SYSTEMATIC VARIANCE CORRECTION METHODS FOR PEPTIDE MICROARRAY DATA

MUTUA JOHN MUTISO

W62/86996/2016

A Thesis Submitted in Partial Fulfilment for the Master's Degree in Medical Statistics in the Institute of Tropical and Infectious Diseases (UNITID) in the University of Nairobi

2019

DECLARATION

This thesis is my original work, and it has not been presented in any other University.

Signature: Date:

Mutua John Mutiso

W62/86996/2016

Supervisors' Approval

This thesis has been submitted for final submission with my/our approval as supervisor(s).

1. Dr. Anne Wang'ombe, Institute of Tropical and Infectious Diseases – University of Nairobi (UNITID)

   Signature: Date:

2. Dr. Charles Sande, KEMRI | Wellcome – Trust Kilifi

   Signature:  Date: 18th November, 2019

3. Dr. Nelson Kibinge, KEMRI | Wellcome – Trust Kilifi

   Signature:  Date: 18th November, 2019

DEDICATION

I dedicate this work to my nuclear family; my wife Berndatte and our son Lincoln; our mum Joyce and sister Mercy; and my long-term friend Dan; thank you for the support and encouragements during this period since I enrolled for the master's degree. My special regards go to my grandaunt Grace Kitonyi and my late granduncle Joseph Kitonyi, for their generosity in financing my master's degree. Thank you to the Almighty God for keeping me healthy and strong during the entire period.

ACKNOWLEDGEMENTS

I wish to acknowledge the following individuals whose contribution has made it possible to develop this piece of work. Timothy Chege who worked on the peptide microarray chip as part of his post-graduate diploma work at KEMRI - Wellcome Trust and he scanned and extracted the data from the slides using GenePix microarray scanner; Elijah Gicheru and Jacqueline Waeni who worked on the assay long before I came at KEMRI – Wellcome Trust Bioscience labs. Thank you to my supervisors for the endless guidance and training meetings we had, especially in the introduction to microarrays. Thank you to the Virus Epidemiology and Control Research Group (VEC) for the research guidance and support; and to the Initiative to Develop African Research Leaders (IDeAL) and the KEMRI-Wellcome Trust as a whole for funding my master's research project.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

CGMRCCSC → Centre for Geographic Medicine Research, Coast Centre Scientific Committee

DDHF → Data-Driven Haar-Fisz Transformation

DDHFm → Data-Driven Haar-Fisz Transformation for Microarrays

DNA → Deoxyribonucleic acid

ELISA → Enzyme-linked immunosorbent assay

ELISPOT → Enzyme-linked immune absorbent spot

HT → High throughput

IgA → Immunoglobulin A

IgG → Immunoglobulin G

IgM → Immunoglobulin M

KCH → Kilifi County Hospital

KEMRI → Kenya Medical Research Institute

LMM → Linear Mixed Model

nm → Nanometre

PAS → Positive Adult-Sera

PCA → Principal Components Analysis

VSN → Variance Stabilisation Normalisation

Abstract

Study background

Protein/peptide microarrays are high throughput (HT) methods with the potential of investigating tens to thousands of probes in a single experiment. However, technical variance creates an inevitable challenge for their application, hence the need for pre-processing strategies. Most methods of correcting to the technical variance have been developed based on DNA microarrays, from which this technology was adopted; however, key chip design differences limit their direct implementation. Microarray designs are flexible, which allows researchers to customise their targets and quality control strategies, hence, there is a need for design-specific pre-processing frameworks. The broad objective of this study was to evaluate sources of technical variation in peptide microarray data and compare performances of technical variance correction methods.

Study design and site

The study was a nested non-experimental study using peptide microarray data assayed for archived plasma samples, of children and infants admitted at Kilifi County Hospital (KCH) with suspected infections. The data was used in the development of the pre-processing framework in the R software environment.

Materials and methods(s)

A peptide microarray chip targeting 49 infectious diseases was used for the assay and GenePix array scanner used for the data extraction. The analysis framework will be developed using the R programming environment.

Findings

The standard methods; local background subtraction, log transformation, combating batch effects algorithm (ComBat), variance stabilising normalisation (VSN) and linear models, did not correct the technical variance significantly from the peptide microarray data. However,

background subtraction using locally smoothed background intensities, and data scaling based on scale parameters calculated from Pooled-Adult Sera (PAS) sample fluorescence intensities achieved maximum technical variance stabilisation.

Conclusion and Recommendation

Technical variance stabilisation in peptide/protein microarray data is achievable. Morphological spot identification should be considered while estimating local background intensities, or spatial smoothing of the estimated intensities to reduce the background intensity estimation bias.

CHAPTER 1: INTRODUCTION

1.1 Background of the Study

Protein/peptide microarrays are a high throughput technology that have gained prominence in the last few decades for their ability to investigate tens to thousands of protein or peptide probes on a single slide. The development of microarrays is based on a concept that as first initiated by Roger Ekins in 1989, and the idea aimed at developing an effective platform for protein functional analysis; which would then use immense biological knowledge attained in decades of genetics and molecular biology.

Protein microarrays are grouped into; peptide microarrays, protein microarrays (purified protein or protein domains) and antibody microarrays (capture arrays) (Berrade, Garcia, & Camarero, 2011; Stoevesandt, Taussig, & He, 2009). Biochemical experiments such as protein-protein binding and enzyme-substrate relationship, biochemical activities and immune responses are investigated using the functional protein microarrays (Sutandy, Qian, Chen, & Zhu, 2013).

The 'proteomics era' has prompted the development of methodologies and technologies for quantification, identification, and characterisation of proteins functions involved in biological processes. Most of these technologies are high-throughput; therefore, extensive application of these methods in drug development and biomarker discovery research is on the rise. Protein microarrays are a great research potential for their capability to provide detailed analysis for the protein functions; which advances knowledge on chemical and biological state of cells. Further, the protein/peptide microarrays are also applied in the evaluation of quality, effectiveness and the safety of newly developed medical products, through detection of adverse events (Bertone & Snyder, 2005; Yu, Schneiderhan-Marra, & Joos, 2010).

Traditional detection methods of antibodies such as ELISA and ELISpot are limited to the analysis of single protein at an instance; however, proteomic analysis demand multiplexed technologies because of limited resources. Protein/peptide microarrays ease proteomic research because of their potential to include tens to thousands of protein/peptide probes, and process multiple study samples in a slide (Yu et al., 2010). The quality of data obtained from microarrays is controlled by control samples and probes (Gagnon-Bartsch & Speed, 2012; Kricka et al., 2009).

Besides the benefits and the promising future of the protein/peptide microarrays, technical variation is a major drawback. As a result, a number of methods and approaches are recommended to correct the technical variation in the data. Mostly, these methods have been developed under DNA microarray platforms, hence, limited research focus on protein/peptide arrays regarding correction of the technical variance. Despite the similarities of DNA and protein/peptide microarrays, there are critical design-related differences that limit direct application of most of the recommended technical variance correction methods; such as variance stabilising normalisation (VSN), Combating Batch Effects (ComBat), linear and non-linear models that have majorly been implemented in DNA microarrays.

While implementing the technical variance correction methods in the data, the identified sources of the variation are used as covariates in the models to stabilise the introduced variance. In peptide microarrays, the key sources of technical variation include; experimental differences by time of sample hybridisation or data scanning; performing assay in different laboratories; different laboratory technicians conducting the assay or processing samples in different slides (Nahtman et al., 2007; Scherer, 2009; Watson et al., 2009). Furthermore, technical variation in microarrays can also be introduced during probe miniaturisation or due to sample contaminations. As a result, technical variation in microarrays is quantified by variation observed in controls samples and control probes among arrays.

Control features in arrays such as spot replication, control probes and samples are used to evaluate effectiveness of technical variance stabilisation methods in microarrays (Lee, Kuo, Whitmore, & Sklar, 2000). Therefore, statistical techniques such as supervised clustering and kernel density plots are useful in comparing performance of different technical variance correction methods – based on their ability to maintain expected data distribution and structure between the negative and positive control samples (Gagnon-Bartsch & Speed, 2012). For example, negative and positive control samples are expected to cluster separately. Therefore this study evaluated the sources of technical variation in peptide microarray data and compared performances of technical variance correction methods.

1.2 Statement of the Problem

Peptide microarray technology is a powerful high throughput tool with the potential of investigating broad humoral immunity based on serum samples or other biological fluids such as sputum. The technology is advantageous because it uses small amounts of samples needed and the design flexibility to target infections of interest; which are investigated using miniaturised peptides. The assay is based on immunofluorescence technology; whereby, signals are expected within peptide spots due to antigen-antibody biological reactions. However, signals are not always observed within the spots due to non-specific binding. The non-specific reactions might be due to sample contamination during or before the assay; quality-related issues with the miniaturisation process; or due to sample-specific factors. The non-specific binding might vary because of the discussed sources of technical variation in microarrays. Therefore, the observed foreground signal is due to the biological antigen-antibody reactions with some influence of the non-specific binding with in the background signal. Among other quality control features, empty spots (miniaturised by buffer only) are included in the microarray chip design to infer the amount of non-specific binding experienced in a mini-array.

The non-specific binding, which is influenced by other sources of technical variation is a significant source of unwanted variation based on data obtained from microarray in previous studies. For instance, variation in experimental factors such as time and laboratory location, slides and laboratory technician, or data extraction machine may introduce systematic bias in the data. The systematic variances ought to be corrected before using the data to answer biological questions. Several methods of correcting systematic variation in microarray data are recommended; most of them are based on statistical approaches of analysis of variance. However, there is no consensus on the best method to adopt, based on its ability to remove the technical variation from microarray data, hence a significant drawback for the application of microarray technology.

1.3 Justification of the study

Microarray designs are flexible, which allow researchers to determine the targets (genes for DNA and antigens for the case of peptide/protein) and the control features to incorporate in the chip. For this reason, specific data pre-processing framework is needed for each microarray chip design developed for specific research work. This specific pre-processing framework allows effective use of the quality control features used in the design. Having a standard microarray data pre-processing framework is a challenge because of these design-related factors. Therefore, adopting existing pre-processing frameworks might not stabilise the technical variance as desired because of key differences in design. There are a number of methods shown to correct for technical variation, especially data from DNA microarrays (Chen et al., 2011; Motakis, Nason, Fryzlewicz, & Rutter, 2006; Sboner et al., 2009). However, their application to peptide/protein microarray data would require critical changes, to map the chip design features. Ultimately, developing the peptide microarray data pre-processing framework that corrects for the technical variation will enhance reproducibility and application of the technology in research (Díez et al., 2012).

1.4 Study Questions

i. What are the sources of the technical variation in the peptide microarray data?

ii. Which method(s) effectively correct the technical variation in the peptide microarray data?

1.5 Research Objective

1.5.1 General Objective

The main objective of this study was to evaluate sources of technical variation in peptide microarray data and compare performances of technical variance correction methods.

Specific objectives

i. To evaluate the sources of technical variation in the peptide microarray data.

ii. To compare different methods of correcting the technical variation in the peptide microarray data.

iii. To compare the normalized and non-normalized data based on the best method of correcting technical variation in the peptide microarray data.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In this chapter, systematic variance correction methods used in microarray data are reviewed. Although this study focuses on peptide microarray data, the methods are reviewed regardless of whether they have been applied to DNA or protein microarrays because the technologies are based on the immunofluorescence technique.

2.2 Batching

Batching is a major source of systematic variation in microarrays, and it has been reported that experimental factors such as time, location, personnel or chip might lead to significant differences in expression levels. Therefore, systematic variance correction strategies need to be applied before analysing the data. In a study done by Watson et al. (2009) on comparison of data normalisation methods used by the EADGENE network established that, based on positive control genes, the distribution of the gene expression values had systematic variation across ten arrays. In their study, they simulated ten arrays by assuming homogeneity of variance across the arrays. In a similar study conducted by Lazar et al. (2013), they found significant differences in distributions of randomly selected expression data for the same gene in two lung cancer studies of the multiple microarray gene expression (MAGE) project.

Kupfer et al. (2012) did a study which focused on evaluating the effect of removing batch effects from microarray data on gene expression differentiation, where the date of sample acquisition was regarded as a source of systematic variation. Using ComBat – an empirical Bayes algorithm implemented in R software, they were able to significantly reduce the effect of technical variation. Based on hierarchical clustering, they observed clusters of rheumatoid arthritis and osteoarthritis groups – confounded by the date of acquisition batching. Similarly, Chen et al. (2011) recommend batch-effect correction in microarray data. In their study, they

aimed at finding a batch adjustment method that would correct variation caused by batching either due to reagents, changes in technicians, scanner effects or environmental conditions and so forth. Among five evaluated methods of correcting for batching, ComBat method stabilised the variation significantly.

2.3 Microarray Data Transformation and Normalisation Methods

Several methods of correcting and stabilising variance in microarray data have been recommended from previous research, especially in gene expression studies. these methods vary from simple scaling such as log transformation methods for more sophisticated statistical methodologies such as mixed models.

2.3.1 Log_2 Transformation

Log transformation is widely used, especially in biological studies, to reduce data variation and make data conform to a normal distribution. This method has also been applied in microarray data to reduce variation (Quackenbush, 2002). A small constant is added before applying log transformation to minimise missing values since negative values are observed, especially when local background correction is applied leading to zero or negative values for spots with same or higher values of the background (Feng et al., 2014).

2.3.2 Quantile Normalisation

Studies have reported that quantile normalisation works is effective in standardising variance and scaling data across arrays. Pan and Zhang (2018) did a study that focused on applying molecular signatures on several datasets. Since each dataset was obtained in different studies, the researchers aimed at removing the inter-study variation. They argued that sources of unwanted variation were unknown – limiting the application of linear models to determine the unknow variation. Therefore, quantile normalisation was used to remove the inter-slide variation, hence reducing classification errors based on the combined dataset.

Qiu et al. (2013) conducted a study to evaluate the impact of rank and quantile normalisation methods on testing power in gene differential analysis. They found that normalising gene expression data before analysis has potential influence on the findings. Comparing the non-normalised data with rank and quantile normalised data, the number of observed true positives had higher standard errors compared with the normalised.

Qiu et al. (2013) performed a study to evaluate the impact of quantile and rank normalisation methods on testing the power of gene differential analysis. Both rank and quantile normalisation improved the power of differential expression analysis. As the effect size increase, the testing power, for instance the number of observed true positive, converge to fixed numbers – which confirms theoretical understanding.

2.3.3 Linear Models

Espín-Pérez et al. (2018) did a comparison study for the performance of statistical methods in correcting batch effects in transcriptome data. They found that linear mixed models (LMM) and ComBat were not significantly different in stabilising the batching variance. However, ComBat had higher sensitivity and specificity than LMM. On the other hand, LMM identified stronger relationships between gene expression and big effect sizes.

Reilly and Valentini (2009) proposed the application of either a linear model with and without interaction effect to correct for systematic variation in spots for both peptide and controls spots. Similarly, Sboner et al. (2009) applied a robust linear model with array, subarray and protein feature as the covariates to normalise the data. They used only the array and subarray effects as the sources of the unwanted variation during prediction, while keeping the variation between protein features. They control protein to estimate the inter- and intra-slide normalisation by comparing with different methods of technical variance correction. Robust linear model performed better in normalising the data compared with global and quantile normalisation methods.

2.3.4 Data-Driven Haar-Fisz Transformation for Microarrays

Motakis et al. (2006) established that the DDHF method was able to stabilise variance and produce fluorescence intensities that assumes normal distribution better compared to other existing methods such as log, generalised log and spread-versus-level plot transformation. Further, they affirmed that the method has a wider range of applicability on the various distribution as much as they have an increasing mean-variance dependence. This method can be applied to microarray data using the DDHFm R package. In comparison with log, generalised log and spread-versus-level plot transformation method, DDHFm strengthens the replicate correlation more efficiently compared to the other methods. This decision is based on the assumptions that the correlation values between the replicates should there was consistency by spot replication.

2.3.5 Variance Stabilizing Normalisation

Variance stabilising normalisation is shown to reduce technical variance better than standard methods such as local background subtraction among other model based methods (Ritchie et al., 2007). In another paper, Thomassen et al. (2009) highlighted that application of standard method of correcting for systematic variance, VSN being among them, worsened signal-to-noise ratio. However, this method has been used to stabilize variance in microarray data in later research (Kamuyu et al., 2018).

Table 1: Examples of studies working methods of correcting systematic variation in microarray data

Author	**Type of microarray**	**Sources of variation**	**Methods of systematic variance correction**	**Applied methods**
(Espín-Pérez et al., 2018)	DNA microarray	Batching	• Local background correction • Linear mixed models (LMM) • Linear models • ComBat	There were small differences between the performance of LMMs and ComBat ○ ComBat identified more true and false positives.
(Gagnon-Bartsch & Speed, 2012)	DNA microarray	Batching with unknown factors	• Background correction • Quantile normalisation • Location and scale adjustment • Remove Unwanted Variation, 2-step (RUV-2) was introduced and compared to the existing method • Combating Batch effects (ComBat) • Surrogate Variable Analysis (SVA).	• The RUV-2 performs better than ComBat and ordinary least squares
(Nahtman et al., 2007)	Peptide microarray	Batching Unspecific binding	• Linear mixed models (LMM) • Log-ratio (base 2) between foreground and background	The LMM allows estimation of the various sources of variability in the peptide microarray data

Author	Type of microarray	Sources of variation	Methods of systematic variance correction	Applied methods
(Pan & Zhang, 2018)	DNA microarray	Batching	• Quantile normalisation • Remove unwanted variation model	Quantile normalisation performed better by correcting of inter-dataset variation.
(Sill, Schröder, Hoheisel, Benner, & Zucknick, 2010)	Antibody microarray	Within-array variation	• Modified rank-invariant selection algorithm (In-vMod) • Global loess normalisation • Variance stabilising normalisation (VSN) • Rank-invariant selection algorithm (InvTseng) • Rank difference weighted global loess (RDWGL) • The Generalized Procrustes Analysis (GPA) - a least-squares method.	• Modified rank-invariant selection algorithm (In-vMod) outperforms the other normalisation methods ○ Selecting non-differentially expressed genes were house-keeping genes are not available ○ Use linear instead of local regression to reduce the effect of extreme values

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Study Type and Design

This study is a nested non-experimental study which aims at comparing and assessing methods of removing technical variability in data obtained from a microarray chip. The study is nested in the 'Identification of molecular signatures of serious acute infections in children' study.

3.2 Study Site

The serum samples used were extracted from blood samples of infants and children who were presented at Kilifi county hospital (KCH). Kilifi county is located in the northern coastal region of Kenya.

3.3 Study Population

The samples used in this study were from infants and children aged 25 days to 18 months who were presented at KCH with symptoms of suspected infections.

3.4 Study Samples

Archived plasma samples for infants and children who were admitted at the paediatric ward of Kilifi County Hospital (KCH) with community-onset of suspected infections were used in the peptide microarray assay.

3.5 Design of the Microarray Chip

The microarray chip has 24 mini arrays and 240 (20 rows by 12 columns) spots per mini array. In the mini array, 98 distinct peptides are investigating antigen-antibody interactions of 49 infectious diseases which are printed in duplicates. Further, IgG commercial control peptide and landmark control peptides for the IgG, IgA, and IgM have also been printed in duplicates. Also, 36 spots within the microarray were left blank to provide fluorescence information on background fluorescence.

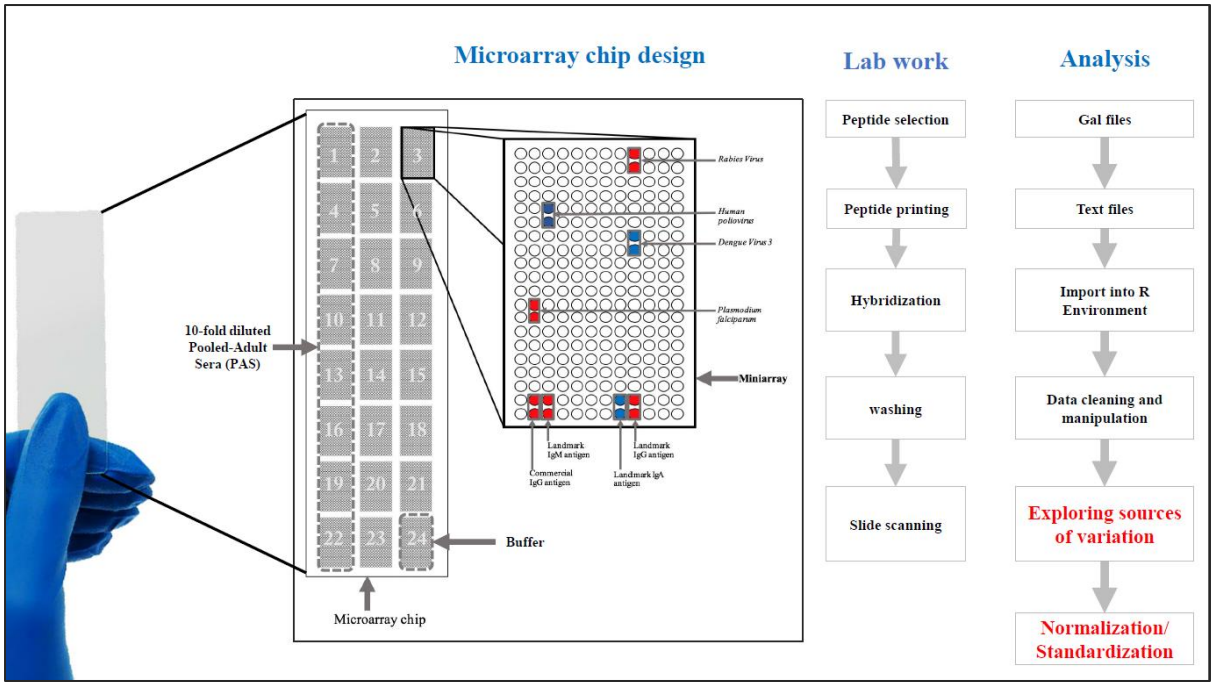

Figure 1: A graphic depicting the overall study design; showing the peptide microarray chip design, the lab assay flow diagram and the data analysis flow diagram

3.6 The Microarray Immunoassay Design

In each mini-array, either serum, positive or negative samples were incubated. All the eight mini arrays in the first column of the microarray chip were dedicated to the PAS; which was the positive reference sample. A decreasing 10-fold concentration level approach was used for the PAS samples, which were incubated from the first to the last mini arrays within the first column. Also, a mini array was set for the buffer (the negative reference sample) in each slide. Eleven slides were assayed by incubating the 161 serum samples to investigate IgG and IgA classes of antibodies. Further, additional eleven slides were assayed to for the IgM antibodies.

3.7 Data Extraction

GenePix 4300 microarray scanner with a GenePix Pro software version 7.3.0.0 was used to extract data from the incubated, dried and electronically saved slides. Different scanning power and wavelengths were used to extract spot fluorescence levels for the IgG, IgA, and IgM. For the reactivity of IgG antibodies, 40% scanning power and the data extracted from the red

channel (635nm). Data on the reactivity of IgA antibodies were scanned using 100% scanning power, and data was extracted from the green channel (532nm) from the same slides. For the IgM antibodies, different slides were used and were scanned using 40% scanning power and data extracted from the red channel (635nm). The GenePix scanner produced GenePix Array Lists (.gal file extension), which were converted into text files with a .txt (TEXT format) file extension.

3.8 Data Management

Data was imported into the R programming environment, it was cleaned and managed, transformed and analysed. Sample identifiers were created from the microarray chip design; hence no participants' personal information was used.

3.9 Applied Data Transformation and Normalisation Methods

The technical variance corrections methods that were evaluated on the peptide microarray data are described in this section. In these methods, the spot median fluorescence intensity (MFI) is the dependent variables and the potential sources of technical variation are used as the covariates. The methods include; $\log_2$ transformation, local background subtraction, Combating Batch Effects algorithm (ComBat), Variance Stabilising Normalisation (VSN), quantile normalisation, Data-Driven Haar-Fisz transformation (DDHF), and linear models. More than one method can be combined to achieve the required technical variation stabilisation.

3.9.1 Log transformation

Log transformation reduces skewness in data and the transformed data conforms to normality. Also, it is a simple scaling normalisation method that reduces the data variation. However, log transformation is only limited to non-zero positive values, where log transformation of zero and negative value leading to infinite numbers and non-numbers respectively. A small constant

is added to all the values before log transformation to reduce the number of missing values resulting due zeros and non-positive numbers.

3.9.2 Local background subtraction

Local background subtraction is a standard method of correcting non-specific binding in immunofluorescence assays. The observed fluorescence intensity in a spot is additively influenced by the background intensity as shown below.

$$observed\ spot\ intensity = true\ intensity + background\ intensity$$

Therefore, an increase in the background intensity, the observed fluorescence intensity in a spot increase at the same rate. As a result, subtracting the observed background intensities from the spots intensities is assumed to result to less biased values that are closer to the true intensities. Therefore, the corrected intensities are then log transformed to reduce the variation and bring the data distribution closer to normality.

3.9.3 Combating Batch Effects (ComBat) Algorithm

The ComBat-based normalisation method assumes a Location and Scale (L/S) adjustment model, described by Johnson et al. (2007). This batch correction algorithm is implemented in R via the sva package through the ComBat function. It adjusts data for known batching factors, using either parametric or non-parametric empirical Bayes frameworks. Therefore, it estimates the prior batch probabilities from the observed fluorescence data. The ComBat algorithm is developed based on the following model.

$$Y_{ijk} = \alpha_k + X\beta_k + \gamma_{ij} + \delta_{ij}\varepsilon_{ijk}$$

Where:

$i \sim 1,2,3, \dots, 240$ spots

$j \sim 1,2,3, \dots, 12$ arrays

$k \sim 1,2,3, \dots, 24$ mini-arrays

Y_{ijk} ~ $\log_2$ transformed MFI intensity

α_i ~ the overall MFI intensity.

X ~ design matrix for slide-level (array-level) batching

β_j ~ Regression coefficients corresponding to X

$\gamma_{ij}\ and\ \delta_{ij}$ ~ Represent additive and multiplicative batch effects of j^{th} batch for i^{th} spot.

The errors ε_{ijk}, assume normal distribution and a constant variance

3.9.4 Quantile normalisation

Quantile normalisation is a method used to standardise distribution of data from two or more distributions. This method assumes that data obtained from each processed sample follow similar distributions. Therefore, by applying the quantile normalisation, distributions of the observed data in all samples are coerced to an average distribution (Hicks & Irizarry, 2014). Below are steps followed in implementing the quantile normalisation in peptide microarray data.

- Spot fluorescence intensities are sorted either ascending or descending in each of the samples (each mini-array).
- Average values are calculated for each rank
- The spot fluorescence intensities are replaced with the average values in each rank
- The resulting data is returned to its original format, hence, quantile normalised.

3.9.5 Variance Stabilising Normalisation (VSN)

VSN assumes a measurement model that has both multiplicative and additive error terms shown below.

$$Y = \propto + \mu e^n + \varepsilon$$

Where;

Y ~ The observed intensity

$\propto$ ~ Intensity offset

μ ~ an intensity without error in arbitrary values

n ~ multiplicative error term

ε ~ additive error term

VSN is a combination of two components: (i) affine transformation which calibrates the systematic factors, and (ii) generalized log (equivalent to $\log_2$ for large intensities) to stabilise the variance (Huber, 2004).

The affine transformation is as shown below:

$$x^* = \frac{x - a}{s}$$

x^* – transformed MFI intensity

x – raw MFI intensity

a – shifting factor

s – scaling factor

Different scaling and shitting factors used for each column, but the same for all rows within a column. For stratified VSN normalisation, different scaling and shifting factors are used for different groups of rows according to the defined categorical variables. In R, VSN is implemented in the VSN and limma (linear models for microarrays) R packages.

3.9.6 Data-Driven Haar-Fisz Transformation (DDHF)

Data-Driven Haar-Fisz Transformation (DDHF) is a data transformation method that is part of Haar-Fisz variance stabilisation methods introduced by Fryzlewicz and Nason (2004). This method is applied on data with monotone increasing mean-variance dependence – a characteristic of microarray data. The DDHF method works effectively when the data is arranged according to mean sequence. In practice, the fluorescence intensities need to be sorted

based on increasing replicate means; and it assumes that the observed mean of replicates define their true ordering. This method has been applied in R programming environment under the DDHFm package.

3.9.7 Linear Models

Technical variance in microarray data can also be corrected using linear models to predict spot intensities by using arrays, subarrays, replicates, control samples and blocks as either fixed effect or random effect variables. Reilly and Valentini (2009) recommended application of linear models, with and without interactions, with array, mini-array and blocks as covariates in removing technical variation.

$Y_{ijk} = \alpha_i + \beta_j + \gamma_k + \varepsilon_{ijk}$

Y_{ijk} ~ observed spot intensity at the slide i, mini-array j and block k.

α_i ~ the slide (array) effect

β_j ~ the mini-array effect

ε_{ijk} ~ the residual signal, with assumed normal distribution, a mean of zero and constant variance.

Linear mixed effects models can also be applied to correct the technical variance. According to Espín-Pérez et al. (2018), arrays and mini-arrays among other assay design related factors can be used as covariates in a mixed model aimed at correcting the observed technical variance. The technical model is modelled as shown in the equation below.

$$Y = Xb + Zu + \varepsilon$$

Y ~ $\log_2$ transformed MFI intensities

X ~ Design matrix of the fixed variables

b ~ Fixed effects

Z ~ Design matrix of the random variables

u ~ Random effects

ε ~ Residual

3.10 Microarray Data Quality Check

3.10.1 Principal Components Analysis (PCA)

The Principal Components Analysis (PCA) was used to evaluate data quality based on the control reference samples and spots. For instance, there were 36 buffer spots in each mini-array aimed at estimating non-specific binding, hence, estimating the background fluorescence. Since the data had multiple variables that were measure on the same scale, that are assumed to produce similar fluorescence intensities, PCA was used to check quality of the buffer spots based on clustering visualised using first and second principal components.

A PCA analysis was performed for each transformation method applied, based on the peptide fluorescence intensities data. the first and second components of the PCA analysis were plotted using scatter plots to identify data clustering based on the sample type. A distinct clustering was expected between data for the negative and positive reference samples.

3.10.2 Wilcoxon Sign Rank Sum Test

The PAS sample in mini-array 1 (PAS sample with the lowest dilution) was used to compare technical variance stabilisation across slides. Wilcoxon Sign Rank Sum non-parametric method was used to test the similarity of fluorescence intensities distributions among slides at 5% level of significance.

CHAPTER 4: RESULTS

4.1 Sources of Variation

Local background subtraction and log transformation are the standard methods used in microarray data for normalisation and stabilisation of technical variance. First, the data were transformed using a log to base 2; then, the transformed data used to investigate sources of technical variation. Performances of the microarray chip were assessed by first checking the correlation of the duplicates; then the consistency of the reference samples (PAS and buffer) across slides.

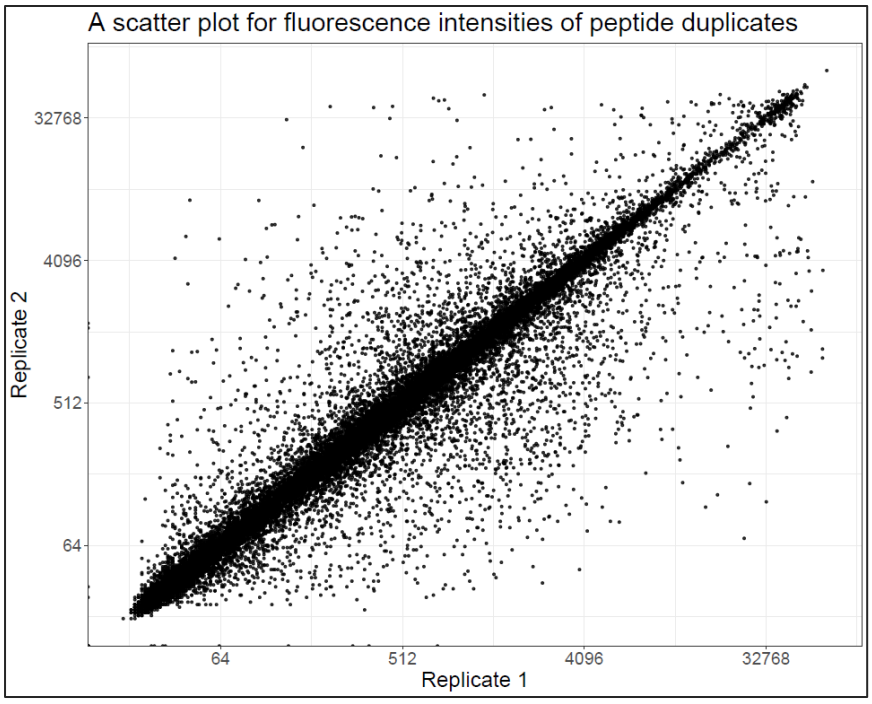

Figure 2: A scatter plot for fluorescence intensities of peptide duplicates (the axis is in log2 scale)

A strong correlation ($\rho = 0.95$; $pvalue < 0.001$) for the fluorescence intensities was found between the duplicate peptides (figure 2). Therefore, data from the duplicates were combined by taking their arithmetic mean.

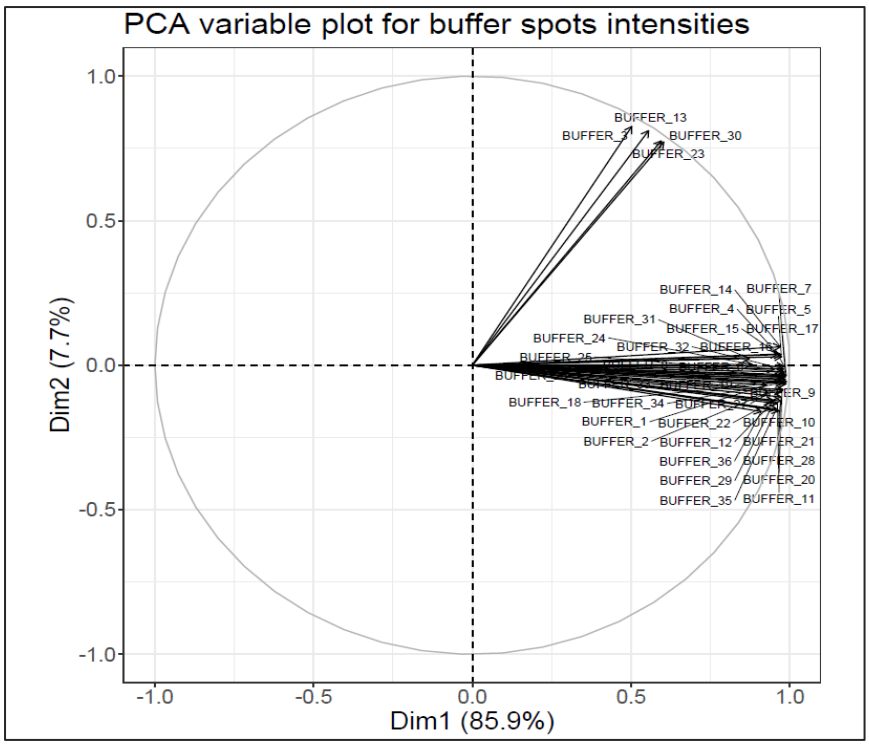

Figure 3: A PCA variable plot for fluorescence intensities of buffer spots

Quality of the buffer spots was evaluated by performing a PCA analysis on a matrix for foreground fluorescence intensities observed in the buffer spots. As shown in figure 3, some of the buffer spots clustered significantly differently; and they were found to have higher fluorescence intensities compared to the other buffer spots. Surprisingly, these buffer spots that clustered differently compared with the other buffer spots were adjacent to either commercial epitope or landmark epitope. Therefore, all the buffer spots that neighbourhood commercial epitopes or landmark epitopes were removed from the dataset.

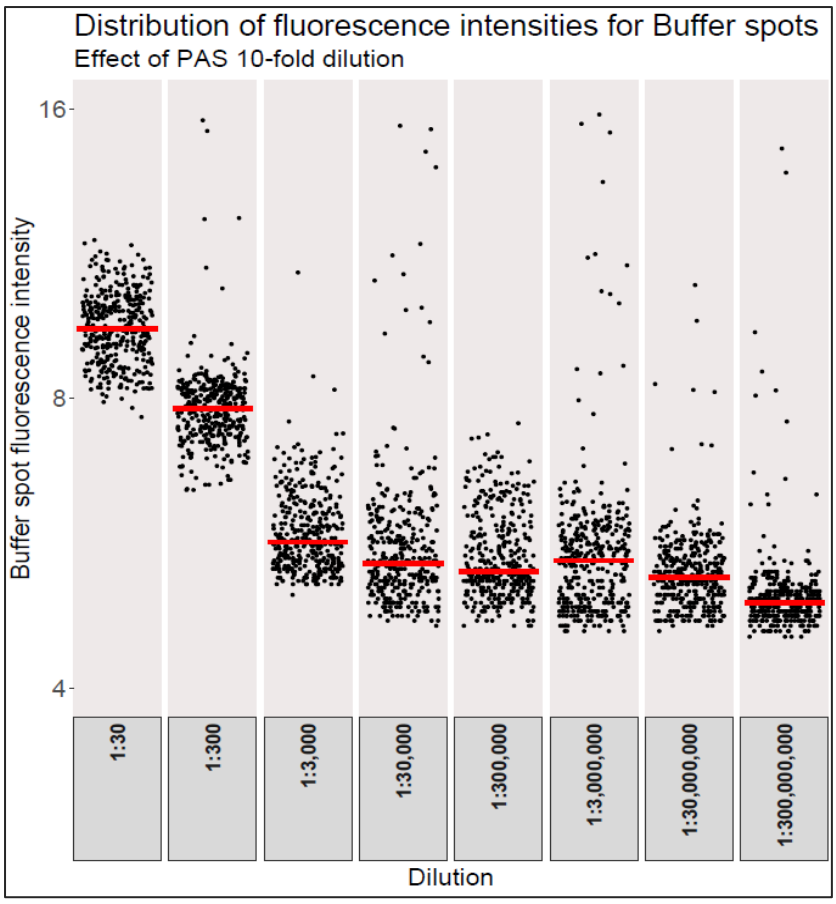

Figure 4: Effect of PAS dilution on the distribution of buffer spots fluorescence intensities

Further analysis was performed to investigate the stability of buffer spots to determine whether they are optimum in explaining non-specific binding. A comparability plot shows that the distribution of buffer spots fluorescence intensities reduce as the dilution of the PAS increases (figure 4).

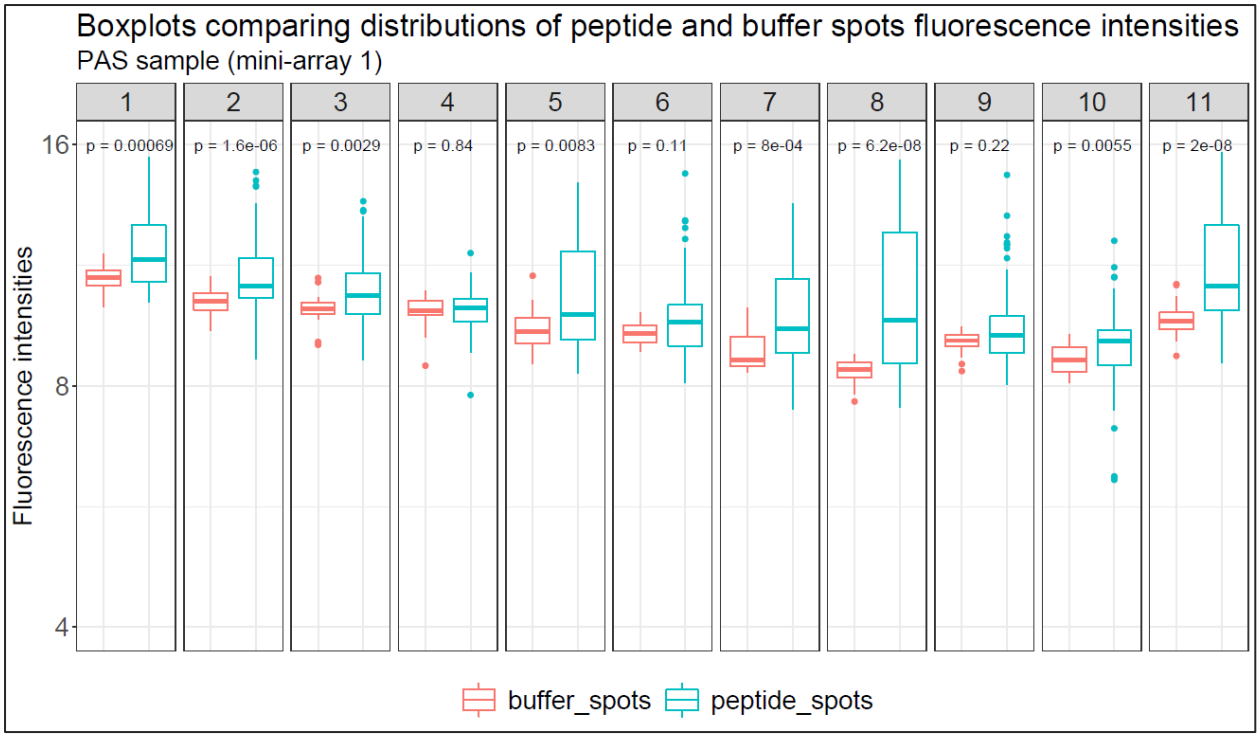

Figure 5: Boxplots comparing distributions of peptide and buffer spots fluorescence intensities across slides for the PAS sample in mini-array 1 (Wilcoxon Sign Rank test p-values included)

The distributions of fluorescence intensities of peptide spots were compared with those of buffer spots across slides for both PAS and buffer samples. In a number of the slides, the distributions of the peptide spots fluorescence intensities were not statistically different with the distribution of the fluorescence intensities of buffer spots in the PAS (mini-array 1; see figure 1) sample; however, the median statistics of the peptide spots were consistently higher (figure 5).

In the buffer sample, the distribution of fluorescence intensities for peptide spots was significantly different from the distribution of the buffer spots at 5% level of significance.

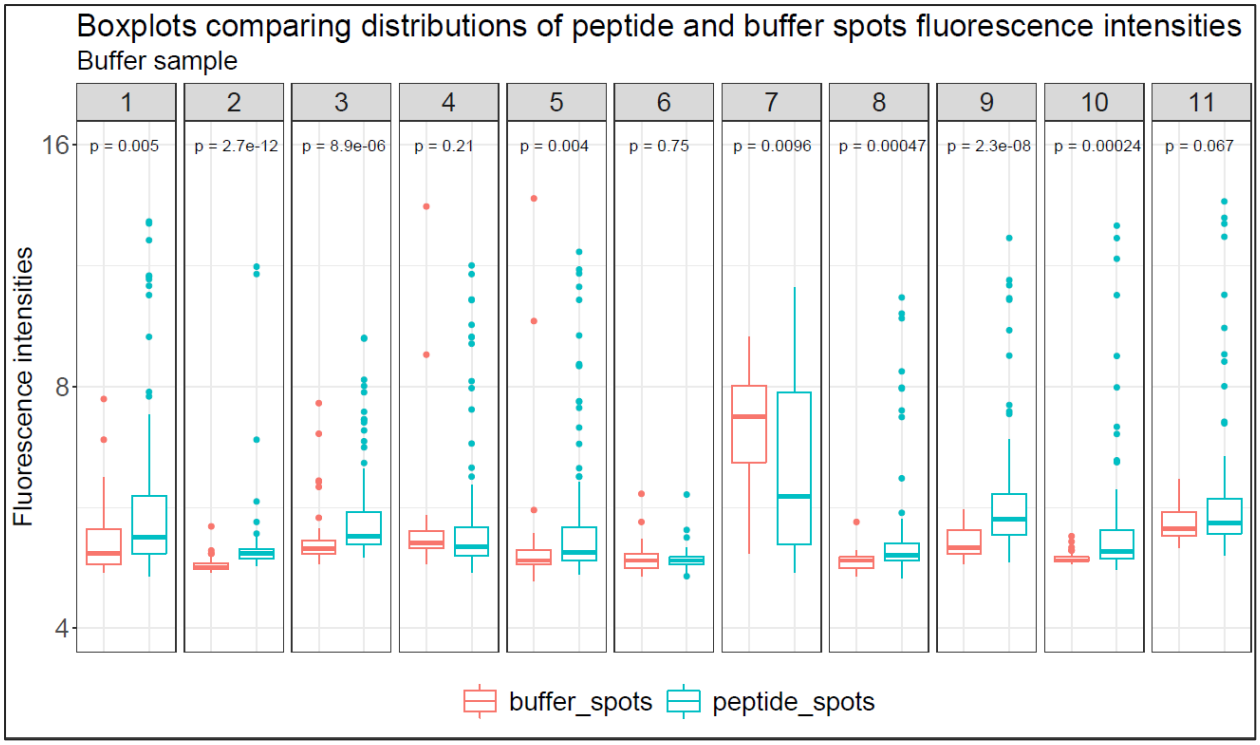

Figure 6: Boxplots comparing distributions of peptide and buffer spots fluorescence intensities across slides for the buffer sample (Wilcoxon Sign Rank test p-values included)

However, the distribution of fluorescence intensities for the peptide spots was not consistently higher across slides (figure 6). A PCA analysis was conducted to determine whether sample type influenced the fluorescence intensities of the buffer spots. As shown in figure 7, the fluorescence intensities of the buffer spots clustered by sample type; PAS (mini-array 1) sample, buffer sample and the study samples.

4.2 Background Intensity Correction Methods

The local background intensities, estimated by the GenePix microarray scanner, were subtracted from the foreground intensities to correct for non-specific binding. The background intensity estimation method assumed by the spots were of fixed sizes, and they were circular. Therefore, median fluorescence intensity was estimated as the local background intensity for the areas surrounding the assumed circular spot. By subtracting the local background intensities from the foreground intensities, some spots ended up with negative intensities. While

transforming the background-subtracted data using log transformation, values less than one were fixed at a value of one, to avoid infinite numbers, non-numbers and negative values in the log-transformed dataset.

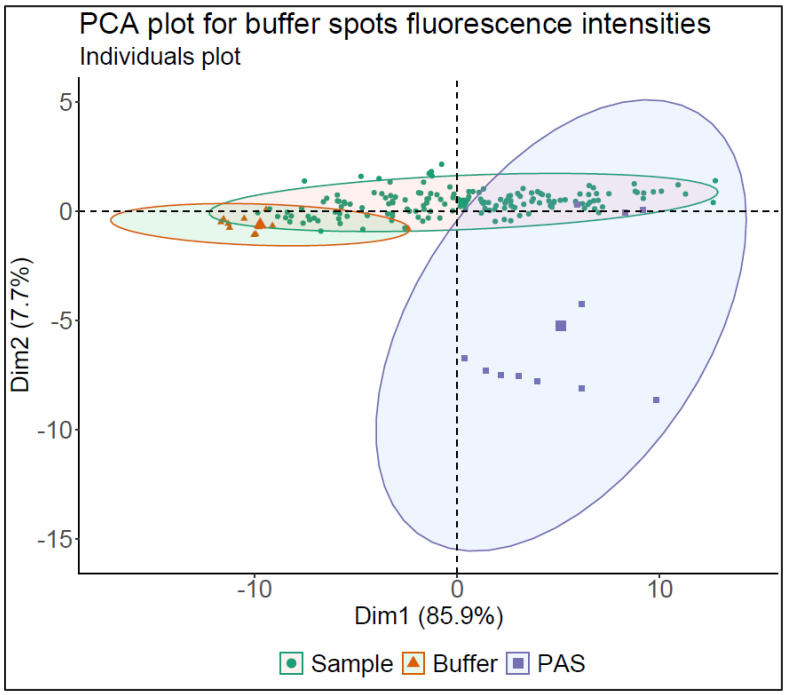

Figure 7: PCA individuals scatter plot for buffer spots fluorescence intensities; clustering by sample type (PAS in mini-array 1, buffer and study samples)

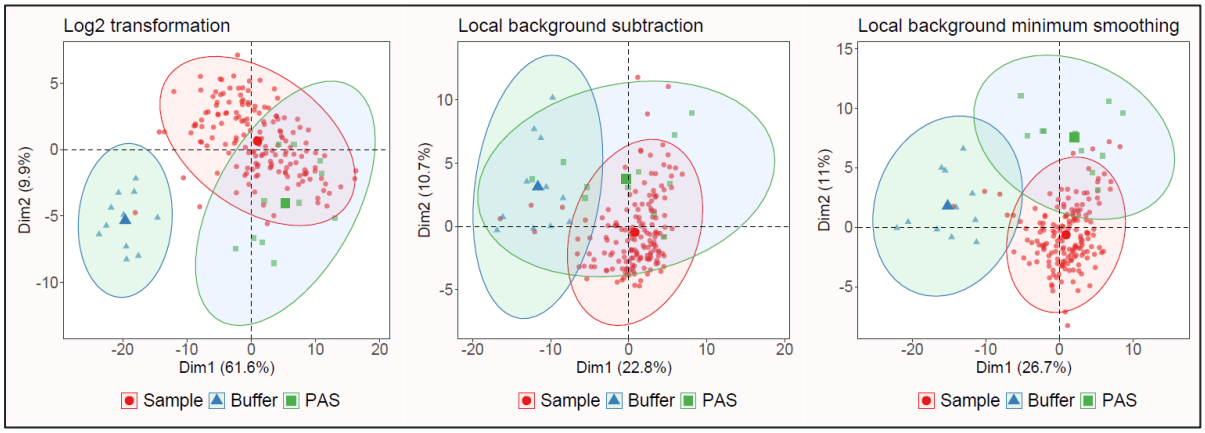

Figure 8: Individual PCA plots of log2 transformation of raw values; local background subtraction and log2 transformation; and background subtraction using local background minimum smoothing and log2 transformation

Table 2: ANOVA table for model significance test comparing with model 0. The table show model selection criteria for the linear mixed-effects method that was compared with other technical variance correction methods discussed in the methodology section.

	Fixed Variables	**Random Variables**	**Weighted**	**DF**	**AIC**	**BIC**	**Log Likelihood**	**deviance**	**Chi-square statistic**	**Chi-Square – DF**	**P-value**
Model 0	Background intensity	Sample	No	4	156411	156446	-78201	156403			
Model 1	Background intensity	Sample, Spot	No	5	124665	124708	-62327	124655	31748.10	1	< 0.001
Model 2	Background intensity, Sample type	Slide, Miniarray, Spot	No	8	124593	124662	-62288	124577	10347.03	1	< 0.001
Model 3	Background intensity, Sample type,	Slide, Miniarray, Spot	Yes (mini-array buffer median)	8	**122888**	122957	-61436	122872	1704.89	0	< 0.001
Model 4	Background intensity	Sample, Spot	Yes (mini-array buffer median)	5	122952	122995	-61471	122942	1712.92	0	< 0.001
Model 5	Background intensity, slide scanning time	Sample, Spot	Yes (mini-array buffer median)	7	122942	123002	-61464	122928	14.63	2	0.00067
Model 6	Background intensity	Slide, Sample type, Miniarray, Spot	No	8	124620	124689	-62302	124604	0	0	1
Model 7	Background intensity	Slide, Sample type	Yes (scanning time)	8	127184	127253	-63584	127168	0	0	1
Model 8	Sample type,	Slide, Miniarray, Spot	No	7	134938	134998	-67462	134924	0	0	1

Local background subtraction was found to be ineffective in removing unwanted variation in the data. According to the PCA unsupervised clustering analysis for the sample types, local background subtraction removes significant biological differences from the data (figure 8). Since local background subtraction performed poorly, other methods of background intensity estimation were sought. Buffer spots were not an option because they were found to be unreliable due to their instability by dilution and sample types. Therefore, model-based methods and spatial methods were assessed. Based on the methods' capability to retain biological differences by showing distinct clustering of the PAS and buffer samples, local background minimum smoothing provided the more reliable results (figure 8).

4.3 Normalisation Methods

Several data normalisation and technical variance stabilisation methods were evaluated. Among the methods of variance stabilisation and data normalisation evaluated include; log to base 2 transformation and ComBat algorithm for correcting batch effects, variance stabilising normalisation (VSN) and quantile normalisation, Data-driven Haar-Fisz transformation and linear mixed-effects model (LMM) (assessed LMM models shown in table 2) and a custom variance stabilisation method that uses PAS (mini-array 1; see figure 1) reference sample fluorescence intensities to calculate stabilisation factors as shown in equation (8). Across all the methods, the data was scaled in a log to base 2. Some of the methods were combined to improve their performance, while others were applied individually.

$$SF_{ij} = \frac{S_{ij}(PAS_1)}{MS_i(PAS_1)} \qquad (8)$$

SF_{ij} → Stabilisation factor for the i^{th} spot in j^{th} slide

$S_{ij}(PAS_1)$ → Background uncorrected fluorescence intensity for i^{th} spot in j^{th} slide in PAS sample in mini-array 1

$MS_i(PAS_1)$ → Background uncorrected fluorescence intensity for i^{th} spot across slides in PAS sample in mini-array 1

The variance stabilisation factor was calculated based on the observed raw spot intensities. The calculation of the factor assumed that the variation observed in the PAS sample in mini-array 1 is an overall claim of the data shift from the actual spot intensity. Therefore, the raw intensities are used to calculate the stabilisation factor because they explain maximum variance experienced.

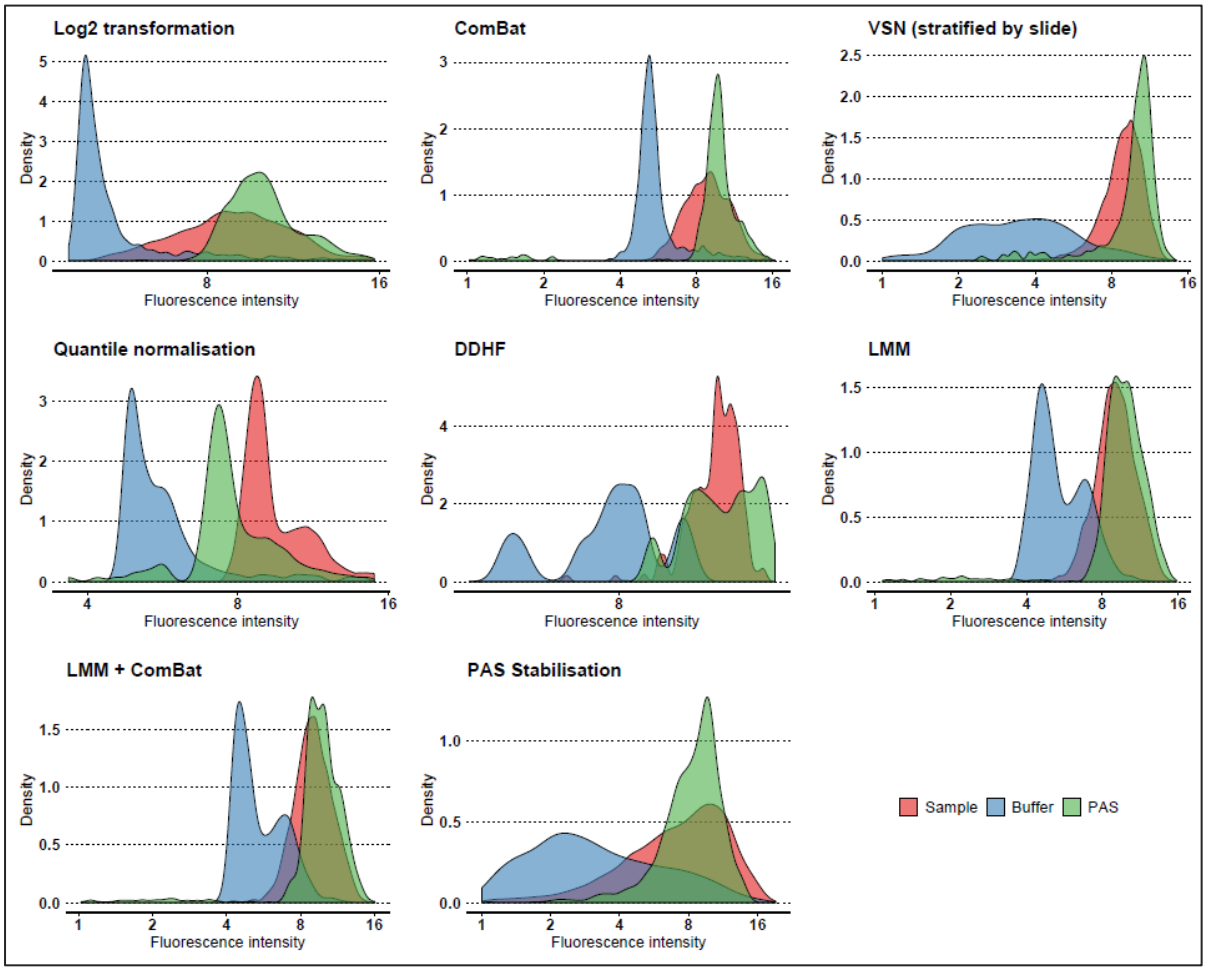

Figure 9: Distribution density plot across normalisation methods; coloured by the sample group

The distributions of the fluorescence intensities by the sample group were compared among the applied methods of technical variance stabilisation (figure 9). The distributions of

fluorescence intensities observed that the buffer sample was expected to have a significantly lower median compared to the PAS sample. The distribution of the spot fluorescence intensities for the study sample should overlap the buffer sample distribution and the PAS sample distributions. Figure 9 shows that DDHF and quantile normalisation methods did not perform well in stabilising the technical variance; the quantile normalisation estimates a significantly higher median of the study sample fluorescence intensities compared to the PAS, which is unlikely possible. DDHF method does not stabilise the variable; the data shows multi-modal distributions.

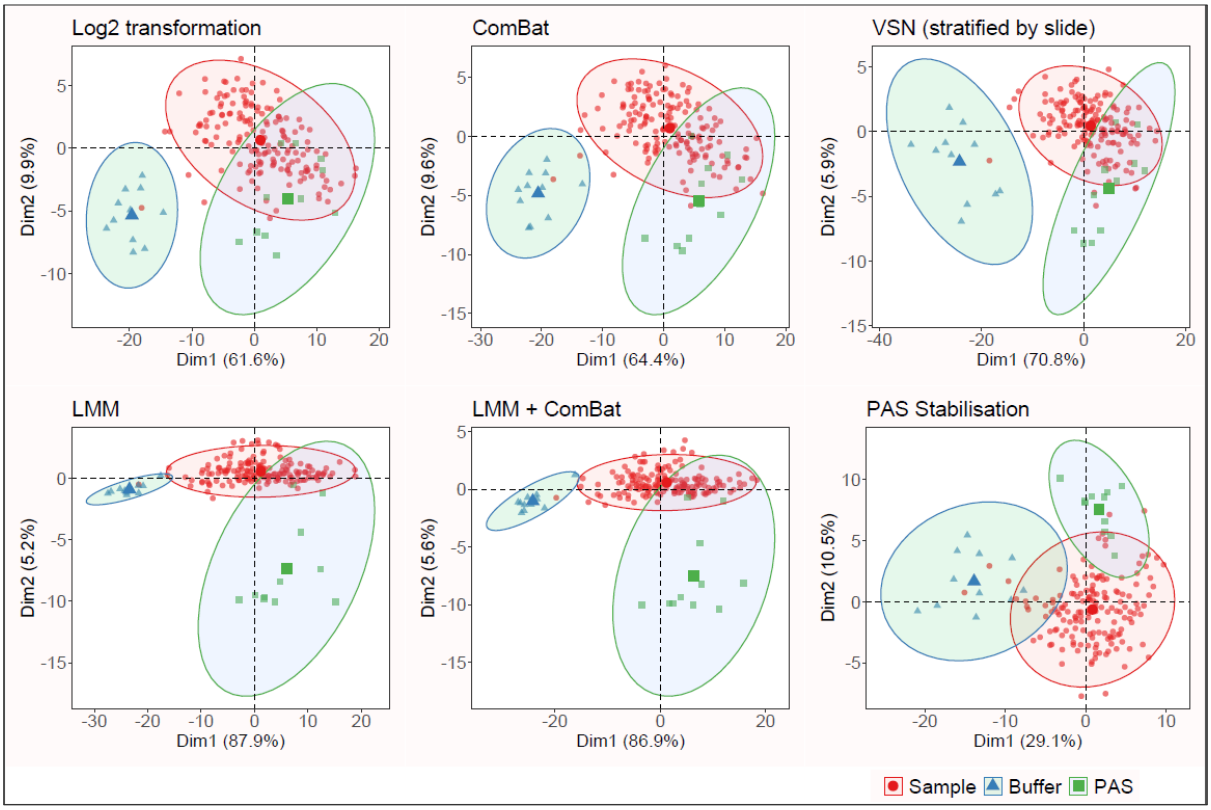

Figure 10: PCA individual scatter plots for the normalisation methods; ellipses show clustering by sample type

Figure 10 shows that the data normalisation and variance stabilisation methods retained the expected biological differences. Some study samples cluster together with the buffer and PAS samples. The amount of variation explained by the first and second principal components vary

by method. Since the biological is retained based on the methods presented in figure 10, technical variance stabilisation is assessed based on the distributions of peptide fluorescence intensities in PAS sample in mini-array 1. These distributions are expected to be the same because it is the same sample run multiple times.

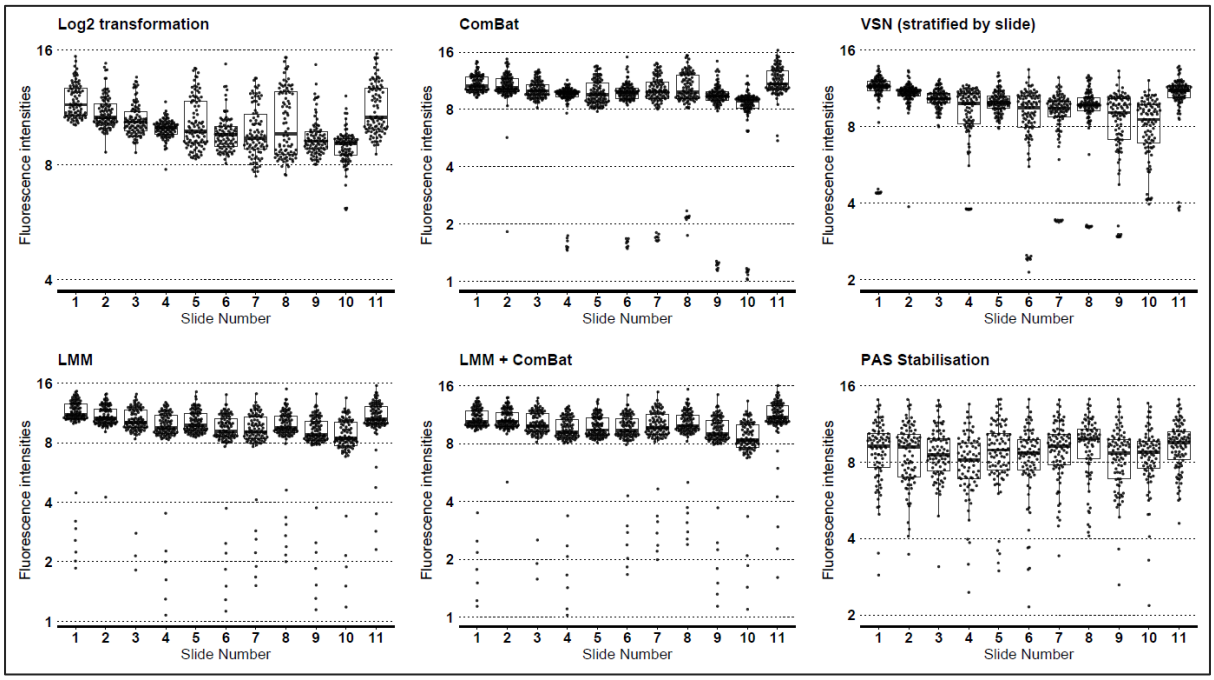

Figure 11: Distributions of peptide spots fluorescence intensities of PAS in mini-array 1; compared across normalisation and technical variance stabilisation methods

Pairwise Wilcoxon Sign Rank Sum tests, corrected for multiple testing using Bonferroni method were performed to identify slides whose peptide spots fluorescence intensities in PAS mini-array 1 are significantly different.

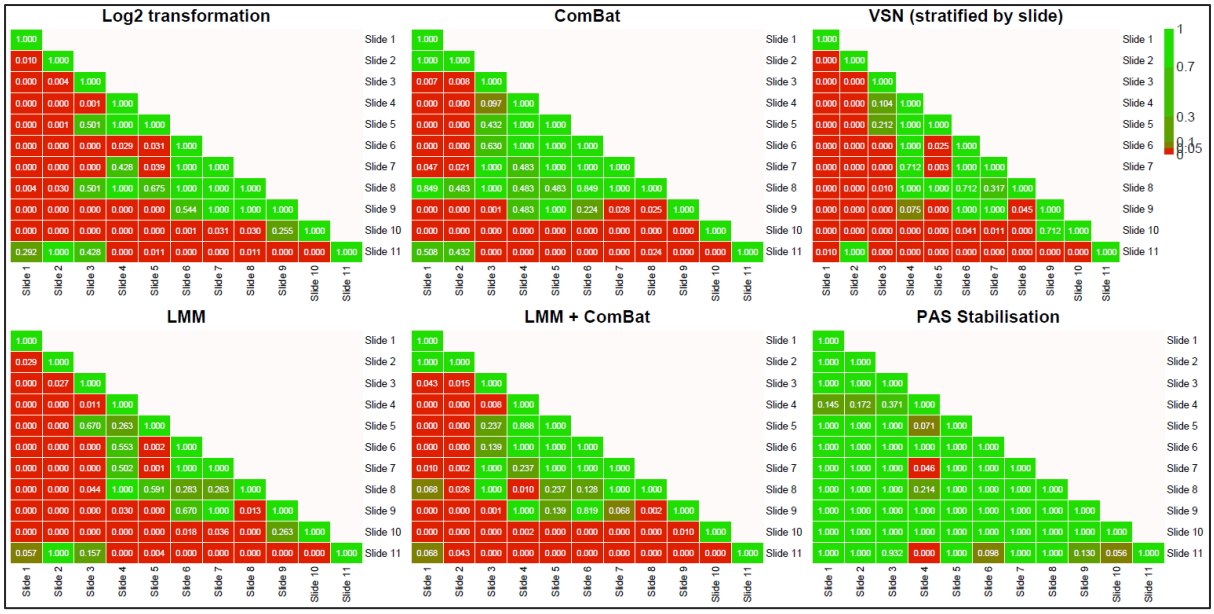

Figure 12: Pairwise comparison of peptide spots fluorescence intensities distributions in PAS in mini-array 1, using Wilcoxon Sign Rank Sum test; each box represents a Bonferroni adjusted Wilcoxon Sign Rank Test p-value; red colour shows p-values less than 0.05, and the red colour fades towards green as the p-value increases.

Figure 12 shows the pairwise comparison of distributions of peptide spots fluorescence intensities in PAS in mini-array 1; the comparison tests were done using Bonferroni corrected Wilcoxon Sign Rank Sum test for all the pairs of slides. The six methods of normalisation and technical variance correction were compared, and the PAS stabilisation method, which corrected background intensity using local background minimum smoothed fluorescence intensities and applied the stabilisation factor produced most desirable results.

4.4 Comparison between normalised and non-normalised data

Log2 transformed raw fluorescence intensities were compared with background-corrected fluorescence intensities using local background minimum smoothed, and variance stabilisation using factors calculated as shown in equation 8. As shown in figure 10, the biological variance is retained, and the variance is stabilised, as shown in figure 11 and 12. Therefore, the non-normalised data is compared with normalised data based on the effect of PAS dilution on the

distribution of peptide spots fluorescence intensities. Also, the decay of maternal antibodies, IgG, is used to compare before and after normalised.

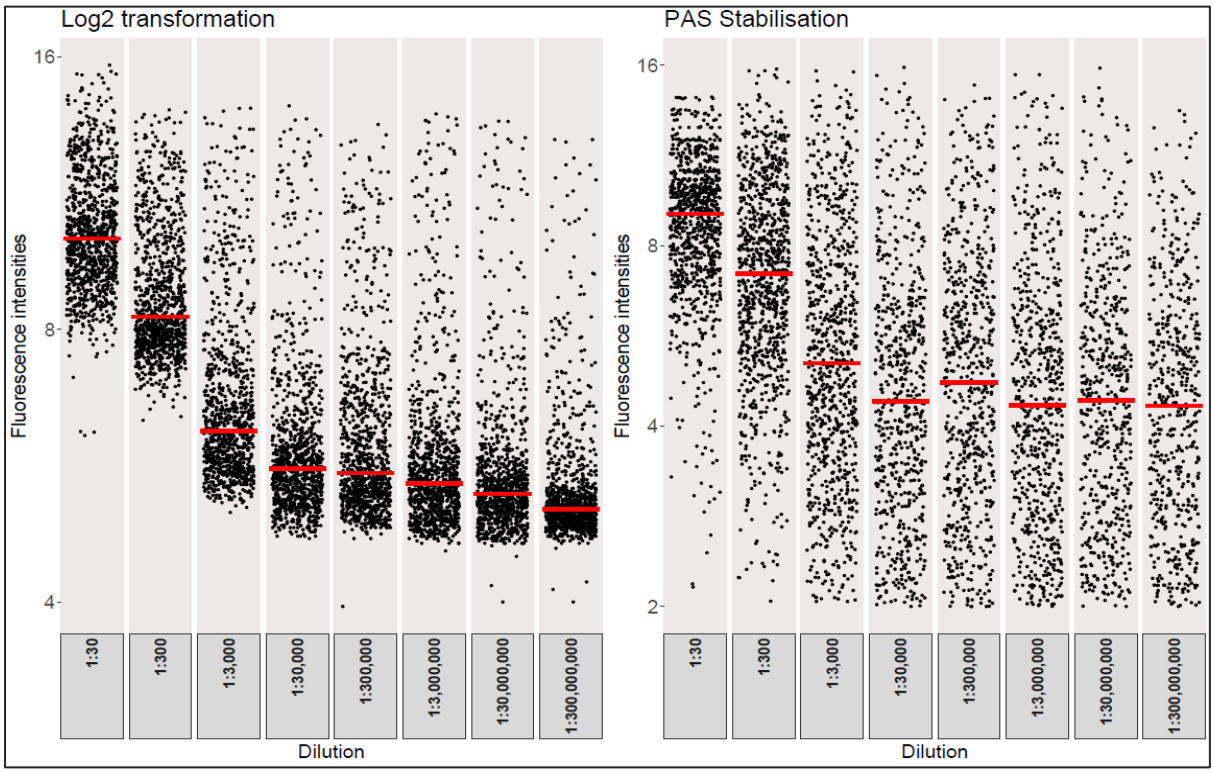

Figure 13:Distribution plots for peptide spots fluorescence intensities by dilution of PAS sample; compared before (log2 transformation) and after (local background correction using minimum smoothed background fluorescence intensities and PAS stabilisation) normalisation

Figure 13 shows that the effect of the 10-fold PAS dilution on the distribution of peptide spots fluorescence intensities is retained after the variance stabilisation.

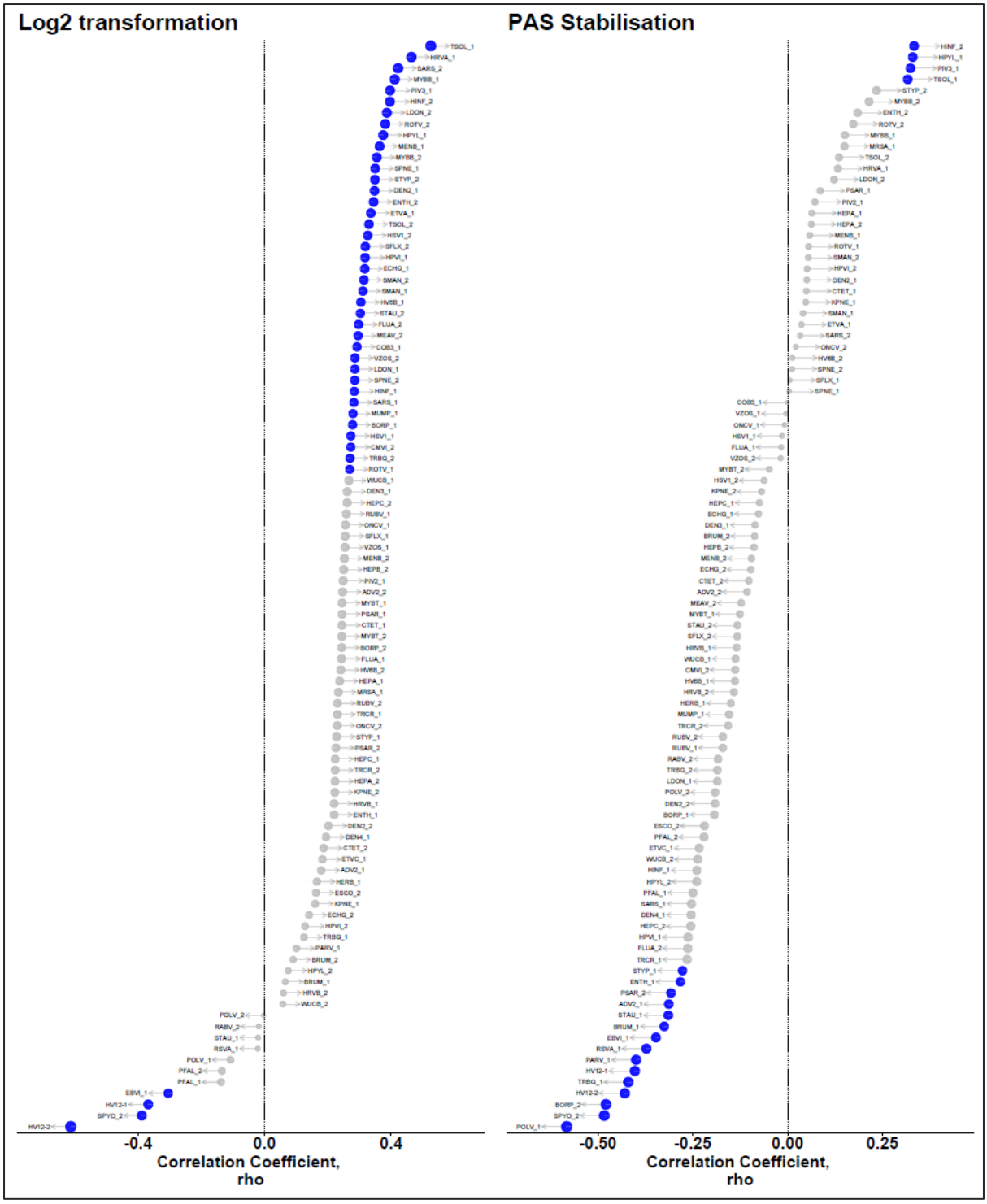

Figure 14: A plot of Spearman correlation between fluorescence intensities for IgG and the first six months of life, comparing before and after normalisation; blue colour shows that the correlation is statistically significant with 95% confidence level.

Before technical variance stabilisation and data normalisation, few peptides were observed to decay in the first six months of life. After technical variance stabilisation and data normalisation, the number of peptides indicating that maternal antibodies decayed increased significantly (figure 14).

CHAPTER 5: DISCUSSION

5.1 Discussion

The objective of this study was to apply technical variance correction methods to peptide microarray; correct the technical variance and normalise the data while retaining the biological differences. Difference approaches and methods were applied, seeking a stabilisation method that does not affect the credibility of the dataset. Further, the study was to develop a pre-processing framework that applies the suggested method, to be used on pre-processing of data obtained from the chip.

Several standard methods used in stabilising technical variance on biological data, especially DNA microarray data, were assessed. Amongst the assessed method is log transformation, background correction, linear models and batch effects correction algorithms among others. The analysis found that the technical variance influenced the dataset by mini-array and slide.

Local background correction is an effective method of reducing technical variance in microarrays while stabilising the technical variance. Therefore, local background subtraction was the first method to apply, aiming at stabilising the variance.

The local background subtraction method removed meaningful biological differences. Although it is assumed that subtracting local background corrects for the technical variation, local background estimation bias could have been introduced by the scanning machine because of the spot properties; size and shape which influences the definition of the background area (Fardin et al., 2007; Lourido et al., 2014; Yang, Buckley, Dudoit, & Speed, 2002).

Since the local background subtraction methods did not yield desirable results, other methods of background estimation were sought because it was evident that non-specific binding was a significant source of unwanted variation. Local minimum background smoothing estimated less biased background intensities compared with buffer spots estimates and NormExp

modelled background intensities (Schützenmeister & Piepho, 2010; Silver, Ritchie, & Smyth, 2009).

The standard method of correcting for technical variance in microarrays did not achieve desirable technical variance stabilisation. These methods include ComBat, log2 transformation, VSN and linear mixed model (Nahtman et al., 2007; Sill et al., 2010). The linear model stabilised the variance better compared with the other methods; however, antibody decay could not be established; hence the data was not quality.

Finally, a more data-driven approach was applied by calculating a stabilisation factor for each peptide based on the PAS sample in mini-array 1. This approach assumed that the spots immunogenicity varied by peptide, hence spot specific stabilisation factored could produce optimal results. Secondly, the method assumed that fluorescence intensities within a slide were all influenced by non-specific binding at the same level. Maximum variance stabilisation was achieved by applying the stabilisation factor on the background-corrected intensities.

5.2 Conclusion

In conclusion, the standard techniques used to stabilise technical variance in microarray did not achieve variance stabilisation. Local background smoothing performs better in correcting the effect of non-specific binding than subtracting the raw estimated local background intensities. A combination of the background correcting using smoothed intensities and the stabilisation factor calculated based on the PAS, the positive reference sample, achieved maximum variance stabilisation compared with existing methods.

5.3 Recommendations

Based on the findings of this study, identifying sources of technical variation and effectively correcting for their effect is essential before the analysis of peptide/protein microarray data. I recommend the use of local background smoothing or morphological spot detection while estimating the background fluorescence intensity. These rigorous methods of background

intensity estimation reduce the bias introduced by the assumption of constant size and circularity (or any other distinct structure) of the spot.

5.4 Study Limitations and Future Research

The peptide microarray data used in this study was processed by one laboratory technician in the same lab and in the same day. As a result, it was difficult to quantified the observed technical variation, although it was observed that the data varied by slide. Therefore, more research should be done on background correction methods for peptide microarrays to provide evidence in this area of research.

References

Berrade, L., Garcia, A. E., & Camarero, J. A. (2011, July 30). Protein microarrays: Novel developments and applications. *Pharmaceutical Research*. https://doi.org/10.1007/s11095-010-0325-1

Bertone, P., & Snyder, M. (2005). Advances in functional protein microarray technology. *FEBS Journal, 272*(21), 5400–5411. https://doi.org/10.1111/j.1742-4658.2005.04970.x

Chen, C., Grennan, K., Badner, J., Zhang, D., Gershon, E., Jin, L., & Liu, C. (2011). Removing batch effects in analysis of expression microarray data: An evaluation of six batch adjustment methods. *PLoS ONE*. https://doi.org/10.1371/journal.pone.0017238

Díez, P., Dasilva, N., González-González, M., Matarraz, S., Casado-Vela, J., Orfao, A., & Fuentes, M. (2012). Data Analysis Strategies for Protein Microarrays. *Microarrays*, *1*(3), 64–83. https://doi.org/10.3390/microarrays1020064

Espín-Pérez, A., Portier, C., Chadeau-Hyam, M., van Veldhoven, K., Kleinjans, J. C. S., & de Kok, T. M. C. M. (2018). Comparison of statistical methods and the use of quality control samples for batch effect correction in human transcriptome data. *PLoS ONE*, *13*(8), e0202947. https://doi.org/10.1371/journal.pone.0202947

Fardin, P., Moretti, S., Biasotti, B., Ricciardi, A., Bonassi, S., & Varesio, L. (2007). Normalization of low-density microarray using external spike-in controls: Analysis of macrophage cell lines expression profile. *BMC Genomics*, *8*, 1–19. https://doi.org/10.1186/1471-2164-8-17

Feng, C., Wang, H., Lu, N., Chen, T., He, H., Lu, Y., & Tu, X. M. (2014). Log-transformation and its implications for data analysis. *Shanghai Archives of Psychiatry*, *26*(2), 105–109. https://doi.org/10.3969/j.issn.1002-0829.2014.02.009

Fryzlewicz, P., & Nason, G. P. (2004). A Haar-Fisz algorithm for poisson intensity estimation. *Journal of Computational and Graphical Statistics*, *13*(3), 621–638. https://doi.org/10.1198/106186004X2697

Gagnon-Bartsch, J. A., & Speed, T. P. (2012). Using control genes to correct for unwanted variation in microarray data. *Biostatistics*, *13*(3), 539–552. https://doi.org/10.1093/biostatistics/kxr034

Guthke, R., Pohlers, D., Huber, R., Kinne, R. W., Kupfer, P., Koczan, D., … Kinne, R. W. (2012). Batch correction of microarray data substantially improves the identification of genes differentially expressed in Rheumatoid Arthritis and Osteoarthritis. *BMC Medical Genomics*, *5*(1), 23. https://doi.org/10.1186/1755-8794-5-23

Hicks, S. C., & Irizarry, R. A. (2014). When to use Quantile Normalization? *Detection of Highly Dangerous Pathogens: Microarray Methods for BSL 3 and BSL 4 Agents*, *234*(4), e15. https://doi.org/10.1101/012203

Huber, W. (2004). Robust calibration and variance stabilization with VSN. *Differences*, 1–14.

Johnson, W. E., Li, C., & Rabinovic, A. (2007). Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics (Oxford, England)*, *8*(1), 118–127. https://doi.org/10.1093/biostatistics/kxj037

Kamuyu, G., Tuju, J., Kimathi, R., Mwai, K., Mburu, J., Kibinge, N., … Osier, F. H. A. (2018). KILchip v1.0: A Novel Plasmodium falciparum Merozoite Protein Microarray to Facilitate Malaria Vaccine Candidate Prioritization. *Frontiers in Immunology*, *9*, 2866. https://doi.org/10.3389/fimmu.2018.02866

Kricka, L. J., Master, S. R., Burt, S. M., Kennedy, J. H., Holder, R. L., Halliday, M. I., … Wisdom, G. B. (2009). Quality control and protein microarrays. *Clinical Chemistry*, *55*(6), 1053–1055. https://doi.org/10.1373/clinchem.2009.126557

Lazar, C., Meganck, S., Taminau, J., Steenhoff, D., Coletta, A., Molter, C., … Nowe, A. (2013). Batch effect removal methods for microarray gene expression data integration: a survey. *Briefings in Bioinformatics*, *14*(4), 469–490. https://doi.org/10.1093/bib/bbs037

Lee, M. L., Kuo, F. C., Whitmore, G. A., & Sklar, J. (2000). Importance of replication in microarray gene expression studies: statistical methods and evidence from repetitive cDNA hybridizations. *Proceedings of the National Academy of Sciences of the United States of America*, *97*(18), 9834–9839. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/10963655

Lin, S. M., Du, P., Huber, W., & Kibbe, W. A. (2008). Model-based variance-stabilizing transformation for Illumina microarray data. *Nucleic Acids Research*, *36*(2), e11–e11. https://doi.org/10.1093/nar/gkm1075

Lourido, L., Sanidade, C. De, Dasilva-freire, N., Ruiz-romero, C., Marko-Varga, G., & Wang, X. (2014). *Genomics and Proteomics for Clinical Discovery and Development*. (G. Marko-Varga, Ed.) (Vol. 6). Dordrecht: Springer Netherlands. https://doi.org/10.1007/978-94-017-9202-8

Motakis, E. S., Nason, G. P., Fryzlewicz, P., & Rutter, G. A. (2006). Variance stabilization and normalization for one-color microarray data using a data-driven multiscale approach. *Bioinformatics*, *22*(20), 2547–2553. https://doi.org/10.1093/bioinformatics/btl412

Nahtman, T., Jernberg, A., Mahdavifar, S., Zerweck, J., Schutkowski, M., Maeurer, M., & Reilly, M. (2007). Validation of peptide epitope microarray experiments and extraction of quality data. *Journal of Immunological Methods*, *328*(1–2), 1–13. https://doi.org/10.1016/j.jim.2007.07.015

Pan, M., & Zhang, J. (2018). Quantile normalization for combining gene-expression datasets.

Biotechnology and Biotechnological Equipment, *32*(3), 751–758. https://doi.org/10.1080/13102818.2017.1419376

Qiu, X., Wu, H., & Hu, R. (2013). The impact of quantile and rank normalization procedures on the testing power of gene differential expression analysis. *BMC Bioinformatics*, *14*(1), 124. https://doi.org/10.1186/1471-2105-14-124

Quackenbush, J. (2002). Microarray data normalization and transformation. *Nature Genetics*, *32*(4S), 496–501. https://doi.org/10.1038/ng1032

Reilly, M., & Valentini, D. (2009). Visualisation and pre-processing of peptide microarray data. In *Methods in Molecular Biology* (Vol. 570, pp. 373–389). https://doi.org/10.1007/978-1-60327-394-7_21

Ritchie, M. E., Silver, J., Oshlack, A., Holmes, M., Diyagama, D., Holloway, A., & Smyth, G. K. (2007). A comparison of background correction methods for two-colour microarrays. *Bioinformatics*, *23*(20), 2700–2707. https://doi.org/10.1093/bioinformatics/btm412

Sboner, A., Karpikov, A., Chen, G., Smith, M., Dawn, M., Freeman-Cook, L., … Gerstein, M. B. (2009). Robust-linear-model normalization to reduce technical variability in functional protein microarrays. *Journal of Proteome Research*, *8*(12), 5451–5464. https://doi.org/10.1021/pr900412k

Scherer, A. (2009). *Batch Effects and Noise in Microarray Experiments*. (A. Scherer, Ed.), *Batch Effects and Noise in Microarray Experiments: Sources and Solutions*. Chichester, UK: John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470685983

Schützenmeister, A., & Piepho, H. P. (2010). Background correction of two-colour cDNA microarray data using spatial smoothing methods. *Theoretical and Applied Genetics*, *120*(2), 475–490. https://doi.org/10.1007/s00122-009-1210-3

Sill, M., Schröder, C., Hoheisel, J. D., Benner, A., & Zucknick, M. (2010). Assessment and optimisation of normalisation methods for dual-colour antibody microarrays. *BMC Bioinformatics*, *11*(1), 556. https://doi.org/10.1186/1471-2105-11-556

Silver, J. D., Ritchie, M. E., & Smyth, G. K. (2009). Microarray background correction: maximum likelihood estimation for the normal-exponential convolution. *Biostatistics*, *10*(2), 352–363. https://doi.org/10.1093/biostatistics/kxn042

Stoevesandt, O., Taussig, M. J., & He, M. (2009). Protein microarrays: High-throughput tools for proteomics. *Expert Review of Proteomics*, *6*(2), 145–157. https://doi.org/10.1586/epr.09.2

Sutandy, F. X. R. R., Qian, J., Chen, C.-S. S., & Zhu, H. (2013). Overview of protein microarrays. *Current Protocols in Protein Science*, *Chapter 27*(SUPPL.72), 1–21. https://doi.org/10.1002/0471140864.ps2701s72

Thomassen, G. O. S., Rowe, A. D., Lagesen, K., Lindvall, J. M., & Rognes, T. (2009). Custom Design and Analysis of High-Density Oligonucleotide Bacterial Tiling Microarrays. *PLoS ONE*, *4*(6), e5943. https://doi.org/10.1371/journal.pone.0005943

Watson, M., Pérez-Alegre, M., Baron, M., Delmas, C., Dovč, P., Duval, M., … de Koning, D.-J. (2009). Analysis of a simulated microarray dataset: Comparison of methods for data normalisation and detection of differential expression (Open Access publication). *Genetics Selection Evolution*, *39*(6), 669. https://doi.org/10.1186/1297-9686-39-6-669

Yang, Y. H., Buckley, M. J., Dudoit, S., & Speed, T. P. (2002). Comparison of Methods for Image Analysis on cDNA Microarray Data. *Journal of Computational and Graphical Statistics*, *11*(1), 108–136. https://doi.org/10.1198/106186002317375640

Yu, X., Schneiderhan-Marra, N., & Joos, T. O. (2010). Protein microarrays for personalized medicine. *Clinical Chemistry*, *56*(3), 376–387.

https://doi.org/10.1373/clinchem.2009.137158

Appendices

Appendix 1: Mini-array layout, 20 rows by 12 columns; the last subscript indicate the replicate number.

miniarray layout	1	2	3	4	5	6	7	8	9	10	11	12
1	HPYL_1_1	BRUM_2_1	ROTV_2_1	TRBG_1_1	TRCR_1_1	CTET_1_1	ETVC_1_1	VZOS_1_1	RABV_2_1	TSOL_1_1	TRCR_2_1	LDON_2_1
2	HPYL_1_2	BRUM_2_2	ROTV_2_2	TRBG_1_2	TRCR_1_2	CTET_1_2	ETVC_1_2	VZOS_1_2	RABV_2_2	TSOL_1_2	TRCR_2_2	LDON_2_2
3	ECHG_2_1	VZOS_2_1	MYBB_2_1	TRBG_2_1	HSV1_2_1	RUBV_1_1	DEN4_1_1	LDON_1_1	MENB_1_1	TSOL_2_1	WUCB_1_1	ONCV_1_1
4	ECHG_2_2	VZOS_2_2	MYBB_2_2	TRBG_2_2	HSV1_2_2	RUBV_1_2	DEN4_1_2	LDON_1_2	MENB_1_2	TSOL_2_2	WUCB_1_2	ONCV_1_2
5	HINF_2_1	RSVA_1_1	POLV_2_1	ECHG_1_1	BRUM_1_1	HV12-2_1	BUFFER_1	ESCO_2_1	SMAN_1_1	HEPC_2_1	ENTH_1_1	ADV2_2_1
6	HINF_2_2	RSVA_1_2	POLV_2_2	ECHG_1_2	BRUM_1_2	HV12-2_2	BUFFER_2	ESCO_2_2	SMAN_1_2	HEPC_2_2	ENTH_1_2	ADV2_2_2
7	PIV3_1_1	COB3_1_1	SMAN_2_1	SARS_2_1	PSAR_1_1	HEPA_1_1	HV6B_1_1	HRVA_1_1	DEN2_1_1	MYBT_1_1	SFLX_1_1	HRVB_1_1
8	PIV3_1_2	COB3_1_2	SMAN_2_2	SARS_2_2	PSAR_1_2	HEPA_1_2	HV6B_1_2	HRVA_1_2	DEN2_1_2	MYBT_1_2	SFLX_1_2	HRVB_1_2
9	STYP_2_1	MUMP_1_1	HPVI_1_1	HEPA_2_1	BORP_1_1	SARS_1_1	MENB_2_1	STYP_1_1	ONCV_2_1	WUCB_2_1	MYBT_2_1	DEN3_1_1
10	STYP_2_2	MUMP_1_2	HPVI_1_2	HEPA_2_2	BORP_1_2	SARS_1_2	MENB_2_2	STYP_1_2	ONCV_2_2	WUCB_2_2	MYBT_2_2	DEN3_1_2
11	HPVI_2_1	SPNE_2_1	PARV_1_1	FLUA_1_1	PSAR_2_1	MRSA_1_1	ADV2_1_1	HEPC_1_1	HRVB_2_1	PIV2_1_1	CMVI_2_1	RUBV_2_1
12	HPVI_2_2	SPNE_2_2	PARV_1_2	FLUA_1_2	PSAR_2_2	MRSA_1_2	ADV2_1_2	HEPC_1_2	HRVB_2_2	PIV2_1_2	CMVI_2_2	RUBV_2_2
13	KPNE_1_1	PFAL_1_1	POLV_1_1	ENTH_2_1	HV6B_2_1	ETVA_1_1	SPNE_1_1	EBVI_1_1	SPYO_2_1	KPNE_2_1	HPYL_2_1	HEPB_2_1
14	KPNE_1_2	PFAL_1_2	POLV_1_2	ENTH_2_2	HV6B_2_2	ETVA_1_2	SPNE_1_2	EBVI_1_2	SPYO_2_2	KPNE_2_2	HPYL_2_2	HEPB_2_2
15	HV12-1_1	STAU_1_1	BORP_2_1	HINF_1_1	HSV1_1_1	FLUA_2_1	SFLX_2_1	STAU_2_1	DEN2_2_1	HERB_1_1	ROTV_1_1	MYBB_1_1
16	HV12-1_2	STAU_1_2	BORP_2_2	HINF_1_2	HSV1_1_2	FLUA_2_2	SFLX_2_2	STAU_2_2	DEN2_2_2	HERB_1_2	ROTV_1_2	MYBB_1_2
17	BUFFER_3	BUFFER_4	BUFFER_5	PFAL_2_1	BUFFER_6	BUFFER_7	BUFFER_8	BUFFER_9	BUFFER_10	CTET_2_1	BUFFER_11	BUFFER_12
18	BUFFER_13	BUFFER_14	BUFFER_15	PFAL_2_2	BUFFER_16	BUFFER_17	BUFFER_18	BUFFER_19	BUFFER_20	CTET_2_2	BUFFER_21	BUFFER_22
19	BUFFER_23	COMM1gG_1	LANDIgM_1	BUFFER_24	BUFFER_25	BUFFER_26	BUFFER_27	LANDIgA_1	LANDIgG_1	MEAV_2_1	BUFFER_28	BUFFER_29
20	BUFFER_30	COMM1gG_2	LANDIgM_2	BUFFER_31	BUFFER_32	BUFFER_33	BUFFER_34	LANDIgA_2	LANDIgG_2	MEAV_2_2	BUFFER_35	BUFFER_36

Appendix 2: Description of the peptides

Peptide ID	Organism	Peptide ID	Organism
ADV2_1	Human adenovirus 2	MEAV_2	Measles virus strain Edmonston
ADV2_2	Human adenovirus 2	MENB_1	Neisseria meningitidis serogroup B H44/76
BORP_1	Bordetella pertussis	MENB_2	Neisseria meningitidis serogroup B H44/76
BORP_2	Bordetella pertussis	MRSA_1	Staphylococcus aureus subsp. aureus MRSA252
BRUM_1	Brugia malayi	MUMP_1	Mumps virus
BRUM_2	Brugia malayi	MYBB_1	Mycobacterium bovis BCG
CMVI_1	Human herpesvirus 5	MYBB_2	Mycobacterium bovis BCG str. Pasteur 1173P2
CMVI_2	Human herpesvirus 5	MYBT_1	Mycobacterium tuberculosis
COB3_1	Coxsackievirus B3	MYBT_2	Mycobacterium tuberculosis
CTET_1	Clostridium tetani	ONCV_1	Onchocerca volvulus
CTET_2	Clostridium tetani	ONCV_2	Onchocerca volvulus
DEN2_1	Dengue virus 2	PSAR_1	Pseudomonas aeruginosa
DEN2_2	Dengue virus 2 Jamaica/1409/1983	PSAR_2	Pseudomonas aeruginosa
DEN3_1	Dengue virus 3	PARV_1	Human parvovirus B19
DEN4_1	Dengue virus 4	PFAL_1	Plasmodium falciparum 3D7
EBVI_1	Human herpesvirus 4	PFAL_2	Plasmodium falciparum 3D7
EBVI_2	Human herpesvirus 4	PIV2_1	Human parainfluenza virus 2
ECHG_1	Echinococcus granulosus	PIV3_1	Human parainfluenza virus 3
ECHG_2	Echinococcus granulosus	POLV_1	Human poliovirus 3 strain Sabin
ENTH_1	Entamoeba histolytica	POLV_2	Human poliovirus 3 strain Sabin
ENTH_2	Entamoeba histolytica	RABV_1	Rabies virus
ESCO_1	Escherichia coli	RABV_2	Rabies virus HEP-FLURY
ESCO_2	Escherichia coli	ROTV_1	Human rotavirus A
ETVA_1	Enterovirus A71	ROTV_2	Human rotavirus MP409
ETVC_1	Enterovirus C	RSVA_1	Human respiratory syncytial virus
FLUA_1	Influenza A virus (A/California/04/2009(H1N1))	RSVB_1	Human respiratory syncytial virus
FLUA_2	Influenza A virus (A/California/04/2009(H1N1))	RSVF_1	Human respiratory syncytial virus
HEPA_1	Human hepatitis A virus Hu/Australia/HM175/1976	RUBV_1	Rubella virus strain Therien
HEPA_2	Human hepatitis A virus Hu/Australia/HM175/1976	RUBV_2	Rubella virus strain Therien
HEPB_1	Hepatitis B virus	SARS_1	SARS coronavirus
HEPB_2	Hepatitis B virus subtype adw2	SARS_2	SARS coronavirus Tor2
HEPC_1	Hepatitis C virus	SFLX_1	Shigella flexneri
HEPC_2	Hepatitis C virus (isolate BK)	SFLX_2	Shigella flexneri 3a
HINF_1	Haemophilus influenzae NTHi 1479	SMAN_1	Schistosoma mansoni
HINF_2	Haemophilus influenzae Serotype B	SMAN_2	Schistosoma mansoni Puerto Rico
HPVI_1	Human papillomavirus	SPNE_1	Streptococcus pneumoniae
HPVI_2	Human papillomavirus type 16	SPNE_2	Streptococcus pneumoniae
HPYL_1	Helicobacter pylori	SPYO_1	Streptococcus pyogenes serotype M5
HPYL_2	Helicobacter pylori	SPYO_2	Streptococcus pyogenes serotype M5
HRVA_1	Human rhinovirus A2	STAU_1	Staphylococcus aureus subsp. aureus COL
HRVA_2	Human rhinovirus A89	STAU_2	Staphylococcus aureus subsp. aureus COL
HRVB_1	Human rhinovirus B14	STYP_1	Salmonella enterica subsp. enterica serovar Typhi
HRVB_2	Human rhinovirus B14	STYP_2	Salmonella enterica subsp. enterica serovar Typhi
HSV1_1	Herpes simplex virus (type 1 / strain 17)	TRBG_1	Trypanosoma brucei gambiense
HSV1_2	Herpes simplex virus (type 1 / strain 17)	TRBG_2	Trypanosoma brucei gambiense
HV12_1	Human herpesvirus 2 or 1	TRCR_1	Trypanosoma cruzi
HV12_2	Human herpesvirus 2 or 1	TRCR_2	Trypanosoma cruzi
HV6B_1	Human herpesvirus 6B	TSOL_1	Taenia solium
HV6B_2	Human herpesvirus 6B	TSOL_2	Taenia solium
KPNE_1	Klebsiella pneumoniae	VZOS_1	Human herpesvirus 3 H-551
KPNE_2	Klebsiella pneumoniae	VZOS_2	Human herpesvirus 3 H-551
LDON_1	Leishmania donovani	WUCB_1	Wuchereria bancrofti
LDON_2	Leishmania donovani	WUCB_2	Wuchereria bancrofti
MEAV_1	Measles virus strain Edmonston		

Appendix 3: Project R code – the analysis framework

```
library(tidyverse)
library(gtools)
library(ggpubr)
library(limma)
library(sva)
library(ggbeeswarm)
library(data.table)
library(lme4)
library(splines)
library(ggthemes)
library(ggrepel)
library(plotly)
library(pheatmap)
library(gplots)
library(ggplotify)

# + slide and miniarray layout directory + ----
slide_layout_dir <- ("DATA/Slide_Layout/")

peptide_layout <-
        read.csv(paste0(slide_layout_dir, "miniarray_layout - rep.csv"))

# Raw data directory
Raw_data_dir <- "DATA/Raw_Data/"

# Age data # Sample no and Age in months

age_data <-
        read.csv("DATA/sample.list.age.csv") %>%
        rename(sample_no = serial) %>%
        mutate(sample_no = as.character(sample_no))

# buffer spots to filter
bad.buffer.spots <- ("_3$|_13$|_23$|_24$|_27$|_30$|_31$|_34$")

# A sample peptides to visualize dilution
serial.dilution.spots <- c("POLV_1", "SPYO_2", "RSVA_1", "HV12-2")


# Custom function to read Raw data files and convert to wide dataset (samples by spots) ----

Read_Array_Data <-
        function(Ig = "IgG", data_type = "Foreground") {
                reader <- function(filename) {
                        df <- fread(filename, skip = "Flags")
                        df$slide <- gsub(
                                pattern = paste0(Raw_data_dir, "|Pmt.*", sep = ""),
                                replacement = "",
                                perl = T,
                                x = filename
                        )
                        df
                }

                channel <- ifelse(Ig == "IgA", "532", "635")
                datatype <-
                        if (data_type == "Foreground") {
                                "F"
                        } else if (data_type == "Background") {
                                "B"
                        }
                var_column <-
                        paste(paste0(datatype, channel), "Median", sep = " ")

                # Raw data file names
                filenames <-
                        list.files(
                                Raw_data_dir,
                                pattern = paste(Ig, ".txt$", sep = ""),
                                full.names = T
```

```
                        )

                # Combine the array datasets
                full_data <-
                        lapply(filenames, reader) %>%
                        bind_rows(.) %>%
                        dplyr::rename(miniarray = "Block")

                sample_ids <-
                        read.csv(paste0(slide_layout_dir, "sample.ids.", Ig, ".csv"),
                                 stringsAsFactors = T)

                layout <-
                        suppressWarnings(sample_ids %>% gather("slide", "sample_id", -c(miniarray)))

                names(layout) <-
                        c("miniarray", "slide", "sample_id")

                # wide dataset
                mydata <-
                        suppressMessages(
                                layout %>%
                                        mutate(
                                                slide = str_replace(slide, "s", repl = "S"),
                                                slide = str_replace(slide, "\.", repl = "_")
                                        ) %>%
                                        left_join(full_data, .) %>% select(slide, miniarray,
sample_id, Name, var_column) %>%
                                        group_by(slide, miniarray, Name, sample_id) %>%
                                        mutate(Name_id_replicate = paste0(Name, "_", row_number()))
%>% ungroup() %>%
                                        arrange(slide, miniarray, Name_id_replicate) %>%
                                        group_by(slide, sample_id, Name_id_replicate) %>%
                                        mutate(
                                                sample_ID = paste0(
                                                        slide,
                                                        "_",
                                                        sample_id,
                                                        "_",
                                                        row_number(),
                                                        "_",
                                                        "miniarray",
                                                        "_",
                                                        miniarray
                                                )
                                        ) %>%
                                        arrange(Name_id_replicate, sample_ID) %>%
                                        ungroup() %>%
                                        select(Name_id_replicate, sample_ID, var_column) %>%
                                        spread(Name_id_replicate, var_column)
                        )
        }

# A function to tidy and extract various forms of the peptide dataset ----

tidying <-
        function(dataset = Read_Array_Data("IgG", "Foreground"),
                 return.data = "wide") {
                require(tidyverse, quiet = T)
                library(gtools)

                tidy <-
                        function(dataset) {
                                dataset %>%
                                        mutate(
                                                sample_no = str_extract(sample_ID, "\\d{4,}"),
                                                slide = str_extract(sample_ID, "Slide_\\d{1,2}"),
                                                slide = str_replace(slide, "\w+_", repl = ""),
                                                miniarray = str_extract(sample_ID,
"miniarray_\\d{1,2}"),
                                                miniarray = str_replace(miniarray, "\w+_", repl = ""),
                                                sample_group = case_when(
                                                        str_detect(sample_ID, "_NC_") ~ "Buffer",
                                                        str_detect(sample_ID, "_PAS_") ~ "PAS",
```

```
                                                        str_detect(
                                                                sample_ID,
"[:digit:]{1,9}_[:digit:]{1,9}_[:digit:]{1,9}"
                                                        ) ~ "Sample",
                                                        str_detect(sample_ID, "__") ~ "Blank"
                                                )
                                        ) %>%
                                        filter(!sample_group == "Blank") %>%
                                        mutate(
                                                sample_group = factor(sample_group, levels =
unique(sample_group)),
                                                slide = factor(slide, levels = mixedsort(unique(
                                                        slide
                                                ))),
                                                miniarray = factor(miniarray, levels =
mixedsort(unique(
                                                        miniarray
                                                )))
                                        ) %>%
                                        select(sample_ID,
                                               sample_no,
                                               slide,
                                               miniarray,
                                               sample_group,
                                               everything()) %>%
                                        gather("spot", "MFI", -c(1:5)) %>%
                                        # filter(!str_detect(spot, bad.buffer.spots)) %>%
                                        spread("spot", "MFI")
                        }

                wide <- tidy(dataset)

                buffer <-
                        wide %>%
                        gather("spot", "MFI", -c(1:5)) %>%
                        filter(str_detect(spot, "BUFFER"))

                full.long <-
                        wide %>%
                        gather("spot", "MFI", -c(1:5)) %>%
                        filter(!str_detect(spot, "BUFFER")) %>%
                        mutate(
                                replicate = case_when(str_detect(spot, "_1$") ~ 1,
                                                      str_detect(spot, "_2$") ~ 2),
                                spot = str_replace(spot, "_1$|_2$", "")
                        ) %>%
                        select(1:6, 8, 7) %>%
                        bind_rows(., buffer)

                # Creating a dataframe for the first replicates
                replicate_1 <-
                        wide %>%
                        select(1:5, ends_with("_1"), -matches("BUFFER")) %>%
                        gather(
                                "spot",
                                "replicate.1",
                                -c(sample_ID, sample_no, slide, miniarray, sample_group)
                        ) %>%
                        mutate(spot = str_replace(spot, "_1$", ""))

                # Creating a dataframe for the second replicates
                replicate_2 <-
                        wide %>%
                        select(1:5, ends_with("_2"), -matches("BUFFER")) %>%
                        gather(
                                "spot",
                                "replicate.2",
                                -c(sample_ID, sample_no, slide, miniarray, sample_group)
                        ) %>%
                        mutate(spot = str_replace(spot, "_2$", ""))

                # Creating a dataframe for buffer spots
                buffer_spots <-
```

```
                        wide %>% select(1:5, matches("BUFFER"))

                buffer.spots.long <- wide %>%
                        select(1:5, matches("BUFFER")) %>%
                        gather("spot",
                               "MFI",
                               -c(sample_ID, sample_no, slide, miniarray, sample_group))

                # Joining into on dataframe & Flagging spots whose replicates have a difference of
more than 20%
                long.data.replicates <-
                        suppressMessages(full_join(replicate_1, replicate_2)) %>%
                        mutate(
                                MFI = (replicate.1 + replicate.2) / 2,
                                replicate.flag = factor(
                                        ifelse((abs(
                                                replicate.1 - replicate.2
                                        )) / MFI < 0.3, 1, 0),
                                        levels = c(0, 1),
                                        labels = c("bad", "good")
                                )
                        ) %>%
                        select(1:6, replicate.flag, everything())

                # Long data without the replicates and flag variables
                antigen.spots.long <-
                        long.data.replicates %>% select(1:6,
                                                        everything(),
                                                        -replicate.flag,
                                                        -replicate.1,
                                                        -replicate.2)

                full.long.data <-
                        bind_rows(antigen.spots.long, buffer.spots.long)

                # Antigen spots including landmark/commercial and the treatment groups
                antigen.spots <-
                        antigen.spots.long %>%
                        group_by(slide, miniarray) %>%
                        spread("spot", "MFI") %>%
                        ungroup()

                # wide format dataset with averaged replicates for both peptides and buffer spots
                wide.summarized <-
                        suppressMessages(full_join(antigen.spots, buffer_spots))

                # Buffer subtracted dataset
                buffer.subtracted.long <-
                        wide %>%
                        gather("spot",
                               "MFI",
                               -c(sample_ID, sample_no, slide, miniarray, sample_group)) %>%
                        group_by(slide, miniarray) %>%
                        mutate(MFI = MFI - MFI[str_detect(spot, "BUFFER_22")]) %>%
                        ungroup()

                # Buffer subtracted (using median buffer) wide dataset
                buffer.subtracted <-
                        buffer.subtracted.long %>% spread("spot", "MFI")


                mget(return.data, ifnotfound = paste0(("THIS DATASET IS NOT FOUND!!!")))[[1]]
        }

## ++ End of tidying function ++ ##----

## Neighbourhood background estimation function ----

# A function to estimate backgound intensity as the median locally estimated background using
neighboured spots for the peptide design

neighbor_bg_smoothing <-
        function(Ig,
                 peptide_layout = paste(slide_layout_dir, "miniarray_layout - rep.csv",
```

```
                        sep = "/"),
 number_cols = 12,
 number_rows = 20) {
require(tidyverse, quietly = T)

peptide_neigbor <-
        function(peptide_layout,
                 number_cols,
                 number_rows) {
                # neigbourhood data
                mydata <- data.frame(
                        x = integer(),
                        n1 = integer(),
                        n2 = integer(),
                        n3 = integer(),
                        n4 = integer(),
                        n5 = integer(),
                        n6 = integer(),
                        n7 = integer(),
                        n8 = integer()
                )
                x <- 1
                while (x <= number_rows * number_cols) {
                        mydata[x, ] <- c(
                                x,
                                x - 1,
                                x + 1,
                                x - number_cols,
                                x + number_cols,
                                x - number_cols - 1,
                                x - number_cols + 1,
                                x + number_cols - 1,
                                x + number_cols + 1
                        )
                        x <- x + 1
                }

                mydata <- mydata %>%
                        mutate_all(
                                .funs = function(x) {
                                        ifelse(x <= 0 | x > number_rows * number_cols,
                                               NA,
                                               x)
                                }
                        ) %>%
                        gather("neigbour", "value", -c(1)) %>%
                        select(-2) %>%
                        arrange(x) %>%
                        filter(!is.na(value))

                peptide_layout <- peptide_layout %>%
                        gather("column", "spot", -c(1)) %>%
                        arrange(miniarray.layout) %>%
                        rownames_to_column("x") %>%
                        select(-c(2:3)) %>%
                        mutate(x = as.numeric(x))

                left_join(mydata, peptide_layout, by = "x") %>%
                        rename(id = x, x = value) %>%
                        left_join(., peptide_layout, by = "x") %>%
                        select(-c(1:2)) %>%
                        rename(spot = spot.x, neighbor = spot.y)
        }

suppressWarnings(
        Read_Array_Data(Ig, "Background") %>%
                gather("spot_x", "MFI", -c(1)) %>%
                group_by(sample_ID) %>%
                right_join(
                        .,
                        peptide_neigbor(peptide_layout, number_cols, number_rows),
                        by = c("spot_x" = "neighbor")
                ) %>%
                group_by(sample_ID, spot) %>%
```

```
                                mutate(min_bg = min(MFI)) %>%
                                ungroup() %>%
                                arrange(sample_ID) %>%
                                select(sample_ID, spot, min_bg) %>%
                                group_by(sample_ID, spot) %>%
                                summarise(min_bg = min(min_bg)) %>%
                                ungroup() %>% spread(spot, min_bg)
                )
        }

# << DATA TRANSFORMATION METHODS >> ----
# + Log2 transformation + ----

log2.transform <- function(Ig) {
        Read_Array_Data(Ig) %>%
                tidying(., "full.long.data") %>%
                mutate(MFI = log2(MFI + 1)) %>%
                spread("spot", "MFI")
}

# + Log2 & ComBat + ----

ComBat.Peptide <- function(Ig, ref.batch = NULL) {
        x <- Read_Array_Data(Ig) %>%
                tidying(., "wide")

        x.vars <- x[, c(1:5)]

        x.matrix <- x[, -c(1:5)] %>%
                as.matrix()

        attr(x.matrix, "dimnames") <-
                list(x$sample_ID, colnames(x.matrix))

        mydata <- x.matrix %>% (function(x) {
                log2(x + 2)
        })
        mydata <-
                suppressMessages(ComBat(
                        t(mydata),
                        x$slide,
                        ref.batch = ref.batch,
                        par.prior = F
                )) %>%
                t() %>%
                cbind(x[, c(2:5)], .) %>%
                rownames_to_column("sample_ID") %>%
                as_tibble() %>%
                tidying("wide.summarized")
}

# +Variance Stabilizing Normalisation (VSN)+ ----

VSN.transform <-
        function(Ig,
                 stratified = FALSE) {
                require(tidyverse)
                require(VSN)
                require(limma)

                x <- Read_Array_Data(Ig) %>%
                        tidying(., "wide.summarized")

                x.matrix <- x[, -c(1:5)] %>%
                        as.matrix()
                x.matrix[is.nan(x.matrix)] <- NA
                attr(x.matrix, "dimnames") <-
                        list(x$sample_ID, colnames(x.matrix))

                # Stratified VSN
                VSN_slide_stratified <-
                        justVSN(x.matrix,
                                strata = x$slide,
                                minData = 20) %>%
```

```
                        cbind(x[, 2:5], .) %>%
                        rownames_to_column("sample_ID") %>%
                        as_tibble()

                # unstratified VSN
                VSN.single_strata <-
                        justVSN(x.matrix) %>%
                        cbind(x[, 2:5], .) %>%
                        rownames_to_column("sample_ID") %>%
                        as_tibble()

                mydata <- ifelse(isTRUE(stratified),
                                 return(VSN_slide_stratified),
                                 return(VSN.single_strata))
        }

# + Quantile normalisation +----

quantile_normalize <-
        function(Ig,
                 buffer.subtracted = FALSE) {
                dataset <- Read_Array_Data(Ig)
                dataset <-
                        suppressMessages(tidying(
                                dataset,
                                ifelse(
                                        isTRUE(buffer.subtracted),
                                        "buffer.subtracted.long",
                                        "full.long.data"
                                )
                        ))

                quantile.normalized <- list()

                data <- split(dataset, dataset$sample_group)

                for (i in 1:length(data)) {
                        quantile.normalized[[i]] <- data[[i]] %>%
                                # filter(sample_group == sample_group[[i]]) %>%
                                group_by(slide, miniarray) %>% # grouping by the slide and sample
                                arrange(MFI, .by_group = TRUE) %>% # Sorting the data ascending
                                mutate(rank = row_number()) %>% # Recoding a new variable to denote
the rank
                                group_by(rank) %>% # grouping by the rank value
                                mutate(MFI = log2(mean(MFI))) %>% # calculating the new normalised
signal intensity
                                ungroup() %>% # ungrouping to restore the original data format
                                select(-c(rank)) %>% # removing the rank variable from the dataset
                                as.data.frame() %>%
                                spread("spot", "MFI")
                }
                bind_rows(quantile.normalized) %>% as_tibble()
        }

# + Data-Driven Haar-Fisz transformation + ----
DDHF.peptide <- function(Ig) {
        new.data.list <- list()
        id.vars <- list()
        dataset <- Read_Array_Data(Ig)
        data.list <-
                tidying(dataset, "long.data.replicates") %>%
                select(1, 6, 8, 9) %>%
                split(., .$sample_ID)

        for (i in 1:length(data.list)) {
                id.vars[[i]] <- data.list[[i]][, 1:2]

                new.data.list[[i]] <-
                        data.list[[i]] %>%
                        select(-c(1, 2)) %>%
                        remove_rownames() %>%
                        as.matrix() %>%
                        DDHFm::DDHFm() %>%
                        (function(x)
```

```
                                ifelse(x < 1 |
                                               is.na(x) |
                                               is.nan(x), 1, x)) %>%
                        cbind(id.vars[[i]], .)

                colnames(new.data.list[[i]]) <-
                        c("sample_ID", "spot", "rep_1", "rep_2")

                new.data.list[[i]] <-
                        new.data.list[[i]] %>%
                        mutate(MFI = (rep_1 + rep_2) / 2,
                               MFI = ifelse(MFI < 1, 1, log2(MFI))) %>%
                        select(-c(3, 4))
        }

        bind_rows(new.data.list) %>%
                spread("spot", "MFI") %>%
                left_join(tidying(return.da = "wide.summarized")[, c(1:5)], .,
                          by = "sample_ID")
}

# + Linear Mixed Model (LMM) + ----

LMM.transform <- function(Ig = "IgG") {
        require(lme4)
        require(tidyverse, quiet = T)
        mydata <-
                inner_join(
                        x = tidying(dataset = Read_Array_Data(Ig, "Foreground"), "full.long"),
                        y = tidying(dataset = Read_Array_Data(Ig, "Background"), "full.long"),
                        by = c(
                                "sample_ID",
                                "sample_no",
                                "slide",
                                "miniarray",
                                "sample_group",
                                "spot",
                                "replicate"
                        ),
                        copy = T
                ) %>%
                filter(!str_detect(spot, "3|13|23|24|27|30|31|34")) %>%
                rename(bg_MFI = MFI.y, spot_MFI = MFI.x) %>%
                group_by(slide, miniarray) %>%
                mutate(
                        med.buffer = median(spot_MFI[str_detect(spot, "BUFFER")]),
                        med.buffer.corrected = spot_MFI - median(spot_MFI)
                ) %>%
                ungroup()

        # Model subset data excluding PAS with smaller concentrations
        model.data <- mydata %>%
                filter(!miniarray %in% c(4, 7, 10, 13, 16, 19, 22))

        my.model <-
                lmer(
                        log2(spot_MFI + 2) ~ log2(bg_MFI + 2) + sample_group + (1 |
                                                                                        slide /
miniarray) + (1 |

spot),
                        data = model.data,
                        REML = FALSE,
                        control = lmerControl(optimizer = "bobyqa"),
                        weights = log2(med.buffer + 2)
                )

        mydata$spot_MFI <-
                predict(my.model,
                        newdata = mydata,
                        allow.new.levels = T)

        buffer <- mydata %>%
                select(1:6, 8) %>%
```

```
                filter(str_detect(spot, "BUFFER")) %>%
                remove_rownames()

        mydata %>%
                filter(!str_detect(spot, "BUFFER")) %>%
                select(1:8) %>%
                spread("replicate", "spot_MFI") %>%
                rename(rep_1 = "1", rep_2 = "2") %>%
                mutate(spot_MFI = (rep_1 + rep_2) / 2) %>%
                select(-c(rep_1, rep_2)) %>%
                bind_rows(., buffer) %>%
                remove_rownames() %>%
                spread("spot", "spot_MFI")
}

# + LMM + ComBat + ----

LMM_ComBat <- function(Ig, ref.batch = NULL) {
        require(sva, quiet = T)
        require(tidyverse, quiet = T)

        mydata <- LMM.transform(Ig)

        mydata_ComBat <-
                suppressMessages(ComBat(t(mydata[, -c(1:5)]),
                                        mydata$slide,
                                        ref.batch = ref.batch)) %>%
                t() %>%
                cbind(mydata[, c(1:5)], .) %>%
                as_tibble()
}

### Variance Stabilisation using the PAS (PAS) with highest concentration -----
# stabilisation factor = MFI pep_i/mean(pep_i accross slides) within the PAS

PAS_stabilisation <- function(Ig, PAS.Miniarray = 1) {
        require(tidyverse)

        data_1 <-
                inner_join(
                        x = tidying(dataset = Read_Array_Data(Ig, "Foreground"), "full.long"),
                        y = tidying(dataset = neighbor_bg_smoothing(Ig, peptide_layout), "full.long"),
                        by = c(
                                "sample_ID",
                                "sample_no",
                                "slide",
                                "miniarray",
                                "sample_group",
                                "spot",
                                "replicate"
                        ),
                        copy = T
                ) %>%
                filter(!str_detect(spot, bad.buffer.spots)) %>%
                rename(bg_MFI = MFI.y, spot_MFI = MFI.x) %>%
                group_by(slide, miniarray) %>%
                mutate(
                        med.buffer = median(spot_MFI[str_detect(spot, "BUFFER")]),
                        med.buffer.corrected = spot_MFI - median(spot_MFI),
                        bg_subtracted = spot_MFI - bg_MFI
                ) %>%
                ungroup()

        # Calculating the stabilisation factor ==> MFI(spot_i)/median(MFI(spot_i)) accross positive
sample
        stabilisation_factor <-
                data_1 %>%
                filter(miniarray == PAS.Miniarray) %>%
                group_by(spot) %>%
                mutate(
                        median_spot_MFI = median(spot_MFI, na.rm = T),
                        stabilisation_factor = ifelse(median_spot_MFI < 5, 1, spot_MFI /
                                                              median_spot_MFI),
```

```
                        stabilisation_factor = ifelse(stabilisation_factor <= 0, 1,
stabilisation_factor),
                        bg_subtracted = bg_subtracted / stabilisation_factor
                ) %>%
                ungroup() %>%
                mutate(spot = ifelse(
                        str_detect(spot, "BUFFER"),
                        spot,
                        paste(spot, replicate, sep = "_")
                )) %>%
                select(slide, spot, stabilisation_factor)

        # Combining the dataset
        data_2 <-
                data_1 %>%
                mutate(spot = ifelse(
                        str_detect(spot, "BUFFER"),
                        spot,
                        paste(spot, replicate, sep = "_")
                )) %>%
                left_join(., stabilisation_factor, by = c("slide", "spot"))


        stabilised_data <- data_2 %>%
                select(sample_ID, spot, bg_subtracted, stabilisation_factor) %>%
                mutate(
                        bg_subtracted = bg_subtracted / stabilisation_factor,
                        bg_subtracted = ifelse(bg_subtracted <= 0, 1, bg_subtracted)
                ) %>%
                select(-stabilisation_factor) %>%
                spread(spot, bg_subtracted) %>%
                tidying("wide.summarized") %>%
                mutate_if(.predicate = is.numeric, .funs = (function(x) {
                        log2(x)
                }))
}

# BACKGROUND CORRECTTION METHODS ----
# + Local background subtraction + ----

Local_bg_subtract <- function(Ig) {
        inner_join(
                x = tidying(dataset = Read_Array_Data(Ig, "Foreground"), "full.long"),
                y = tidying(dataset = Read_Array_Data(Ig, "Background"), "full.long"),
                by = c(
                        "sample_ID",
                        "sample_no",
                        "slide",
                        "miniarray",
                        "sample_group",
                        "spot",
                        "replicate"
                ),
                copy = T
        ) %>%
                filter(!str_detect(spot, bad.buffer.spots)) %>%
                rename(bg_MFI = MFI.y, spot_MFI = MFI.x) %>%
                group_by(slide, miniarray) %>%
                mutate(bg_subtracted = spot_MFI - bg_MFI) %>%
                ungroup() %>%
                mutate(
                        spot = ifelse(
                                str_detect(spot, "BUFFER"),
                                spot,
                                paste(spot, replicate, sep = "_")
                        ),
                        bg_subtracted = ifelse(bg_subtracted <= 0, 1, bg_subtracted)
                ) %>%
                select(sample_ID, spot, bg_subtracted) %>%
                spread(spot, bg_subtracted) %>%
                tidying("wide.summarized") %>%
                mutate_if(.predicate = is.numeric, .funs = (function(x) {
                        log2(x)
                }))
```

```
}

# + Neighbourhood background subtraction + ----
moving_min_bg_subtract <- function(Ig) {
        inner_join(
                x = tidying(dataset = Read_Array_Data(Ig, "Foreground"), "full.long"),
                y = tidying(dataset = neighbor_bg_smoothing(Ig, peptide_layout), "full.long"),
                by = c(
                        "sample_ID",
                        "sample_no",
                        "slide",
                        "miniarray",
                        "sample_group",
                        "spot",
                        "replicate"
                ),
                copy = T
        ) %>%
                filter(!str_detect(spot, bad.buffer.spots)) %>%
                rename(bg_MFI = MFI.y, spot_MFI = MFI.x) %>%
                group_by(slide, miniarray) %>%
                mutate(
                        med.buffer = median(spot_MFI[str_detect(spot, "BUFFER")]),
                        med.buffer.corrected = spot_MFI - median(spot_MFI),
                        bg_subtracted = spot_MFI - bg_MFI
                ) %>%
                ungroup() %>%
                mutate(
                        spot = ifelse(
                                str_detect(spot, "BUFFER"),
                                spot,
                                paste(spot, replicate, sep = "_")
                        ),
                        bg_subtracted = ifelse(bg_subtracted <= 0, 1, bg_subtracted)
                ) %>%
                select(sample_ID, spot, bg_subtracted) %>%
                spread(spot, bg_subtracted) %>%
                tidying("wide.summarized") %>%
                mutate_if(.predicate = is.numeric, .funs = (function(x) {
                        log2(x)
                }))
}

# << VALIDATION PLOTS >> ----
#
# + Peptide dilution plot + ----

serial.plot <- function(data, ...) {
        data %>%
                gather("spot", "MFI", -c(1:5)) %>%
                mutate(miniarray = factor(miniarray, levels = mixedsort(unique(miniarray)))) %>%
                filter(miniarray %in% c(1, 4, 7, 10, 13, 16, 19, 22),
                       spot %in% serial.dilution.spots) %>%
                ggplot(aes(miniarray, MFI, col = slide)) +
                geom_smooth(aes(group = slide), se = F, span = 1) +
                facet_wrap(~ spot, nrow = 1) +
                scale_color_brewer(palette = "Paired") +
                labs(x = "Serial Dilution", y = "MFI") +
                # scale_x_discrete(labels = c("1:30", "1:300", "1:3k", "1:30k","1:300k",
                #                             "1:3m", "1:30m", "1:300m"))+
                # scale_y_continuous(trans = "log2")+
                theme_bw() +
                theme(
                        panel.grid.major = element_blank(),
                        axis.text.x = element_blank(),
                        axis.title = element_text(size = 16),
                        legend.text = element_text(size = 16),
                        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "in"),
                        axis.ticks.x = element_blank(),
                        ...
                )
}

# + Pairwise Comparison of Significant differences in PAS medians +----
```

```
wilcoxon.pvalue.heatmap <- function(data, ...) {
        require(pheatmap)
        require(RColorBrewer)

        miniarray.1 <- data %>%
                gather("spot", "MFI", -c(1:5)) %>%
                filter(miniarray == 1,
                       !str_detect(spot, "BUFFER|LAND|COMM"))

        diff.test <-
                kruskal.test(miniarray.1$MFI ~ miniarray.1$slide)

        pairwise.test <-
                pairwise.wilcox.test(miniarray.1$MFI, miniarray.1$slide) # , p.adj = 'bonf'

        p.values.df <- round(pairwise.test$p.value, 3)

        p.values.df <- rbind(p.values.df, "1" = NA)
        p.values.df <- cbind(p.values.df, "11" = NA)

        p.values.df <-
                p.values.df[mixedsort(row.names(p.values.df)), ]

        diag(p.values.df) <- 1

        col_names <- vector("numeric")
        for (i in 1:(nrow(p.values.df))) {
                col_names[i] <- paste("Slide", i, sep = " ")
        }

        row.names(p.values.df) <- col_names
        colnames(p.values.df) <- col_names

        pheatmap(
                p.values.df,
                cluster_rows = F,
                cluster_cols = F,
                color = colorpanel(9, low = "red", high = "green")[c(2, 5:9)],
                breaks = c(0, 0.05, 0.1, 0.3, 0.7, 1),
                legend_breaks = c(0, 0.05, 0.1, 0.3, 0.7, 1),
                na_col = "snow1",
                border_color = "snow1",
                angle_col = 90,
                display_numbers = T,
                fontsize = 16,
                fontsize_number = 10,
                number_format = "%.3f",
                number_color = "snow1",
                fontsize_col = 12,
                fontsize_row = 12,
                y = unit(3, "npc"),
                ...
        )
}

# + BOXPLOTS of peptides in PAS (highest concentration) accross slides + ----

PAS.Boxplots <- function(data, figure.title = "", ...) {
        data %>%
                gather("spot", "MFI", -c(1:5)) %>%
                filter(miniarray == 1,
                       MFI > 1,
                       !str_detect(spot, "BUFFER|LAND|COMM")) %>%
                ggplot(aes(slide, MFI)) +
                geom_boxplot(alpha = 0, size = 0.6) +
                geom_quasirandom(
                        size = 0.7,
                        fill = "snow1",
                        col = "black",
                        alpha = 0.7
                ) +
                scale_y_continuous(trans = "log2", ...) +
                labs(title = figure.title,
                     x = "Slide Number",
```

```
                    y = "Fluorescence intensities") +
                theme_wsj(color = "snow2",
                          title_family = "sans") +
                theme(
                        text = element_text(size = 16),
                        title = element_text(size = 14),
                        axis.line = element_line(size = 1),
                        legend.text = element_text(size = 16),
                        axis.title = element_text(size = 16),
                        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "in")
                )
}


# + Principal Components plot showing clusters of sample types (PAS, NC, the study Sample) + ----

PCA.plot <- function(data, figure.title, ...) {
        require(factoextra)
        require(FactoMineR)
        require(plotly)

        my.data <- data %>%
                filter(!miniarray %in% c(4, 7, 10, 13, 16, 19, 22)) %>%
                select(-matches("COMM|LAND|BUFFER")) %>%
                as.data.frame() %>%
                column_to_rownames("sample_ID") %>%
                select(-1)

        my.PCA <- PCA(
                my.data,
                quali.sup = 1:3,
                scale.unit = T,
                graph = F
        )

        dims <- my.PCA$ind$contrib

        plot_3d <- plot_ly(
                x = dims[, 1],
                y = dims[, 2],
                z = dims[, 3],
                type = "scatter3d",
                mode = "markers",
                color = factor(my.data$sample_group)
        )

        print(plot_3d)

        fviz_eig(my.PCA) %>% print()

        fviz_pca_ind(
                my.PCA,
                geom.ind = "point",
                addEllipses = T,
                pointsize = 2,
                alpha.ind = 0.6,
                habillage = 3,
                repel = T,
                title = figure.title,
                axes = c(1, 2)
        ) + # , ellipse.type = 'confidence'
                scale_color_brewer(palette = "Set1") +
                theme_bw() +
                theme(
                        axis.text = element_text(size = 16),
                        axis.title = element_text(size = 16),
                        title = element_text(size = 16),
                        legend.title = element_blank(),
                        legend.key = element_blank(),
                        panel.grid = element_blank(),
                        legend.text = element_text(size = 18),
                        rect = element_rect(fill = "snow1"),
                        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "in"),
                        ...
```

```
                )
}

# + Density plot grouped by sample groups (Filter for transformed MFI > 1 - in log2 scale) + ----
density.plot <- function(data, figure.title = "", ...) {
        data %>%
                # select(-c(cv_20[[1]])) %>%
                gather("spot", "MFI", -c(1:5)) %>%
                filter(MFI > 1) %>%
                filter(!miniarray %in% c(4, 7, 10, 13, 16, 19, 22)) %>%
                ggplot() +
                geom_density(aes(MFI, fill = sample_group),
                             alpha = 0.6,
                             position = "identity",
                             ...) +
                scale_x_continuous(trans = "log2") +
                scale_fill_brewer(palette = "Set1") +
                labs(y = "Density",
                     x = "Fluorescence intensity",
                     title = figure.title) +
                theme_wsj(color = "snow1",
                          title_family = "sans") +
                theme(
                        title = element_text(size = 14),
                        legend.title = element_blank(),
                        axis.line = element_line(size = 1),
                        legend.text = element_text(size = 14),
                        axis.title = element_text(size = 14),
                        axis.text = element_text(size = 14),
                        strip.text.x = element_text(face = "bold"),
                        strip.background = element_rect(colour = "grey95", fill = "grey95"),
                        panel.border = element_rect(colour = "grey95"),
                        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "in"),
                        ...
                )
}

# + PAS dilution + ----
PAS_Dilution <- function(Ig, method, fig.title, ...) {
        PAS.labels <- c(
                "1" = "1:30",
                "4" = "1:300",
                "7" = "1:3,000",
                "10" = "1:30,000",
                "13" = "1:300,000",
                "16" = "1:3,000,000",
                "19" = "1:30,000,000",
                "22" = "1:300,000,000"
        )

        method(Ig) %>%
                filter(sample_group %in% c("PAS")) %>%
                gather("spot", "MFI", -c(1:5)) %>%
                filter(MFI > 1,
                       !str_detect(spot, "BUFFER|COMM|LAND")) %>%
                ggplot(aes(1, MFI)) +
                geom_jitter(size = 0.8, ...) +
                scale_y_continuous(trans = "log2", ...) +
                labs(x = "Dilution",
                     y = "Fluorescence intensities",
                     title = fig.title) +
                facet_grid(
                        ~ miniarray,
                        labeller = labeller(miniarray = as_labeller(PAS.labels)),
                        switch = "x"
                ) +
                stat_summary(
                        fun.y = median,
                        col = "red",
                        geom = "crossbar",
                        size = 0.6,
                        ymin = 0,
                        ymax = 0
                ) +
```

```
                theme_bw() +
                theme(
                        axis.text.x = element_blank(),
                        axis.title = element_text(size = 18),
                        axis.text.y = element_text(size = 18),
                        title = element_text(size = 18),
                        axis.ticks.x = element_blank(),
                        axis.line.x = element_blank(),
                        strip.placement = "inside",
                        panel.grid = element_blank(),
                        panel.background = element_rect(fill = "snow2"),
                        panel.border = element_blank(),
                        strip.text.x = element_text(
                                angle = 90,
                                face = "bold",
                                size = 14,
                                # family = 'serif',
                                vjust = 0.5,
                                hjust = 1
                        ),
                        strip.switch.pad.grid = unit(0, "in")
                )
}


# + Correlation between age (in months) and antibody responses (antibody decay) [Forest plot like
plot]+ ---- ## For all the peptides
Corr_plot <- function(Ig,
                      method = LMM_ComBat,
                      title,
                      age_filter = 6,
                      filter_coef = 0) {
        # Filter non-coniciding similar peptides
        peptide_corr <- suppressWarnings(
                log2.transform(Ig) %>%
                        select(-matches("BUFFER|LAND|COMM")) %>%
                        gather("peptide", "MFI", -c(1:5)) %>%
                        mutate(
                                pep_dupli = ifelse(str_detect(peptide, "_1|-1"), "p_1", "p_2"),
                                peptide = str_replace(peptide, "_1|-1|_2|-2", "")
                        ) %>%
                        spread(pep_dupli, MFI) %>%
                        filter(!is.na(sample_no), !is.na(p_1), !is.na(p_2)) %>%
                        split(., .$peptide) %>%
                        map(
                                ~ cor.test(.$p_1, .$p_2, method = "spearman", na.action = "na.omit")
                        ) %>%
                        map_dfc(~ c(.$p.value, .$estimate)) %>% t() %>% as.data.frame() %>%
                        rownames_to_column("peptide_id") %>%
                        rename(p.value = V1, coef = V2) %>% filter(coef < abs(filter_coef))
        )

        message(
                paste(
                        "A List of peptides with a correlation less than absolute",
                        abs(filter_coef),
                        "for same target peptides",
                        sep = " "
                )
        )
        print(peptide_corr$peptide_id)

        data_1 <- suppressWarnings(
                method(Ig) %>%
                        gather("spot", "MFI", -c(1:5)) %>%
                        filter(!str_detect(spot, "BUFFER|LAND|COMM")) %>%
                        filter(sample_no %in% age_data$sample_no) %>%
                        inner_join(., age_data, by = "sample_no") %>%
                        mutate(peptide = str_replace(spot, "_1|-1|_2|-2", "")) %>%
                        filter(age_m < 6, !peptide %in% peptide_corr$peptide_id) %>%
                        split(., .$spot) %>%
                        map(~ cor.test(.$age_m, .$MFI, method = "spearman")) %>%
                        map_dfc(~ c(.$p.value, .$estimate)) %>%
                        t() %>% as.data.frame() %>%
```

```
                        rownames_to_column("epitope") %>%
                        rename(p.value = V1,
                               coef = V2) %>%
                        mutate(
                                signif = ifelse(p.value < 0.05, 1, 0),
                                signif = factor(signif, levels = c(0, 1)),
                                x = seq_along(coef)
                        )
        )

        data_1 %>%
                ggplot(aes(reorder(x, coef), coef)) +
                geom_text_repel(
                        aes(
                                x = reorder(x, coef),
                                y = coef,
                                label = epitope
                        ),
                        family = "sans",
                        nudge_y = ifelse(data_1$coef < 0, -0.1, 0.1),
                        size = 3,
                        segment.alpha = 0.2,
                        segment.size = 1,
                        segment.color = "grey20",
                        arrow = arrow(
                                type = "open",
                                ends = "first",
                                length = unit(0.08, "in")
                        )
                ) +
                geom_point(
                        aes(
                                fill = p.value,
                                col = signif,
                                size = abs(coef)
                        ),
                        show.legend = F,
                        alpha = 0.9
                ) +
                geom_linerange(
                        aes(ymin = 0, ymax = coef),
                        col = "snow1",
                        show.legend = F,
                        alpha = 0.2,
                        size = 0.1
                ) +
                scale_color_manual(values = c("0" = "grey", "1" = "blue")) +
                scale_size_continuous(range = c(2, 6)) +
                scale_y_continuous(limits = c(min(data_1$coef) - 0.1, max(data_1$coef) + 0.1)) +
                labs(y = "Correlation Coefficient,
rho",
                     x = "Peptides") +
                coord_flip() +
                labs(title = title) +
                theme_wsj(color = "brown2", title_family = "sans") +
                geom_hline(yintercept = 0,
                           lty = "dotted",
                           size = 0.1) +
                theme(
                        axis.title.y = element_blank(),
                        axis.text.y = element_blank(),
                        panel.grid = element_blank(),
                        axis.text.x = element_text(size = 22),
                        title = element_text(size = 25),
                        axis.title = element_text(size = 22, face = "bold"),
                        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "in")
                )
}


## Peptide boxplots compared with buffer
Spots_distribution_boxplots <-
        function(Ig, method) {
                method(Ig) %>%
```

```
                        select(-matches("LAND|COMM")) %>%
                        gather("peptide", "MFI", -c(1:5)) %>%
                        # mutate(pep_dupli = ifelse(str_detect(peptide, '_1|-1'),'p_1','p_2'),
                        #        peptide = str_replace(peptide,'_1|-1|_2|-2', '')) %>%
                        ggplot(aes(reorder(peptide, MFI, median, na.rm = T), MFI)) +
                        geom_boxplot(aes(fill = str_detect(peptide, "BUFFER"))) +
                        labs(x = "spot", y = "Transformed MFI - Log2 Scale") +
                        scale_y_continuous(trans = "log2") +
                        theme_bw() +
                        theme(
                                plot.margin = unit(c(0.1, 0.1, 0.1, 0.1), "in"),
                                panel.grid = element_blank(),
                                axis.ticks = element_blank(),
                                axis.text.x = element_text(angle = 90),
                                legend.position = "none"
                        )
        }


fold_change_median_buffer <-
        function(Ig, method = log2.transform) {
                method(Ig) %>%
                        select(-matches("LAND|COMM")) %>%
                        gather("peptide", "MFI", -c(1:5)) %>%
                        group_by(slide, miniarray) %>%
                        mutate(fold_change = MFI - median(MFI[str_detect(peptide, "BUFFER")], na.rm =
T)) %>%
                        filter(!str_detect(peptide, "BUFFER")) %>%
                        ggplot(aes(
                                reorder(peptide, fold_change, median, na.rm = T),
                                fold_change
                        )) +
                        geom_boxplot() +
                        labs(x = "peptide", y = "fold change (ref - median BUFFER)") +
                        geom_hline(yintercept = c(-2, 2), col = "red") +
                        theme_bw() +
                        theme(
                                plot.margin = unit(c(0.1, 0.1, 0.1, 0.1), "in"),
                                panel.grid = element_blank(),
                                axis.ticks = element_blank(),
                                axis.text.x = element_text(angle = 90),
                                legend.position = "none"
                        )
        }

# Forest plot - like correlation plot of age and antibody responses filtered based on fold difference
# Compared with median of the Buffer spots

Corr_plot_FoldDiff_filtered <-

        function(Ig,
                 method,
                 fold_filter_method = log2.transform,
                 FoldDiff = 2,
                 age_filter = 6,
                 ...) {
                ## Filter peptide with fold change above 2
                fold_filter_method(Ig) %>%
                        select(-matches("LAND|COMM")) %>%
                        gather("peptide", "MFI", -c(1:5)) %>%
                        group_by(slide, miniarray) %>%
                        mutate(fold_change = MFI - median(MFI[str_detect(peptide, "BUFFER")])) %>%
                        ungroup() %>%
                        filter(!str_detect(peptide, "BUFFER")) %>%
                        group_by(peptide) %>%
                        mutate(median_foldchange = median(fold_change)) %>%
                        filter(median_foldchange >= FoldDiff) %>%
                        distinct(peptide) -> filtered_peptides

                data_1 <- method(Ig, ...) %>%
                        gather("spot", "MFI", -c(1:5)) %>%
                        filter(!str_detect(spot, "BUFFER|LAND|COMM")) %>%
                        filter(sample_no %in% age_data$sample_no) %>%
                        inner_join(., age_data, by = "sample_no") %>%
```

```
                        mutate(peptide = str_replace(spot, "_1|-1|_2|-2", "")) %>%
                        filter(age_m < age_filter, spot %in% filtered_peptides$peptide) %>%
                        split(., .$spot) %>%
                        map(~ cor.test(.$age_m, .$MFI, method = "spearman")) %>%
                        map_dfc(~ c(.$p.value, .$estimate)) %>%
                        t() %>%
                        as.data.frame() %>%
                        rownames_to_column("epitope") %>%
                        rename(p.value = V1,
                               coef = V2) %>%
                        mutate(
                                signif = ifelse(p.value < 0.05, 1, 0),
                                signif = factor(signif, levels = c(0, 1)),
                                x = seq_along(coef)
                        )

                data_1 %>%
                        ggplot(aes(reorder(x, coef), coef)) +
                        geom_text_repel(
                                aes(
                                        x = reorder(x, coef),
                                        y = coef,
                                        label = epitope
                                ),
                                family = "sans",
                                nudge_y = ifelse(data_1$coef < 0, -0.1, 0.1),
                                size = 3,
                                segment.alpha = 0.2,
                                segment.size = 1,
                                segment.color = "grey20",
                                arrow = arrow(
                                        type = "open",
                                        ends = "first",
                                        length = unit(0.08, "in")
                                )
                        ) +
                        geom_point(
                                aes(
                                        fill = p.value,
                                        col = signif,
                                        size = abs(coef)
                                ),
                                show.legend = F,
                                alpha = 0.9
                        ) +
                        scale_color_manual(values = c("0" = "grey", "1" = "blue")) +
                        scale_size_continuous(range = c(2, 6)) +
                        scale_y_continuous(limits = c(min(data_1$coef) - 0.1, max(data_1$coef) + 0.1))
+
                        labs(y = "Correlation Coefficient,

rho",
                             x = "Peptides") +
                        coord_flip() +
                        theme_wsj(color = "brown2", title_family = "sans") +
                        geom_hline(yintercept = 0,
                                   lty = "dotted",
                                   size = 0.1) +
                        theme(
                                axis.title.y = element_blank(),
                                axis.text.y = element_blank(),
                                panel.grid = element_blank(),
                                axis.title = element_text(size = 14, face = "bold"),
                                plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "in")
                        )
        }


# << Coeffients of Variations for the different Methods >> -----
# Function to extract the CVs

coef_var_plot <- function(Ig) {
        CV.fun <- function(dataset) {
                dataset %>%
                        select(-matches("BUFFER|LAND|COMM")) %>%
```

```
                filter(miniarray %in% c(1)) %>%
                gather("spot", "MFI", -c(1:5)) %>%
                group_by(spot) %>%
                summarise(coef = (function(x) {
                        (sd(x, na.rm = T) / mean(x, na.rm = T)) * 100
                })(MFI))
        }

        # List of the dataset
        data.list <- list(
                "Log2 transformation" = log2.transform(Ig),
                ComBat = ComBat.Peptide(Ig),
                DDHF = DDHF.peptide(Ig),
                "Moving-min BG subtraction" = moving_min_bg_subtract(Ig),
                "VSN-stratified (slide)" = VSN.transform(Ig, TRUE),
                "VSN single strata" = VSN.transform(Ig),
                LMM = LMM.transform(Ig),
                "LMM + ComBat" = LMM_ComBat(Ig),
                PAS_stabilisation = PAS_stabilisation(Ig)
        )


        data.list1 <- map(data.list, CV.fun)

        Peptide.CVs <- data.list1[[1]]

        i <- 2
        while (i <= length(data.list1)) {
                Peptide.CVs <- Peptide.CVs %>%
                        left_join(., data.list1[[i]], by = "spot")
                i <- i + 1
        }

        colnames(Peptide.CVs) <- c("spot", names(data.list))

        # CV distributions in all the methods applied
        # xlabels = c(seq(0,40,5), seq(40,200,60))
        # cv.trans = function(x){pmin(x,40) + 0.05*pmax(x-40,0)}


        CVs_Plot <-
                Peptide.CVs %>%
                gather("Method", "CV", -c(spot)) %>%
                filter(CV <= 20) %>%
                mutate(Method = factor(
                        Method,
                        labels = names(data.list),
                        levels = names(data.list)
                )) %>%
                ggplot(aes(reorder(Method, CV, median, na.rm = T), CV)) +
                geom_boxplot(alpha = 0, outlier.alpha = 0) +
                # geom_rect(aes(xmin = 0.3,xmax = 11.7, ymin = 40, ymax = 40.001), fill = 'grey')+
                # scale_y_continuous(limits = c(0,NA), breaks = cv.trans(xlabels), labels = xlabels)+
                geom_quasirandom(size = 0.4) +
                # scale_y_continuous(limits = c(0,100))+
                geom_hline(yintercept = 5,
                           col = "red",
                           size = 0.8) +
                labs(y = "Coefficient of Variation(%)",
                     x = "Normalization method") +
                theme_bw() +
                theme(
                        axis.text = element_text(face = "bold",
                                                 size = 12),
                        panel.grid.minor.x = element_blank(),
                        panel.grid = element_line(color = "grey97"),
                        axis.title = element_text(size = 16)
                ) +
                coord_flip()

        return(list(Peptide.CVs, CVs_Plot))
}
```

Appendix 4: Research proposal KNH-UON ERC research ethics approval letter

UNIVERSITY OF NAIROBI
COLLEGE OF HEALTH SCIENCES
P O BOX 19676 Code 00202
Telegrams: varsity
Tel:(254-020) 2726300 Ext 44355

KNH-UON ERC
Email: uonknh_erc@uonbi.ac.ke
Website: http://www.erc.uonbi.ac.ke
Facebook: https://www.facebook.com/uonknh.erc
Twitter: @UONKNH_ERC https://twitter.com/UONKNH_ERC

KENYATTA NATIONAL HOSPITAL
P O BOX 20723 Code 00202
Tel: 726300-9
Fax: 725272
Telegrams: MEDSUP, Nairobi

Ref: KNH-ERC/A/313

14th August, 2019

Mutua John Mutiso
Reg. No.W62/86996/2016
UNITID
College of Health Sciences
University of Nairobi

KENYATTA NATIONAL HOSPITAL APPROVED 14 AUG 2019 KNH/UoN-ERC P.O. BOX 20723-00202 NAIROBI

Dear John

RESEARCH PROPOSAL: SYSTEMATIC VARIANCE CORRECTION METHODS FOR PEPTIDE MICROARRAY DATA (P397/05/2019)

This is to inform you that the KNH- UoN Ethics & Research Committee (KNH- UoN ERC) has reviewed and **approved** your above research proposal. The approval period is 14th August 2019 - 13th August 2020.

This approval is subject to compliance with the following requirements:

a. Only approved documents (informed consents, study instruments, advertising materials etc) will be used.
b. All changes (amendments, deviations, violations etc.) are submitted for review and approval by KNH-UoN ERC before implementation.
c. Death and life threatening problems and serious adverse events (SAEs) or unexpected adverse events whether related or unrelated to the study must be reported to the KNH-UoN ERC within 72 hours of notification.
d. Any changes, anticipated or otherwise that may increase the risks or affect safety or welfare of study participants and others or affect the integrity of the research must be reported to KNH- UoN ERC within 72 hours.
e. Clearance for export of biological specimens must be obtained from KNH- UoN ERC for each batch of shipment.
f. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. (*Attach a comprehensive progress report to support the renewal*).
g. Submission of an *executive summary* report within 90 days upon completion of the study. This information will form part of the data base that will be consulted in future when processing related research studies so as to minimize chances of study duplication and/ or plagiarism.

For more details consult the KNH- UoN ERC website **http://www.erc.uonbi.ac.ke**

Yours sincerely,

PROF.M.L. CHINDIA
SECRETARY, KNH-UoN ERC

c.c. The Principal, College of Health Sciences, UoN
The Director, CS, KNH
The Chairperson, KNH- UoN ERC
The Assistant Director, Health Information, KNH
The Director, UNITID,UoN
Supervisors: Dr. Anne Wang'ombe, UNITID, UON
Dr. Charles Sande, KEMRI/Wellcome-Trust Kilifi
Dr.Nelson Kibinge, KEMRI/Wellcome-Trust Kilifi

Appendix 5: Turnitin plagiarism report page

Thesis Draft

ORIGINALITY REPORT

9%	6%	3%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ir.jkuat.ac.ke Internet Source	1%
2	datadryad.org Internet Source	1%
3	Submitted to RMIT University Student Paper	<1%
4	Submitted to University of Cambridge Student Paper	<1%
5	Submitted to University of Sheffield Student Paper	<1%
6	stackoverflow.com Internet Source	<1%
7	Submitted to Kenyatta University Student Paper	<1%
8	Submitted to Universiti Malaysia Kelantan Student Paper	<1%
9	insis.vse.cz Internet Source	<1%