The empiricial versions are easy to compute and are appropriate statistics for the comparison, because a good imputation method should capture the overall dependence as well as the dependence in the tails.
Since we have to determine the empirical versions of the bivariate dependence coefficients for all variable pairs (k, l), $k, l \in \{1, \ldots, 6\}$ with $k \neq l$, we have to calculate 15 empirical statistics each, as there are 15 unique variable pairs.
Due to the fact that we cannot directly derive the real population values for all pairwise dependence measures from the R-vine, we use a large sample of 100,000 observations simulated from the R-vine distribution and derive the empirical versions. In this very large sample the empirical values will be very close to the real ones.
The resulting empirical Kendall's tau matrix is given by

$$\hat{\tau}^{lrg} = \begin{pmatrix} 1.000 & 0.845 & 0.813 & 0.756 & 0.737 & 0.730 \\ 0.845 & 1.000 & 0.813 & 0.739 & 0.727 & 0.753 \\ 0.813 & 0.813 & 1.000 & 0.824 & 0.799 & 0.794 \\ 0.756 & 0.739 & 0.824 & 1.000 & 0.717 & 0.654 \\ 0.737 & 0.727 & 0.799 & 0.717 & 1.000 & 0.683 \\ 0.730 & 0.753 & 0.794 & 0.654 & 0.683 & 1.000 \end{pmatrix},$$

and the empirical tail dependence matrix is given by

$$\hat{\varrho}_{L/U}^{lrg} = \begin{pmatrix} 1.000 & 0.970 & 0.956 & 0.927 & 0.915 & 0.913 \\ 0.864 & 1.000 & 0.957 & 0.919 & 0.908 & 0.927 \\ 0.814 & 0.819 & 1.000 & 0.962 & 0.949 & 0.947 \\ 0.712 & 0.672 & 0.829 & 1.000 & 0.904 & 0.863 \\ 0.668 & 0.651 & 0.793 & 0.628 & 1.000 & 0.879 \\ 0.652 & 0.708 & 0.780 & 0.474 & 0.563 & 1.000 \end{pmatrix},$$

$$\text{with } \hat{\varrho}_{L/U}^{lrg}(k,l) = \begin{cases} \hat{\varrho}_L^{lrg}(k,l) & \text{, for } k \leq l \\ \hat{\varrho}_U^{lrg}(k,l) & \text{, for } k > l. \end{cases}$$

We can see that all pairwise Kendall's tau coefficients are positive and quite high in a range between 0.654 and 0.845. Likewise the alternative upper and lower tail dependence coefficients are all positive. Especially the lower tail dependence coefficients have high

values in a range between 0.863 and 0.970. The upper tail dependence coefficients range between 0.474 and 0.864.
However as the large sample statistic $\hat{\theta}^{lrg}$ might differ from the statistics $\hat{\theta}_k$, $k = 1, \ldots, K$, which we get from smaller samples, it would be unfair to compare $\hat{\theta}^{lrg}$ with the statistic $\hat{\theta}_k^{I_p}$ derived from an imputed data set. Instead we will look at the difference $\Delta_k^{I_p} := \hat{\theta}_k^{I_p} - \hat{\theta}_k$, because the most we can expect from an imputation algorithm is that it identifies the dependence structure of the full data set D_k, and not from an underlying distribution, which might not be nicely represented in the simulated data set.
Obviously the better the imputation algorithm works, the closer should $\Delta_k^{I_p}$ be to 0. As all Kendall's tau coefficients and alternative tail dependence coefficients are clearly positive, a negative (positive) $\Delta_k^{I_p}$ indicates that the pooled estimate from the imputed data sets is lower (higher) than the estimate from the complete data set. In our test setup we get K distances $\Delta_k^{I_p}$, one for each $k \in \{1, \ldots, K\}$. Finally we will analyze the different imputation methods graphically by plotting the K distances for each imputation method in a single box plot and then compare the box plots from the different imputation methods.
In order to check also multivariate behavior, we will also look at the log-likelihood described in Definition 3.24. In contrast to the pairwise dependence measures, only one estimate per data set is obtained. In detail, we compare the log-likelihood of RV with the the simulated complete data sets against the pooled one of the imputed data sets.

5.6 Simulations and results

Next we test our imputation method in several scenarios and benchmark it against other methods.

5.6.1 Checks

Scenario

In the first scenario we construct a special good-natured scenario, which should be easy to impute using our own vine copula imputation method. We want to check whether the algorithm is working correctly and which steps are potentially introducing uncertainty.
In the general test data setup notation from Section 5.5.1, we choose $K = 100$ and $L = 1$. This means we simulate 100 data sets and impute each once by every imputation method. The number of observations in the simulated data sets $D_k,\ k = 1, \ldots, K$ is chosen to be $10,000$. This quite high number of observations ensures that the empirical Kendall's tau values are relatively close to the real ones, not influenced too much by random variation, and even with a high percentage of incomplete observations the complete cases are still numerous enough for a decent model estimation.
Then the missingness in the data sets is introduced. We create missing values, so that 90% of the observations are incomplete. We only create missing data patterns which are ad hoc imputable for the R-vine RV as identified by Algorithm 4.15, as we know that the vine imputation method could perform very well imputing these missing data patterns, if it correctly identifies the R-vine structure. However we distinguish two scenarios. In the

first scenario we only create missing data patterns, where at most 3 variables are missing, and the second one has no restriction, so that at most 5 variables are missing. The corresponding data sets are denoted by $D_1^{M(3)}, \ldots, D_{100}^{M(3)}$ and $D_1^{M(5)}, \ldots, D_{100}^{M(5)}$. These scenarios help us to check if and how much variation is added to the results when more variables are missing. In detail, the ad hoc imputable missing data patterns are given by (2), (4), (1, 2), (2, 4), (4, 6), (1, 2, 4), (1, 2, 5), (2, 4, 6), (4, 5, 6), (1, 2, 4, 5), (1, 2, 4, 6), (1, 2, 5, 6), (2, 4, 5, 6), (3, 4, 5, 6), (1, 2, 3, 4, 5), (1, 2, 3, 4, 6), (1, 2, 3, 5, 6), (1, 2, 4, 5, 6), (1, 3, 4, 5, 6) and (2, 3, 4, 5, 6). For every observation which shall be incomplete we choose randomly one of the ad hoc imputable missing data patterns, with either at most 3, or at most 5, variables missing. The variable coding from 1 to 6 is the same as in Section 5.3.
After specifying the missingness, we can now specify the imputation methods we want to use for our scenario to check the vine copula imputation method. Let $(\mathcal{V}, \mathcal{B}(\mathcal{V}), \theta(\mathcal{B}(\mathcal{V}))$ describe the R-vine from Section 5.3, with $\mathcal{V}$ being the tree-structure, $\mathcal{B}(\mathcal{V})$ the copula families for every edge, and $\theta(\mathcal{B}(\mathcal{V}))$ the parameters for the family of each copula. In the first test method we pass the exact model including the tree structure, copula families and parameters to the imputation algorithm. As the data was simulated from this model and only ad hoc imputable missing data patterns were created, we expect good results. This scenario is mostly for a test whether the algorithm for the imputation works correctly. This will be called the *oracle method*. Furthermore we test three other methods. These methods will be named *parameter method*, *family method* and *free method*. In all three methods an R-vine is fitted to the complete part of the data and the imputation methods try to simulate from these R-vines the ad hoc imputable missing data patterns. In the *parameter method* we let the algorithm estimate the copula parameters by itself, while we still pass the structure and the copula families. In the *family method* the algorithm additionally has to estimate the copula families, while we still pass the structure. Last, in the *free method* we do not pass any information about the real distribution and use Algorithm 4.19 in its plain form.
Although we want to check the R-vine imputation method we will run also the multivariate normal Joint Modeling and the Fully Conditional Specification with our tests, denoted by *norm method* and *fcs method* respectively. This allows us to directly see in which step our algorithm possibly loses most in comparison. The methods chosen for comparison are the state-of-the-art multiple imputation methods and we can expect them to work well.
In all checks for this scenario we will compare only the pairwise Kendall's tau values. Later we will also use the tail dependence measures and the log-likelihood.

Results

The resulting box plots of the Kendall's tau distances $\Delta_k^{I_p}$, for $k = 1, \ldots, K$ and $I_p \in$ {oracle, paramter, family, free, fcs, norm} are shown in Figure 5.6 and Figure 5.7 for the case with at most 3 variables missing, and in Figure 5.8 and Figure 5.9 for the case with at most 5 variables missing. We will start with the discussion of the results for the case with at most 3 variables missing.

The oracle method performs well for all variables pairs. The median of the distance measures is always at the null and the variation is low, at a level of about 0.005 in both

directions symmetrically. This result is not surprising, as from the beginning the oracle method knew the underlying model and could simulate from it. Nonetheless we can see that the simulation seems to work correctly here.
The family and parameter methods' median is also at the null for all variable pairs, but the variation is about twice as high as for the oracle case. This might be a direct consequence that this time the underlying model was not known and parameters or parameters and families had to be estimated on the complete part of the data first. As we can see clearly for the variable pairs *viscera.wt-shell.wt* and *whole.wt-viscera.wt*, sometimes the family method has less variation in the distance measure than the parameter method, even though we gave it less information about the underlying model. However this might just be the explanation of this effect. Since we do not restrict the R-vine fit by fixing the families, we allow for more flexibility, which might result in a better fit. When we take a closer look which families in the R-vine fit to the complete part of the data were chosen, we see that in average 3.4 of the 15 families, that one has to specify in a 6-dimensional R-vine, were chosen differently. A more detailed summary is given in Table 5.5.

Different families	0	1	2	3	4	5	6	7	8	9	10	11
Frequency	1	9	26	26	22	7	0	0	4	3	1	1

Table 5.5: Frequency table of how many families were estimated differently in the R-vine fit to the complete part of the data for the family method compared to the real underlying R-vine RV

The last vine copula method which we evaluate in this scenario is the free method, which was an unchanged implementation of our imputation Algorithm 4.19. For 5 variable pairs the median is not at the null. These are the pairs *diameter-whole.wt*, *diameter-shucked.wt*, *diameter-viscera.wt*, *shucked.wt-viscera.wt* and *shucked.wt-shell.wt*. In the other 10 out of 15 variable pairs the median hit the null. However at the first glance we can see that we have now by far more variation as in the previous methods. Some outliers of the distance measure for the variable pair *shucked.wt-shell.wt* are even about -0.2. One reason for the high variation might of course be that we only imputed each data set once. Later we will use a multiple imputation approach to check how this will change. Another reason is of course that the R-vine model fitted to the complete part of the data can have a completely different structure and not all of the missing data patterns are any longer ad hoc imputable. For some variable pairs we do not see a significant increase in the variation of the distance measure. These are the variable pairs *length-diameter*, *length-whole.wt*, *length-viscera.wt* and *whole.wt-viscera.wt*. In fact the free method even has less variation in the Kendall's tau distance measure for the variable pair *length-viscera.wt* than the parameter method and family method have. Of course this can also be explained by a higher flexibility in the free method, which had to estimate the R-vine structure from the complete part of the data. The underlying R-vine structure of RV was recognized in 0 cases when fitting an R-vine to the complete part of the data. At least the first tree coincided with the one from RV in 49 of the 100 cases.
In comparison the two benchmark methods, namely norm and fcs, have a constantly small

variation on all variable pairs, but a bit higher than the oracle method. While the median of the fcs method is fairly close to the null, the norm method systematically underestimates the Kendall's tau coefficients, with the median of the distance measures being always below the null. However also the median of the fcs method is below the null for example for the variable pair *length-whole.wt*, for which all vine copula methods perform better.

The case with at most 5 variables missing per incomplete observation yields very similar results. We will not discuss everything again and will focus on changes. The variation for the family and parameter method increases, as could be expected by introducing higher missingness. A more unusual change is that the variation of the free method decreases. A possible explanation might be, that in every possible R-vine structure for our data set, the missing data patterns with exactly 5 variables missing are always ad hoc imputable, as we already wrote down for the general case in Table 4.4. Another obvious change is the performance of the benchmark. The fcs method now also struggles to hit the null with its median and does not outperform the norm method as before. For many variable pairs, as for example *length-shell.wt*, the median of the Kendall's tau distance measures of the norm method is closer to the null than the fcs method. Still the variation of both benchmark methods is constant over all variable pairs, but also increased compared to before.

Conclusive, we can say that in most cases the median of our vine imputation method was at the null as it was desired. However it was not the case for every variable pair. In the results we saw high variation for the free method, but more tests are necessary here as each incomplete data set was imputed only once. As long as the structure of the underlying model was known it had no major impact when copula parameters or families had to be estimated on the subset of complete observations first. Estimating the structure in contrast, brings in higher variation and uncertainty. This was not unexpected, because if the underlying structure is not identified, there are non ad hoc imputable missing data patterns occurring.

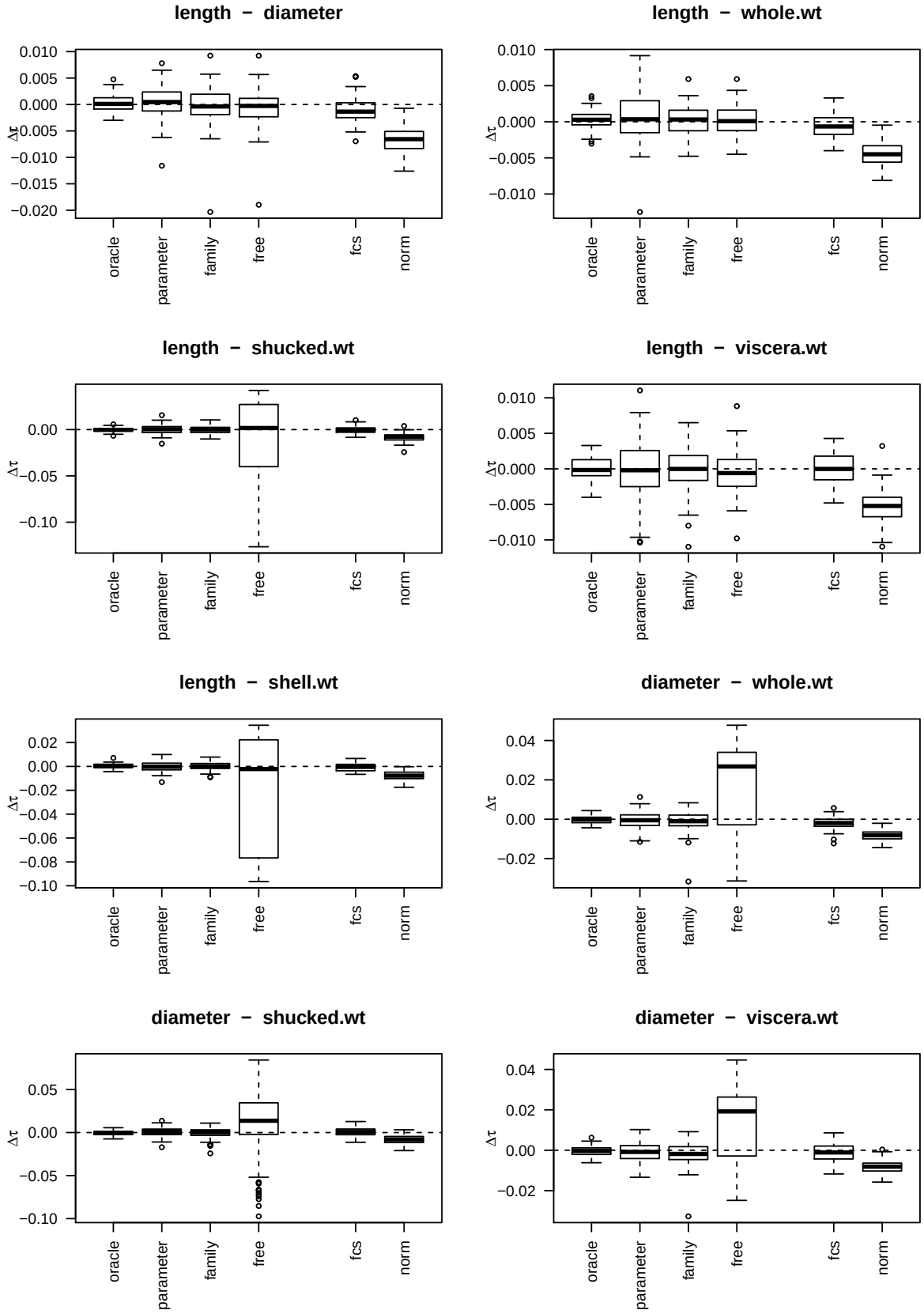


Figure 5.6: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods oracle, parameter, family, free, fcs and norm in the case with at most 3 variables missing.

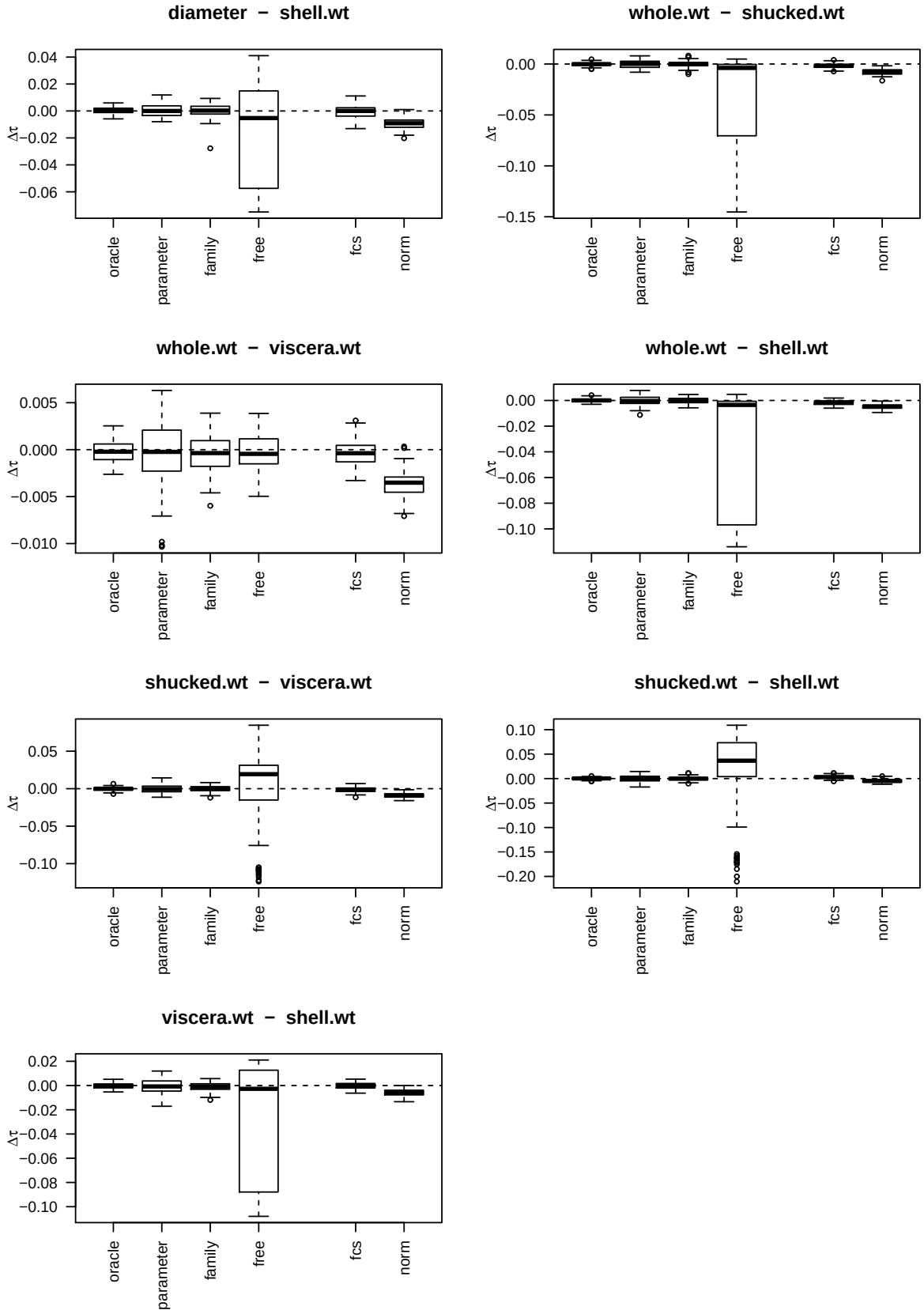


Figure 5.7: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods oracle, parameter, family, free, fcs and norm in the case with at most 3 variables missing (continued).

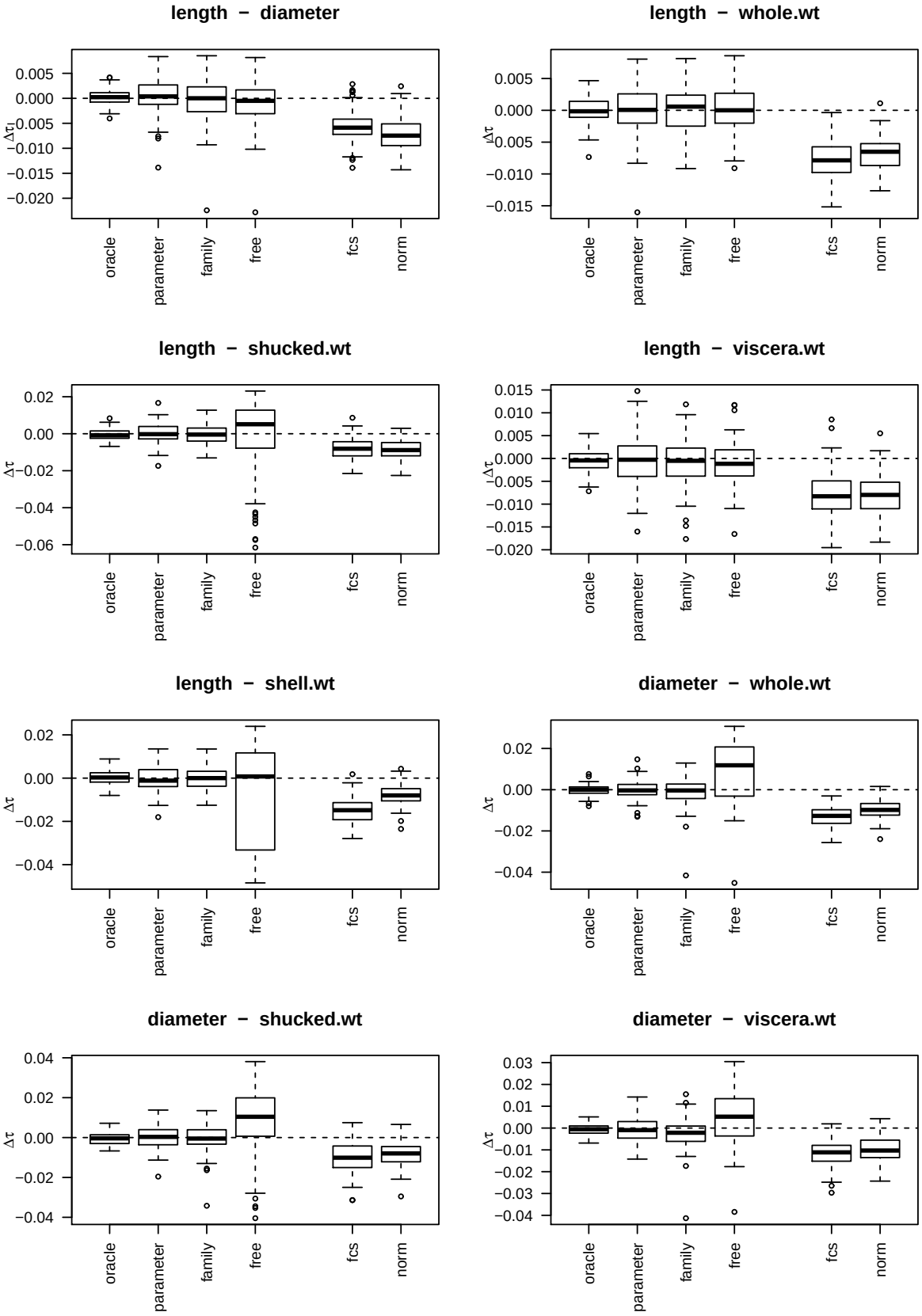


Figure 5.8: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods oracle, parameter, family, free, fcs and norm in the case with at most 5 variables missing.

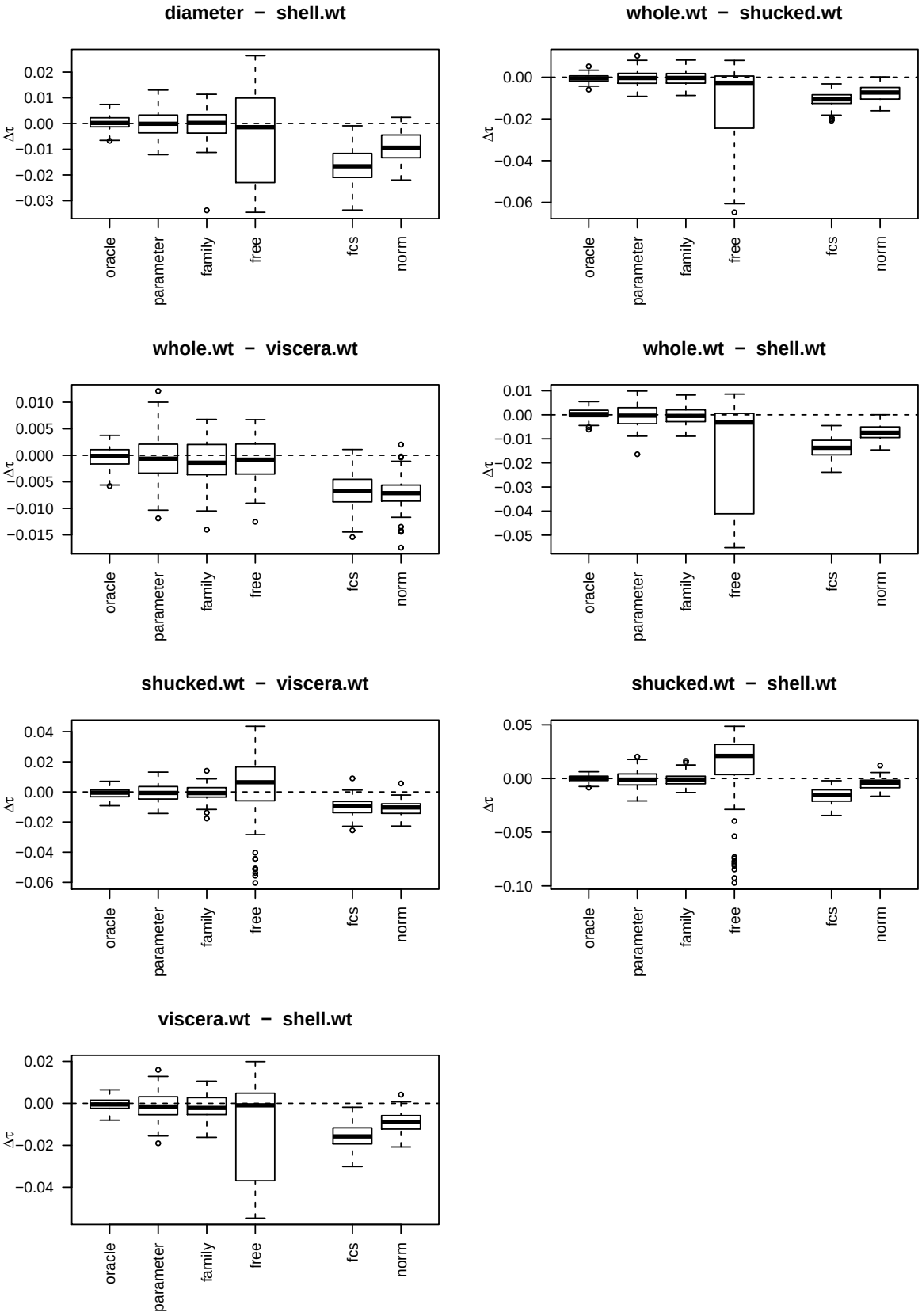


Figure 5.9: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods oracle, parameter, family, free, fcs and norm in the case with at most 5 variables missing (continued).

5.6.2 Benchmarking with knn and CoImp

Scenario

As we saw in the checks from the last section, the vine copula imputation algorithm seems generally to be working for a special constructed case. Still, there was high variation in the results. However, we might get rid of this variation using multiple imputation.
In the general test data setup notation from Section 5.5.1, we choose $K = 100$ and $L = 10$. This means we simulate 100 data sets and impute each 10 times by every imputation method. The number of observations in the simulated data sets $D_k,\ k = 1, \dots, K$ is chosen to be $1,000$.
Then the missingness is introduced using Algorithm 5.4, with perc.miss $= 0.1$ and max.var.miss $= 5$, this time with no further restritions on the occurring missing data patterns.
The imputation methods of choice for comparison in this scenario are besides our own vine copula imputation method, the CoImp method and the k-nearest neighbors method. In fact knn is deterministic and will be run only once per incomplete data set and not 10 times. Instead, for the knn we will run 3 different specifications for k, the number of neighbors used, namely $k = 5$, $k = 15$ and $k = 30$. These two methods, which we will compare with our own one, are not state-of-the-art and we expect them to be potentially worse or at least show some weaknesses.
For the evaluation only Kendall's tau values are calculated as we do not need any further measure if a method is clearly outperformed.

Results

The resulting box plots of the Kendall's tau distances $\Delta_k^{I_p}$, for $k = 1, \dots, K$ and $I_p \in \{\text{vine, coimp, knn5, knn15, knn30}\}$ are shown in Figure 5.10 and Figure 5.11.

Better than in the previous checks, the median of the vine method is close to the null for all variable pairs. Nevertheless it is clearly off the null for the variable pairs *length-shell.wt*, *diameter-shell.wt*, *whole.wt-shell.wt*, *shucked.wt-shell.wt* and *viscera.wt-shell.wt*. Strikingly these are all variable pairs of *shell.wt*. A possible reason might be that this variable is in the tree structure of RV, which is also illustrated in Figure 5.4, in the trees 2 to 4 in the conditioned set of a node which is not a leaf. The restriction to build an R-vine where the variable is only in the conditioned set of a leaf in every tree might be too restrictive. Obviously, the vine imputation method has no longer a lot higher variation for some of the variable pairs. For all variable pairs the box plots of the vine method show symmetric variation of about 0.02 around the median. The coimp method in contrast is always systematically underestimating Kendall's tau by around 0.05. The knn methods perform very similar for the different specifications for the number of nearest neighbors and also underestimate the Kendall's tau, however at a lower level of about 0.01. In most cases it seems that among the knn methods, the method with 30 neighbors yields the best results, but there is only a slight difference.

As the coimp method and the knn method systematically underestimated the Kendall's

tau coefficients they are no good choice for an imputation method. Our own vine imputation method worked better now we used 10 imputations instead of one as in the previous checks and we got rid of the very high variation in the results. Unfortunately our method did not yield unbiased estimates for for all variable pairs. Especially for one variable, all pairwise Kendall's tau estimates seemed to be biased and the restrictions in the R-vine fitting might in general be too restrictive. Nonetheless our vine imputation method outperformed the two other methods and next we will extensively test our method against the two state-of-the-art multiple imputation methods.

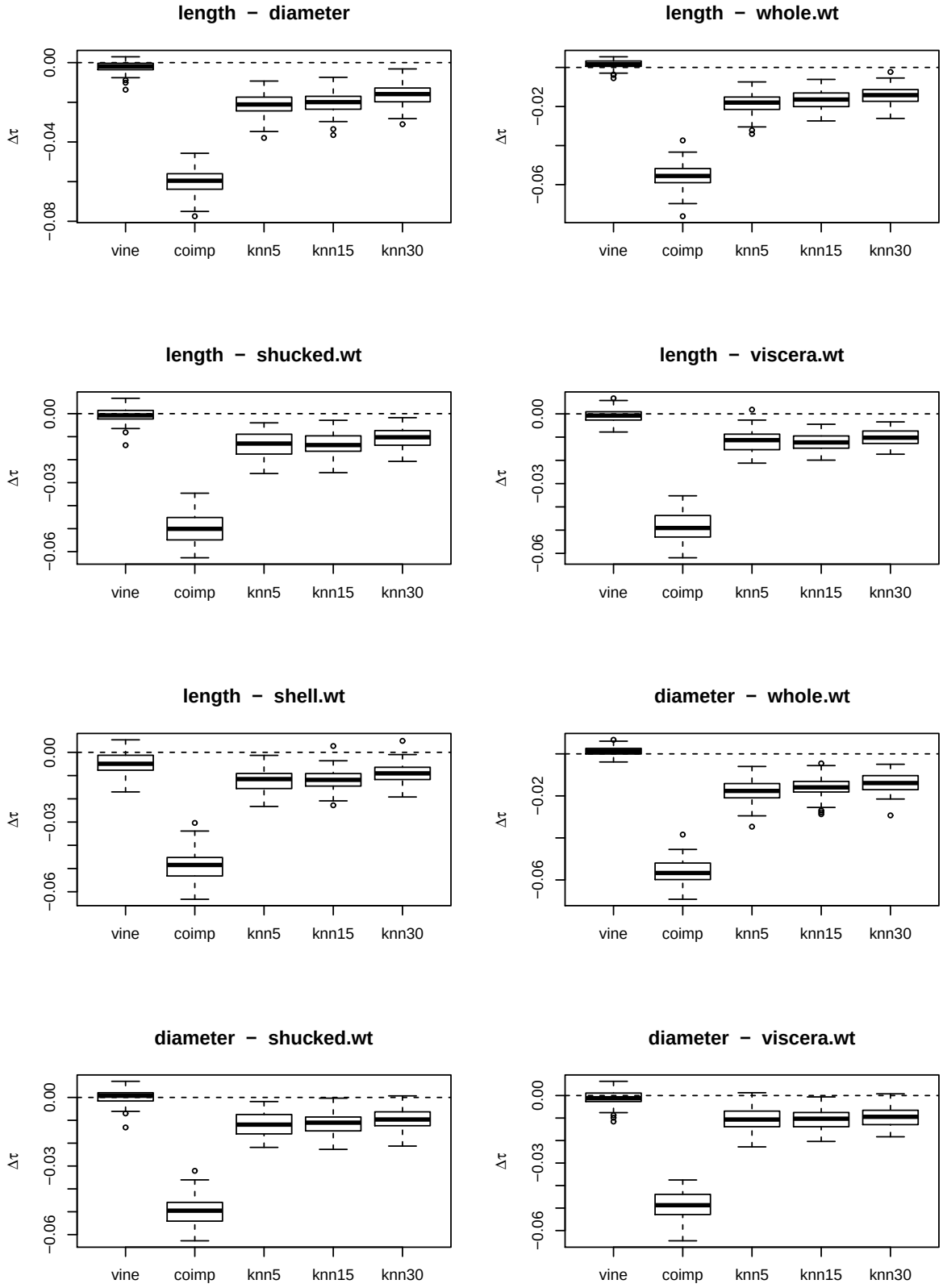


Figure 5.10: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, coimp, knn5, knn15 and knn30 in a scenario with 10% incomplete observations.

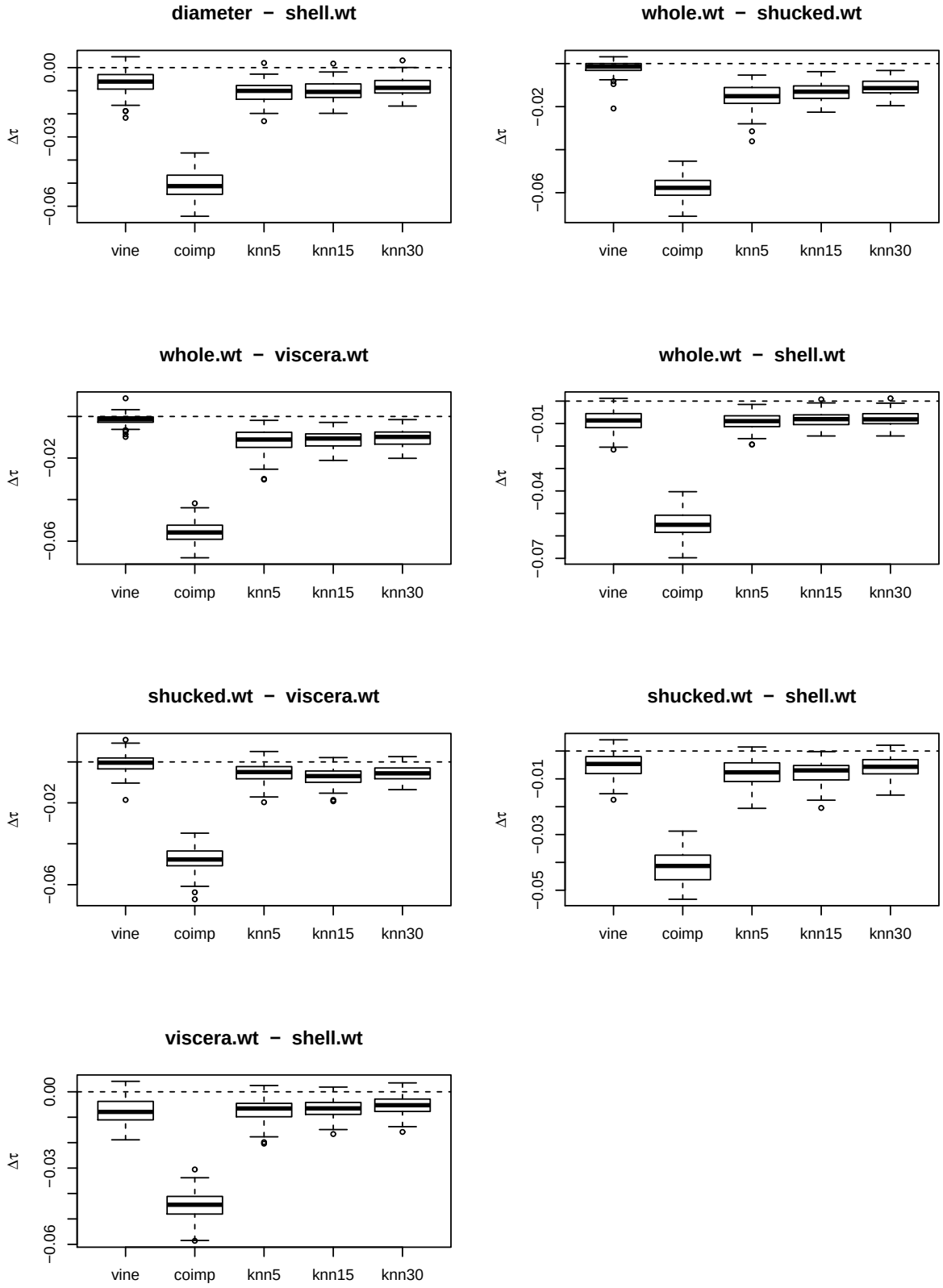


Figure 5.11: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, coimp, knn5, knn15 and knn30 in a scenario with 10% incomplete observations (continued).

5.6.3 Benchmarking with multivariate normal JM and FCS

Scenario

We saw that our own vine copula imputation method performed better than the CoImp and knn imputation methods identifying and reproducing the dependence structure of the simulated data sets in the last section. However now we want to test our method in detail against the two state-of-the-art multiple imputation methods, namely the multivariate normal Joint Modeling and the Fully Conditional Specification, denoted by *norm method* and *fcs method* respectively.
In the general test data setup notation from Section 5.5.1, we choose $K = 100$ and $L = 100$. This means we simulate 100 data sets and impute each 100 times by every imputation method. The high number of imputations ensures that the variation around the pooled estimate in the evaluation is small. The number of observations in the simulated data sets $D_k,\ k = 1, \ldots, K$ is chosen to be $1,000$.
Then the missingness is introduced using Algorithm 5.4. We will check two cases. The first one with a relatively low percentage of incomplete observations with *perc.miss*= 0.1 and the second one with a high percentage of incomplete observations with *perc.miss*= 0.5 . In both cases we set *max.var.miss*= 5.
We will evaluate pairwise Kendall's tau, pairwise alternative upper/lower tail dependence and the log-likelihood for the multivariate performance.

Results

This time we will evaluate the distance measure $\Delta_k^{I_p}$, for $k = 1, \ldots, K$ and $I_p \in$ {vine, fcs, norm} not only for Kendall's tau, but also for the alternative upper/lower tail dependence and the log-likelihood. The resulting box plots of the Kendall's tau distances $\Delta_k^{I_p}$ are shown in Figure 5.12 and Figure 5.13 for the case with 10% incomplete observations, and in Figure 5.14 and Figure 5.15 for the case with 50% incomplete observations.

In the case with 10% incomplete observations, our vine method yields similar results as in the last scenario. We can again see biased estimates for all variable pairs including the *shell.wt* variable. However the increased number of imputations resulted in a overall lower variation. The maximum outlier is about -0.02 for the variable pair *viscera.wt-shell.wt*. The highest bias in the median is about -0.01 for the same variable pair. In general the bias is at a very small level over all variable pairs. Comparing the vine method to the two other methods, we see that the vine method has a higher variation, but for all methods the variation is very symmetric around the median. The fcs method is clearly performing best. The median of the norm method is slightly but systematically under the null, while the vine also overestimates the dependence for the variable pairs *length-whole.wt* and *diameter-whole.wt*.
In the case with 50% incomplete observations we have very similar results. The variation increases for all methods with the higher missing percentage, but the vine method still has the highest variation. The norm method still systematically underestimates the Kendall's tau coefficients, while the fcs method works best.

Although we saw that the vine imputation method had difficulties for some variable pairs, overall the bias was on a decent level. Nevertheless we have to admit that especially the fcs method worked better and saw how good it captured the pairwise Kendall's tau dependence. Next we check if the benchmark methods are performing just as well when we analyze the tail dependence.

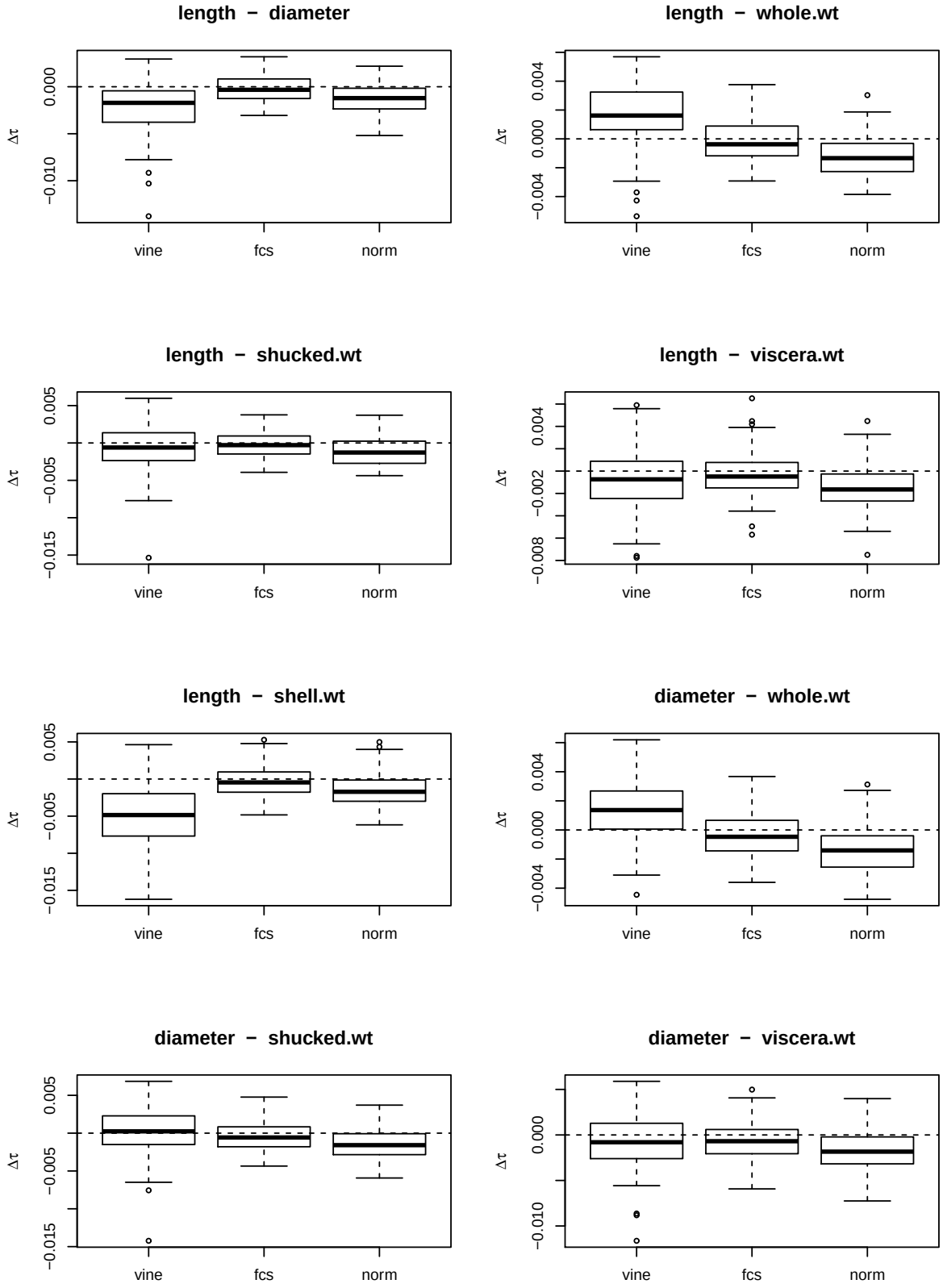


Figure 5.12: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations.

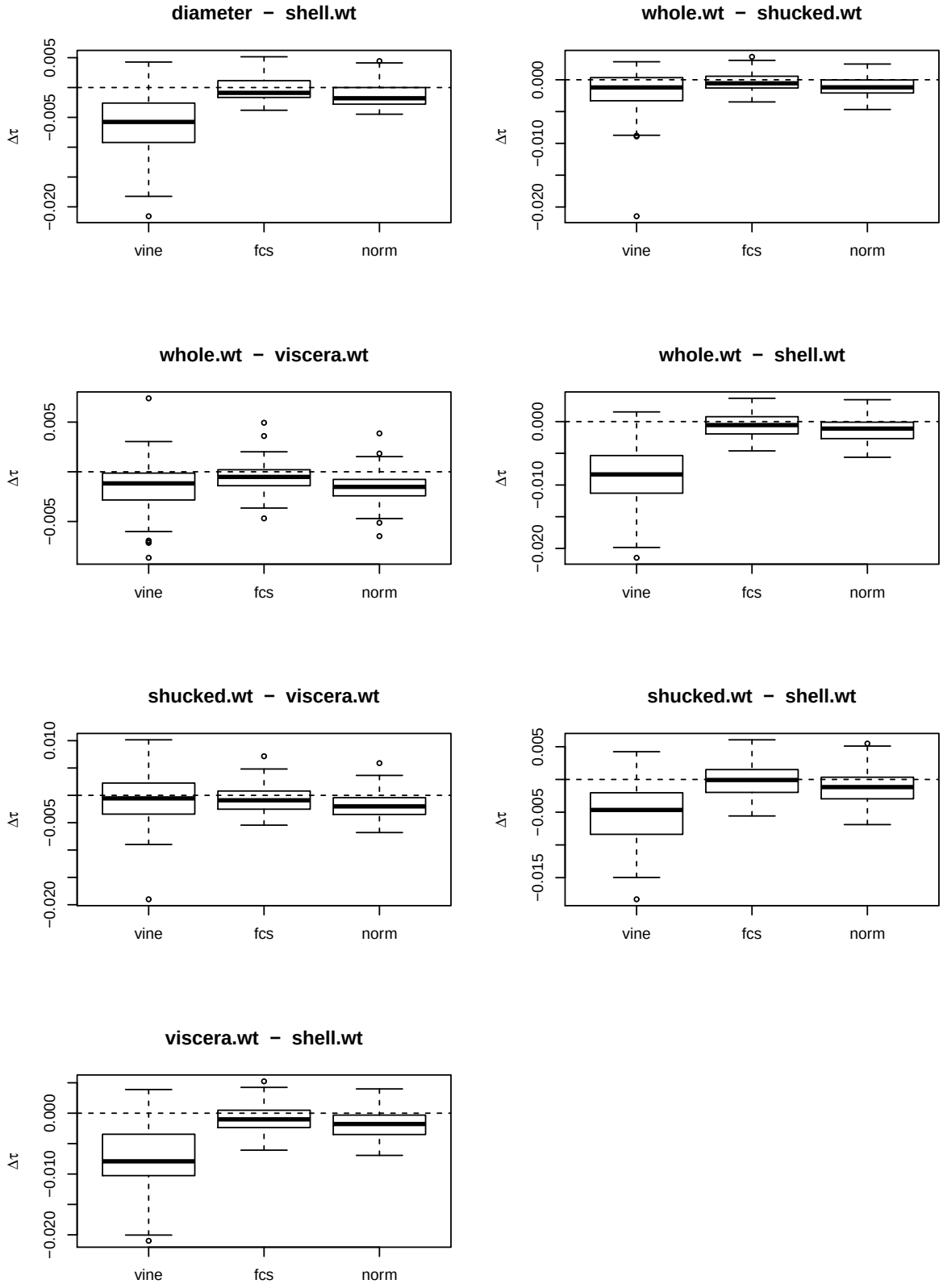


Figure 5.13: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations (continued).

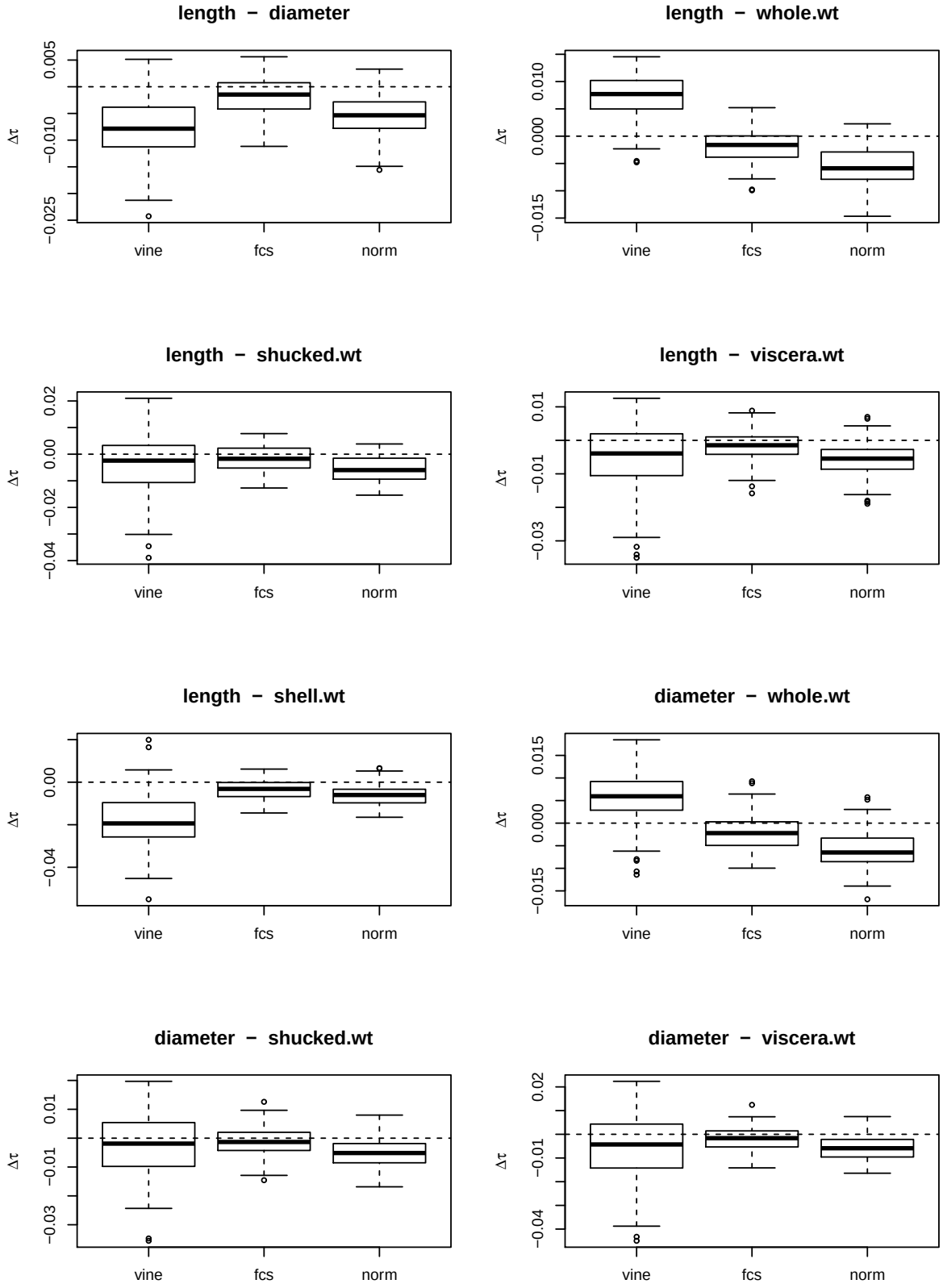


Figure 5.14: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 50% incomplete observations.

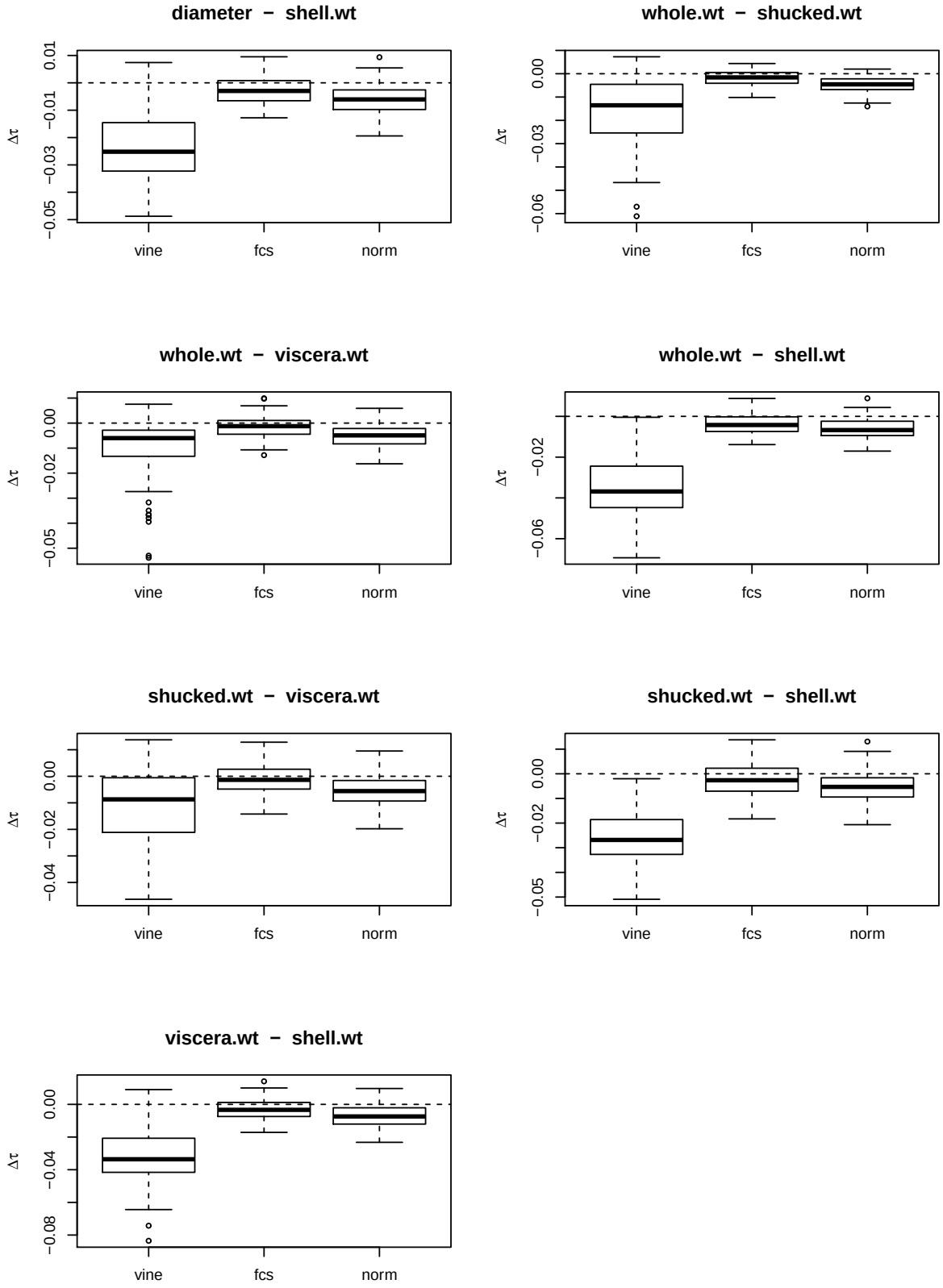


Figure 5.15: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 50% incomplete observations (continued).

Next we will look at the distance measures $\Delta_k^{I_p}$ for the alternative lower and upper tail dependence. The resulting box plots for the case with 10% incomplete observations for the lower tail dependence are shown in Figure 5.16 and Figure 5.17, while the box plots for the upper tail dependence are shown in Figure 5.18 and Figure 5.19. The respective box plots for the case with 50% incomplete observations are shown in the Figures 5.20-5.23.

Let's start again with the discussion of the case with 10% incomplete observations. For the lower tail dependence and most variable pairs, as for example *length-shucked.wt*, *diameter-viscera.wt* or *whole.wt-shucked.wt* the median of the vine method is closest to the null compared to the two other methods. The two exceptions, where the fcs method performs better, are the variable pairs *diameter-shell.wt* and *whole.wt-shell.wt*. Still, the vine method has higher variation and more outliers than the fcs method. The outliers are off the null up to 0.07 for the variable pair *shucked.wt-shell.wt*, but the median of the distance measures is never exceeding a deviation to null of over 0.02. The mass of the distance measures for the fcs method and the norm method is for all variable pairs below the null. While the deviation is in some pairs very small for the fcs method, it is significant for the norm method for all pairs. In contrast the vine method underestimates the lower tail dependence only for the 5 variable pairs which include the *shell.wt* variable.
Looking at the upper tail dependence distance measure, we cannot see a clear winner as for the lower tail dependence. All box plots are on the same level. The vine method has once again the highest variation among the compared methods, but the differences are small. We can also see again that the median of the fcs method and norm method are below the null for all variable pairs, whereas the median of the distances from the norm method is always the lowest. In contrast the median for the distance measure of the upper tail dependence in the vine method also can be above null as for the variable pair *length-whole.wt*.
Let's continue with the discussion of the lower and upper tail dependence in the case with 50% incomplete observations. At the first glance at the lower tail dependence coefficients we see that the norm method now lags far behind the two other methods. The median of the norm method's distance measures is even at -0.2 for the variable pair *shucked.wt-shell.wt*, which is really high for a possible value range of $[-1, 1]$ in the tail dependence coefficients. Overall all imputation methods now have higher variation attributable to the higher missingness. The vine method still has the highest variation with single outliers that are really off the median. For most variable pairs the median of the vine method is still the closest to null, the two clear exceptions are the variable pairs *diameter-shell.wt* and *whole.wt-shell.wt*.
For the lower tail dependence we have similar changes. Compared to the case with 10% incomplete observations, again all methods gain variation and the norm method loses most, systematically underestimating the upper tail dependence. Also the median of the fcs method is below null for all variable pairs. The median of the vine method is always closest to the null, besides the exceptional variable pair *whole.wt-shell.wt*.

Conclusive we can say that our vine imputation performed clearly better than the two other methods capturing the pairwise alternative lower and upper tail dependence coefficients. Especially for the really high lower tail dependence values the multivariate normal

model seemed to be really off the real values, but also the fcs method which was the best for Kendall's tau showed weaknesses and systematically underestimated the tail dependence. We saw once again that higher missingness, causes higher variation. However the vine method still had the highest variation compared to the benchmarks. While for the Kendall's tau we saw that the vine method failed for all variable pairs including the *shell.wt* variable, we cannot see this effect at the same level for the tail dependence. After we saw the different performances of the methods capturing the pairwise Kendall's tau coefficients and the pairwise alternative lower and upper tail dependence coefficients, we will also evaluate the log-likelihood for the imputed data sets to analyze the multivariate performance.

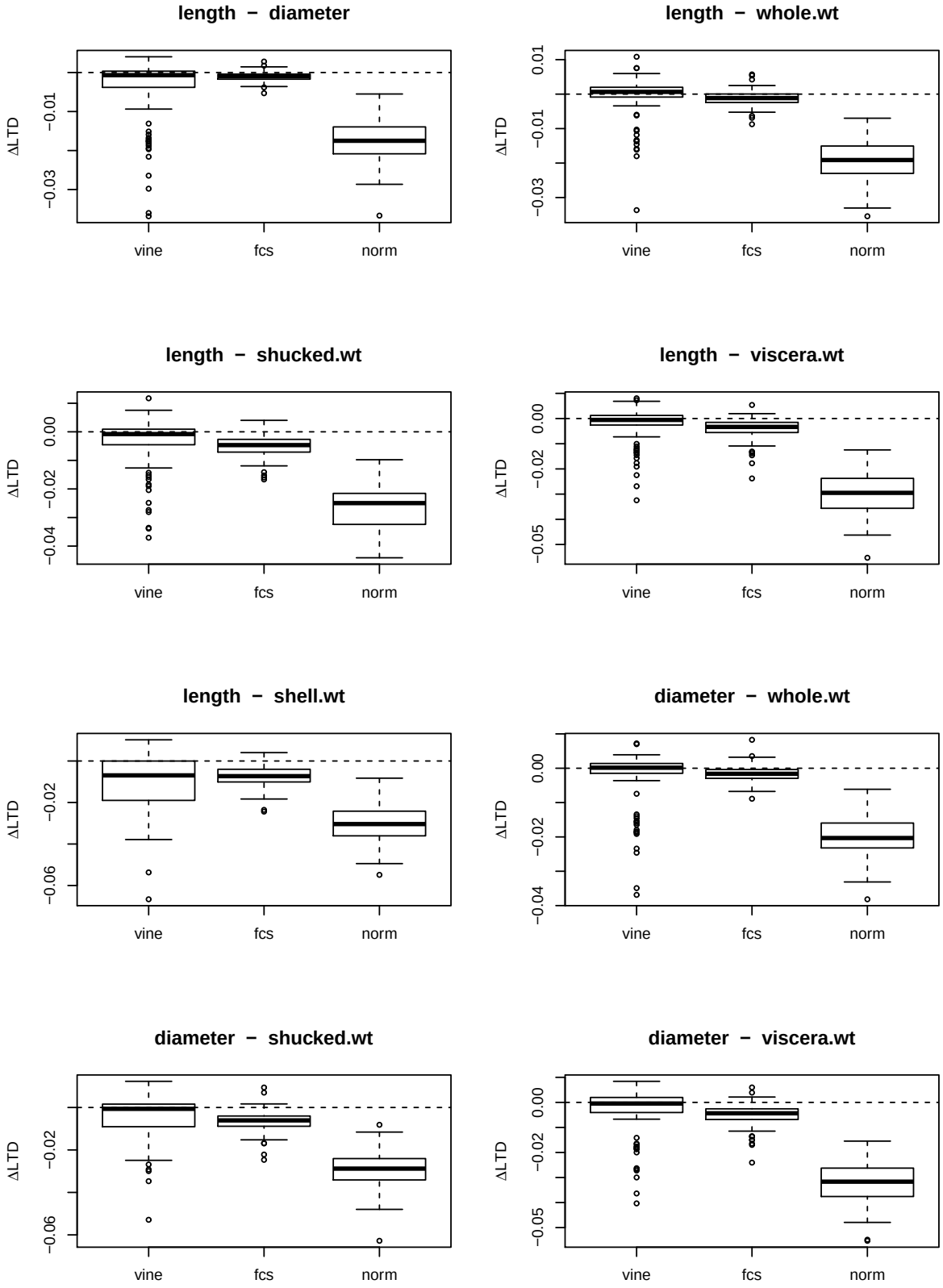


Figure 5.16: Box plots of the lower tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations.

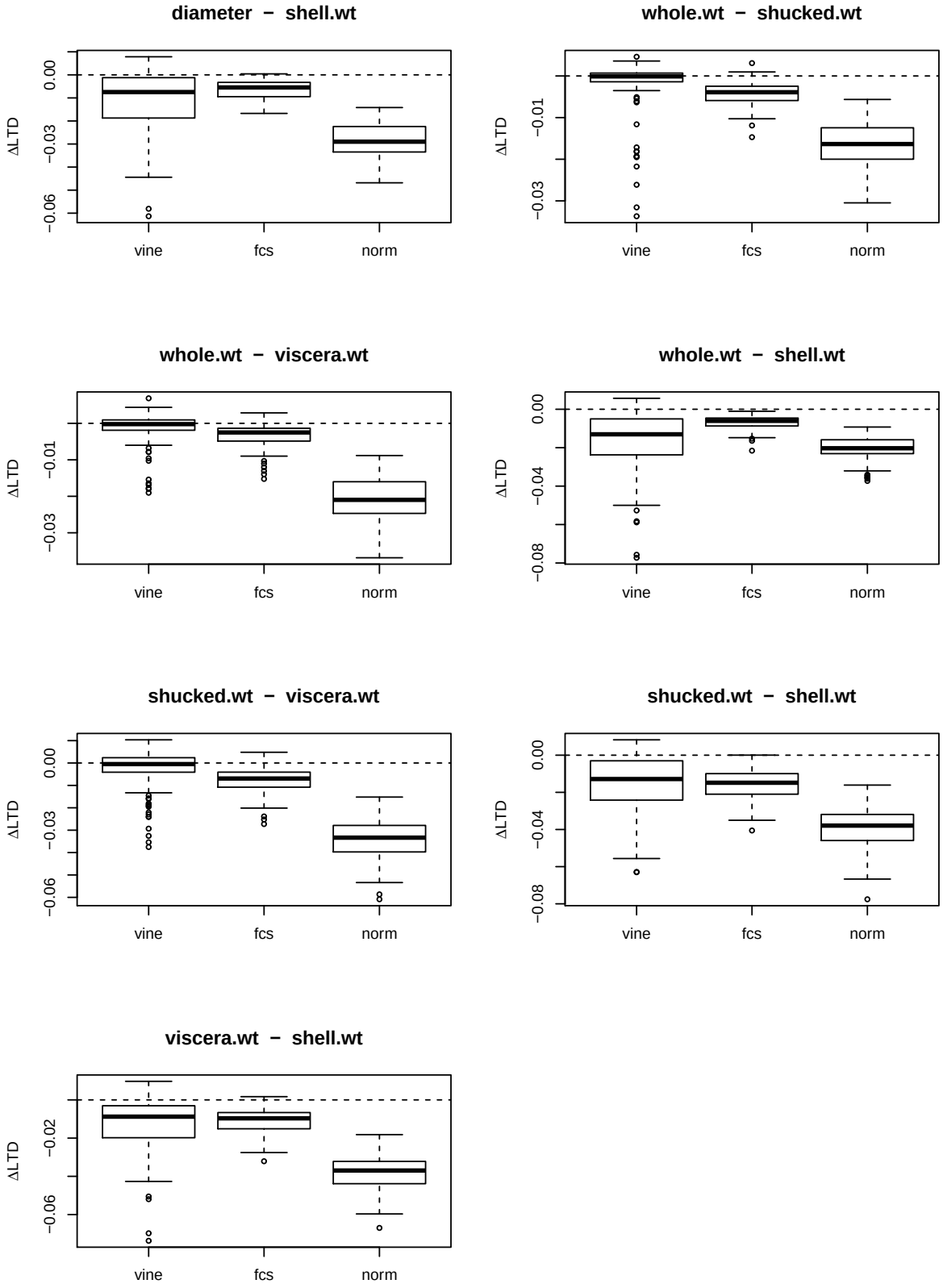


Figure 5.17: Box plots of the lower tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations (continued).

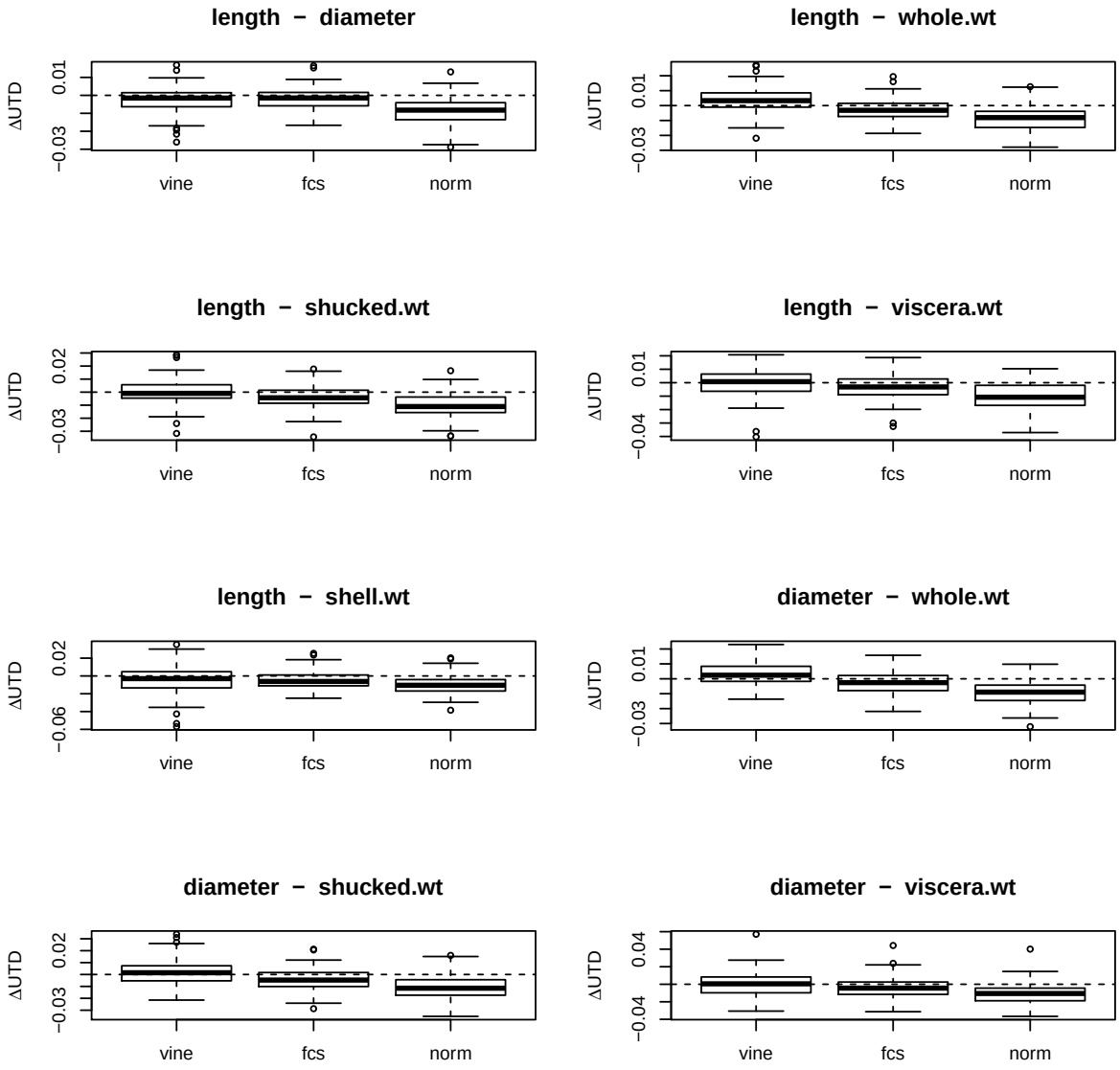


Figure 5.18: Box plots of the upper tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations.

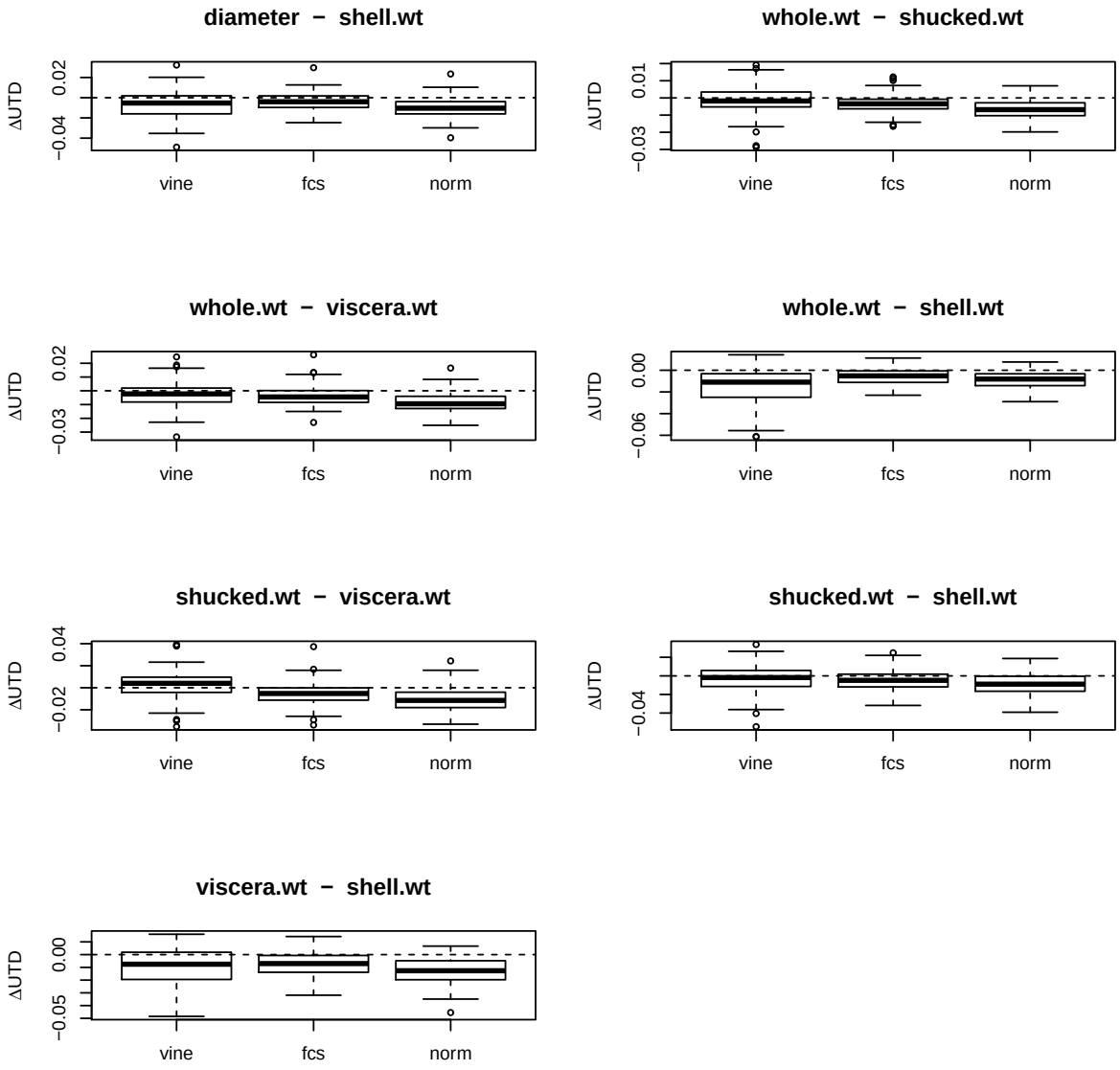


Figure 5.19: Box plots of the upper tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations (continued).

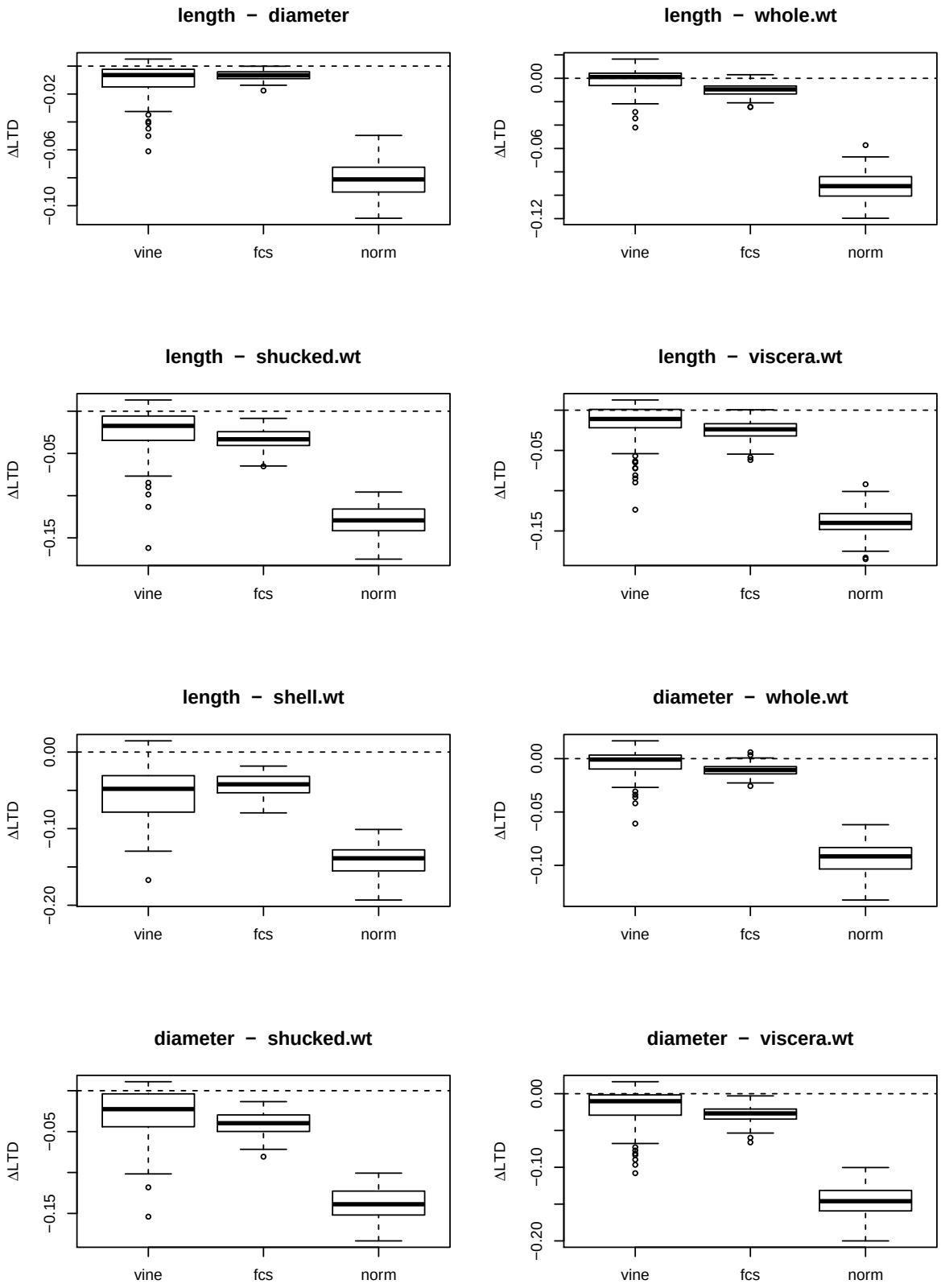


Figure 5.20: Box plots of the lower tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 50% incomplete observations.

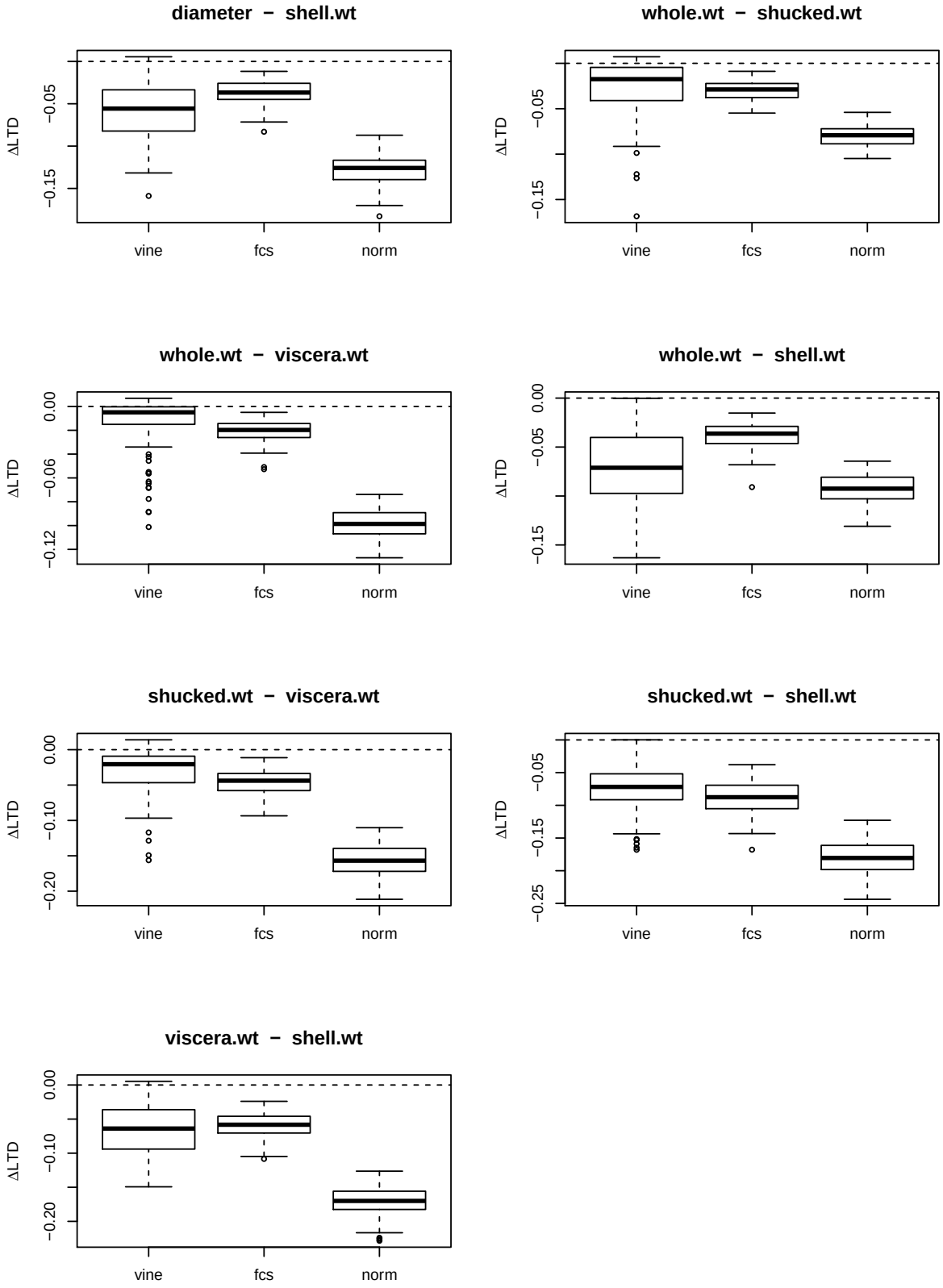


Figure 5.21: Box plots of the lower tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 50% incomplete observations (continued).

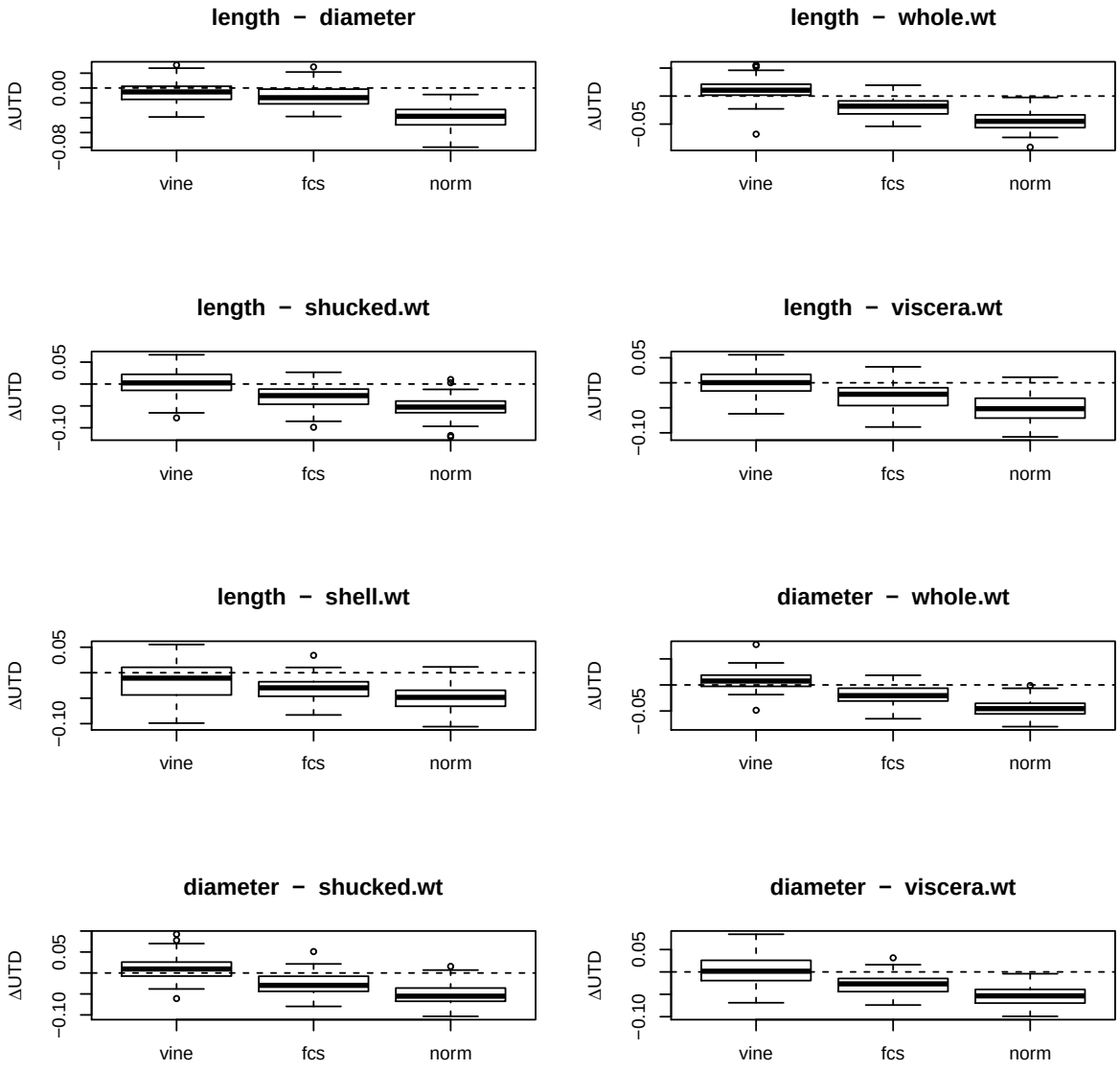


Figure 5.22: Box plots of the upper tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 50% incomplete observations.

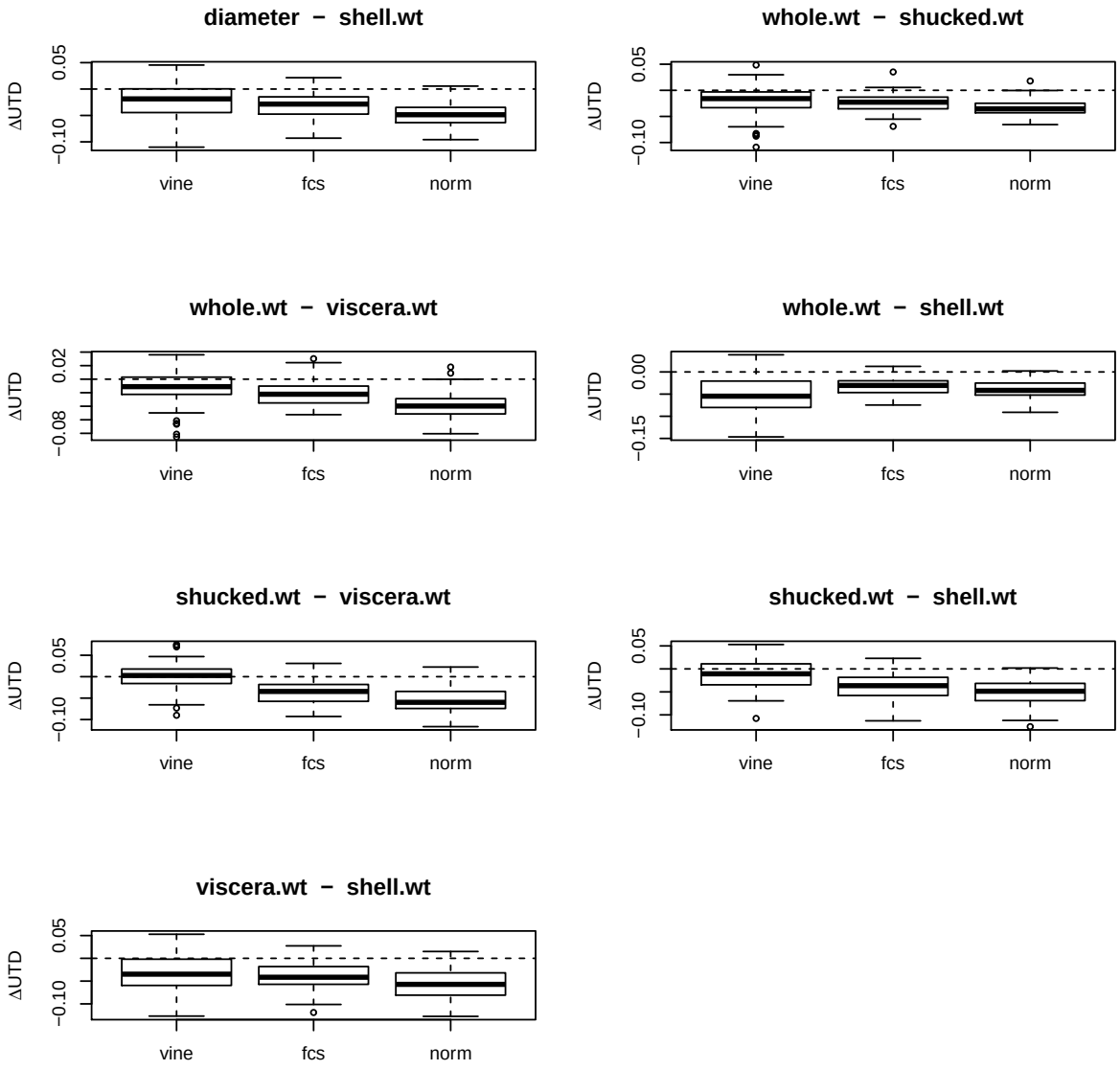
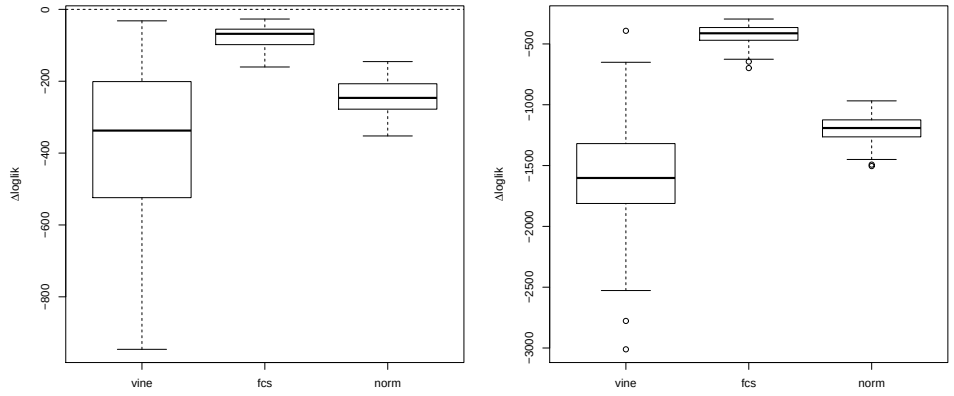


Figure 5.23: Box plots of the upper tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 50% incomplete observations (continued).

As we saw the fcs method yielded the best results for Kendall's tau, but the vine method was better for the tail dependence. So we will now evaluate the log-likelihood for the imputed data sets to also factor in the multivariate performance. We will once more have a look at the box plots of the distance measures $\Delta_k^{I_p}$, this time for the log-likelihood statistic. In contrast to the pairwise dependence coefficients analysis, we only yield one statistic per imputed data set. The box plots for the two cases with 10% and 50% incomplete observations are shown in Figure 5.24.
Here we can clearly see that the fcs method performs best and the vine method, not only has the highest variation, but also is off the null the most. The restrictions in the R-vine fitting seem to influence the overall fit more than fitting a multivariate normal model to the data.



(a) Scenario with 10% incomplete observations

(b) Scenario with 50% incomplete observations

Figure 5.24: Box plots of the log-likelihood distance measures for the imputation methods vine, fcs and norm.

In Appendix C we show again results for the same scenario with 10% incomplete observations with the only difference, that all simulated data sets $D_k,\ k = 1, \ldots, K$ have only 500 observations instead of $1,000$. To summarize the results briefly, we can see that a smaller sample size increases the variation for all imputation methods. Overall the results were very similar to the ones with the higher sample size and no method struggled significantly more with the smaller sample size than the others.

Chapter 6

Conclusion

In this master thesis we first gave an introduction to missing data theory, imputation methods and copulas with a focus on pair copula constructions and R-vines in the first three chapters. Missing data is a very common occurrence in real world data, nonetheless almost all analytical methods and software packages were designed for complete data. Copulas have the advantage that we can separate the modeling of the marginal distributions from the modeling of the dependence structure. Additionally, R-vine copulas have the advantage that they consist only out of bivariate building blocks and come along with a high flexibility in the model building. This makes copulas interesting for imputation as it is crucial to correctly identify and capture the dependence structures of a data set in order to impute values adequately. In Chapter 4 we first derived a theory to find out which conditional distributions of an R-vine can be derived analytically in a closed form using only available bivariate copula. As we are able to simulate from the derived conditional distributions via the inverse transform sampling, each conditional distribution gives us a solution to impute one missing data pattern. We called these missing data patterns ad hoc imputable. Unfortunately, not all conditional distributions can be derived in this way and the proportion of ad hoc imputable missing data patterns is decreasing with the dimension of the data, as described in Table 4.3. So we extended our theory to be able to have a solution for the other missing data patterns. The idea was to build an R-vine model with restrictions on the R-vine structure selection, so that a missing data pattern of choice is ad hoc imputable at the end. By iterating and repeating this step all missing data patterns can be imputed and we derived the final general vine copula imputation method Algorithm 4.19. Finally, in Chapter 5 we tested our own imputation method in several scenarios against other methods, evaluating how well they capture the dependence structure of the data. It seems that as long as the structure of the underlying model is identified correctly, our developed vine copula imputation method seems to work well. The simulation works especially well for all ad hoc imputable missing data patterns. For the other missing data patterns, the variation increases. This is a direct consequence of the restrictive R-vine model selection. Nevertheless, the CoImp and the k-nearest neighbor imputation method were outperformed, as they both systematically underestimated the pairwise Kendall's tau dependence coefficients. Last, we compared our vine copula imputation method to the two state-of-the-art imputation methods, namely the multivariate normal Joint Modeling (JM) and the Fully Conditional Specification (FCS). While the JM

and FCS imputation methods did a better job capturing Kendall's tau coefficients, our own method performed significantly better in capturing the tail dependence coefficients. Especially in the case of very high tail dependence, the multivariate normal model was clearly outperformed, but also the FCS systematically underestimated the tail dependence coefficients. Still our method had higher variation in the results than the two state-of-the-art multiple imputation methods. We concluded the benchmark with the evaluation of the log-likelihood of the imputed data and saw that our own method did worse than the two others in reproducing the multivariate structure and still needs some improvement. The JM as well as the FCS showed why they are the state-of-the-art multiple imputation methods, but we saw that they are not working best in capturing the tail dependence and copulas might be the tool to get in there.
Possible further research could be the use of semi-parametric copulas for a higher flexibility. Also the order in which the variables are imputed, when there are several possibilities, might have an impact on the results. The restrictive R-vine model building seems to seriously distort the overall performance for the imputation of specific variables, which seems to be depending on the R-vine structure of the fit. It would be good to detect potentially problematic variables early in the tree structure and see if they can be treated with another approach.

Appendix A

Bivariate copula family distributions

In the following the distribution functions of copula families mentioned in the thesis are presented and an explanation about rotated copulas is given.

A.1 Independence copula

$$C_{12}(u, v) = uv$$

A.2 Gaussian copula

aabdfd

$$C_{12}(u, v) = \Phi(\Phi^{-1}(u), \Phi^{-1}(v); \theta),$$

for $\theta \in [-1, 1]$, where $\Phi(\cdot)$ is the standard normal distribution, and $\Phi(\cdot, \cdot; \theta)$ is the bivariate normal distribution with zero mean vector and correlation θ.

A.3 Student t copula

$$C_{12}(u, v) = \int_{-\infty}^{t_\nu^{-1}(u)} \int_{-\infty}^{t_\nu^{-1}(v)} \frac{\Gamma(\frac{\nu+2}{2})}{\Gamma(\frac{\nu}{2})\sqrt{(\pi\nu)^2(1-\theta^2)}} \left(1 + \frac{x^2 - 2\theta xy + y^2}{\nu(1-\theta^2)}\right)^{-\frac{\nu+2}{2}} dx\, dy,$$

for $\theta \in [-1, 1], \nu \in \mathbb{N}$, where t_ν^{-1} is the quantile function of the Student t distribution with ν degrees of freedom and $\Gamma(\cdot)$ is the gamma function.

A.4 Clayton copula

$$C_{12}(u, v) = (u^{-\theta} + v^{-\theta} - 1)^{-\frac{1}{\theta}},$$

for $0 < \theta < \infty$.

A.5 Gumbel copula

$$C_{12}(u,v) = exp\left\{-\left((-\ln u)^{\theta} + (-\ln v)^{\theta}\right)^{\frac{1}{\theta}}\right\},$$

for $\theta \geq 1$.

A.6 Frank copula

$$C_{12}(u,v) = -\frac{1}{\theta}\ln\left(\frac{1}{1-e^{-\theta}}((1-e^{-\theta}) - (1-e^{-\theta u})(1-e^{-\theta v}))\right),$$

for $\theta \in [-\infty, \infty]\backslash 0$.

A.7 Remark to rotated copulas

To obtain the rotated bivariate versions of the Clayton copula or Gumbel copula we use counterclockwise rotations of the copula density.

90° $c_{12}^{90}(u,v) := c(1-u,v)$

180° $c_{12}^{180}(u,v) := c(1-u,1-v)$

270° $c_{12}^{270}(u,v) := c(u,1-v)$

The motivation for this is to extend the range of dependence.

Appendix B

Presentation of the VineCopulaImpute R-package

VineCopulaImpute functionalities

This document is for illustration of the *VineCopulaImpute* R-package which was developed in the course of the Master's thesis about "Identification of Directly Imputable Missing Data Patterns Using R-vine Copulas and Application to Multiple Imputation". References refer to this thesis. All functions included in this package will be presented here.

At first we load the necessary packages and set a seed in order to yield reproducible results.

```
library(plyr)
library(ghyp)
library(VineCopula)
library(VineCopulaPlus)
library(VineCopulaImpute)
set.seed(101)
```

The first small tool of the *VineCopulaImpute* package is the *makemdp()* function. Often missing data patterns are given as $[0,1]^d$ vector, where a 0 indicates missing data. The *makemdp()* function just returns the positions, where a vector has value 0.

```
makemdp(c(1,0,1,0,0))
```

```
## [1] 2 4 5
```

For the presentation of other functions, we will need a dataset. We will use a subset of the abalone data from https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data. Our subset contains 1000 observations of female abalone only. As the variables *height* and *rings* from the original ablone data are not really continuous or discrete, they were dropped.

```
data(abalone1000f)
```

At first we bring the data to the u-scale by fitting an univariate Generalized Hyperbolic distribution to each variable, followed by the probability integral transform with the *uvunifhyperb()* function.

```
abalone.uni<-uvunifhyperb(abalone1000f)
```

Now we can fit an R-vine to the u-data by the *RVineStructureSelect()* function from the *VineCopula* package. The copula families we allow are the independence copula, the Gaussian copula, the Student t copula, the Clayton copula with rotations, the Gumbel copula with rotations as well as the Frank copula.

```
aba.Rvine<-RVineStructureSelect(abalone.uni,familyset=c(0,1,2,3,4,5,13,14,23,24,33,34))
```

So far the R-vine matrix is not of diagonal form as defined in Lemma 4.1.

```
aba.Rvine$Matrix
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    0
## [2,]    5    2    0    0    0    0
## [3,]    4    5    4    0    0    0
## [4,]    6    4    5    5    0    0
## [5,]    3    6    6    6    3    0
## [6,]    2    3    3    3    6    6
```

The function *diagforce()* from the *VineCopulaImpute* package brings the R-vine matrix to diagonal form either with unchanged first column as in

```
aba.Rvine.diag.a<-diagforce(aba.Rvine)
aba.Rvine.diag.a$Matrix
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    0
## [2,]    5    5    0    0    0    0
## [3,]    4    2    2    0    0    0
## [4,]    6    4    4    4    0    0
## [5,]    3    6    6    6    6    0
## [6,]    2    3    3    3    3    3
```

or with changed first column if the argument *alternative* is used

```
aba.Rvine.diag.b<-diagforce(aba.Rvine,alternative=TRUE)
aba.Rvine.diag.b$Matrix
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    0    0    0    0    0
## [2,]    1    1    0    0    0    0
## [3,]    2    4    4    0    0    0
## [4,]    4    6    2    2    0    0
## [5,]    6    3    6    6    6    0
## [6,]    3    2    3    3    3    3
```

The output in both cases is an RVM object. Not only the R-vine structure matrix, also the family and parameter matrices are changed.

For a R-vine matrix of diagonal form, the *RVineAdhocMdp()* function finds all missing data patterns using Algorithm 4.15, for which a closed form of the conditional distribution function of the missing variables, given the observed ones, in general can be derived analytically.

```
RVineAdhocMdp((aba.Rvine.diag.a$Matrix))
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 1 5
##
## [[3]]
## [1] 1 5 2
##
## [[4]]
## [1] 1 5 2 4
##
## [[5]]
## [1] 1 5 2 4 6
##
## [[6]]
```

```
## [1] 1 5 2 4 3
##
## [[7]]
## [1] 1 5 2 6
##
## [[8]]
## [1] 1 5 2 6 3
##
## [[9]]
## [1] 1 5 4
##
## [[10]]
## [1] 1 5 4 6
##
## [[11]]
## [1] 1 5 4 6 3
##
## [[12]]
## [1] 1 2
##
## [[13]]
## [1] 1 2 4
##
## [[14]]
## [1] 1 2 4 6
##
## [[15]]
## [1] 1 2 4 6 3
##
## [[16]]
## [1] 5
##
## [[17]]
## [1] 5 4
##
## [[18]]
## [1] 5 4 6
##
## [[19]]
## [1] 5 4 6 3
##
## [[20]]
## [1] 5 4 6 3 2
```

One of the 20 missing data patterns that are imputable is $(5, 4)$ in line 17. This means in our 6-dimensional data, that we can simulate the variables 4 and 5 by simulating from corresponding conditional distribution functions. More precisely it is possible to simulate variable 4 from $F_{4|1,2,3,6}$ and then variable 5 from $F_{5|1,2,3,4,6}$. The order of the variables in the output is the reverse order for the simulation. We will store this missing data pattern for later applications.

```
mdp.a<-c(5,4)
```

The *diagforce()* function introduced earlier has another functionality if we use the argument *diagvec*. Here we can insert every missing data pattern we got from the *RVineAdhocMdp()* function. The output is an R-vine

object where the diagonal of the matrix starts with the *diagvec* argument.

```
aba.Rvine.diag.c<-diagforce(aba.Rvine,diagvec=mdp.a)
aba.Rvine.diag.c$Matrix
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    0    0    0    0    0
## [2,]    1    4    0    0    0    0
## [3,]    2    1    1    0    0    0
## [4,]    4    2    6    6    0    0
## [5,]    6    6    3    2    2    0
## [6,]    3    3    2    3    3    3
```

We create some test data by simulating 1000 observations from the fitted R-vine by means of the *VineCopula* function *RVineSim()*

```
aba.sim<-RVineSim(1000,aba.Rvine)
```

This data so far is complete and we will now artificially create an MCAR data, with the *MCARdata()* function from the *VineCopulaImpute* package. Besides a complete data set in the *data* argument, it can be specified how many variables are missing per observations at most by *max.var.miss* and how many observations shall be incomplete in total by *perc.miss*. The algorithm used here is Algorithm 5.4.

```
aba.incompl<-MCARdata(data=aba.sim,max.var.miss=5,perc.miss=0.3)
```

To summarize the missing data patterns and order the data by missing data patterns we use the *MdpPatSort()* function from the *VineCopulaImpute* package.

```
mdp.obj<-MdpPatSort(aba.incompl)
```

It has two outputs. The first is a table of all occuring missing data pattterns, sorted by number of observed variables.

```
mdp.obj$mdp.pat
```

```
##            Freq obs.var len diameter whole.wt shucked.wt viscera.wt shell.wt
## Pattern 1   700       6   1        1        1          1          1        1
## Pattern 2    10       5   0        1        1          1          1        1
## Pattern 3     7       5   1        0        1          1          1        1
## Pattern 4    11       5   1        1        0          1          1        1
## Pattern 5    11       5   1        1        1          0          1        1
## Pattern 6     5       5   1        1        1          1          0        1
## Pattern 7    11       5   1        1        1          1          1        0
## Pattern 8     6       4   0        0        1          1          1        1
## Pattern 9     2       4   0        1        0          1          1        1
## Pattern 10    4       4   1        0        0          1          1        1
## Pattern 11    2       4   0        1        1          0          1        1
## Pattern 12    5       4   1        0        1          0          1        1
## Pattern 13    6       4   1        1        0          0          1        1
## Pattern 14    4       4   0        1        1          1          0        1
## Pattern 15    3       4   1        0        1          1          0        1
```

```
## Pattern 16    4       4   1       1       0       1       0       1
## Pattern 17    1       4   1       1       1       0       0       1
## Pattern 18    4       4   0       1       1       1       1       0
## Pattern 19   10       4   1       0       1       1       1       0
## Pattern 20    4       4   1       1       0       1       1       0
## Pattern 21    2       4   1       1       1       0       1       0
## Pattern 22    6       4   1       1       1       1       0       0
## Pattern 23    3       3   0       0       0       1       1       1
## Pattern 24    2       3   0       0       1       0       1       1
## Pattern 25    3       3   0       1       0       0       1       1
## Pattern 26    3       3   1       0       0       0       1       1
## Pattern 27    3       3   0       0       1       1       0       1
## Pattern 28    2       3   0       1       0       1       0       1
## Pattern 29    3       3   1       0       0       1       0       1
## Pattern 30    7       3   0       1       1       0       0       1
## Pattern 31    4       3   1       0       1       0       0       1
## Pattern 32    1       3   1       1       0       0       0       1
## Pattern 33    2       3   0       0       1       1       1       0
## Pattern 34    2       3   0       1       0       1       1       0
## Pattern 35    4       3   1       0       0       1       1       0
## Pattern 36    5       3   1       0       1       0       1       0
## Pattern 37    3       3   1       1       0       0       1       0
## Pattern 38    4       3   0       1       1       1       0       0
## Pattern 39    1       3   1       0       1       1       0       0
## Pattern 40    5       3   1       1       0       1       0       0
## Pattern 41    2       3   1       1       1       0       0       0
## Pattern 42    4       2   0       0       0       0       1       1
## Pattern 43    3       2   0       0       0       1       0       1
## Pattern 44    6       2   0       0       1       0       0       1
## Pattern 45    3       2   0       1       0       0       0       1
## Pattern 46    4       2   1       0       0       0       0       1
## Pattern 47    4       2   0       0       0       1       1       0
## Pattern 48    3       2   0       0       1       0       1       0
## Pattern 49    5       2   0       1       0       0       1       0
## Pattern 50    5       2   1       0       0       0       1       0
## Pattern 51    5       2   0       0       1       1       0       0
## Pattern 52    4       2   0       1       0       1       0       0
## Pattern 53    6       2   1       0       0       1       0       0
## Pattern 54    6       2   0       1       1       0       0       0
## Pattern 55    2       2   1       0       1       0       0       0
## Pattern 56    3       2   1       1       0       0       0       0
## Pattern 57    7       1   0       0       0       0       0       1
## Pattern 58   11       1   0       0       0       0       1       0
## Pattern 59   15       1   0       0       0       1       0       0
## Pattern 60   11       1   0       0       1       0       0       0
## Pattern 61    8       1   0       1       0       0       0       0
## Pattern 62    8       1   1       0       0       0       0       0
```

For example in the last row we see that in 8 cases the variable 1 is given while the others are missing. The second output is the data set exactly ordered by the missing data patterns as in the first output. If we look at the last 9 observations we see again that we have there 8 cases where variable 1 is given while the others are missing.

```
mdp.obj$data.sort[992:1000,]
```

```
##              len  diameter whole.wt shucked.wt viscera.wt shell.wt
##  [1,]         NA 0.4701258       NA         NA         NA       NA
##  [2,] 0.65752417        NA       NA         NA         NA       NA
##  [3,] 0.65921719        NA       NA         NA         NA       NA
##  [4,] 0.64936341        NA       NA         NA         NA       NA
##  [5,] 0.07180502        NA       NA         NA         NA       NA
##  [6,] 0.67690092        NA       NA         NA         NA       NA
##  [7,] 0.74856413        NA       NA         NA         NA       NA
##  [8,] 0.52179019        NA       NA         NA         NA       NA
##  [9,] 0.82778386        NA       NA         NA         NA       NA
```

Now we can start with the imputation functions of the *VineCopulaImpute* package. The easiest one, which is the basic building block for all other imputation functions, is the *RVineAdhocMdpImp()* function. For a given R-vine it imputes a data block in which only one missing data pattern is inherent for all observations. For example take a data block with 10 observations and the missing data pattern $(5, 4)$.

```
aba.incompl.small.a<-aba.sim[1:10,]
aba.incompl.small.a[,mdp.a]<-NA
aba.incompl.small.a
```

```
##              len  diameter  whole.wt shucked.wt viscera.wt  shell.wt
##  [1,] 0.2020129 0.2236253 0.2229589         NA         NA 0.3721984
##  [2,] 0.6783106 0.6832659 0.5347435         NA         NA 0.5848666
##  [3,] 0.8525338 0.9003201 0.8768763         NA         NA 0.7319726
##  [4,] 0.2739792 0.2489346 0.2433077         NA         NA 0.4116668
##  [5,] 0.9352310 0.9153604 0.9503409         NA         NA 0.9233189
##  [6,] 0.4722414 0.3846019 0.3827821         NA         NA 0.4233471
##  [7,] 0.2743852 0.2722303 0.2867176         NA         NA 0.2067727
##  [8,] 0.7098494 0.5390492 0.7016345         NA         NA 0.7606895
##  [9,] 0.6832287 0.6849157 0.6068856         NA         NA 0.7716936
## [10,] 0.5999631 0.5796427 0.5533628         NA         NA 0.4525109
```

It is important that the missing data pattern used, is an output of the *RVineAdhocMdp()* function, otherwise it will produce an error later. The missing data block can now be imputed by simulating from the conditional distribution functions of the given R-vine. Additionally we need an *poss.sim.order* argument for the simulation order as input. The missing data patterns $(4, 5)$ and $(5, 4)$ coincide, however for our R-vine it is only possible to simulate first variable 4 and then variable 5, not vice versa. Read the simulation order argument $(5, 4)$ as "Simulate variable 5 after variable 4", not to be confused with the order. Note that the simulation order can even be longer as for example $(5, 4, 6)$, as long as it is an output of the *RVineAdhocMdp()* function.

```
aba.imp.small.a<-RVineAdhocMdpImp(data.incomplx=aba.incompl.small.a,
                                  RVM=aba.Rvine,
                                  poss.sim.order=c(5,4),
                                  mdp=c(4,5))
aba.imp.small.a
```

```
##              len  diameter  whole.wt shucked.wt viscera.wt  shell.wt
##  [1,] 0.2020129 0.2236253 0.2229589  0.1573086  0.1650949 0.3721984
##  [2,] 0.6783106 0.6832659 0.5347435  0.4234438  0.6626921 0.5848666
```

```
##  [3,] 0.8525338 0.9003201 0.8768763  0.9711437  0.6735325 0.7319726
##  [4,] 0.2739792 0.2489346 0.2433077  0.2771175  0.1709118 0.4116668
##  [5,] 0.9352310 0.9153604 0.9503409  0.8903761  0.8423557 0.9233189
##  [6,] 0.4722414 0.3846019 0.3827821  0.5483932  0.2433567 0.4233471
##  [7,] 0.2743852 0.2722303 0.2867176  0.2947984  0.4860477 0.2067727
##  [8,] 0.7098494 0.5390492 0.7016345  0.6662312  0.5664358 0.7606895
##  [9,] 0.6832287 0.6849157 0.6068856  0.5533028  0.6258841 0.7716936
## [10,] 0.5999631 0.5796427 0.5533628  0.7341290  0.3404579 0.4525109
```

Now we saw that we are able to impute one of the adhoc imputable missing data patterns identified by *RVineAdhocMdp()*. The *RVineAdhocImp()* function imputes all adhoc imputeable missing data patterns. If we look at the first 40 observations from our incomplete test data we see 9 missing data patterns.

```
aba.incompl.small.b<-aba.incompl[1:40,]
MdpPatSort(aba.incompl.small.b)$mdp.pat
```

```
##           Freq obs.var len diameter whole.wt shucked.wt viscera.wt shell.wt
## Pattern 1   32       6   1        1        1          1          1        1
## Pattern 2    1       4   1        0        0          1          1        1
## Pattern 3    1       4   1        1        1          1          0        0
## Pattern 4    1       3   0        1        0          1          0        1
## Pattern 5    1       3   0        1        1          0          0        1
## Pattern 6    1       2   0        0        0          1          1        0
## Pattern 7    1       2   1        0        0          1          0        0
## Pattern 8    1       1   0        0        0          0          1        0
## Pattern 9    1       1   0        0        0          1          0        0
```

After imputing all adhoc imputable missing data patterns, only 6 are left.

```
aba.imp.small.b<-RVineAdhocImp(aba.incompl.small.b,aba.Rvine)
MdpPatSort(aba.imp.small.b)$mdp.pat
```

```
##           Freq obs.var len diameter whole.wt shucked.wt viscera.wt shell.wt
## Pattern 1   35       6   1        1        1          1          1        1
## Pattern 2    1       4   1        0        0          1          1        1
## Pattern 3    1       4   1        1        1          1          0        0
## Pattern 4    1       3   0        1        0          1          0        1
## Pattern 5    1       2   0        0        0          1          1        0
## Pattern 6    1       2   1        0        0          1          0        0
```

Finally we come to the implementation of a general imputation algorithm after we introduce the *AddOrderVars()* function. Suppose the variables 1,2,3 and 4 are missing in a subset of the data set.

```
mdp.b<-c(1,2,3,4)
```

Then if we fit an R-vine to the variables 5 and 6, and want to add the other variables one by one, by looking at the Kendall's tau matrix of the complete data, we should do it, according to Definition 4.17, in this order:

```
AddOrderVars(mdp.b,aba.incompl)
```

```
## [1] 3 4 1 2
```

So if we have given a incomplete data set and want to impute it using Algorithm 4.18, we just call the *RVineImpute()* function. Further arguments to be passed are for the *RVineStructureSelect()* function, such as the copula families to be used. If the *progress* argument is set to TRUE, the user is informed about how many missing data patterns are still to be imputed after each iteration.

```
aba.imp.u<-RVineImpute(aba.incompl,
                       progress=TRUE,
                       familyset=c(0,1,2,3,4,5,13,14,23,24,33,34))
```

```
## 61 missing data patterns to impute...

## 41 missing data patterns to impute...

## 36 missing data patterns to impute...

## 25 missing data patterns to impute...

## 21 missing data patterns to impute...

## 14 missing data patterns to impute...

## 11 missing data patterns to impute...

## 10 missing data patterns to impute...

## 8 missing data patterns to impute...

## 6 missing data patterns to impute...

## 4 missing data patterns to impute...

## 2 missing data patterns to impute...

## 1 missing data pattern to impute...

## 0 missing data patterns to impute...
```

After the imputation is done, we transform the u-data back to its original scale with the *uvbacktranhyp()* function. The argument *origdata* tells from which data set the u-scale transformation was done.

```
aba.imp.x<-uvbacktranhyp(origdata=abalone1000f,unidata=aba.imp.u)
```

Note, earlier we transformed the complete data to the u-scale. In general this is not possible in an imputation background as there is no complete data. So the u-scale transformation is always based on the incomplete data, and so should in general the argument *origdata* of the *uvbacktranhyp()* function be also the incomplete data.

Appendix C

Additional Simulation Study Results for a Smaller Sample Size

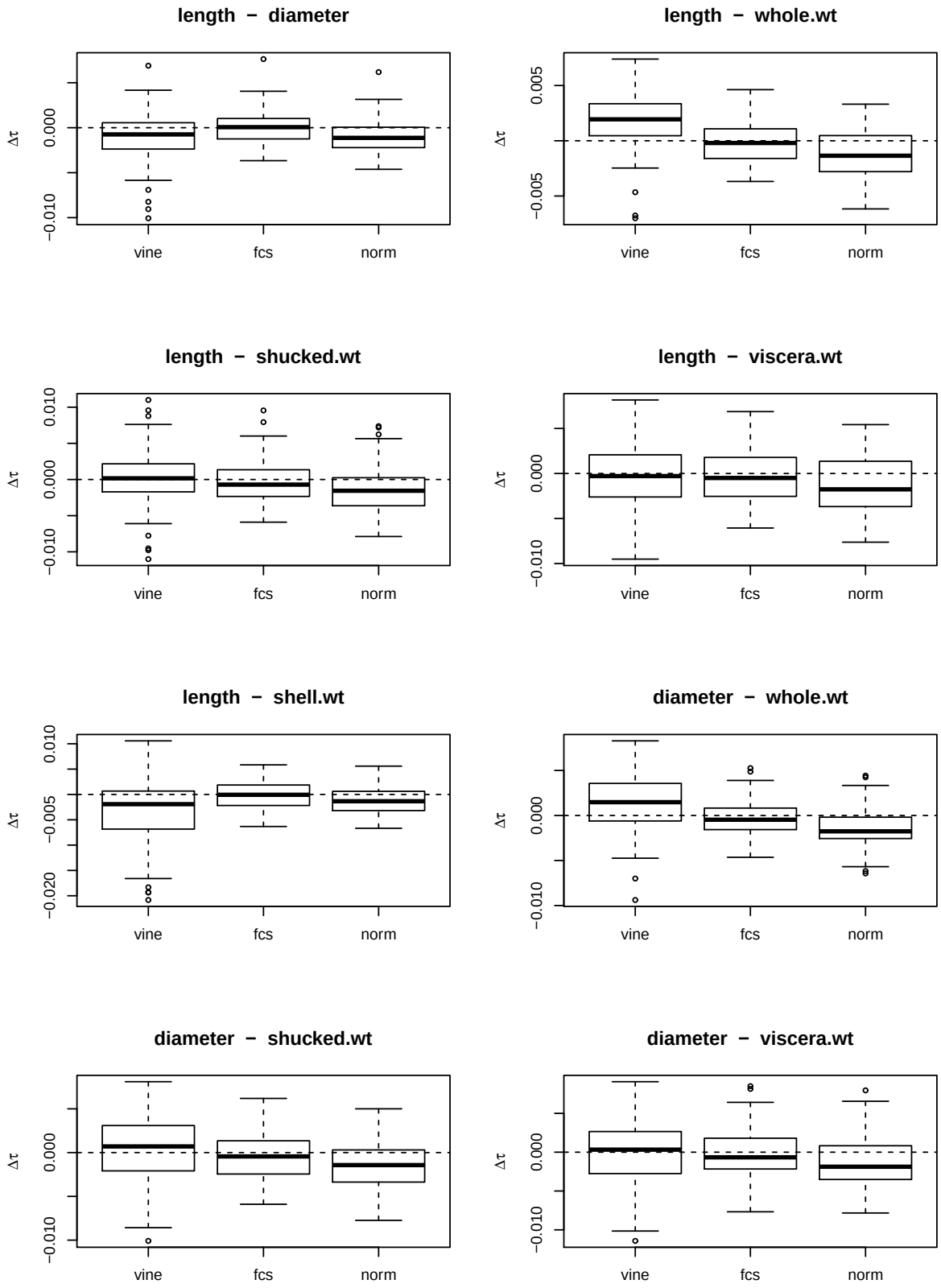


Figure C.1: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations.

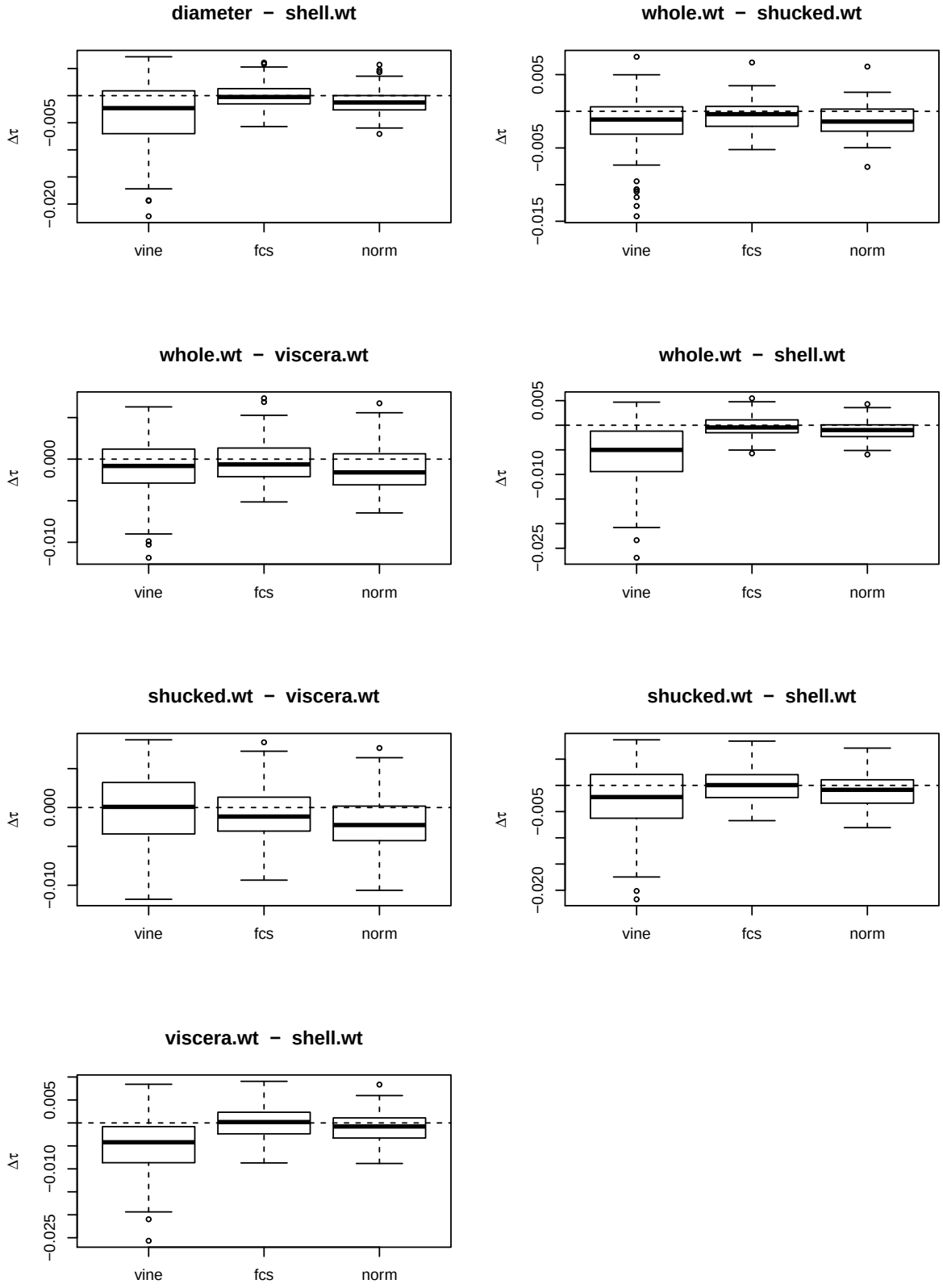


Figure C.2: Box plots of the Kendall's tau distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations (continued).

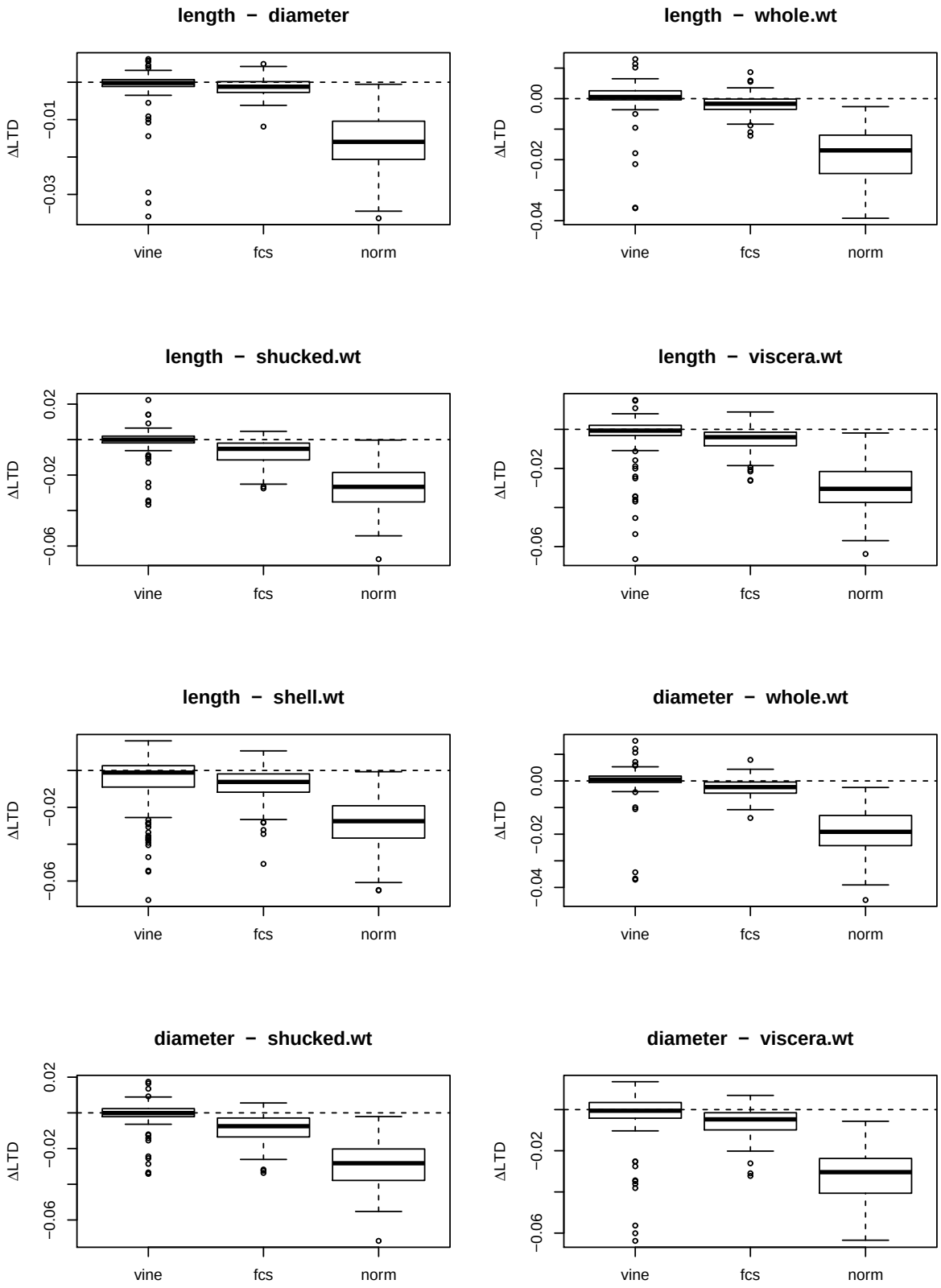


Figure C.3: Box plots of the lower tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations.

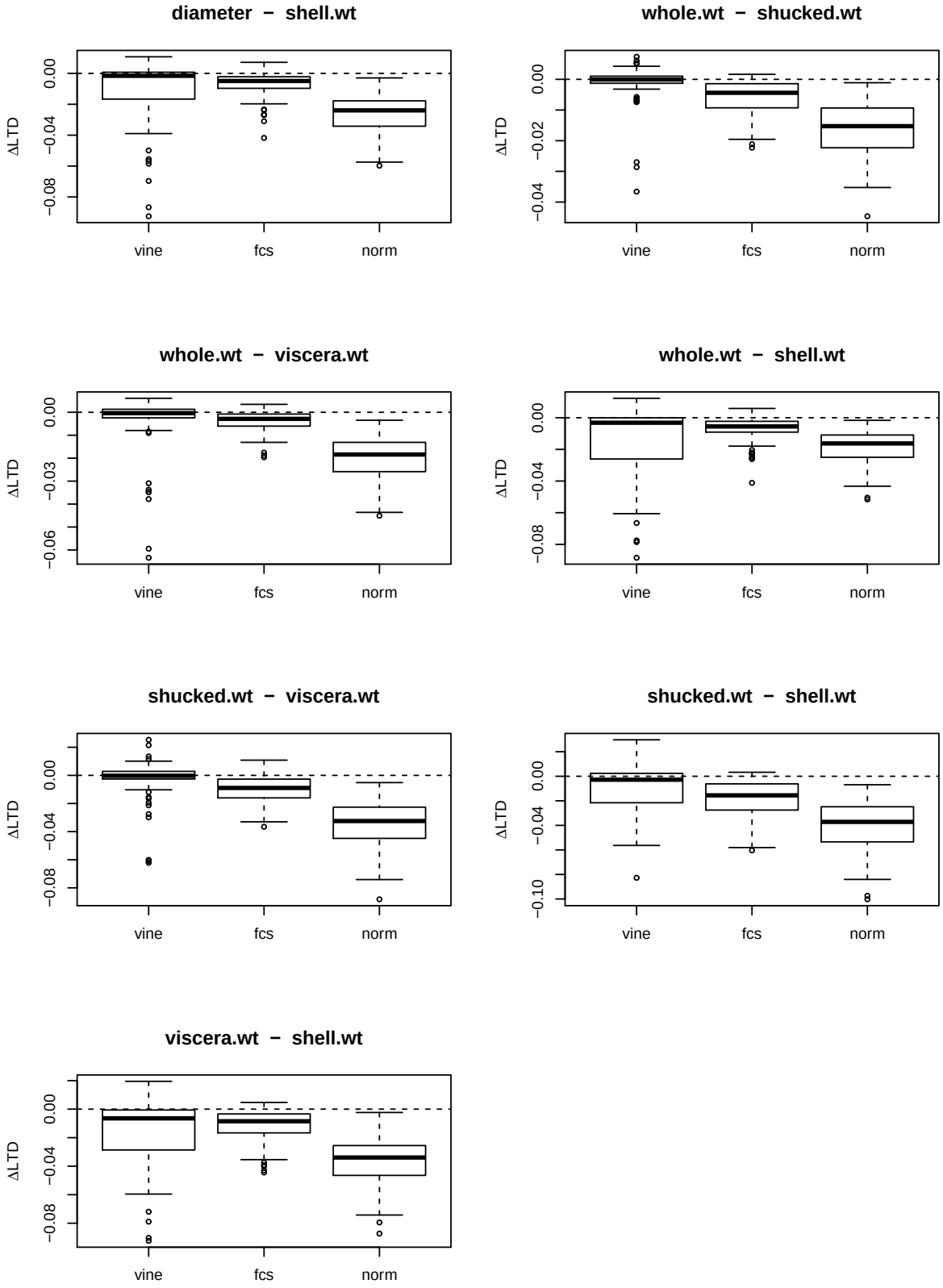


Figure C.4: Box plots of the lower tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations (continued).

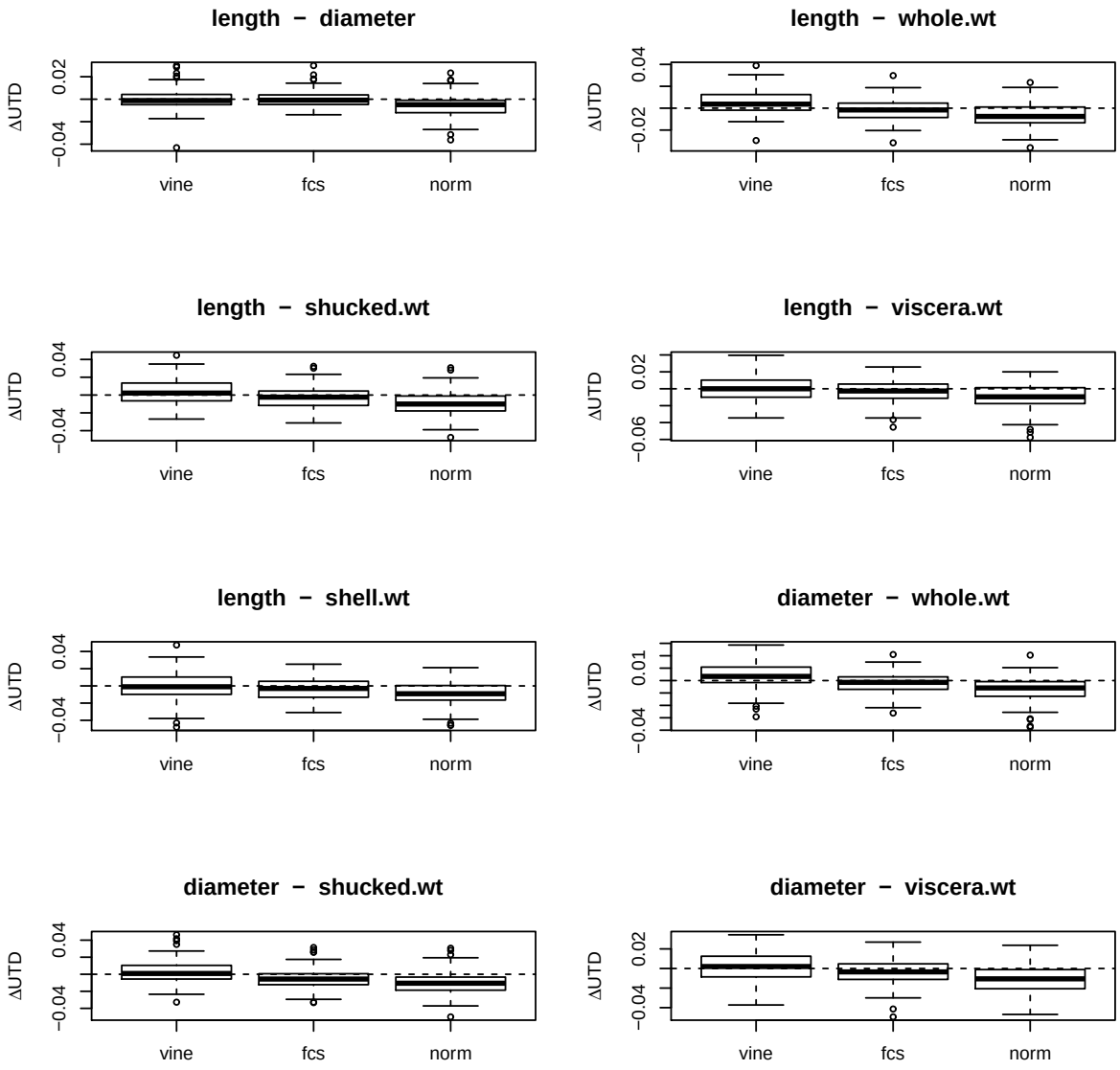


Figure C.5: Box plots of the upper tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations.

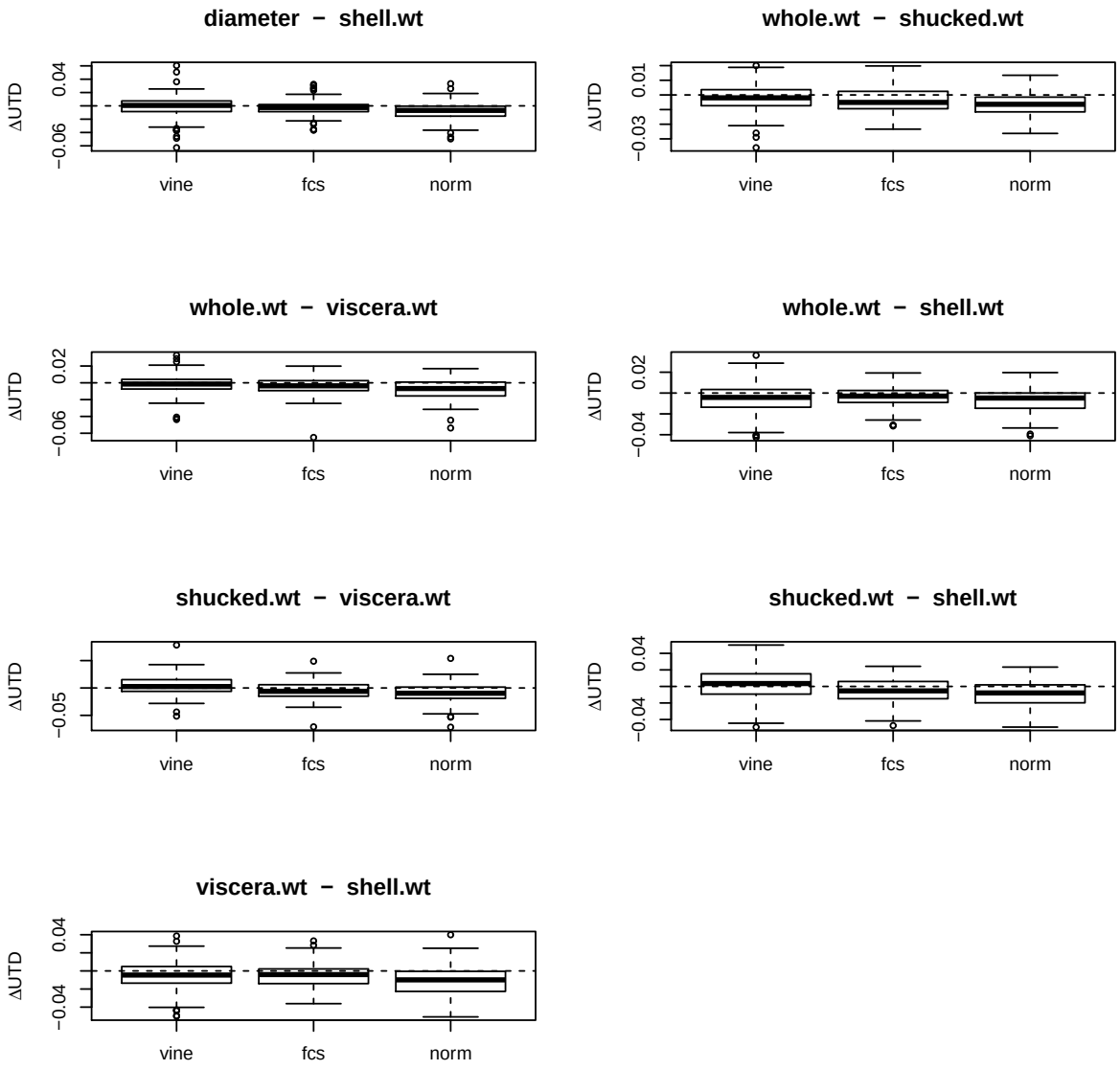


Figure C.6: Box plots of the upper tail dependence distance measures for all variable pairs of the abalone data for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations (continued).

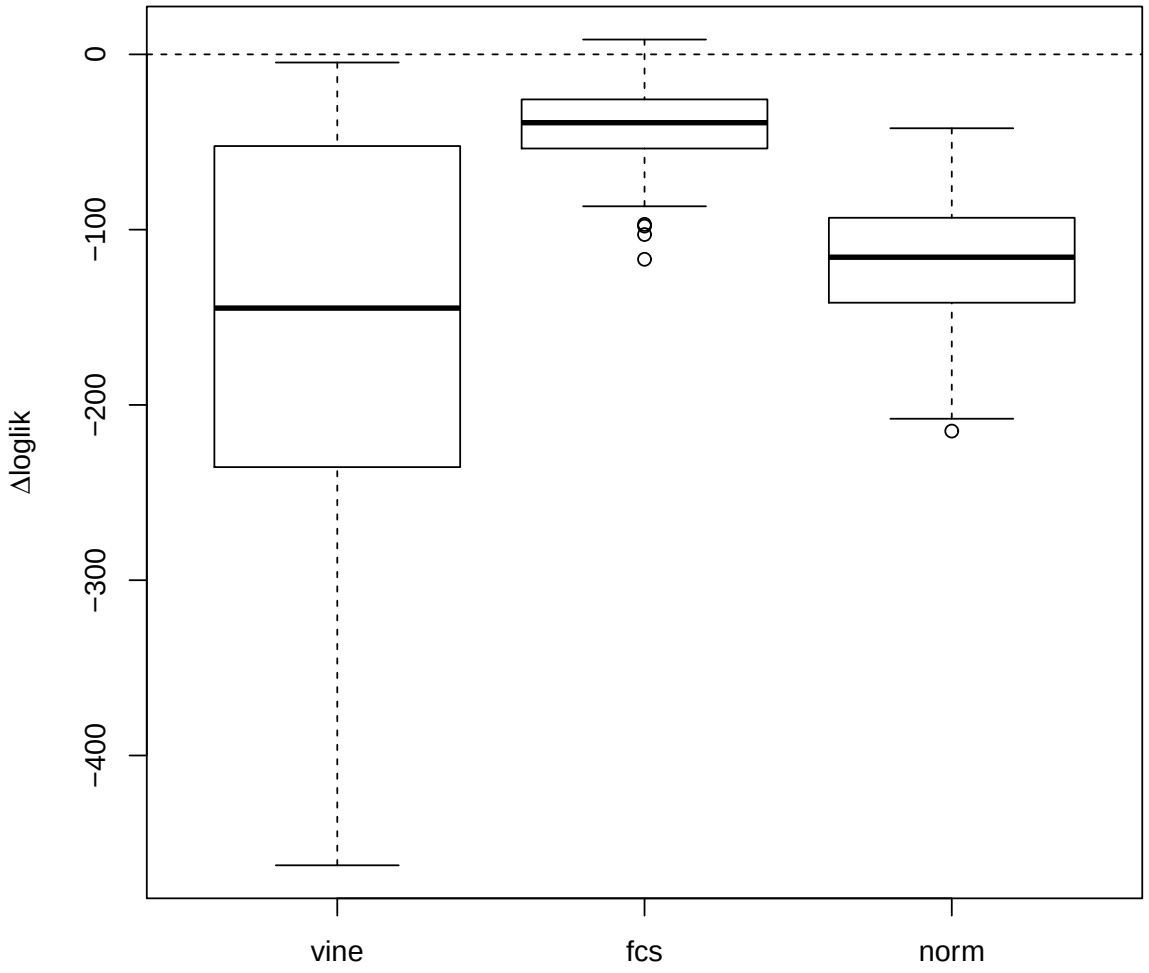


Figure C.7: Box plots of the log-likelihood distance measures for the imputation methods vine, fcs and norm in a scenario with 10% incomplete observations.

Bibliography

Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics 44*(2), 182–198.

Barndorff-Nielsen, O. (1977). Exponentially decreasing distributions for the logarithm of particle size. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 353*(1674), 401–419.

Barndorff-Nielsen, O. and C. Halgreen (1977). Infinite divisibility of the hyperbolic and generalized inverse gaussian distributions. *Zeitschrift f r Wahrscheinlichkeitstheorie und Verwandte Gebiete 38*(4), 309–311.

Bedford, T. and R. M. Cooke (2002). Vines–a new graphical model for dependent random variables. *The Annals of Statistics 30*(4), 1031–1068.

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Carpenter, J. R. and M. G. Kenward (2013). *Multiple imputation and its application*. Statistics in practice. Chichester: Wiley.

Czado, C. and J. Stoeber (2012). Pair copula constructions. In J.-F. Mai and M. Scherer (Eds.), *Simulating copulas*, Volume v. 4 of *Series in quantitative finance*, pp. 185–230. London: Imperial College Press.

Demirtas, H., S. A. Freels, and R. M. Yucel (2008). Plausibility of multivariate normality assumption when multiply imputing non-gaussian continuous outcomes: A simulation assessment. *Journal of Statistical Computation and Simulation 78*(1), 69–84.

Di Lascio, F. Marta L., S. Giannerini, and A. Reale (Eds.) (2014). *Imputation of complex dependent data by conditional copulas: analytic versus semiparametric approach: In: Gilli, M., Gonzalez-Rodriguez, G., Nieto-Reyes, A. (Eds.) Book of the 21st International Conference on Computational Statistics (COMPSTAT 2014)*.

Di Lascio, F. Marta L., S. Giannerini, and A. Reale (2015). Exploring copulas for the imputation of complex dependent data. *Statistical Methods & Applications 24*(1), 159–175.

Di Lascio, Francesca M. L. and S. Giannerini (2014). Coimp: Copula based imputation method. R package version 0.2-3, `http://CRAN.R-project.org/package=CoImp`.

Dißmann, J. (2010). Statistical inference for regular vines and application. Master's thesis, Technical University of Munich, Munich.

Dißmann, J., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis 59*, 52–69.

Dong, Y. and C.-Y. J. Peng (2013). Principled missing data methods for researchers. *SpringerPlus 2*(1), 222.

Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern classification* (2. ed. ed.). A Wiley-interscience publication. New York [u.a.]: Wiley.

Enders, C. K. (2010). *Applied missing data analysis*. Methodology in the social sciences. New York and London: Guilford.

Ennett, C. M., M. Frize, and C. Walker (2008). Imputation of missing values by integrating neural networks and case-based reasoning. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference 2008*, 4337–4341.

Essama-Nssah, B. (2006). *Propensity Score Matching And Policy Impact Analysis - A Demonstration In Eviews*. The World Bank.

Goodnight, J. H. (1978). *The sweep operator: Its importance in statistical computing*, Volume R-106 of *SAS technical report*. Cary, NC: SAS Institute.

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics 27*(4), 857.

Graham, J. W. (2009). Missing data analysis: making it work in the real world. *Annual review of psychology 60*, 549–576.

Graham, J. W., A. E. Olchowski, and T. D. Gilreath (2007). How many imputations are really needed? some practical clarifications of multiple imputation theory. *Prevention science : the official journal of the Society for Prevention Research 8*(3), 206–213.

Gross, J. L. and J. Yellen (2006). *Graph theory and its applications* (2nd ed. ed.). Discrete mathematics and its applications. Boca Raton and London: Chapman & Hall/CRC.

Joe, H. (1996). Families of m-variate distributions with given margins and $m(m-1)/2$ bivariate dependence parameters. In L. Rüschendorf, B. Schweizer, and M. D. Taylor (Eds.), *Distributions with fixed marginals and related topics*, Institute of Mathematical Statistics Lecture Notes - Monograph Series, pp. 120–141. Hayward, CA: Institute of Mathematical Statistics.

Jonsson, P. and C. Wohlin (11-17 Sept. 2004). An evaluation of k-nearest neighbour imputation using likert data. In *10th International Symposium on Software Metrics, 2004. Proceedings*, pp. 108–118.

Krupskii, P. and H. Joe (2014). Tail-weighted measures of dependence. *Journal of Applied Statistics 42*(3), 614–629.

Lichman, M. (2013). UCI machine learning repository: Abalone data set. `https://archive.ics.uci.edu/ml/datasets/Abalone`.

Little, R. J. A. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American Statistical Association 83*(404), 1198.

Luethi, D. and W. Breyman (2013). ghyp: A packageon the generalized hyperbolic distribution and its special cases. R package version 1.5.6, `http://CRAN.R-project.org/package=ghyp`.

McNeil, A. J., R. Frey, and P. Embrechts (2005). *Quantitative risk management: Concepts, techniques and tools / Alexander J. McNeil, Rüdiger Frey, Paul Embrechts*. Princeton series in finance. Princeton, N.J. and Woodstock: Princeton University Press.

Nelsen, R. B. (2006). *An introduction to copulas* (2nd ed. ed.). Springer series in statistics. New York, N.Y.: Springer.

Revelle, W. (2015). psych: Procedures for psychological, psychometric, and personality research. R package version 1.5.8, `http://CRAN.R-project.org/package=psych`.

Rubin, D. B. (1976). Inference and missing data. *Biometrika 63*(3), 581–592.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. Wiley series in probability and mathematical statistics. Applied probability and statistics, 0271-6232. New York and Chichester: Wiley.

Schafer, J. L. (1997). *Analysis of incomplete multivariate data*, Volume 72 of *Monographs on statistics and applied probability*. London: Chapman & Hall.

Schepsmeier, U., J. Stoeber, E. C. Brechmann, B. Graeler, T. Nagler, and Erhardt Tobias (2015). Vinecopula: Statistical inference of vine copulas. R package version 1.6-1, `http://CRAN.R-project.org/package=VineCopula`.

Tanner, M. A. and W. H. Wong (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association 82*(398), 528.

Tanner, M. A. and W. H. Wong (2010). From em to data augmentation: The emergence of mcmc bayesian computation in the 1980s. *Statistical Science 25*(4), 506–516.

Templ, M., A. Alfons, A. Kowarik, and B. Prantner (2015). Vim: Visualization and imputation of missing values. R package version 4.4.1, `http://CRAN.R-project.org/package=VIM`.

van Buuren, S. (2012). *Flexible imputation of missing data*. Chapman & Hall/CRC interdisciplinary statistics series. Boca Raton, FL: CRC Press.

van Buuren, S., J. P. Brand, C. G. Groothuis-Oudshoorn, and D. B. Rubin (2006). Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation 76*(12), 1049–1064.

van Buuren, S., K. Groothuis-Oudshoorn, A. Robitzsch, G. Vink, L. Doove, and S. Jolani (2015). mice: Multivariate imputation by chained equations. R package version 2.25, `http://CRAN.R-project.org/package=mice`.

Vink, G., L. E. Frank, J. Pannekoek, and S. van Buuren (2014). Predictive mean matching imputation of semicontinuous variables. *Statistica Neerlandica 68*(1), 61–90.