BIVARIATE FUNCTIONAL NORMALIZATION OF METHYLATION ARRAY DATA

By

CLIFFORD YACAS

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

MCMASTER UNIVERSITY
Department of Mathematics and Statistics

DECEMBER 2020

ACKNOWLEDGMENT

Many people supported me during my education and work for this thesis. I would like to thank my supervisor, Dr. Angelo Canty, who paved the way for me to embark on many new educational adventures. His encouragement and guidance greatly inspired me in my pursuit of interesting research and I could not have done this without his help and guidance with my research throughout the COVID-19 pandemic. Furthermore, I would like to thank the great people / peers that I have met throughout this masters, especially the Msc. Basement Dwellers, who have helped me more than they know. I would also like to thank Dr. Ben Bolker and Dr. Guillaume Paré for examining my defense. Lastly, I would like to thank my friends and family for their love and support.

BIVARIATE FUNCTIONAL NORMALIZATION OF METHYLATION ARRAY DATA

ABSTRACT

DNA methylation plays a key role in disease analysis, especially for studies that compare known large scale differences in CpG sites, such as cancer/normal studies or between-tissues studies. However, before any analysis can be done, data normalization and preprocessing of methylation data are required. A useful data preprocessing pipeline for large scale comparisons is Functional Normalization (FunNorm), (Fortin et al., 2014) implemented in the *minfi* package in R. In FunNorm, the univariate quantiles of the methylated and unmethylated signal values in the raw data are used to preprocess the data. However, although FunNorm has been shown to outperform other preprocessing and data normalization processes for these types of studies, it does not account for the correlation between the methylated and unmethylated signals into account; the focus of this paper is to improve upon FunNorm by taking this correlation into account. The concept of a bivariate quantile is used in this study as an attempt to take the correlation between the methylated and unmethylated signals into consideration. From the bivariate quantiles found, the partial least squares method is then used on these quantiles in this preprocessing. The raw datasets used for this research were collected from the European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI) website. The results from this preprocessing algorithm were then compared and contrasted to the results from FunNorm. Drawbacks, limitations and future research are then discussed.

Contents

List of Tables

List of Figures

Dedication

This thesis is dedicated to Frendell, who always does his best.

Chapter One

Introduction

1.1 Background Information on Epigenetics

Before epigenetics is discussed, it is important understand the basis of genetics. This basis starts with *Deoxyribonucleic acid*, or DNA. A DNA molecule is a double-stranded polymer that carries the genetic information of a living organism. Each single strand of one DNA molecule is made up of *nucleotides*, which are molecules made up of a sugar group, a phosphate group and one of four nitrogen bases, which are *purines* composed of adenine (A) and guanine (G) and *pyrimidines* composed of cytosine (C) and thymine (T). Two strands are joined together by binding of the complementary bases to form a double helix structure. The binding rules for DNA, named the Watson-Crick base pairings rules, are that A binds with T and G binds with C. Due to this pairing rule, each strand of DNA contains the exact same genetic information, with different coding. A sequence of DNA that produces a protein or other functional element is referred to as a *gene* and the *genome* is the complete set of genetic information in an organism. Genomes are stored in chromosomes, which are made up of long strands of DNA tightly coiled together around histone proteins. Each cell within the human body contains the complete genome, which consists of 2 copies of the 22 autosomal chromosomes and 2 sex chromosomes. Lastly, a locus is a specific, fixed position on the genome where a particular gene or genetic marker is located.

Epigenetics is the study of changes in organisms caused by modification of gene expression rather than alteration of the genetic code itself. For a gene to produce a protein, DNA is first converted to *Ribonucleic Acid* or RNA, which is comprised of the coding regions for functional elements spliced together. Expression is the amount of RNA produced for a given gene in a cell. This varies across cells for the same gene and across genes in the same cell. Epigenetic changes involve genetic control by factors other than an individual's DNA sequence, usually involving modifications via chemical compounds that regulate the gene's activity. These changes can switch genes "on" or "off" and determine which proteins are made. By "turning on" or expressing certain sets of genes and "turning off" or inhibiting other sets, these changes are one of the main ways in which cells, tissues, and organs in most living organisms differ from one another. The epigenome comprises all of the chemical compounds that have been added to the entirety of a genome as a way to regulate the expression of all the genes within the genome. The chemical compounds of the epigenome are not part of the DNA sequence, but are physically attached to DNA. Epigenetic modifications remain as cells divide and in some cases can be inherited through the generations. Environmental influences, such as a person's diet and exposure to pollutants, can also impact the epigenome. Within cells, there are three systems that can interact with each other to silence genes: histone modifications, RNA-associated silencing, and DNA methylation. However, DNA methylation usually is the main focus in epigenetic research as it is the most common epigenetic signaling tool that cells use to lock genes in the "off" position, as well as proven to be important in Epigenome Wide Association Studies, or EWAS, which is the study of the association of any epigenetic marks across the whole genome. This is primarily due to DNA methylation being easily quantifiable at genetic loci across the entire genome. Note however that other forms of EWAS also exist (Rakyan et al., 2011).

1.2 Background Information on Methylation

DNA methylation is a biological process by which methyl groups (a carbon atom with 3 hydrogen atoms attached) are added to the DNA molecule. In mammals, DNA methylation occurs at cytosines in any context of the genome; however, more than 98% of DNA methylation occurs in a CpG dinucleotide, or Cytosine—phosphate—Guanine context, abbreviated in the literature as CpG sites. Relatedly, *a CpG island* is a region on the genome in which there are a large number of repeating CpG sites. (B. Jin et al., 2011). The addition of these methyl groups then inhibits the expression of the gene to which they are attached. Research has shown that methylation is an important component in numerous cellular processes, and has been linked to the formation of human diseases, such as cancer. In particular, research has shown that there are significant differences in overall and specific methylation levels between different tissue types and between normal cells and cancer cells from the same tissue (Z. Jin and Liu, 2018).

Recent advances in medical technology have been made in regards to obtaining and analyzing methylation data. In particular, Illumina, a biotechnology company, has developed the Infinium HumanMethylation microarray assay, which offers a cost-effective, high throughput method for quantitatively assessing methylation across the genome using limited amounts of DNA. The HumanMethylation450 (450K) BeadChip assays DNA methylation at 482,421 CpG sites using bisulfide treated DNA, which is the treatment of DNA with sodium bisulfite ($NaHSO_3$).This chemical compound converts unmethylated cytosines into another nucleotide called uracil (U). The cytosines that have not converted into uracil are methylated. The treated DNA is then amplified through *Whole genome amplification* which is a method for generating copies of an entire genome when the starting amount of DNA material is limited. After amplification, the unmethylated cytosines appear as thymines. (Bibikova et al., 2011).

1.3 Background Information on the 450K array

The Infinium HumanMethylation450 array, or 450K array for short, makes it possible to assess the methylation status of >450,000 CpG sites located throughout the genome. It provides genome-wide coverage featuring comprehensive gene region and coverage of regions of the genome that contain a large number of CpG dinucleotide repeats, or CpG islands, as well as content outside of these regions that are important for studies and analysis. Its uniqueness lies in the use of two different types of chemical probes named Infinium I and Infinium II assays (or Type I and Type II assays), that are used on bisulfite treated DNA which then attach to CpG regions of the bisulfite treated DNA that have been selectively chosen by methylation experts.

In terms of Type I probes, there are two bead types for each CpG site per locus. These probes comprise of a bead and single-stranded DNA oligonucleotides, DNA strands comprised of a small amount of nucleotides, that differ in sequence only at the free, unconnected end. Bead types will correspond to the methylated cytosine locus and the other will correspond to the unmethylated cytosine locus, which has been converted into uracil during bisulfite treatment, and converted to thymine after amplification. The bisulfite-converted amplified DNA products are split into single strands and hybridized, or combine, to the chip to either the methylation-specific probe or the non-methylation probe. Hybridization is followed by single-base extension with dideoxynucleotides (ddNTPs) attached with either biotin or 2,4-dinitrophenol (DNP). The ddCTP and ddGTP are labeled with biotin while ddATP and ddUTP are labeled with DNP, with biotin fluorescing green and DNP fluorescing red when staining occurs. After the staining process, the chip is scanned to show the intensities of the unmethylated and methylated bead types. These attached probes will then fluoresce green for methylated signals and red for unmethylated signals. The light intensities are then measured to obtain the methylated and unmethylated values at each chosen CpG site. The main difference between Type I assays and Type II assays is that the Type I assay uses two

types of probes, with one probe used for the methylation measurement and another probe for the unmethylated measurement, whereas the Type II assay uses a single probe for both methylated and unmethylated measurements. The two probes on the Type I assay have the same base extension, whereas The Type II assay uses a single probe for both alleles, and base extension depends on the methylation state of the hybridized genomic DNA molecule.

Due to their different chemistries, the Type I and Type II probes each have their distinct advantages. Since Type II probes use half the physical space on the BeadChip compared with Type I, it is easier to manufacture these probes. As such, these probe types consists of approximately two-thirds of the a $450K$ array, thereby allowing a more in-depth coverage of a patient's genome. However, Type I probes' design characteristics mean they can measure methylation at more CpG dense regions than Type II probes (Pidsley, Zotenko, et al., 2016). Furthermore, the overall distribution of methylation values measured using Type I and Type II probes is different and it has been reported that Type II probes are sometimes both less reproducible and sensitive (Wu et al., 2013).

Figure 1.1 shows a basic outline of both assays. Also, it is worth noting that the 450K array contains 848 control probes. These probes are divided into negative control probes (613), probes intended for between array normalization (186) and the remainder (49), which are designed for quality control. For a more in-depth understanding of the 450K array, refer to (Bibikova et al., 2011).

From the analysis of methylated and unmethylated intensities of 450K data, the methylation level is then estimated based on the measured intensities of these pair of probes. To date, two methods have been proposed to measure the methylation level; the Beta value and the M-value.

The Beta value, ranging from 0 to 1, has been widely used to measure the percentage of methylation, a method which is recommended by Illumina. The Beta-value for an i^{th} interrogated CpG site is defined the following equation:

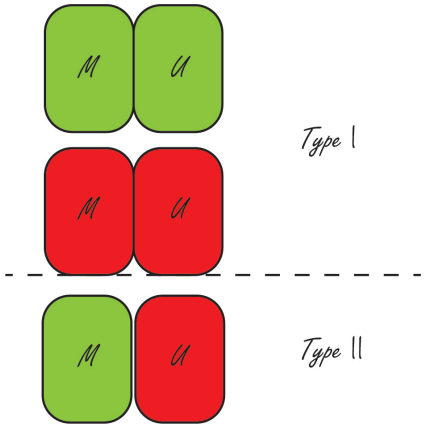

Figure 1.1 For the Infinium I design, both signals are measured in the same color: one probe for the methylated signal and one probe for the unmethylated signal. For Infinium II design, only one probe is used. The intensity read in the green channel measures the methylated signal, and the intensity read in the red channel measures the unmethylated signal.

$$\text{Beta}_i = \frac{\max\left(y_{i,\text{methy}}, 0\right)}{\max\left(y_{i,\text{unmethy}}, 0\right) + \max\left(y_{i,\text{methyl}}, 0\right) + \gamma_1} \tag{1.1}$$

where $y_{i,\text{methy}}$ and $y_{i,\text{unmethy}}$ are the intensities measured by the *ith* methylated and unmethylated probes, respectively. To avoid negative values , any negative values will be reset to 0. Illumina recommends adding a constant offset γ_1 (by default, $\gamma_1 = 100$) to the denominator to regularize the Beta value when both methylated and unmethylated probe intensities are low. The Beta value results in a number between 0 and 1. Equation 1.1 thus shows an intuitive approach to seeing methylation levels; Beta values close to 0 are sites that are highly unmethylated, wheres values close to 1 are sites that are highly methylated.

The M-value is calculated as the log_2 ratio of the intensities of methylated probe versus unmethylated probe as shown in the equation below:

$$M_i = \log_2\left(\frac{\max\left(y_{i,\text{methy}}, 0\right) + \gamma_2}{\max\left(y_{i,\text{unmethy}}, 0\right) + \gamma_2}\right) \tag{1.2}$$

where γ_2 is usually equal to 1 in this case and is added to the intensity values to prevent unexpected big changes due to small intensity estimation errors, as well as to avoid division by 0 and/or taking logs of 0. In literature, the Beta-value has a more intuitive biological

interpretation, and the Beta-value statistics are more widely used than the M-value counterpart when reporting the results to investigators. However, the M-value has shown to be more statistically valid for the differential analysis of methylation levels, as the M-value method provides much better performance in terms of Detection Rate (DR) and True Positive Rate (TPR) for both highly methylated and unmethylated CpG sites (Du et al., 2010).

However, before looking at these values of methylation, due the nature of microarrays having non-biological issues such as non-specific hybridization (sequences other than the intended target of a selected probe), bleeding of the signal into background and human issues such as unequal DNA amounts, data normalization or preprocessing is a critical step to consider before data analysis (Wang et al., 2015).

Several methods have been developed to preprocess and normalize the 450K array data, in order to adjust for assay type or color bias, subtract background signals, eliminate systematic errors and other batch effects (non-biological factors in an experiment that causes changes in the data produced by the experiment). One method is quantile normalization, developed by Bolstad et al. (2003), for gene expression microarrays but has been since been extended for 450K data by Touleimat and Tost (2012). In quantile normalization, the signal intensity of a probe replaces with the mean intensity of the probes that have the same ordinal rank from all studied samples, and thus makes the distribution of probe intensities from each array the same. In a related manner, when a reference distribution is defined, quantile normalization is done by replacing the signal intensity of a probe in the test distribution with signal intensity of the probe in the reference distribution that has the same rank as the probe in question. However, it is noted that raw data should be preprocessed by color balance adjustment and background correction before quantile normalization (Wang et al., 2015).

For quantile normalization of 450K data described by Touleimat and Tost (2012), a stratified version of quantile normalization is used, with the method being applied to methylated and unmethylated intensities separately. The distribution of Type I and Type II signals are forced to be the same by first quantile normalizing the Type II probes across samples and

then interpolating a reference distribution to which the Type I probes will be normalized. However, before this interpolation is done, the probes are stratified by probe regions where they vary, with these regions specified by Touleimat and Tost (2012).

In recent years, an extension to quantile normalization created by (Fortin et al., 2014), named Functional Normalization, has been made that removes unwanted technical variation and batch effects. For the purpose of this thesis, background information of Functional Normalization is needed.

1.4 Functional Normalization

Functional normalization extends the idea of quantile normalization by adjusting for known covariates measuring unwanted variation and was developed by Fortin et al. (2014).

1.4.1 Definitions

Let $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ be high dimensional, continuous observations and let $Z_{i,j}$ be their associated scalar covariates, with $i = 1, \ldots, n$ indicating samples and $j = 1, \ldots, m$ indexing covariates. The known covariates are chosen such that they are associated with unwanted variation and unassociated with biological variation. As such, functional normalization attempts to remove their influence. Hence, consider the empirical quantile function for the marginal distribution of $\mathbf{Y}_i$, q_i^{emp}. Since quantile functions are defined on the unit interval, to evaluate them pointwise, let $r \in [0, 1]$. The model in pointwise form is the following:

$$q_i^{\text{emp}}(r) = \alpha(r) + \sum_{j=1}^{m} Z_{i,j}\beta_j(r) + \varepsilon_i(r) \tag{1.3}$$

where $\alpha(r)$ is the mean of the quantile functions across all samples, $\beta_j(r)$ are the coefficient functions associated with the covariates and ε_i are the error functions that are assumed to be independent and centered around 0. $\sum_{j=1}^{m} Z_{i,j}\beta_j(r)$ represents variation in the quantile

functions explained by the covariates. From how $Z_{i,j}$ is chosen, functional normalization removes unwanted variation by regressing out $\sum_{j=1}^{m} Z_{i,j}\beta_j(r)$. After obtaining estimates $\widehat{\beta}_j(r)$ for $j = 1, \ldots, m$, the functional normalized quantiles can be obtained as such:

$$q_i^{\text{Funnorm}}(r) = q_i^{\text{emp}}(r) - \sum_{j=1}^{m} Z_{i,j}\widehat{\beta}_j(r) \tag{1.4}$$

$\mathbf{Y}_i$ is then transformed into the functional normalized quantity $\widetilde{\mathbf{Y}}_i$ using the formula

$$\widetilde{\mathbf{Y}}_i = q_i^{\text{Funnorm}}\ \left(F_i^{\text{emp}}\left(\mathbf{Y}_i\right)\right) \tag{1.5}$$

where $F_i^{\text{emp}} = (q_i^{\text{emp}})^{-1}$. This guarantees that the marginal distribution of $\widetilde{\mathbf{Y}}_i$ has q_i^{Funnorm} as its quantile function.

To obtain estimates for $\widehat{\beta}_j(r)$ for $j = 1, \ldots, m$, note that Equation 1.3 is an example of function-on-scalar regression. Function-on-scalar regression makes assumptions about the smoothness of the coefficient functions and uses a penalized framework due to the observations usually appearing noisy and non-smooth, such as the use of the $l1$ and $l2$ norm (Reiss et al., 2010). However, since $\mathbf{Y}_i$ are high dimensional and continuous, the jumps of the empirical quantile functions are very small, thereby allowing to bypass the smoothing approach used in traditional function-on-scalar regression. Thus, Fortin et al. (2014) proposes the use of a dense grid of H equidistant points between 0 and 1,, where the dimension of H is chosen to be much smaller than the dimension of $\mathbf{Y}_i$. Subsequently, Equation 1.3 reduces pointwise to a standard linear model on this grid. Since the empirical quantile functions $q^{\text{emp}}(r)$ have very small jumps, the variation between parameter estimates from the linear models of two neighbouring grid points are small. As such, H standard linear model fits can be used to compute estimates $\widehat{\alpha}(h)$ and $\widehat{\beta}_j(h), j = 1, \ldots, m$, with $\{h \in d/H : d = 0, 1, \ldots, H\}$. Estimates then can be formed for $\widehat{\alpha}(r)$ and $\widehat{\beta}_j(r), j = 1 \ \ldots, m$, for any $r \in [0, 1]$ via linear interpolation. This allows for faster computation time than other well established methods.

From the general framework, it is easy to see how functional normalization is related to quantile normalization. In particular, if we let $Z_{i,j}$ be covariates that associate with both

wanted and unwanted variation, then the application of functional normalization removes all variation and is equivalent to quantile normalization. In contrast, including no covariates makes the model comparable to no normalization at all. By choosing covariates that only measure unwanted technical variation, functional normalization will remove the variation explained by these technical measurements and preserve biological variation.

1.4.2 Functional Normalization for 450K Array Data

As mentioned before, the 450K array contains 848 control probes, none of which are designed to measure a biological signal. As such, Fortin et al. (2014) proposed using these control probes as surrogates for the effect of non-specific hybridization. Furthermore, this paper also uses out-of-band probes, which are the measurement of intensities of Type I probes in the "wrong" color channel. For more information on "out-of-band" probes, refer to Triche et al. (2013). The control probes and out-of-band probes are then transformed into 42 summary measures, with the control probes contributing 38 of these 42 measures and the out-of-band probes contributing four. For an in-depth description of how these summarized measurements were obtained, refer to Fortin et al. (2014).

These 42 summary control measures allow the application of functional normalization to 450K array data to remove unwanted batch effects. For functional normalization for 450K array data, the algorithm is applied to the methylated (M) and unmethylated (U) channels separately. Due to the relationship between the methylation values and the control probes differing between Type I and Type II probes, functional normalization is also applied separately by probe type to obtain more representative quantile distributions. Furthermore, probes on the sex chromosomes are normalized (11,232 and 416 probes for the X and Y chromosomes, respectively) separately from the autosomal probes. For each of the two sex chromosomes, males and females were normalized separately when there are sufficient sample sizes in each gender to allow it. For the X chromosome, functional normalization is used,

and for the Y chromosome, regular quantile normalization is used. The reason for this is the number of H points is chosen to be 500, whereas the number of probes on the Y chromosome is 413. Since the dimension of Y is assumed to have a much larger dimension than the dimension of H, the assumptions of functional normalization are violated. This results in multiple applications of functional normalization, using the exact same covariate matrix. From the 42 summary control measurements, the first $m = 2$ principal components of the summary control measures are used as covariates in functional normalization based on empirical observations on several data sets (Fortin et al., 2014)

1.5 Limitations of Functional Normalization

One notable observation from beta values obtained from methylation data for the 450K array is that, in almost all cases, the beta values are bimodal in nature. An example of this is shown in Figure 1.2.

However, from the definition of Beta-values, as well as biological domain knowledge,

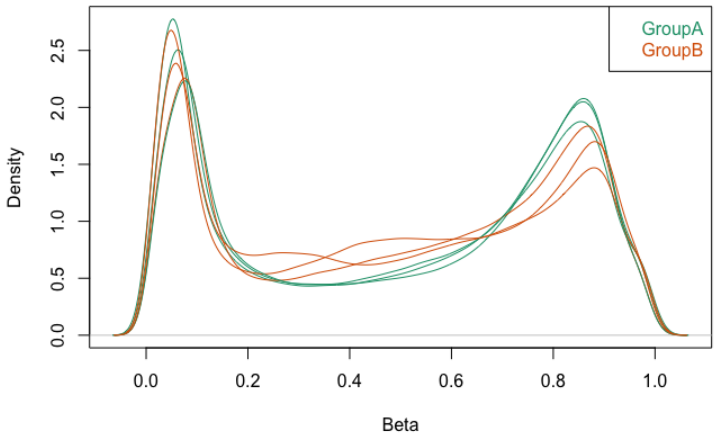

Figure 1.2 The Bimodal Nature of Beta Values from methylation usually seen on 450K array data. Note that Group A values stem from sample tissues that are normal / healthy and Group B values are from sample tissues that have cancer.

this bimodal nature is due to the values of the methylated and unmethylated signals. In particular, from a biological standpoint, it makes sense that when a region of DNA is highly methylated, then its methylation signal would be relatively high, whereas its unmethylated signal should be relatively low. The converse is also true, and thus, there is usually a negative correlation between the methylated and unmethylated signals (Pidsley, Wong, et al., 2013). However, in most preprocessing and data normalization methods for the 450K, this correlation is never taken into consideration. As such, the main objective of this research is to develop a way to take this correlation into account when doing functional normalization. In particular, the research proposed in this thesis tries to improve on functional normalization via the use of a definition of a bivariate quantile, as well as the use of partial least squares to take the correlation between the two signals into account.

Chapter Two

Bivariate Quantiles

As stated before, to take the correlation of the methylated and unmethylated signals into account when doing preprocessing and data normalization of 450K array data, the use of a bivariate quantile is applied in this research. There are two definitions of a bivariate quantile that have been looked at for the purpose of this research, the first definition coming from Chen and Welsh (2002) and the second coming from Vineshkumar and Nair (2019). However, as research was conducted, particular limitations with these definitions of bivariate quantiles were found when applying these definitions on the 450K array data, which will be outlined later in this thesis. As such, the need to develop a new definition of a bivariate quantile was needed, which will, again, be discussed later.

2.1 Bivariate Quantiles defined by Chen and Welsh (2002)

Chen and Welsh (2002) firstly defines the North-South (NS) bivariate quantile points, and then defines bivariate quantile points using the NS definition.

2.1.1 Definitions

Suppose that the direction is fixed for convenience from south to north. Likewise, let F be the joint cumulative density function (CDF) for X_1 and X_2, F_1 be the CDF of X_1, F_2 be the CDF

of X_2 and F_{12} be the joint CDF evaluated at $X_1 = x_1$ and $X_2 = F_2^{-1}(\alpha_1 + \alpha_2)$. Then the (α_1, α_2) NS bivariate quantile point is the vector $\xi(\alpha_1, \alpha_2) = \left(F_{12}^{-1}(\alpha_1, \alpha_2), F_2^{-1}(\alpha_1 + \alpha_2)\right)'$ which satisfies

$$F_2^{-1}(\alpha_1 + \alpha_2) = \inf\{x_2 : F_2(x_2) \geq \alpha_1 + \alpha_2\}$$

and

$$F_{12}^{-1}(\alpha_1, \alpha_2) = \inf\left\{x_1 : F\left(x_1, F_2^{-1}(\alpha_1 + \alpha_2)\right) \geq \alpha_1\right\}$$

for $\alpha_1, \alpha_2 \geq 0$ and $\alpha_1 + \alpha_2 \leq 1$. The α th NS bivariate quantile point is defined as $\xi(\alpha) = \xi\left(\frac{1}{2}\alpha, \frac{1}{2}\alpha\right)$ for $0 \leq \alpha \leq 1$,.

As noted by Chen and Welsh, using the north-south direction, while natural, is fixed and arbitrary. As such, using the definition of NS bivariate quantile points, as well as using the distribution of X to specify the appropriate direction, Chen and Welsh develop a definition of bivariate quantile points.

Thus, suppose that $X = (X_1, X_2)'$ has location vector μ and positive definite spread matrix Σ. Then, Since Σ is positive definite, there exists an orthogonal matrix P such that $\Sigma = PAP'$, where A is the diagonal matrix of eigenvalues $\lambda_1 \leq \lambda_2$ of Σ. Set $\Sigma^{1/2} = PA^{1/2}$ so that $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$. Let the bivariate vectors σ_1' and σ_2' denote the rows of $\Sigma^{-1/2'}$. Then let

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \Sigma^{-1/2'}(X - \mu)$$

The joint distribution function of Y_1 and Y_2 is denoted by G and the marginal distribution functions of Y_1 and Y_2 are denoted by G_1 and G_2, respectively. Then the for $\alpha_1, \alpha_2 \geq 0$ and $\alpha_1 + \alpha_2 \leq 1$, the bivariate vector $\eta(\alpha_1, \alpha_2)$ is an (α_1, α_2) the bivariate quantile point if

$$\eta(\alpha_1, \alpha_2) = \mu + \Sigma^{1/2}\xi^*(\alpha_1, \alpha_2)$$

where $\xi^*(\alpha_1, \alpha_2) = \left(G_{12}^{-1}(\alpha_1, \alpha_2), G_2^{-1}(\alpha_1 + \alpha_2)\right)'$ is the (α_1, α_2) th NS bivariate quantile point of $Y = \Sigma^{-1/2'}(X - \mu)$.

In the application to 450K data, no assumptions on the distributions of the methylated and unmethylated values were placed, and as such, the empirical marginal distributions were used instead. As such the sample (α_1, α_2) th NS bivariate quantile point $\hat{\xi}(\alpha_1, \alpha_2)$, is defined as in the original definition of the NS bivariate quantile, but with F_2 and F being replaced by $\hat{F}_2$ and $\hat{F}$, respectively.

2.1.2 Limitations

In the original functional normalization algorithm, $H = 500$ to find 500 equidistant quantiles, ranging from 0 to 1 for each sample and done separately for methylated and unmethylated signals. For the purposes of this research, as well as time constraints, we do not want to significantly deviate from the original functional normalization method. As such, we would want these bivariate quantiles of methylated and unmethylated signals to have non-decreasing monotonicity, as well as interpretability, similar to that of univariate quantiles. Hence, it is appropriate to use the definition of $\xi(\alpha) = \xi\left(\frac{1}{2}\alpha, \frac{1}{2}\alpha\right), 0 \leq \alpha \leq 1$, to find the bivariate quantiles of a single sample. Using the definition seems to perform as described on elliptical-shaped data, as shown in Figure 2.1. In the figure, the chosen bivariate quantile points follow the orientation of the data in space. However, for our purposes, we would like to use a definition of a bivariate quantile that bears some resemblance to the univariate quantile in regards on how the quantile function relates to the density function in the univariate setting. As such, we would hopefully like to see these bivariate quantiles have some relation to their joint CDF. However, examining Figure 2.1, this does not seem to be the case. Notably, in the third plot of Figure 2.1, although the chosen quantile points in the plot follow the definition described by Chen and Welsh (2002), it is fairly obvious that all the points have a probability smaller than or equal to 0.5.

For our purposes, we would like to have a definition of a bivariate quantile that can get the full range of values of methylated and unmethylated signals so we can have values akin

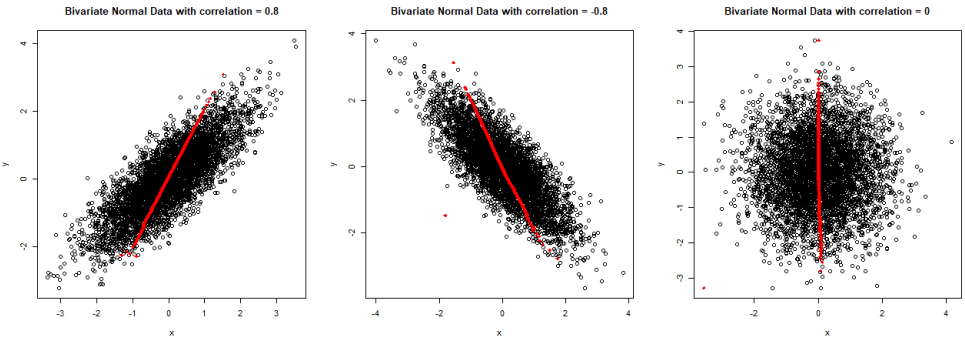

Figure 2.1 α Bivariate Quantiles of Bivariate Normal Distribution with correlation $\rho = 0.8$, -0.8 and 0 respectively, with the sample size for each being $n = 5000$. The red points represent the calculated α bivariate quantile levels as defined by Chen and Welsh (2002), with $\alpha \in [0, 1], a_i < a_j, i = 1, 2...500$

to the univariate quantiles retrieved in the original Functional Normalization algorithm.

2.2 Bivariate quantile by Vineshkumar and Nair (2019)

Upon looking at the limitations of the quantiles proposed by Chen and Welsh (2002), we then looked at the definition of a bivariate quantile proposed by Vineshkumar and Nair (2019). The bivariate quantile function of (X_1, X_2) is defined in this paper as the pair $(Q_1(u_1), Q_{21}(u_2|u_1))$, where $Q_1(u)$ is

$$Q_1(u_1) = \inf\{x_1 | F_1(x_i) \geq u_1\},$$

and Q_{21} is given as

$$Q_{21}(u_2|u_1) = \inf\{x_2 | P(X_2 \leq x_2 | X_1 > Q_1(u_1)) \geq u_2\}$$

where $0 \leq u_i \leq 1, i = 1, 2$

As stated previously, for the sake of non-decreasing monotonicity, as well as interpretability, we define the α^{th} bivariate quantile point of (X_1, X_2) in the same manner as above, but

with $u_1 = u_2 = \alpha$. Also, following in a similar manner to the previous bivariate quantile definition defined by Chen and Welsh, we use the empirical CDFs of X_1 and X_2 as in our case, we make no assumption on the univariate distributions. Using these empirical CDFs should not be an issue since for X_1 and X_2, there are >450,000 values each.

2.2.1 Limitations

One alluring aspect of this definition is that there is no particular restriction placed on the sum of u_1 and u_2, which allows us to theoretically find larger values of the X_2 component of the bivariate quantile. Although this definition of a bivariate quantile does not have the same caveats as the previous definition defined by (Chen and Welsh, 2002), it is not without its own concerns. When finding α^{th} bivariate quantile points of the methylated and unmethylated data of the 450K array, defined as $(X_{1,\alpha}, X_{2,\alpha})$, we then compared α to the empirical joint CDF of the methylated and unmethylated signals, $\hat{F}(x_1, x_2)$, evaluated at $(X_{1,\alpha}, X_{2,\alpha})$. Table 2.1 below shows this comparison for the samples of the *minfiData* dataset at the 10% quantiles for Type I green, Type I red and Type II signals.

From Table 2.1, we see that at each α level, the joint probabilities calculated from these bivariate quantile points greatly differ from the α level value. We would hope that from a definition of a bivariate quantile that α and $\hat{F}_{X_1,X_2}(X_1 = x_{1,\alpha}, X_2 = x_{2,\alpha})$ would be relatively similar, since it seems more natural for a non-decreasing bivariate quantile to have a relation with the related joint CDF.

2.3 Building a new definition for Bivariate Quantiles

We consider the development of an α^{th} bivariate quantile points via the diagonal line of the unit square, $[0, 1]^2$, from point $(0, 0)$ to $(1, 1)$. However, we still would like to take the correlation between methylated and unmethylated signals along with the usage of the joint CDF. Copulas provide a natural bridging of these ideas, since copulas can model dependencies of

α	Type I Green	Type I Red	Type II
0.10	0.0001-0.0004	0.0003-0.0012	0.0031-0.0091
0.20	0.0014-0.0027	0.0019-0.0054	0.0247-0.0467
0.30	0.0065-0.0088	0.0077-0.0164	0.0785-0.1170
0.40	0.0150-0.0207	0.0235-0.0398	0.1564-0.2039
0.50	0.0317-0.0425	0.0565-0.0925	0.2484-0.3038
0.60	0.0676-0.1112	0.1143-0.1816	0.3532-0.4168
0.70	0.1399-0.2371	0.2068-0.3115	0.4690-0.5423
0.80	0.2593-0.4196	0.3428-0.4819	0.5979-0.6746
0.90	0.4825-0.6700	0.5613-0.7178	0.7463-0.8052

Table 2.1 Comparisons Between α and ranges of $F_{X_1,X_2}(X_1 = x_{1,\alpha}, X_2 = x_{2,\alpha})$ for Type I Green, Type I Red and Type II probes signals for all samples in the *MinfiData* dataset

variables, as their joint CDF is defined in terms of the marginal CDFs. We thus consider the use of copulas to come up with a definition of a monotonically non-decreasing bivariate quantile. Before the definition of the copula-based bivariate quantile can be discussed, background information on copulas is needed.

2.3.1 Copulas

Consider a random vector $(X_1, X_2, \ldots, X_d)$. By applying the probability integral transform to each component, the random vector

$$(U_1, U_2, \ldots, U_d) = (F_1(X_1), F_2(X_2), \ldots, F_d(X_d))$$

has uniformly distributed marginals.

The copula of $(X_1, X_2, \ldots, X_d)$ is defined as the joint cumulative distribution function of $(U_1, U_2, \ldots, U_d)$

$$C(u_1, u_2, \ldots, u_d) = \Pr[U_1 \leq u_1, U_2 \leq u_2, \ldots, U_d \leq u_d]$$

The copula C contains all information on the dependence structure between the components of $(X_1, X_2, \ldots, X_d)$ whereas the marginal cumulative distribution functions F_i contain all information on the marginal distributions. Copulas have been used in multiple applications, most notably the Gaussian copula in quantitative finance (Li, 1999). Other uses include medicine, specifically in neuroscience (Onken et al., 2009). However, for our purposes, we would like to have a more concrete relation between copulas and the marginal distributions, as seen in the previous bivariate quantile definitions. Fortunately, this is possible through Sklar's Theorem.

Sklar's Theorem

Sklar's theorem provides the theoretical foundation for the application of copulas (Sklar, 1959). Sklar's theorem states that every multivariate cumulative distribution function

$$F(x_1, \ldots, x_d) = \Pr[X_1 \leq x_1, \ldots, X_d \leq x_d]$$

of a random vector $(X_1, X_2, \ldots, X_d)$ can be expressed in terms of its marginals $F_i(x_i) = \Pr[X_i \leq x_i]$ and a copula C:

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d))$$

The theorem also states that, given F, the copula is unique on $\operatorname{Ran}(F_1) \times \cdots \times \operatorname{Ran}(F_d)$, which is the cartesian product of the ranges of the marginal cdf's. This implies that the copula is unique when the marginals, F_i, are continuous. The converse is also true: given a copula $C : [0,1]^d \to [0,1]$ and marginals $F_i(x)$ then $C(F_1(x_1), \ldots, F_d(x_d))$ defines a d-dimensional cumulative distribution function. From here, we note that a copula, especially copulas stemming from bivariate data, can be modelled by its dependence structure, which is usually a scatterplot modelling of the marginal CDFs against each other.

Name of Copula	Bivariate Copula $C_\theta(u,v)$	parameter θ
Ali–Mikhail–Haq	$\frac{uv}{1-\theta(1-u)(1-v)}$	$\theta \in [-1,1]$
Clayton	$\left[\max\left\{u^{-\theta}+v^{-\theta}-1;0\right\}\right]^{-1/\theta}$	$\theta \in [-1,\infty)\backslash\{0\}$
Frank	$-\frac{1}{\theta}\log\left[1+\frac{(\exp(-\theta u)-1)(\exp(-\theta v)-1)}{\exp(-\theta)-1}\right]$	$\theta \in R\backslash\{0\}$
Gumbel	$\exp\left[-\left((-\log(u))^\theta+(-\log(v))^\theta\right)^{1/\theta}\right]$	$\theta \in [1,\infty)$
Independence	uv	
Joe	$1-\left[(1-u)^\theta+(1-v)^\theta-(1-u)^\theta(1-v)^\theta\right]^{1/\theta}$	$\theta \in [1,\infty)$

Table 2.2 Most Prominent Bivariate Archimedean Copulas (Nelsen, 2011)

Archimedean copulas

Archimedean copulas are copulas that admit an explicit formula. Archimedean copulas allow modeling dependence in arbitrarily high dimensions with only one parameter, governing the strength of dependence.

A copula C is called Archimedean if it has the form:

$$C\left(u_1,\ldots,u_d;\theta\right)=\psi^{[-1]}\left(\psi\left(u_1;\theta\right)+\cdots+\psi\left(u_d;\theta\right);\theta\right)$$

where $\psi:[0,1]\times\Theta\to[0,\infty)$ is a continuous, strictly decreasing and convex function such that $\psi(1;\theta)=0$. θ is a parameter within some parameter space Θ . ψ is the so-called generator function and $\psi^{[-1]}$ is its pseudo-inverse defined by

$$\psi^{[-1]}(t;\theta)=\begin{cases}\psi^{-1}(t;\theta) & \text{if } 0\le t\le\psi(0;\theta)\\ 0 & \text{if } \psi(0;\theta)\le t\le\infty\end{cases}$$

In the literature, there are many bivariate Archimedean copulas that are used to model the dependence of bivariate data. Table 2.2 highlights the most prominent bivariate Archimedean copulas.

For parameter estimation, each copula has statement that connects its parameters to Kendall's Tau (τ) and Spearman's Rho (ρ), both of which are measures of non-parametric

rank correlations, and can be formulated as special cases of a more general correlation coefficient. However, it is noted in literature that the it is more common to use Kendall's Tau over Spearman's Rho for parameter estimation (Wysocki, 2015).

Kendall's Tau (τ)

Consider $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ a set of observations of the joint random variables X and Y respectively. Any pair of observations (x_i, y_i) and (x_j, y_j), where $i < j$, are said to be *concordant* if the ranks for both elements agree; that is, if both $x_i > x_j$ and $y_i > y_j$; or if both $x_i < x_j$ and $y_i < y_j$. Furthermore, they are said to be *discordant* if $x_i > x_j$ and $y_i < y_j$, or if $x_i < x_j$ and $y_i > y_j$. If $x_i = x_j$ or $y_i = y_j$, the pair is neither. Kendall's τ is defined as:

$$\tau = \frac{(\text{ number of concordant pairs }) - (\text{ number of discordant pairs })}{\binom{n}{2}}$$

If the *agreement* between the two rankings, wis perfect (the two rankings are identical), $\tau = 1$. Similarly, if the *disagreement* between the two rankings, is perfect (one ranking is the reverse of the other), $\tau = -1$. On the other hand, if X and Y are independent, then τ is approximately zero, as every pair has equal chance of being concordant or discordant. Kendall's τ's relationship with the parameters of the listed bivariate Archimedean copula are shown in Table 2.3. Note that

$$D_1(\alpha) = \frac{1}{\alpha} \int_0^{\alpha} \frac{t}{e^t - 1} dt$$

and

$$D_J(\theta) = \int_{t=0}^{1} \frac{\left[\ln\left(1 - t^{\theta}\right)^{-}\right]\left(1 - t^{\theta}\right)}{t^{\beta - 1}} dt$$

is the Debye function of the first kind and Debye function respectively.

Using Archimedean copulas gives us an advantage since the majority of them are written in explicit form, especially bivariate Archimedean copulas. By utilizing these bivariate Archimedean copulas with explicit form, as well as everything else mentioned above, we now develop a definition of a bivariate quantile.

Name of Copula	Relationship to τ ($\tau =$)
Ali–Mikhail–Haq	$\frac{3\theta-2}{3\theta} - \frac{2(1-\theta)^2\ln(1-\theta)}{3\theta^2}$
Clayton	$\frac{\theta}{\theta+2}$
Frank	$1 - \frac{4}{\theta}\left[1 - D_1(\theta)\right]$
Gumbel	$\frac{\theta-1}{\theta}$
Joe	$1 + \frac{4}{\theta}D_J(\theta)$

Table 2.3 Kendall's τ relationship to the parameters of the mentioned bivariate copulas. Table taken from Karakas et al. (2017)

Definition of the copula-based bivariate quantile

Let $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ be bivariate observations. Then, from Sklar's Theorem, we have the following:

$$F(x, y) = C(F_1(x), F_2(y))$$

where $F(x, y)$ is the joint distribution of X and Y, $F_1(x)$ and $F_2(y)$ are the marginal distributions of X and Y respectively, and C is a bivariate copula, ideally with an explicit form. Now, let $F_1(x) = u$ and let $F_2(y) = g(u)$, where $g(u)$ is a monotonically increasing, continuous function, with the restrictions $u = 1 \implies g(u) = 1$ and $u = 0 \implies g(u) = 0$. Then we have

$$C(F_1(x), F_2(y)) = C(u, g(u))$$

Now, set $F(x, y)$ to a value α, where $\alpha \in [0, 1]$. The α^{th} bivariate quantile point is thus the solutions to the equation

$$\alpha = C(u, g(u))$$

which is thus the point $(F_1^{-1}(u), F_2^{-1}(g(u)))$, where

$$F_1^{-1}(u) = \inf\{x | F_1(x) \geq u\}$$

and

$$F_2^{-1}(g(u)) = \inf\{x | F_2(x) \geq g(u)\}$$

where $u \in [0, 1]$.

With this definition of a bivariate quantile, we hope to overcome the limitations that were observed with the previous definitions.

Chapter Three

Bivariate Functional Normalization Methodology

Using the copula-based bivariate quantile defined in Chapter 2, we now discuss how we build a bivariate version of the functional normalization algorithm. In the original functional normalization algorithm from Fortin et al. (2014), they change the quantiles of the methylated and unmethylated signals via the use of a linear fitting for each signal separately by univariate principal component regression. However, to take the correlation into account when normalizing the distributions of the methylated and unmethylated signals, we propose the use of partial least squares modelling instead of the regular principal component regression done in the original method. This allows for a bivariate response linear model rather than two separate univariate response.

3.1 Partial Least Squares Regression (PLSr) - Background

Developed by Wold (1966), partial least squares regression (PLSr) is a generalization of multiple linear regression (MLR) and is similar to principal component regression (PCR). The method is used to analyze data with strongly collinear, correlated, noisy, and numerous predictor X variables, and it can also simultaneously model several response variables Y

taking into account correlations between them. The main focus of PLS is to extract the *latent variables* (LVs) accounting for as much of the variation between the covariates as possible while modeling the responses well. This is achieved indirectly by extracting the latent variables T and U, from sampled covariates and responses, respectively. In general:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathcal{E}_X$$
$$\mathbf{Y} = \mathbf{U}\mathbf{Q}^T + \mathcal{E}_Y$$

where $\mathbf{X}$ is an $n \times k$ matrix of predictors, $\mathbf{Y}$ is an $n \times m$ matrix of responses; $\mathbf{T}$ and $\mathbf{U}$ are $n \times a$ matrices that are, respectively, projections of $\mathbf{X}$ (the $\mathbf{X}$ score matrix) and projections of $\mathbf{Y}$ (the $\mathbf{Y}$ scores). Furthermore, $\mathbf{P}$ and $\mathbf{Q}$ are, respectively, $k \times a$ and $m \times a$ orthogonal loading matrices, and matrices $\mathcal{E}_X$ and $\mathcal{E}_Y$ are the error terms, assumed to be independent and identically distributed random normal variables. Note that for PLS $\mathbf{X}$ is approximated by the first a components chosen and takes care of measurement error through $\mathcal{E}_X$. The decompositions of $\mathbf{X}$ and $\mathbf{Y}$ are done to maximize the covariance between $\mathbf{T}$ and $\mathbf{U}$. The extracted $\mathbf{X}$ scores, $\mathbf{T}$ are then used to predict the Y scores, $\mathbf{U}$. The predicted $\mathbf{Y}$ scores are then used to construct predictions for the responses. Formally, PLSR methods solve the following maximization problem:

$$\max_{\mathbf{w},\mathbf{c}} \{\mathrm{cov}\,(\mathbf{X}_i\mathbf{w}_i, \mathbf{Y}_i\mathbf{c}_i)\}$$

subject to

$$\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i = 1, \mathbf{c}_i^{\mathrm{T}}\mathbf{c}_i = 1$$

and

$$\mathbf{t}_i^T\mathbf{t}_j = 0, \mathbf{u}_i^T\mathbf{u}_j = 0$$

for all $i \neq j, i = 1, 2, ...A$.

where $\mathbf{w}_i, \mathbf{c}_i$ and $\mathbf{t}_i, \mathbf{u}_i$ are weight vectors and score vectors for $\mathbf{X}$ and $\mathbf{Y}$, respectively. A simple schematic of PLS can be shown in Figure 3.1.

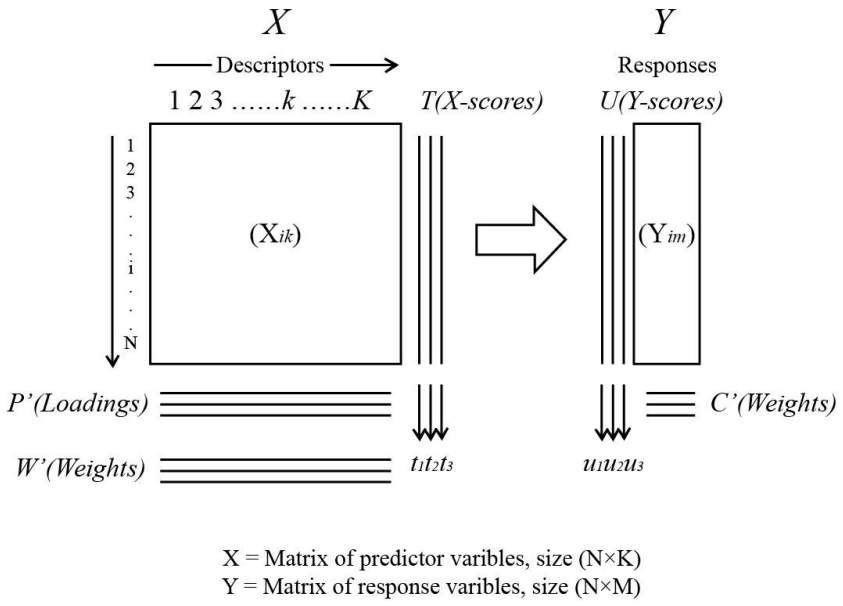

Figure 3.1 A General Schematic of Partial Least Squares from Filzmoser, Serneels, et al. (2009)

For PLSR, the components are obtained iteratively. It starts with the singular value decomposition of the crossproduct matrix $\mathbf{S} = \mathbf{X}^T\mathbf{Y}$, thus including information on variation in both $\mathbf{X}$ and $\mathbf{Y}$, and on the correlation between them. The first left and right singular vectors, $\mathbf{w}$ and $\mathbf{c}$, are used as weight vectors for $\mathbf{X}$ and $\mathbf{Y}$, respectively, to obtain scores $\mathbf{t}$ and $\mathbf{u}$ via the following equations:

$$\mathbf{t} = \mathbf{X}\mathbf{w} = \mathbf{E}\mathbf{w}$$

$$\mathbf{u} = \mathbf{Y}\mathbf{c} = \mathbf{F}\mathbf{c}$$

Here, where $\mathbf{E}$ and $\mathbf{F}$ are initialised as $\mathbf{X}$ and $\mathbf{Y}$, respectively. The X scores $\mathbf{t}$ are often normalised as $\mathbf{t} = \mathbf{t}/\sqrt{\mathbf{t}^T\mathbf{t}}$. Next, X and Y loadings are obtained by the following equations:

$$\mathbf{p} = \mathbf{E}^T\mathbf{t}$$

$$\mathbf{q} = \mathbf{F}^T\mathbf{t}$$

The data matrices are then "deflated"; the information related to this latent variable, in the form of the outer products $\mathbf{t}\mathbf{p}^T$ and $\mathbf{t}\mathbf{q}^T$, is subtracted from the (current) data matrices $\mathbf{E}$ and $\mathbf{F}$.

$$\mathbf{E}_{n+1} = \mathbf{E}_n - \mathbf{t}\mathbf{p}^T$$
$$\mathbf{F}_{n+1} = \mathbf{F}_n - \mathbf{t}\mathbf{q}^T$$

where the dimensions of $\mathbf{t}, \mathbf{p}$ and $\mathbf{q}$ are $n \times 1$, $k \times 1$ and $m \times 1$ respectively. The estimation of the next component then can start from the SVD of the crossproduct matrix $\mathbf{E}_{n+1}^T \mathbf{F}_{n+1}$. After every iteration, vectors $\mathbf{w}, \mathbf{t}, \mathbf{p}$ and $\mathbf{q}$ are saved as columns in matrices $\mathbf{W}, \mathbf{T}, \mathbf{P}$ and $\mathbf{Q}$, respectively. The relationship between the weights to the original $\mathbf{X}$ matrix is given by

$$\mathbf{T} = \mathbf{X}\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1} = \mathbf{X}\mathbf{R}$$

where $\mathbf{R} = \mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}$. Using this relationship, the regression coefficients, β, can be calculated, and converted to the realm of the original variables by pre-multiplying matrix $\mathbf{R}$. We thus get the following equation :

$$\beta = \mathbf{R}\left(\mathbf{T}^T\mathbf{T}\right)^{-1}\mathbf{T}^T\mathbf{Y} = \mathbf{R}\mathbf{T}^T\mathbf{Y} = \mathbf{R}\mathbf{Q}^T$$

where here only the first l components are used. The number of optimal components are usually determined by cross-validation.

There are many variants of the PLS algorithm for estimating the factor and loading matrices $\mathbf{T}, \mathbf{U}, \mathbf{P}$ and $\mathbf{Q}$. The most prominent algorithms used are the kernel algorithm by Lindgren et al. (1993), the classic orthogonal scores algorithm (known as the "Nonlinear Iterative Partial Least Squares" or NIPALS algorithm) that was developed along with PLS regression and the SIMPLS algorithm, developed by (Jong, 1993). Due to producing the same results as the original NIPALS algorithm and being faster on average than the other mentioned methods, the kernel algorithm is chosen for our needs. Briefly, the kernel algorithm uses the results from the original NIPALS algorithm along with the calculation of $\mathbf{w}, \mathbf{c}, \mathbf{t}$ and $\mathbf{u}$ using eigenvalue-eigenvector equations described in Jöreskog and Wold (1982).

3.2 Bivariate Quantiles for 450K array data

As in the original functional normalization algorithm for 450K array data, we attempt to first retrieve 500 equidistant quantiles from each sample. However, instead of extracting the univariate quantiles from methylated and unmethylated signals separately, we treat the methylated and unmethylated signals as a bivariate data set, with $\mathbf{X}_i$ and $\mathbf{Y}_i$ being the methylated and unmethylated data values from sample i respectively, and retrieve the bivariate quantiles from each sample.

However, before the bivariate quantiles are retrieved from each sample, an appropriate copula must be chosen. Again, we restricted ourselves to bivariate Archimedean copulas for reasons of computational complexity and time constraints. However, of the bivariate Archimedean copulas listed in Table 2.2, only the Ali–Mikhail–Haq (AMH) copula and the Frank copula can model both positive and negative dependence of $\mathbf{X}_i$ and $\mathbf{Y}_i$, whereas the other listed copulas can only model positive dependence (Ruppert and Matteson, 2015). Investigating the AMH copula further reveals that when using this copula, Kendall's τ is bounded. In particular, from Table 2.1 and Table 2.2, the parameter θ for the AMH copula lies in the interval $[-1, 1]$ and its relationship to τ is the formula

$$\tau(\theta) = \frac{3\theta - 2}{3\theta} - \frac{2(1-\theta)^2 \ln(1-\theta)}{3\theta^2}$$

From the formula, when $\theta = -1$, $\tau(\theta) = \frac{5-8\ln 2}{3}$ and when $\theta = 1$, $\tau(\theta) = \frac{1}{3}$. Hence, this copula is not suitable for obtaining bivariate quantiles of methylation data. On the other hand, the Frank copula does not have this restriction. In particular, for the Frank copula, $\tau(\theta)$ is the following:

$$1 - \frac{4}{\theta}\left[1 - D_1(\theta)\right]$$

where D_1 is the Debye function of the first kind. Expanding out the equation gives

$$\tau(\theta) = 1 - \frac{4}{\theta} + \frac{4}{\theta^2}\int_0^\theta \frac{t}{e^t - 1} dt$$

Now, with the use of numerical integration, as $\theta \to +\infty$ the integral $\int_0^\theta \frac{t}{e^t-1}dt \approx 1.644934$, making $\lim_{\theta\to+\infty} \tau(\theta) = 1$. In a similar manner, as $\theta \to -\infty$,

$$\lim_{\theta\to-\infty} \int_0^\theta \frac{t}{e^t-1}dt = \lim_{\theta\to-\infty} \int_\theta^0 \frac{-t}{e^t-1}dt \leq \lim_{\theta\to-\infty} \int_\theta^0 \frac{-t}{-1}dt = -\infty$$

From the above, using L'Hospital's Rule and the Fundamental Theory of Calculus, $\lim_{\theta\to-\infty} \frac{4}{\theta^2} \int_0^\theta \frac{t}{e^t-1}dt = -2$, which then implies $\lim_{\theta\to-\infty} \tau(\theta) = -1$. Thus, $\tau \in (-1, 1)\backslash\{0\}$ for the Frank Copula. As such, with $\tau's$ flexibility in range for the Frank copula, as well as being able to model positive and negative correlation, the Frank copula is a suitable candidate to obtain the bivariate quantiles of the methylated and unmethylated signals.

3.3 Building the Algorithm

For implementing the algorithm, *R* 4.0.0 was used (R Core Team, 2020).

In terms of building a bivariate version of the functional normalization algorithm, we first acknowledge that we would want to build it in a very similar manner to the original functional normalization method. As such, we reuse a lot of the original functions from the functional normalization algorithm (Fortin et al., 2014).

In the original paper, it has been suggested to use data that have already been background corrected using the normal-exponential out-of-band (noob) method; a background correction method with dye-bias normalization for Illumina Infinium methylation arrays (Triche et al., 2013). Briefly, the noob process is as follows. Let $\mathrm{X_b} \sim \mathrm{N}(\mu, \sigma^2)$ and $\mathrm{X_S} \sim \mathrm{Exp}(\gamma)$, and the observed foreground intensity $\mathrm{X_f} = \mathrm{X_S} + \mathrm{X_b}$. Parameters are estimated from the background distribution using all control probes, and the signal parameter γ from the observed foreground intensities with the background mean subtracted $(X_f - \mu)$ The conditional expectation of the signal given the observed foreground and background is computed by,

$$E[X_s|X_f] = \mu_{sf} + \sigma^2 \frac{\phi(0; \mu_{sf}, \sigma^2)}{1 - \Phi(0; \mu_{sf}, \sigma^2)}$$

where $\mu_{s,f} = x_f - \mu - \sigma^2/\gamma, \phi(\cdot)$ the standard normal density and Φ the cumulative normal distribution. Due to getting slightly better results in the original functional normalization method demonstrated by Fortin et al. (2014), we include the use of background correction or background and dye bias correction in our algorithm via the `preprocessNoob` function in the *minfi* package.

We also use the same the 42 summary control measurements mentioned in the original functional normalization method as our covariates, since they have been shown to be surrogate for the unwanted, non-biological variation that we wish to regress out.

To get the bivariate quantiles via the Frank copula, the Kendall's τ correlation between the methylated and unmethylated signals of the sample in question must be calculated. However, it has been stated in the R documentation that in base R, computation of Kendall's τ using the `cor` function is very slow, as its implementation was meant for smaller datasets (R Core Team, 2020). We overcome this issue by using the `cor.fk` function in the *pcaPP* package in *R* (Filzmoser, Fritz, et al., 2018). Once the Kendall's τ correlation is obtained, the parameter θ needs to found. Using the equation in Table 2.1 will allow one to find the parameter θ. However, it is apparent from the equation that there is not a closed form expression for θ and hence, numerical inversion methods must be done to solve for θ. To find θ_{Frank}, we use the `BiCopTau2Par` function found in the *VineCopula* package (Nagler et al., 2019). After obtaining θ, with the use of our definition of a bivariate quantile, we obtain the bivariate quantiles via probe type. To keep things simple and understandable, we used the diagonal of the copula, $C(F_X(x), F_Y(y))$, to gather our two sets of bivariate quantiles. In other words, from our definition of a our copula-based bivariate quantile, we use $F_Y(y) = g(F_X(x)) = F_X(x) = g(u) = u$.

For $g(u) = u$, we get the following equation

$$\alpha_j = -\frac{1}{\theta} \log \left[1 + \frac{(\exp(-\theta u) - 1)^2}{\exp(-\theta) - 1} \right]$$

where α_j is one of the 500 probabilities that are initially chosen. Formally, $\alpha_j \in [0, 1]$, where

$j = 1, 2, 3...500$, $j < k \implies \alpha_j < \alpha_k$. Rearranging the above equation gives us

$$\exp(-2\theta u) - 2\exp(-\theta u) + \exp(-\theta) - \exp(-\theta(\alpha_j + 1)) + \exp(-\theta\alpha_j) = 0$$

Substituting $\exp(-\theta) = w$ then gives us

$$w^{2u} - 2w^u + w - w^{\alpha_j+1} + w^{\alpha_j} = 0$$

Doing another substitution, $s = w^u$, we get the equation:

$$s^2 - 2s + q - q^{\alpha_j+1} + q^{\alpha_j} = 0$$

which makes it apparent that the variable u can be solved via quadratic equation. We thus get

$$u(\alpha_j) = \frac{\ln(1 \pm \sqrt{-e^{-\theta} - e^{-\theta\alpha_j} + e^{-\theta(\alpha_j+1)} + 1})}{-\theta}$$

Through empirical analysis via analyzing the functions behavior on the *minfiData* dataset, as well as looking at the denominator of the equation, when θ is negative, the positive root, to get positive values of u. Similarly, when θ is positive, we need the negative root to obtain positive values of u.

Hence, through our analysis, for $u \in [0, 1]$,

$$u(\alpha_j) = \frac{\ln(1 - \text{sign}(\theta)\sqrt{-e^{-\theta} - e^{-\theta\alpha_j} + e^{-\theta(\alpha_j+1)} + 1})}{-\theta}$$

After obtaining the bivariate quantile for each sample, we now mimic the original functional normalization method by regressing out the unwanted variation through known covariates that can measure unwanted variation. However, unlike the original method where they used principal component regression to regress out the unwanted variation, we use PLS regression instead.

In the original Functional Normalization algorithm, unwanted variation was removed via the following steps. First, for a given probe type, a matrix of quantiles of signal intensities (either methylated or unmethylated), $\mathbf{H}$, is obtained, where $\mathbf{H}$ is $500 \times N$, where N is

the number of samples and the 500 columns correspond to one of the equidistant quantile values, $\alpha_1 ... \alpha_{500}$. Then, the algorithm fixes potential problems with extreme quantile values by setting the rows of $\mathbf{H}$ where $\alpha_1 = 0$ to a value of zero, as well as setting the largest methylated value (or the α_{500} quantiles) of each sample to be greater than the previous quantile (α_{499}) value by a value of 1000.

Then, the across-samples mean is obtained for each quantile level, denoted by $\bar{h}_{i\bullet} = \frac{1}{N}\sum_{j=1}^{N} h_{i,j}, i = 1 \dots 500$ From here on, a matrix of 'residuals', named $\mathbf{H}^*$ is created via the equation

$$\mathbf{H}^* = \mathbf{H} - \bar{\mathbf{h}}_{\mathbf{N}\bullet}\mathbf{J}_{\mathbf{1}\times\mathbf{N}}$$

where $\bar{\mathbf{h}}_{\mathbf{N}\bullet}^{\mathrm{T}} = \left[\bar{h}_{1\bullet}, ..., \bar{h}_{500\bullet}\right]$ with dimension 1×500 and $\mathbf{J}_{\mathbf{1}\times\mathbf{N}}$ is a $1 \times N$ matrix (row vector) of all ones. In other words, the values at a given quantile level are centered via their corresponding mean. The rows of this new 'residual' matrix, are then regressed separately via principal component regression, which thus regresses out the unwanted variation for each quantile. The residuals of the linear fitting are then placed in a new matrix, $\mathbf{H}^{\mathrm{new}}$ and the previous row means, $\bar{h}_{i\bullet}$, are added to this new matrix by the corresponding row, or simply

$$\mathbf{H}^{\mathrm{new}} = \mathbf{H}^* + \bar{\mathbf{h}}_{\mathbf{N}\bullet}\mathbf{J}_{\mathbf{1}\times\mathbf{N}}$$

This is done so as to preserve the inherent noise and biological differences within the quantile functions. The quantiles are then regularized to ensure a monotonically increasing and non-negative quantile function. α_i is set to the cumulative maximum of $\alpha_1, , \alpha_i$.

Linear interpolation is then used between quantile values for each sample, creating a new target quantile distribution. The signal intensities of each sample are then normalized via quantile normalization using this new target distribution as the reference distribution for quantile normalization. This process is repeated separately by probe type and separately for methylated and unmethylated signals. The sex chromosomes are normalized separately from the autosomal signals and for these males and females are normalized separately if there are sufficient samples from each gender to do this reliably. For the X chromosome, functional

normalization is used, and for the Y chromosome, quantile normalization is used.

For our study, we use a similar process to the original Functional Normalization, but instead with the use of PLS regression. We first note that we also fix potential problems with extreme quantile values the exact way as it was done in the original algorithm. As such, for the 500^{th} bivariate quantile for each sample to be greater than the previous quantile (α_{499}) value by a value of 1000, or $(q_{m,500}, q_{u,500}) = (q_{m,499}, q_{u,499}) + (1000, 1000)$. For a particular α^{th} bivariate quantile, we firstly create a matrix $\mathbf{Y}_\alpha$ with dimensions $N \times 2$, with each row being the n^{th} sample's α^{th} bivariate quantile, with the first column being methylated signals and the second column being unmethylated signals.

$$\mathbf{Y}_\alpha = \begin{bmatrix} \hat{F}^{-1}_{\text{Meth}_1}(u(\alpha_1)) & \hat{F}^{-1}_{\text{Unmeth}_1}(u(\alpha_1)) \\ \hat{F}^{-1}_{\text{Meth}_2}(u(\alpha_2)) & \hat{F}^{-1}_{\text{Unmeth}_2}(u(\alpha_2)) \\ \vdots & \vdots \\ \hat{F}^{-1}_{\text{Meth}_N}(u(\alpha_N)) & \hat{F}^{-1}_{\text{Unmeth}_N}(u(\alpha_N)) \end{bmatrix}$$

Relatedly, akin to functional normalization, we subtract the column means from values in the corresponding column, which are the averages of the methylated and unmethylated signals across samples at a given quantile level, α, denoted by $\bar{y}_{\cdot 1} = \frac{1}{N}\sum_{i=1}^{N} \hat{F}^{-1}_{\text{Meth}_i}(u_{\alpha,i})$ and $\bar{y}_{\cdot 2} = \frac{1}{N}\sum_{i=1}^{N} \hat{F}^{-1}_{\text{Unmeth}_i}(u_{\alpha,i})$,creating a 'residual' matrix, denoted by $\mathbf{Y}^*_\alpha$

$$\mathbf{Y^*}_\alpha = \begin{bmatrix} \hat{F}^{-1}_{\text{Meth}_1}(u(\alpha_1)) - \bar{\mathbf{y}}_{\cdot\mathbf{1}} & \hat{F}^{-1}_{\text{Unmeth}_1}(u(\alpha_1)) - \bar{\mathbf{y}}_{\cdot\mathbf{2}} \\ \hat{F}^{-1}_{\text{Meth}_2}(u(\alpha_2)) - \bar{\mathbf{y}}_{\cdot\mathbf{1}} & \hat{F}^{-1}_{\text{Unmeth}_2}(u(\alpha_2)) - \bar{\mathbf{y}}_{\cdot\mathbf{2}} \\ \vdots & \vdots \\ \hat{F}^{-1}_{\text{Meth}_N}(u(\alpha_N)) - \bar{\mathbf{y}}_{\cdot\mathbf{1}} & \hat{F}^{-1}_{\text{Unmeth}_N}(u(\alpha_N)) - \bar{\mathbf{y}}_{\cdot\mathbf{2}} \end{bmatrix}$$

We then regress on $\mathbf{Y}^*_\alpha$ with the control probe summary matrix as our covariates via PLS regression via the `plsr` function in the *pls* package (Mevik et al., 2019). From the fitting, we obtain the $N \times 2$ residual matrix, $\mathbf{E}_{\mathbf{Y}^*_\alpha}$. The previous column means are then added to the residuals from the PLS regression fitting so as to preserve inherent noise and differences

in the quantile functions, again, similar to the original algorithm, hence creating these new fitted quantiles, $\mathbf{Y}_{\alpha}^{\text{new}}$.

$$\mathbf{Y}_{\alpha}^{\text{new}} = \mathbf{E}_{\mathbf{Y}_{\alpha}^{*}} + \mathbf{J}_{N\times 1}\begin{bmatrix}\bar{y}_{\cdot 1} & \bar{y}_{\cdot 2}\end{bmatrix}$$

where $\mathbf{J}_{N\times 1}$ is a $N \times 1$ matrix (column vector) of all ones.

The process is repeated for all 500 α bivariate quantiles. These new quantiles are then regularized to ensure a monotonically increasing and non-negative quantile function in the same fashion as done in Functional Normalization, but with our bivariate quantiles instead.

We then use the `.normalizeMatrix` function in the *minfi* package to separately normalize the methylated and unmethylated signals (Fortin et al., 2014). This process is repeated for all probe types. For the purposes of this thesis we only applied our method to the 22 autosomal chromosomes and not the sex chromosomes.

In terms of the choosing the number of components that one should use for the PLS regression fitting, we need to consider the varying sample sizes, since as N increases, so too does the number of possible components that can be chosen, up to a maximum of 42 components. However, we would hope that using a small number of components in the PLS regression fitting will capture the variation explained by non-biological factors and preserve the inherent biological variation within a 450k array dataset. As such, for research purposes, we look at the number of components $m = 1, 2, 3, 4$ when running our method on chosen datasets.

We also note that due to the nature of doing PLS regression on each of these 500 quantiles gives us different X scores for each regression and thereby giving us different regression coefficients at each quantile. Table 3.1 demonstrates this fact when the number of components, $m = 3$ on a small dataset in the *minfiData* package, which is one of the datasets in which we applied our algorithm. More information on the datasets used will be given in Section 3.3.1.

In regards to this observation, although the regressions done at each quantile level are

	10% quantile			50% quantile			90% quantile		
Component	1	2	3	1	2	3	1	2	3
Sample 1	-5.8023145	3.4642573	-1.4369603	-8.4256324	0.2360034	0.3656026	8.3658829	-0.2202471	0.2791163
Sample 2	1.5235729	0.7891839	-1.3192991	0.1660735	2.6473295	1.4078039	-0.2488104	2.2306782	0.6004411
Sample 3	2.1216217	-1.3350432	-5.1282881	1.6584443	1.5399841	-4.1920118	-1.702056	1.3763422	-4.3431328
Sample 4	0.4501457	2.1691718	4.1739757	0.4360793	-3.9013887	-1.1998152	0.1686988	-0.8399016	0.8058703
Sample 5	4.5932469	0.8115803	1.7893483	4.0185043	0.9703661	1.3648776	-3.7310464	2.7135289	2.4275566
Sample 6	-2.8862727	-5.8991501	1.9212236	2.146531	-1.4922944	2.2535429	-2.8526689	-5.2604006	0.2301484

Table 3.1 The x-scores of the PLS regression fitting of the 10%, 50% and 90% quantiles for Type I Green probes, with the number of components,$m = 3$ on the $minfiData$ dataset

essentially using different regression coefficients, doing so may allow the possibility that batch effects may have more pronounced effects at particular levels of methylated and unmethylated signals.

3.3.1 Applying the Methodology to Datasets

As shown throughout this thesis, we have built our methodology using the *minfiData* dataset as a reference. To see how methodology works on other larger datasets, we have chosen Methylation data obtained from the European Bioinfomatics Institute website (https://www.ebi.ac.uk/). The particular dataset we used was 'E-GEOD-68777 - Association of DNA Methylation with Acute Mania and Inflammatory Markers' created by Sabunciyan et al. (2015). For ease of reference, we will refer to the latter as the *Mania* dataset.

MinfiData dataset

The `minfiData` dataset is a R package containing a dataset that is used in tutorials for applying pipelines to 450K data in R (Hansen et al., 2019). This particular dataset stems from 6 samples across 2 groups from 450K methylation arrays, with half of the group having cancer and the other half as control. We compare results from our method to the results from Functional Normalization and to the raw dataset where no preprocessing is done (no

noob and no normalization). Figures 3.2, 3.3 and 3.4 show the results from no processing, Functional Normalization and the developed normalization method respectively.

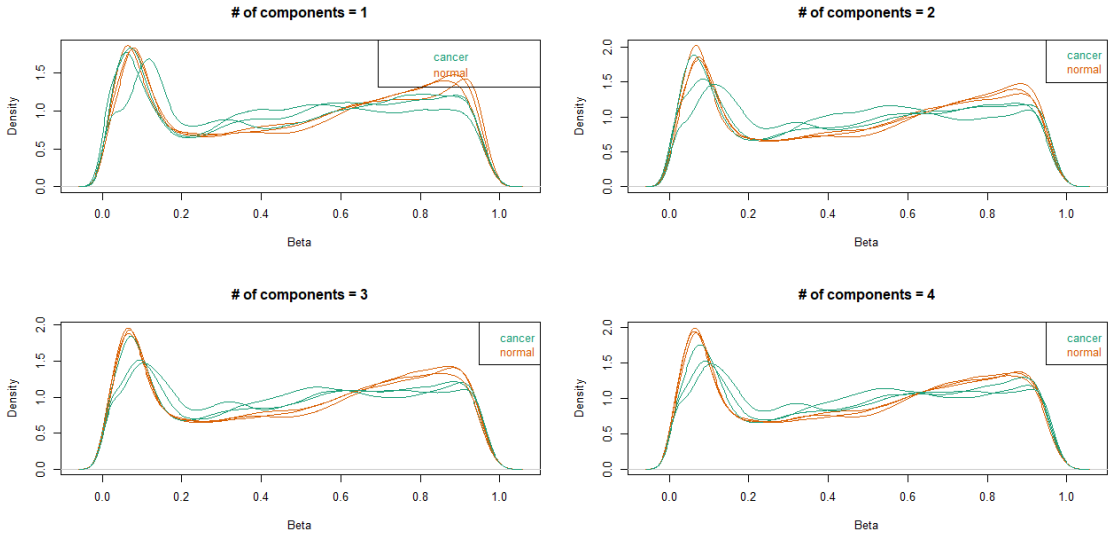

Figure 3.4 Beta Value Density Plots for the *minfiData* dataset, with number of components chosen m =1,2,3,4

Comparing our results to the results obtained from Functional Normalization and the unprocessed data, we seem to be taking the correlation between the methylated and un-

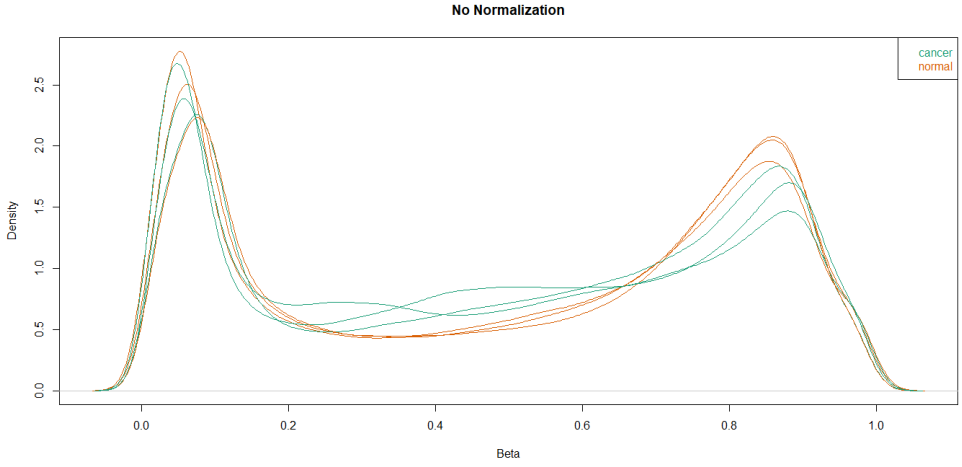

Figure 3.2 Beta Value Density Plots for the *minfiData* dataset with no normalization done

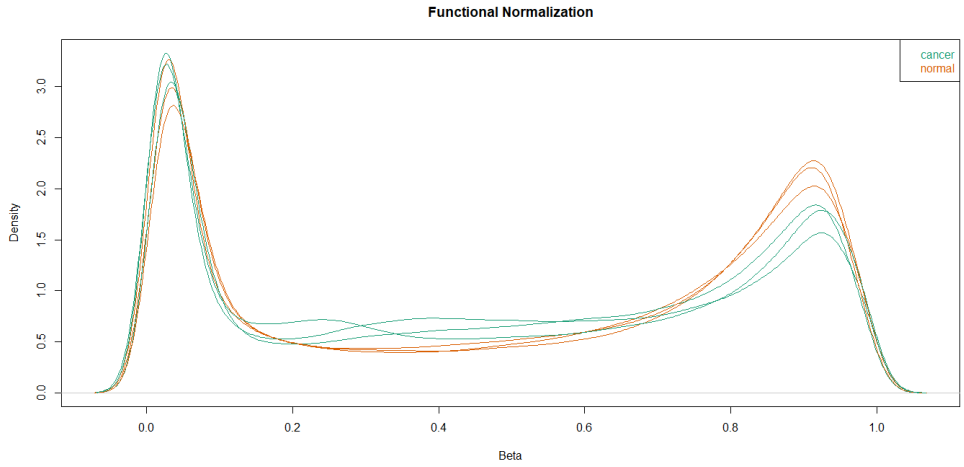

Figure 3.3 Beta Value Density Plots for the *minfiData* dataset with Functional Normalization and dye bias correction done

methylated signals into account and thus lowering the peaks at the extremes of the data. Furthermore, we seem to be keeping the integrity of the distributions separable via group type (cancer vs. control). Examining Figure 3.4 more closely, we see that as the number of m components increases, the closer the distributions come together. Furthermore, it seems that the distributions come closer together via group type. We also see that peak at right side of the Beta-value distribution at $m = 3, 4$ seems to nearly flatten out when compared to Figures 3.2 and 3.3. As such, this new method seems very promising. However, problems arise when looking at other datasets, specifically datasets with higher peaks at the extremes, as well as with larger samples.

Mania Dataset

The *Mania* dataset is a genome wide DNA methylation profiling of serum samples from 20 patients hospitalized with acute mania and 20 unaffected individuals using the Illumina 450K methylation arrays (Sabunciyan et al., 2015). In a similar fashion to the $minfiData$ dataset, we compare our results from our method to the results from Functional Normalization and to the raw dataset where no preprocessing is done.

We now see a problem with our method. Although we are taking the correlation between the methylated and unmethylated signals into account, the lowering in extremely large peaks at the extremes of the beta distribution causes the distribution to essentially be 'squished', causing this multimodal distribution to form and thus most likely adding more variation to the data. Upon further inspection of our method, we note that the use of the built in function in the Functional Normalization algorithm, `.normalizeMatrix`, may not be a good idea. This is due to the function assuming that there are equal spacings between the quantiles at the univariate level. Although we have made our quantiles to be monotonically increasing as such with Functional Normalization, the equal spacings of our bivariate quantiles do not translate well in the univariate sense. Thus, the linear interpolation done by `.normalizeMatrix` does not work well with our bivariate quantiles. It is at this point where we try to revise our original method in an attempt to improve the performance.

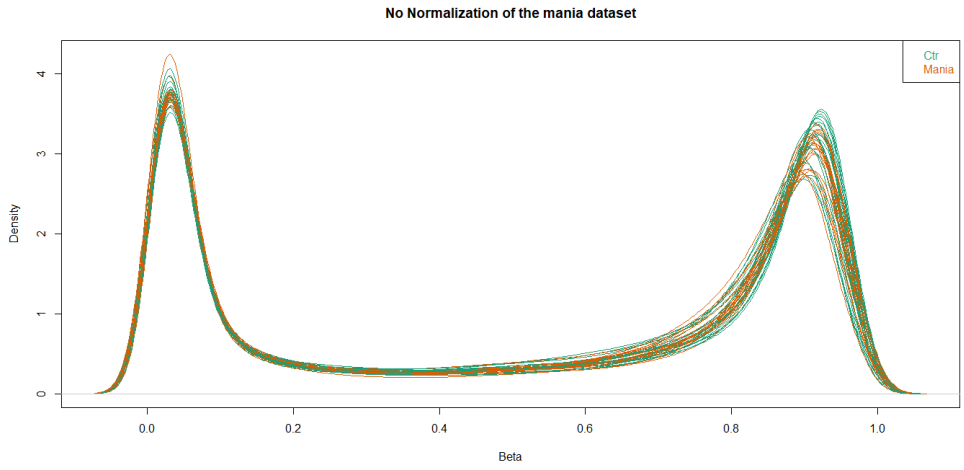

Figure 3.5 Beta Value Density Plots for the *Mania* dataset with no normalization done

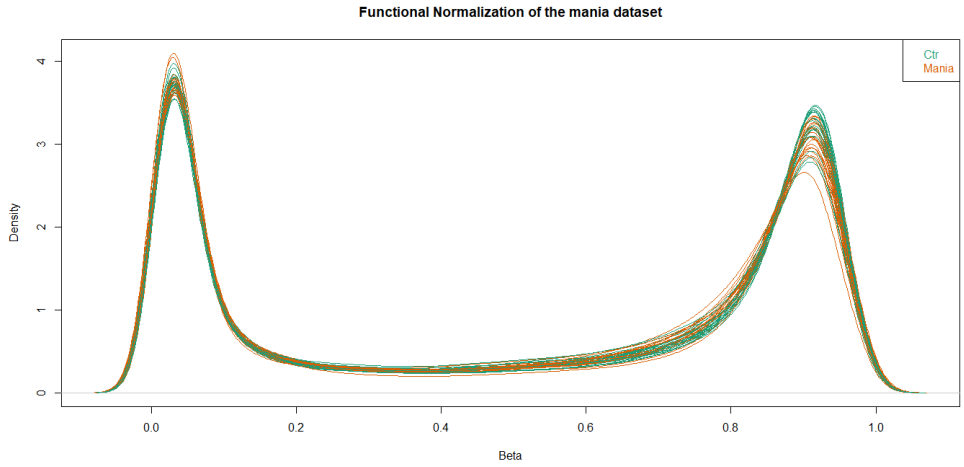

Figure 3.6 Beta Value Density Plots for the *Mania* dataset with Functional Normalization and dye bias correction done

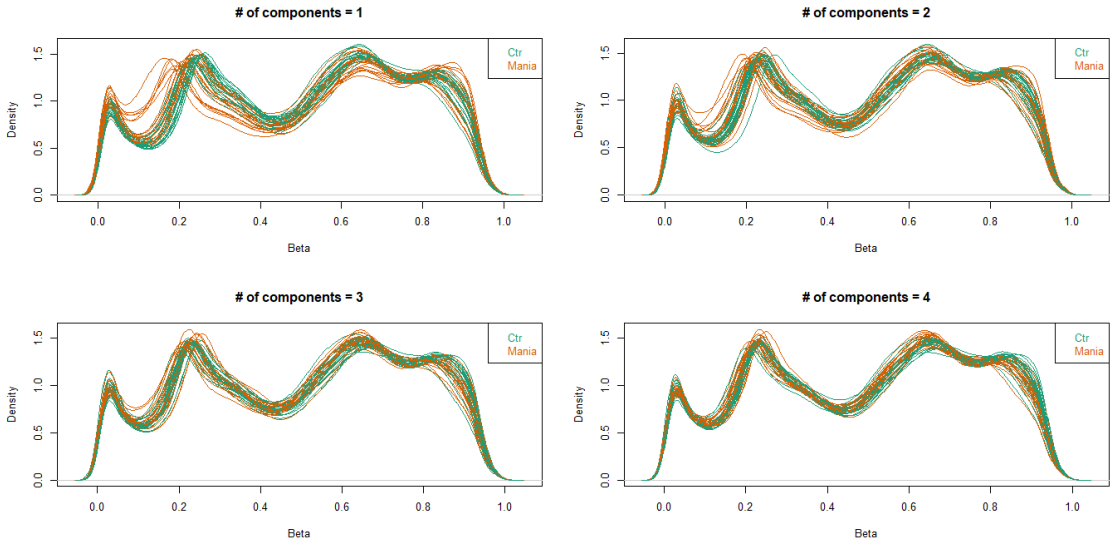

Figure 3.7 Beta Value Density Plots for the *Mania* dataset, with number of components chosen m =1,2,3,4

Chapter Four

Variations of Fitting for Bivariate Functional Normalization

We have tried to alleviate the issues with our proposed method shown in the previous chapter with methods that will be discussed throughout this chapter. We use the *Mania* dataset to see how well these methods work. Also, when doing the fitting for all methods using PLSr, we always used $m = 3$, because this value seems to make the beta-value distributions more similar to each other compared to $m = 1, 2$ and does not show much difference to when $m = 4$. Figures 4.1 and 4.2 show the beta-value densities via probe type of the *Mania* dataset, with no preprocessing and the original Functional Normalization with noob correction respectively.

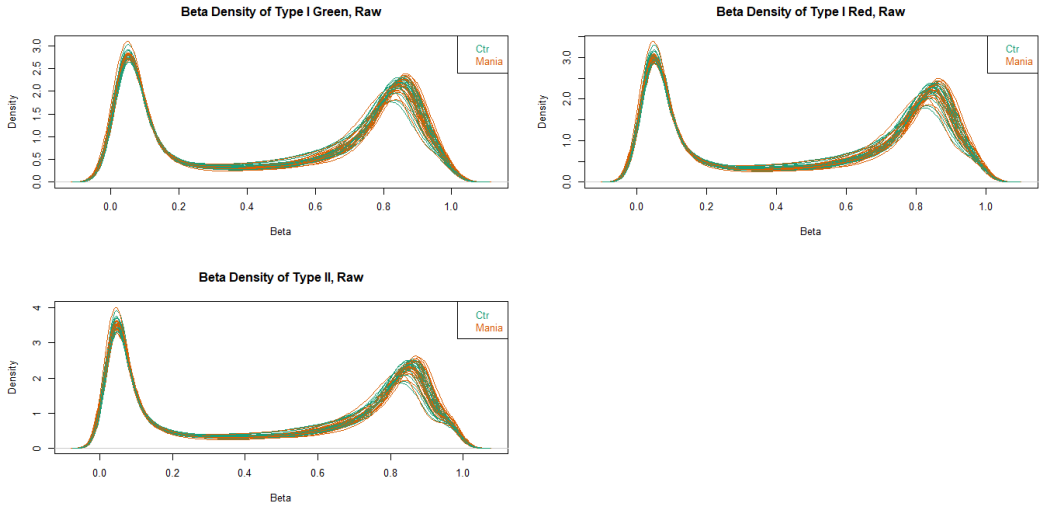

Figure 4.1 Beta Value Density Plots for the *Mania* dataset via probe type, with no preprocessing.

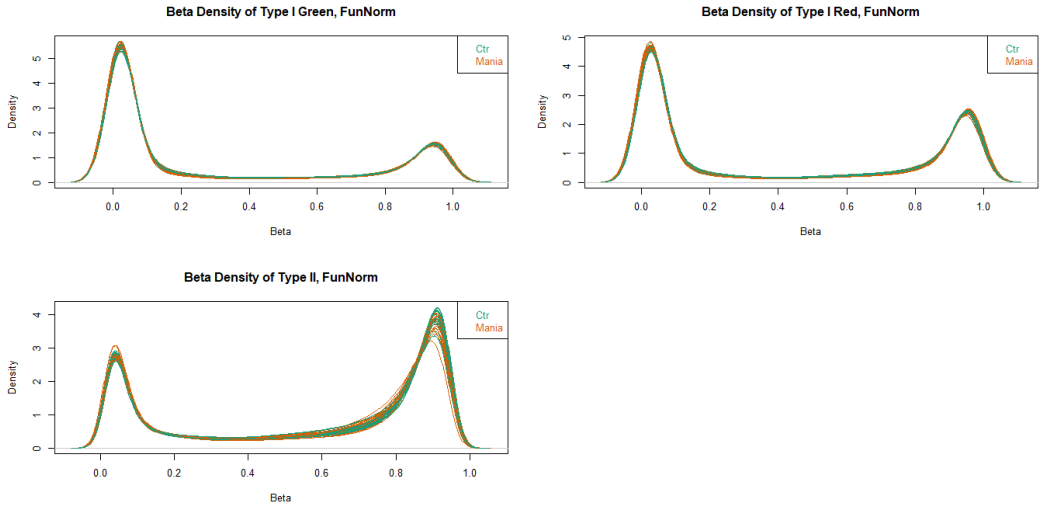

Figure 4.2 Beta Value Density Plots for the *Mania* dataset via probe type, with Functional Normalization and noob.

4.1 Variation Method 1: Bivariate Adjustment Using Bivariate Quantiles

4.1.1 Methodology

First, assume there are n arrays and we are choosing H quantiles on each at probability values $0 < p_1 < p_2 < \cdots < p_H < 1$. Let these bivariate quantiles be denoted $q_{i,1}, \ldots, q_{i,H}$ for $i = 1, \ldots, n$ and they are found using our bivariate quantile method with the Frank copula. At level p_h we fit a model to $q_{1,h}, \ldots, q_{n,h}$ and, after subtracting out the effect of the control variables, we obtain the normalized quantiles $\hat{q}_{1,h}, \ldots, \hat{q}_{n,h}$. After doing this for $h = 1, \ldots, H$ we have $\hat{q}_{i,1}, \ldots, \hat{q}_{i,H}$ on each array $i = 1, \ldots, n$. Now, from the normalization process, we can write $\hat{q}_{i,h} = q_{i,h} + (\hat{q}_{i,h} - q_{i,h}) = q_{i,h} + \hat{a}(q_{i,h})$, where $\hat{a}_{i,h}$ represents the adjustment for the p_h quantile on array i. We then try to make a similar adjustment for points on array i which are close to $q_{i,h}$ in some sense. Due to the path of quantiles that we have chosen (going along the diagonal of the Copula with $v = u$), Euclidean distance does not seem to be the ideal measure of how close points are to the normalized quantiles. Instead, we propose to use the empirical distribution function to obtain the adjustment. In particular we find h_0 such that

$$\hat{F}_i(q_{i,h_0}) \leq \hat{F}_i(y_{i,j}) < \hat{F}_i(q_{i,h_0+1})$$

We then set

$$\hat{a}(y_{i,j}) = \hat{a}(q_{i,h_0}) + \frac{\hat{F}_i(y_{i,j}) - \hat{F}_i(q_{i,h_0})}{\hat{F}_i(q_{i,h_0+1}) - \hat{F}_i(q_{i,h_0})} \left(\hat{a}(q_{i,h_0+1}) - \hat{a}(q_{i,h_0})\right)$$

We use this adjustment to get $\hat{y}_{i,j} = y_{i,j} + \hat{a}(y_{i,j})$

There are two situations in which this is not possible; when $\hat{F}_i(y_{i,j}) < \hat{F}_{(q_{i,1})}$ and when $\hat{F}_i(y_{i,j}) > \hat{F}_{(q_i,K)}$. To alleviate this issue, we add the two following quantiles:

$$q_{i,0} = \left(\min(m_{i,1}, \ldots, m_{i,M}), \min(u_{i,1}, \ldots, u_{i,M})\right) \quad q_{i,K+1} = \left(\max(m_{i,1}, \ldots, m_{i,M}), \max(u_{i,1}, \ldots, u_{i,M})\right)$$

where $m_{i,1}, \ldots, m_{i,M}$ are the methylation probe values on array i and $u_{i,1}, \ldots, u_{i,M}$ are the unmethylation probe values. We treat these exactly like the other quantiles in the sense that we would fit a model and find the adjustments at each of these minimum and maximum levels. Note that these points are guaranteed to satisfy the inequalities

$$\hat{F}_i\left(q_{i,0}\right) \leq \hat{F}_i\left(y_{i,j}\right) \quad \hat{F}_i\left(q_{i,K+1}\right) \geq \hat{F}_i\left(y_{i,j}\right) \quad j = 1, \ldots, K$$

so now it will always be possible to find a k_0 (which might be 0 or K). We would now be fitting a total of $K + 2$ models. We propose using $K = 499$ which results in $p_j = j/500, j = 0, ..., 500$.

For implementation of this method, we need to obtain the empirical joint distribution of each probe separately by probe type. To do this as efficiently as possible, the `ebvcdf()` function in the *bivariate* package in R was used (Spurdle, 2020). We compare the results via probe type. Again, before applying our method, we apply the noob dye bias correction.

4.1.2 Results

Refer to Figure 4.3 below for the results when we applied our method to the Type I Green probes.

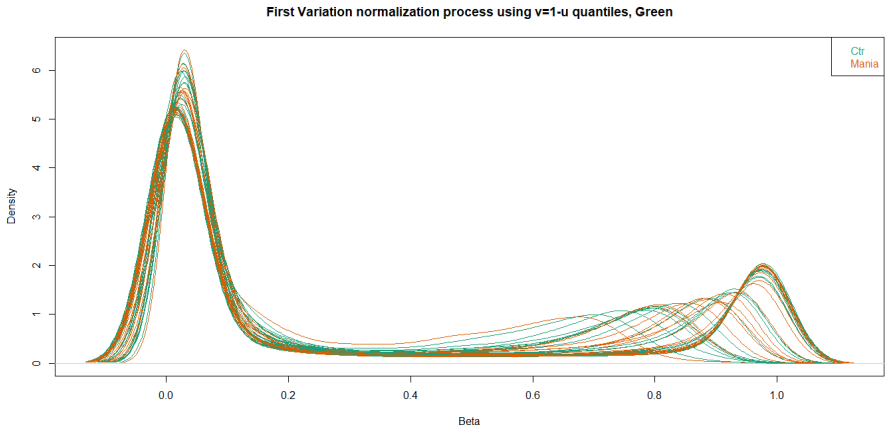

Figure 4.3 Beta Value Density Plots for the *Mania* dataset of the Type I Green with our first variation and dye bias correction done

From the results that we have obtained on the Type I probes, it is easy to see that the method proposed seems to cause multiple shifts for some samples in the Beta-value distributions, which is a behaviour not seen in the raw data or its preprocessed form via Functional Normalization. This is most likely due to the use of our bivariate quantiles and how we have chosen them. Since we set $v = u$ from our definition of a bivariate quantile, this gives us both increasing methylated and unmethylated values as the quantile level increases. We would then not have a situation for some points where, for a given sample, methylated values would be significantly larger then the unmethylated values, or vice versa. This causes the corresponding Beta value fitted with our monotonically increasing bivariate quantiles not being able to be fitted properly at beta-values approaching 1. Simply, our quantiles do not cover the full range of beta-values, as demonstrated for the Type I Green probes in the *Mania* dataset in Figure 4.4. We now discuss a second variation to our proposed method in an attempt to increase the range of beta-values covered.; using the the probability points stemming from the off-diagonal of the copula for the fitting.

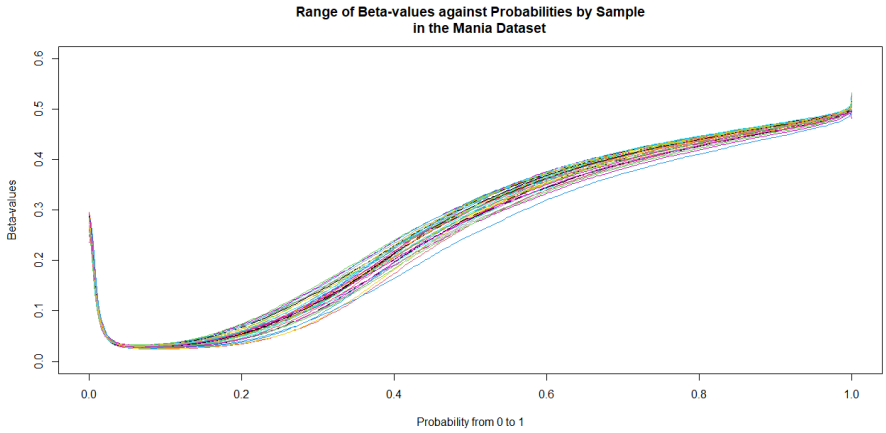

Figure 4.4 Beta-values from quantiles plotted against probabilities from 0-1 for Type 1 Green probes for the *Mania* dataset. As the probability increases, the corresponding Beta-values derived from the quantiles are limited to at most 0.5

4.2 Variation Method 2: Bivariate Adjustment Using Probability Points

We now look at the probability points obtained from the off-diagonal of the copula so as to provide better coverage of large beta values.

4.2.1 Methodology

Let $h(u) = 1 - u$. To obtain these off-diagonal points, we must solve the equation:

$$\alpha = P(X_1 \leq F_1^{-1}(u), X_2 \geq F_2^{-1}(h(u))) = u - C(u, h(u))$$

where $h(u)$ is a monotonically decreasing, continuous function, $u = 1 \implies h(u) = 0$ and $u = 0 \implies h(u) = 1$ and

$$F_1^{-1}(u) = \inf\{x | F_1 \geq u\}$$

and

$$F_2^{-1}(h(u)) = \inf\{x | F_2 \geq h(u)\}$$

In a similar manner to the bivariate quantiles, for $h(u) = 1 - u$, we have

$$\alpha_j = -\frac{1}{\theta}\log\left[1 + \frac{(\exp(-\theta u) - 1)(\exp(-\theta(1-u)) - 1)}{\exp(-\theta) - 1}\right]$$

Rearranging the equation, we get

$$\exp(\theta\alpha_j)\exp(-\theta u)\exp(-\theta) - \exp(\theta\alpha_j)\exp(-\theta u) = 2\exp(-\theta) - \exp(-\theta u) - \exp(-\theta)\exp(\theta u)$$

Now, let $z = \exp(-\theta u)$. Since $z \neq 0$, we can multiply both sides of the equation by z. Rearranging and collecting like terms then gives us

$$z^2\left(\exp(\theta\alpha)\exp(-\theta)-\exp(\theta\alpha)+1\right)-2\exp(-\theta)z+\exp(-\theta)=0$$

We again use quadratic formula to solve for z and hence find u. Let $a=\exp(\theta\alpha)\exp(-\theta)-\exp(\theta\alpha)+1$, $b=2\exp(-\theta)$ and $c=\exp(-\theta)$. We hence get

$$u(\alpha_j)=\frac{\ln\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}}{-\theta}$$

In a very similar fashion to when $g(u)=u$ for the bivariate quantiles, through empirical analysis, as well as looking at the denominator of the equation, when θ is negative, we need $1+\sqrt{...}$, to get positive values of u. Similarly, when θ is positive, we need $1-\sqrt{...}$ to obtain positive values of u. We thus get

$$u(\alpha_j)=\frac{\ln\dfrac{-b-\text{sign}(\theta)\sqrt{b^2-4ac}}{2a}}{-\theta}$$

After obtaining these bivariate probability probability points, which we will denote by $r_{i,k}$, we follow suite to the first variation method stated before. Let

$$\hat{\tilde{F}}\left(c_i,d_i\right)=P\left(\text{Meth}_i\leq c_i,\text{Unmeth}_i\geq d_i\right)=P\left(Meth_i\leq c_i\right)-P\left(Meth_i\leq c_i,Unmeth_i\leq d_i\right)$$

Then, through a similar process to what was mentioned before, we get k_0 such that $\hat{\tilde{F}}_i\left(r_{i,k_0}\right)\leq\hat{\tilde{F}}_i\left(y_{i,j}\right)<\hat{\tilde{F}}_i\left(r_{i,k_0+1}\right)$ and have the following adjustment:

$$\hat{\tilde{a}}\left(y_{i,j}\right)=\hat{\tilde{a}}\left(r_{i,k_0}\right)+\frac{\hat{F}_i\left(y_{i,j}\right)-\bar{F}_i\left(r_{i,k_0}\right)}{\hat{F}_i\left(r_{i,k_0+1}\right)-\hat{F}_i\left(r_{i,k_0}\right)}\left(\hat{\tilde{a}}\left(r_{i,k_0+1}\right)-\hat{\tilde{a}}\left(r_{i,k_0}\right)\right)$$

We then use this adjustment to get $\hat{\tilde{y}}_{i,j}=y_{i,j}+\hat{\tilde{a}}\left(y_{i,j}\right)$

Comparing the new chosen probability points to the bivariate quantile points, we see that the new chosen points cross into the regions of the data that are highly dense, whereas our original quantiles only partially do this. The reason for this is most likely related to the negative correlation that is observed in the (Meth,Unmeth) data in each array. This

is demonstrated in Figures 4.5, 4.6 and 4.7, where each figure shows the scatterplot of methylated vs unmethylated values and its corresponding dependence structure via probe type. The black points refer to the probability points chosen and the orange points refer to the bivariate quantiles. By taking these dense regions into account by using these probability points instead, we should get a better fit.

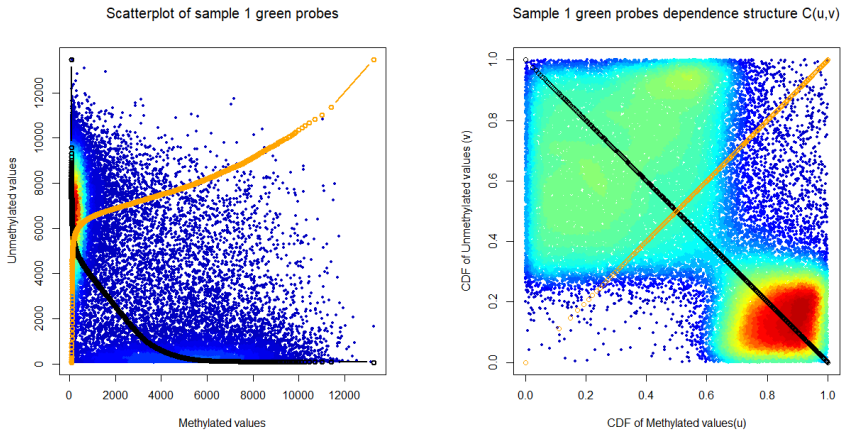

Figure 4.5 Scatterplot of Methylated vs. Unmethylated values and the corresponding dependence structure for Type I Green probes.

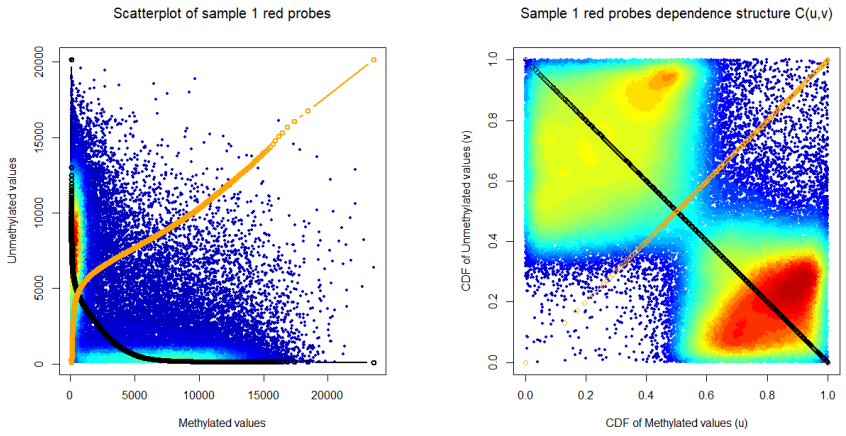

Figure 4.6 Scatterplot of Methylated vs. Unmethylated values and the corresponding dependence structure for Type I Red probes.

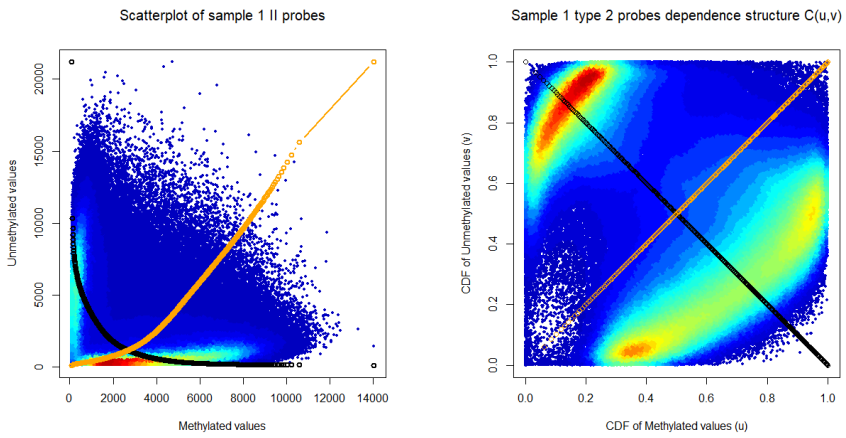

Figure 4.7 Scatterplot of Methylated vs. Unmethylated values and the corresponding dependence structure for Type II probes.

4.2.2 Results

We present our results graphically in Figure 4.8. From this figure we see that the green and red probes beta distributions have stabilized more compared to our first variation method that we suggested before, as well as not exhibiting multimodal behaviour as seen before in our first proposed method that was discussed in Chapter 3. However, when looking at the Type II probes, the beta value distributions vary a noticeable amount between samples, especially towards values of beta $= 1$. This is not ideal since we would like these distributions to be as cohesive as possible. We thus tried to develop a way to attenuate this deviation between samples through the combination of both variation methods mentioned before.

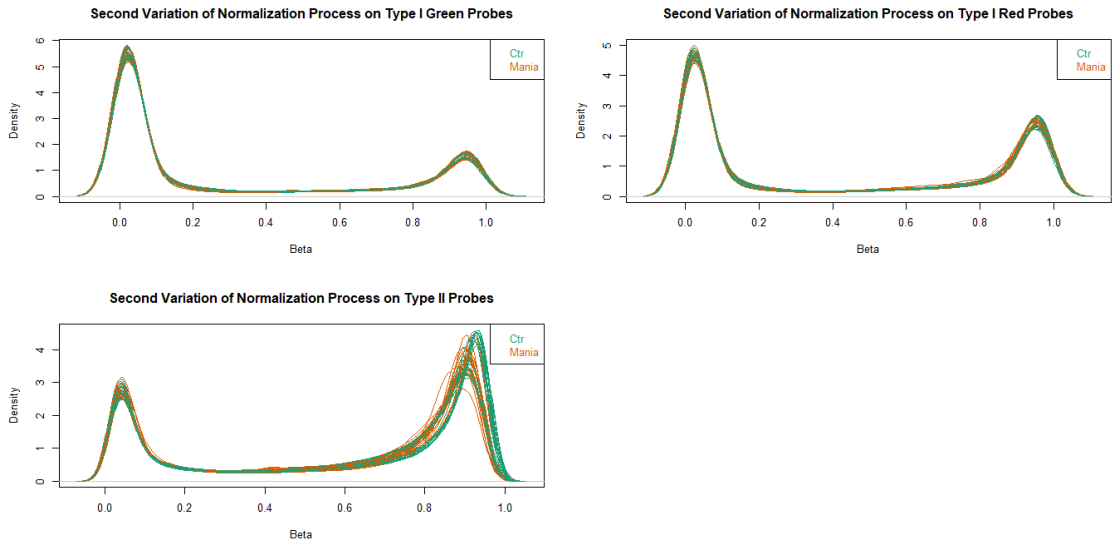

Figure 4.8 Beta Value Density Plots for the *Mania* dataset of the Type I Green, Type I Red and Type II probes with Second Variation preprocessing done

4.3 Variation Method 3: Bivariate Adjustment Using Bivariate Quantiles and Probability Points

4.3.1 Methodology

Using the bivariate quantiles and probability points obtained in the first two variation methods, we would get the two adjustments

$$\hat{a}\left(y_{i,j}\right) = \hat{a}\left(q_{i,k_0}\right) + \frac{\hat{F}_i\left(y_{i,j}\right) - \hat{F}_i\left(q_{i,k_0}\right)}{\hat{F}_i\left(q_{i,k_0+1}\right) - \hat{F}_i\left(q_{i,k_0}\right)}\left(\hat{a}\left(q_{i,k_0+1}\right) - \hat{a}\left(q_{i,k_0}\right)\right)$$

and

$$\hat{\tilde{a}}\left(y_{i,j}\right) = \hat{\tilde{a}}\left(r_{i,k_0}\right) + \frac{\hat{F}_i\left(y_{i,j}\right) - \bar{F}_i\left(r_{i,k_0}\right)}{\hat{F}_i\left(r_{i,k_0+1}\right) - \hat{F}_i\left(r_{i,k_0}\right)}\left(\hat{\tilde{a}}\left(r_{i,k_0+1}\right) - \hat{\tilde{a}}\left(r_{i,k_0}\right)\right)$$

We then use these adjustments to get $\hat{y}_{i,j} = y_{i,j} + \hat{a}\left(y_{i,j}\right)$ and $\hat{\tilde{y}}_{i,j} = y_{i,j} + \hat{\tilde{a}}\left(y_{i,j}\right)$. We then use a weighted average of the fits to obtain the final fitted value using the Euclidean distance from the point y_{ij}. To do this, we obtain the distances, d_1 and d_2 such that

$$d_1 = ||y_{ij} - 0.5(q_{i,k_0} + q_{i,k_0+1})||$$

and

$$d_2 = ||y_{ij} - 0.5(r_{i,k_0} + r_{i,k_0+1})||$$

The final adjusted y bivariate point is then

$$y_{i,j}^{final} = \frac{\frac{1}{d_1}}{\frac{1}{d_1}+\frac{1}{d_2}}\hat{y}_{i,j} + \frac{\frac{1}{d_2}}{\frac{1}{d_1}+\frac{1}{d_2}}\hat{\tilde{y}}_{i,j} = \frac{d_2}{d_1+d_2}\hat{y}_{i,j} + \frac{d_1}{d_1+d_2}\hat{\tilde{y}}_{i,j}$$

4.3.2 Results

From the results shown in Figure 4.9 , it seems that this new method, which was developed to normalize the probe distributions better via removing deviating densities between samples as seen prior, is causing more of this behaviour. Comparing to the results from our second variation method, the differences in beta-value densities between samples are also apparent in Type I green and Type I red probes. It is at this point in time where although we have other ideas that we would like to try out, we could not due to the time constraints of the thesis.

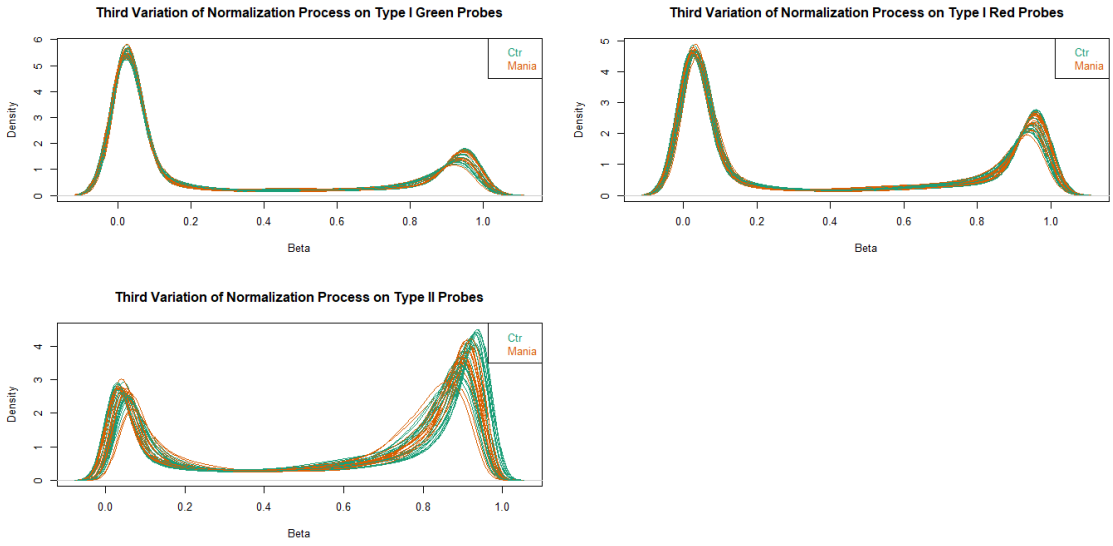

Figure 4.9 Beta Value Density Plots for the *Mania* dataset of the Type I Green, Type I Red and Type II probes with Third Variation preprocessing done

Chapter Five

Limitations and Future Work

In this work, we have conducted multiple methodologies that try to extend Functional Normalization to a bivariate setting. However, this has proved to be quite challenging. For our first methodology discussed in Chapter 3, using our bivariate quantiles and normalizing the methylated and unmethylated values separately does not generalize to all 450K data, as shown when applied to the *Mania* dataset. In Chapter 4, we try to alleviate the issues found from our first proposed method by doing adjustments to the methylated and unmethylated values using joint probability distributions and adjustments stemming from the adjustments done to our quantiles and probability points after normalizing via PLSr. However, although the new methods to perform better than our previous attempt in Chapter 3, as well as the unprocessed raw data, the deviation of beta-value distributions between samples remains concerning. Furthermore, retrieving the empirical joint distribution of each probe type using the *bivariate* package created by Spurdle (2020) is computationally difficult. This is probably due to this method is not designed to deal with samples of the size we get from the 450K array, and is likely to be even worse with the newer EPIC arrays. It is very apparent from our results that more work must be done towards this project. However, due to time constraints, other methods which were could not be implemented. Some ideas that we would have liked to implement are the use of mixture copulas and bilinear interpolation.

For our purposes, the Frank copula was used due to being able to model negative de-

pendence, as well as having a closed, solvable form compared to other copulas, such as the Gaussian and the Student-t copula; a special trait of Archimedean copulas. However, looking at the dependence structures from samples of the *Mania* dataset, we see that the structures do not follow the Frank copula. We illustrate this by comparing the dependence structure of Type I Green, Type I Red and Type II probes from the first sample of the *Mania* dataset to simulated data of the same size stemming from the Frank copula via the *copula* package by Hofert et al. (2020). The comparisons are shown in Figures 5.1, 5.2 and 5.3. The Frank copulas in the figures are simulated by using the Kendall's Tau (τ) value retrieved from the sample's methylated and unmethylated values via probe type. This shows how the dependence structures stemming from the sample should theoretically look if the data is from a Frank copula. In fact, the particular correlation structure we see here is not well modelled by any of the commonly used copulas in the literature.

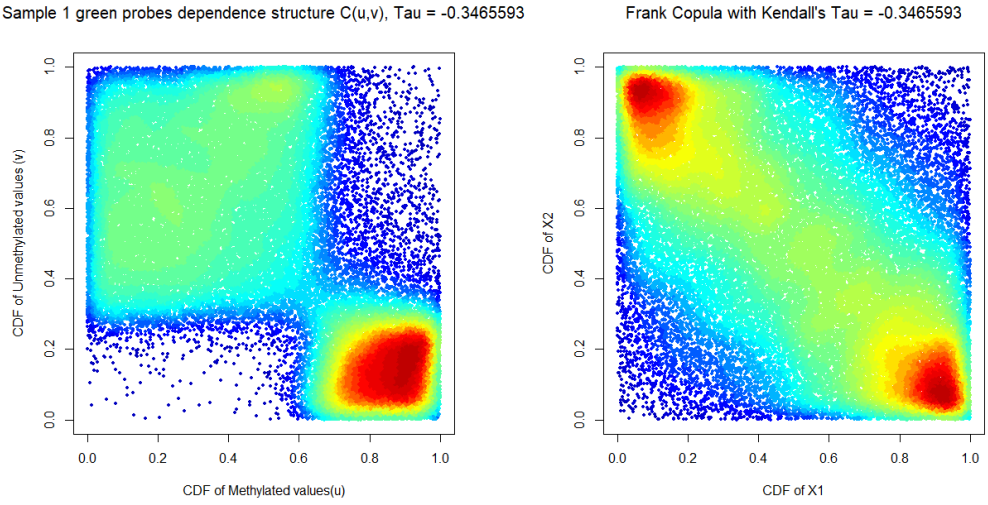

Figure 5.1 Dependence Structure of Type I Green Probes of the First Sample of the *Mania* dataset and its corresponding simulated Frank Copula

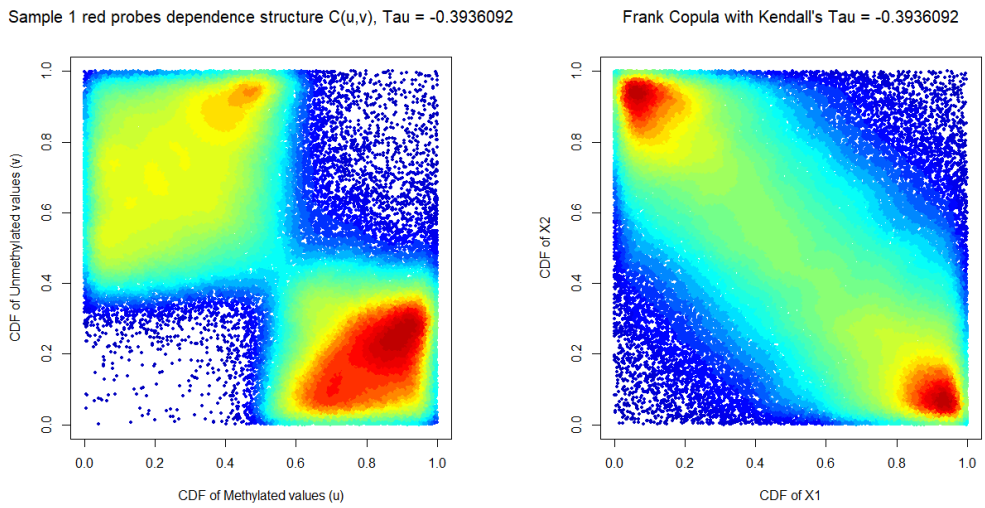

Figure 5.2 Dependence Structure of Type I Red Probes of the First Sample of the *Mania* dataset and its corresponding simulated Frank Copula

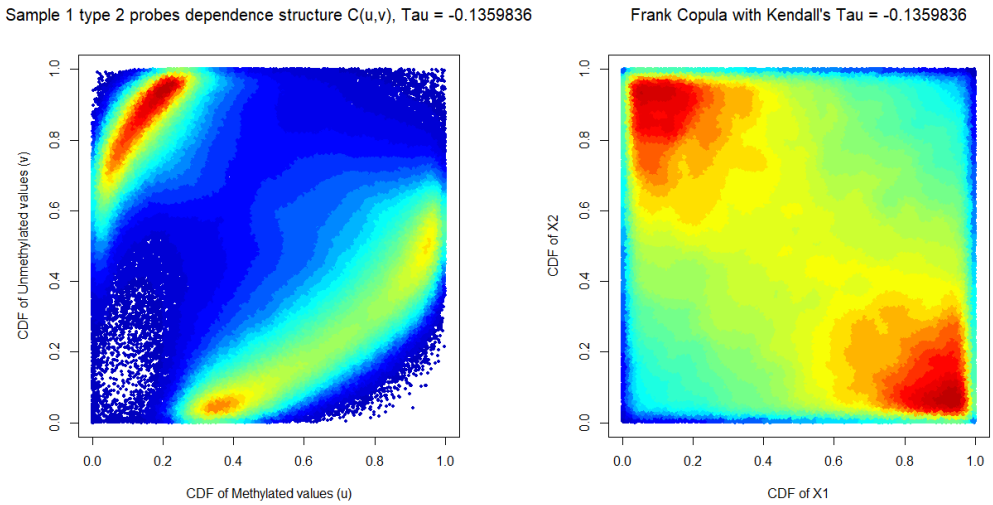

Figure 5.3 Dependence Structure of Type II Probes of the First Sample of the *Mania* dataset and its corresponding simulated Frank Copula

In the original Functional Normalization, linear interpolation was used. A natural extension of this to the bivariate case would be bilinear interpolation. Bilinear interpolation allows for interpolating functions of two variables on a rectilinear grid and is performed using linear interpolation first in one direction, and then again in the other direction. The issue

of using this method is choosing the points needed for the fitting, as choosing these grid points is non-trivial. If given more time, we would make the function needed for bilinear intrpolation the adjustments done to our chosen points after regressing out the unwanted variation by either PLSr or principal component regression.

We note that some recent work that has been made towards this matter. In particular, one idea that has shown some promise is the use of Functional Normalization directly on the Beta values or M-values instead of using them on methylated and unmethylated signals. However, a drawback of this method is that once this particular normalization method is that once this normalization procedure is done, there is no way to look at how the methylated and unmethylated signals are affected and hence, cannot be retrieved. As such, due to the complex nature of the problem, further research is needed.

REFERENCES

Bibikova, M. M., Barnes, B. P., Tsan, C. L., Ho, V., Klotzle, B., Le, J., Delano, D., Zhang, L., Schroth, G., Gunderson, K., Fan, J.-B., & Shen, R. (2011). High density DNA methylation array with single CpG site resolution. *Genomics*, *98*(4), 288–295. https://doi.org/10.1016/j.ygeno.2011.07.007

Bolstad, B., Irizarry, R., Astrand, M., & Speed, T. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, *19*(2), 185–193. https://doi.org/10.1093/bioinformatics/19.2.185

Chen, L.-A., & Welsh, A. (2002). Distribution-function-based bivariate quantiles. *Journal of Multivariate Analysis*, *83*(1), 208–231. https://doi.org/10.1006/jmva.2001.2043

Du, P., Zhang, X., Huang, C.-C., Jafari, N., Kibbe, W. A., Hou, L., & Lin, S. M. (2010). Comparison of beta-value and m-value methods for quantifying methylation levels by microarray analysis. *BMC Bioinformatics*, *11*(1). https://doi.org/10.1186/1471-2105-11-587

Filzmoser, P., Fritz, H., & Kalcher, K. (2018). *Pcapp: Robust PCA by projection pursuit* (R package version 1.9-73). https://CRAN.R-project.org/package=pcaPP

Filzmoser, P., Serneels, S., Maronna, R., & Espen, P. V. (2009). Robust multivariate methods in chemometrics. *Comprehensive Chemometrics*, 681–722. https://doi.org/10.1016/b978-044452701-1.00113-7

Fortin, J.-P., Labbe, A., Lemire, M., Zanke, B. W., Hudson, T. J., Fertig, E. J., Greenwood, C. M., & Hansen, K. D. (2014). Functional normalization of 450k methylation array data improves replication in large cancer studies. *Genome Biology*, *15*(11). https://doi.org/10.1186/s13059-014-0503-2

Hansen, K. D., Aryee, M., & Timp, W. (2019). *Minfidata: Example data for the illumina methylation 450k array* (R package version 0.30.0).

Hofert, M., Kojadinovic, I., Maechler, M., & Yan, J. (2020). *Copula: Multivariate dependence with copulas* (R package version 1.0-0). https://CRAN.R-project.org/package=copula

Jin, B., Li, Y., & Robertson, K. (2011). DNA Methylation: Superior or Subordinate in the Epigenetic Hierarchy? *Genes Cancer*, *2*(6), 607–617. https://doi.org/10.1177/1947601910393957

Jin, Z., & Liu, Y. (2018). DNA Methylation in Human Diseases. *Genes Diseases*, *5*(1), 1–8. https://doi.org/10.1016/j.gendis.2018.01.002

Jong, S. D. (1993). Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, *18*(3), 251–263. https://doi.org/10.1016/0169-7439(93)85002-x

Jöreskog, K. G., & Wold, H. (1982). *Systems under indirect observation: Causality, structure, prediction*. North-Holland.

Karakas, A. M., Karakas, M., & Dogan, M. (2017). Archimedean copula estimation parameter with Kendall distribution function. *Cumhuriyet Science Journal*, *38*(4), 619–625. https://doi.org/10.17776/csj.348292

Li, D. X. (1999). On default correlation: A copula function approach. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.187289

Lindgren, F., Geladi, P., & Wold, S. (1993). The kernel algorithm for PLS. *Journal of Chemometrics*, *7*(1), 45–59. https://doi.org/10.1002/cem.1180070104

Mevik, B.-H., Wehrens, R., & Liland, K. H. (2019). *pls: Partial least squares and principal component regression* (R package version 2.7-2). https://CRAN.R-project.org/package=pls

Nagler, T., Schepsmeier, U., Stoeber, J., Brechmann, E. C., Graeler, B., & Erhardt, T. (2019). *Vinecopula: Statistical inference of vine copulas* (R package version 2.3.0). https://CRAN.R-project.org/package=VineCopula

Nelsen, R. B. (2011). *An Introduction to Copulas*. Springer.

Onken, A., Grünewälder, S., Munk, M. H. J., & Obermayer, K. (2009). Analyzing short-term noise dependencies of spike-counts in macaque prefrontal cortex using copulas and the flashlight transformation. *PLoS Computational Biology*, *5*(11). https://doi.org/10.1371/journal.pcbi.1000577

Pidsley, R., Wong, C. C. Y., Volta, M., Lunnon, K., Mill, J., & Schalkwyk, L. C. (2013). A data-driven approach to preprocessing Illumina 450K methylation array data. *BMC Genomics*, *14*(1), 293. https://doi.org/10.1186/1471-2164-14-293

Pidsley, R., Zotenko, E., Peters, T. J., Lawrence, M. G., Risbridger, G. P., Molloy, P., Djik, S. V., Muhlhausler, B., Stirzaker, C., & Clark, S. J. (2016). Critical evaluation of the illumina MethylationEPIC BeadChip microarray for whole-genome DNA methylation profiling. *Genome Biology*, *17*(1). https://doi.org/10.1186/s13059-016-1066-1

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. http://www.R-project.org/

Rakyan, V. K., Down, T. A., Balding, D. J., & Beck, S. (2011). Epigenome-wide association studies for common human diseases. U.S. National Library of Medicine. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3508712/

Reiss, P. T., Huang, L., & Mennes, M. (2010). Fast function-on-scalar regression with penalized basis expansions. *The International Journal of Biostatistics*, *6*(1). https://doi.org/10.2202/1557-4679.1246

Ruppert, D., & Matteson, D. S. (2015). Copulas. In *Statistics and Data Analysis for Financial Engineering: With R examples* (pp. 183–215). New York, NY, Springer New York. https://doi.org/10.1007/978-1-4939-2614-5_8

Sabunciyan, S., Maher, B., Bahn, S., Dickerson, F., & Yolken, R. H. (2015). Association of DNA Methylation with Acute Mania and Inflammatory Markers. *Plos One*, *10*(7). https://doi.org/10.1371/journal.pone.0132001

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut Statistique de l'Université de Paris*, *8*, 229–231.

Spurdle, A. (2020). *Bivariate: Bivariate probability distributions* (R package version 0.5.0). https://CRAN.R-project.org/package=bivariate

Touleimat, N., & Tost, J. (2012). Complete pipeline for Infinium® Human Methylation 450K Beadchip data processing using subset quantile normalization for accurate DNA methylation estimation. *Epigenomics*, *4*(3), 325–341. https://doi.org/10.2217/epi.12.21

Triche, T. J., Weisenberger, D. J., Berg, D. V. D., Laird, P. W., & Siegmund, K. D. (2013). Low-level processing of illumina infinium dna methylation beadarrays. *Nucleic Acids Research*, *41*(7). https://doi.org/10.1093/nar/gkt090

Vineshkumar, B., & Nair, N. U. (2019). Bivariate quantile functions and their applications to reliability modelling. *STATISTICA*, *79*(1), 3–21. https://doi.org/10.6092/issn.1973-2201/8024

Wang, T., Guan, W., Lin, J., Boutaoui, N., Canino, G., Luo, J., Celedón, J. C., & Chen, W. (2015). A systematic study of normalization methods for infinium 450k methylation data using whole-genome bisulfite sequencing data. *Epigenetics*, *10*(7), 662–669. https://doi.org/10.1080/15592294.2015.1057384

Wold, H. (1966). Estimation of principal components and related models by iterative least squares (P. R. Krishnaiah, Ed.). *Multivariate Analysis*, 391–420.

Wu, M. C., Joubert, B. R., Kuan, P.-F., Håberg, S. E., Nystad, W., Peddada, S. D., & London, S. J. (2013). A systematic assessment of normalization approaches for the

Infinium 450K methylation platform. *Epigenetics*, *9*(2), 318–329. https://doi.org/10.4161/epi.27119

Wysocki, W. (2015). Kendall's Tau and Spearman's Rho for n-dimensional Archimedean copulas and their asymptotic properties. *Journal of Nonparametric Statistics*, *27*(4), 442–459. https://doi.org/10.1080/10485252.2015.1070849

Appendix A

A.1 Chapter 3 Methodology

```
#---- Chapter 3 Methodology ----
#---- creating a function similar to 'preprocessFunNorm' ----

# need minfi package for supplementary functions for 450K data
# need pcaPP package for faster computation of Kendall's Tau
# need VineCopula package to calculate parameter of copula from
# Kendall's Tau
# need pls package for efficient partial least squares computation
library(minfi)
library(pcaPP)
library(copula)
library(VineCopula)
library(pls)

# Main Function
# We note that many of the functions are either directly
# from the original FunNorm function.
# We also note that the functions we've created are conceptually
   based
# on original FunNorm but have been changed in a way to implement
# our ideas in Chapter 3.
```

```
preprocessFunnorm.bivar <- function(rgSet, ncomp=3, sex = NULL,
    bgCorr = TRUE, dyeCorr = TRUE, keepCN = TRUE, ratioConvert = TRUE,
     verbose = TRUE) {

  # makes sure that object is minfi-backed
  .isMatrixBackedOrStop(rgSet, "preprocessFunnorm.bivar")

  # makes sure that the object is'RGChannelSet' or '
     RGChannelSetExtended'
  .isRGOrStop(rgSet)
  rgSet <- updateObject(rgSet)

  # Background correction and dye bias normalization via noob
  # (preprocessNoob() function):
  if (bgCorr){
    if(verbose && dyeCorr) {
      message("[preprocessFunnorm.bivar] Background and dye bias
         correction with noob")
    } else {
      message("[preprocessFunnorm.bivar] Background correction with
         noob")
    }
    gmSet <- preprocessNoob(rgSet, dyeCorr = dyeCorr)
    if(verbose) message("[preprocessFunnorm.bivar] Mapping to genome"
       )
    gmSet <- mapToGenome(gmSet)
  } else {
    if(verbose) message("[preprocessFunnorm.bivar] Mapping to genome"
```

```
    )
  gmSet <- mapToGenome(rgSet)
}
# tells the user if the inner function should verbose
subverbose <- max(as.integer(verbose) - 1L, 0)

if(verbose) message("[preprocessFunnorm.bivar] Quantile extraction"
   )

# extraction of control probes from data as described by
# Fortin et al. (2014)
extractedData <- .extractFromRGSet450k(rgSet)
rm(rgSet)

# adds sex vector if not specified before
if (is.null(sex)) {
  gmSet <- addSex(gmSet, getSex(gmSet, cutoff = -3))
  sex <- rep(1L, length(gmSet$predictedSex))
  sex[gmSet$predictedSex == "F"] <- 2L
}

if(verbose) message("[preprocessFunnorm.bivar] Normalization")

# Gets A matrix of copy number values of the data
# Copy number variation is a phenomenon in which
# sections of the genome are repeated and the number
# of repeats in the genome varies between individuals.
# Needed to convert data to ratio data. In our case,
```

```
# to Beta-values
if(keepCN) {
  CN <- getCN(gmSet)
}

# The main function that does our quantile normalization process
# specified in the thesis in Chapter 3
gmSet <- .normalizeFunnorm450k.bivar(object = gmSet,
                                     extractedData = extractedData,
                                     ncomp = ncomp, sex=sex,
                                        verbose = subverbose)

# Stores preprocess method used.
preprocessMethod <- c(preprocessMethod(gmSet),
                      mu.norm = sprintf("Funnorm.bivar, ncomp=%s",
                         ncomp))

# Gets Beta-values if specified
if(ratioConvert) {
  grSet <- ratioConvert(gmSet, type = "Illumina", keepCN = keepCN)
  if(keepCN) {
    assay(grSet, "CN") <- CN
  }
  grSet@preprocessMethod <- preprocessMethod
  return(grSet)
} else {
  gmSet@preprocessMethod <- preprocessMethod
  return(gmSet)
```

```
  }
}

#####   Functions and Helper Functions   #####

# Gets the indices of the probes belonging in either
# Type I Green, Type I Red, Type II and Sex chromosomes
.getFunnormIndices <- function(object) {
  .isGenomicOrStop(object)
  probeType <- getProbeType(object, withColor = TRUE)
  autosomal <- (seqnames(object) %in% paste0("chr", 1:22))
  indices <- list(IGrn = which(probeType == "IGrn" && autosomal),
                  IRed = which(probeType == "IRed" && autosomal),
                  II = which(probeType == "II" && autosomal),
                  X = which(seqnames(object) == "chrX"),
                  Y = which(seqnames(object) == "chrY"))
  indices
}


# The main function that does our normalization process
# specified in Chapter 3
.normalizeFunnorm450k.bivar <- function(object, extractedData, ncomp,
    sex, verbose = FALSE) {
  normalizeQuantiles.bivar <- function(Meth, Unmeth, indices, sex =
     NULL) {
    # Get the methylated and unmethylated probe signals via
    # probe type
```

```
  Meth1 <- Meth[indices,,drop=FALSE]
  Unmeth1 <- Unmeth[indices,,drop=FALSE]
  # The old bivariate quantiles of the data as specified in the
     thesis.
  # We use the Frank copula and use v = u for the copula C(u,v).
  oldQuantiles <- .bivariate.quantiles.frank.final(Meth = Meth1,
     Unmeth = Unmeth1,
                                                   alpha = probs)

  # regressing out unwanted variation by PLS regression
  # for either the autosomes or sex chromosomes
  if(is.null(sex)) {
    newQuantiles <- .returnNewQuantiles(controlMatrix = model.
       matrix,
                                        quantiles = oldQuantiles,
                                           ncomp = ncomp)
  } else {
    newQuantiles <- .returnFitBySex.bivar(controlMatrix = model.
       matrix,
                                          quantiles = oldQuantiles,
                                          ncomp = ncomp, sex = sex)
  }
  # This function was from the original FunNorm
  # where linear interpolation will be done between the quantiles
  # after regressing out the variation.
  .normalizeMatrix(cbind(Meth1,Unmeth1), newQuantiles)
}
indicesList <- .getFunnormIndices(object)
```

```
# The control matrix is built from the extracted data
# by using the summary statistics defined in Fortin et al. (2014)
# for the control probes
model.matrix <- .buildControlMatrix450k(extractedData)

# Remove the first and last quantile level since we set the 1st
   quantile
# to be 0 and the last quantile to be the 499th quantile value +
   1000
probs <- seq(from = 0, to = 1, length.out = 500)[c(-1,-500)]
Meth <- getMeth(object)
Unmeth <- getUnmeth(object)
Combined <- cbind(Meth,Unmeth)
n = ncol(Meth)
if (ncomp > 0){

  # autosomes normalization
  for (type in c("IGrn", "IRed", "II")) {
    indices <- indicesList[[type]]
    if(length(indices) > 0) {
      if(verbose) message(sprintf("[normalizeFunnorm450k.bivar] 
         Normalization of the %s probes", type))
      Combined[indices,] <- normalizeQuantiles.bivar(Meth = Meth,
                                                     Unmeth =
                                                        Unmeth,
                                                     indices =
                                                        indices,
```

```
                                                    sex = NULL)

    }
  }
  # X-chrom normalization
  indices <- indicesList[["X"]]
  if(length(indices) > 0) {
    if(verbose) message("[normalizeFunnorm450k.bivar] Normalization
       of the X-chromosome")
    Combined[indices,] <- normalizeQuantiles.bivar(Meth =Meth,
                                                   Unmeth = Unmeth,
                                                   indices =
                                                      indices,
                                                   sex = sex)
    #()
  }
}
#()

Meth = Combined[,c(1:n)]
Unmeth = Combined[,c((n+1):(2*n))]

# Y-chrom normalization
# For Y-chrom, we use quantile normalization defined by Bolstad et
   al. (2003)
# This method is in the preprocessCore package but is imported with
    minfi
indices <- indicesList[["Y"]]
```

```
if(length(indices) > 0) {
  if(verbose) message("[normalizeFunnorm450k.bivar] Normalization
     of the Y-chromosome")
  sex <- as.character(sex)
  levels <- unique(sex)
  nSexes <- length(levels)
  #()
  if (nSexes == 2) {
    level1 <- levels[1]
    level2 <- levels[2]
  }
  if (nSexes == 2) {
    if (sum(sex == level1)>1) {
      Meth[indices, sex==level1]   <- preprocessCore::normalize.
         quantiles(Meth[indices, sex == level1, drop=FALSE])
      Unmeth[indices, sex==level1] <- preprocessCore::normalize.
         quantiles(Unmeth[indices, sex == level1,drop=FALSE])
    }
    if (sum(sex == level2)>1) {
      Meth[indices, sex==level2]   <- preprocessCore::normalize.
         quantiles(Meth[indices, sex == level2,drop=FALSE])
      Unmeth[indices, sex==level2] <- preprocessCore::normalize.
         quantiles(Unmeth[indices, sex == level2,drop=FALSE])
    }
  } else {
    Meth[indices,] <- preprocessCore::normalize.quantiles(Meth[
       indices,])
    Unmeth[indices,] <- preprocessCore::normalize.quantiles(Unmeth[
```

```
            indices,])
      }
    }

    assay(object, "Meth") <- Meth
    assay(object, "Unmeth") <- Unmeth
    return(object)
}



### To extract quantiles and control probes from rgSet
.extractFromRGSet450k <- function(rgSet) {
    rgSet <- updateObject(rgSet)
    controlType <- c("BISULFITE␣CONVERSION␣I",
                     "BISULFITE␣CONVERSION␣II",
                     "EXTENSION",
                     "HYBRIDIZATION",
                     "NEGATIVE",
                     "NON-POLYMORPHIC",
                     "NORM_A",
                     "NORM_C",
                     "NORM_G",
                     "NORM_T",
                     "SPECIFICITY␣I",
                     "SPECIFICITY␣II",
                     "TARGET␣REMOVAL",
                     "STAINING")
```

```
#()
array <- annotation(rgSet)[["array"]]
#()
## controlAddr <- getControlAddress(rgSet, controlType =
   controlType, asList = TRUE)
ctrls <- getProbeInfo(rgSet, type = "Control")
#()
if(!all(controlType %in% ctrls$Type))
  stop("The‘rgSet‘does not contain all necessary control probes")
ctrlsList <- split(ctrls, ctrls$Type)[controlType]
#()

redControls <- getRed(rgSet)[ctrls$Address,,drop=FALSE]
#()
redControls <- lapply(ctrlsList, function(ctl) redControls[ctl$
   Address,,drop=FALSE])
#()
greenControls <- getGreen(rgSet)[ctrls$Address,,drop=FALSE]
#()
greenControls <- lapply(ctrlsList, function(ctl) greenControls[ctl$
   Address,,drop=FALSE])
#()

## Extraction of the undefined negative control probes
oobRaw <- getOOB(rgSet)
probs <- c(0.01, 0.50, 0.99)
greenOOB <- t(colQuantiles(oobRaw$Grn, na.rm = TRUE, probs = probs)
   )
```

```
  redOOB    <- t(colQuantiles(oobRaw$Red, na.rm=TRUE,  probs = probs))
  oob       <- list(greenOOB = greenOOB, redOOB = redOOB)

  return(list(
    greenControls = greenControls,
    redControls = redControls,
    oob = oob, ctrlsList = ctrlsList,
    array = array))
}


## Extraction of the Control matrix from the control probes in
   extracted data
## These are the summary statistics of the control probes
## as specified in Fortin et al. (2014)
.buildControlMatrix450k <- function(extractedData) {
  getCtrlsAddr <- function(exType, index) {
    ctrls <- ctrlsList[[index]]
    addr <- ctrls$Address
    names(addr) <- ctrls$ExtendedType
    na.omit(addr[exType])
  }

  array <- extractedData$array
  greenControls <- extractedData$greenControls
  redControls <- extractedData$redControls
  controlNames <- names(greenControls)
  ctrlsList <- extractedData$ctrlsList
```

```
## Bisulfite conversion extraction for probe type II:
index <- match("BISULFITE␣CONVERSION␣II", controlNames)
redControls.current <- redControls[[ index ]]
bisulfite2 <- colMeans2(redControls.current, na.rm = TRUE)

## Bisulfite conversion extraction for probe type I:
index <- match("BISULFITE␣CONVERSION␣I", controlNames)
if (array=="IlluminaHumanMethylation450k"){
  addr <- getCtrlsAddr(exType = sprintf("BS␣Conversion␣I%sC%s", c("
     ␣", "-", "-"), 1:3), index = index)
} else {
  addr <- getCtrlsAddr(exType = sprintf("BS␣Conversion␣I%sC%s", c("
     -", "-"), 1:2), index = index)
}
greenControls.current <- greenControls[[ index ]][addr,,drop=FALSE]
if (array=="IlluminaHumanMethylation450k"){
  addr <- getCtrlsAddr(exType = sprintf("BS␣Conversion␣I-C%s", 4:6)
     , index = index)
} else {
  addr <- getCtrlsAddr(exType = sprintf("BS␣Conversion␣I-C%s", 3:5)
     , index = index)
}
redControls.current <- redControls[[ index ]][addr,, drop=FALSE]
if (nrow(redControls.current)==nrow(greenControls.current)){
  bisulfite1 <- colMeans2(redControls.current + greenControls.
     current, na.rm = TRUE)
} else {
```

```
  bisulfite1 <- colMeans2(redControls.current, na.rm=TRUE) +
     colMeans2(greenControls.current, na.rm = TRUE)
}


## Staining
index <- match("STAINING", controlNames)
addr <- getCtrlsAddr(exType = "Biotin␣(High)", index = index)
stain.green <- t(greenControls[[ index ]][addr,,drop=FALSE])
addr <- getCtrlsAddr(exType = "DNP␣(High)", index = index)
stain.red <- t(redControls[[ index ]][addr,, drop=FALSE ])

## Extension
index <-    match("EXTENSION", controlNames)
addr <- getCtrlsAddr(exType = sprintf("Extension␣(%s)", c("A", "T")
   ), index = index)
extension.red <- t(redControls[[index]][addr,,drop=FALSE])
colnames(extension.red) <- paste0("extRed", 1:ncol(extension.red))
addr <- getCtrlsAddr(exType = sprintf("Extension␣(%s)", c("C", "G")
   ), index = index)
extension.green <- t(greenControls[[index]][addr,,drop=FALSE])
colnames(extension.green) <- paste0("extGrn", 1:ncol(extension.
   green))

## Hybridization should be monitored only in the green channel
index <- match("HYBRIDIZATION", controlNames)
hybe <- t(greenControls[[index]])
colnames(hybe) <- paste0("hybe", 1:ncol(hybe))
```

```
## Target removal should be low compared to hybridization probes
index <- match("TARGET␣REMOVAL", controlNames)
targetrem <- t(greenControls[[index]])
colnames(targetrem) <- paste0("targetrem", 1:ncol(targetrem))

## Non-polymorphic probes
index <- match("NON-POLYMORPHIC", controlNames)
addr <- getCtrlsAddr(exType = sprintf("NP␣(%s)", c("A", "T")),
   index = index)
nonpoly.red <- t(redControls[[index]][addr, ,drop=FALSE])
colnames(nonpoly.red) <- paste0("nonpolyRed", 1:ncol(nonpoly.red))
addr <- getCtrlsAddr(exType = sprintf("NP␣(%s)", c("C", "G")),
   index = index)
nonpoly.green <- t(greenControls[[index]][addr, ,drop=FALSE])
colnames(nonpoly.green) <- paste0("nonpolyGrn", 1:ncol(nonpoly.
   green))

## Specificity II
index <- match("SPECIFICITY␣II", controlNames)
greenControls.current <- greenControls[[index]]
redControls.current <- redControls[[index]]
spec2.green <- t(greenControls.current)
colnames(spec2.green) <- paste0("spec2Grn", 1:ncol(spec2.green))
spec2.red <- t(redControls.current)
colnames(spec2.red) <- paste0("spec2Red", 1:ncol(spec2.red))
spec2.ratio <- colMeans2(greenControls.current, na.rm = TRUE) /
  colMeans2(redControls.current, na.rm = TRUE)
```

```
## Specificity I
index <- match("SPECIFICITY␣I", controlNames)
addr <- getCtrlsAddr(exType = sprintf("GT␣Mismatch␣%s␣(PM)", 1:3),
   index = index)
greenControls.current <- greenControls[[index]][addr,,drop=FALSE]
redControls.current <- redControls[[index]][addr,,drop=FALSE]
spec1.green <- t(greenControls.current)
colnames(spec1.green) <- paste0("spec1Grn", 1:ncol(spec1.green))
spec1.ratio1 <- colMeans2(redControls.current, na.rm = TRUE) /
  colMeans2(greenControls.current, na.rm = TRUE)

index <- match("SPECIFICITY␣I", controlNames) # Added that line
addr <- getCtrlsAddr(exType = sprintf("GT␣Mismatch␣%s␣(PM)", 4:6),
   index = index)
greenControls.current <- greenControls[[index]][addr,,drop=FALSE]
redControls.current <- redControls[[index]][addr,,drop=FALSE]
spec1.red <- t(redControls.current)
colnames(spec1.red) <- paste0("spec1Red", 1:ncol(spec1.red))
spec1.ratio2 <- colMeans2(greenControls.current, na.rm = TRUE) /
  colMeans2(redControls.current, na.rm = TRUE)
spec1.ratio <- (spec1.ratio1 + spec1.ratio2) / 2

## Normalization probes:
index <- match(c("NORM_A"), controlNames)
normA <- colMeans2(redControls[[index]], na.rm = TRUE)
index <- match(c("NORM_T"), controlNames)
normT <- colMeans2(redControls[[index]], na.rm = TRUE)
```

```
index <- match(c("NORM_C"), controlNames)
normC <- colMeans2(greenControls[[index]], na.rm = TRUE)
index <- match(c("NORM_G"), controlNames)
normG <- colMeans2(greenControls[[index]], na.rm = TRUE)

dyebias <- (normC + normG)/(normA + normT)

oobG <- extractedData$oob$greenOOB
oobR <- extractedData$oob$redOOB
oob.ratio <- oobG[2,]/oobR[2,]
oobG <- t(oobG)
colnames(oobG) <- paste0("oob", c(1,50,99))

model.matrix <- cbind(
  bisulfite1, bisulfite2, extension.green, extension.red, hybe,
  stain.green, stain.red, nonpoly.green, nonpoly.red,
  targetrem, spec1.green, spec1.red, spec2.green, spec2.red, spec1.
     ratio1,
  spec1.ratio, spec2.ratio, spec1.ratio2, normA, normC, normT,
     normG, dyebias,
  oobG, oob.ratio)


## Imputation
for (colindex in 1:ncol(model.matrix)) {
  if(any(is.na(model.matrix[,colindex]))) {
    column <- model.matrix[,colindex]
    column[is.na(column)]    <- mean(column, na.rm = TRUE)
```

```
      model.matrix[ , colindex] <- column
    }
  }

  ## Scaling
  model.matrix <- scale(model.matrix)

  ## Fixing outliers
  model.matrix[model.matrix > 3] <- 3
  model.matrix[model.matrix < (-3)] <- -3

  ## Rescaling
  model.matrix <- scale(model.matrix)

  return(model.matrix)
}

# The old quantiles in a bivariate manner using our definition
# of a bivariate quantile. We use the Frank copula with C(u,v)
# where v =u in our case.
.bivariate.quantiles.frank.final = function(Meth, Unmeth, alpha){
  n = ncol(Meth)
  Meth.sort = apply(Meth, 2, sort)
  Unmeth.sort = apply(Unmeth, 2, sort)
  bivariate.list = c()

  for (i in 1:n){
    m = length(Meth[,i])
```

```
    tau = cor.fk(Meth[,i],Unmeth[,i])
    theta = BiCopTau2Par(family = 5, tau = tau)
    roots = (log(1 - sign(theta)* sqrt(-exp(-theta)
                                       - exp(-theta*alpha)
                                       + exp(-theta*(alpha+1)) + 1)))
                                           / -theta
    quants.meth = Meth.sort[,i][m*roots]
    quants.unmeth = Unmeth.sort[,i][m*roots]
    b.quants = cbind(quants.meth,quants.unmeth)
    colnames(b.quants) = c(paste(colnames(Meth)[i], "Meth" ,sep = '.'
       ),
                           paste(colnames(Unmeth)[i], "Unmeth", sep =
                               '.'))
    bivariate.list = cbind(bivariate.list, b.quants)
  }
  row.names(bivariate.list) = as.character(alpha)
  return(bivariate.list)
}



# Return the normalized quantiles in a bivariate manner
# for the autosomes
.returnNewQuantiles = function(controlMatrix, quantiles, ncomp){
  n =  ncol(quantiles)
  quantiles = rbind(numeric(n),quantiles, quantiles[nrow(quantiles),]
      + 1000)
  rownames(quantiles)[1] = '0.0'; rownames(quantiles)[500] = '1.0'
```

```
  newQuantiles = array(0L, dim(quantiles))
  #()
  for (i in 2:nrow(quantiles)){
    #()
    quant = matrix(quantiles[i,], n/2,2, byrow = T)
    #()
    meanFunction = colMeans2(quant)
    res = sweep(quant,2,meanFunction)
    fit = plsr(res ~ controlMatrix,ncomp = ncomp)
    newfit = sweep(fit$residuals[, , paste(as.character(ncomp), '
       comps')],2,-meanFunction)
    newQuantiles[i,] = as.vector(newfit)
  }
  newQuantiles = .regularizeQuantiles(newQuantiles)

  #Putting row and column names for convenience
  nam = c()
  nam = append(nam,colnames(quantiles)[seq(1,n-1,2)])
  nam = append(nam,colnames(quantiles)[seq(2,n,2)])
  colnames(newQuantiles) = nam
  row.names(newQuantiles) = row.names(quantiles)
  return(newQuantiles)
}

### Return the normalized quantile functions in a bivariate manner
   for sex
.sortbysex = function(sex, levels, newQuantiles1, newQuantiles2){
```

```
#note that newQuantiles1 is for female in this example
#and newQuantiles2 is for male in this example
n = ncol(newQuantiles1)/2
m = ncol(newQuantiles2)/2
l = length(sex)
# m+n = l
newQuantiles1.meth = newQuantiles1[,c(1:n)]
newQuantiles1.unmeth = newQuantiles1[,c((n+1):(2*n))]
newQuantiles2.meth = newQuantiles2[,c(1:m)]
newQuantiles2.unmeth = newQuantiles2[,c((m+1):(2*m))]
meth.sort =  matrix(0,nrow=500,ncol=(n+m))
unmeth.sort = matrix(0,nrow=500,ncol=(n+m))
#()
for (i in 1:l){
  if (sex[i] == levels[1]){
    meth.sort[,i] <- newQuantiles1.meth[,1]
    unmeth.sort[,i] <- newQuantiles1.unmeth[,1]
    newQuantiles1.meth = as.matrix(newQuantiles1.meth[,-1])
    newQuantiles1.unmeth = as.matrix(newQuantiles1.unmeth[,-1])
    #()
  } else{
    meth.sort[,i] <- newQuantiles2.meth[,1]
    unmeth.sort[,i] <- newQuantiles2.unmeth[,1]
    newQuantiles2.meth = as.matrix(newQuantiles2.meth[,-1])
    newQuantiles2.unmeth = as.matrix(newQuantiles2.unmeth[,-1])
    #()
  }
}
```

```
  fulllist = cbind(meth.sort,unmeth.sort)
  return(fulllist)
}

# Helper function for .returnFitBySex.bivar
# Needed when normalization should be done separately by sex
.sex.bivariate = function(sex){
  sex.biv = c()
  for (i in 1:length(sex)){
    sex.val = rep(sex[i],2)
    sex.biv = append(sex.biv,sex.val)
  }
  return(sex.biv)
}

.returnFitBySex.bivar <- function(controlMatrix, quantiles, ncomp,
   sex) {
  stopifnot(is.matrix(quantiles))
  stopifnot(is.matrix(controlMatrix))
  sex    <- as.character(sex)
  levels <- unique(sex)
  nSexes <- length(levels)
  if (nSexes == 2) {
    sex1 <- sum(sex == levels[1])
    sex2 <- sum(sex == levels[2])

  } else {
    sex1 <- sum(sex == levels[1])
```

```
  sex2 <- 0
}

## When normalization should not be performed by sex separately:
if ((sex1 <= 10) | (sex2 <= 10)) {
  newQuantiles <- .returnNewQuantiles(controlMatrix = controlMatrix
     ,
                                      quantiles = quantiles,
                                      ncomp = ncomp)
} else {
  sex.bivar = .sex.bivariate(sex=sex)

  quantiles1 <- quantiles[, sex.bivar == levels[1]]
  controlMatrix1 <- controlMatrix[sex == levels[1], ]
  newQuantiles1 <- .returnNewQuantiles(controlMatrix =
     controlMatrix1,
                                       quantiles = quantiles1,
                                       ncomp = ncomp)

  quantiles2 <- quantiles[, sex.bivar == levels[2]]
  controlMatrix2 <- controlMatrix[sex == levels[2], ]
  newQuantiles2 <- .returnNewQuantiles(controlMatrix =
     controlMatrix2,
                                       quantiles = quantiles2,
                                       ncomp = ncomp)

  newQuantiles = .sortbysex(sex = sex, levels=levels,
                            newQuantiles1=newQuantiles1,
```

```
                                newQuantiles2=newQuantiles2)
  }

  return(newQuantiles)
}

# Normalize a matrix of intensities via linear interpolation
# between the quantile values and
# quantile normalization using this new distribution as target/
   reference
# distribution.
# This then makes the new methylated and unmethylated values come
# from this distribution
.normalizeMatrix <- function(intMatrix, newQuantiles) {
  n <- nrow(newQuantiles)
  normMatrix <- sapply(1:ncol(intMatrix), function(i) {
    crtColumn <- intMatrix[ , i]
    crtColumn.reduced <- crtColumn[!is.na(crtColumn)]
    ## Generation of the corrected intensities:
    target <- sapply(1:(n-1), function(j) {
      start <- newQuantiles[j,i]
      end <- newQuantiles[j+1,i]
      if (!isTRUE(all.equal(start,end))){
        sequence <- seq(start, end,( end-start)/n)[-(n+1)]
      } else {
        sequence <- rep(start, n)
      }
      return(sequence)
```

```
    })
    target <- as.vector(target)
    result <- preprocessCore::normalize.quantiles.use.target(matrix(
       crtColumn.reduced), target)
    return(result)
  })
  return(normMatrix)
}

# To ensure a monotonically increasing and non-negative quantile
   function
.regularizeQuantiles <- function(x){
  x[x<0] <- 0
  colCummaxs(x)
}

# Some other helper functions obtained fromt the utilities in minfi
# so to make sure the data we work with is the correct object class
.isMatrixBackedOrStop <- function(object, FUN) {
  if (!.isMatrixBacked(object)) {
    stop("'", FUN, "()' only supports matrix-backed minfi objects.",
         call. = FALSE)
  }
}

.isMatrixBacked <- function(object) {
  stopifnot(is(object, "SummarizedExperiment"))
  all(vapply(assays(object), is.matrix, logical(1L)))
```

```
}

.isRGOrStop <- function(object) {
  if (!is(object, "RGChannelSet")) {
    stop("object␣is␣of␣class␣'", class(object), "',␣but␣needs␣to␣be␣
       of␣",
         "class␣'RGChannelSet'␣or␣'RGChannelSetExtended'")
  }
}

.isGenomicOrStop <- function(object) {
  if (!is(object, "GenomicMethylSet") && !is(object, "GenomicRatioSet
     ")) {
    stop("object␣is␣of␣class␣'", class(object), "',␣but␣needs␣to␣be␣
       of␣",
         "class␣'GenomicMethylSet'␣or␣'GenomicRatioSet'")
  }
}
```

Appendix B

B.1 Chapter 4 Methodologies

```
#---- Chapter 4: Second Methodology ----

# Need minfi package for supplementary functions for 450K data.
# Need pcaPP pacakge for faster computation of Kendall's Tau.
# Need VineCopula package to calculate parameter of copula from
# Kendall's Tau.
# Need pls package for efficient partial least squares computation.
# Need bivariate package for efficient computation of joint
   probabilities
library(minfi)
library(VineCopula)
library(pcaPP)
library(bivariate)
library(pls)

#---- Functions ---
# Getting the probabilities that stem from Methylated values for
   faster
# computation
getmeth.probs = function(Meth){
  n = ncol(Meth)
```

```
  Fhat = vector("list",n)
  Fhat1 = matrix(NA, nrow=nrow(Meth), ncol=n)
  for (i in 1:n) {
    Fhat[[i]] = ecdf(Meth[,i])
  }
  for (i in 1:n) {
    Fhat1[,i] = Fhat[[i]](Meth[,i])
  }
  return(Fhat1)
}

# This function retrieves the joint probabilities for each probe
# for each sample.
get.bivariate.probabilities = function(Meth, Unmeth){
  n = ncol(Meth)
  Fhat = vector("list",n)
  Fhat1 = matrix(NA, nrow=nrow(Meth), ncol=n)
  for (i in 1:n) {
    Fhat[[i]] = ebvcdf(Meth[,i],Unmeth[,i])
  }
  for (i in 1:n) {
    Fhat1[,i] = Fhat[[i]](Meth[,i], Unmeth[,i])
  }
  return(Fhat1)
}


# We now retrieve the probability points discussed in Chapter 4.2
```

```
# by letting v =1-u for the copula C(u,v)
bivariate.quantiles.frank.final.2 = function(Meth, Unmeth, alpha){
  n = ncol(Meth)
  Meth.sort = apply(Meth, 2, sort)
  Unmeth.sort = apply(Unmeth, 2, sort)
  bivariate.list = c()
  for (i in 1:n){
    m = length(Meth[,i])
    tau = cor.fk(Meth[,i],Unmeth[,i])
    theta = BiCopTau2Par(family = 5, tau = tau)

    a = exp(theta*(alpha - 1)) - exp(theta*alpha) + 1
    b = -2*exp(-theta)
    c = exp(-theta)

    roots = (log((-b + sqrt(b^2 -4*a*c))/(2*a))) / (-theta)
    roots2 = 1 - roots

    roots[roots == 0] <- 1/m
    roots2[roots2 == 0] <-1/m

    quants.meth = Meth.sort[,i][m*roots]
    quants.unmeth = Unmeth.sort[,i][m*roots2]

    b.quants = cbind(quants.meth,quants.unmeth)
    colnames(b.quants) = c(paste(colnames(Meth)[i], "Meth" ,sep = '.'
       ),
                           paste(colnames(Unmeth)[i], "Unmeth", sep =
```

```
                                    '.'))
    bivariate.list = cbind(bivariate.list, b.quants)
  }
  return(bivariate.list)
}

# For the methodology described in 4.2, we retrieve the minimums and
   maximums
# instead of setting the minimums to 0 and the maximums to the second
    highest
# value + 1000
returnNewQuantiles2 = function(controlMatrix, quantiles, ncomp){

  n =  ncol(quantiles)
  newQuantiles = array(0L, dim(quantiles))

  for (i in 1:nrow(quantiles)){
    quant = matrix(quantiles[i,], n/2,2, byrow = T)
    meanFunction = colMeans2(quant)
    res = sweep(quant,2,meanFunction)
    fit = plsr(res ~ controlMatrix,ncomp = ncomp)
    newfit = sweep(fit$residuals[, , paste(as.character(ncomp), '
       comps')],2,-meanFunction)
    newQuantiles[i,] = as.vector(newfit)
  }
  newQuantiles = .regularizeQuantiles(newQuantiles)
  #Putting row and column names for convenience, in case if you want
     to look
```

```
  nam = c()
  nam = append(nam,colnames(quantiles)[seq(1,n-1,2)])
  nam = append(nam,colnames(quantiles)[seq(2,n,2)])
  colnames(newQuantiles) = nam
  row.names(newQuantiles) = row.names(quantiles)
  return(newQuantiles)
}

# We do this regularization in this way since the values in these
# bivariate probability points are bidirectional, ie when
# one value increases, the other value decreases
.regularizeQuantiles <- function(x){
  n = ncol(x) /2
  x[x<0] <- 0
  x = cbind(colCummaxs(x[,1:n]),
            colCummins(x[,(n+1):(2*n)]))
}

# We now take get the adjustments that were done to the quantiles
# by the PLS regression and store them
adjustment = function(oldQuantiles, newQuantiles){
  n = ncol(oldQuantiles)
  old.meth = oldQuantiles[,seq(1,(n - 1),by=2)]
  old.unmeth = oldQuantiles[,seq(2,n,by=2)]
  #browser()
  oldquants = cbind(old.meth,old.unmeth)
  adjust = newQuantiles - oldquants
  return(adjust)
```

```
}

# The adjustment process described in the first and
# second Variation Methodologies
normalize.bivariate = function(Meth, Unmeth, joint.ecdf,
                               quantiles.edf, adjustments){
  n = ncol(Meth)
  m = nrow(Meth)
  s = dim(quantiles.edf)[1]

  adjust.meth = adjustments[,1:n]
  adjust.unmeth = adjustments[,(n+1):(2*n)]

  Meth.new = matrix(NA, nrow=m,  ncol = n)
  Unmeth.new =  matrix(NA, nrow=m,  ncol = n)
  for (i in 1:n){
    for (j in 1:(s-1)){

      f2 = quantiles.edf[(j+1),i]
      f1 = quantiles.edf[j,i]

      indices = which( f1<=joint.ecdf[,i] & joint.ecdf[,i] < f2)
      fhat = joint.ecdf[indices, i]
      y = cbind(Meth[indices,i],Unmeth[indices,i])

      adjust.k2 = cbind(adjust.meth[(j+1),i], adjust.unmeth[(j+1),i])
      adjust.k1 = cbind(adjust.meth[j,i], adjust.unmeth[j,i])
```

```
      weight = (fhat - f1) / (f2 - f1)
      diff.k = adjust.k2 - adjust.k1

      adjust.y = cbind(adjust.k1[1] + weight*diff.k[1], adjust.k1[2]
         + weight*diff.k[2])
      yhat = y + adjust.y

      # Methylated and Unmethylated values must be >= 0
      yhat[yhat<0] <-0
      Meth.new[indices, i] = yhat[,1]
      Unmeth.new[indices, i] = yhat[,2]
    }
  }
  return(cbind(Meth.new,Unmeth.new))
}

# ---- Applying the methodology to the Mania dataset ----

# Retrieving the dataset from where is was stored
RGSet <- read.metharray.exp('D:/methyldatamedium/methyldatamedium')

# Getting the phenotypes of the samples (whether a patient has Mania
   or not)
type.mania = c()
for (i in seq(1,80, by =2)){
  l = as.character(type_mania[i,1])
  type.mania = append(type.mania, l)
}
```

```
# We follow the steps done from FunNorm to the data prior
# to normalization. We use the functions from the Chapter 3
# methodology to get what we need
gmSet <- preprocessNoob(RGSet, dyeCorr = T)
gmSet <- mapToGenome(gmSet)

extractedData = .extractFromRGSet450k(RGSet)
mod.mat = .buildControlMatrix450k(extractedData)

indices = .getFunnormIndices(gmSet)
Meth.cor = getMeth(gmSet)
Unmeth.cor = getUnmeth(gmSet)

probs <- seq(from = 0, to = 1, length.out = 501)



#---- Type I Green probes ----
mat1.meth = Meth.cor[indices[[1]],,drop=FALSE]
mat1.unmeth = Unmeth.cor[indices[[1]],,drop=FALSE]

Meth.green.probs = getmeth.probs(mat1.meth)

system.time({joint.edf.green =
  get.bivariate.probabilities(mat1.meth,mat1.unmeth)})
# user    system   elapsed
# 1017.23   2.57  1044.25
```

```
# Using the law of total probability P(a) = P(a & b) + P(a & b^c) to
   get
# P(Meth<M,Unmeth>U)
joint.prob.green.2 = Meth.green.probs - joint.edf.green


# Probability points stemming from v = 1-u and the corresponding
   adjustments
quants.green.2 = bivariate.quantiles.frank.final.2(mat1.meth,
                                                   mat1.unmeth,
                                                   probs)
newQuantiles.green.2 = returnNewQuantiles2(controlMatrix = mod.mat,
                                           quantiles = quants.green
                                              .2,
                                           ncomp = 3)

edf.quantiles.green.2 = joint.edf.of.quantiles.2(mat1.meth,mat1.
   unmeth,
                                                 quants.green.2)

adjust.green.2 = adjustment(oldQuantiles = quants.green.2,
                            newQuantiles = newQuantiles.green.2)

#Second Variation Normalization
fit1.green = normalize.bivariate(mat1.meth,mat1.unmeth,
                                 joint.ecdf = joint.prob.green
```

```
                                        .2,
                                    quantiles.edf = edf.quantiles.
                                        green.2,
                                    adjustments = adjust.green.2)

# Getting the Methylated and Unmethylated signals
fitting.green.meth = fitting.green[,1:40]
fitting.green.unmeth = fitting.green[,41:80]

# beta-values
fit.beta.green = fitting.green.meth / (fitting.green.meth + fitting.
    green.unmeth +100)

#plot
densityPlot(fit.beta.green, sampGroups = type.mania)

#---- Type I Red Probes ----
mat2.meth = Meth.cor[indices[[2]],,drop=FALSE]
mat2.unmeth = Unmeth.cor[indices[[2]],,drop=FALSE]

Meth.red.probs = .getmeth.probs(mat2.meth)
system.time({joint.prob.red.1 =
  .get.bivariate.probabilities(mat2.meth,mat2.unmeth)})
#user  system elapsed
#3755.82  157.16 3977.30

# Using the law of total probability P(a) = P(a & b) + P(a & b^c) to
    get
```

```
# P(Meth<M,Unmeth>U)
joint.prob.red.2 = Meth.red.probs - joint.prob.red.1


# Probability points stemming from v = 1-u and the corresponding
   adjustments
quants.red.2 = .bivariate.quantiles.frank.final.2(mat2.meth,
                                                  mat2.unmeth,
                                                  probs)

newQuantiles.red.2 = .returnNewQuantiles2(controlMatrix = mod.mat,
                                          quantiles = quants.red.2,
                                          ncomp = 3)

edf.quantiles.red.2 = .joint.edf.of.quantiles.2(mat2.meth,mat2.unmeth
   ,
                                                quants.red.2)

adjust.red.2 = .adjustment(oldQuantiles = quants.red.2,
                           newQuantiles = newQuantiles.red.2)

# Second Variation Normalization
fit1.red = normalize.bivariate(mat2.meth,mat2.unmeth,
                                    joint.ecdf = joint.prob.red.2,
                                    quantiles.edf = edf.quantiles.red
                                       .2,
                                    adjustments = adjust.red.2)
```

```
# Methylated and Unmethylated values
fitting.red.meth = fit1.red[,1:40]
fitting.red.unmeth = fit1.red[,41:80]

# Beta-values
fit.beta.red = fitting.red.meth / (fitting.red.meth + fitting.red.
   unmeth +100)
#plot
densityPlot(fit.beta.red, sampGroups = type.mania)
```