2012•2013
FACULTY OF SCIENCES
Master of Statistics: Biostatistics

Masterproef

Implementation of mitigated fraction estimators in SAS based on existing R package

Promotor :
Prof. dr. Ziv SHKEDY

Promotor :
Dr. DAN LIN

Victoriah Nyaga

Master Thesis nominated to obtain the degree of Master of Statistics , specialization Biostatistics

Transnational University Limburg is a unique collaboration of two universities in two countries: the University of Hasselt and Maastricht University.

universiteit hasselt
KNOWLEDGE IN ACTION

Universiteit Hasselt | Campus Hasselt | Martelarenlaan 42 | BE-3500 Hasselt
Universiteit Hasselt | Campus Diepenbeek | Agoralaan Gebouw D | BE-3590 Diepenbeek

universiteit hasselt | Maastricht University

2012•2013
FACULTY OF SCIENCES
Master of Statistics: Biostatistics

Masterproef

Implementation of mitigated fraction estimators in SAS based on existing R package

Promotor :
Prof. dr. Ziv SHKEDY

Promotor :
Dr. DAN LIN

Victoriah Nyaga

Master Thesis nominated to obtain the degree of Master of Statistics , specialization Biostatistics

universiteit hasselt | Maastricht University

To my younger sister Vivian,

Acknowledgements

The journey through the Masters' programme is a culmination of efforts from different people to whom I am very grateful. I begin by thanking Dr. Charles Mbogo, Dr. Thomas Achia and Mrs. Wang'ombe for believing in me and writing a propelling recommendation for the VLIROUS scholarship.

Secondly, I wish to thank the Flemish Interuniversity Council (VLIR) for offering me the scholarship through which the Masters programme and my stay in Belgium was financed. Much gratitude to Mrs. Martine Machiels and Mr. Marc Thoelen for the fine arrangements of accommodation, flights, City trips … etc. The timely and meticulous arrangements made my relocation and stay in Belgium enjoyable and unforgettable.

Thirdly, I acknowledge the teaching staff at Center for Statistics (CenStat) who taught with vigor and passion. Special thanks to my thesis supervisors Prof. dr Ziv Shkedy and Dr. Dan Lin for their guidance and support.

Many thanks to all my classmates and especially those whom I worked with in various projects. Working in groups was a challenge, but it was an opportunity to learn how to work in an interdisciplinary and international environment. I really enjoyed the consultations and sharing of ideas.

Most importantly, I wish to thank my family; mum and dad for believing in me and letting me pursue my academic interest, their prayers, words of encouragement, guidance and warm love. To my younger brother and sister; thank you guys for your support and love.

I offer mass gratitude to all other people who helped me in one way or another during the Masters' programme and my stay in Belgium; the overall Kenyan community; JOSS, and many others that I might have forgotten.

Finally, I am grateful to Almighty God to whom I attribute all my success and to whom my life belongs.

To all, I will be forever grateful.

Victoriah Nyaga

September 1, 2013.

Preface

"Productivity is never an accident. It is always the result of a commitment to excellence, intelligent planning, and focused effort."

Paul J. Meyer

American entrepreneur and author

Table of Contents

List of Tables and Appendix

Tables

Appendices

Original Publication

This dissertation is based on the following publication;

Siev, D. (2005). An estimator of intervention effect on disease severity. Journal of Modern Applied Statistical Methods 4(2): 500-508.

Abstract

Mitigated fraction is frequently used to evaluate the effect of an intervention in reducing the severity of a particular outcome, a common measure in vaccines study. It utilizes rank of the observations and measures the overlap of the two distributions using their stochastic ordering. In a vaccine trial, mitigated fraction is used to estimate the relative increase in probability that a disease will be less severe in the vaccinated group.

SAS macros have been developed using SAS/IML in equivalence with existing R functions in MF package to estimate the mitigated fraction both for independent and clustered data. The macros also provide asymptotic and bootstrap-based confidence interval.

The macros were evaluated using real life data from a vaccine study and were validated by comparing output generated by the equivalent existing R functions available in MF package.

Keywords

Bootstrap, Macro, Mitigated fraction, R, SAS/IML.

Chapter 1

Introduction

1.1 Background

Depending on the study design and nature of the response, different types of vaccine effects of interest may be estimated. These include; vaccine efficacy for susceptibility (VEs); a measure of how well protected a vaccinated animal/person is, vaccine efficacy for progression (VEp); measures vaccine efficacy in preventing a post-infection outcome, etc. The main difference between VEs and VEp is that VEs evaluates susceptibility and exposure needs to be taken into account while VEp is conditional on being infected; and so the progression with the infected is important (Halloran *et al.*, 2009). In their book, Halloran *et al*. (2009) provide the formulas to compute the different types of vaccine effects.

Evaluating the effect of vaccination on an outcome that occurs after infection, VEp, requires comparison of morbidity or mortality in infected vaccinated subjects with that in infected unvaccinated individuals. In such a setting, the interest could be the effect of vaccine on; the probability of developing disease, the time from infection to developing the disease. For this two vaccine effects, the outcome is a binary response. Another post-infection vaccine effect is the effect of vaccination on reducing severity of disease for which the outcome is a continuous response indication the severity of disease in vaccinated and unvaccinated subjects (Halloran *et al.*, 2009).

The vaccine effect which has drawn most attention and is of main interest to researchers is prevented fraction (PF), commonly known as vaccine efficacy (VEs). Of the three post-infection vaccine effects; substantial attention and research has been given to estimation of vaccine effect on probability of developing disease and on time to developing the disease.

Estimation of post-infection vaccine effect where vaccination reduces the degree of disease severity is an important and current problem in vaccine studies. In such a setting, a recently developed estimator; mitigated fraction (MF) is an appropriate measure to use. It is a conditional estimate in a sense that it is estimated conditional on disease/infection. In probabilistic terms, MF measures the effect of vaccination on probability of severe disease (Halloran *et al*., 2009). In contrast to PF estimation where the typical analysis includes all individuals in the study, MF estimation includes only those that become infected. Careful definition of the disease outcomes and levels of disease severity is critical (Halloran *et al.*, 2009).

Early works related to mitigated fraction include Greenwood and Yule (1915) who presented data from Pearson on the effect of smallpox vaccination to prevent death by comparing the number of cases recovering to those dying of smallpox.

Research on estimating vaccine efficacy on post-infection outcome is continuing. Hudgens *et al*. (2004) presented a technical report on causal vaccine effects on binary post-infection outcomes. Siev (2005) outlined the origin and structure of MF and also described its similarities to PF. He (Siev, 2005) presented several estimators of MF for various purposes based on the Wilcoxon rank sum statistic. The estimators use ranks of the observations and measures the overlap of two empirical distributions based on their stochastic ordering. In its computation, the study subjects are ranked depending on the disease severity which can be graded by a continuous measure or discrete assessment. The ranks assigned to the outcomes in the vaccinated group are summed up and the vaccine judged to be effective when the sum is sufficiently large.

In 2012, Dr. Siev (Siev, 2012) went on and implemented a package to calculate several types of MF for clustering data and with bootstrap options in R programming Language (R Core Team, 2013).

1.2 Objective

The main objective of this thesis project was to implement the existing functions of MF estimators and confidence intervals in SAS 9.2 (SAS Institute, 2011). The algorithms are wrapped up in SAS macros and majorly implemented using the Interactive Matrix Language (IML). The newly developed macros are then used on a real life vaccine trial to estimate the mitigated fraction and the 95% confidence intervals with/without the effect of clustering.

The contents of this thesis are organized as follows: Chapter 2 gives a short description of a vaccine study to set the context. Chapter 3 covers the methodology of estimating the MF and presents the various estimators of MF, confidence interval estimation procedures, and a description of the SAS macros. Chapter 4 provides the results and their corresponding interpretation; while Chapter 5 provides a comparison with R functions and output. The final chapter ends with some discussion and conclusion. The complete SAS macros are included in the Appendix.

Chapter 2

Case Study

2.1 Study Design

A generalized randomized block design study was setup involving dogs to demonstrate immunogenicity of a vaccine under development against challenge with virus of interest in susceptible dogs. In total thirty dogs were used; ten dogs were in the control group (T01), and the other twenty were in the vaccinated group (T02). The dogs were allocated to treatment groups and rooms according to a randomization plan produced by a Biometrician. The dogs were healthy and sero-negative (SN) to the virus by serum neutralization (SN titer <2) and free of the virus by virus isolation on Day 0 (day of treatment/vaccination). Unthrifty animals were excluded from the study prior to administration of the vaccine.

2.2 Vaccination

During the vaccination phase, the dogs were housed in two separate isolation rooms by treatment with two dogs per room for the control group and four animals per room for the vaccinate group. Strict bio-security measures were followed. The room housing dogs in the control group was entered first to avoid any cross-contamination. The vaccines were administered on Day 0. The vaccination phase of the study had a completely random design with subsampling. Rooms were the sub-sample and dogs were the sub-sub-sample. Room was the experimental unit for treatment. The experimental units for treatment were not replicated.

2.3 Virus Challenge

A day before challenge administration, the dogs were co-mingled according to the allotment in two rooms with six animals per room (four dogs from T02 and two dogs from T01). Each

dog then received a target challenge dose of the virus by intranasal aerosolization. The challenge phase of the study had a generalized block design in two rooms. Blocking was based on date of birth and litter and animal was the experimental unit. There were five clusters in total.

2.4 Statistical Analysis

The number of days that a dog had virus detected in the nasal swabs post challenge (first day virus detected through the last day virus detected) was calculated for each dog and used as the response variable. The following will be calculated; mitigated fraction for independent and clustered data and Hodges-Lehmann median of differences (Hodges and Lehmann, 1963). The accompanying 95% bootstrap confidence intervals will be calculated as well. Block will be the stratification variable.

Chapter 3

Methodology

3.1 Definitions

The MF estimators proposed by Siev (2005) are mainly based on the Wilcoxon rank sum statistic (Wilcoxon, 1945). The response variable is a continuous measure or discrete assessment indicating the grading of disease severity where larger values indicate more disease severity.

3.1.1 Wilcoxon rank sum statistic (W)

The Wilcoxon rank sum statistic is a non-parametric statistical method based on ranks. In its computation, the study subjects are ranked depending on the disease severity. Tied observations are assigned the average rank. Let the ranks assigned to the vaccinated subjects to be denoted by S_1, S_2, …,S_n. The ranks assigned to the outcomes in the vaccinated group are summed up to form Wilcoxon rank sum statistic W;

$$W = S_1 + S_2 + \ldots + S_n, \quad (1)$$

and the vaccine judged to be effective when the sum is sufficiently large than a constant c denoting a specified critical value, say $W \geq c$.

3.1.2 Mitigated Fraction (MFr)

Beyond assessing the (non-)effectiveness of the vaccine as achieved by computing the Wilcoxon rank sum statistic, one may wish to consider the effect of vaccination on the relative probability that the disease is milder. This can be computed using MF expressed as follows:

$$MF = 1 - \frac{t_2}{t_0} \quad (2)$$

where t_2 is the estimated probability that a vaccinated subject disease status is severe than that of the unvaccinated subject while t_0 is the probability of greater severe disease in the absence of the vaccination.

Let Y_{ih} be the response in of subject *h (h = 1, 2, …, n_i)* in group *i(i=1,2)*. A non-parametric estimator, mean ridit (Bross, 1958), T_i, is then defined to distinguish between units from two populations; the control and vaccinated groups, $T_i = prob(Y_{ih} > Y_{jk})$. T_i is an estimate of the probability that a subject *h* in group *i* is 'worse-off' than a subject *k* in the comparison group *j*. The mean ridit estimate for the comparison group *j* will always be 0.5.

The estimator can then rescaled to range from -1 to 1 as follows; MF= $\theta_i = 2(T_i - 0.5)$ with 0 corresponding to the null probability; when the two distributions are stochastically identical and positive values indicating that the vaccine is protective and negative values indicate otherwise. Wolfe and Hogg (1971) showed how the estimate T_i may be recovered from the Wilcoxon rank sum or the Mann-Whitney U statistic as follows;

$$T_i = \left\{W_i - \frac{n_i(n_i+1)}{2}\right\} * \frac{1}{n_i n_j} = \frac{U_i}{n_i n_j} \quad (3)$$

where W_i is the sum of ranks in group *i*, n_i is the sample size in group *i*. U_i is the number of times a y_{jk} precedes a y_{ih} expressed as;

$$U_i = \sum_{k=1}^{nj} \sum_{h=1}^{ni} H(y_{jk}, y_{ih}); \quad (4)$$

and

$$H(a, b) = \begin{cases} 1 \; if \; a < b \\ 0 \; if \; a > b \\ 0.5 \; if \; a = b \end{cases} \quad (5)$$

Substituting $\theta_i = 2(T_i - 0.5)$ gives,

$$MF = \theta_i = \frac{\{2W_i - n_i(1 + n_i + n_j)\}}{n_i n_j} \quad (6)$$

3.1.3 Mitigated Fraction for clustered/stratified data (MFClus)

The estimate T_i from clustered data is expressed as follows;

$$T_i = \frac{\sum_r U_{ir}}{\sum_r n_{ir} n_{jr}} \quad (7)$$

where r indexes the strata. Basically the statistic is averaging the U statistic over the clusters. Clusters that do not include both treatments are excluded. For matched pairs, the expression reduces to a simple binomial fraction;

$$T_i = \sum_r I(y_{jr} < y_{ir})/R \quad (8)$$

where R is the number of pairs and I(.) is an indicator function. In this setting; interval estimation can be done using the familiar methods for binomial fractions.

3.1.4 Subject Components (MFSubj)

Mitigated fraction may be decomposed into contribution by individual subjects. The contribution by a vaccinated subject *h* is;

$$s_h = \frac{2}{n_1} \sum_{k=1}^{n_1} H(y_{2h}, y_{1k}) - 1 \quad (9)$$

Mitigated fraction is then the mean of the individual subject components expressed as;

$$MF = \frac{1}{n_1} \sum_{j=1}^{n2} s_h \quad (10)$$

3.1.5 Hodges-Lehman Median Estimator (HL)

After evaluating whether the vaccine is effective or not; one would wish to determine by how much the two groups differ. The Hodges-Lehmann Median of differences gives an estimate of how big is the difference between two populations. It is a robust method and provides the best unbiased estimator of the median of all possible pairwise differences of responses from control and vaccinated subjects (Helsel and Hirsch, 2002). It is an estimator of shift in

location between two distributions assuming that they have the same shape. Let n_1 and n_2 be the sample size of the control and vaccinated groups, respectively. Then, $n_1 * n_2$ pairs of observations can be made and each pair gives a difference of values. The Hodges-Lehman estimator will be the median of the $n_1 * n_2$ differences denoted by;

$$HL = \text{Median}\,(Y_{ih} - Y_{jk});\ h\,(k) = 1, 2, \ldots, n_1(n_2) \quad (11)$$

3.2 Confidence Interval Estimation

Confidence intervals can be derived using normal approximation based on asymptotic theory. However, these asymptotic approximations can be quite inaccurate in practice and they impose assumptions that are preferably avoided (Siev, 2005). Furthermore, the normal approximation framework breaks down when the sample size is small. A very useful alternative, especially in cases where parameter variance function has no closed-form formula is the bootstrap methods (Efron and Tibshirani, 1993). Bootstrap methods allow one to make inference from the data without making strong distributional assumptions about the statistic being computed. Bootstrap uses resampling with replacement (Monte Carlo resampling) to estimate a parameter sampling distribution from which the confidence intervals may be easily obtained. It is worth noting that bootstrap methods are computationally intensive and still under development.

3.2.1 Equal Tailed Confidence Interval (Percentile Confidence Interval)

The equal tailed (1-α)*100% level confidence interval for a parameter θ is of the form $[\hat{\theta} - \hat{a}, \hat{\theta} - \hat{b}]$ where $\hat{\theta}$ is the point estimate and $\hat{a}$, $\hat{b}$ are chosen such that;

$$P\left(\theta < \hat{\theta} - \hat{a}\right) = P\left(\theta > \hat{\theta} + \hat{b}\right) = \frac{1}{2}\alpha \quad (12)$$

This type of confidence intervals is simple to compute and always provide admissible parameter values within the interval. However, the coverage error is substantial if the distribution of θ is not nearly symmetric (Carpenter and Bithell, 2000).

3.2.2 Highest Probability Density Confidence Interval (HPD)

The (1-α)-level HPD confidence interval is the shortest interval such that P (a ≤θ≤b) = 1- α, where a and b are the upper and lower confidence intervals. For a unimodal symmetric parameter distribution, the equal tail and the HPD confidence intervals coincide. The HPD confidence intervals would be more suitable than the percentile confidence intervals when the parameter distribution is skewed.

3.2.3 Bias Corrected and Accelerated Confidence Intervals (BCa)

The development of the BCa confidence intervals followed the shortcomings of the equal tailed confidence intervals (Carpenter and Bithell, 2009) and in particular skewness and/or bias. The BCa confidence intervals are of the form; [$\hat{\theta}^{\alpha_1}$, $\hat{\theta}^{\alpha_2}$] where;

$$\alpha_1 = \phi(\hat{z}_0 + \frac{\hat{z}_0 + z^{(\alpha)}}{1 - \hat{a}(\hat{z}_0 + z^{(\alpha)})}),$$

$$\alpha_2 = \phi(\hat{z}_0 + \frac{\hat{z}_0 + z^{1-(\alpha)}}{1 - \hat{a}(\hat{z}_0 + z^{1-(\alpha)})}),$$

$$\hat{z}_0 = \phi^{-1}(\frac{\#\{\hat{\theta}_b^* < \hat{\theta}\}}{B}),$$

$$\hat{a} = \frac{\sum_{i=1}^{n}(\hat{\theta}^{(.)} - \hat{\theta}^{(-i)})^3}{6\{\sum_{i=1}^{n}(\hat{\theta}^{(.)} - \hat{\theta}^{(-i)})^2\}^{\frac{3}{2}}}. \quad (13)$$

$\hat{z}_0$ is the bias correction,

$\hat{a}$ the acceleration coefficient to adjust for non-constant variance within the resampled datasets,

ϕ is the standard normal cumulative distribution function,

$\hat{\theta}_b^*$ is the b[th] bootstrap replication,

$\hat{\theta}^{(.)}$ is the parameter estimate computed from all observations,

$\hat{\theta}^{(.)}$is the parameter estimate computed with all but the i^{th} observation, and

B is the number of bootstraps.

When the parameters 'a' and 'z_0' are zero, the BCa and equal tailed intervals coincide. While the obtained parameter values are all admissible, the calculation of the acceleration parameter 'a' can be tedious especially in complex parameter problems. Furthermore, the coverage error will increase as α goes to zero. The BCa confidence intervals are correct and accurate (Carpenter and Bithell, 2000). In some cases, the BCa confidence intervals could wider or shorter than the percentile confidence intervals.

3.3 Development of SAS Modules

The computation of the above discussed estimators has been implemented as SAS programs. The programs are written in SAS/IML language, which is mainly used to manipulate numeric and character vectors/matrices. The SAS/IML grammar is very similar to that of R statistical software. It is not possible to use the SAS procedures within the SAS/IML unless by use of submit/endsubmit statements.

Currently, there exist no SAS procedures to provide the various MF estimators apart from the Hodges-Lehmann estimator of location shift provided by PROC NPAR1WAY. The SAS programs are a translation of an existing R package to compute the MF estimators.

To facilitate the use of the programs, SAS macro Language has been used for passing parameters. The following SAS code can be used to include the SAS macro in a SAS program for obtaining the mitigated fraction.

```
FILENAME fileloc 'G: \SAS_MF';
%INCLUDE fileloc(MFr);
```

The table below gives a summary of the written macros and their purposes;

Table 1: Summary of the SAS macros and their purposes

SAS Macro	Purpose	Parameter arguments
`%MFR();`	Compute mitigated fraction.	`IN=,Y=,GRP=G,CON=,VAC=,`
`%MFBOOT();`	Provide bootstrap confidence intervals for mitigated fraction.	`IN=,Y=,GRP=,CON=,VAC=,NBOOT=,ALPHA=,HPD=,BCA=,RETURNBOOT=,`
`%HLBOOT();`	Provide mitigated fraction, Hodges-Lehmann, and Quartile differences and their bootstrap confidence intervals.	`IN=,Y=,GRP=,CON=,VAC=,NBOOT=,ALPHA=,HPD=,BCA=,RETURNBOOT=,`
`%MFCLUS();`	Compute mitigated fraction in clustered data.	`IN=,Y=,GRP=,CON=,VAC=,CLUSTER=,CLUSTERNUM=,`
`%MFCLUSBOOT();`	Provide bootstrap confidence for mitigated fraction in clustered data.	`IN=,Y=,GRP=,CON=,VAC=,CLUSTER=,NBOOT=,RETURNBOOT=,HPD=,BOOTCLUSTER=,BOOTUNIT=,ALPHA=,CLUSTERNUM=,`
`%MFMP();`	Compute mitigated fraction in paired data.	`DATA=,IN=,Y=RESP,GRP=GROUP,CON=,VAC=,CLUSTER=,CLUSTERNUM=,ALPHA=,DF=,TDIST=,`
`%MFSUBJ();`	Compute the subject components of mitigated fraction.	`IN=,Y=,GRP=,CON=,VAC=,`

1. **`%MFR(IN=,Y=,GRP=G,CON=,VAC=,)`**

Purpose: To compute the mitigated fraction.

Parameters: `IN` = *dataset* - Input SAS dataset.

Y = *variable* - A continuous response variable.

GRP = *variable* - A factor variable with at least two levels.

CON = *'text'* - Text indicating the control/reference group.

VAC = *'text'* - Text indicating the comparative group.

Output: A point estimate of mitigated fraction (mfr).

2. **%MFBOOT(IN=,Y=,GRP=,CON=,VAC=,NBOOT=,ALPHA=,HPD=,BCA=, RETURNBOOT=,);**

Purpose: To do the resampling and compute (1-α)*100% percentile (perc), highest probability density (HPD) and bias corrected and accelerated bootstrap confidence intervals (BCa). By default the percentile confidence intervals are computed.

Parameters: IN = *dataset* - Input SAS dataset.

Y = *variable* - A continuous response variable.

GRP = *variable* - A factor variable with at least two levels.

CON = *'text'* - Text indicating the control/reference group.

VAC = *'text'* - Text indicating the comparative group.

NBOOT = *number* - Number of bootstrap samples to take.

ALPHA = *number* - The complement of the required confidence level.

HPD = *0/1* - Estimate HPD confidence intervals? Yes (1) or No (0).

BCA = *0/1* - Estimate BCa confidence intervals? Yes (1) or No (0).

RETURNBOOT = *0/1* - Save the bootstrap samples? Yes (1) or No (0)

SEED = *number* - Random seed number.

Output: The estimated (observed) mitigated fraction (mfr), the $\alpha/2*100^{th}$, 50^{th} and $(1-\alpha/2)*100^{th}$ bootstrap quantiles as calculated by the equal tailed (perc), HPD and BCa procedures of constructing confidence interval. In the output also, is the number of bootstraps performed (nboot) and the value of the complement of the required confidence level (apha).

3. %MFCLUS(IN=,Y=,GRP=,CON=,VAC=,CLUSTER=,CLUSTERNUM=);

Purpose: To compute mitigated fraction in each cluster as well as the overall point estimate.

Parameters: IN = *dataset* - Input SAS dataset.

Y = *variable* - A continuous response variable.

GRP = *variable* - A factor variable with at least two levels.

CON = *'text'* - Text indicating the control/reference group.

VAC = *'text'* - Text indicating the comparative group.

CLUSTER = *variable* - Variable name with the cluster labels.

CLUSTERNUM = *0/1* - Are cluster level numeric (1) or character (0).

Output: The estimates of the Wilcoxon rank sum statistic, Mann-Whitney statistic, mean ridit, sample sizes for each treatment group, and mitigated fraction for each cluster. Also provided are the overall estimates. Any clusters that have excluded because of missing one or both treatment groups are also shown in the output.

4. %MFCLUSBOOT(IN=,Y=,GRP=,CON=,VAC=,CLUSTER=,NBOOT=,RETURNBOOT=,HPD=, BOOTCLUSTER=, BOOTUNIT=, ALPHA=, CLUSTERNUM=);

Purpose: To do the resampling and provide (1-α)*100 % equal tailed and HPD bootstrap confidence intervals for stratified mitigated fraction.

Parameters: IN = *dataset* - Input SAS dataset.

Y = *variable* - A continuous response variable.

GRP = *variable* - A factor variable with at least two levels.

CON = *'text'* - Text indicating the control/reference group.

VAC = *'text'* - Text indicating the comparative group.

NBOOT = *number* - Number of bootstrap samples to take.

ALPHA = *number* - The complement of the required confidence level.

HPD = *0/1* - Estimate highest density intervals? Yes (1) or No (0).

RETURNBOOT = *0/1* - Save the bootstrap samples? Yes (1) or No (0).

BOOTCLUSTER = *0/1* - Resample the cluster? Yes (1) or No (0).

BOOTUNIT = *0/1* - Resample the units within cluster? Yes (1) or No (0).

CLUSTERNUM = *0/1* - Cluster labels numeric? Yes (1) or No (0).

SEED = *number* - Random seed number.

For a generalized randomized block design we recommend to bootstrap the clusters as well as the units within the clusters. For randomized complete block design, bootstrapping the clusters only is recommended to maintain balance within the clusters.

Output: The overall estimated (observed) stratified mitigated fraction along with the bootstrap quantiles at $\alpha/2*100^{th}$, 50^{th} and $(1-\alpha/2)*100^{th}$ as calculated by the equal tailed, HPD procedures. The number of bootstraps performed (nboot) and the value of alpha indicating the complement of the required confidence level is also included in the output.

5. %MFMP(DATA=,IN=,Y=RESP,GRP=GROUP,CON=,VAC=,CLUSTER=, CLUSTERNUM=,ALPHA=,DF=,TDIST=);

Purpose: To compute the mitigated fraction in each pair, the overall point estimate, as well as the (1-α)*100% asymptotic confidence intervals.

Parameters: DATA = *0/1* - 0 if the input is the trinomial vector, 1 if the full dataset is available.

IN = *dataset* - Input SAS dataset, the vector or full dataset.

ALPHA = *number* - The complement of the required confidence level.

DF = *number* - Degrees of freedom.

TDIST = *0/1* - Use quantiles from students' t (1) or Gaussian (0) distribution for confidence intervals?

When DATA = *1*, the following parameters are required to be passed on estimate the trinomial vector.

Y = *variable* - A continuous response variable.

GRP = *variable* - A factor variable with at least two levels.

CON = *'text'* - Text indicating the control/reference group.

VAC = *'text'* - Text indicating the comparative group.

CLUSTER = *variable* - Variable indicating the clusters.

CLUSTERNUM = *0/1* - To indicate whether the clusters are character (0) or numerical (1).

Output: The (1-α)*100% asymptotic confidence interval computed using quantiles either from normal or students' t distribution and value of alpha indicating the complement of the required confidence level. If the quantiles from the students' distribution are used, then the degrees of freedom as also displayed.

6. %MFSUBJ(IN=,Y=,GRP=,CON=,VAC=,);

Purpose: To compute the contribution of each subject to the overall mitigated fraction.

Parameters: IN = *dataset* - Input SAS dataset.

Y = *variable* - A continuous response variable.

GRP = *variable* - A factor variable with at least two levels.

CON = *'text'* - Text indicating the control/reference group.

VAC = *'text'* - Text indicating the comparative group.

Output: The estimate for the overall mitigated fraction. Summary tables showing how many unique values of mitigated fraction were computed and their corresponding estimates. For each unique subject mitigated fraction, the range of the response outcome is also displayed.

7. `%HLBOOT(IN=,Y=,GRP=,CON=,VAC=,NBOOT=,ALPHA=,HPD=,BCA=,RETURNBOOT=);`

Purpose: To estimate the Hodges-Lehmann median of the differences, the (1-α)*100% equal tailed and HPD bootstrap confidence intervals of the quartiles in the control and vaccinated group and their corresponding difference of quartiles.

Parameters: `IN` = *dataset* - Input SAS dataset.

`Y` = *variable* - A continuous response variable.

`GRP` = *variable* - A factor variable with at least two levels.

`CON` = *'text'* - Text indicating the control/reference group.

`VAC` = *'text'* - Text indicating the comparative group.

`NBOOT` = *number* - Number of bootstrap samples to take.

`ALPHA` = *number* - The complement of the required confidence level.

`HPD` = *0/1* - Estimate HPD confidence intervals? Yes (1) or No (0).

`BCA` = *0/1* - Estimate BCa confidence intervals? Yes (1) or No (0).

`RETURNBOOT` = *0/1* - Save the bootstrap samples? Yes (1) or No (0).

`SEED` = *number* - Random seed number.

Output: The estimated (observed) mitigated fraction and Hodges-Lehman median of differences. The 25^{th}, 50^{th} and 75^{th} quartiles of the response outcome in the control and treatment group and the corresponding difference. The corresponding $\alpha/2*100^{th}$, 50^{th} and $(1-\alpha/2)*100^{th}$ bootstrap quantiles. The number of bootstraps performed (nboot) and value of alpha indicating the complement of the required confidence level.

Chapter 4

Results

4.1 Mitigated Fraction

Evaluation

```
%MFR(IN=DF1, Y=TOT_DAYS, GRP=TRT, CON='T01', VAC='T02');
```

SAS output

Estimated Mitigated Fraction

MF
0.9000

Interpretation

The relative increase in the probability that a vaccinated dog had a less severe disease than a unvaccinated dog was 0.9. This implies that the vaccine benefited 90% of half of the vaccinated dogs which in the absence of vaccination would have been more severely affected than the unvaccinated dogs. From the relation that MF = 2*T – 1, T= 0.95, implying that 95% of the unvaccinated dogs are expected to be more severely affected than the vaccinated dogs. Since this estimated mitigated fraction is greater than 0, then the vaccine is said to be protective.

4.2 Bootstrap confidence intervals for Mitigated Fraction

Evaluation

```
%MFBOOT(IN=DF1, Y=TOT_DAYS, GRP=TRT, CON="T01", VAC="T02", ALPHA=0.05,
HPD=1, BCA=1, RETURNBOOT=0, NBOOT=10000, SEED=123);
```

SAS output

Bootstrap Confidence Intervals

	Observed	Lower	Median	Upper
Percentile	0.9000	0.7000	0.9000	1.0000
HPD	0.9000	0.7000	0.9000	1.0000
BC.a	0.9000	0.5000	0.8000	1.0000

Number of Bootstraps

10000

Alpha

0.05

Interpretation

The estimated (observed) mitigated fraction was 0.9. The 95% equal tailed and highest probability density confidence intervals were similar suggesting symmetry of the parameter distribution with lower and upper interval at 0.7 and 1, respectively. The BCa confidence intervals were wider with lower and upper interval at 0.5 and 1, respectively. It is worth noting, that it is by chance that the BCa confidence intervals are wider.

4.3 Hodges-Lehmann Median of Differences

Evaluation

```
%HLBOOT(IN=DF1, Y=TOT_DAYS, GRP=TRT, CON='T01', VAC='T02',NBOOT=10000,
ALPHA=0.05,HPD=1,BCA=1,RETURNBOOT=0, SEED=123);
```

SAS output

Estimated Mitigated Fraction

	Observed	Lower	Median	Upper
Percentile	0.9000	0.7000	0.9000	1.0000
HPD	0.9000	0.7000	0.9000	1.0000
BC.a	0.9000	0.5000	0.8000	1.0000

Hodges-Lehmann Estimate

	Observed	Lower	Median	Upper
Percentile	-5.0000	-6.0000	-5.0000	-4.5000
HPD	-5.0000	-6.0000	-5.0000	-5.0000

Estimated Quartile Differences

Quartile	Observed	Lower	Median	Upper
Q25	-5.0000	-5.0000	-5.0000	0.0000
Q50	-5.0000	-6.0000	-5.0000	-4.5000
Q75	-6.0000	-7.0000	-6.0000	-5.0000

Estimated Quartiles of ‘con’

Quartile	Observed	Lower	Median	Upper
Q25	5.0000	0.0000	5.0000	5.0000
Q50	5.0000	4.5000	5.0000	6.0000
Q75	6.0000	5.0000	6.0000	7.0000

Estimated Quartiles of 'vac'

Quartile	Observed	Lower	Median	Upper
Q25	0.0000	0.0000	0.0000	0.0000
Q50	0.0000	0.0000	0.0000	0.0000
Q75	0.0000	0.0000	0.0000	0.0000

Interpretation

The estimated (observed) median of differences of responses in vaccinated and unvaccinated dogs was -5.0 implying that location of distribution of responses of vaccinated dogs was shifted by 5 units(days) relative to the distribution of the responses of unvaccinated dogs. The 95% equal tailed and HPD confidence intervals were [-6.0, -4.5] and [-6.0, -5.0], respectively. Since the equal tailed and HPD confidence intervals do not coincide, it suggested that the parameter distribution is skewed. From the 50th quartile of 'vac' and 'con' responses, no virus was detected in half of the vaccinated dogs but was detected after five days in half of the dogs that were given the control vaccine.

4.3 Stratified Mitigated Fraction

Evaluation

```
%MFCLUS(IN=DF1, Y=TOT_DAYS, GRP=TRT, CON='T01', VAC='T02', CLUSTER=BLK,
CLUSTERNUM=1);
```

SAS output

Estimated Mitigated Fraction

Overall

w	u	t	n1	n2	mf
53.0000	38.0000	0.9500	10.0000	20.0000	0.9000

By Cluster

Cluster	w	u	t	n1	n2	mf
1	9.0000	6.0000	0.7500	2.0000	4.0000	0.5000
2	11.0000	8.0000	1.0000	2.0000	4.0000	1.0000
3	11.0000	8.0000	1.0000	2.0000	4.0000	1.0000
4	11.0000	8.0000	1.0000	2.0000	4.0000	1.0000
5	11.0000	8.0000	1.0000	2.0000	4.0000	1.0000

No excluded clusters

Interpretation

There were five clusters each with two and four dogs in the control and active vaccine groups, respectively. The estimated mitigated fraction was 0.5 in clusters 1 and 1 for clusters 2 to 5. Since the estimated mitigated fractions in all the clusters are greater than 0, the vaccine was effective in reducing disease severity. The overall stratified mitigated fraction was 0.9. None of the clusters were excluded in the computing the overall mitigated fraction.

4.4 Bootstrap confidence intervals for stratified Mitigated Fraction

Evaluation

```
%MFCLUSBOOT(IN=DF1,Y=TOT_DAYS,GRP=TRT,CON='T01',VAC='T02',CLUSTER=BLK,
NBOOT=10000,RETURNBOOT=0,HPD=1,BOOTCLUSTER=1,BOOTUNIT=1,ALPHA=0.05,
CLUSTERNUM=1, SEED=123);
```

SAS output

Bootstrap Confidence Intervals

	Observed	Lower	Median	Upper
Percentile	0.9000	0.6000	0.9000	1.0000
HPD	0.9000	0.7000	0.9000	1.0000

Number of Bootstraps

10000

Alpha

0.05

Interpretation

The estimated (observed) stratified mitigated fraction was 0.9. The 95% equal tailed and HPD were [0.60-1.00] and [0.70-1.00], respectively. Since the two types of confidence intervals do not coincide and there is an indication of skewness in the parameter distribution. The number of bootstrap samples taken was ten thousand.

Chapter 5

Comparison with R functions and results

The table below shows the existing R functions equivalent to the SAS macros and from which the SAS macros were translated.

Table 2: Equivalent functions in R and SAS.

R function	Comparison	SAS Macro
MFr	The parameters required in both functions and the output is similar.	%MFR
MFBoot	All other parameters apart from those needed to perform the bootstrap are similar. In R, b bootstraps are performed in B cycles such that the total number of samples is b*B. Therefore, two parameters B and b are required in R while in SAS only one parameter is needed; nboot. The point estimates are similar; but the confidence interval could be different due to sampling.	%MFBOOT
MFClus	The parameters required in both functions and the output is similar.	%MFCLUS
MFClusBoot	Just like in MFBoot, all other parameters apart from the bootstrapping parameters are similar. The output is similar and differences occur due to sampling. In R, the output of MFClus is displayed as well whereas in SAS it is omitted.	%MFCLUSBOOT
MFmp	The parameters required in both functions and the output is similar.	%MFMP
MFSubj	The parameters required in both functions and the output is similar.	%MFSUBJ
HLBoot	Just like in MFBoot, all other parameters apart from the bootstrapping parameters are similar. The output is similar and differences occur due to	%HLBOOT

	sampling.	

Generally, the parameters passed on to the functions and the point estimates from R and SAS are similar. The bootstrap confidence intervals might differ, but this is expected because the quantiles are computed based on different samples as a result of bootstrapping.

Chapter 6

Discussion and Conclusion

In this project, SAS algorithms based on an existing R package are implemented for estimating mitigated fraction for independent and clustered data. The algorithms have been applied using data from a real life canine vaccine trial and output compared with that obtained in R.

The mitigated fraction is easily calculated from the Wilcoxon statistic. While the procedures for point estimation have been laid out, procedures for hypothesis testing are yet to be developed. Use of non-parametric test allows one to avoid certain parametric assumptions; however, this does not eliminate the need for forethought in study design. Therefore, careful thought should thus be taken to ensure the validity of the disease severity (Siev, 2005).

The definitions and macros on mitigated fraction presented in this project were focused on expressing the vaccine effect as a measure based on probability rather than a measure formed from an average as would be case by used of linear mixed model. Just like linear mixed model allows for vaccine effect estimation in presence of covariates, the stratified mitigated fraction attempts to account for covariates. However, only one covariate was allowed; blocking. Thus, more research should be done on extending the existing methodologies to allow for inclusion of several covariates.

The SAS algorithms use the macro language to pass parameters and are written using the SAS/IML. SAS/IML is preferred over the DATA step because the DATA step uses records. However, the computations involved in estimation of mitigated fraction are easier to deal with in matrix format available in SAS/IML environment. Moreover, SAS/IML would be more efficient than use of arrays within a DATA step.

There is a limitation in using the macros. SAS IML is not part of base SAS and therefore only users with the appropriate SAS installation will be able to use the macros. To be more effective, only basic functions of SAS/IML language are used such that the user does not need access to the full library of the SAS/IML to perform the analysis.

The macros present a way to compute the various MF estimators, thus overcoming the problem where such estimates are not easily computed due to non-availability of implemented procedures in SAS. The macros can be extended further to estimate PF and MF together as components of a nested model Siev (2005). Furthermore, the macros could serve as starting point for further development and integration into the SAS system.

From the analysis of the canine vaccine trial; we concluded that the vaccine was effective since the relative increase in the probability that a vaccinated dog had a less severe disease than a non-vaccinated dog was 0.9.

References

1. Bross, I. D. J. (1958). How to use ridit analysis. *Biometrics*, **14**: 18-38.
2. Carpenter, J and Bithell, J. (2000). Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Statistics in Medicine*, **19**: 1141-1164.
3. Greenwood. M. and Yule, U. G. (1915). The statistics of anti-typhoid and anti-cholera inoculations, and the interpretation of such statistics in general. *Proceedings of the Royal Society of Medicine*, **8**(2): 113-94.
4. Halloran, M. E., Struchiner C. J., and Longini I. M. (2009). *Design and Analysis of Vaccine Studies*. Springer
5. Halloran, M. E., Struchiner, C. J., and Longini, I. M. (1997). Study Designs for Evaluating Different Efficacy and Effectiveness Aspects of Vaccines. *American Journal of Epidemiology*, **146**(10): 789-803.
6. Hodges, J. L. and Lehmann E. L. (1963). Estimates of location based on rank tests. *The Annals of Mathematical Statistics*, **34**: 598-611.
7. Hudgens, M. G. and Halloran M. E. (2004). Causal Vaccine on Binary Post-infection Outcomes. *Technical Report 04-03*. Department of Biostatistics, Rollins School of Public Health, Emory University, Atlanta, GA.
8. R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.
9. SAS Institute (2011). The SAS system for Windows. Release 9.2. SAS Institute Inc., Cary, NC, USA.
10. Siev, D. (2005). An Estimator of intervention effect on disease severity. Journal of *Modern Applied Statistical Methods*, **4**(2): 500-508.

11. Siev, D. (2012). MF: Mitigated Fraction. R package version 4.1.
12. Efron B. and Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman & Hall: New York.
13. Helsel, D. R. and Hirsch, R. M. (1992). *Statistical Methods in Water Resources*. Elsevier, New York.
14. Kendrick, P. L. and Eldering, G. A. (1939). Study in active immunization against pertussis. *American Journal of Hygiene*, **29**:133–53.
15. Wilcoxon, F. (1945). Individual Comparison by Ranking Methods. *Biometrics*, **1**:80-83.
16. Wolfe, D. A. and Hogg, R. V. (1971). On constructing statistics and reporting data. *American Statistician*, **25**(4), 27-30

Appendix

A.1 MFr

```
%macro mfr(in=,y=,grp=, con=, vac=,);
  proc iml;
    use &in; /*READ SAS DATA TO IML MATRIX*/
    read all var{&y} into CONTY where (&grp = &con );
    read all var{&y} into VACY where (&grp = &vac );
    RESP = CONTY//VACY;
    NCONT = nrow(CONTY);
    NVAC = nrow(VACY);
    NTOT = nrow(RESP);
    RESP = ranktie(RESP); /*RANK AND BREAK TIES BY TAKING AVG*/
    W = sum(RESP[1:NCONT]);
    STAT = (2*W - NCONT*(1 + NCONT + NVAC))/(NCONT*NVAC);
    print, 'Estimated Mitigated Fraction', STAT[format=.4 label='MF'];
  quit;
%mend mfr;
```

A.2 MFBoot

```
%macro MFBOOT(in=, y=, grp=, con=, vac=, alpha=, hpd=, bca=,returnboot=,
nboot=,seed=,);
proc iml;
  call randseed(&seed);
      use &in; /*READ SAS DATA TO IML MATRIX*/
      read all var{&y} into CONTY where (&grp = &con);
      read all var{&y} into VACY where (&grp = &vac);
      RESP = CONTY//VACY;
      NCONT = nrow(CONTY);
      NVAC = nrow(VACY);
      NRESP = NCONT + NVAC;
      /*BOOTSTRAP*/
      start boot(x);
            lenx = nrow(x);
            rep=1;
            do while (rep <= lenx);
                        probs = J(lenx, 1, 1);
                        call randgen(probs, "Uniform");
                        which = max(probs);
                        index = 1;
                        found = 0;
                        do until (found = 1);
                              if probs[index] = which then found = 1;
                              else index = index + 1;
                        end;
                        if rep=1 then sample=x[index];
                        else sample = sample//x[index];
                        rep = rep + 1;
                  end;
            return(sample);
      finish;
      /*================== QUANTILE FUNCTION ============*/
      start which(x, y);
            n = nrow(x);
            indices = {.};
```

```
      do i=1 to n;
            if x[i] > y[i] then
                  indices = indices//i;
      end;
      nindex = nrow(indices);
      indices = indices[2:nindex];
      return(indices);
finish;
start quantile(x, probs);
      n = nrow(x);
      np = nrow(probs);
      index = 1 + (n - 1)*probs;
      lo = floor(index);
      hi = ceil(index);
      call sort(x);
      qs = x[lo];
      i = which(index, lo);
      vec = index - lo;
      h = vec[i];
      qs[i] = (1 - h)#qs[i] + h#x[hi[i]];
      return(qs);
finish;
/*================== The HPD ===========*/
start emp_hpd(X);
      LENX = nrow(X);
      LOWER =  do(0.001, 0.05, 0.001);
      UPPER = do(0.95, 0.999, 0.001);
      UPPER_INT = quantile(X, UPPER);
      LOWER_INT = quantile(X, LOWER);
      CI = LOWER_INT||UPPER_INT;
      INTERVALS = UPPER_INT - LOWER_INT;
      CI = CI||INTERVALS;
      found = 0;
      index = 1;
      do while (found = 0);
            if INTERVALS[index] = min(INTERVALS) then found=1;
            else index = index + 1;
      end;
      HPD = CI[index,1:2];
      return(HPD);
finish;
/*RANKSUM STATISTIC*/
start wfn(XY, NX);
      STAT = ranktie(XY);
      STAT = sum(STAT[1:NX]);
      return(STAT);
finish;
WB = {.};
NUMBOOT = &nboot;
do replicate=1 to NUMBOOT; /*THE BOOTSTRAP*/
      BOOTXY = boot(CONTY)//boot(VACY);
      STATB= wfn(BOOTXY, NCONT); /*COMPUTE THE STATS ROW-WISE*/
      WB = WB//STATB;
end;
WB = WB[2:(NUMBOOT+1)];
LENWB = nrow(WB);
MFBOOT = ((2#WB - NCONT#(1 + NRESP))/(NCONT#NVaC));
MFOBS = ((2#wfn(RESP, NCONT) - NCONT#(1 + NRESP))/(NCONT#NVaC));
QPROB = (&alpha/2)||0.5||(1-&alpha/2);
QMF = quantile(MFBOOT, QPROB);
STAT = MFOBS||t(QMF);
```

```
      mattrib STAT colname={'Observed', 'Lower', 'Median', 'Upper'}
      rowname='Percentile';
      if &hpd then do;
            HPDMF = emp_hpd(MFBOOT);
            PROB = 0.5;
            MED = quantile(MFBOOT, PROB);
            HPDCI = MFOBS||HPDMF[1]||MED||HPDMF[2];
            STAT = STAT//HPDCI;
            mattrib STAT rowname={ 'Percentile' 'HPD'};
      end;
      if &bca then do;
            Z0 = probit(sum(MFBOOT < MFOBS)/(&nboot));
            NXI = J(NCONT, 1,NCONT-1)//J(NVAC, 1, NCONT);
            NYI = J(NCONT, 1, NVAC)//J(NVAC, 1, NVAC-1);
            THETA = {.};
            do k=1 to NRESP;
                  if k=1 then LEAVEOUT = RESP[2:NRESP];
                  if k>1 & k<NRESP then LEAVEOUT = RESP[1:(k-
                  1)]//RESP[(k+1):NRESP];
                  if k=NRESP then LEAVEOUT = RESP[1:(NRESP-1)];
                        THETAI = ((2#wfn(LEAVEOUT, NXI[k])#(1 + NXI[k] +
                        NYI[k]))/(NXI[k]#NYI[k]));
                        THETA = THETA//THETAI;
                  end;
            THETA = THETA[2:NRESP];
            THETA_HAT = mean(THETA);
            ACC = sum((THETA_HAT - THETA)##3)/(6#sum((THETA_HAT -
            THETA)##2)##(3/2));
            Z1 = probit(&alpha/2);
            Z2 = probit(1 - &alpha/2);
            A1 = probnorm(Z0 + (Z0 + Z1)/(1 - ACC * (Z0 + Z1)));
            A2 = probnorm(Z0 + (Z0 + Z2)/(1 - ACC#(Z0 + Z2)));
            A5 = probnorm(Z0 + Z0/(1 - ACC#Z0));
            QPROB = A1//A5//A2;
            STUFF = ACC//Z0//A1//A2;
            mattrib STUFF colname={'ACC', 'Z0', 'A1', 'A2'};
            QMF = quantile(MFBOOT, QPROB);
            STAT = STAT//(MFOBS||t(QMF));
            mattrib STAT rowname={ 'Perc.' 'BC.a'};
      end;
      if &hpd & &bca then mattrib STAT rowname={ 'Perc.' 'HPD' 'BC.a'};
      print STAT[format=.4 label='Bootstrap Confidence Intervals'];
      print NUMBOOT[label='Number of Bootstraps'];
      print &alpha[label='alpha'];
      if &returnboot then do;
            create boot from MFBOOT;
            append from MFBOOT;
            close boot;
      end;
quit;
%MEND mfboot;
```

A.3 MFClus

```
%macro mfclus(in=,y=,grp=,con=,vac=,cluster=,clusternum=,);
  proc iml;
   use &in;
   read all var{&cluster} into cluster;
   STRATA = unique(cluster);
   NSTRATA = ncol(STRATA);
```

```
colnames = {'w', 'u', 'r', 'n1', 'n2', 'mf'};
   do STRATUM = 1 to NSTRATA;
         SELECT = STRATA[STRATUM];
         read all var{&y} into CONTY where (&grp = &con & &cluster =
         SELECT); /*READ SAS DATA TO IML MATRIX*/
         read all var{&y} into VACY where (&grp =  &vac & &cluster =
         SELECT);
         NCONT = nrow(CONTY);
         NVAC = nrow(VACY);
         if (NCONT > 0 & NVAC > 0) then do;
               RESP = CONTY//VACY;
               NTOT = nrow(RESP);
               RESP = ranktie(RESP); /*BREAK TIES BY AVERAGE*/
               W = sum(RESP[1:NCONT]);
               U = W - (NCONT*(NCONT + 1))/2;
               R = U/(NCONT*NVAC);
               MF = 2*R -1;
               X = t(W//U//R//NCONT//NVAC//MF);
               if STRATUM = 1 then do;
               OUTNONMISS=X;
               STRAT = SELECT;
               end;
               if STRATUM > 1 then do;
               OUTNONMISS = OUTNONMISS//X;
                     STRAT = STRAT//SELECT;
               end;
         end;
         if (NCONT = 0 | NVAC = 0) then do;
               X = t(.//.//.//NCONT//NVAC//.);
               if STRATUM = 1 then do;
               EXCLUDEDCLUSTERS = SELECT;
                     OUTMISS = X;
               end;
               if STRATUM > 1 then do;
               EXCLUDEDCLUSTERS = EXCLUDEDCLUSTERS//SELECT;
                     OUTMISS = OUTMISS//X;
               end;
         end;
   end;
   OUT = OUTNONMISS//OUTMISS;
   ALL = OUT[+,];
   DENOMR = sum(t(OUT[,4])#t(OUT[,5]));
   R = ALL[2]/DENOMR;
   MF = 2*R - 1;
   ALL[3] = R;
   ALL[6] = MF;
   create mfclus from MF;  /*SAVE THE MF*/
   append from MF;
   close mfclus;
   BYCLUSTERMF = OUT[,6];
   create byclustermf from byclustermf;    /*SAVE THE CLUSTERS MF'S*/
   append from byclustermf;
   close byclustermf;
   if &clusternum then     STRAT = char(STRAT);
   print, 'Estimated Mitigation Fraction',ALL[label='Overall'
   colname=COLNAMES format=.4];
   STRATNAME = STRAT//EXCLUDEDCLUSTERS;
   print, , OUT[label='By Cluster' format=.4 colname=colnames
   rowname=STRATNAME];
   NSTRAT = nrow(STRAT);
   if NSTRAT^=NSTRATA then do;
```

```
            EXCLUDEDCLUSTERS = t(EXCLUDEDCLUSTERS);
            print, , EXCLUDEDCLUSTERS[label='Excluded Clusters'];
      end;
      if NSTRAT=NSTRATA then do;
            print, 'No Clusters excluded';
      end;
 QUIT;
%MEND;
```

A.4 MFClusBoot

```
%macro mfclusboot(in=, y=, grp=, con=, vac=, cluster=, nboot=, returnboot=,
hpd=, bootcluster=,bootunit=,alpha=,clusternum=, seed=,);

proc iml;
  call randseed(&seed);
      /*================ INDEXING  ======*/
      start position(X, Y);
            INDEX = 1;
            COUNTER = 1;
            FOUND=0;
            do until(FOUND=1);
                  if Y[COUNTER] = X then do;
                        FOUND=1;
                        INDEX = COUNTER;
                  end;
                  COUNTER = COUNTER + 1;
            end;
            return(INDEX);
      finish;
      /*=======================  QUANTILE FUNCTION ========*/
      start which(x, y);
            n = nrow(x);
            indices = {.};
            do i=1 to n;
            if x[i] > y[i] then
                  indices = indices//i;
            end;
            nindex = nrow(indices);
            indices = indices[2:nindex];
            return(indices);
      finish;

      start quantile(x, probs);
            n = nrow(x);
            np = nrow(probs);
            index = 1 + (n - 1)*probs;
            lo = floor(index);
            hi = ceil(index);
            call sort(x);
            qs = x[lo];
            i = which(index, lo);
            vec = index - lo;
            h = vec[i];
            qs[i] = (1 - h)#qs[i] + h#x[hi[i]];
            return(qs);
      finish;
      /*=============== THE HPD  ============*/
      start emp_hpd(X);
```

```
      LENX = nrow(X);
      LOWER =  do(0.001, 0.05, 0.001);
      UPPER = do(0.95, 0.999, 0.001);
      UPPER_INT = quantile(X, UPPER);
      LOWER_INT = quantile(X, LOWER);
      CI = LOWER_INT||UPPER_INT;
      INTERVALS = UPPER_INT - LOWER_INT;
      CI = CI||INTERVALS;
      found = 0;
      index = 1;
      do while (found = 0);
            if INTERVALS[index] = min(INTERVALS) then found=1;
            else index = index + 1;
      end;
      HPD = CI[index,1:2];
      return(HPD);
finish;
/*=============== THE BOOTSTRAP  ==================*/
start boot(x);
      lenx = nrow(x);
      rep=1;
      do while (rep <= lenx);
                  probs = J(lenx, 1, 1);
                  call randgen(probs, "Uniform");
                  which = max(probs);
                  index = 1;
                  found = 0;
                  do until (found = 1);
                        if probs[index] = which then found = 1;
                        else index = index + 1;
                  end;
                  if rep=1 then
                  sample=x[index];
                  else
                        sample = sample//x[index];
                  rep = rep + 1;
            end;
      return(sample);
finish;
/*=============== WBOOT  ==============*/
start wboot(X, Y, NB);
      NX = nrow(X);
      NY = nrow(Y);
      if NX=1 then XB=J(NB, 1, X);
            if NX > 1 then do;
                  lenx = nrow(X);
                  XB = J(NB, lenx, .);
                  do rep=1 to NB; /*THE BOOTSTRAP*/
                    XB[rep,]  = t(boot(X));
                  end;
end;
            if NY=1 then YB=J(NB, 1, Y);
            if NY > 1 then do;
                  LENY = nrow(Y);
                  YB = J(NB, LENY, .);
                  do rep=1 to NB; /*THE BOOTSTRAP*/
                  YB[rep,]= t(boot(Y));
                  end;
            end;
            MAT = XB||YB;
            do r=1 to nb;
```

```
                  ROWSEL = MAT[r,];
                  ROWSEL = ranktie(ROWSEL);
                  STAT = sum(ROWSEL[1:nx]);
            if r=1
            then WSTAT = STAT;
            else
               WSTAT = WSTAT//STAT;
            end;
      return(WSTAT);
finish;
use &in;
read all var{&cluster} into cluster;
STRATA = unique(cluster);
NSTRATA = ncol(STRATA);
      do STRATUM = 1 to NSTRATA;
      SELECT = STRATA[STRATUM];
      read all var{&y} into CONTY where (&grp = &con & &cluster =
      SELECT); /*READ SAS DATA TO IML MATRIX*/
      read all var{&y} into VACY where (&grp =  &vac & &cluster =
      SELECT);
      NCONT = nrow(CONTY);
      NVAC = nrow(VACY);
      if (NCONT > 0 & NVAC > 0) then do;
            RESP = CONTY//VACY;
            NTOT = nrow(RESP);
            RESP = ranktie(RESP); /*BREAK TIES BY AVERAGE*/
            W = sum(RESP[1:NCONT]);
            U = W - (NCONT*(NCONT + 1))/2;
            R = U/(NCONT*NVAC);
            MF = 2*R -1;
            X = t(W//U//R//NCONT//NVAC//MF);
            if STRATUM = 1 then do;
            OUTNONMISS=X;
            STRAT = SELECT;
            end;
            if STRATUM > 1 then do;
            OUTNONMISS = OUTNONMISS//X;
                  STRAT = STRAT//SELECT;
            end;
      end;
      if (NCONT = 0 | NVAC = 0) then do;
            X = t(.//.//.//NCONT//NVAC//.);
            if STRATUM = 1 then do;
            EXCLUDEDCLUSTERS = SELECT;
                  OUTMISS = X;
            end;
            if STRATUM > 1 then do;
            EXCLUDEDCLUSTERS = EXCLUDEDCLUSTERS//SELECT;
                  OUTMISS = OUTMISS//X;
            end;
      end;
end;
OUT = OUTNONMISS//OUTMISS;
ALL = OUT[+,];
DENOMR = sum(t(OUT[,4])#t(OUT[,5]));
R = ALL[2]/DENOMR;
MFOBS = 2*R - 1;
NSTRAT = nrow(STRAT);


if (&bootcluster=1) then do;
```

```
            NBOOT = &nboot;
            if &clusternum then
                  STRATB = J(NBOOT, NSTRAT, .);
            else  STRATB = J(NBOOT, NSTRAT, ".");
            do rep=1 to NBOOT;
                  row = t(boot(STRAT));
                  STRATB[rep,] = row;
            end;
 end;
if (&bootcluster = 0) then
      STRATB = shape(STRAT,&nboot, NSTRAT);
if (&bootunit = 0) then do;
      do STRATUM=1 to NSTRAT;
            SELECT = STRAT[STRATUM];
            use &in;
            read all var{&y} into CONTY where (&grp = &con  &
            &cluster = SELECT); /*READ SAS DATA TO IML MATRIX*/
            read all var{&y} into VACY where (&grp = &vac & &cluster
            = SELECT);
            RESP = CONTY//VACY;
            NCONT = nrow(CONTY);
            NVAC = nrow(VACY);
            NRESP = NCONT + NVAC;
            RESP = ranktie(RESP);
            STATW = sum(RESP[1:NCONT]);
            if STRATUM = 1 then do;
             W = STATW;
             STATU = STATW - (NCONT#(NCONT + 1))/2;
             U = STATU;
             N1N2 = (NCONT#NVAC);
            end;
            if STRATUM > 1 then do;
             W = W//STATW;
             STATU = STATW - (NCONT#(NCONT + 1))/2;
             U = U//STATU;
             N1N2 = N1N2//(NCONT#NVAC);
            end;
      end;
      MATW = J(&nboot, NSTRAT, .);
      MATU = MATW;
      MATN1N2 = MATW;
      do j=1 to (&nboot);
            do k = 1 to NSTRAT;
                  SEL = STRATB[j,k];
                  INDEX = position(SEL, STRAT);
                  MATW[j,k] = W[INDEX];
                  MATU[j,k] = U[INDEX];
                  MATN1N2[j,k] = N1N2[INDEX];
            end;
      end;
      VECW = MATW[,+];
      VECU = MATU[,+];
      VECN1N2 = MATN1N2[,+];
      R = VECU/VECN1N2;
      MF = 2*R - 1;
end;
if (&bootunit=1) then do;
      MATW = J(&nboot, NSTRAT, .);
      MATU = MATW;
      MATN1N2 = MATW;
      do STRATUM=1 to NSTRAT;
```

```
                  SELECT = STRAT[STRATUM];
                  use &in;
                  read all var{&y} into CONTY where (&grp = &con  &
                  &cluster = SELECT); /*READ SAS DATA TO IML MATRIX*/
                  read all var{&y} into VACY where (&grp = &vac & &cluster
                  = SELECT);
                  RESP = CONTY//VACY;
                  NCONT = nrow(CONTY);
                  NVAC = nrow(VACY);
                  NEACHI = sum(STRATB=SELECT);
                  if STRATUM=1 then
                  NEACH=NEACHI;
                  else
                        NEACH = NEACH//NEACHI;
                  filler = wboot(CONTY, VACY, NEACHI);
                  PICK=1;
                  do j=1 to (&nboot);
                        do k=1 to NSTRAT;
                              SEL = STRATB[j,k];
                              if SEL=SELECT then do;
                                    MATW[j,k] = filler[PICK];
                                    MATU[j,k] = MATW[j,k] - (NCONT#(NCONT +
                                    1))/2;
                                    MATN1N2[j,k] = NCONT#NVAC;
                                    PICK = PICK + 1;
                              end;
                        end;
                  end;
            end;
            VECW = MATW[,+];
            VECU = MATU[,+];
            VECN1N2 = MATN1N2[,+];
            R = VECU/VECN1N2;
            MF = 2#R -1;
      end;
      Q = 0.5||(&alpha/2)||(1-&alpha/2);
       /*OBSERVED STATS FROM MFCLUS*/
      MFQ = quantile(MF, Q);
      STAT = MFOBS||t(MFQ);
      mattrib STAT ;
      if &hpd then do;
            HPDMF = emp_hpd(MF);
            STAT = STAT//(t(STAT[1:2])||HPDMF);
      end;
      if &hpd then
      mattrib STAT rowname={'Percentile', 'HPD'};
      else mattrib STAT rowname='Percentile';
      if &returnboot then do;
            create bootsample from MF;
            append from MF;
            close bootsample;
      end;
      NBOOT = &nboot;
      print, STAT[label='Cluster Bootstrap Confidence Intervals' format=.4
      colname={'Obs' 'Med' 'Lower' 'Upper'}];
      print , NBOOT[label='Bootstrap Samples'];
      print, &alpha[label='Alpha'];
quit;
%mend mfclusboot;
```

A.5 MFmp

```
%macro mfmp(data=,in=, y=, grp=,con=,vac=,cluster=,
clusternum=,alpha=,df=,tdist=,);
      %macro execute(data=,);
            %if &data %then
            %do;
                  %include fileloc(MFClus);
                  %mfclus(in=&in, y=&y,
                  grp=&grp,con=&con,vac=&vac,cluster=&cluster,
                  clusternum=&clusternum);
            %end;
      %mend execute;
      %execute(data=&data);
      proc iml;
            if &data then do;
                  use byclustermf;
                  read all var{col1} into MF;
                  close byclustermf;
                  LABELS = {1 0 -1};
                  X = J(1, 3, 0);
                  lenmf = nrow(MF);
                  do i=1 to lenmf;
                        if MF[i] = 1 then
                              X[1] = X[1] + 1;
                  if MF[i] = 0 then
                              X[2] = X[2] + 1;
                  if MF[i] = -1 then
                              X[3] = X[3] + 1;
                  end;
            end;
            if &data = 0 then do;
                  use &in;
                  read all var{col1} into X;
                  close &in;
                  X = t(X);
            end;
            N = sum(X);
            P = X/N;
            V = (diag(P) - t(P)*P)/N;
            A = {1 0 -1}; grad=A;
            B = A*t(P);
            VB = A*V*t(A);
            GRADL = (1/(P[1]-P[3])||0||1/(P[1]-P[3]));
            LOGB = log(B);
            VLOGB = GRADL*V*t(GRADL);
            if &tdist & &df=0 then do;
                  DF = N-2;
                  Q = tinv(0.5||&alpha/2||1-&alpha/2, DF);
            end;
            if (&df>0) then do;
                  DF = &df;
                  Q = tinv(0.5||&alpha/2||1-&alpha/2, &df);
            end;
            if (&tdist=0) & (&df=0) then
                  Q = probit(0.5||&alpha/2||1-&alpha/2);
            CI = B + Q#sqrt(VB);
            print, CI[format=.4 label='MF Matched Pairs' colname={'Point',
            'Lower', 'Upper'}];
            print, &alpha[label='Alpha'];
```

```
            if &tdist then
                  print, DF[label='Degrees of Freedom'];
      quit;
%mend mfmp;
```

A.6 MFSubj

```
%macro mfSUBJ(in=, y=, grp=,con=,vac=);
proc iml;
      use calflung; /*READ SAS DATA TO IML MATRIX*/
      read all var{resp} into CONTY where (group = 'con');
      read all var{resp} into VACY where (group = 'vac');
      RESP = CONTY//VACY;
      NCONT = nrow(CONTY);
      NVAC = nrow(VACY);
      ntot=nrow(RESP);
      RESP = ranktie(RESP);
      NRESP = NCONT + NVAC;
      W = sum(RESP[1:NCONT]);
      MF = (2*W - NCONT*(1 + NCONT + NVAC))/(NCONT*NVAC);
      U = W - (NCONT*(NCONT + 1))/2;
      UJ = J(NVAC,1,.);
      do J=1 to NVAC;
            P1 = sum(VACY[J]<CONTY);
            P2 = P1 + sum(VACY[J]=CONTY);
            STATUJ = mean(p1//p2);
            UJ[J] = STATUJ;
      end;
      R = U/(NCONT#NVAC);
      RJ = UJ/NCONT;
      MFJ = 2#RJ - 1;
      RANK = RESP[(NCONT+1):nRESP];
      SUBJ = VACY||rank||UJ||RJ||MFJ;
      MFUNIQ = unique(MFJ);
      NLEN = ncol(MFUNIQ);
      FREQ = J(NLEN, 1, .);
      MAXY = J(NLEN, 1, .);
      MINY = J(NLEN, 1, .); /*AN EMPTY VECTOR*/
      do I=1 to NLEN;
            SELECT = MFUNIQ[I];
            HOLDFREQ = 0;
            do k=1 to NVAC;
                  if SUBJ[k,5]=SELECT then do;
                        if k= 1 then
                              HOLDY = SUBJ[k,1];
                        else HOLDY = HOLDY//SUBJ[k,1];
                        HOLDFREQ = HOLDFREQ + 1;
                  end;
            end;
            HOLDMAXY = max(HOLDY);
            HOLDMINY = min(HOLDY);
            MAXY[I] = HOLDMAXY;
            MINY[I] = HOLDMINY;
            FREQ[I] = HOLDFREQ;
      end;
      SUBJ=t(MFUNIQ)||FREQ||MINY||MAXY;
      print, MF[format=.4];
      print, SUBJ[label='MF Subject Components'  format=.4 colname={'MF.j'
'Freq' 'Min.y' 'Max.y'}];
quit;
```

```
%mend mfsubj;
```

A.7 HLBoot

```
%macro hlboot(in=, y=, grp=, con=,
vac=,nboot=,alpha=,hpd=,bca=,returnboot=, seed=,);
  proc iml;
  call randseed(&seed);
    use &in; /*READ SAS DATA TO IML MATRIX*/
    read all var{&y} into CONTY where (&grp = &con);
      read all var{&y} into VACY where (&grp = &vac);
      RESP = CONTY//VACY;
      NCONT = nrow(CONTY);
      NVAC = nrow(VACY);
      NRESP = NCONT + NVAC;
            /*=============== THE BOOTSTRAP  =*/
      start boot(x);
            lenx = nrow(x);
            rep=1;
            do while (rep <= lenx);
                        probs = J(lenx, 1, 1);
                        call randgen(probs, "Uniform");
                        which = max(probs);
                        index = 1;
                        found = 0;
                        do until (found = 1);
                              if probs[index] = which then found = 1;
                              else index = index + 1;
                        end;
                        if rep=1 then
          sample=x[index];
                        else sample = sample//x[index];
                        rep = rep + 1;
                  end;
            return(sample);
      finish;
      /*================================= QUANTILE FUNCTION =*/
      start which(x, y);
            n = nrow(x);
            indices = {.};
            do i=1 to n;
                  if x[i] > y[i] then
                        indices = indices//i;
            end;
            nindex = nrow(indices);
            indices = indices[2:nindex];
            return(indices);
      finish;
      start quantile(x, probs);
            n = nrow(x);
            np = nrow(probs);
            index = 1 + (n - 1)*probs;
            lo = floor(index);
            hi = ceil(index);
            call sort(x);
            qs = x[lo];
            i = which(index, lo);
            vec = index - lo;
            h = vec[i];
            qs[i] = (1 - h)#qs[i] + h#x[hi[i]];
```

```
            return(qs);
      finish;
/*=================================The HPD======*/
      start emp_hpd(X);
            LENX = nrow(X);
            LOWER =  do(0.001, 0.05, 0.001);
            UPPER = do(0.95, 0.999, 0.001);
            UPPER_INT = quantile(X, UPPER);
            LOWER_INT = quantile(X, LOWER);
            CI = LOWER_INT||UPPER_INT;
            INTERVALS = UPPER_INT - LOWER_INT;
            CI = CI||INTERVALS;
            found = 0;
            index = 1;
            do while (found = 0);
                  if INTERVALS[index] = min(INTERVALS) then found=1;
                  else index = index + 1;
            end;
            HPD = CI[index,1:2];
            return(HPD);
      finish;
      /*THE RANK SUM STATISTIC*/
      start wfn(XY, NX);
            STAT = ranktie(XY);
            STAT = sum(STAT[1:NX]);
            return(STAT);
      finish;
      /*HODGES-LEHMAN ESTIMATOR*/
      start HLfn(XY, NX);
            X = XY[1:NX];
            Y = XY[(NX+1):nrow(XY)];
            NY = nrow(XY) - NX;
            MATX = t(shape(X, NY, NX));
            MATY = shape(Y, NX, NY);
            DIF = shape(MATY-MATX, NX#NY, 1);
            prob = 0.5;
            MEDDIF = quantile(DIF, prob);
            return(MEDDIF);
      finish;
      /*OBSERVED STATS*/
      MF = ((2#wfn(RESP, NCONT) - NCONT#(1 + NRESP))/(NCONT#NVAC));
      HL = HLfn(RESP, NCONT);
      PROB = {0.25, 0.5, 0.75};
      QX = quantile(CONTY, PROB);
      QY = quantile(VACY, PROB);
      QDIF = QY - QX;
      W = J(&nboot, 1, .);
      H = J(&nboot, 1, .);
      QMATX = J(&nboot, 3, .);
      QMATY=J(&nboot, 3, .);
      XBMAT = J(&nboot, NCONT, .);
      YBMAT = J(&nboot, NVAC, .);

      do rep=1 to &nboot;   /*Bootstrapping*/
      XBMAT[rep,] = t(boot(CONTY));
      YBMAT[rep,] = t(boot(VACY));
/*DO COMPUTATIONS ROW-WISE*/
                  ROWXSEL = t(XBMAT[rep,]);
                  ROWYSEL = t(YBMAT[rep,]);
                  QXR = quantile(ROWXSEL, PROB);
                  QYR = quantile(ROWYSEL, PROB);
```

```
            WR = wfn(XBMAT[rep,]||YBMAT[rep,], NCONT);
            HLR = HLfn(t(XBMAT[rep,]||YBMAT[rep,]), NCONT);
            QMATX[rep,] = t(QXR);
            QMATY[rep,] = t(QYR);
            W[rep] = WR;
            H[rep] = HLR;
end;
QMATDIF = QMATY - QMATX;
MFVEC = ((2#w - NCONT#(1 + NRESP))/(NCONT#NVAC));
QPROB = 0.5||&alpha/2||1-&alpha/2;
QMF = quantile(MFVEC, PROB);
MFSTAT = shape(MF//QMF, 1, 4);
QHL = quantile(H, QPROB);
HLSTAT = shape(HL//QHL, 1, 4);
QQD =  J(3,3,.);
QQX = J(3,3,.);
QQY = J(3,3,.);
do c = 1 to ncol(QMATDIF); /*DO COMPUTATIONS COLUMN-WISE*/
      SELECT = QMATDIF[,c];
      QQDR = quantile(SELECT, QPROB);
      QQD[,c] = QQDr;
      SELECT = QMATX[,c];
      QQXR = quantile(SELECT, QPROB);
      QQX[,c] = QQXR;
      SELECT = QMATY[,c];
      QQYR = quantile(SELECT, QPROB);
      QQY[,c] = QQYR;
end;
QQDIFSTAT = QDIF||t(QQD);
QXSTAT = QX||t(QQX);
QySTAT = QY||t(QQy);
if &hpd then do;
      HPDMF = emp_hpd(MFVEC);
      PROB = 0.5;
      MEDMF = quantile(MFVEC, PROB);
      MFSTAT = MFSTAT//(MF||MEDMF||HPDMF);
      HPDHL = emp_hpd(H);
      MEDH = quantile(H, PROB);
      HLSTAT = HLSTAT//(HL||MEDH||HPDHL);
end;
if &bca then do;
      Z0 = probit(sum(MFVEC < MF)/(&nboot));
      NXI = j(NCONT, 1,NCONT-1)//j(NVAC, 1, NCONT);
      NYI = j(NCONT, 1, NVAC)//j(NVAC, 1, NVAC-1);
      THETA = J(NRESP,1,.);
      do k=1 to NRESP;
            if k=1 then LEAVEOUT = RESP[2:NRESP];
            if k>1 & k<NRESP then LEAVEOUT = RESP[1:(k-
            1)]//RESP[(k+1):NRESP];
            if k=NRESP then LEAVEOUT = RESP[1:(NRESP-1)];
            THETAI = ((2#wfn(LEAVEOUT, NXI[k])#(1 + NXI[k] +
            NYI[k]))/(NXI[k]#NYI[k]));
            THETA[K] = THETAI;
      end;
      THETA_HAT = mean(THETA);
      ACC = sum((THETA_HAT - THETA)##3)/(6#sum((THETA_hat -
      THETA)##2)##(3/2));
      Z1 = probit(&alpha/2);
      Z2 = probit(1 - &alpha/2);
      A1 = probnorm(Z0 + (Z0 + Z1)/(1 - ACC#(Z0 + Z1)));
      A2 = probnorm(Z0 + (Z0 + Z2)/(1 - ACC#(Z0 + Z2)));
```

```
            A5 = probnorm(Z0 + Z0/(1 - ACC#Z0));
            QPROB = A1//A5//A2;
            STUFF = ACC//Z0//A1//A2;
            mattrib STUFF colname={'ACC', 'Z0', 'A1', 'A2'};
            QMF = quantile(MFVEC, QPROB);
            MFSTAT = MFSTAT//(MF||QMF[2]||QMF[1]||QMF[3]);
      end;
      if &returnboot then do;
            create bootsample from MFVEC;
            append from MFVEC;
            close bootsample;
      end;
      if &hpd then
      mattrib MFSTAT rowname={'equal tailed', 'highest density'};
      if &hpd then
      mattrib HLSTAT rowname={'equal tailed', 'highest density'};
      else
      mattrib HLSTAT rowname={'equal tailed'};
      if &bca then
      mattrib MFSTAT rowname={'equal tailed', 'BC.a'};
      if &hpd & &bca then
      mattrib MFSTAT rowname={'equal tailed', 'highest density', 'BC.a'};
      if (&hpd = 0) & (&bca = 0) then
      mattrib MFSTAT rowname={'equal tailed'};
      print MFSTAT[label='Mitigated Fraction' format=.4 colname={'Observed'
      'Median' 'Lower' 'Upper'}];
      print HLSTAT[label='Hodges-Lehman' format=.4 colname={'Observed'
      'Median' 'Lower' 'Upper'}];
      print QQDIFSTAT[label='Quartile Differences' format=.4 rowname =
      {'Q25', 'Q50', 'Q75'} colname={'Observed' 'Median' 'Lower' 'Upper'}];
      print QXSTAT[label='Quartiles of con' format=.4 rowname = {'Q25',
      'Q50', 'Q75'} colname={'Observed' 'Median' 'Lower' 'Upper'}];
      print QYSTAT[label='Quartiles of vac' format=.4 rowname = {'Q25',
      'Q50', 'Q75'} colname={'Observed' 'Median' 'Lower' 'Upper'}];
quit;
%mend hlboot;
```

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Implementation of mitigated fraction estimators in SAS based on existing R package

Richting: **Master of Statistics-Biostatistics**
Jaar: **2013**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Nyaga, Victoriah

Datum: **11/09/2013**