# On Optimal Prediction Rules With Prospective Missingness and Bagged Empirical Null Inference in Large-Scale Data

By

Sarah Fletcher Mercaldo

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Biostatistics

September 30, 2017

Nashville, Tennessee

Approved:

Robert A. Greevy, Ph.D.

Jeffrey D. Blume, Ph.D.

Matthew S. Shotwell, Ph.D.

Thomas G. Stewart, Ph.D.

Melinda C. Aldrich, Ph.D.

To the love of my life, Nate

and

To my infinitely supportive parents, Wayne and Tammy

ACKNOWLEDGEMENTS

This dissertation could not have been completed without the great support that I have received from numerous people over the years. I offer my heartfelt thanks to the following people:

To my advisor, Dr. Jeffrey Blume, I sincerely owe him a debt of gratitude for his availability, sensibility, and unending kindness. He shepherded me for seven years as my mentor and friend, and encouraged me to persevere in my academic pursuits.

To my dissertation committee, Dr. Robert Greevy, Dr. Matthew Shotwell, Dr. Melinda Aldrich, and Dr. Thomas Stewart, I am grateful for their ideological insights and biostatistical expertise. Their viewpoints were invaluable throughout the progression of my dissertation.

To Dr. Eric Grogan and the Department of Thoracic Surgery, I thank them for the collaborative projects that inspired questions and sparked ideas for the research process. Their financial support of my research assistantship allowed me hands on experience that honed my skills as a biostatistician.

To my husband Nate Mercaldo, I love him for being my everything-my computer processor, my study partner, my personal think-tank, and my steadfast fan. He provided levity when I wanted to throw in the towel, and words of wisdom when I needed them most.

To Mom and Dad, I appreciate their selfless sacrifices to provide me every educational and emotional support possible. I could not have accomplished this without their years of prayers, reassurance, and belief in my abilities. I owe everything to them for giving me this beautiful life and constant outpouring of love.

To my sisters and extended family, I value their notes, phone calls, and doses of reality as they cheered me on through this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# LIST OF ABBREVIATIONS

AUC Area Under the Receiver Operating Characteristic Curve

B-H Benjamini-Hochberg

BEN Bagged Empirical Null

CCS Complete Case Submodels

EPE Expected Prediction Error

Fdr Global False Discovery Rate

fdr local False Discovery Rate

FWER Family-wise Error Rate

MAR Missing At Random

MARY Missing at Random where missingness depends on $Y$

MCAR Missing Completely At Random

MI Multiple Imputation

MICE Multivariate Imputations by Chained Equations

MIMI Multiple Imputation with Missingness Indicators

MNAR Missing Not At Random

MNARY Missing Not at Random where missingness depends on $Y$

PMKS Pattern Mixture Kernel Submodels

PMM Predicted Mean Matching

PMY Pattern Mixture $Y$

# CHAPTER 1

# INTRODUCTION

Risk prediction models are often constructed with the intention of estimating a predicted probability of disease or outcome for a new patient in the clinic. In order to apply these established prediction models, all the model covariates need to be known. Often this is not the case, and a strategy needs to be in place for a model to be used on a new person in the presence of missing data. Using common imputation strategies such as zero imputation or mean imputation may lead to subpar predictions, and using a more computational imputation strategy such as Multiple Imputation (MI), as outlined by van Buuren (2012), Harrell (2013), Janssen et al. (2010), Moons et al. (2006) and others, is often an unfeasible solution for on the spot computations. We detail an approach for dealing with missing covariate data in prediction models (in both the construction and application phase) whose goal is to maintain the predictive accuracy of the model (rather than parameter estimation as the ultimate goal). This approach requires no imputation and preserves the prediction accuracy achieved with MI. Since missing data are a common problem in both the construction and implementation of prediction algorithms, practical and accurate methods for accommodating missing data are sorely needed.

Our proposed solution presented in Chapter 2, is to use Pattern Mixture Kernel Submodels (PMKS) - a series of submodels for every missing data pattern that are fit using only data from that pattern. PMKS are a computationally efficient remedy for both stages. We show that PMKS yields the most predictive algorithm among all naive missing data strategies, such as zero imputation, mean imputation, and even more advanced strategies such as multiple imputation. We present detailed simulations and a real data application where the degree of improvement is highly dependent on the missingness mechanism and the effect size of missing predictors. When the data are Missing at Random (MAR), MI can yield comparable forecasting performance but generally requires a larger computational cost. When data are Missing Not at Random (MNAR), often the case in biomedical data, PMKS yields significantly better prediction than other imputation strategies.

An additional benefit of PMKS is its large sample equivalency to the limiting predictions from a multiple imputation procedure that uses a mean model dependent on missingness indicators (something we call the MIMI model). Consequently, the MIMI model can be used to help evaluate the MAR assumption in practice; a very

useful tool for both prediction and inference.

Large scale data presents the problem of testing thousands, if not millions, of hypotheses simultaneously. Chapter 3 details an approach to improve inference in large scale data by combining two well-known statistical techniques. First, we propose $p$-values that are standardized to the empirical null distribution (instead of the theoretical null) (Efron, 2012$a$). Second, we propose model averaging $p$-values by bootstrap aggregation (Bagging) to account for model uncertainty and selection procedures (Breiman, 1996). The combination of these two key ideas yields Bagged Empirical Null $p$-values (BEN $p$-values). BEN $p$-values improve operating characteristics of inferential procedures when the majority of tests are null - an assumption in these large-scale inference problems.

We have applied our proposed solution to pseudo-simulated gene expression data and the famous Golub leukemia data (Golub et al., 1999), and have demonstrated that our method is superior by having the most desirable Type I/Type II error tradeoff. A strength of our approach is the ability to consider any set of models (parameters, link function, non-parametric model) during bagging, as well as not being confined to the conventional theoretical null as the testing distribution. By incorporating bootstrapping, model selection, and empirical null procedures, the BEN algorithm has the advantage of using multi-dimensional gene selection metrics, beyond the single adjusted $p$-value traditionally used. We even provide biological verification that the findings are reproducible. Prior clinical experiments support our top selected gene which is overlooked by the seminal Golub paper.

Finally, Chapter 4 gives recommendations for how the imputation model should be adapted, regarding the inclusion and exclusion of the outcome, for the three risk prediction model developmental stages - construction, validation and application. Specifically, inclusion of the outcome ($Y$) in the imputation model is known to produce unbiased and efficient parameter estimates during the model construction (Moons et al., 2006). We show that using the outcome in the imputation model when data are missing in the out-of-sample validation set, leads to covariate imputations whose corresponding risk predictions will result in artificially increased model discrimination statistics. For a logistic prediction model, the validated AUC, Brier and Logarithmic scores are optimistically biased.

Extensive simulations and an application of our recommendations are presented. We demonstrate that in this setting where the prediction model will be used on patients with a high probability of missing data, it is strongly suggest that the outcome only be in the imputation of missing predictor values during model construction to

provide unbiased model parameters. Another imputation algorithm should be in place, excluding the outcome, for the imputation of missing covariates in the validation sample. These recommendations extend beyond logistic risk models to any type of prediction models.

Collectively, the entirety of these three papers will be widely applicable to a variety of biomedical settings. Chapter 2 will allow for near instantaneous predictions when out-of-sample individuals have missing predictors, a frequent reality where current imputation methods are degrading model accuracy. Chapter 3 will be widely applicable to a variety large-scale inference problems, and the BEN algorithm will lead to more robust and reproducible biological findings- a real concern in high-dimensional data. Chapter 4 suggests thoughtful consideration for the role of the outcome as part of the imputation model, and recommends the appropriate scenarios for its inclusion or exclusion. We intend for this work to provide a basis for future work in missing data, prediction, and large scale inference methodology.

# CHAPTER 2

# MISSING DATA AND PREDICTION: THE PATTERN MIXTURE KERNEL SUBMODEL

## 2.1 Abstract

Missing data are a common problem for both the construction and implementation of a prediction algorithm. Pattern mixture kernel submodels (PMKS) - a series of submodels for every missing data pattern that are fit using only data from that pattern - are a computationally efficient remedy for both stages. Here we show that PMKS yield the most predictive algorithm among all standard missing data strategies. Specifically, we show that the expected loss of a forecasting algorithm is minimized when each pattern-specific loss is minimized. Simulations and a re-analysis of the SUPPORT study confirms that PMKS generally outperforms zero-imputation, mean-imputation, complete-case analysis, complete-case submodels, and even multiple imputation (MI). The degree of improvement is highly dependent on the missingness mechanism and the effect size of missing predictors. When the data are Missing at Random (MAR) MI can yield comparable forecasting performance but generally requires a larger computational cost. We see that predictions from the PMKS are equivalent to the limiting predictions for a MI procedure that uses a mean model dependent on missingness indicators (the MIMI model). Consequently, the MIMI model can be used to assess the MAR assumption in practice. The focus of this paper is on out-of-sample prediction behavior; implications for model inference are only briefly explored.

## 2.2 Introduction

### 2.2.1 The Problem

While missing data are problematic for both estimation and prediction, the statistical literature has been largely focused on addressing the impact of missing data on estimation procedures and parameter inference. Missing data present a two-fold problem for forecasting: first, in building a model, and second, in using the model to make out-of-sample predictions for individuals with missing predictors. Here we focus on the second problem, specifically evaluating what Wood et al. (2015) define as Pragmatic Model Performance, which refers to the model's performance in a future clinical setting where some individuals may have partly missing predictors.

Table 2.1: Comparison of imputation methods for application of an established prediction model to an out-of-sample individual. PMKS: Pattern Mixture Kernel Submodel, CCS: Complete Case Submodel, MIMI: Multiple Imputation with Missingness Indicators, MI: Multiple Imputation.

| Imputation Strategy | Out-of-sample Imputation Requirement | Pros | Cons |
|---|---|---|---|
| **Zero Imputation** | Nothing | Negligible computation time | Zero may not be an appropriate value<br>Probably results in incorrect predictions |
| **Mean Imputation** | Unconditional means | Negligible computation time | Only works for the average individual |
| **Cond. Mean Imputation** | Conditional model for every missing pattern | Lower computation time<br>Can approximate a MI procedure | Large bias/variance tradeoff for MNAR |
| **CCS** | Submodels | Negligible computation time<br>May be advantageous if data are MAR<br>Fittable for unobserved patterns | Large bias/variance tradeoff for MNAR |
| **PMKS** | Submodels | Negligible computation time<br>Works for any missingness mechanism | May be less efficient if data are MAR<br>Patterns with low membership may not fit well |
| **MIMI** | Original data/Conditional distribution<br>Computer/Imputation engine | Works for any missingness mechanism<br>Allows for efficient parameter estimation<br>Established method | High computational cost<br>Not viable in the clinic |
| **MI** | Original data/Conditional distribution<br>Computer/Imputation engine | Works when data are MAR | High computational clinic<br>Not viable in the clinic<br>Large bias/variance tradeoff for MNAR |

It is often assumed that imputation methods, because they improve parameter estimation procedures, also improve out-of-sample prediction performance. However, this is just speculation, and our investigation indicates that this is the exception rather than the rule. Ideally, it would be possible to find a prediction rule that uses simple imputation or did not require imputation, and thus was much less computationally burdensome and more readily applied in practice. The impact of missing data for out-of-sample prediction is uniformly underestimated. A poor imputation algorithm at the prediction stage can drastically reduce a model's overall prediction performance, and data frequently go missing at this stage in real life.

### 2.2.2 Current Approaches to Imputation

Typical strategies for dealing with missing predictors are driven by practical constraints. Common strategies include zero imputation and mean imputation, which are trivial to implement, but often lead to poor predictions. Conditional mean imputation and multiple imputation can be implemented with accessible software when fitting a model, but they are rarely used in the clinic when predictors are missing for out-of-sample predictions (Janssen et al., 2009). The advantages and drawbacks of imputation methods for out-of-sample imputation procedures are listed in Table 2.1. The obvious issue, not well addressed in the literature, is the extent to which these approaches degrade prediction performance, as later shown.

Multiple Imputation (MI) draws multiple placeholder values from conditional dis-

tributions derived from the observed data (van Buuren, 2012; Janssen et al., 2010; Harrell, 2013), uses the placeholder values to fit the model, and then combines the models using Rubin's rules (Rubin, 2009). When the data are missing at random (MAR), MI can substantially increase inferential efficiency by leveraging information in incomplete data records. The 'best' predictions from a multiply imputed prediction model are the model's predictions averaged over all imputation sets (Vergouwe et al., 2010; Wood et al., 2015). Recently, the popularity of MI as the primary 'principled' strategy for constructing and applying a prediction model in the presence of missing data has grown (Harrell, 2013; Janssen et al., 2009).

However, applying a multiply imputed prediction model to an out-of-sample individual who is missing predictor information is not straightforward. This is because, technically speaking, the predictions need to be re-estimated with the imputation procedure based on the original data and the new out-of-sample record if you want to apply Predictive Mean Matching or K-Nearest Neighbor imputation techniques (as we did in our simulatons and examples). Of course, this requires the original data, the imputation datasets, and substantial on-demand computing power, which is often impractical in real world settings. Moreover, this approach often has a heavy computational burden and is not easily programmed in web applications (because the imputation algorithm must be repeated for every out-of-sample prediction). One could ignore this step and use the multiply-imputed model along with some one-step imputation procedure using the saved chain equations or fitted conditional distributions, but this process is not likely to be congenial with the original fitting approach.

### 2.2.3 Proposed Solution

Here we propose using an approach that we call the Pattern Mixture Kernel Submodels (PMKS) procedure. PMKS postulates a unique prediction model for each missing data pattern and estimates that model using only the subjects in that pattern. As a result, PMKS requires no imputation algorithm. Details are provided in section 2.4.1. As a forecasting algorithm, PMKS benefits greatly from the reduction in prediction bias that comes with the pattern-specific approach. The loss in efficiency that can result when the data are MAR, or when the patterns have common parameters, is often small in comparison. Moreover, we will see the PMKS is optimal in the sense of minimizing the expected prediction loss. Because of these advantages, we anticipate that PMKS will have broad impact in the arena of big data where prediction is paramount and MI is often computationally unfeasible e.g., when using an

entire system of electronic medical records.

### 2.2.4 Organization

Section 2.3 defines our notation and provides a brief background to key missing data concepts. Section 2.4 describes our proposed methods, provides a simple example, and draws connections between PMKS and MI models in some generality. Section 2.5 describes extensive simulations of PMKS in order to establish that wide range of setting under which PMKS excels. Section 2.6 describes the performance of PMKS compared to other imputation strategies applied to the SUPPORT Study, a multi-center two phase study of 9105 patients, from which a day 3 Physiology Score (SPS) was predicted (Phillips et al., 1995). Section 2.7 provides some brief concluding remarks.

## 2.3 Notation and Background

### 2.3.1 Notation

Let $\boldsymbol{Y} = (Y_1, ..., Y_n)$ be the vector of $n$-length observed responses. With our focus on prediction, we assume all responses are observed. This assumption can be relaxed, but it is not necessary to our discussion here. Predictors (covariates) are denoted by a $(n \times p)$ matrix $\boldsymbol{X} = (\boldsymbol{X_1}, ..., \boldsymbol{X_p})$ where $\boldsymbol{X_j} = (X_{1j}, ..., X_{nj})^T$ for $j = 1, ..., p$ predictor vectors of length $n$. Let $\boldsymbol{M} = \{M_{ij}\}$ be the $(n \times p)$ matrix of missing data indicators where indicator $M_{ij} = 1$ if $X_{ij}$ is missing and $M_{ij} = 0$ if $X_{ij}$ is observed for $i = 1, ..., n$ individuals and $j = 1, ..., p$ parameters.

To differentiate between models, we will use different greek symbols for their parameters. For example, parameters in the pattern mixture kernel submodels (PMKS) will be denoted with a $\boldsymbol{\gamma}$. Parameters in a traditional regression mean model $E[Y|\boldsymbol{X}] = \boldsymbol{X\beta}$ - those typically of interest in an estimation setting - will be denoted by $\boldsymbol{\beta}$. Notice this is a strong assumption with regards to the missing data, forcing the same mean-response function for every missing data pattern, and implying a MAR mechanism within a single marginal model.

$$E[Y|\boldsymbol{X}] = \boldsymbol{X\beta} \tag{2.1}$$

Parameters representing the effects of the missingness indicators, $\boldsymbol{M}$, will be denoted by $\boldsymbol{\delta}$; these parameters will distinguish our MIMI model (defined in Section 2.4.5) from a traditional MI model.

### 2.3.2 Pattern Mixture and Selection Models

Our approach has roots in the established literature on pattern mixture models (Little, 1993). The traditional pattern mixture model factorization is:
$P(\boldsymbol{Y}, \boldsymbol{M}|\boldsymbol{X}, \boldsymbol{\gamma}, \boldsymbol{\pi}) = P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{M}, \boldsymbol{\gamma})P(\boldsymbol{M}|\boldsymbol{X}, \boldsymbol{\pi})$, where $\boldsymbol{\pi}$ is a parameter vector for the missingness mechanism (Little and Rubin, 2014). The pattern-mixture approach allows for a different response model in each missing data pattern. An alternative formulation is the selection model: $P(\boldsymbol{Y}, \boldsymbol{M}|\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{\omega}) = P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{\omega})P(\boldsymbol{M}|\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\omega})$ where $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are parameter vectors (Little and Rubin, 2014; Little and Wang, 1996). This factorization describes the (single) marginal response model. In this paper, we will not explicitly consider selection models except to use them to generate data from certain missing data mechanisms that cannot be simulated otherwise. While the selection model allows for $\boldsymbol{Y}$ and $\boldsymbol{M}$ to be either independent or dependent, the pattern-mixture model is used when the response model changes by missing data pattern. This flexibility, it turns out, lends an advantage to our proposed PMKS prediction algorithm.

### 2.3.3 Missingness Mechanisms

To describe missing data, we use the following missingness mechanisms: Missing Completely at Random (MCAR), Missing at Random (MAR), Missing Not at Random (MNAR), Missing at Random where the missingness depends on $Y$ (MARY), and Missing Not at Random where the missingness depends on $Y$ (MNARY) (Little and Rubin, 2014). The latter two mechanisms can only be simulated in the selection model formulation. A more detailed description of these missingness mechanisms is given later in Section 2.3.

If the missingness mechanism is MCAR, then pattern mixture and selection models are equivalent (Little, 1993). When the data are not MCAR, the parameters of the kernel functions associated with the selection and pattern mixture models have different interpretations and care must be taken when estimating and interpreting them. The selection model describes the marginal relationship of $\boldsymbol{Y}$ on $\boldsymbol{X}$ while the pattern mixture model describes the relationship of $\boldsymbol{Y}$ on $\boldsymbol{X}$ conditional on $\boldsymbol{M}$. Marginal effects from the selection model are generally not identifiable in the context of a pattern mixture model, although some parameterizations can be identified though complete case restrictions that essentially force equality restraints on certain parameters (Little, 1993). Identifiability is obviously a problem when the goal is estimation and data are MNAR. However, here our goal is prediction and complex

re-parameterizations of marginal effects are not a major impediment even if the mapping is not easily reversed. If one marginal model is truly of interest, it is always possible to marginalize over the pattern-specific model. Of course, how that model should be interpreted when the data are not MAR is not immediately clear.

### 2.3.4 Complete Case Models and Submodels

Two alternatives to multiple imputation are complete case models and complete case submodels. A complete case analysis simply ignores the records with missing data and estimates a single model. Note that complete-case models have routinely poor prediction performance when the data are not Missing Completely at Random (MCAR) (Knol et al., 2010; Janssen et al., 2009). Complete Case Submodels (CCS) use all available data to fit a submodel for each missing data pattern. That means, for CCS, a data record often contributes to multiple patterns/submodels. This approach is vastly different from a complete-case analysis where just a single model is fit using only subjects with complete data.

Note that the complete case model does not solve the problem with missing out-of-sample predictors; some type of imputation is still required. In contrast, CCS do not have this problem. Their predictions come from the submodel that matches the out-of-sample missing data pattern and so no imputation is needed. For PMKS, each data record contributes only to a single pattern/submodel, which turns out to be a key difference compared to CCS.

## 2.4 Methods

### 2.4.1 Pattern Mixture Kernel Submodels

PMKS has roots in pattern mixture model, using the kernel of the pattern mixture model as a prediction machine. These pattern-specific models are the PMKS. To make predictions, we fit the set of models $\{\hat{f}_1, ..., \hat{f}_k\}$ where $\hat{f}_m = \hat{f}_m(\boldsymbol{X}, \boldsymbol{M})$ is the pattern mixture model in pattern $m = 1, ..., k$ where $k \leq 2^p$ different patterns. Here we consider straightforward prediction algorithms such as $\hat{f}_m = E(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{M}; \hat{\boldsymbol{\gamma}}_m)$ where $\hat{\boldsymbol{\gamma}}_m$ is the vector of estimated pattern specific parameters. However, our findings apply to any generalized linear model or machine learning prediction technique. Although up to $k = 2^p$ different models might have to be fit, in practice only a small fraction of those patterns are observed.

Each PMKS is fit using only subjects from that pattern. This is in contrast to Complete Case Submodels (CCS) where the submodels are fit using all available data.

Table 2.2: Comparison of Pattern Mixture Kernel Submodels (PMKS) and Complete Case Submodels (CCS).

| **Pattern** | **PMKS** ($\hat{f}_m$) | **CCS** ($\hat{g}_m$) |
|---|---|---|
| 1:$X_1^{obs}, X_2^{obs}$ | $E[Y\|X_1, X_2, M_1 = 0, M_2 = 0] = \gamma_{0,1} + \gamma_{1,1}X_1 + \gamma_{2,1}X_2$ | $E[Y\|X_1, X_2] = \beta^*_{0,1} + \beta^*_{1,1}X_1 + \beta^*_{2,1}X_2$ |
| 2:$X_1^{miss}, X_2^{obs}$ | $E[Y\|X_2, M_1 = 1, M_2 = 0] = \gamma_{0,2} + \gamma_{2,2}X_2$ | $E[Y\|X_2] = \beta^*_{0,2} + \beta^*_{2,2}X_2$ |
| 3:$X_1^{obs}, X_2^{miss}$ | $E[Y\|X_1, M_1 = 0, M_2 = 1] = \gamma_{0,3} + \gamma_{1,3}X_1$ | $E[Y\|X_1] = \beta^*_{0,3} + \beta^*_{1,3}X_1$ |
| 4:$X_1^{miss}, X_2^{miss}$ | $E[Y\|M_1 = 1, M_2 = 1] = \gamma_{0,4}$ | $E[Y] = \beta^*_{0,4}$ |
| $\gamma_{p,m}, \beta^*_{p,m}$ represents the effect of the $p^{th}$ covariate in pattern $m$ | | |

This subtle difference turns out to be critically important; as CCS is only appropriate under certain missing data mechanisms. For comparison, we denote the set of CCS models as $\{\hat{g}_1, ..., \hat{g}_k\}$ where $\hat{g}_m = \hat{g}_m(\boldsymbol{X}) = E(\boldsymbol{Y}|\boldsymbol{X}; \hat{\boldsymbol{\beta}}^*{}_m)$ for patterns $m = 1, ..., k$. Here we use $\hat{\boldsymbol{\beta}}^*{}_m$ as the estimated pattern specific parameter vector. The asterisk here is used to distinguish the CCS parameters from those obtained in a complete case analysis ($\hat{\beta}$). The difference between $\hat{f}_m$ and $\hat{g}_m$ is illustrated below in the context of a simple example. Later in Section 3.4, we discuss how to fit PMKS and CCS when a pattern is not observed or the data are too sparse. CCS have the advantage of being fittable in many of these situations with the obvious drawback of its strong dependence on the MAR assumption.

### 2.4.2 A Simple Example

A simple illustration helps to fix ideas. Consider a linear model for continuous outcome $Y$ with two covariates $X_1$, $X_2$. There are only four missing data patterns: (1) $X_1$, $X_2$ both observed; (2) $X_1$ missing, $X_2$ observed; (3) $X_1$ observed, $X_2$ missing; (4) $X_1$, $X_2$ both missing. The pattern mixture kernel submodels are the set of corresponding response models in each missing data pattern. These are given in table 2.2.

Using the above notation, the estimated PMKS response function for $E[Y|X_1, M_1 = 1, M_2 = 0]$ is $\hat{f}_3$ while the CCS analogue for $E[Y|X_1]$ is $\hat{g}_3$. Note that $\gamma_{p,m}$ does not necessarily equal $\beta^*_p$ for any $m = 1, 2, 3, 4$. Also, none these submodels corresponds to the typical marginal model expressed as $E[Y|X_1, X_2] = \beta_0 + \beta_1 X_1 + \beta_2 X_2$ where the parameters $\boldsymbol{\beta}$ represent traditional direct effects. It is tempting to assume that the model based on pattern 1 would yield proper estimates of this marginal model, but this only happens when the data are MAR on the covariates (and not on the response). This is because that is the only case where the marginal model corresponds exactly with the data generating mechanism in each pattern. That is, it is the only

case where the marginal model is true mean response model for every pattern (White and Carlin, 2010; Bartlett et al., 2014).

### 2.4.3 Prediction Performance of PMKS

PMKS is computationally efficient for missing data because it avoids the issue; it fits a series of models in which none have any missing data. Fitting each submodel is now straightforward because the missing data problem has been avoided. The key realization is that minimizing the expected loss in each pattern amounts to minimizing the expected loss marginally. Thus, we need only use standard techniques to fit and cross-validate the pattern specific models.

#### *2.4.3.1 Minimizing the Expected Prediction Error*

Minimizing the expected prediction error in each pattern will, in turn, minimize the overall expected prediction error. To see this, note that:

$$\begin{aligned} E_{Y|X}[L(Y, \hat{f}(\boldsymbol{X}))] &= E_M \left[ E_{Y|\boldsymbol{X},\boldsymbol{M}} \left[ L\left(Y, \hat{f}_m\right) \right] \right] \\ &= \sum_M P(M) E_{Y|\boldsymbol{X},\boldsymbol{M}} \left[ L\left(Y, \hat{f}(\boldsymbol{X}, \boldsymbol{M})\right) \right] \end{aligned}$$

where $\hat{f}_m = \hat{f}_m(\boldsymbol{X}, \boldsymbol{M})$. Hence, selecting $\hat{f}_m$ to minimize the pattern specific expected loss, $E_{Y|X,M}\left[L\left(Y, \hat{f}_m(\boldsymbol{X}, \boldsymbol{M})\right)\right]$, will in turn minimize the overall loss $E_{Y|X}[L(Y, \hat{f}(\boldsymbol{X}))]$.

Here $L\left(Y, \hat{f}_m(\boldsymbol{X}, \boldsymbol{M})\right)$ must be a properly defined pattern-specific loss function. The loss function is flexible; it could be squared error or 0/1 loss (Hastie et al., 2009). But the pattern-specific restriction is important; the result might not hold for certain metrics where predictions in one pattern are compared with predictions in another (for example, the area under the ROC curve). In that case, the overall AUC is not equal to the average of pattern specific AUCs, which is the property we need in order to take advantage of this approach. Fortunately, most common loss functions of interest in prediction problems have this property.

Note that constructing a prediction model within each missing data pattern effectively resolves the missing data dilemma because the missing predictors are missing for everyone in that pattern and only marginal effects can be estimated. The result implies that, in practice, prediction models should be constructed and cross validated within each pattern in order to maximize predictive ability. The only reason to do this marginally is if the MAR assumption is known to hold. Then direct estimation of $E_{Y|\boldsymbol{X}}\left[Y, \hat{f}(\boldsymbol{X})\right]$ is complex, in part, because only a single model $\hat{f}$ is used for all

predictions and fitting that model with missing data requires a complex algorithm such as multiple imputation or the EM algorithm to handle the missing data. Since each PMKS is directly estimable with routine tools, the right side of the equation can be fit, minimized, and cross validated rather easily. This simple argument impacts practice because $M$ is often ignored in the modeling stage due to historical concerns about the missing indicator method.

*2.4.3.2 PMKS loss is a weighted average of a full and reduced model*

For a linear model the squared prediction error is a common and relevant loss function. To examine the bias-variance tradeoff in PMKS, it is helpful to revisit a simple example given by Shmueli (2010) in which the Expected Prediction Error (EPE) is evaluated for a "fully specified model" (large) versus an "underspecified model" (small). Suppose data come from the model $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$ with $\epsilon \sim N(0, 1)$. When no predictors are missing we estimate the full model as $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$. Here the expected prediction error (EPE) is the sum of the bias, variance, and irreducible error of the predictions or fitted values (Hastie et al., 2009):

$$\text{EPE}_L = E\left[\left(Y - \hat{f}(x_1, x_2)\right)^2\right] = \sigma^2\left(1 + [1 \quad x_1 \quad x_2](X_L' X_L)[1 \quad x_1 \quad x_2]'\right)$$

where $\text{EPE}_L$ denotes the EPE of the full model. In contrast the EPE of the underspecified or submodel is given by:

$$\text{EPE}_S = E\left[\left(Y - \hat{f}^*(x_1)\right)^2\right] = ((\gamma_0 + x_1\gamma_1) - (\beta_0 + \beta_1 x_1 + \beta_2 x_2))^2 + \sigma^2[1 \quad x_1](X_S' X_S)[1 \quad x_1]'$$

where $\hat{f}^*(x) = \hat{\beta^*}_{0,1} + \hat{\beta^*}_{1,1} x_1$. Note that in this case $\hat{f}^*(x) = \hat{g}_2$ . The EPE of the PMKS model is just a weighted average of the large and small prediction models.

$$\text{EPE}_{PMKS} = \sum_m P(M = m)\text{EPE}_m = \text{EPE}_L(1 - P(M)) + \text{EPE}_S P(M)$$

To illustrate the bias variance tradeoff, we simulated the EPE in Figure 2.1. The simulation fixes $X_1 = x_1$, and draws from the conditional distribution $X_2 | X_1 = x_1 \sim N(\mu_2 + \frac{\sigma_2}{\sigma_1}\rho_{1,2}(x_1 = \mu_1), (1 - \rho_{1,2}^2)\sigma_2^2)$. The EPE for the correctly specified full model is just the irreducible error, whereas the EPE for the underspecified model increases as the out-of-sample predictor moves away from its population mean.

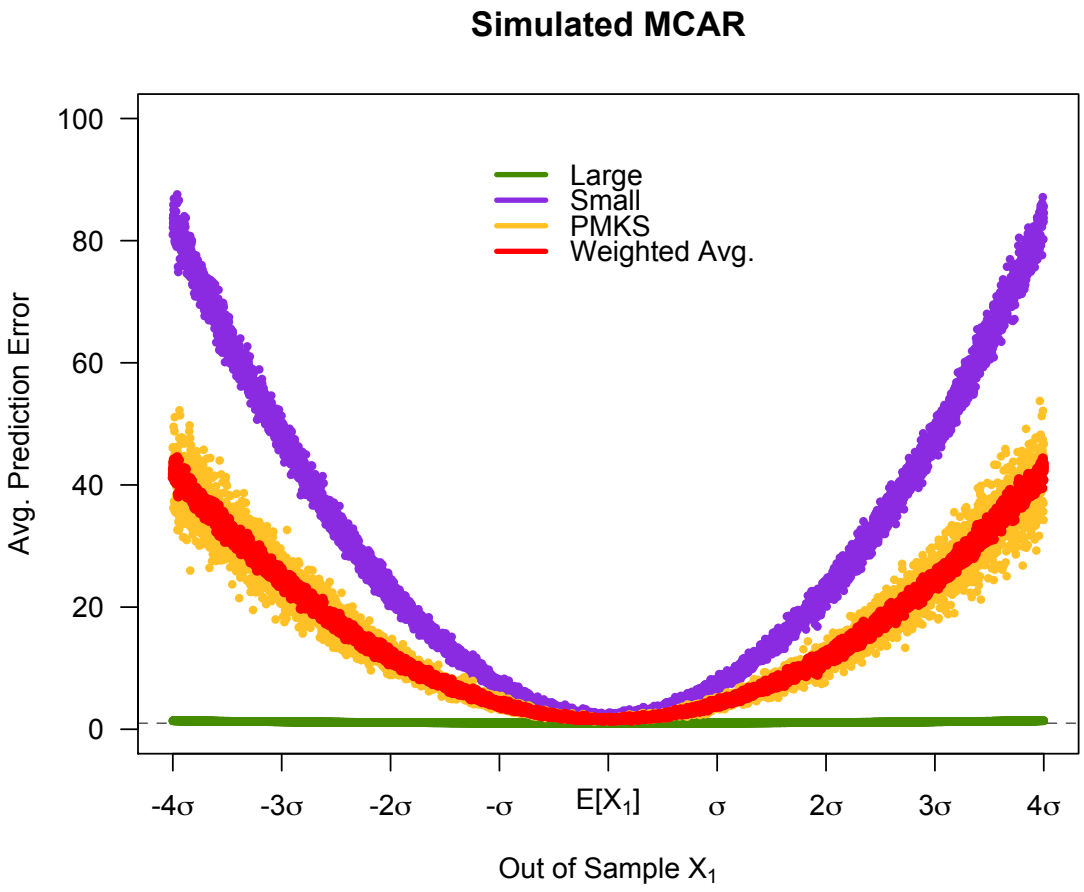



Figure 2.1: Comparison of Expected Prediction Error for the Large fully specified model: $E[Y|X_1, X_2] = \beta_0 + \beta_1 X_1 + \beta_2 X_2$, and Small underspecified model: $E[Y|X_1] = \beta_{0,3} + \beta^*_{1,3} X_1$. Pattern Mixture Kernel Submodel (PMKS) predictions are a weighted average of the Large and Small models, weighted by P(M=1)

The out-of-sample prediction error from the large model is given by the green line in Figure 2.1, and is equal to the model variance. If the data were generated from the large model and predictions were given from the small model that includes only $X_2$, then the expected prediction error is approximated by the purple points in Figure 2.1.

The yellow points in Figure 2.1 denote the prediction error that arose from the PMKS in this setting; $\hat{f}_1$ makes predictions when all data are available and $\hat{f}_2$ makes predictions when only $X_2$ is available. Clearly, PMKS has smaller EPE for every out-of-sample $X_1$. In this case the probability of missingness was 50%, $P(M_1 = 1) = 0.5$.

### 2.4.4 Practical Considerations

As discussed earlier, multiple imputation has a substantial computational burden for out-of-sample predictions with missing data because the imputation algorithm must be repeated adding the new person in the data run. PMKS, on the other hand, do not need to be re-computed for every new prediction. The upfront computational effort is large for $2^p$ patterns, but it is often minor compared to the MI machinery required in the same setting and in practice only a fraction of available patterns are observed. When data are sparse within a pattern, it may not be possible to fit the PMKS. In such cases it is necessary to make assumptions and the CCS approach is a reasonable option. This hybrid approach has worked well for us in practice, in large part because the contribution to the EPE for patterns that are too sparse to fit with PMKS is often negligible.

If $p$ is very large, and storing $2^p$ prediction models seems unreasonable, there are several options. First, fit models only for observable patterns, ignoring patterns not observed. Second, only fit models for patterns in which the missing variable, or combinations of missing variables, are 'important' to the predictions. Third, if the data are available in real time, it may be possible to fit PMKS on demand, since imputation is not necessary, and this reduces the need to store all $2^p$ models simultaneously. Lastly, shrinkage methods to the MIMI model, discussed in Section 2.4.5, can indicate how best to borrow strength over the patterns.

It is important to distinguish between the computation cost between the in-sample model construction phase and out-of-sample prediction phase. Both PMKS and MI could have high in-sample computation cost depending on the number of predictors and the data size. But as described in Table 2.1, the out-of-sample computational costs for PMKS are negligible, whereas for MI they can remain intensive, even for a single individual. Importantly, PMKS does not require missing data mechanisms to be consistent in the data used to construct the model and the target population. This is because, conditional on the missingness pattern, the data are effectively MAR. The MAR assumption implies $(M \perp X_j)|\boldsymbol{X_{-j}}, \forall j = 1..p$. PMKS reformulates this assumption as $(M \perp X_j)|(\boldsymbol{X_{-j}}\boldsymbol{M_{-j}})$, allowing the MAR assumption to hold within each pattern (Little, 1993).

### 2.4.5 PMKS as the limit of MIMI model

There are some interesting connections between PMKS and MI. PMKS is the limit of a congenial MI procedure when the mean model depends on the missing data

indicators. This new MI procedure - which we call this the Multiple Imputation with Missingness Indicators (MIMI) model - can be used to assess the MAR assumption in practice. The MIMI model also makes the context clear about what elements of the model can be assumed identical across the patterns for inference purposes. The utility of missingness indicators can only be realized through an imputation procedure, an often overlooked and important point. The implications for estimation will be discussed elsewhere, as the focus of this paper is on prediction, but the connection is an important one.

The MIMI model is a multiple imputation model that is dependent on the indicators $M_i$ from $i = 1, .., p$. Typically, the mean model would depend on the missingness indications. For example, in the past example with $p = 2$ covariates, $X_1$ and $X_2$, we might consider the following mean model:

$$\begin{aligned} E[Y|X_1, X_2, M_1, M_2] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \delta_1 M_1 + \delta_2 M_2 \\ + \delta_3 X_1 M_1 + \delta_4 X_2 M_2 + \delta_5 X_1 M_2 + \delta_6 X_2 M_1 \end{aligned} \tag{2.2}$$

where the $\boldsymbol{\beta}$ parameters represent the traditional direct effects of interest and the $\boldsymbol{\delta}$ parameters, which we will call auxiliary parameters, explain how the traditional effects change by missing data pattern. If the data are MCAR, then $\delta_i = 0 \; \forall \; i$. Otherwise, the traditional effects might not exist as the dependencies in the model parameters can be complex. An informal test of the $\boldsymbol{\delta}$ parameters may provide some insight into the observed missing data mechanism. If there is no evidence to suggest that the indicators contribute to the model predictions, this may help to make the case for a MAR mechanism. Shrinkage methods may also be applied to the delta $\boldsymbol{\delta}$ parameters to help assess which covariates have non-ignorable missing data mechanisms, and will be addressed in future work.

Note, Molenberghs et al. (2008) describes a longitudinal setting where every MNAR model can be decomposed into a set of MAR models. Molenberghs et al. (2008) rightly asserts that this duality is problematic for inference about a parameter. However, as noted in the paper, these representations will yield different predictions and so the implications are different in out context where prediction is the objective. The results from Molenberghs et al. (2008) are important in that they align with our results in the non-longitudinal setting, were predictions must match those from the natural pattern mixture model (that is MAR by definition conditional on the pattern) in order to retain its predictive optimality.

Unfortunately, Model 2.2 cannot be properly fit unless the missing predictors are

imputed. When the missing data are imputed with a proper MI algorithm, the coefficients of Model 2.2 are essentially identified by that algorithms imputation scheme and the auxiliary parameters become estimable under those assumptions. Model 2.2 is an example of the Multiple Imputation with Missingness Indicators (MIMI) model. Note that MIMI makes certain assumptions about the missingness mechanism, and these assumptions will lead to different fits of the auxiliary parameters. This flexibility is both good and bad; good because we could use the auxiliary parameters to check assumptions, and bad because the auxiliary parameters are inherently unidentifiable.

Adding missingness indicators to a model (when the goal was parameter estimation) has received criticism historically. The simple plug-in varieties of the missing-indicator method yields biased parameter estimates even in simple cases where data are missing completely at random (MCAR) (Allison, 2001; Groenwold et al., 2012). The classical missing-indicator method fills in a constant (often zero or the overall mean) for the missing values and augments the data design matrix with a binary indicator for each covariate with missing values. However, when the missing-indicator method is combined with proper imputation methods the model produces unbiased parameter estimates in the same cases in which complete case estimation is unbiased (Jones, 1996; Dardanoni et al., 2011, 2015). That is, when $M_i$ and $Y$ are conditionally independent given $X_i$, then for any choice of imputation matrix, the OLS estimate of Model 2.2 coincides with the OLS estimate of $\boldsymbol{\beta}$ in the complete case model (Bartlett et al., 2014; Jones, 1996; Dardanoni et al., 2011; White and Carlin, 2010). Thus, the MIMI model is essentially an extension of the ideas in Jones (1996) to a more flexible multiple imputation setting.

Although the missing-indicator method has been heavily investigated in the context of inference, this method has not been explored for prediction where a bias-variance tradeoff may be more desirable. The MIMI model leverages multiple imputation to create placeholders for the missing data that do not negatively impact the model's predictive ability. But of course, the value of these imputations does impact the estimation of direct effects and the properties of those estimators.

The connection between the MIMI model and PMKS can be seen through a simple rearrangement of the mean Model 2.2. There are differences; the MIMI model forces constant variance across all missing data patterns, whereas PMKS allows the variance to change by pattern. PMKS are most easily understood as projections of the true pattern-specific model into the space of observed covariates. As such, slope parameters for observed covariates may be distorted if missing covariates are correlated with observed covariates. In those cases, the model is really estimating the total

effect when the direct effect is the quantity of interest. PMKS is a series of models based on the total effects that can be estimated from the data at hand, while MIMI tries to reparameterize each patter-specific mean model and average the direct effects of interests.

Applying the plug-in principle, the MIMI mean model reduces to the PMKS when conditional mean imputation is used to impute missing covariates. Denote the imputed covariates as $X_i^* = E[X_1|X_2] = \alpha_0 + \alpha_2 X_2$ if $X_{i1}$ is missing and $X_{i1}$ otherwise.

Rearranging the MIMI model we have:

$$\begin{aligned} E[Y|X_1, X_2, M_1, M_2] =& (\beta_0 + \delta_1 M_1 + \delta_2 M_2) \\ & (\beta_1 + \delta_3 M_1 + \delta_5 M_2) X_1 \\ & (\beta_2 + \delta_4 M_2 + \delta_6 M_1) X_2 \end{aligned}$$

Which just reduces to PMKS. To illustrate, for the 4 patterns in our running example:

$$\begin{aligned} E[Y|X_1, X_2, M_1 = 0, M_2 = 0] &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 \\ E[Y|X_1, X_2, M_1 = 1, M_2 = 0] &= (\beta_0 + \delta_1) + (\beta_1 + \delta_3) E[X_1|X_2] + (\beta_2 + \delta_6) X_2 \\ &= (\beta_0 + \delta_1) + (\beta_1 + \delta_3)(\alpha_0 + \alpha_2 X_2) + (\beta_2 + \delta_6) X_2 \\ &= (\beta_0 + \delta_1 + \beta_1 \alpha_0 + \delta_3 \alpha_0) + (\beta_2 + \delta_6 + \beta_1 \alpha_2 + \delta_3 \alpha_2) X_2 \\ &= \gamma_0 + \gamma_2 X_2 \end{aligned}$$

Note that model $E[Y|X_1, X_2, M_1 = 1, M_2 = 0] = \gamma_0 + \gamma_2 X_2$ is the submodel including only the covariate $X_2$ fit within the group of individuals who are missing the covariate $X_1$ (this is why the conditioning on $M$ is important) and is equivalent to $\hat{f}_2$ in Section 2.4.1. Hence, PMKS and MIMI are two parameterizations of the 'same' model. To examine this, we next present simulations in the linear model case under a wide variety of missing data mechanisms. This principle extends to the GLM setting, on the linear predictor scale, and is currently under investigation by the authors. More complex prediction machines that are highly non-linear should exhibit the same general behavior.

## 2.5 Simulations

We generated $n$ multivariate normal predictor vectors according to $\binom{x_1}{x_2} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = (3, 3)$ and $\boldsymbol{\Sigma} = \left(\begin{smallmatrix} 1 & 0.5 \\ 0.5 & 1 \end{smallmatrix}\right)$, for example, are set to provide certain predictor profiles in terms of their correlation. Simulated outcomes $Y$ are generated from various combinations of $x_1$ and $x_2$. The pattern mixture model formulation uses $\boldsymbol{X}$ to induce one of three missing data mechanisms, MCAR, MAR, or MNAR. The outcome $Y$ is then generated from the MIMI mean model using the true $\boldsymbol{X}$ values and the simulated missing data indicators. Here the missingness may only depend on the predictors vector $X$. In contrast, the selection model formulation simulates $Y$ from the marginal model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$, where $\epsilon \sim N(0, 1)$. Missing data indictors are then induced according to the desired mechanism. Note that here the missingness may depend on the outcome $Y$. A more complex model can always be reduced to a linear combination of non-missing variables, and missing variables, and so this simple example is representative of more complicated situations.

We simulated the following five missing data mechanisms as defined in Table 2.3 for this situation: MCAR, MAR, MNAR, MARY, and MNARY. The latter two mechanisms could only be simulated in the selection model formulation. We forced the missingness data mechanism to be consistent between the in-sample and out-of-sample populations, and $\nu_0$ of Table 2.3 is empirically calculated to maintain the desired probability of missingness.

Table 2.3: Missing data mechanisms used for simulation. $\nu_0$ is empirically calculated to allow the probability of missingness to maintain the desired level. $expit = \frac{e^x}{1+e^x}$.

| | **Missing Data Mechanism for $X_1$** |
|---|---|
| **MCAR** | $P(M) = expit(\nu_0)$ |
| **MAR** | $P(M) = expit(\nu_0 + \nu_2 X_2)$ |
| **MARY** | $P(M) = expit\left(\nu_0 + \nu_{2,Y}\left(\frac{Y/\sigma_y + X_2}{\sqrt{2(1+cor(Y,X_2))}}\right)\right)$ |
| **MNAR** | $P(M) = expit(\nu_0 + \nu_1 X_1)$ |
| **MNARY** | $P(M) = expit\left(\nu_0 + \nu_{1,Y}\left(\frac{Y/\sigma_y + X_1}{\sqrt{2(1+cor(Y,X_1))}}\right)\right)$ |

### 2.5.1 Parameters

Parameter profiles explored were $\beta_1 = 1, 3, 5$, $\rho = 0, 0.5, 0.75$, $P(M_1 = 1) = 0.20, 0.50, 0.75$, and $n = 50, 200, 500, 1000$. We present here only one case that was

largely representative of our findings: $\beta_1 = 3$, $\rho = 0.5$, $P(M_1 = 1) = 0.50$, and $n = 1000$. For the out of sample population we assumed one-by-one enrollment. Missing data was imputed by zero imputation, unconditional mean imputation, single conditional mean imputation using a Bayesian conditional mean model, single conditional mean imputation using a frequentist conditional mean model, or multiple imputation (predictive mean matching, 10 imputations). We fixed the imputation engine based on the in-sample population to closely mimic real world application of these methods.

### 2.5.2 Simulation procedure

We compared the performance of PMKS, complete case model predictions, complete case submodel (CCS) predictions, traditional MI and the MIMI imputation model. The full simulation procedure was as follows: (1) data are generated and missing data indicators are generated according to the missing data mechanism in Table 2.3, as described in Section 2.5; (2) missing data are imputed; (3) the MI model, MIMI model, CCS, and PMKS models are fit; (4) step 1 is repeated to obtain a new out-of-sample population; (5) individuals are imputed one by one, using the above imputation procedures, assuming a fixed imputation engine from the in-sample population; (6) individual predictions and performance measures are computed; (7) steps 1 through 6 are repeated 10,000 times.

A squared error loss function was used to compare performance of the approaches. For example the squared error loss across all missing data patterns in the PMKS is $\frac{1}{n}\sum_i \sum_j P(M_i = 1)(Y_{ij} - \hat{Y}_{ij})^2$ where $j = 1, \ldots, n$ subjects and $i = 1, \ldots, m$ patterns. This loss is the averaged over the 10,000 simulations to approximate the expected loss. Table 4.1 shows the average squared imputation error for predictor $x_1$ as a function of imputation strategy and missingness mechanism.

### 2.5.3 Simulation Results

Results are presented for the following set of parameters: $\beta_0 = 1, \beta_1 = 3, \beta_2 = 1, \delta_1 = 1, \delta_3 = 1, P(M_1 = 1) = 0.5, \nu_1 = 1, \nu_2 = 1, \nu_{1,Y} = 1, \nu_{2,Y} = 1$. There were negligible differences in pattern specific and total squared error loss for the MCAR missing data mechanism. For all missing data scenarios, MI and conditional mean imputations resulted in a biased parameter estimation. This bias appears most clearly in predictions for observations without missing data (blue dots in Figure 2.2). When $Y$ is added to the MI model, the model parameters had negligible bias. However, since

the out-of-sample $Y$ is missing, the out-of-sample imputations of $x_1$ have greater bias than the imputation model in which $Y$ is not included resulting in a higher total prediction error (e.g., see Table 2.4).

When $Y$ is generated from a selection model formulation, all methods perform similarly (apart from MI as described above) under the MAR missing data mechanism. When data are MNAR, PMKS and the MIMI models have slightly lower total and pattern specific squared error loss compared to the traditionally available methods. When $Y$ is generated under the pattern mixture formulation with a MNAR missing data mechanism (MNAR PMY), PMKS and MIMI have both lower pattern specific contributions to the prediction error (PE) in the pattern where $x_1$ is missing, and lower total prediction error compared to all other methods.

As might be expected, PMKS and CCS have different out-of-sample prediction performance when the missing data are not Missing at Random (MAR). In fact, PMKS minimizes the expected prediction loss regardless of missingness mechanism, while CCS tends to rival PMKS only when the data are MAR. We will see that when the data are modified to induce a Missing Not at Random (MNAR) mechanism, PMKS has optimal predictions on average compared to traditional methods.

As both the strength of the missingness mechanism and the beta coefficient associated with the missing variable increase, the magnitude of the differences in methods favors PMKS/MIMI over all the other methods.

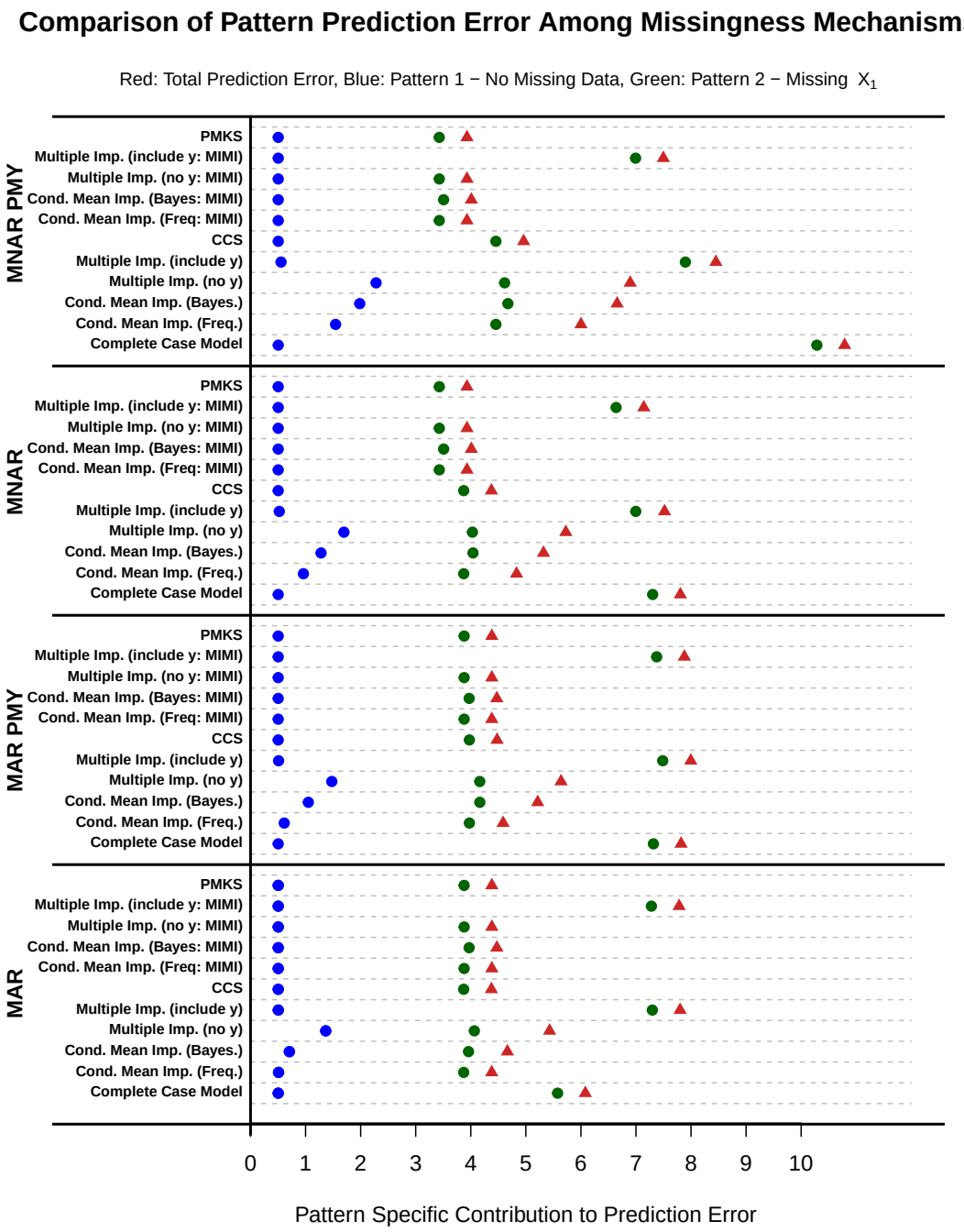



Figure 2.2: Simulation Results set of parameters: $\beta_0 = 1, \beta_1 = 3, \beta_2 = 1, \delta_1 = 1, \delta_3 = 1, P(M_1 = 1) = 0.5, \nu_1 = 1, \nu_2 = 1, \nu_{1,Y} = 1, \nu_{2,Y} = 1$. The missing data mechanisms Missing at Random (MAR) and Missing Not at Random (MNAR) were generated under a Pattern Mixture $Y$ (PMY) and Selection Model $Y$ formulation. Red triangles represent the Total Prediction Error (TPE) summed over all missing data patterns. Blue circles represent the PE for pattern 1 where there is no missing data. Green circles represent the PE for pattern 2 in which $x_1$ is missing.

Table 2.4 of out-of-sample imputations of $x_1$ provides insight into some of the biases seen in Figure 2.2. When $Y$ is included in the imputation model during model construction, parameter estimates tend to be unbiased. However, when $Y$ is included as part of the fixed in-sample MI model, but $Y$ is assumed to be missing for the out-of-sample individual, MI performed using predictive mean matching and chained equations bias the imputations of $x_1$. In this case $x_1$ has the largest squared error of all the imputations procedures for every missing data mechanism apart from unconditional mean imputation. Although the apparent bias in imputations for missing covariates seem small, their total contribution over all individuals can be quite significant. These results show that biases in imputing missing predictors leads to poorer downstream predictions and larger prediction error for the outcome.

Table 2.4: Squared Imputation Error of the true $X_1$ compared to the imputed $X_1$ under different imputation methods and missing data mechanisms: Imputation Error of $X_1 = \sum_i (X_{1i} - \hat{X}_{1i})^2$. MAR: Missing at Random, MAR PMY: Missing at Random with Pattern Mixture Y calculation, MNAR: Missing Not at Random, MNAR PMY: Missing Not at Random with Pattern Mixture Y calculation, Cond. Mean: Conditional Mean Imputation using a regression model and a Bayes procedure, MI: Multiple Imputation with and without Y in the imputation algorithm. The Mean (SD) are presented for each type of imputation and missingness mechanism.

| | MAR | MAR PMY | MNAR | MNAR PMY |
|---|---|---|---|---|
| Unconditional Mean | 0.56 (0.03) | 0.56 (0.03) | 0.76 (0.03) | 0.76 (0.03) |
| Cond. Mean | 0.38 (0.02) | 0.38 (0.02) | 0.53 (0.03) | 0.53 (0.03) |
| Cond. Mean (Bayes) | 0.49 (0.03) | 0.49 (0.03) | 0.69 (0.03) | 0.69 (0.03) |
| MI (No Y) | 0.47 (0.03) | 0.47 (0.03) | 0.61 (0.03) | 0.61 (0.03) |
| MI (Y) | 0.76 (0.06) | 0.74 (0.06) | 0.75 (0.6) | 0.71 (0.08) |

Authors have explored in detail the advantages of including Y in the imputation model (Moons et al., 2006). Using $Y$ in the imputation model during model construction leads to unbiased estimates of regression coefficients. Whereas this may be a fine approach during the model building (in-sample) population, it is not practical in the prediction setting where the outcome is unknown and would be imputed as part of the fixed imputation model. The chained equations imputation model can lead to biased imputations for the missing covariates (in these simulations $X_1$), when the outcome (in these simulations $Y$) is imputed as part of the chain. We do not present the situation in which $Y$ was used in the in-sample imputation model to produce unbiased regression estimates, but not included in the out-of-sample imputation model - a combination which would have less propagated imputation bias. Even though it

may seem that the inclusion of $Y$ in the imputation model will lower prediction error, careful thought and attention need to be placed on the practicality of this, as well as the statistical implications.

## 2.6 Application: SUPPORT Data Example

The Study to Understand Prognoses and Preferences for Outcomes and Risks of Treatments (SUPPORT) was a multi-center, two phase study, of 9105 patients. The primary goal of the study was to model survival over a 180-day period in seriously ill hospitalized adults (Phillips et al. (1995)). A component of the SUPPORT prognostic model was the SUPPORT day 3 Physiology Score (SPS), a risk score created to account for various sources of health variation and co-morbidities. The SUPPORT physiology score can range from 0 to 100 and was derived from the following covariates: Disease group (4 levels), Partial pressure of oxygen in the arterial blood, Mean blood pressure, White blood count, Albumin, APACHE III respiration score, temperature, Heart rate per minute, Bilirubin, Creatinine, and Sodium. The SPS model allowed Mean Blood pressure, White blood count, Albumin, Temperature, HR, Bilirubin, Creatinine, and Sodium to have a nonlinear association with SPS, and included certain interactions with disease group and albumin, and disease group and white blood count.

For our illustrative example, we choose to model SPS score because it was a known quantity and we could be sure there were not any major predictive factors that were completely missing. We allowed for stochastic variation by using a less sophisticated predictive model (e.g., non-linear terms and interactions were excluded and the disease group variable was dropped). This provides a controlled setting in which we can adequately assess the behavior of our predictive models. We note that obtaining a valid SPS score was important because it was the most important prognostic factor in the SUPPORT survival model.

After excluding an individual missing SPS score, and one individual missing all covariates, 9103 individuals remained of which 3842 had complete data, 2323 were missing partial pressure of oxygen in the arterial blood, 212 were missing white blood count, 3370 were missing albumin, 2599 were missing bilirubin, and 66 were missing creatinine, resulting in 23 observed missing data patterns, and 1024 possible missing data patterns. Ten-fold cross-validation was used to compare the squared error loss of MI, CCSM, MIMI and PMKS within missing data patterns, as well as total average squared error loss, weighted by proportion of individuals in each pattern.

### 2.6.1 SUPPORT Example Results

For each method, ten-fold cross validation of the prediction models was implemented. For the patterns with less than or equal to $N = (p+1)*2 = 22$ subjects, the complete case submodel was used, and the hybrid PMKS/CCS approach (as described in Section 3.2) was implemented. For the original SUPPORT data, all methods performed similarly both across and within patterns. In our simulation, we saw similar results when data were MAR, giving rise to the possibility that these data also follow a MAR mechanism. To exaggerate the missing data mechanism, we induced a MNARY mechanism by adding 25 units to individuals SPS scores who were missing the covariate partial pressure of oxygen in the arterial blood (pafi). This resulted in a large reduction in PE under PMKS compared to traditional MI methods and CCS, for the patterns in which partial pressure of oxygen in the arterial blood was missing.

The original data results are shown in the two sub-figures in the left of Figure 2.3. The total model PE does not differ between the four methods. When a MNAR mechanism is induced in the support data, as shown in the two left sub-figures, PMKS and MIMI outperform CCS and MI. In the patterns for which partial pressure of oxygen in the arterial blood (pafi) is missing, the benefits of PMKS and MIMI compared to CCS and MI are apparent. For these patterns, both the unweighted PE (Figure 2.3 top-right) and weighted PE (Figure 2.3 bottom-right) show this reduction in PE. The model PE, which is the sum of all the pattern specific contributions to the PE results in approximately a 50% reduction in PE for PMKS/MIMI compared to MI, and a 40% reduction in PE for PMKS/MIMI compared to CCS.

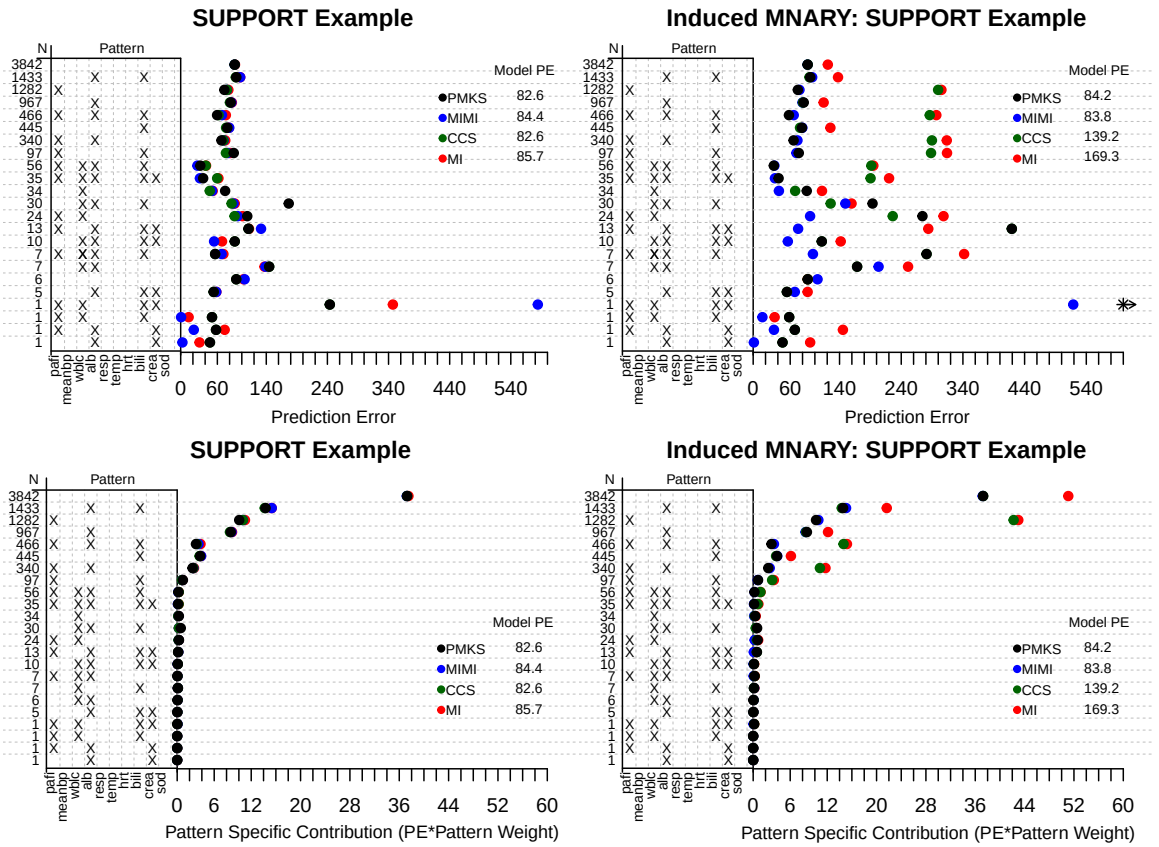



Figure 2.3: The covariates included in the SPS prediction model include Partial pressure of oxygen in the arterial blood (pafi), Mean blood pressure (meanbp), White blood count (wblc), Albumin (alb), APACHE III respiration score (resp), temperature (temp), Heart rate per minute (hrt), Bilirubin (bili), Creatinine (crea), and Sodium (sod). There are 23 patterns present in the SUPPORT data, and missing covariates are denoted with 'X'. $N$ is the total number of subjects in each missing data pattern. Pattern Mixture Kernel Submodels (PMKS), Multiple Imputation with Missingness Indicators (MIMI), Complete Case Submodels (CCS), and traditional Multiple Imputation (MI) methods are all compared. The top two figures are the unweighted pattern specific PE, and the bottom two figures are the pattern specific contribution to the PE in which the partial PE is weighted by the observed proportion of individuals in each pattern.

## 2.7 Remarks

Statistical literature abounds with imputation methods for model inference, but there are very few practical solutions for obtaining predictions for new individuals who do not present with all of the necessary predictors. In this paper, we have shown that PMKS provides competitive if not optimal predictions for a variety of missing data mechanisms, and has large gains in computation time since external data and imputation models are no longer needed to make new predictions. Our method is robust, straightforward to implement, and can easily be extended for any generalized linear model and models with nonlinear effects (this is ongoing work).

While PMKS is clearly optimal in the sense of minimizing an expected prediction loss, the procedure also has implications for model inference under non-MAR scenarios. It leads to an important extension of classical MI procedures for estimation where the MAR can be relaxed or assessed. Moreover, PMKS is more computationally efficient thanMI procedures. In the age of big data, this is an important consideration and driving factor in most scientific contexts with big data.

### 2.7.1 Remark A: Conditioning Y on X and M

One might ask whether we are interested in the model marginalized over $\boldsymbol{M}$, $E[Y|\boldsymbol{X}] = \boldsymbol{X}\boldsymbol{\beta}$, or the conditional model, $E[Y|\boldsymbol{X}, \boldsymbol{M}] = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{M}\boldsymbol{\delta}$. This is a philosophical question with many differing viewpoints. For inferential purposes (which we do not consider here) it has been argued that the marginal model is the model of interest, however in many situations can be imagined in which the conditional model is the simpler way to express a complicated marginal model. From the prediction point of view, PMKS/MIMI model is robust to both situations and therefore is the preferred method to use.

By assuming the marginal model is correct, one is making the assumption that data are MAR and the $\boldsymbol{\delta}$ parameters are zero. As the number of covariates increase, this assumption in practice seems to be more plausible. A way to asses this in practice is to evaluate whether the MI model and PMKS/MIMI model give similar results, as we did in the SUPPORT example.

### 2.7.2 Remark B: The Relationship between $Y$ and $M$

The relationship between $Y$ and $M$ plays an important role in our modeling assumptions. An outcome generated from a selection model formulation is assumed to be independent of the missing data mechanism, such that the outcome would be the same regardless of whether covariate information is missing or observed. The pattern mixture model formulation assumes that the missing data mechanism is part of the response model, such that the outcome can be depend on the missing data pattern. Both mechanisms are thought to exist in biomedical data, but unfortunately the two approaches represent fundamentally different descriptions of the underlying natural process. The two approach will only coincide when $\delta_1 = ... = \delta_p = 0$ (the MAR assumption). The $\boldsymbol{\beta}$ parameters could be adjusted to account for the differential missingness across the two approaches, but then the model are not comparable.

### 2.7.3 Remark C: Extending to Generalized Linear Models and Other Prediction Approaches

We performed the same set of simulations assuming a true logistic regression model, where we used a logarithmic scoring rule to compare methods. The general ordering of results holds and will be explored in future papers. These results extend to random forests as well. For example, our results suggest a random forest should be fit by pattern, using only the data in that pattern.

### 2.7.4 Final remark

Care should be taken when developing clinical prediction models when missing data is present. Prediction models should be fit with PMKS and estimation should be conducted under MIMI for optimal results, since these methods are robust to a wide variety of missing data mechanisms compared to commonly available MI and CCS methods.

## 2.8 Appendix A. Code for Simulations and Case Study

```
library(plyr)
library(rms)
library(mice)
library(cvTools)
#library(caret)
library(MASS)
library(Hmisc)
library(lme4)
library(mi)
library(doSNOW)

########################
# Functions to use
########################


#This function creates missing data indicators
#for your dataset and then appends those indicators
#To the original dataset

create.miss.ind <- function(DATA){
 tmp.dat <- as.data.frame(is.na(DATA)*1)
 names(tmp.dat) <- paste('m.',names(tmp.dat),sep="")
    cbind(DATA,tmp.dat)
}

#This function creates missing data indicators
#for your dataset

create.miss.ind.only <- function(DATA){
 tmp.dat <- as.data.frame(is.na(DATA)*1)
 names(tmp.dat) <- paste('m.',names(tmp.dat),sep="")
    tmp.dat
}



r

#Function to combine two data frames and match them up by column names
rbind.match.columns <- function(input1, input2) {
    n.input1 <- ncol(input1)
    n.input2 <- ncol(input2)
    if (n.input2 < n.input1) {
        TF.names <- which(names(input2) %in% names(input1))
        column.names <- names(input2[, TF.names])
    } else {
        TF.names <- which(names(input1) %in% names(input2))
        column.names <- names(input1[, TF.names])
    }
    return(rbind(input1[, column.names], input2[, column.names]))
}

#predict an out of sample individual where you have a single model created
#by mice, takes the mice model object and the newdata from a new individual
predict.oos.mice <- function(FIT, NEWDATA){
mod.call <- as.character(FIT$call1)[3]
mod.call <- strsplit(mod.call, '\\(')[[1]][2]
mod.call <- strsplit(mod.call, ',')[[1]][1]
lp <- NA
bb <- FIT$qbar
mm <- model.matrix(as.formula(mod.call), NEWDATA)
if(length(bb) != ncol(mm)) {
stop('issue with mm')
} else {
lp <- mm %*% matrix(bb, ncol=1)
}
lp
}

#AUC function
auc=function(score,status){
######################
```

```
## auc version 1.0
## Compute Area under Rmpirical ROC curve by Trapezoidal Rule
## Author: J. Blume
## Date:   July 2014
######################

pos=score[status==1]
neg=score[status==0]

ct1=sum(outer(pos,neg,">"))
ct2=sum(outer(pos,neg,"=="))
den=length(pos)*length(neg)
auc=(ct1+0.5*ct2)/den
auc=max(auc,1-auc)
auc

}


#Function to calculate the brier score
brier.score <- function(pred, outcome){
mean((pred - outcome)^2)
}

#Function to calculate the logarithmic scoring rule
logarithmic.scoring.rule <- function(pred, outcome){

mean(outcome*log(pred) + (1-outcome)*log(1-pred))

}

############
#1) Find all the patterns of missingness
#2) For patterns of missingness with membership >= 2p + 2, fit a submodel with all the
#   observed covariates, ussing only the people in that pattern.
#3) For patterns of missingness with mempership < 2p + 2 fit the complete case submodel, which
#   only includes the observed covariates in that patter, but if fitted with ALL the
#   individuals in the data set
#4) Allow function to handle both linear and logistic models.
#5) Output prediction models in a way that would be usable with a validation
#   dataset or new person
#####!!!!!!!!!!!!!!!For all unobserved patterns do the complete case submodel


pmks <- function(DATA, model, logistic=TRUE){
    mod.DATA        <- get_all_vars(as.formula(model), data=DATA)
    SDATA   <- mod.DATA[,-1] #remove the outcome
    tmp.dat      <- as.data.frame(is.na(SDATA)*1)
tmp.pattern  <- factor(apply(tmp.dat,1,function(z) paste(z,collapse="")))
all.patterns <- factor(apply(expand.grid(rep(list(0:1),ncol(SDATA))),1,function(z) paste(z,collapse="")))
obs.patterns <- unique(tmp.pattern)
tmp.info     <- split(seq(nrow(SDATA)), tmp.pattern)
mp.levels    <- levels(tmp.pattern)
mp.pattern   <- do.call(rbind, lapply(as.list(mp.levels),function(ZZ) strsplit(ZZ,'')[[1]]))
mp.info     <- data.frame(cbind(names(tmp.info), unlist(lapply(tmp.info, length))),
                          stringsAsFactors= FALSE)
rownames(mp.info) <- seq(nrow(mp.info))
colnames(mp.info) <- c('mp','n')
if(length(setdiff(all.patterns,obs.patterns)) == 0){
  empty.patterns = NULL
} else {
empty.patterns <- data.frame(mp = factor(setdiff(all.patterns,obs.patterns)), n=0)
}
mp.info <- rbind(mp.info,empty.patterns)

mod.rhs <- strsplit(model, '~')[[1]][1]
mod.lhs <- strsplit(model, '~')[[1]][2]
mod.lhs <- strsplit(mod.lhs,' ')[[1]]
mod.lhs <- mod.lhs[!(mod.lhs%in%c('','+','~'))]

cc <- ncol(model.matrix(as.formula(model),mod.DATA))

threshold <- cc*2
mp.info$use.ptmx <- (as.numeric(mp.info$n)>=threshold)*1
```

```
reg.out <- vector('list', length(all.patterns))
names(reg.out) <- mp.info$mp.info


for(ixx in seq(nrow(mp.info))) {
col.keep  <- which(strsplit(mp.info$mp[ixx],'')[[1]]=='0')
if(length(col.keep)==0){
new.mod <- as.formula(paste(mod.rhs,1,sep='~'))
} else {new.mod   <- as.formula(paste(mod.rhs,paste(mod.lhs[col.keep],collapse='+'),
                     sep='~'))}

if(mp.info$use.ptmx[ixx]==1) {
if(logistic == TRUE){
reg.out[[ixx]] <- list(pattern = mod.lhs[col.keep],
mod = glm(new.mod,data=mod.DATA[tmp.info[[ixx]],],family = 'binomial'))
} else {
  reg.out[[ixx]] <- list(pattern = mod.lhs[col.keep],
                         mod = glm(new.mod,data=mod.DATA[tmp.info[[ixx]],],family = 'gaussian'))
}
} else {
  if(logistic == TRUE){
  reg.out[[ixx]] <- list(pattern = mod.lhs[col.keep],
    mod  = glm(new.mod,data=mod.DATA, family='binomial'))
  } else {
  reg.out[[ixx]] <- list(pattern = mod.lhs[col.keep],
                           mod  = glm(new.mod,data=mod.DATA, family='gaussian'))
  }
}
}
reg.out
}


#Fit all the complete case submodels
ccsm <- function(DATA, model, logistic=TRUE){
    mod.DATA        <- get_all_vars(as.formula(model), data=DATA)
    SDATA   <- mod.DATA[,-1] #remove the outcome
    tmp.dat      <- as.data.frame(is.na(SDATA)*1)
tmp.pattern  <- factor(apply(tmp.dat,1,function(z) paste(z,collapse="")))
all.patterns <- factor(apply(expand.grid(rep(list(0:1),ncol(SDATA))),1,function(z) paste(z,collapse="")))
obs.patterns <- unique(tmp.pattern)
tmp.info     <- split(seq(nrow(SDATA)), tmp.pattern)
mp.levels    <- levels(tmp.pattern)
mp.pattern   <- do.call(rbind, lapply(as.list(mp.levels),function(ZZ) strsplit(ZZ,'')[[1]]))
mp.info     <- data.frame(cbind(names(tmp.info), unlist(lapply(tmp.info, length))),
                          stringsAsFactors= FALSE)
rownames(mp.info) <- seq(nrow(mp.info))
colnames(mp.info) <- c('mp','n')
empty.patterns <- data.frame(mp = factor(setdiff(all.patterns,obs.patterns)), n=0)
mp.info <- rbind(mp.info,empty.patterns)

mod.rhs <- strsplit(model, '~')[[1]][1]
mod.lhs <- strsplit(model, '~')[[1]][2]
mod.lhs <- strsplit(mod.lhs,' ')[[1]]
mod.lhs <- mod.lhs[!(mod.lhs%in%c('','+','~'))]

reg.out <- vector('list', length(all.patterns))
names(reg.out) <- mp.info$mp.info

for(ixx in seq(nrow(mp.info))) {
col.keep  <- which(strsplit(mp.info$mp[ixx],'')[[1]]=='0')
if(length(col.keep)==0){
new.mod <- as.formula(paste(mod.rhs,1,sep='~'))
} else {new.mod   <- as.formula(paste(mod.rhs,paste(mod.lhs[col.keep],collapse='+'),
                     sep='~'))}
if(logistic == TRUE){
   # Use complete case submodel
  reg.out[[ixx]] <- list(pattern = mod.lhs[col.keep],
    mod  = glm(new.mod, data=mod.DATA, family = 'binomial'))
} else {reg.out[[ixx]] <- list(pattern = mod.lhs[col.keep],
                               mod  = glm(new.mod, data=mod.DATA, family = 'gaussian'))}
}
reg.out
```

```
}


#Predict for a new individual using a PMKS or CCSM object.
predict.sm <- function(prediction.data, model, pmks.object, logistic = TRUE){
mod.DATA     <- get_all_vars(as.formula(model), data=prediction.data)
  PDATA   <- mod.DATA[,-1] #remove the outcome
tmp.dat      <- as.data.frame(is.na(PDATA)*1)
tmp.pattern  <- factor(apply(tmp.dat,1,function(z) paste(z,collapse="")))
tmp.info     <- split(seq(nrow(PDATA)), tmp.pattern)
mp.levels    <- levels(tmp.pattern)
mp.pattern   <- do.call(rbind, lapply(as.list(mp.levels),function(ZZ) strsplit(ZZ,'')[[1]]))
mp.info     <- data.frame(cbind(names(tmp.info), unlist(lapply(tmp.info, length))),
                          stringsAsFactors= FALSE)
rownames(mp.info) <- seq(nrow(mp.info))
colnames(mp.info) <- c('mp','n')
  mod.rhs <- strsplit(model, '~')[[1]][1]
  mod.rhs <- strsplit(mod.rhs,' ')[[1]]
mod.lhs <- strsplit(model, '~')[[1]][2]
mod.lhs <- strsplit(mod.lhs,' ')[[1]]
mod.lhs <- mod.lhs[!(mod.lhs%in%c('','+','~'))]
pred.out <- vector('list', nrow(mp.info))
 #For the different patterns
for(ixx in seq(length(tmp.info))){
col.keep  <- which(strsplit(mp.info$mp[ixx],'')[[1]]=='0')
pattern   <- mod.lhs[col.keep]
which.mod <- which(lapply(pmks.object, function(z) identical(z$pattern, pattern))==TRUE)


if(logistic == TRUE){

      pred.out[[ixx]] <- list(
                    numeric.pattern = mp.info$mp[ixx],

                    pattern = pattern,

                    mod = pmks.object[[which.mod]]$mod,

         lin.pred = predict(pmks.object[[which.mod]]$mod,
            prediction.data[tmp.info[[ixx]],]),
         truth = prediction.data[tmp.info[[ixx]],mod.rhs],
         strat.auc = auc(expit(predict(pmks.object[[which.mod]]$mod,
            prediction.data[tmp.info[[ixx]],])),
            prediction.data[tmp.info[[ixx]],mod.rhs]),
         strat.brier =  brier.score(expit(predict(pmks.object[[which.mod]]$mod,
            prediction.data[tmp.info[[ixx]],])),
            prediction.data[tmp.info[[ixx]],mod.rhs]),
         strat.logscore =  logarithmic.scoring.rule(expit(predict(pmks.object[[which.mod]]$mod,
                                        prediction.data[tmp.info[[ixx]],])),
                                  prediction.data[tmp.info[[ixx]],mod.rhs]))
} else {
  pred.out[[ixx]] <- list(
    numeric.pattern = mp.info$mp[ixx],

    pattern = pattern,

    mod = pmks.object[[which.mod]]$mod,

    lin.pred = predict(pmks.object[[which.mod]]$mod,
                       prediction.data[tmp.info[[ixx]],]),
    truth = prediction.data[tmp.info[[ixx]],mod.rhs],

    strat.brier =  brier.score(predict(pmks.object[[which.mod]]$mod,
                                       prediction.data[tmp.info[[ixx]],]),
                               prediction.data[tmp.info[[ixx]],mod.rhs]))
}
}
pred.out
}


#Function to find all the observed missing data patterns
which.pattern <- function(DATA, model){
```

```
  mod.DATA        <- get_all_vars(as.formula(model), data=DATA)
  SDATA        <- mod.DATA[,-1] #remove the outcome
  tmp.dat         <- as.data.frame(is.na(SDATA)*1)
  tmp.pattern     <- factor(apply(tmp.dat,1,function(z) paste(z,collapse="")))

   pattern <- tmp.pattern
   pattern
}


#Get metrics by pattern for a logistic model
get_metric <- function(DATA) {
  colnames(DATA) <- c('pred','true','pattern')
  do.call(rbind, by(DATA, pattern,function(xx) {
    data.frame('log'=logarithmic.scoring.rule(xx$pred,xx$true),
               'auc'=auc(xx$pred,xx$true),
               'brier'=brier.score(xx$pred,xx$true))}))
}

#Get metrics by pattern for a linear model
get_metric_linear <- function(DATA) {
  colnames(DATA) <- c('pred','true','pattern')
  do.call(rbind, lapply( split(DATA, DATA$pattern),
                         function(xx) data.frame('brier'=brier.score(xx$pred,xx$true),
                                                 'prop.pattern' = (length(xx$true)/nrow(DATA))) ))
}

#MSE from a fitted model
mse <- function(sm) {
  mse <- mean(sm$residuals^2)
  return(mse)
}

#logit function
logit <- function(p) log(p/(1-p))


#Empirically calculate the intercept for the missing data
#mechanism
miss.int <- function(miss.mech, betaM=1,p.miss){
  if(ncol(miss.mech)!=2){
    logit(p.miss) - betaM*mean(miss.mech[,1])
  } else {
    logit(p.miss) - betaM*mean((((miss.mech[,2]/sd(miss.mech[,2]))
                                + miss.mech[,1])/as.numeric(sqrt(2*(1+cor(miss.mech[,2],miss.mech[,1]))))))
  }
}

#Function to create a data frame for simulation
dat.frame <- function(n.tr,
                      mu.z,
                      mu.w,
                      mu.q,
                      s.z,
                      s.w,
                      s.q,
                      s.wz,
                      s.zq,
                      s.wq,
                      p.miss,
                      missing.type,
                      b.x,
                      b.z,
                      b.w,
                      b.m,
                      mu.e,
                      s.e,
                      betaM=1){
  x = rep(1,n.tr)
  mu = c(mu.z,mu.w,mu.q)
  Sigma = matrix(c(1,s.wz,s.zq,
                   s.wz,1,s.wq,
                   s.zq,s.wq,1), byrow=TRUE, ncol=3)
  dat =  as.data.frame(mvrnorm(n=n.tr, mu=mu, Sigma=Sigma, empirical=TRUE))
```

```
  colnames(dat) = c('z','w','q')
  coef.tru = rbind( b.x , b.z , b.w)
  dat$y=cbind(x,dat[,'z'], dat[,'w'])%*%coef.tru + rnorm(n.tr,mean=mu.e,sd=s.e)

  if(missing.type=='MCAR'){
    m = rbinom(n.tr,1,p.miss)
    y.pmm = b.x*x + b.z*dat[,'z'] +  b.w*dat[,'w'] + b.m*m + rnorm(n.tr,mean=mu.e,sd=s.e)
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], q=dat[,'q'],m=m, y=dat[,'y'], y.pmm = y.pmm)
  } else if(missing.type=='MAR'){
    m = rbinom(n.tr, 1, expit(miss.int(miss.mech=cbind(dat[,'w']),
                                       betaM=betaM,p.miss) + betaM*dat[,'w']))
    y.pmm = b.x*x + b.z*dat[,'z'] +  b.w*dat[,'w'] + b.m*m + rnorm(n.tr,mean=mu.e,sd=s.e)
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], q=dat[,'q'], m=m, y=dat[,'y'], y.pmm = y.pmm)
  } else if(missing.type=='MNAR'){
    m = rbinom(n.tr, 1, expit(miss.int(miss.mech=cbind(dat[,'z']),
                                       betaM=betaM,p.miss) + betaM*dat[,'z']))
    y.pmm = b.x*x + b.z*dat[,'z'] +  b.w*dat[,'w'] + b.m*m + rnorm(n.tr,mean=mu.e,sd=s.e)
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], q=dat[,'q'], m=m, y=dat[,'y'], y.pmm = y.pmm)
  } else if(missing.type=='MARY'){
    m = rbinom(n.tr, 1,
               expit(miss.int(miss.mech=dat[,c('w','y')], betaM=betaM,p.miss)
                     + betaM*(((dat[,'y']/sd(dat[,'y'])) + dat[,'w'])/as.numeric(sqrt(2*(1+cor(dat[,'y'],dat[,'w']))))))
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], q=dat[,'q'], m=m, y=dat[,'y'])
  } else if(missing.type=='MNARY'){
    m = rbinom(n.tr, 1,
               expit(miss.int(miss.mech=dat[,c('z','y')], betaM=betaM,p.miss)
                     + betaM*(((dat[,'y']/sd(dat[,'y'])) + dat[,'z'])/as.numeric(sqrt(2*(1+cor(dat[,'y'],dat[,'z']))))))
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'],  q=dat[,'q'], m=m, y=dat[,'y'])
  }

}



#Create an out of sample data frame, returns a list
# with different kinds of imputations done on the
# out of sample individuals
oos.data <- function(n.tr.new,
                     mu.z.new,
                     mu.w.new,
                     mu.q.new,
                     s.z.new,
                     s.w.new,
                     s.q.new,
                     s.wz.new,
                     s.zq.new,
                     s.wq.new,
                     p.miss.new,
                     missing.type.new,
                     b.x.new,
                     b.z.new,
                     b.w.new,
                     mu.e.new,
                     s.e.new,
                     betaM.new=1,
                     n.imp = 5,
                     original.dat.miss,
                     Y.PM
) {

  dat.new <- dat.frame(n.tr=n.tr.new,
                       mu.z = mu.z.new,
                       mu.w = mu.w.new,
                       mu.q = mu.q.new,
                       s.z = s.z.new,
                       s.w = s.w.new,
                       s.q = s.q.new,
                       s.wz = s.wz.new,
                       s.zq = s.zq.new,
                       s.wq = s.wq.new,
                       p.miss = p.miss.new,
                       missing.type = missing.type.new,
                       b.x = b.x.new,
                       b.z = b.z.new,
```

```
                    b.w = b.w.new,
                    b.m = betaM.new,
                    mu.e = mu.e.new,
                    s.e = s.e.new,
                    betaM = betaM.new)
if(Y.PM == TRUE) {dat.new$y <- dat.new$y.pmm}

dat.miss.new <- dat.new
dat.miss.new$z <- ifelse(dat.new$m==1,NA,dat.new$z)

imputed.dat.cm.mipack <- imputed.dat.cm.Hmisc <- dat.miss.new
imputed.dat.mi <- imputed.dat.miq <- imputed.dat.miy <- rep(list(dat.miss.new),n.imp)

dat.cc.mean <- dat.miss.new
dat.cc.mean[which(is.na(dat.cc.mean$z)==TRUE),'z'] <- mean(original.dat.miss$z, na.rm=TRUE)

dat.cond.mean <- dat.miss.new
mod.predict.z <- lm(z ~ w, data = original.dat.miss)
dat.cond.mean[dat.cond.mean$m==1,'z'] <- predict(mod.predict.z, dat.miss.new[dat.miss.new$m==1,])

for(i in which(dat.new$m==1)){
  newperson <- dat.miss.new[i,]
  newperson$y <- NA
  addNewPatient <- rbind.match.columns(original.dat.miss, newperson)

  #Single Conditional Mean Imputation Using MI
  mdf <- suppressMessages(suppressWarnings(missing_data.frame(addNewPatient[,c('x','z','w')])))

  #the expectation argument is conditional mean expectation
  mdf <-  suppressMessages(change(mdf, y=c('x','z','w'),what = "imputation_method", to = "expectation"))
  imputations <- mi(mdf,n.iter=1,n.chains=1,verbose=FALSE,parallel=FALSE)

  imputed.dat.cm.mipack[i,c('x','z','w')] <- complete(imputations,1)[nrow(addNewPatient),c('x','z','w')]

  f <- aregImpute( ~ w + z, data=addNewPatient, n.impute=1,type='regression',pr=FALSE)
  imputed.dat.cm.Hmisc[i,c('z','w')] <- as.data.frame(impute.transcan(f,
                                                                      imputation=1, data=addNewPatient, list.out=TRUE,
                                                                      pr=FALSE, check=FALSE))[nrow(addNewPatient),c('z','w')]

  # imp.pmm  <- aregImpute(~ w + z, n.impute=n.imp, x=TRUE,
  #                        nk=3, tlinear=F, data=addNewPatient, pr=FALSE)
  # imp.pmm.q  <- aregImpute(~ w + z + q, n.impute=n.imp, x=TRUE,
  #                          nk=3, tlinear=F, data=addNewPatient, pr=FALSE)
  # imp.pmm.y  <- aregImpute(~  w + z + y, n.impute=n.imp, x=TRUE,
  #                          nk=3, tlinear=F, data=addNewPatient, pr=FALSE)


  imp.pmm  <- aregImpute(~ w + z, n.impute=n.imp, x=TRUE,
                         nk=0, tlinear=TRUE, data=addNewPatient, pr=FALSE)
  imp.pmm.q  <- aregImpute(~ w + z + q, n.impute=n.imp, x=TRUE,
                           nk=0, tlinear=TRUE, data=addNewPatient, pr=FALSE)
  imp.pmm.y  <- aregImpute(~  w + z + y, n.impute=n.imp, x=TRUE,
                           nk=0, tlinear=TRUE, data=addNewPatient, pr=FALSE)

  for(j in 1:n.imp) {
    imputed.dat.mi[[j]][i,c('z')] <- as.data.frame(impute.transcan(imp.pmm, imputation=j,
                                                                   data=addNewPatient, list.out=TRUE,
                                                                   pr=FALSE, check=FALSE))[nrow(addNewPatient),c('z')]

    imputed.dat.miq[[j]][i,c('z')] <- as.data.frame(impute.transcan(imp.pmm.q, imputation=j,
                                                                    data=addNewPatient, list.out=TRUE,
                                                                    pr=FALSE, check=FALSE))[nrow(addNewPatient),c('z')]

    imputed.dat.miy[[j]][i,c('z')] <- as.data.frame(impute.transcan(imp.pmm.y, imputation=j,
                                                                    data=addNewPatient, list.out=TRUE,
                                                                    pr=FALSE, check=FALSE))[nrow(addNewPatient),c('z')]
  }

}

list(
  dat.miss.new = dat.miss.new,
  imputed.dat.cm.mipack = imputed.dat.cm.mipack,
  imputed.dat.cm.Hmisc = imputed.dat.cm.Hmisc,
```

```
    imputed.dat.mi.avg = Reduce("+",imputed.dat.mi)/length(imputed.dat.mi),
    imputed.dat.miq.avg = Reduce("+",imputed.dat.miq)/length(imputed.dat.miq),
    imputed.dat.miy.avg = Reduce("+",imputed.dat.miy)/length(imputed.dat.miy),
    dat.new = dat.new,
    dat.mean = dat.cc.mean,
    dat.cond.mean =  dat.cond.mean
  )


}

#Simulation
simulation <- function(n.tr,
                       mu.z,
                       mu.w,
                       mu.q,
                       s.z,
                       s.w,
                       s.q,
                       s.wz,
                       s.zq,
                       s.wq,
                       p.miss,
                       missing.type,
                       b.x,
                       b.z,
                       b.w,
                       mu.e,
                       s.e,
                       betaM,
                       n.tr.new,
                       mu.z.new,
                       mu.w.new,
                       mu.q.new,
                       s.z.new,
                       s.w.new,
                       s.q.new,
                       s.wz.new,
                       s.zq.new,
                       s.wq.new,
                       p.miss.new,
                       missing.type.new,
                       b.x.new,
                       b.z.new,
                       b.w.new,
                       mu.e.new,
                       s.e.new,
                       betaM.new,
                       n.imp,
                       n.sim,
                       Y.PM
                       ){
results = list(is.mse = data.frame(mse.truth = rep(NA,n.sim), mse.truth.int = rep(NA,n.sim),
                                   mse.cc = rep(NA,n.sim), mse.full.cm.mipack = rep(NA,n.sim),
                                   mse.full.cm.Hmisc = rep(NA, n.sim), mse.marg.cm.mipack = rep(NA, n.sim),
                                   mse.marg.cm.Hmisc = rep(NA, n.sim), mse.full.mi.Hmisc = rep(NA,n.sim),
                                   mse.marg.mi.Hmisc = rep(NA, n.sim), mse.full.miy.Hmisc = rep(NA, n.sim),
                                   mse.marg.miy.Hmisc = rep(NA, n.sim), mse.full.miq.Hmisc = rep(NA, n.sim),
                                   mse.marg.miq.Hmisc = rep(NA, n.sim), mse.full.miyq.Hmisc = rep(NA, n.sim),
                                   mse.marg.miyq.Hmisc = rep(NA, n.sim),
                                   mse.full.cond.mean = rep(NA, n.sim), mse.marg.cond.mean = rep(NA, n.sim)),
              is.pattern.00 = data.frame(oracle.marg.mse = rep(NA, n.sim), oracle.marg.prop = rep(NA, n.sim),
                                         oracle.full.mse = rep(NA, n.sim), oracle.full.prop = rep(NA, n.sim),
                                         cc.mean.mse = rep(NA, n.sim),cc.mean.prop = rep(NA, n.sim),
                                         full.cm.mipack.mse = rep(NA, n.sim),full.cm.mipack.prop = rep(NA, n.sim),
                                         marg.cm.mipack.mse = rep(NA, n.sim),marg.cm.mipack.prop = rep(NA, n.sim),
                                         full.cm.Hmisc.mse = rep(NA, n.sim), full.cm.Hmisc.prop = rep(NA, n.sim),
                                         marg.cm.Hmisc.mse = rep(NA, n.sim),marg.cm.Hmisc.prop = rep(NA, n.sim),
                                         full.mi.Hmisc.mse = rep(NA, n.sim),full.mi.Hmisc.prop = rep(NA, n.sim),
                                         marg.mi.Hmisc.mse = rep(NA, n.sim),marg.mi.Hmisc.prop = rep(NA, n.sim),
                                         full.miq.Hmisc.mse = rep(NA, n.sim),full.miq.Hmisc.prop = rep(NA, n.sim),
                                         marg.miq.Hmisc.mse = rep(NA, n.sim),marg.miq.Hmisc.prop = rep(NA, n.sim),
                                         full.miy.Hmisc.mse = rep(NA, n.sim),full.miy.Hmisc.prop = rep(NA, n.sim),
                                         marg.miy.Hmisc.mse = rep(NA, n.sim) ,marg.miy.Hmisc.prop = rep(NA, n.sim) ,
```

```
                                metric.pmks.mse = rep(NA, n.sim),metric.pmks.prop = rep(NA, n.sim),
                                metric.ccsm.mse = rep(NA, n.sim),metric.ccsm.prop = rep(NA, n.sim),
                               full.cond.mean = rep(NA, n.sim), full.cond.mean.prop = rep(NA, n.sim),
                               marg.cond.mean = rep(NA,n.sim), marg.cond.mean = rep(NA, n.sim)),
    is.pattern.10 = data.frame(oracle.marg.mse = rep(NA, n.sim), oracle.marg.prop = rep(NA, n.sim),
                               oracle.full.mse = rep(NA, n.sim), oracle.full.prop = rep(NA, n.sim),
                               cc.mean.mse = rep(NA, n.sim),cc.mean.prop = rep(NA, n.sim),
                               full.cm.mipack.mse = rep(NA, n.sim),full.cm.mipack.prop = rep(NA, n.sim),
                               marg.cm.mipack.mse = rep(NA, n.sim),marg.cm.mipack.prop = rep(NA, n.sim),
                               full.cm.Hmisc.mse = rep(NA, n.sim), full.cm.Hmisc.prop = rep(NA, n.sim),
                               marg.cm.Hmisc.mse = rep(NA, n.sim),marg.cm.Hmisc.prop = rep(NA, n.sim),
                               full.mi.Hmisc.mse = rep(NA, n.sim),full.mi.Hmisc.prop = rep(NA, n.sim),
                               marg.mi.Hmisc.mse = rep(NA, n.sim),marg.mi.Hmisc.prop = rep(NA, n.sim),
                               full.miq.Hmisc.mse = rep(NA, n.sim),full.miq.Hmisc.prop = rep(NA, n.sim),
                               marg.miq.Hmisc.mse = rep(NA, n.sim),marg.miq.Hmisc.prop = rep(NA, n.sim),
                               full.miy.Hmisc.mse = rep(NA, n.sim),full.miy.Hmisc.prop = rep(NA, n.sim),
                               marg.miy.Hmisc.mse = rep(NA, n.sim), marg.miy.Hmisc.prop = rep(NA, n.sim),
                               metric.pmks.mse = rep(NA, n.sim),metric.pmks.prop = rep(NA, n.sim),
                               metric.ccsm.mse = rep(NA, n.sim),metric.ccsm.prop = rep(NA, n.sim),
                               full.cond.mean = rep(NA, n.sim), full.cond.mean.prop = rep(NA, n.sim),
                               marg.cond.mean = rep(NA,n.sim), marg.cond.mean.prop = rep(NA, n.sim)),
    oos.mse = data.frame(mse.cc.mean.new= rep(NA,n.sim),
                         mse.cc.cm.Hmisc.new= rep(NA,n.sim),
                         mse.full.cm.mipack.new= rep(NA,n.sim),
                         mse.marg.cm.mipack.new= rep(NA,n.sim),
                         mse.full.cm.Hmisc.new= rep(NA,n.sim),
                         mse.marg.cm.Hmisc.new= rep(NA,n.sim),
                         mse.full.mi.Hmisc.new= rep(NA,n.sim),
                         mse.marg.mi.Hmisc.new= rep(NA,n.sim),
                         mse.full.miq.Hmisc.new= rep(NA,n.sim),
                         mse.marg.miq.Hmisc.new= rep(NA,n.sim),
                         mse.full.miy.Hmisc.new= rep(NA,n.sim),
                         mse.marg.miy.Hmisc.new= rep(NA,n.sim),
                         oracle.marg= rep(NA,n.sim),
                         oracle.full= rep(NA,n.sim),
                         mse.full.cond.mean = rep(NA,n.sim),
                         mse.marg.cond.mean = rep(NA, n.sim)
                        ),
    oos.pattern.00 = data.frame(oracle.marg.mse = rep(NA, n.sim), oracle.marg.prop = rep(NA, n.sim),
                                oracle.full.mse = rep(NA, n.sim), oracle.full.prop = rep(NA, n.sim),
                                cc.mean.mse = rep(NA, n.sim),cc.mean.prop = rep(NA, n.sim),
                                cc.cm.mipack.mse = rep(NA, n.sim),cc.cm.mipack.prop = rep(NA, n.sim),
                                cc.cm.Hmisc.mse = rep(NA, n.sim),cc.cm.Hmisc.prop = rep(NA, n.sim),
                                full.cm.mipack.mse = rep(NA, n.sim),full.cm.mipack.prop = rep(NA, n.sim),
                                marg.cm.mipack.mse = rep(NA, n.sim),marg.cm.mipack.prop = rep(NA, n.sim),
                                full.cm.Hmisc.mse = rep(NA, n.sim), full.cm.Hmisc.prop = rep(NA, n.sim),
                                marg.cm.Hmisc.mse = rep(NA, n.sim),marg.cm.Hmisc.prop = rep(NA, n.sim),
                                full.mi.Hmisc.mse = rep(NA, n.sim),full.mi.Hmisc.prop = rep(NA, n.sim),
                                marg.mi.Hmisc.mse = rep(NA, n.sim),marg.mi.Hmisc.prop = rep(NA, n.sim),
                                full.miq.Hmisc.mse = rep(NA, n.sim),full.miq.Hmisc.prop = rep(NA, n.sim),
                                marg.miq.Hmisc.mse = rep(NA, n.sim),marg.miq.Hmisc.prop = rep(NA, n.sim),
                                full.miy.Hmisc.mse = rep(NA, n.sim),full.miy.Hmisc.prop = rep(NA, n.sim),
                                marg.miy.Hmisc.mse = rep(NA, n.sim) ,marg.miy.Hmisc.prop = rep(NA, n.sim) ,
                               metric.pmks.mse = rep(NA, n.sim),metric.pmks.prop = rep(NA, n.sim),
                                metric.ccsm.mse = rep(NA, n.sim),metric.ccsm.prop = rep(NA, n.sim),
                               full.cond.mean = rep(NA, n.sim), full.cond.mean.prop = rep(NA, n.sim),
                               marg.cond.mean = rep(NA,n.sim), marg.cond.mean.prop = rep(NA, n.sim)),
    oos.pattern.10 = data.frame(oracle.marg.mse = rep(NA, n.sim), oracle.marg.prop = rep(NA, n.sim),
                                oracle.full.mse = rep(NA, n.sim), oracle.full.prop = rep(NA, n.sim),
                                cc.mean.mse = rep(NA, n.sim),cc.mean.prop = rep(NA, n.sim),
                                cc.cm.mipack.mse = rep(NA, n.sim),cc.cm.mipack.prop = rep(NA, n.sim),
                                cc.cm.Hmisc.mse = rep(NA, n.sim),cc.cm.Hmisc.prop = rep(NA, n.sim),
                                full.cm.mipack.mse = rep(NA, n.sim),full.cm.mipack.prop = rep(NA, n.sim),
                                marg.cm.mipack.mse = rep(NA, n.sim),marg.cm.mipack.prop = rep(NA, n.sim),
                                full.cm.Hmisc.mse = rep(NA, n.sim), full.cm.Hmisc.prop = rep(NA, n.sim),
                                marg.cm.Hmisc.mse = rep(NA, n.sim),marg.cm.Hmisc.prop = rep(NA, n.sim),
                                full.mi.Hmisc.mse = rep(NA, n.sim),full.mi.Hmisc.prop = rep(NA, n.sim),
                                marg.mi.Hmisc.mse = rep(NA, n.sim),marg.mi.Hmisc.prop = rep(NA, n.sim),
                                full.miq.Hmisc.mse = rep(NA, n.sim),full.miq.Hmisc.prop = rep(NA, n.sim),
                                marg.miq.Hmisc.mse = rep(NA, n.sim),marg.miq.Hmisc.prop = rep(NA, n.sim),
                                full.miy.Hmisc.mse = rep(NA, n.sim),full.miy.Hmisc.prop = rep(NA, n.sim),
                                marg.miy.Hmisc.mse = rep(NA, n.sim), marg.miy.Hmisc.prop = rep(NA, n.sim),
                                metric.pmks.mse = rep(NA, n.sim),metric.pmks.prop = rep(NA, n.sim),
                                metric.ccsm.mse = rep(NA, n.sim),metric.ccsm.prop = rep(NA, n.sim),
```

```
                                    full.cond.mean = rep(NA, n.sim), full.cond.mean.prop = rep(NA, n.sim),
                                    marg.cond.mean = rep(NA,n.sim), marg.cond.mean.prop = rep(NA, n.sim)),
              oos.z.mse = data.frame(dat.miss.new= rep(NA, n.sim),
                                     imputed.dat.cm.mipack = rep(NA, n.sim),
                                     imputed.dat.cm.Hmisc = rep(NA, n.sim),
                                     imputed.dat.mi.avg = rep(NA, n.sim),
                                     imputed.dat.miq.avg = rep(NA, n.sim),
                                     imputed.dat.miy.avg = rep(NA, n.sim),
                                     dat.new = rep(NA, n.sim),
                                     dat.mean = rep(NA, n.sim),
                                     dat.cond.mean = rep(NA, n.sim)     )
              )
#Replicate Simulation
for(RS in 1:n.sim){
  dat = dat.frame(n.tr=n.tr,
                  mu.z = mu.z,
                  mu.w = mu.w,
                  mu.q = mu.q,
                  s.z = s.z,
                  s.w = s.w,
                  s.q = s.q,
                  s.wz = s.wz,
                  s.zq = s.zq,
                  s.wq = s.wq,
                  p.miss = p.miss,
                  missing.type = missing.type,
                  b.x = b.x,
                  b.z = b.z,
                  b.w = b.w,
                  b.m = betaM,
                  mu.e = mu.e,
                  s.e = s.e,
                  betaM = betaM)

  if(Y.PM == TRUE) {dat$y <- dat['y.pmm'] }

  #Full Model that we could use if we had all the data-we will use this model to get the 'truth'
  mod.truth <- lm(y ~ z + w, data=dat)
  mod.truth.int <- lm(y ~ (z + w)*m, data=dat)

  #From those people make some of their information missing
  dat.miss <- dat
  dat.miss$z <- ifelse(dat$m==1,NA,dat$z)

  datdis <<- datadist(dat.miss)
  options(datadist='datdis')

  #complete case model
  mod.cc <- lm(y ~  z + w, data=dat.miss)
  mod.ccsm.w <- lm(y ~ w, data=dat.miss)

  ###############################################################################
  #Conditional Mean Imputations and Results Using both the MI package and Hmisc
  ###############################################################################

  #mi package
  mdf <- suppressMessages(suppressWarnings(missing_data.frame(dat.miss[,c('x','z','w')])))
  #the expectation argument is conditional mean expectation but the underlying model is Bayesian
  mdf <-  suppressMessages(change(mdf, y=c('x','z','w'),what = "imputation_method", to = "expectation"))
  imputations <- mi(mdf,n.iter=1,n.chains=1,verbose=FALSE,parallel=FALSE)
  dat.miss.cm.mipack <- dat.miss
  dat.miss.cm.mipack[,c('x','z','w')] <- complete(imputations,1)[,c('x','z','w')]

  #regression imputation from Hmisc
  f <- aregImpute( ~ w + z, data=dat.miss, n.impute=1,type='regression',pr=FALSE)
  dat.miss.cm.Hmisc <- dat.miss
  dat.miss.cm.Hmisc[,c('z')] <- as.data.frame(impute.transcan(
    f, imputation=1, data=dat.miss,
    list.out=TRUE, pr=FALSE,
    check=FALSE))[,c('z')]

  ##My conditional mean imputation with no error
  dat.miss.cond.mean <- dat.miss
  mod.pred.z <- lm(z ~ w, data = dat.miss)
```

```
dat.miss.cond.mean[ dat.miss.cond.mean$m==1,'z'] <- predict(mod.pred.z,  dat.miss.cond.mean[ dat.miss.cond.mean$m==1,])
mod.full.cond.mean <- lm(y ~ (z + w)*m, data=dat.miss.cond.mean)
mod.marg.cond.mean <- lm(y ~ z + w, data=dat.miss.cond.mean)

#Fit model with the imputed conditional means indicator for missingness
mod.full.cm.mipack <- lm(y ~ (z + w)*m, data=dat.miss.cm.mipack)
mod.full.cm.Hmisc <- lm(y ~ (z + w)*m, data=dat.miss.cm.Hmisc)
mod.marg.cm.mipack <- lm(y ~  z + w, data=dat.miss.cm.mipack)
mod.marg.cm.Hmisc <- lm(y ~ z + w, data=dat.miss.cm.Hmisc)

### PMKS
mod.zmiss <- lm(y ~ w, data=dat.miss[dat.miss$m==1,])
pmks.mod <- pmks(DATA = dat.miss, model = "y ~ z + w", logistic = FALSE)
ccsm.mod <- ccsm(DATA = dat.miss, model = "y ~ z + w", logistic = FALSE)

#Missing data pattern percentages
mpp <- (table(mdf@patterns)/n.tr)

##################################################
#Mulitple Imputation Using a Congenial Model with pmm
##################################################
imp.congenial <- aregImpute( ~  w + z, data=dat.miss, n.impute=10,pr=FALSE)
mod.full.mi.Hmisc <- fit.mult.impute(y ~ (z + w)*m, ols,imp.congenial,data=dat.miss,
                                     pr=FALSE)
mod.marg.mi.Hmisc <- fit.mult.impute(y ~ (z + w), ols,imp.congenial,data=dat.miss,pr=FALSE)

##################################################
#Mulitple Imputation Using a Congenial Model with Y
##################################################

imp.y <- aregImpute( ~ w + z + y, data=dat.miss, n.impute=10,pr=FALSE)
mod.full.miy.Hmisc <- fit.mult.impute(y ~ (z + w)*m, ols,imp.y,data=dat.miss,pr=FALSE)
mod.marg.miy.Hmisc <- fit.mult.impute(y ~ (z + w), ols,imp.y,data=dat.miss,pr=FALSE)

#####################################################################
#Mulitple Imputation Using an Imputation Model with extra information q
#####################################################################

imp.q <- aregImpute( ~ w + z + q, data=dat.miss, n.impute=10,pr=FALSE)
mod.full.miq.Hmisc <- fit.mult.impute(y ~ (z + w)*m, ols,imp.q,data=dat.miss,pr=FALSE)
mod.marg.miq.Hmisc <- fit.mult.impute(y ~ (z + w), ols,imp.q,data=dat.miss,pr=FALSE)

##############################################################################
#Mulitple Imputation Using an Imputation Model with extra information y and q
##############################################################################

imp.yq <- aregImpute( ~  w + z + q + y, data=dat.miss, n.impute=10,pr=FALSE)
mod.full.miyq.Hmisc <- fit.mult.impute(y ~ (z + w)*m, ols,imp.yq,data=dat.miss,pr=FALSE)
mod.marg.miyq.Hmisc <- fit.mult.impute(y ~ (z + w), ols,imp.yq,data=dat.miss,pr=FALSE)



##########################
# In Sample Pattern Metrics
##########################

pattern.is <- which.pattern(dat.miss, model = 'y ~ z + w')

oracle.marg.pattern.is <- get_metric_linear(data.frame(predict(mod.truth, dat),
                                                       dat[,'y'],
                                                       pattern.is))

oracle.full.pattern.is <- get_metric_linear(data.frame(predict(mod.truth.int, dat),
                                                       dat[,'y'],
                                                       pattern.is))

dat.cc.pred <- dat.miss
dat.cc.pred[which(is.na(dat.cc.pred$z)==TRUE),'z'] <- mean(dat.miss$z, na.rm=TRUE)

cc.mean.pattern.is <- get_metric_linear(data.frame( predict(mod.cc, dat.cc.pred),
                                                       dat.miss[,'y'],
                                                       pattern.is))

full.cond.mean.pattern.is <- get_metric_linear(data.frame(predict(mod.full.cond.mean),
```

```
                                                    dat.miss[,'y'],
                                                    pattern.is))

marg.cond.mean.pattern.is <- get_metric_linear(data.frame(predict(mod.marg.cond.mean),
                                                    dat.miss[,'y'],
                                                    pattern.is))

full.cm.mipack.pattern.is <- get_metric_linear(data.frame(predict(mod.full.cm.mipack),
                                                    dat.miss[,'y'],
                                                    pattern.is))

marg.cm.mipack.pattern.is <- get_metric_linear(data.frame(predict(mod.marg.cm.mipack),
                                                    dat.miss[,'y'],
                                                    pattern.is))

#CM with Hmisc

full.cm.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.full.cm.Hmisc),
                                                   dat.miss[,'y'],
                                                   pattern.is))

marg.cm.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.marg.cm.Hmisc),
                                                    dat.miss[,'y'],
                                                    pattern.is))

#MI predictions
###

full.mi.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.full.mi.Hmisc),
                                                   dat.miss[,'y'],
                                                   pattern.is))

marg.mi.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.marg.mi.Hmisc),
                                                   dat.miss[,'y'],
                                                   pattern.is))


#MI q predictions

full.miq.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.full.miq.Hmisc),
                                                    dat.miss[,'y'],
                                                    pattern.is))

marg.miq.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.marg.miq.Hmisc),
                                                    dat.miss[,'y'],
                                                    pattern.is))


#MI y predictions
full.miy.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.full.miy.Hmisc),
                                                    dat.miss[,'y'],
                                                    pattern.is))

marg.miy.Hmisc.pattern.is <- get_metric_linear(data.frame(predict(mod.marg.miy.Hmisc),
                                                    dat.miss[,'y'],
                                                    pattern.is))

#Submodel predictions
is.pmks <- predict.sm(dat.miss, 'y ~ z + w', pmks.object = pmks.mod, logistic = FALSE)
is.ccsm <- predict.sm(dat.miss, 'y ~ z + w', pmks.object = ccsm.mod, logistic = FALSE)
metric.pmks.is <- data.frame(mse = c(is.pmks[[1]]$strat.brier, is.pmks[[2]]$strat.brier),
                             proportion = c(length(is.pmks[[1]]$truth)/n.tr, length(is.pmks[[2]]$truth)/n.tr))
rownames(metric.pmks.is) <- c(is.pmks[[1]][1], is.pmks[[2]][1])
metric.ccsm.is <- data.frame(mse = c(is.ccsm[[1]]$strat.brier, is.ccsm[[2]]$strat.brier),
                             proportion = c(length(is.ccsm[[1]]$truth)/n.tr, length(is.ccsm[[2]]$truth)/n.tr))
rownames(metric.ccsm.is) <- c(is.ccsm[[1]][1], is.ccsm[[2]][1])



##############################################################################################
#Get new sample, make people missing, imput their missing data, and then get model predictions
# - Assume 1 by 1 predictions
##############################################################################################
```

```
new.sample <- oos.data(n.tr.new,
                       mu.z.new,
                       mu.w.new,
                       mu.q.new,
                       s.z.new,
                       s.w.new,
                       s.q.new,
                       s.wz.new,
                       s.zq.new,
                       s.wq.new,
                       p.miss.new,
                       missing.type.new,
                       b.x.new,
                       b.z.new,
                       b.w.new,
                       mu.e.new,
                       s.e.new,
                       betaM.new=betaM.new,
                       n.imp = n.imp,
                       original.dat.miss = dat.miss,
                       Y.PM = Y.PM
                       )



pattern <- which.pattern(new.sample[["dat.miss.new"]], model = 'y ~ z + w')

#The oracle model
oracle.marg <- mean((new.sample[["dat.new"]][,'y'] -
                       predict(mod.truth, new.sample[["dat.new"]]))^2)

oracle.full <- mean((new.sample['dat.new']][,'y']  -
                       predict(mod.truth.int, new.sample[["dat.new"]]))^2)

oracle.marg.pattern <- get_metric_linear(data.frame(predict(mod.truth, new.sample[["dat.new"]]),
                             new.sample[["dat.new"]][,'y'],
                             pattern))

oracle.full.pattern <- get_metric_linear(data.frame(predict(mod.truth.int, new.sample[["dat.new"]]),
                                                    new.sample[["dat.new"]][,'y'],
                                                    pattern))


#CM with MI pack on the complete case model
mse.cc.mean.new <- mean((new.sample[["dat.mean"]][,'y'] -
                                predict(mod.cc,new.sample[["dat.mean"]]))^2)

mse.cc.mean.new.pattern <- get_metric_linear(data.frame( predict(mod.cc,new.sample[["dat.mean"]]),
                                                         new.sample[["dat.mean"]][,'y'],
                                                         pattern))

mse.cc.cm.Hmisc.new <- mean((new.sample[["imputed.dat.cm.Hmisc"]][,'y'] -
                               predict(mod.cc,new.sample[["imputed.dat.cm.Hmisc"]]))^2)

cc.cm.mipack.pattern <- get_metric_linear(data.frame( predict(mod.cc,new.sample[["imputed.dat.cm.mipack"]]),
                                                      new.sample[["imputed.dat.cm.mipack"]][,'y'],
                                                     pattern))

cc.cm.Hmisc.pattern <- get_metric_linear(data.frame( predict(mod.cc,new.sample[["imputed.dat.cm.Hmisc"]]),
                                                      new.sample[["imputed.dat.cm.Hmisc"]][,'y'],
                                                       pattern))
#Strict Conditional Mean

mse.full.cond.mean.new <- mean((new.sample[["dat.cond.mean"]][,'y'] -
                                 predict(mod.full.cond.mean,new.sample[["dat.cond.mean"]]))^2)


mse.marg.cond.mean.new <- mean((new.sample[["dat.cond.mean"]][,'y'] -
                                  predict(mod.marg.cond.mean,new.sample[["dat.cond.mean"]]))^2)

full.cond.mean.pattern <- get_metric_linear(data.frame(predict(mod.full.cond.mean,new.sample[["dat.cond.mean"]]),
                                                       new.sample[["dat.cond.mean"]][,'y'],
                                                       pattern))
```

```
marg.cond.mean.pattern <- get_metric_linear(data.frame(predict(mod.marg.cond.mean,new.sample[["dat.cond.mean"]]),
                                                       new.sample[["dat.cond.mean"]][,'y'],
                                                       pattern))
#CM with MI pack

mse.full.cm.mipack.new <-  mean((new.sample[["imputed.dat.cm.mipack"]][,'y'] -
                                   predict(mod.full.cm.mipack,new.sample[["imputed.dat.cm.mipack"]]))^2)

mse.marg.cm.mipack.new <-  mean((new.sample[["imputed.dat.cm.mipack"]][,'y'] -
                                   predict(mod.marg.cm.mipack,new.sample[["imputed.dat.cm.mipack"]]))^2)


full.cm.mipack.pattern <- get_metric_linear(data.frame(predict(mod.full.cm.mipack,new.sample[["imputed.dat.cm.mipack"]]),
                                                     new.sample[["imputed.dat.cm.mipack"]][,'y'],
                                                     pattern))

marg.cm.mipack.pattern <- get_metric_linear(data.frame(predict(mod.marg.cm.mipack,new.sample[["imputed.dat.cm.mipack"]]),
                                                       new.sample[["imputed.dat.cm.mipack"]][,'y'],
                                                       pattern))

#CM with Hmisc
mse.full.cm.Hmisc.new <- mean((new.sample[["imputed.dat.cm.Hmisc"]][,'y'] -
                                 predict(mod.full.cm.Hmisc,new.sample[["imputed.dat.cm.Hmisc"]]))^2)

mse.marg.cm.Hmisc.new <- mean((new.sample[["imputed.dat.cm.Hmisc"]][,'y'] -
                                 predict(mod.marg.cm.Hmisc,new.sample[["imputed.dat.cm.Hmisc"]]))^2)

full.cm.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.full.cm.Hmisc,new.sample[["imputed.dat.cm.Hmisc"]]),
                                                     new.sample[["imputed.dat.cm.Hmisc"]][,'y'],
                                                      pattern))

marg.cm.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.marg.cm.Hmisc,new.sample[["imputed.dat.cm.Hmisc"]]),
                                                     new.sample[["imputed.dat.cm.Hmisc"]][,'y'],
                                                     pattern))

#MI predictions
###

mse.full.mi.Hmisc.new <-  mean((new.sample[["imputed.dat.mi.avg"]][,'y'] -
                                   predict(mod.full.mi.Hmisc,new.sample[["imputed.dat.mi.avg"]]))^2)

mse.marg.mi.Hmisc.new <-  mean((new.sample[["imputed.dat.mi.avg"]][,'y'] -
                                   predict(mod.marg.mi.Hmisc,new.sample[["imputed.dat.mi.avg"]]))^2)

full.mi.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.full.mi.Hmisc,new.sample[["imputed.dat.mi.avg"]]),
                                                     new.sample[["imputed.dat.mi.avg"]][,'y'],
                                                     pattern))

marg.mi.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.marg.mi.Hmisc,new.sample[["imputed.dat.mi.avg"]]),
                                                     new.sample[["imputed.dat.mi.avg"]][,'y'],
                                                     pattern))


#MI q predictions
mse.full.miq.Hmisc.new <- mean((new.sample[["imputed.dat.miq.avg"]][,'y'] -
                                   predict(mod.full.miq.Hmisc,new.sample[["imputed.dat.miq.avg"]]))^2)

mse.marg.miq.Hmisc.new <- mean((new.sample[["imputed.dat.miq.avg"]][,'y'] -
                                   predict(mod.marg.miq.Hmisc,new.sample[["imputed.dat.miq.avg"]]))^2)

full.miq.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.full.miq.Hmisc,new.sample[["imputed.dat.miq.avg"]]),
                                                       new.sample[["imputed.dat.miq.avg"]][,'y'],
                                                     pattern))

marg.miq.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.marg.miq.Hmisc,new.sample[["imputed.dat.miq.avg"]]),
                                                       new.sample[["imputed.dat.miq.avg"]][,'y'],
                                                       pattern))


#MI y predictions
mse.full.miy.Hmisc.new <- mean((new.sample[["imputed.dat.miy.avg"]][,'y'] -
                                   predict(mod.full.miy.Hmisc,new.sample[["imputed.dat.miy.avg"]]))^2)

mse.marg.miy.Hmisc.new <- mean((new.sample[["imputed.dat.miy.avg"]][,'y'] -
```

```
                                predict(mod.marg.miy.Hmisc,new.sample[["imputed.dat.miy.avg"]]))^2)

 full.miy.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.full.miy.Hmisc,new.sample[["imputed.dat.miy.avg"]]),
                                                         new.sample[["imputed.dat.miy.avg"]][,'y'],
                                                         pattern))

 marg.miy.Hmisc.pattern <- get_metric_linear(data.frame(predict(mod.marg.miy.Hmisc,new.sample[["imputed.dat.miy.avg"]]),
                                                         new.sample[["imputed.dat.miy.avg"]][,'y'],
                                                         pattern))

 #Submodel predictions
 oos.pmks <- predict.sm(new.sample[["dat.miss.new"]], 'y ~ z + w', pmks.object = pmks.mod, logistic = FALSE)
 oos.ccsm <- predict.sm(new.sample[["dat.miss.new"]], 'y ~ z + w', pmks.object = ccsm.mod, logistic = FALSE)
 metric.pmks <- data.frame(mse = c(oos.pmks[[1]]$strat.brier, oos.pmks[[2]]$strat.brier),
                           proportion = c(length(oos.pmks[[1]]$truth)/n.tr.new, length(oos.pmks[[2]]$truth)/n.tr.new))
 rownames(metric.pmks) <- c(oos.pmks[[1]][1], oos.pmks[[2]][1])
 metric.ccsm <- data.frame(mse = c(oos.ccsm[[1]]$strat.brier, oos.ccsm[[2]]$strat.brier),
                           proportion = c(length(oos.ccsm[[1]]$truth)/n.tr.new, length(oos.ccsm[[2]]$truth)/n.tr.new))
 rownames(metric.ccsm) <- c(oos.ccsm[[1]][1], oos.ccsm[[2]][1])


 ############################
 #  Storing all the results
 ############################
results[['is.mse']][RS,] <- c(
     #In sample MSE
    mse(mod.truth),
    mse(mod.truth.int),
    mse(mod.cc),
    mse(mod.full.cm.mipack),
    mse(mod.full.cm.Hmisc),
    mse(mod.marg.cm.mipack),
    mse(mod.marg.cm.Hmisc),
    mse(mod.full.mi.Hmisc),
    mse(mod.marg.mi.Hmisc),
    mse(mod.full.miy.Hmisc),
    mse(mod.marg.miy.Hmisc),
    mse(mod.full.miq.Hmisc),
    mse(mod.marg.miq.Hmisc),
    mse(mod.full.miyq.Hmisc),
    mse(mod.marg.miyq.Hmisc),
    mse(mod.full.cond.mean),
    mse(mod.marg.cond.mean)
    )

results[['is.pattern.00']][RS,] <- c(
  oracle.marg.pattern.is[1,] ,
  oracle.full.pattern.is[1,],
  cc.mean.pattern.is[1,],
  full.cm.mipack.pattern.is[1,],
  marg.cm.mipack.pattern.is[1,],
  full.cm.Hmisc.pattern.is[1,],
  marg.cm.Hmisc.pattern.is[1,],
  full.mi.Hmisc.pattern.is[1,],
  marg.mi.Hmisc.pattern.is[1,],
  full.miq.Hmisc.pattern.is[1,],
  marg.miq.Hmisc.pattern.is[1,],
  full.miy.Hmisc.pattern.is[1,],
  marg.miy.Hmisc.pattern.is[1,],
  metric.pmks.is[1,],
  metric.ccsm.is[1,],
  full.cond.mean.pattern.is[1,],
  marg.cond.mean.pattern.is[1,]
)

results[['is.pattern.10']][RS,] = c(
  oracle.marg.pattern.is[2,] ,
  oracle.full.pattern.is[2,],
  cc.mean.pattern.is[2,],
  full.cm.mipack.pattern.is[2,],
  marg.cm.mipack.pattern.is[2,],
  full.cm.Hmisc.pattern.is[2,],
  marg.cm.Hmisc.pattern.is[2,],
  full.mi.Hmisc.pattern.is[2,],
```

```
  marg.mi.Hmisc.pattern.is[2,],
  full.miq.Hmisc.pattern.is[2,],
  marg.miq.Hmisc.pattern.is[2,],
  full.miy.Hmisc.pattern.is[2,],
  marg.miy.Hmisc.pattern.is[2,],
  metric.pmks.is[2,],
  metric.ccsm.is[2,],
  full.cond.mean.pattern.is[2,],
  marg.cond.mean.pattern.is[2,])

#Out of sample MSE
results[['oos.mse']][RS,] <-  c(
     mse.cc.mean.new,
     mse.cc.cm.Hmisc.new,
     mse.full.cm.mipack.new,
     mse.marg.cm.mipack.new,
     mse.full.cm.Hmisc.new,
     mse.marg.cm.Hmisc.new,
     mse.full.mi.Hmisc.new,
     mse.marg.mi.Hmisc.new,
     mse.full.miq.Hmisc.new,
     mse.marg.miq.Hmisc.new,
     mse.full.miy.Hmisc.new,
     mse.marg.miy.Hmisc.new,
     oracle.marg,
     oracle.full,
     mse.full.cond.mean.new,
     mse.marg.cond.mean.new
   )

results[['oos.pattern.00']][RS,] <- c(
     oracle.marg.pattern[1,] ,
     oracle.full.pattern[1,],
     mse.cc.mean.new.pattern[1,],
     cc.cm.mipack.pattern[1,],
     cc.cm.Hmisc.pattern[1,],
     full.cm.mipack.pattern[1,],
     marg.cm.mipack.pattern[1,],
     full.cm.Hmisc.pattern[1,],
     marg.cm.Hmisc.pattern[1,],
     full.mi.Hmisc.pattern[1,],
     marg.mi.Hmisc.pattern[1,],
     full.miq.Hmisc.pattern[1,],
     marg.miq.Hmisc.pattern[1,],
     full.miy.Hmisc.pattern[1,],
     marg.miy.Hmisc.pattern[1,],
     metric.pmks[1,],
     metric.ccsm[1,],
     full.cond.mean.pattern[1,],
     marg.cond.mean.pattern[1,]
     )

   results[['oos.pattern.10']][RS,] = c(
     oracle.marg.pattern[2,] ,
     oracle.full.pattern[2,],
     mse.cc.mean.new.pattern[2,],
     cc.cm.mipack.pattern[2,],
     cc.cm.Hmisc.pattern[2,],
     full.cm.mipack.pattern[2,],
     marg.cm.mipack.pattern[2,],
     full.cm.Hmisc.pattern[2,],
     marg.cm.Hmisc.pattern[2,],
     full.mi.Hmisc.pattern[2,],
     marg.mi.Hmisc.pattern[2,],
     full.miq.Hmisc.pattern[2,],
     marg.miq.Hmisc.pattern[2,],
     full.miy.Hmisc.pattern[2,],
     marg.miy.Hmisc.pattern[2,],
     metric.pmks[2,],
     metric.ccsm[2,],
     full.cond.mean.pattern[2,],
     marg.cond.mean.pattern[2,])

   new.sample.z <- data.frame(do.call(cbind,lapply(new.sample, function(z) cbind(z$z))))
```

```
    names(new.sample.z) <- names(new.sample)
    results[['oos.z.mse']][RS,] = c(apply(new.sample.z, 2,function(z) mean((z - new.sample.z$dat.new)^2)))

}
results
}



###############
#EPE Function
##############
#This function simulates the data for Figure 1 comparing the Expected prediction error for the large and small model

Install Libraries
library(MASS)
library(DMwR)
library(mice)
library(doSNOW)
library(rms)

#######################################
####  Training data & prediction interval functions
#######################################
expit <- function(x) exp(x)/(1 + exp(x))
logit <- function(p) log(p/(1-p))

miss.int <- function(miss.mech, betaM=1,p.miss){
  if(ncol(miss.mech)!=2){
    logit(p.miss) - betaM*mean(miss.mech[,1])
  } else {
    logit(p.miss) - betaM*mean((((miss.mech[,2]/sd(miss.mech[,2]))
                                + miss.mech[,1])/as.numeric(sqrt(2*(1+cor(miss.mech[,2],miss.mech[,1]))))))
  }
}


dat.frame <- function(n.tr,mu.z,mu.w,s.z,s.w,s.wz,p.miss,
                      missing.type,
                      b.x,b.z,b.w,
                      mu.e,s.e,betaM=1){
  x = rep(1,n.tr)
  mu = c(mu.z,mu.w)
  Sigma = matrix(c(1,s.wz,s.wz,1), byrow=TRUE, ncol=2)
  dat =  as.data.frame(mvrnorm(n=n.tr, mu=mu, Sigma=Sigma, empirical=TRUE))
  colnames(dat) = c('z','w')
  coef.tru = rbind( b.x , b.z , b.w)
  dat$y=cbind(x,dat[,'z'], dat[,'w'])%*%coef.tru
  + rnorm(n.tr,mean=mu.e,sd=s.e)

  if(missing.type=='MCAR'){
    m = rbinom(n.tr,1,p.miss)
    y.pmm = cbind(x,dat[,'z'], dat[,'w'])%*%coef.tru + betaM*m + betaM*dat[,'z']*m + betaM*dat[,'w']*m
    +  rnorm(n.tr,mean=mu.e,sd=s.e)
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], m=m, y=dat[,'y'], y.pmm = y.pmm)
  } else if(missing.type=='MAR'){
    m = rbinom(n.tr, 1, expit(miss.int(miss.mech=cbind(dat[,'w']),
                                       betaM=betaM,p.miss) + betaM*dat[,'w']))
    y.pmm = cbind(x,dat[,'z'], dat[,'w'])%*%coef.tru + betaM*m + betaM*dat[,'z']*m + betaM*dat[,'w']*m
    + rnorm(n.tr,mean=mu.e,sd=(s.e + 2*m))
    data.frame(x=x, z=dat[,1], w=dat[,2], m=m, y=dat[,'y'], y.pmm = y.pmm)
  } else if(missing.type=='MNAR'){
    m = rbinom(n.tr, 1, expit(miss.int(miss.mech=cbind(dat[,'z']),
                                       betaM=betaM,p.miss) + betaM*dat[,'z']))
    y.pmm = cbind(x,dat[,'z'], dat[,'w'])%*%coef.tru + betaM*m + betaM*dat[,'z']*m + betaM*dat[,'w']*m
    + rnorm(n.tr,mean=mu.e,sd=(s.e + 2*m))
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], m=m, y=dat[,'y'], y.pmm = y.pmm)
  } else if(missing.type=='MARY'){
    m = rbinom(n.tr, 1,
               expit(miss.int(miss.mech=dat[,c('w','y')], betaM=betaM,p.miss)
                     + betaM*(((dat[,'y']/sd(dat[,'y'])) + dat[,'w'])/as.numeric(sqrt(2*(1+cor(dat[,'y'],dat[,'w']))))))
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], m=m, y=dat[,'y'])
  } else if(missing.type=='MNARY'){
    m = rbinom(n.tr, 1,
```

```
              expit(miss.int(miss.mech=dat[,c('z','y')], betaM=betaM,p.miss)
                    + betaM*(((dat[,'y']/sd(dat[,'y'])) + dat[,'z'])/as.numeric(sqrt(2*(1+cor(dat[,'y'],dat[,'z']))))))
    data.frame(x=x, z=dat[,'z'], w=dat[,'w'], m=m, y=dat[,'y'])
  }

}

sims=1000;
n.y.tr = 100;
n.tr=10000;
b.x=0.5; b.z = 3; b.w = 3;
mu.z=3;
mu.w=3;
s.z=1;
s.w=1;
s.wz=0.5;
p.miss=0.5;
missing.type='MCAR';
lo.z=-1; hi.z=7;
acc=0.001;
mu.e=0;
s.e=1;
mu.cond.imp=0;
s.cond.imp=1;
imputed='conditional';
orig.imp='includey';
y.lo=7; y.hi=15;
b.m=1
missing.type.orig='MCAR';
PMMY = FALSE


EPE.T  = function(sims=10000,
                  n.y.tr = 10,
                  n.tr=50,
                  b.x=bx, b.z = bz, b.w = bw,
                  mu.z=mean.z,
                  mu.w=mean.w,
                  s.z=sd.z,
                  s.w=sd.w,
                  s.wz=sd.wz,
                  p.miss,
                  missing.type.orig,
                  missing.type,
                  lo.z=-1, hi.z=7,
                  acc=0.01,
                  mu.e=mean.e,
                  s.e=sd.e,
                  mu.cond.imp,
                  s.cond.imp,
                  imputed,
                  orig.imp,
                  y.lo=7, y.hi=15,
                  b.m = b.m,
                  PMMY = FALSE) {

  coef.tru = rbind( b.x , b.z , b.w)

  ## Predictors (Intercept in X)
  xz.dat  = replicate(n.y.tr,
                      dat.frame(n.tr = n.tr,
                                mu.z = mu.z,
                                mu.w = mu.w,
                                s.z = s.z,
                                s.wz = s.wz,
                                p.miss = p.miss,
                                missing.type = missing.type.orig,
                                b.x = b.x,
                                b.z = b.z,
                                b.w = b.w,
                                mu.e = mu.e,
                                s.e = s.e,
                                betaM=b.m
                      ),
```

```
                    simplify=FALSE)

data.tr.list = lapply(xz.dat, function(q) {data.frame(x=q$x, z=q$z,
                                                      w=q$w, m=q$m,
                                                      y = q$y,
                                                      y.pmm = q$y.pmm)})


epes = lapply(data.tr.list, function(q) {

  if(PMMY==FALSE){
    y.tr       = q$y
  } else {
    y.tr= q$y.pmm}
  x.tr       = q$x
  z.tr.truth = q$z
  w.tr = q$w
  m.z.tr      = q$m
  z.tr        = ifelse(q$m==1,NA,q$z)
  observed.miss    = sum(m.z.tr)/n.tr

  #Complete Case Design Matrix
  Z.cc = cbind(x.tr[m.z.tr==0],z.tr[m.z.tr==0],w.tr[m.z.tr==0],y.tr[m.z.tr==0])
  Z.orig.withMiss = cbind(x.tr,z.tr,w.tr,y.tr)

  #Design Matrix Imputation Model
  Z.s    = cbind(x.tr[m.z.tr==0],w.tr[m.z.tr==0])
  bhat.z  = chol2inv(chol(t(Z.s)%*%Z.s))%*%t(Z.s)%*%z.tr[m.z.tr==0]

  #Get beta for a imputation model that includes y
  Z.y    = cbind(x.tr[m.z.tr==0],w.tr[m.z.tr==0], y.tr[m.z.tr==0])
  bhat.zy  = chol2inv(chol(t(Z.y)%*%Z.y))%*%t(Z.y)%*%z.tr[m.z.tr==0]

  #Impute E[z ~ x + w] for those individuals missing
  if(orig.imp=='noy'){
    z.tr[is.na(z.tr)] = cbind(x.tr[m.z.tr==1],w.tr[m.z.tr==1])%*%bhat.z
  } else if(orig.imp=='includey'){
    z.tr[is.na(z.tr)] = cbind(x.tr[m.z.tr==1],w.tr[m.z.tr==1],y.tr[m.z.tr==1])%*%bhat.zy
  }


  ## Design Matrix
  if(PMMY==FALSE){
    X.truth = cbind(x.tr,z.tr.truth, w.tr)
  } else {
    X.truth = cbind(x.tr,z.tr.truth,w.tr,m.z.tr,z.tr.truth*m.z.tr,w.tr*m.z.tr)
  }



  X.l    = cbind(x.tr,z.tr,w.tr)
  X.s    = cbind(x.tr,w.tr)
  X.l.cc  = cbind(x.tr[m.z.tr==0],z.tr[m.z.tr==0],w.tr[m.z.tr==0])
  X.s.cc  = cbind(x.tr[m.z.tr==1], w.tr[m.z.tr==1])

  ## Least Squares estimates

  bhat.truth = chol2inv(chol(t(X.truth)%*%X.truth))%*%t(X.truth)%*%y.tr
  bhat.l     = chol2inv(chol(t(X.l)%*%X.l))%*%t(X.l)%*%y.tr
  bhat.s    = chol2inv(chol(t(X.s)%*%X.s))%*%t(X.s)%*%y.tr
  bhat.l.cc  = chol2inv(chol(t(X.l.cc)%*%X.l.cc))%*%t(X.l.cc)%*%y.tr[m.z.tr==0]
  bhat.s.cc  = chol2inv(chol(t(X.s.cc)%*%X.s.cc))%*%t(X.s.cc)%*%y.tr[m.z.tr==1]

  #NOTE: bhat.s != bhat.s.cc because the lengths are different!

  ################################################
  # This Section Starts the Part of the Simulation
  # code that includes the OUT OF SAMPLE population
  ################################################

  ## Create Prediction Set
  z.pred    = seq(lo.z, hi.z, acc)
  #should I predict w based on z.pred
  w.pred   = rnorm(length(z.pred),
```

```
                mean=(mu.w + s.wz*(s.w/s.z)*(z.pred - mu.z)), sd = (1-s.wz^2)*s.w^2)
x.pred    = rep(1,length(z.pred))
n.pred   = length(z.pred)

z.pred.sim  = rep(z.pred, each = sims)
x.pred.sim = rep(x.pred, each = sims)
w.pred.sim = rep(w.pred, each = sims)
## Make new true responses (y) at x.new (sim replicates)


m.pred    = if(missing.type=='MCAR'){
  rbinom(length(z.pred),1,p.miss)
} else if(missing.type=='MAR'){
  rbinom(length(z.pred), 1,
         expit(miss.int(
           miss.mech=cbind(w.pred),betaM=b.m,p.miss=p.miss) + b.m*w.pred))
} else if(missing.type=='MNAR'){
  rbinom(length(z.pred), 1,
         expit(miss.int(miss.mech=cbind(z.pred),
                        betaM=b.m,p.miss=p.miss) + b.m*z.pred))
} else if(missing.type=='MARY'){
  rbinom(length(z.pred.sim), 1,
         expit(miss.int(miss.mech=cbind(w.pred.sim,y.new),
                        betaM=b.m,p.miss=p.miss)
               + b.m*(((y.new/sd(y.new))
                       + w.pred.sim)/as.numeric(sqrt(2*(1+cor(y.new,w.pred.sim)))))))
} else if(missing.type=='MNARY'){
  rbinom(length(z.pred.sim), 1,
         expit(miss.int(miss.mech=cbind(z.pred.sim,y.new),
                        betaM=b.m,p.miss=p.miss)
               + b.m*(((y.new/sd(y.new))
                       + z.pred.sim)/as.numeric(sqrt(2*(1+cor(y.new,z.pred.sim)))))))
}

if(PMMY==FALSE){
  y.new = cbind(x.pred.sim,z.pred.sim,w.pred.sim)%*%coef.tru + rnorm(n.pred*sims,mean=mu.e,sd=s.e)
} else {
  y.new = cbind(x.pred.sim,z.pred.sim,w.pred.sim)%*%coef.tru
  + b.m*rep(m.pred,each=sims) +  b.m*z.pred.sim*rep(m.pred,each=sims)
  + b.m*rep(m.pred,each=sims)*w.pred.sim
  + rnorm(n.pred*sims,mean=mu.e,sd=(s.e + 2*rep(m.pred,each=sims)))
}


Z.pred.miss = cbind(x.pred[m.pred==1],NA,w.pred[m.pred==1],NA)

prop.pred.miss  =  sum(m.pred)/length(z.pred)
n.pred.miss          =  length(z.pred) - sum(m.pred)
n.pred.notmiss       =  length(z.pred) - sum(abs(m.pred-1))
n.pred.sim     =  length(z.pred.sim)
n.pred  = length(z.pred)

z.pred.sim.miss  = rep(z.pred[m.pred==1], each = sims)
z.pred.sim.notmiss  = rep(z.pred[m.pred==0], each = sims)

x.pred.sim.miss = rep(x.pred[m.pred==1], each = sims)
x.pred.sim.notmiss = rep(x.pred[m.pred==0], each = sims)

w.pred.sim.miss = rep(w.pred[m.pred==1], each = sims)
w.pred.sim.notmiss = rep(w.pred[m.pred==0], each = sims)

y.new.miss      = y.new[rep(m.pred,each=sims)==1]
y.new.notmiss  = y.new[rep(m.pred,each=sims)==0]

#Impute for the missing z's
z.pred.imputed = z.pred
z.pred.imputed.sim = z.pred.sim

z.pred.imputed[m.pred==1] = cbind(x.pred[m.pred==1],w.pred[m.pred==1])%*%bhat.z +
  rnorm(sum(m.pred==1),mean=mu.cond.imp,sd=s.cond.imp)
z.pred.sim.imputed = rep(z.pred.imputed, each = sims)

## Model predictions fhat(x.pred,z.pred) ; out-of-sample
yhat.new.truth    = cbind(x.pred.sim,z.pred.sim,w.pred.sim)%*%bhat.truth
```

```
yhat.new.l        = cbind(x.pred.sim,z.pred.sim,w.pred.sim)%*%bhat.l
yhat.new.s      = cbind(x.pred.sim,w.pred.sim)%*%bhat.s

#These take into account missing data
yhat.new.l.pmks     = cbind(x.pred.sim.notmiss,z.pred.sim.notmiss,w.pred.sim.notmiss)%*%bhat.l.cc
yhat.new.s.pmks    = cbind(x.pred.sim.miss,w.pred.sim.miss)%*%bhat.s.cc
yhat.new.l.imputed   = cbind(x.pred.sim,z.pred.sim.imputed,w.pred.sim)%*%bhat.l
yhat.new.l.cc    = cbind(x.pred.sim.notmiss,z.pred.sim.notmiss,w.pred.sim.notmiss)%*%bhat.l.cc
yhat.new.s.cc   = cbind(x.pred.sim.miss,w.pred.sim.miss)%*%bhat.s


## Prediction Errors ( (y.new-fhat)^2 ) ; out-of-sample
err.truth    = matrix(((y.new-yhat.new.truth)^2),ncol=n.pred,nrow=sims,byrow=FALSE)
err.l    = matrix(((y.new-yhat.new.l)^2),ncol=n.pred,nrow=sims,byrow=FALSE)
err.s     = matrix(((y.new-yhat.new.s)^2),ncol=n.pred,nrow=sims,byrow=FALSE)
err.l.imputed = matrix(((y.new-yhat.new.l.imputed)^2),ncol=n.pred,nrow=sims,byrow=FALSE)


err.l.pmks   = matrix(((y.new.notmiss-yhat.new.l.pmks)^2),
                      ncol=n.pred-sum(m.pred),nrow=sims,byrow=FALSE)
err.s.pmks = matrix(((y.new.miss-yhat.new.s.pmks)^2),
                    ncol=sum(m.pred),nrow=sims,byrow=FALSE)
err.l.cc   = matrix(((y.new.notmiss-yhat.new.l.cc)^2),
                    ncol=n.pred-sum(m.pred),nrow=sims,byrow=FALSE)
err.s.cc  = matrix(((y.new.miss - yhat.new.s.cc)^2),
                   ncol=sum(m.pred),nrow=sims,byrow=FALSE)


# Mean prediction error (over sims) out-of-sample & overall
avg.pe.truth = colMeans(err.truth) # square matrix only
avg.pe.l     = colMeans(err.l) # square matrix only
avg.pe.s     = colMeans(err.s) # square matrix only
avg.pe.l.pmks  = colMeans(err.l.pmks) # square matrix only
avg.pe.s.pmks   = colMeans(err.s.pmks)  # square matrix only
avg.pe.l.imputed = colMeans(err.l.imputed) # square matrix only
avg.pe.l.cc   = colMeans(err.l.cc) # square matrix only
avg.pe.s.cc   = colMeans(err.s.cc)  # square matrix only


merged.ls.sm  = cbind(c(z.pred[m.pred==0],z.pred[m.pred==1]),c(avg.pe.l.pmks,avg.pe.s.pmks))
merged.ls = cbind(c(z.pred[m.pred==0],z.pred[m.pred==1]),c(avg.pe.l.cc,avg.pe.s.cc))

avg.pe.pmks  = merged.ls.sm[sort(merged.ls.sm[,1],index.return=TRUE)$ix,2]
avg.pe.cc    = merged.ls[sort(merged.ls[,1],index.return=TRUE)$ix,2]

###########
# Exact Equations
##########
X.p     = cbind(x.pred,z.pred, w.pred)
X.p.imputed = cbind(x.pred, z.pred.imputed, w.pred)
X.p.small = cbind(x.pred, w.pred)

## Hat Matricies for prediction
hat.truth = diag(X.p%*%chol2inv(chol(t(X.truth)%*%X.truth))%*%t(X.p))
hat.l    = diag(X.p%*%chol2inv(chol(t(X.l)%*%X.l))%*%t(X.p))
hat.s    = diag(X.p.small%*%chol2inv(chol(t(X.s)%*%X.s))%*%t(X.p.small))
hat.l.cc = diag(X.p[m.pred==0,]%*%chol2inv(chol(t(X.l.cc)%*%X.l.cc))%*%t(X.p[m.pred==0,]))
hat.s.cc  = diag(X.p.small[m.pred==1,]%*%chol2inv(chol(t(X.s.cc)%*%X.s.cc))%*%t(X.p.small[m.pred==1,]))
hat.l.imputed = diag(X.p.imputed%*%chol2inv(chol(t(X.l)%*%X.l))%*%t(X.p.imputed))


var.truth.tr = s.e^2
var.l.tr     = s.e^2
var.l.x      = s.e^2

bias.truth.tr = cbind(x.pred,z.pred,w.pred)%*%(bhat.truth-coef.tru)
bias.l.tr    = cbind(x.pred,z.pred,w.pred)%*%(bhat.l-coef.tru)

bias.truth.x = 0  # Function assumes large model is true
bias.l.x     = 0  # ONLY TRUE WITH MCAR/MAR

epe.truth.tr = var.truth.tr * (1) + bias.truth.tr^2
epe.truth.x = var.l.x * (1+hat.truth) + bias.l.x^2
```

```
    epe.l.tr = var.l.tr * (1) + bias.l.tr^2
    epe.l.x = var.l.x * (1+hat.l) + bias.l.x^2

    # Verify var and bias of small model for generalized matrix z
    var.s.tr = s.e^2
    bias.s.tr = cbind(x.pred,z.pred,w.pred)%*%(rbind(bhat.s,0)-coef.tru)

    var.s.x = s.e^2*diag(cbind(x.pred,w.pred)%*%chol2inv(chol(t(X.s)%*%X.s))%*%t(cbind(x.pred,w.pred)))
    bias.s.x = (cbind(x.pred,w.pred)%*%chol2inv(chol(t(X.s)%*%X.s))%*%t(X.s)%*%y.tr)- cbind(x.pred,z.pred,w.pred)%*%coef.tru

    epe.s.tr = as.numeric(var.s.tr) * (1) + bias.s.tr^2
    epe.s.x =  bias.s.x^2 + var.s.x + s.e^2



    ############
    # End Exact Equations
    ############

    data.frame(avg.pe.truth,
               avg.pe.l,
               avg.pe.s,
               avg.pe.pmks,
               avg.pe.cc,
               avg.pe.l.imputed,
               weighted = avg.pe.l*(1-prop.pred.miss) +
                 avg.pe.s*(prop.pred.miss),
               z.pred,
               epe.truth.tr,
               epe.truth.x,
               epe.l.tr,
               epe.l.x,
               epe.s.tr,
               epe.s.x,
               weighted.epe = epe.truth.x*(1-prop.pred.miss) +
                 epe.s.x*(prop.pred.miss)

    )
  })

  mean.epes = Reduce("+", epes) / length(epes)
  save(mean.epes, '/Users/SFM/Desktop/SimulatedMCAR.Rda')
}
```

# CHAPTER 3

# BAGGED EMPIRICAL NULL *P*-VALUES: A METHOD TO ACCOUNT FOR MODEL UNCERTAINTY IN LARGE SCALE INFERENCE

## 3.1 Abstract

When conducting large scale inference, such as genome-wide association studies or image analysis, nominal $p$-values are often adjusted to improve control over the family-wise error rate (FWER). When the majority of tests are null, procedures controlling the False discovery rate (Fdr) can be improved by replacing the theoretical global null with its empirical estimate. However, these other adjustment procedures remain sensitive to the working model assumption. Here we propose two key ideas to improve inference in this space. First, we propose $p$-values that are standardized to the empirical null distribution (instead of the theoretical null). Second, we propose model averaging $p$-values by bootstrap aggregation (Bagging) to account for model uncertainty and selection procedures. The combination of these two key ideas yields *bagged empirical null p-values* (*BEN p-values*) that often dramatically alter the rank ordering of significant findings. Moreover, we find that a multidimensional selection criteria based on BEN $p$-values and bagged model fit statistics is more likely to yield reproducible findings. A re-analysis of the famous Golub Leukemia data is presented to illustrate these ideas. We uncovered new findings in these data, not detected previously, that are backed by published bench work pre-dating the Gloub experiment. A *pseudo-simulation* using the leukemia data is also presented to explore the stability of this approach under broader conditions, and illustrates the superiority of the BEN $p$-values compared to the other approaches.

## 3.2 Introduction

Modern day technology like microarrays, RNA-sequencing, and fMRI Imaging, has given rise to a new era of statistical methods for high-throughput science. These methods are commonly referred to as large-scale inference, and can produce data corresponding to millions of statistical hypotheses for simultaneous testing. However, the repeated application of classical hypothesis testing methods can lead to concerns regarding control over the inflated Family-wise Error Rate (FWER), global Type I Error rate, and reduced statistical power (Efron, 2012*a*).

Bonferroni adjustments, which provides strict control of the FWER, are highly

conservative in the large-scale context and they are often avoided because of the dramatic loss of power that is associated with their use (Sham and Purcell, 2014). The literature is turning to other $p$-values adjustments, such as the Benjamini-Hochberg (B-H) False discovery rate (Fdr) procedure (Benjamini and Hochberg, 1995), which improves power by relaxing control of the FWER. The FWER can remain inflated, so long as the Fdr remains below a pre-specified threshold. This yields increased power because the FWER inflation is reduced, but not eliminated. Note that it is not possible to control both the FWER and Fdr simultaneously. Both quantities are functions of the per comparison Type I Error rate, so a compromise between must be made.

When the majority of tests are null and Fdr adjustments are warranted, Efron (2012*a*) advocated for replacing the theoretical null with its empirical estimate to improve the operating characteristics of Fdr procedures. This *empirical Fdr* procedure is an empirical Bayes procedure where the empirical null distribution is estimated from the mixture distribution of null and non-null test statistics. A more desirable error-rate tradeoff results because the mixture distribution of test statistics does not necessarily follow the theoretical null distribution in large-scale data, often because of a complex correlation structure (Efron, 2012*a*).

While the use of the empirical null distribution has so far been limited to Fdr computations, this idea is readily propagated to the computation of $p$-values in large-scale data. Specifically, we propose standardizing $p$-value or test statistics of interest to the empirical null distribution. Despite the well documented problems with $p$-values, inference based on them is still more intuitive to applied scientists than inference based on false discovery rates. This approach has the added advantage of limiting the confusion between the inferential roles of Fdr and the $p$-value. The Fdr measures the tendency of the observed results to be misleading, while the $p$-value measures the degree to which the data are compatible with the null hypothesis (Blume, 2011). The former measures the uncertainly of the observed findings, while the latter is the metric of the strength of statistical evidence. We found through our investigations that empirically standardized $p$-values result in a desirable error-rate compromise between Bonferroni and Fdr methods.

We also observed extreme dependence of $p$-value and Fdr inference on working model assumptions. A common example is the assumption of a simple or trivially adjusted regression model in Genome wide association studies (GWAS) studies. In many cases, the sensitivity of the findings to this assumption was much higher than it was to the choice of the significance cutoff. To address this issue, we propose account-

ing for model uncertainty by model averaging via bootstrap aggregation (Bagging) when computing $p$-values or Fdrs. The resulting bagged empirical null (BEN) $p$-values almost always dramatically altered the rank ordering of significant findings. We also examined simple procedures that selecting findings on the basis of both BEN $p$-values and bagged model fit statistics. The idea is to favor significant findings from the model that fits the data better. Not surprisingly, this tended to yield reproducible findings more often because the stability of the inferential model is accounted for.

A potential downside to this approach is the additional computation time and knowledge base needed to bag models and compute the empirical null distribution. However, the changes in the $p$-value ranking were so dramatic in our examples and simulations, that we suggest this step never be skipped. An analysis of the Golub Leukemia data, using our proposed BEN $p$-value approach to rank potential genes of interest, leads us to new discoveries that were confirmed in animal models that had been published before the Golub et al. (1999) study was implemented. This provides some external biological confirmation for the adequacy of our new methods.

### 3.2.1 Background

Large-scale data is often characterized by the simultaneous acquisition of data on many variables or endpoints, in such a manner that the number of observations $n$ is often a magnitude or more than the number of endpoints $N$. GWAS, functional magnetic resonance imaging (fMRI), and Tandem mass spectrometry (MS/MS) are some examples of scientific processes that may yield such large data sets. While the science and relevance of these large-scale endeavors has evolved over the last decade, the popular statistical approaches for handling simultaneous testing of many hypothesis - massive multiple comparison adjustments - has not seen similar growth.

The Bonferroni procedure, popularized in the statistical literature by Dunn (1959), remains the popular choice in many genetic sub-specialties largely due to its simplicity. For example, the commonly used Bonferroni threshold of $5 \times 10^{-8}$ will control the FWER to 0.05 if 1 million gene hypotheses are tested simultaneously (Pe'er et al., 2008). However, this adjustment is associated with a massive loss in statistical power. The Bonferroni procedure was never intended for these large-scale scenarios where individual hypotheses are of interest (Perneger, 1998; Miller, 2012).

Competitors to the Bonferroni procedure have been considered, with some yielding substantial power gains in small-scale settings (Wright, 1992). As an alternative, Benjamini and Hochberg (1995) and separately Shaffer (1995) proposed relaxing con-

trol of the FWER and instead controlling the two-tailed Global False discovery rate (Section 3.7.1 elaborates). Of course, the ranking and selection of top $p$-values for the purpose of identifying findings as scientifically significant and worthy of further study has seen much debate, e.g. see Sham and Purcell (2014).

Storey (2002) proposed a Bayesian interpretation of the Fdr, and Efron (2012$a$) extended this work by proposing an empirical Bayes generalization of the Benjamini and Hochberg procedure: the Empirical Fdr and empirical local Fdr (fdr). The empirical Bayes approach relies on the assumption that a large majority of the individual null hypotheses are in fact null. As a result, the mixture distribution of observed test statistics can be well estimated using a classic empirical Bayes argument. An advantageous property of the Fdr is that the procedure is scalable as a function of the number of tests. Unlike other multiple-testing adjustments the Fdr procedure is not a test of a composite null hypothesis against a single alternative, and therefore can identify individual tests as significant (Mark E Glickman et al., 2015).

Importantly, nearly all adjustment procedures including B-H control of the Fdr or control of the two-tailed Global Fdr under the theoretical null, do not alter the $p$-value ranks; instead they simply adjusts the level at which a given $p$-value is considered "significant". The lone exception is the local false discovery rate, and this is due to the lack of smoothness of the empirical density estimator. The failure of these methods to change the rankings of signifiant endpoints raises the question of whether these procedures have different discrimination capability or are simply re-calibrations of each other. As we will see, the bagging and use of empirical null $p$-values does alter the rankings while maintaining error-rate control, which is an exciting advance.

### 3.2.2 Organization of Paper

The main idea is to use bagged empirical null $p$-values and bagged model fit criteria to identify interesting and statistically significant findings in large-scale contexts. We compare this new approach to the popular approaches in use today. The proposed methods are applied to a Leukemia microarray data set (Golub et al., 1999), studied to select genes which are differentially expressed in Acute Myeloid Leukemia (AML) versus Acute Lymphoblastic Leukemia (ALL), and BEN $p$-values combined with bagged model fit statistics are compared to current methods. Section 3.3 describes the leukemia gene expression data, gives an overview of the proposed Bagged Empirical Null (BEN) $p$-values and Fdr comparators, and provides a detailed algorithm for calculation of BEN $p$-values. Section 3.4 describes an innovative pseudo-

simulation used to assess the proposed methods, and applies the BEN algorithm to the leukemia data. Section 3.5 presents the results and compares the performance of commonly used large-scale inference analysis techniques. Section 3.6 discusses our findings from the case study and provides some concluding remarks.

## 3.3 Materials and Methods

Here we define and describe the computational algorithm for our novel Bagged Empirical Null (BEN) $p$-values, establish simulations to examine its performance and apply it to the famous Gloub leukemia data set (Golub et al., 1999).

### 3.3.1 Leukemia data

The publicly available leukemia data consists of gene expression data for classification of leukemia into two types, Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML), the latter of which has the poorer prognosis. Each type of leukemia responds to different chemotherapies, so correct classification is important to a patients' treatment success. The leukemia data consist of 72 patients, 47 ALL cases and 25 AML cases all genotyped using Affymetrix Hgu6800 chips, resulting in 7129 gene expressions. The most extreme value was excluded, resulting in 7128 gene expressions used to identify interesting genes whose expression levels differ between ALL and AML subjects. Additional covariate information for patients, such as dichotomized age (children vs. adults), drawn sample location (peripheral blood vs. bone marrow), and gender (males vs. females) are available for analyses. The raw data are publicly available online via the 'golubEsets' package in Bioconductor. Gender was missing for 23 patients, and for each simulation missing covariate information was imputed using single imputation and predictive mean matching. As described by Efron (2012$a$), data were normalized in order to eliminate response disparities among microarrays and reduce the impact of outlying values.

### 3.3.2 The Empirical Null

$P$-values and Fdr computations rely on the assumption that the theoretical null density is known. However, when conducting genome-wide studies, this is a strong assumption that is likely false for a significant number of genes or SNPs. Nevertheless, it is highly believable that a very large proportion are null, leading to a situation where the null mixing distribution can be well estimated from the data at hand. As

described by Efron (2012*a*), there are several reasons the theoretical null distribution may fail. (1) Failed mathematical assumptions: It is possible that the test statistics are identically distributed as $N(0,1)$ but not independent, which means the mixing distribution is no longer $N(0,1)$. (2) Correlation across sampling units: Minor experimental defects can manifest themselves as correlation across sampling units. (3) Correlation across cases: Independence among the genes is not needed for valid false discovery rate inference only if we are using the correct null distribution. (4) Unobserved covariates: Additional confounding variables, not accounted for using naive approaches, are perhaps the most common reason why the theoretical null may fail, making it critical to account for model uncertainty.

For an individual gene $i$, the usual null hypothesis is $H_{0i}$: gene $i$ is null. The corresponding $z$-statistic, under normal assumptions, follows a standard normal distribution such that $H_{0i}$: $z_i \sim N(0,1)$. However, by examining the empirical mixing distribution of $z_i$'s, we can assess if the typical theoretical null is supported by the data. We differentiate the individual null densities, $f_0(z_i)$, and non-null densities, $f_1(z_i)$, for $i = 1 \ldots N$ genes. We denote the mixture null comprised of all the $z$-values, $f(z) = \pi_0 f_0(z) + \pi_1 f_1(z)$ where $\pi_0$ and $\pi_1$ are the null and non-null prior probabilities (Efron, 2012*a*).

The empirical null can be estimated by central matching or maximum likelihood estimation, both of which rely on the assumption that some central region of the empirical distribution exists where all genes are not differentially expressed (details provided in Section 3.7.2). Empirical null methods essentially accommodate a "blurry" null hypotheses, in which the uninteresting cases can deviate in minor ways from the theoretical null formulation.

Note that variations of the empirical null are possible depending on the desired modeling assumptions. For example, one can assume normality and estimate both the mean and the variance. Or, alternatively, one could fix the mean at zero and estimate the variance. The latter is intriguing, as it seems natural that empirical null should also be zero centered because the correlation would not affect centering. However, we explored variations like these and found little difference when compared to the classic empirical null algorithms, so the results are not included here.

### 3.3.3 Model Selection and Bootstrap Aggregation

Misspecification of the underlying model can be a real problem for reliable inference. For example, when testing for differentially expressed genes, a common practice

is to perform $N$ $t$-tests, where $N$ is equal to the number of genes. This is equivalent to $N$ univariate linear models, each regressing gene expression on disease status. Often, a logistic regression model is more appropriate, since the goal is to use gene expression to predict leukemia status. In addition, the set of models considered ought to be adjusted for potentially confounding effects, allow for nonlinear effects, and possibly include robust standard errors. All of these things will significantly impact the final inference, but they are often ignored in routine applications.

A way to address many of these concerns is to use bootstrap aggregation (bagging) (Breiman, 1996; Efron, 2012*b*). This method averages estimands of interest from a set of bootstrapped models. Bagging is often applied to situations where the estimator heavily depends on the sample, and perturbations of the sample may lead to significant changes in the statistical measure. Bagging has been shown to improve biological inference on gene sets (Jaffe et al., 2013), however the usefulness of this method has only been investigated for estimating the probability that a significant gene finding will replicate, so our application here is novel.

### 3.3.4 Algorithm

The algorithm for computing BEN $p$-values consists of the following steps:

1. Choose the set of generalized linear models appropriate for the statistical question; including potential covariate adjustments, interactions, non-linear effects, and robust standard errors.

2. Resample the data $B$ times, and fit the entire set of models from Step 1 in each of the $B$ bootstrap resamples, for each of the $N$ genes being tested.

3. Obtain the model fit statistic (e.g., AUC or $R^2$), and the z-statistic corresponding to the hypothesis test of interest (i.e., z-value corresponding to, say, gene expression in the model).

4. Use the $N$ z-values from each model to estimate the Empirical Null distribution, $N(\mu, \sigma)$.

5. Standardize the test statistics to the empirical null distribution. Compute $p$-values and Fdrs under the empirical null as usual.

6. For each of the N endpoints, select the model with the 'best' model fit (e.g., best AIC) and save all statistics of interest for that model.

7. Repeat $B$ times.

8. Take the average of the $B$ statistics for each of the $N$ genes.

9. Report the average empirical null $p$-value and averaged model fit statistic (e.g., AUC). These are the bagged statistics.

10. Flag genes as significant if they meet a multi-dimensional threshold on the BEN $p$-value and bagged fit statistic.

#### *3.3.4.1 Step 3: Calculating z-values for each variable of interest from the p-values associated with the gene covariate in each model*

Calculate the $p$-value in the usual way associated with the gene covariate of interest from each model. From the $p$-value , calculate $z$-values such that $z_i = \Phi(p)$ for $i = 1, \ldots, N$ genes. If genes are allowed to have nonlinear associations, and/or interactions, use the $p$-value from the chunk test, and then transform back to $z$-scale in the same manner.

#### *3.3.4.2 Details on Step 4: Estimating the Empirical Null*

There are several ways to 'bag' the empirical null distribution. The most principled approach is described in the algorithm above, where each model is fit, and an empirical null distribution is estimated from the $z$ values obtained from a single model. If there are $m$ models to choose from, $m$ empirical nulls will be estimated, and $m*N$ statistics, will be calculated in total. We call this the 'principled' approach because the underling working model is held fixed across the genes. Then, for each of the $N$ genes, the set of desired statistics is chosen from the model with the 'best fit' (e.g., lowest AIC). An alternative is this: after the $m$ models are fit, the $z$-values are selected from the 'best fit' model, and then a *single* empirical null is estimated from the $N$ $z$-values. Here the working model is allowed to vary across the genes. Lastly, one could estimate a single empirical null distribution from the combined $m*N$ $z$-values across all models. This approach adds a layer of correlation to the mixture distribution but is not favoring the 'best' fit model. These last two approaches to computing the empirical null distribution have the advantage of only one computation cycle, speeding up the algorithm. There was little difference between the empirical null procedures in practice, and we found that other approaches did not significantly alter the BEN $p$-value rankings. For this paper, we used the 'principled' approach, and the two alternative methods are explored in Section 3.7.3 of the Supplementary Material since they appear to be viable shortcuts that merit further exploration.

## 3.4 Simulation and Implementation

### 3.4.1 Pseudo-Simulation

Large-scale data is often defined by its rich and complex correlation structure, which often extends non-uniformly over columns, rows, and clusters. As such, it is very hard to simulate realistic large-scale data from scratch. Because of this, we took a "Pseudo-Simulation" approach. Our idea is to take the subset of Gloub null genes and remove any mean effects via a highly parameterized regression including several covariates and interactions. The matrix of residuals retains the complex correlation structure, upon which we add back fixed effects via an assumed regression model (this is our simulation engine). We refer to this simulation as 'Pseudo' because we are not generating *new* error structures. In effect, we get a simulation using a real-world error structure and thus retain the complexity of these large-scale data that is often critical to the evaluation of novel methods.

Specifically, we did the following: In the original unchanged leukemia data, fit univariate linear models for each gene expression, and then took the subset of genes whose $p$-value associated with the AML/ALL coefficient was greater than 0.3. This resulted in 3172 genes whose gene expressions we assumed have little to no association with leukemia type. We then randomly selected 30 genes within this subset, computed the residuals after regressing over as many effects as we could, and induced 10 genes to have a strong association, 10 genes to have a moderate association, and 10 genes to have a weak association between leukemia type and gene expression.

To selectively adjust certain gene expression levels and leukemia relationships, new expression data was estimated by adding new fitted values,

$$\begin{aligned}\{\texttt{Gene Expression}^{*}\}_j &= \{\boldsymbol{X}\boldsymbol{\beta}^{*}\}_j \\ &= \beta^{*}_{0j} + \beta^{*}_{1j}\{\texttt{Leukemia Type}\} + \beta^{*}_{2j}\{\texttt{Gender}\} + \beta^{*}_{3j}\{\texttt{Sample}\} \\ &\quad + \beta^{*}_{4j}\{\texttt{Leukemia Type x Gender}\} \\ &\quad + \beta^{*}_{5j}\{\texttt{Leukemia Type x Sample}\} \\ &\quad + \beta^{*}_{6j}\{\texttt{Gender x Sample}\}\end{aligned}$$

to the original residuals, $\hat{\epsilon}_j = \{\texttt{Gene Expression}\}_j - \{\boldsymbol{X}\hat{\boldsymbol{\beta}}\}_j$, for each of the induced genes $j = 1 \ldots 30$. This results in

$$\{\texttt{Gene Expression}^{\text{NEW}}\}_j = \{\texttt{Gene Expression}^{*}\}_j + \hat{\epsilon}_j$$

This allows us to specify any level of complexity in the model, while preserving

the original error structure from the data. Each set of 10 strong, moderate, and weak genes comprised of 2 genes induced to be associated with just leukemia type ($\beta_1^*$), 2 genes induced to have a gender and a leukemia type by gender interaction ($\beta_2^*$,$\beta_4^*$), 2 genes induced to have a leukemia type and a leukemia type by gender interaction ($\beta_1^*$,$\beta_4^*$), 2 genes induced to have a leukemia type and a leukemia type by sample interaction ($\beta_1^*$,$\beta_5^*$), and 2 genes induced to have a leukemia type, leukemia type by gender, and a leukemia type by sample interaction ($\beta_1^*$,$\beta_4^*$,$\beta_5^*$). Strong gene associations, moderate gene associations, and weak gene associations were induced to have $\boldsymbol{\beta^*}$ values 7, 4, and 2 times the original leukemia type coefficients respectively. These values were chosen such that the multiplicative effect of the coefficient outweighed their corresponding empirical standard errors.

The pseudo-simulated data was then normalized, and the BEN algorithm was implemented. The pseudo-simulations were conducted under linear regression models because (1) they provide a direction comparison to methods based on single univariate $t$-statistics, common in the literature and (2) their residuals are well defined when compared to logistic regression. We expect the results would extend to logistic regression, and our primary example uses logistic regression. The following linear models were considered for the BEN algorithm:

(1) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\}$

(2) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\}$

(3) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Gender}\}$

(4) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\} + \beta_3\{\texttt{Gender}\}$

(5) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\} + \beta_3\{\texttt{Gender}\} + \beta_4\{\texttt{Leukemia Type x Sample}\}$

(6) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\} + \beta_3\{\texttt{Gender}\} + \beta_4\{\texttt{Leukemia Type x Gender}\}$

(7) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\} + \beta_3\{\texttt{Gender}\} + \beta_4\{\texttt{Sample x Gender}\}$

(8) $E[\texttt{Gene Expression}] = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\} + \beta_3\{\texttt{Gender}\} + \beta_4\{\texttt{Leukemia Type x Gender}\} + \beta_5\{\texttt{Leukemia Type x Sample}\}$

### 3.4.2 Logistic Regression Models Applied to the Leukemia Data

Two separate implementations of the BEN algorithm were performed for the leukemia data using both linear regression and logistic regression. Since the set of logistic models are more appropriate for answering the scientific question at hand, these are the models discussed here. The linear regression results are included in Section 3.7.4 of the Supplement.

7128 regression models were fit, and the models included all combinations of the covariates gene expression (continuous), site of sample (peripheral blood vs. bone marrow), and gender (male vs. female). In two of the regression models, gene expression was modeled flexibly using restricted cubic splines, as gene expression does not necessarily have a linear relationship with leukemia type.

The following models were considered for each of the 500 bootstrap aggregations:

(1) $logit(E[\texttt{Leukemia Type}]) = \beta_0 + \beta_1\{\texttt{Gene Expression}\}$

(2) $logit(E[\texttt{Leukemia Type}]) = \beta_0 + \beta_1\{\texttt{Gene Expression}\} + \beta_2\{\texttt{Sample}\}$

(3) $logit(E[\texttt{Leukemia Type}]) = \beta_0 + \beta_1\{\texttt{Gene Expression}\} + \beta_2\{\texttt{Gender}\}$

(4) $logit(E[\texttt{Leukemia Type}]) = \beta_0 + \beta_1\{\texttt{Gene Expression}\} + \beta_2\{\texttt{Sample}\} + \beta_3\{\texttt{Gender}\}$

(5) $logit(E[\texttt{Leukemia Type}]) = \beta_0 + \beta_{1,2}\{rcs(\{\texttt{Gene Expression}\}, 3)\}$

(6) $logit(E[\texttt{Leukemia Type}]) = \beta_0 + \beta_{1,2}\{rcs(\{\texttt{Gene Expression}\}, 3)\} + \beta_3\{\texttt{Sample}\} + \beta_4\{\texttt{Gender}\}$

For models (1) through (4) the test of $\beta_1$ provides $N = 7128$ Wald $z$-statistics used in the BEN algorithm. For models (5) and (6) the $p$-value from the chunk test of $\beta_1 = \beta_2 = 0$, the terms associated with the non-linear effect of gene expression, is transformed back to the $z$-statistic as described in 3.3.4.1. In order to compare results to more traditional methods, univariate logistic regression models were fit, and the $N = 7128$ $z$-values were used to calculate $p$-values and empirical null $p$-values. All $p$-values were adjusted using the Bonferroni and Benjamini-Hochberg procedures for comparison.

## 3.5 Results

### 3.5.1 Pseudo-Simulation

Univariate linear regression models were fit for every gene expression using the manipulated data: $E\{\texttt{Gene Expression}\}_i = \beta_{0i} + \beta_{1i}\{\texttt{Leukemia Type}\}$ for $i = 1 \ldots 3172$. The unadjusted, Bonferroni, and Benjamini-Hochberg $p$-values, along with each model $R^2$ was computed. Table 3.1 presents the results of the pseudo-simulation after performing 250 simulations of 250 bootstrap aggregations.

Investigating all subsets of $p$-values $\leq 0.05$, the BEN $p$-values found more truly associated genes than the Bonferroni corrected $p$-values from the univariate models. This increase in power comes with the cost of a slight increase in empirical False Discovery Rate

(FDR) for the BEN $p$-values compared to the Bonferroni corrected $p$-values. Interestingly, we estimated 115 genes from the univariate model to have $p$-values $\leq 0.05$, and only 16 of those had true disease status/gene expression relationships. Bagging $p$-values reduces the total number genes with $p$-values $\leq 0.05$ to approximately 105, with about 20 of those being true relationships. Our results indicate that bagging alone yields a desirable increase in power and decrease in FDR apart from the empirical null methodology. BEN $p$-values have a comparable power to unadjusted and empirical null $p$-values, but they result in a significantly reduced FDR.

When using the multi-dimensional metric, $p$-values $\leq 0.05$ and $R^2 \geq 0.5$ (from either the univariate models or the bagged models), the BEN $p$-values and Bagged $p$-values have the largest power and have similar FDR control. When selecting genes based on $p$-value alone, EN, unadjusted, and Bagged $p$-values have similar properties (with Bagged $p$-values having the greatest power), however when selecting genes sets based on this dual metric the usefulness of bagging and empirical null procedures is apparent.

It may be tempting to find the traditional Bonferroni $p$-values still favorable since the empirical FDR is low (especially when using both the $p$-value and model fit statistic), however the Bonferroni $p$-values mostly select the genes whose true model is univariate (i.e. 4 of the 6 models where Leukemia Type and Gene expression, $\beta_1^*$, was induced to have a relationship). So the Bonferroni adjustment might be considered ultra-conservative if the underling model is correctly specified, but if not the adjustment appears to overly penalize. All other models were not selected by the Bonferroni $p$-values. Bagging allows for the results to favor the 'best fit' model, which is more likely to be close to the correct but unknown model, and this results in more true discoveries.

Note that in the pseudo-simulation the Empirical Null (EN) $p$-values tended to have more false discoveries than the unadjusted $p$-values. We expect this to be true if we assume 3142 genes are truly null, then we would expect 157 false discoveries by chance alone, a result in line with the EN $p$-values. The combination of bagging and empirical null procedures is more likely to select the truly differentially expressed genes (Table 3.1). It also has a more desirable FDR/Power tradeoff, and therefore should be preferred over other $p$-value adjustment techniques.

### 3.5.2 Application to Leukemia Data

#### *3.5.2.1 Naive Approaches to Analysis of Leukemia Data*

For comparison, the usual analyses were performed using the leukemia data. Simple Logistic regression models were fit, $logit(E[\texttt{Leukemia Type}_i]) = \beta_0 + \beta_1\{\texttt{Gene Expression}_i\}$ for $i = 1 \ldots 7128$ genes. The conventional analysis is shown in Figure 3.1, where the unadjusted p-values, Bonferroni corrected $p$-values, and global two-tailed B-H (Fdr) $p$-values are presented, sorted from smallest to largest. The black dashed line represents the same

| | True Discoveries | | | | False Discoveries | Power | FDR |
|---|---|---|---|---|---|---|---|
| | Strong | Moderate | Weak | All | Ordered ↓ Type I Error* | #True Dis./#Induced Genes | #False Dis./#Total Dis. |
| ***p*-value $\leq$ 0.05** | | | | | | | |
| EN | 8 (7,8) | 6 (5,7) | 3 (2,4) | 17 (16,19) | 138 (136,139) | 0.57 (0.53,0.63) | 0.89 (0.88,0.9) |
| Unadjusted | 7 (6,8) | 6 (5,7) | 3 (2,4) | 16 (15,18) | 99 (99,100) | 0.53 (0.5,0.6) | 0.86 (0.85,0.87) |
| Bagged | 9 (8,10) | 8 (7,9) | 4 (3,5) | 21 (19,22) | 85 (74,101) | 0.67 (0.63,0.73) | 0.81 (0.78,0.84) |
| **BEN** | 8 (7,9) | 6 (5,7) | 2 (1,2) | 16 (14,17) | 12 (11,13) | 0.53 (0.47,0.57) | 0.43 (0.4,0.46) |
| Bagged B-H | 8 (7,9) | 6 (5,7) | 2 (1,2) | 16 (14,17) | 11 (10,13) | 0.53 (0.47,0.57) | 0.42 (0.39,0.46) |
| EN B-H | 5 (4,7) | 3 (2,4) | 0 (0,1) | 8 (8,11) | 8 (8,8) | 0.3 (0.27,0.37) | 0.47 (0.42,0.5) |
| B-H | 5 (4,6) | 3 (2,4) | 0 (0,1) | 8 (7,10) | 8 (8,8) | 0.3 (0.23,0.33) | 0.47 (0.44,0.53) |
| EN Bonferroni | 5 (4,6) | 2 (2,4) | 0 (0,1) | 7 (6,9) | 5 (5,5) | 0.23 (0.2,0.3) | 0.42 (0.36,0.45) |
| Bonferroni | 4 (3,5) | 2 (1,3) | 0 (0,0) | 6 (5,8) | 3 (3,3) | 0.23 (0.17,0.27) | 0.3 (0.27,0.38) |
| Bagged Bonferroni | 7 (6,8) | 4 (3,5) | 0 (0,0) | 11 (9,12) | 2 (1,2) | 0.37 (0.3,0.4) | 0.14 (0.09,0.17) |
| BEN B-H | 6 (5,7) | 2 (1,3) | 0 (0,0) | 8 (7,10) | 0 (0,0) | 0.27 (0.23,0.33) | 0 (0,0) |
| BEN Bonferroni | 5 (4,6) | 1 (0,2) | 0 (0,0) | 6 (5,7) | 0 (0,0) | 0.2 (0.17,0.23) | 0 (0,0) |
| ***p*-value $\leq$ 0.05 & $R^2 \geq$ 0.5** | | | | | | | |
| Bagged | 7 (6,8) | 4 (3,5) | 0 (0,1) | 11 (9,12) | 2 (1,2) | 0.37 (0.3,0.4) | 0.12 (0.08,0.15) |
| **BEN** | 7 (6,8) | 4 (3,5) | 0 (0,1) | 11 (9,12) | 1 (1,1) | 0.37 (0.3,0.4) | 0.08 (0.08,0.1) |
| Bagged B-H | 7 (6,8) | 4 (3,5) | 0 (0,1) | 11 (9,12) | 1 (1,1) | 0.37 (0.3,0.4) | 0.09 (0.08,0.11) |
| Bagged Bonferroni | 7 (6,8) | 3 (2,4) | 0 (0,0) | 10 (9,11) | 1 (1,1) | 0.33 (0.3,0.37) | 0.09 (0.08,0.1) |
| BEN B-H | 6 (5,7) | 2 (1,3) | 0 (0,0) | 8 (7,10) | 0 (0,0) | 0.27 (0.23,0.33) | 0 (0,0) |
| BEN Bonferroni | 5 (4,6) | 1 (0,2) | 0 (0,0) | 6 (5,7) | 0 (0,0) | 0.2 (0.17,0.23) | 0 (0,0) |
| EN | 2 (1,3) | 0 (0,1) | 0 (0,0) | 2 (1,3) | 0 (0,0) | 0.07 (0.03,0.1) | 0 (0,0) |
| Unadjusted | 2 (1,3) | 0 (0,1) | 0 (0,0) | 2 (1,3) | 0 (0,0) | 0.07 (0.03,0.1) | 0 (0,0) |
| EN B-H | 2 (1,3) | 0 (0,1) | 0 (0,0) | 2 (1,3) | 0 (0,0) | 0.07 (0.03,0.1) | 0 (0,0) |
| B-H | 2 (1,3) | 0 (0,1) | 0 (0,0) | 2 (1,3) | 0 (0,0) | 0.07 (0.03,0.1) | 0 (0,0) |
| EN Bonferroni | 2 (1,3) | 0 (0,1) | 0 (0,0) | 2 (1,3) | 0 (0,0) | 0.07 (0.03,0.1) | 0 (0,0) |
| Bonferroni | 2 (1,3) | 0 (0,1) | 0 (0,0) | 2 (1,3) | 0 (0,0) | 0.07 (0.03,0.1) | 0 (0,0) |

Table 3.1: Results of the Pseudo-simulation with 250 replications and 250 bootstrap aggregations. The Median and (IQR) are presented here. There are 3172 genes tested in total, with 10 genes induced to have a strong, 10 genes induced to have a moderate, and 10 genes induced to have a weak association between Gene Expression and Leukemia Type. True Discoveries are those genes found which had a true induced relationship between Gene Expression and Leukemia type. False Discoveries are those genes which had no induced relationship between gene expression and leukemia type. The FDR is the number of false discoveries out of all discovered genes. The Power is the number of true differentially expressed genes out of the 30 genes induced to have a true relationship between gene expression and leukemia type. *$p$-value type is ordered by descending Type I Error (Number of False discoveries/Truly Null Genes). BEN: Bagged Empirical Null, B-H: Benjamini-Hochberg, EN: Empirical Null.

$p$-values calculated under the empirical $N(0.13, 1.70)$ null using the maximum likelihood method (slight mean shift with large variance inflation). The solid blue line is the B-H $p$-values (the two-tailed global Fdrs) calculated under the theoretical null, and the solid red line is the Bonferroni corrected $p$-values. The EN $p$-values are an interesting compromise between the strict Bonferroni and more relaxed B-H $p$-value adjustments. For these data, there are 391 EN $p$-values $\leq 0.05$, compared to 7 Bonferroni $p$-values and 971 B-H $p$-values. EN $p$-values appear to self-adjust, providing a more desirable error-rate tradeoff than Bonferroni or B-H $p$-values, without having to do post-hoc adjustments.

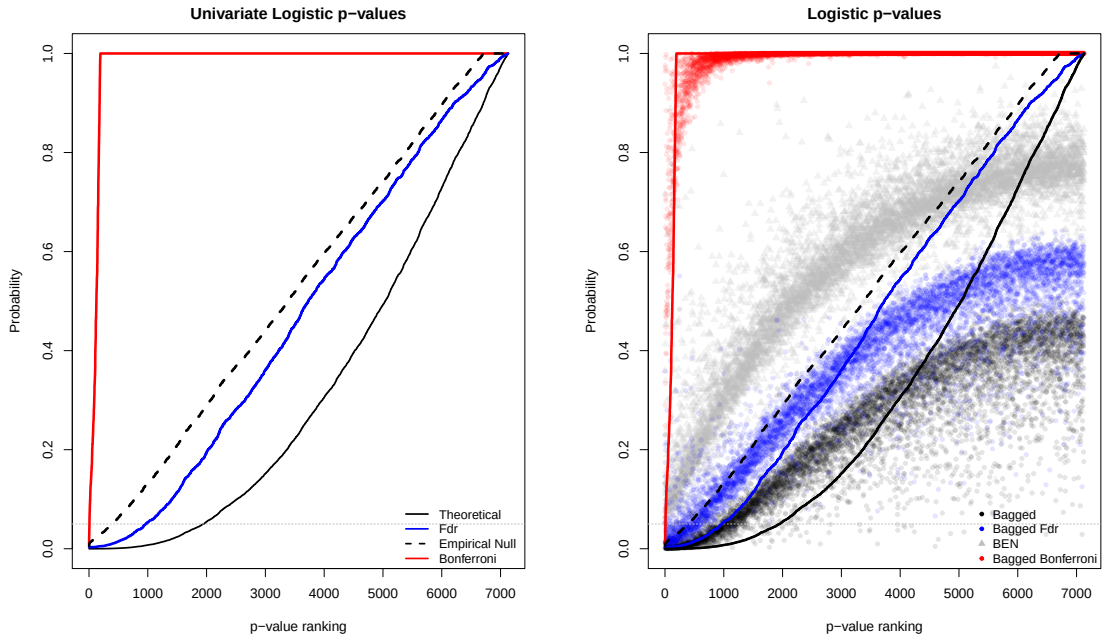



Figure 3.1: Left Figure: The black solid line represents the unadjusted $p$-values associated with the gene expression coefficient from each of the 7128 univariate logistic regression models. The $p$-values are sorted from smallest to largest. The black dashed line represents the p-values recalibrated using the empirical $N(0.13, 1.70)$ null derived from the maximum likelihood method. The solid blue line represents The Benjamini-Hochberg (B-H) $p$-values, equivalent to the two-tailed global false discovery rate (Fdr) calculated under the theoretical null. The solid red line represents the Bonferroni corrected p-values under the theoretical null. Right Figure: The black dots are the bagged $p$-values calculated under the theoretical null. The gray triangles are the bagged empirical null (BEN) $p$-values. The blue dots are the bagged B-H $p$-values calculated under the theoretical null. The red dots are the bagged Bonferroni $p$-values calculated under the theoretical null. All points are sorted based on the original unadjusted $p$-values.

The bagged theoretical null $p$-values, and bagged theoretical null B-H $p$-values are shrunk towards 0.5, compared to the non-bagged counterparts (Figure 3.1: right plot). Since $p$-values follow a Uniform(0,1) distribution under the null hypothesis, the mean of the null $p$-values is approximately 0.5 (Section 3.7.5 of the Supplement), so this behavior is expected. The display of bagged results is ordered by the original unadjusted $p$-value magnitude.

*3.5.2.2 Previous reports on differentially expressed genes for leukemia*

There is no known list or compilation of differentially expressed genes related to the leukemia. A literature search yielded 5 independent and highly cited papers that present gene findings, specific to leukemia, based on their analyses. We reviewed and compared their findings and compared these results to the genes the BEN algorithm ultimately selected. Golub et al. (1999); Lee et al. (2003); Bø and Jonassen (2002); Zhou et al. (2004); Tong and Schierz (2011) use a variety of statistical methods to classify or predict gene expressions associated with leukemia type. Note there is a lot of variability in the reported genes between studies.

Golub et al. (1999) reports 50 top genes (25 genes most differentially expressed in AML patients, and 25 genes most differentially expressed ALL patients). Lee et al. (2003) reports the 27 genes which are the best classifiers of ALL vs. AML using a Bayesian variable selection approach. Bø and Jonassen (2002) report the top 50 genes for ALL/AML class separation using their all pairs subset selection procedure. Zhou et al. (2004) report the top 20 important genes selected using their proposed Bayesian gene selection algorithm. Tong and Schierz (2011) report the 22 genes selected by their genetic algorithm-neural network. For comparison, we also compare the top 20 smallest univariate logistic regression $p$-value and the 7 Bonferroni adjusted $p$-values $\leq 0.05$ associated with leukemia type.

The BEN algorithm resulted in 22 genes who met the dual criterion of BEN $p$-value $\leq 0.1$ and bagged AUC $\geq 0.90$ (Table 3.2). Of this set, the gene with the smallest $p$-value is M83667 a known transcription factor of the NF-IL6-beta protein. A study by Natsuka et al. (1992), that predates the leukemia study, found there was a drastic increase in expression of NF-IL6 messenger RNA (mRNA) during the differentiation to a macrophage lineage in mouse myeloid leukemia cells using mouse models. The original Golub et al. (1999) paper does not report this gene as one of top 50 genes differentially expressed in AML/ALL patients. Lee et al. (2003) and Zhou et al. (2004) also fail to identify this gene. Given the prior prominence of M83667, which appears to be a major omission.

*3.5.2.3 Comparison of p-values*

We can visualize the singular effects of just bagging $p$-values, of just recalibrating the $p$-values to the empirical null distribution, and the additive effect of bagging and empirical null procedures in Figure 3.2. Compared to the original univariate $p$-values from the univariate logistic regression models, the bagged $p$-values are shrunk towards 0.5. Those $p$-values which were most likely null had corresponding bagged $p$-values that were shrunk towards 0.5, and the original $p$-values which were closer to 0 had larger bagged $p$-values. Notice that the empirical null $p$-values are a one-to-one function of the original $p$-values, and incorporating the empirical null amounts to shifting the original $p$-values away from 0. The BEN $p$-values in Figure 3.2 show a left horizontal and vertical shift of the $p$-values compared to the original

| Gene | BEN p-value | AUC | Golub | Bo | Lee | Tong | Zhou | Univariate $p$ | Bonferroni $p$ |
|---|---|---|---|---|---|---|---|---|---|
| M83667 | 0.057 | 0.922 | | X | | X | | X | X |
| M33195 | 0.064 | 0.912 | | | | | | X | |
| U22376 | 0.066 | 0.904 | X | | X | | X | X | |
| X97267 | 0.069 | 0.908 | | | | X | | | |
| M62762 | 0.073 | 0.927 | X | | | | | | |
| X52056 | 0.075 | 0.913 | | X | | X | | | |
| X78669 | 0.076 | 0.902 | | | | | | | |
| M31211 | 0.076 | 0.927 | X | X | | | X | X | |
| M22960 | 0.077 | 0.922 | | | X | X | | | |
| M19507 | 0.078 | 0.932 | | X | X | X | | | |
| Z15115 | 0.080 | 0.954 | X | X | | | X | X | |
| U41635 | 0.082 | 0.902 | | | | | | | |
| D14664 | 0.082 | 0.910 | | | | | | | |
| L09717 | 0.082 | 0.909 | | | | | | | |
| X17042 | 0.084 | 0.910 | X | X | | | | | |
| U16954 | 0.084 | 0.922 | | X | | | | | |
| J03801 | 0.088 | 0.901 | | | | | | | |
| M32304 | 0.090 | 0.906 | | | | | | | |
| M93056 | 0.092 | 0.907 | | | | | | | |
| U77948 | 0.095 | 0.911 | | X | | | | | |
| U57721 | 0.097 | 0.913 | | | | | | | |
| X07743 | 0.098 | 0.917 | | | X | | | | |

Table 3.2: The 22 leukemia data genes with bagged empirical null (BEN) $p$-values $\leq 0.1$ and bagged AUC $\geq 0.9$. Genes are sorted by (BEN) $p$-values from smallest to largest. We reviewed Golub et al. (1999); Lee et al. (2003); Bø and Jonassen (2002); Zhou et al. (2004); Tong and Schierz (2011), and an X is placed where the authors reported a gene we also selected. The univariate $p$-values are derived from the univariate logistic regressions of the original data, and the resulting top 20 $p$-values are compared to the BEN results. The Bonferroni correction was applied to the univariate model $p$-values, and all Bonferroni $p$-values $\leq 0.05$ are compared to the BEN results.

$p$-values. We investigated the effects of just bootstrapping the $p$-values and include those results in Section 3.7.6 of the Supplement. We compared the top 50 $p$-values reported by Golub et al. (1999), the 7 $p$-values with a Bonferroni adjustment $\leq 0.05$, and NF-IL6 beta protein mRNA, the top gene selected by the BEN algorithm 2-dimensional criterion in Table 3.2.

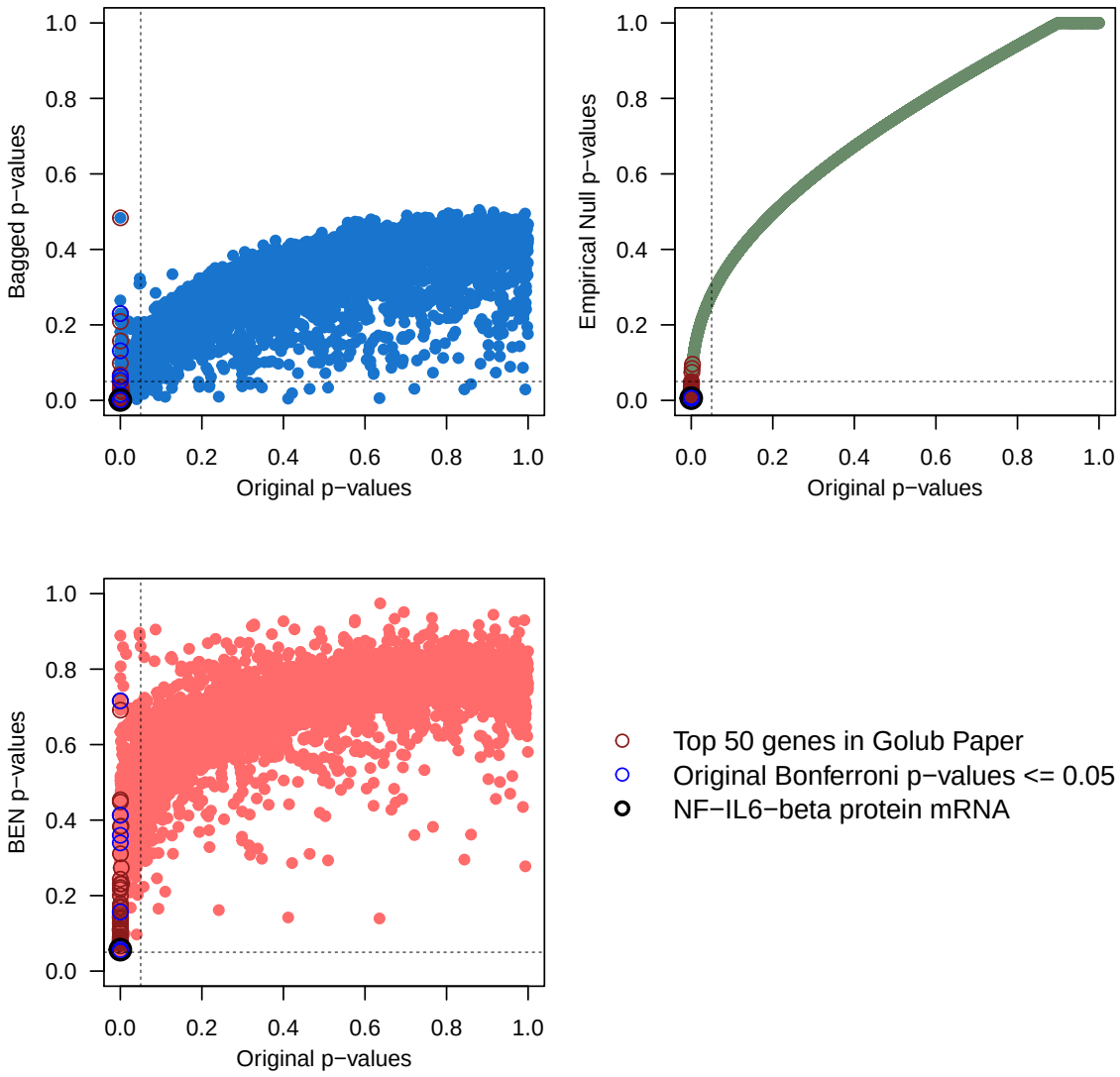



Figure 3.2: The original $p$-values from the univariate logistic regressions are plotted against the Bagged $p$-values, Empirical $N(0.13, 1.70)$ Null recalibrated $p$-values, and the Bagged Empirical Null (BEN) $p$-values. The open red circles are the 50 $p$-values Golub et al. (1999) reported as differentially expressed, and the open blue circles are the Bonferroni adjusted $p$-values $\leq 0.05$ from the original univariate logistic regressions. The bold black open circle is the top gene selected by the two-dimensional $p$-value and AUC criterion of the BEN algorithm, which corresponds the NF-IL6-beta protein mRNA gene.

#### *3.5.2.4 Visualizing the 2-dimensional criteria*

We propose using a dual criterion of $p$-values and model fit statistics (AUC) for selecting a subset of genes for further investigation. In Figure 3.3 we visualize the relationship between $p$-values and AUC from the leukemia data, and it is clear that even small $p$-values can have a low corresponding AUC. The AUC cutoff criterion (horizontal dashed lines) could be adjusted up or down to further restrict or increase the set of genes for study.

We have chosen 0.9, but this threshold (or the similar $R^2$ threshold in Table 3.1) could be selected specifically to choose a subset whose size is feasible for further study.

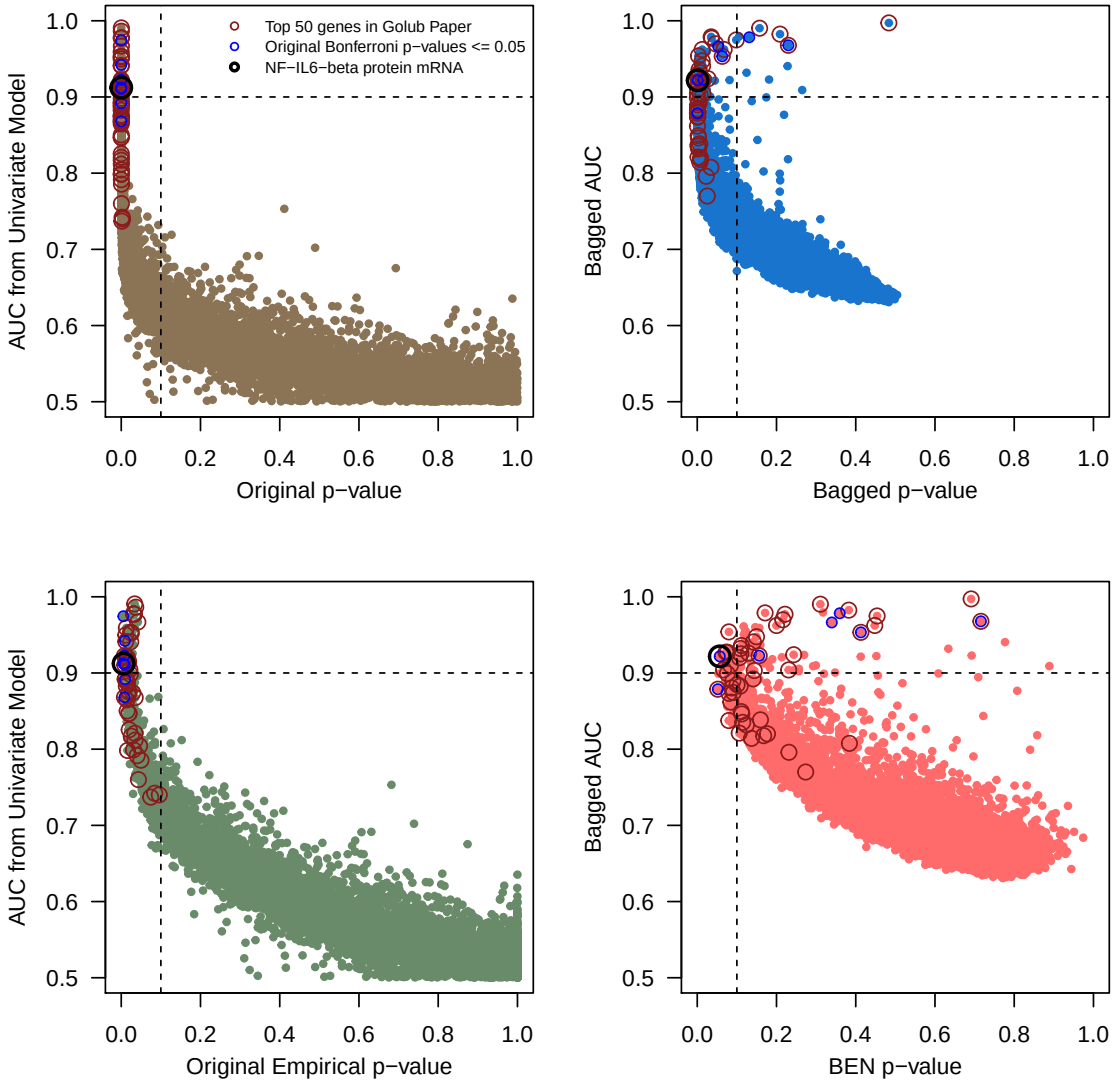



Figure 3.3: The original $p$-values from the univariate logistic regressions and Empirical $N(0.13, 1.70)$ Null recalibrated $p$-values, are plotted against the corresponding model AUC. The Bagged $p$-values, and the Bagged Empirical Null (BEN) $p$-values are plotted against the Bagged AUC. The open red circles are the 50 $p$-values Golub et al. (1999) reported as differentially expressed, and the open blue circles are the Bonferroni adjusted $p$-values $\leq 0.05$ from the original univariate logistic regressions. The bold black open circle is the top gene selected by the two-dimensional $p$-value and AUC criterion of the BEN algorithm, which corresponds the NF-IL6-beta protein mRNA gene.

## 3.6 Discussion

We have developed a Bagged Empirical Null strategy for obtaining $p$-values that do not need to be adjusted post-hoc when performing large scale inference. Based on our

simulations and examples, combining these $p$-values with bagged model fit statistics appears to be advantageous as it tends to select truly differentially expressed genes more often than traditional $p$-value corrections. This direct approach to $p$-value calculations may be more useful when the goal is to identify the maximum number of truly significant genes, while controlling the FDR and maximizing the power.

Our motivation comes from gene expression microarray data analysis, where the same set of genes does not seem to be reproducible across different experiments of statistical methodologies. In this unique setting, it is highly impractical to pre-specify a different model for every gene, and so for situations where a large number of tests are to be evaluated (as in genetic and other high-dimensional data), current methodology can benefit from bootstrap aggregating procedures.

We have applied our method to pseudo-simulated gene expression data and the original leukemia data, and have demonstrated that our method is superior by having the most desirable Type I/Type II error tradeoff. A strength of our approach is the ability to consider any set of models (parameters, link function, non-parametric model) during bagging, as well as not being confined to the conventional theoretical null as the testing distribution. By incorporating bootstrapping, model selection, and empirical null procedures, the BEN algorithm has the advantage of using multi-dimensional gene selection metrics, beyond the single adjusted $p$-value traditionally used. Consequently, the BEN algorithm will lead to more robust and reproducible biological findings.

## 3.7 Appendix A. Description of Supplementary Materials

The reader is referred to the following supplementary materials for technical appendices, additional results of the BEN algorithm using linear models when applied to the leukemia data.

### 3.7.1 Remark A: Evidential quantities

After observing data, instead of worrying about proper control the pre-test probability of collecting misleading data, one should instead worry about the probability that the observed data are misleading. The analogue is a shift from a focus on sensitivity/specificity to positive/negative predictive value (Blume, 2011). Along these lines, Blume (2011) notes that the measure of the strength of evidence (the evidence metric), the probability that a study design will generate misleading evidence (the error probabilities), and the probability that observed data are misleading (false discovery rates) are three inferential concepts that should not be confused. Efron argues that once the data are collected, say when analyzing genetic microarray data, it is more relevant to control the number of forthcoming false positive results (i.e., control the Fdr at the expense of the Type I error rate), and that the general null hypothesis (that all the null hypotheses are true) is rarely of interest (Efron, 2012*a*).

### 3.7.2 Remark B: Estimating the Empirical Null Distribution

Central matching fits a normal distribution to central percentage of the z-values associated with each individual hypothesis test, and estimates a mean, variance, and proportion of null hypotheses. Note that slight irregularities in the central histogram can derail this method. Maximum likelihood estimation fits a truncated normal distribution to a region assumed to contain only null results, and multiples this distribution by the probability that those genes are null. The product of two exponential families can be maximized, and values for the mean, standard deviation and proportion of nulls can be estimated. The MLE method is less susceptible to central irregularities but comes at the cost of possible increased bias. (Efron, 2012*a*).

### 3.7.3 Remark C: Alternative ways to Bag the Empirical Null Distribution

There are several ways to 'bag' the empirical null distribution. The most principled approach is described in the BEN algorithm, where each model is fit, and an empirical null distribution is estimated from the $z$ values obtained from a single model. If there are $m$ models to choose from, $m$ empirical nulls will be estimated, and $m*N$ statistics, will be calculated in total. We call this the 'principled approach' because the underlying working

model is held fixed across the genes. Then, for each of the $N$ genes, the set of statistics is chosen from the model with the 'best fit' (e.g., lowest AIC). In the figures below this will be referred to as the 'Principled Empirical Null'. An alternative is this: after the $m$ models are fit, the $z$-values are selected from the 'Best fit' model, and then a *single* empirical null is estimated from the $N$ $z$-values. Here the working model is allowed to vary across the genes. In the below figures, this is referred to as the 'Mixed Empirical Null'. Lastly, one would estimate a single empirical null distribution from the combined $m*N$ $z$-values across all models. This approach adds a layer of correlation to the mixture distribution but is not favoring the 'Best' fit model. In the below figures this method will be referred to as the 'Large Correlated Empirical Null'. These last two approaches to computing the empirical null distribution have the advantaged of need to only one computation cycle, greatly speeding the algorithm at the cost of inducing some additional correlation.

The Principled Empirical Null and the Large Correlated Empirical Null behave very similarly. However, the Mixed Empirical Null has BEN $p$-values shifted towards one compared to the original univariate $p$-values. There was little difference between the empirical null procedures in practice, and we found the other approaches did not significantly alter the BEN $p$-value rankings. The properties of the Large Correlated Empirical Null should be explored as a way to save computation time of $m$ empirical nulls in the BEN algorithm.

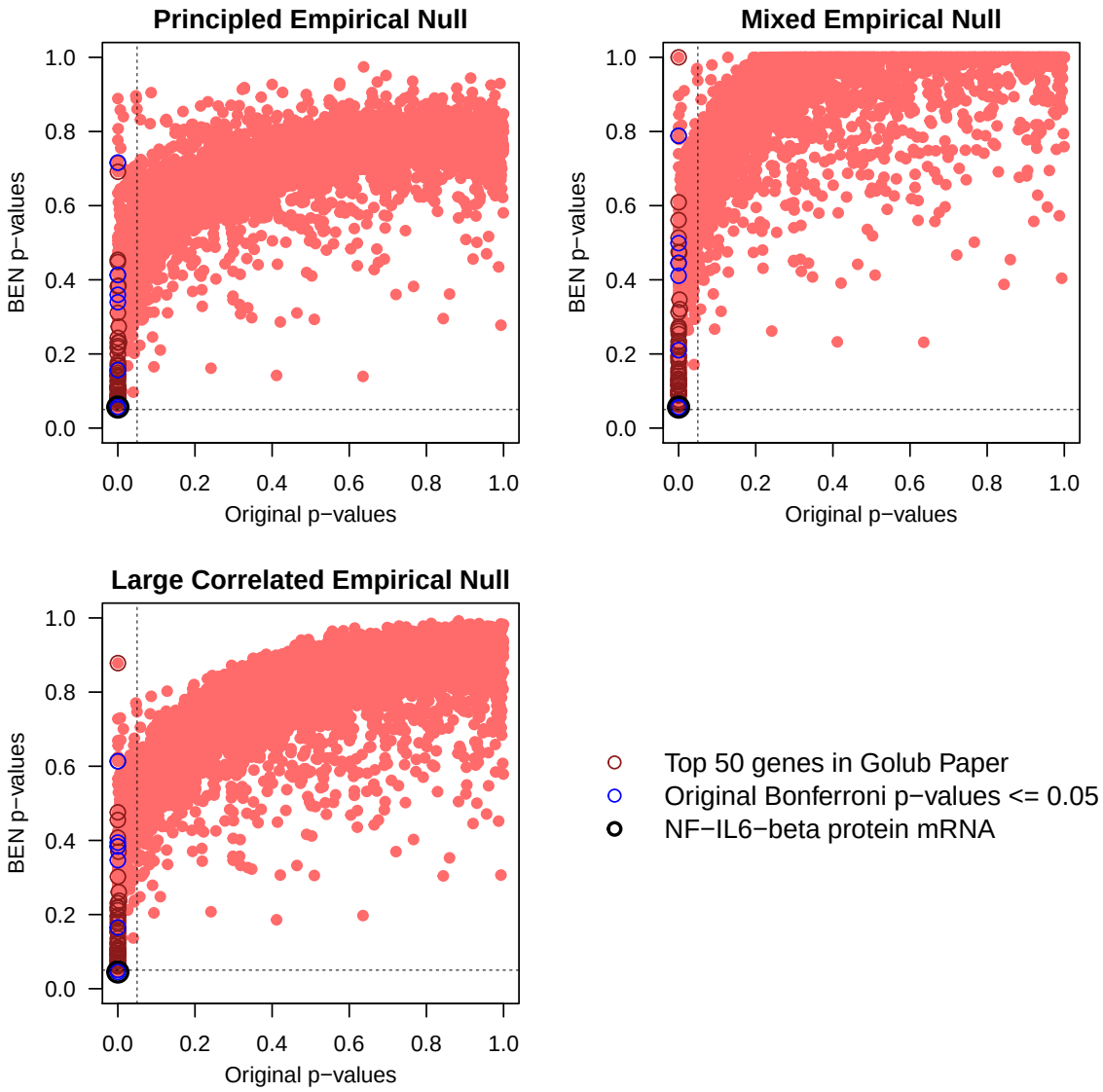



Figure 3.4: The original $p$-values from the univariate logistic regressions are plotted against the Bagged $p$-values, Empirical $N(0.13, 1.60)$ Null recalibrated $p$-values, and the Bagged Empirical Null (BEN) $p$-values. The open red circles are the 50 $p$-values Golub et al. (1999) reported as differentially expressed, and the open blue circles are the Bonferroni adjusted $p$-values $\leq 0.05$ from the original univariate logistic regressions. The bold black open circle is the top gene selected by the two-dimensional $p$-value and AUC criterion of the BEN algorithm, which corresponds the NF-IL6-beta protein mRNA gene.

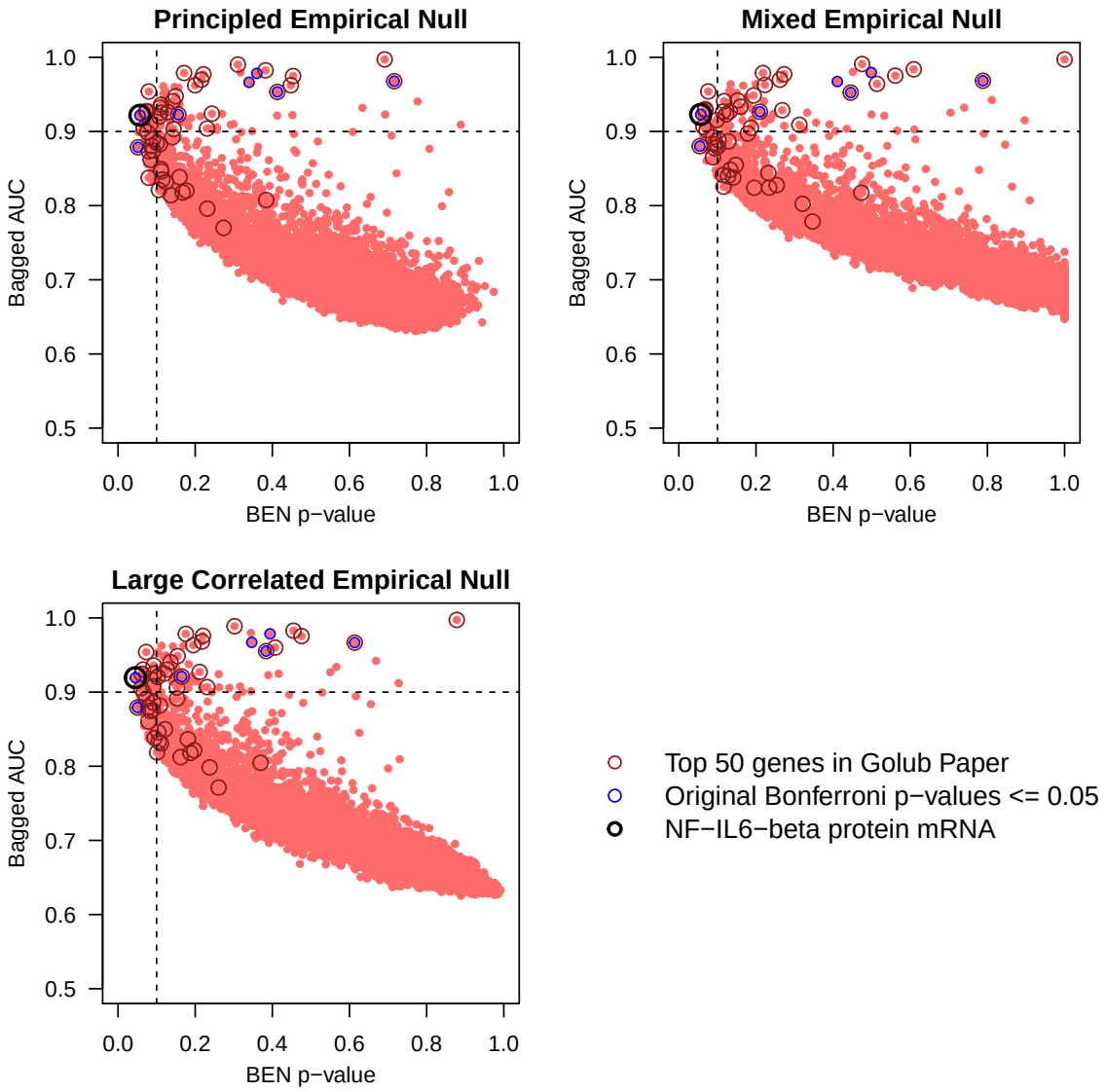



Figure 3.5: The original $p$-values from the univariate logistic regressions and Empirical $N(0.13, 1.60)$ Null recalibrated $p$-values, are plotted against the corresponding model AUC. The Bagged $p$-values, and the Bagged Empirical Null (BEN) $p$-values are plotted against the Bagged AUC. The open red circles are the 50 $p$-values Golub et al. (1999) reported as differentially expressed, and the open blue circles are the Bonferroni adjusted $p$-values $\leq 0.05$ from the original univariate logistic regressions. The bold black open circle is the top gene selected by the two-dimensional $p$-value and AUC criterion of the BEN algorithm, which corresponds the NF-IL6-beta protein mRNA gene.

### 3.7.4 Remark D:Bagged Linear Regression of Leukemia Data

For the first example, two-sample $t$-statistics and corresponding and $p$-values were computed from the leukemia data assuming unequal variances for all 7128 genes. From the $p$-values, $z$-values were calculated such that $z_i = \Phi(p)$ for $i = 1, .., 7128$. This is the most simplistic method, and is presented to compare results with Efron $(2012a)$.

The following models were considered for this analysis:

$$E(\texttt{Gene Expression}) = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} \tag{3.1}$$

$$E(\texttt{Gene Expression}) = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\} \tag{3.2}$$

$$E(\texttt{Gene Expression}) = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Gender}\} \tag{3.3}$$

$$E(\texttt{Gene Expression}) = \beta_0 + \beta_1\{\texttt{Leukemia Type}\} + \beta_2\{\texttt{Sample}\} + \beta_3\{\texttt{Gender}\} \tag{3.4}$$

For models (1) through (4) the test of $\beta_1$ provides $N$ Wald-Z statistic used to generate the empirical null, unadjusted $p$-values, and Fdrs. For comparison, $p$-values will be adjusted using the Bonferroni and Benjamini-Hochberg procedures, and all measures will be computed under both the theoretical and empirical null.

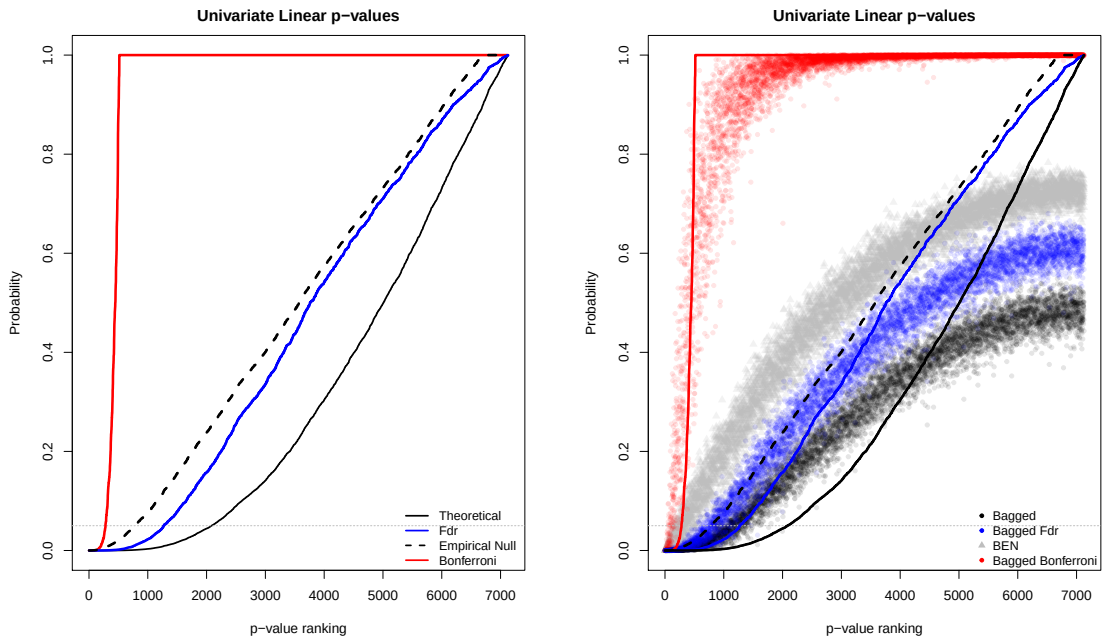



Figure 3.6: Left Figure: The black solid line represents the unadjusted $p$-values associated with the gene expression coefficient from each of the 7128 univariate linear regression models. The $p$-values are sorted from smallest to largest. The black dashed line represents the p-values recalibrated using the empirical $N(0.13, 1.60)$ null derived from the maximum likelihood method. The solid blue line represents The Benjamini-Hochberg (B-H) $p$-values, equivalent to the two-tailed global false discovery rate (Fdr) calculated under the theoretical null. The solid red line represents the Bonferroni corrected p-values under the theoretical null. Right Figure: The black dots are the bagged $p$-values calculated under the theoretical null. The gray triangles are the bagged empirical null (BEN) $p$-values. The blue dots are the bagged B-H $p$-values calculated under the theoretical null. The red dots are the bagged Bonferroni $p$-values calculated under the theoretical null. All points are sorted based on the original unadjusted $p$-values.

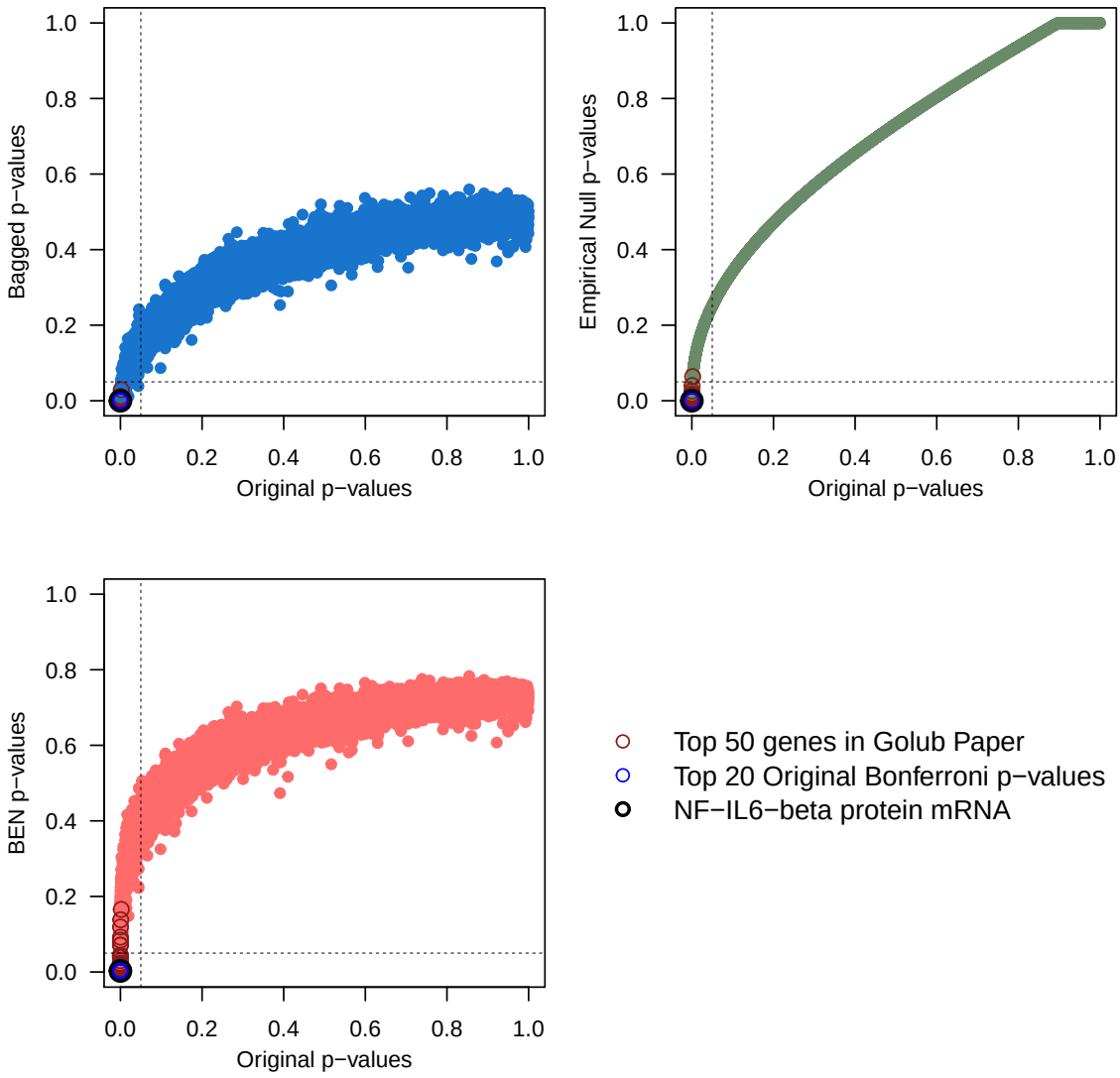



Figure 3.7: The original $p$-values from the univariate logistic regressions are plotted against the Bagged $p$-values, Empirical $N(0.13, 1.60)$ Null recalibrated $p$-values, and the Bagged Empirical Null (BEN) $p$-values. The open red circles are the 50 $p$-values Golub et al. (1999) reported as differentially expressed, and the open blue circles are the Bonferroni adjusted $p$-values $\leq 0.05$ from the original univariate logistic regressions. The bold black open circle is the top gene selected by the two-dimensional $p$-value and AUC criterion of the BEN algorithm, which corresponds the NF-IL6-beta protein mRNA gene.

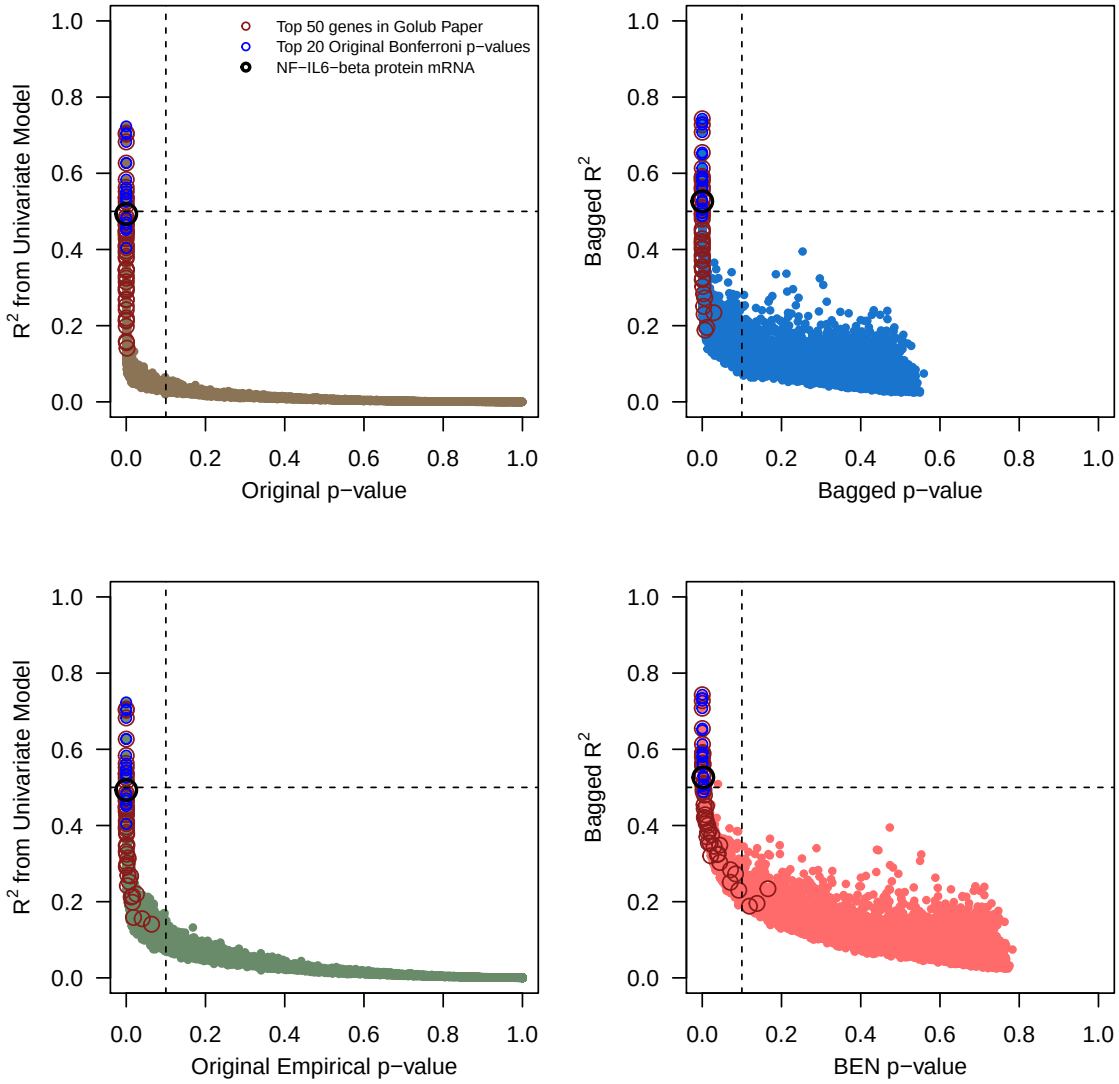



Figure 3.8: The original $p$-values from the univariate linear regressions and Empirical $N(0.13, 1.60)$ Null recalibrated $p$-values, are plotted against the corresponding model $R^2$. The Bagged $p$-values, and the Bagged Empirical Null (BEN) $p$-values are plotted against the Bagged $R^2$. The open red circles are the 50 $p$-values Golub et al. (1999) reported as differentially expressed, and the open blue circles are the Bonferroni adjusted $p$-values $\leq 0.05$ from the original univariate linear regressions. The bold black open circle is the NF-IL6-beta protein mRNA gene.

| Gene | BEN p-value | $R^2$ | Golub | Bo | Lee | Tong | Zhou | Univariate $p$ | Bonferroni $p$ |
|---|---|---|---|---|---|---|---|---|---|
| M23197 | 0.000 | 0.743 | X | X | X | | X | X | X |
| U46499 | 0.000 | 0.734 | | X | X | | | X | X |
| M31523 | 0.000 | 0.708 | X | X | X | | X | X | X |
| M84526 | 0.000 | 0.728 | X | X | X | | | X | X |
| D88422 | 0.000 | 0.650 | | X | X | X | | X | X |
| M27891 | 0.000 | 0.655 | X | X | X | | X | X | X |
| HG1612-HT1612 | 0.000 | 0.633 | | X | X | X | X | X | |
| M92287 | 0.000 | 0.613 | X | X | X | | X | X | X |
| M63138 | 0.000 | 0.580 | X | X | | | | X | X |
| J05243 | 0.000 | 0.581 | | | X | | X | X | |
| Z15115 | 0.001 | 0.585 | X | X | | | X | X | X |
| X59417 | 0.001 | 0.556 | X | X | | | | X | |
| X95735 | 0.001 | 0.564 | X | | X | | | X | X |
| M16038 | 0.001 | 0.592 | X | X | X | | | X | X |
| L09209 | 0.001 | 0.560 | | X | X | X | | X | X |
| X51521 | 0.001 | 0.559 | | X | | X | X | X | X |
| M31211 | 0.001 | 0.533 | X | X | | | X | X | |
| M11722 | 0.001 | 0.591 | | X | X | X | | X | X |
| M63379 | 0.001 | 0.509 | | X | | | | | |
| L47738 | 0.002 | 0.533 | X | | X | | X | | |
| X62320 | 0.002 | 0.532 | | X | | X | | X | |
| M55150 | 0.002 | 0.504 | X | X | | | | | X |
| M31303 | 0.002 | 0.589 | X | X | | | X | | |
| X62654 | 0.002 | 0.530 | | X | | X | | | X |
| X56468 | 0.002 | 0.556 | | | | | | | |
| U05259 | 0.002 | 0.561 | X | | X | | | | |
| M83667 | 0.003 | 0.526 | | X | | X | | X | |
| M29696 | 0.004 | 0.508 | X | | | | | | X |
| Z23064 | 0.004 | 0.502 | | | | | | | |
| D88270 | 0.004 | 0.526 | | | | | | | X |
| U22376 | 0.004 | 0.510 | X | | X | | X | | |
| S50223 | 0.004 | 0.511 | X | X | | | | | |
| M33195 | 0.004 | 0.500 | | | | | | | |
| U57721 | 0.005 | 0.504 | | | | | | | |
| M84371 | 0.005 | 0.529 | | X | | | | | |
| HG1322-HT5143 | 0.040 | 0.509 | | | | | | | |

Table 3.3: The 36 leukemia data genes with bagged empirical null (BEN) $p$-values $\leq 0.1$ and bagged $R^2 \geq 0.5$. Genes are sorted by (BEN) $p$-values from smallest to largest. We reviewed Golub et al. (1999); Lee et al. (2003); Bø and Jonassen (2002); Zhou et al. (2004); Tong and Schierz (2011), and an X is placed where the authors reported a gene we also selected. The univariate $p$-values are derived from the univariate linear regressions of the original data, and the resulting top 20 $p$-values are compared to the BEN results. The Bonferroni correction was applied to the univariate model $p$-values, and all Bonferroni $p$-values $\leq 0.05$ are compared to the BEN results.

### 3.7.5 Remark E: Shrinkage of Bagged $p$-values

The bagged theoretical null $p$-values, and bagged theoretical null B-H $p$-values are attenuated towards 0.5, compared to the non-bagged counterparts. Since $p$-values follow a Uniform(0,1) distribution under the null hypothesis, the mean of the null $p$-values is approximately 0.5.

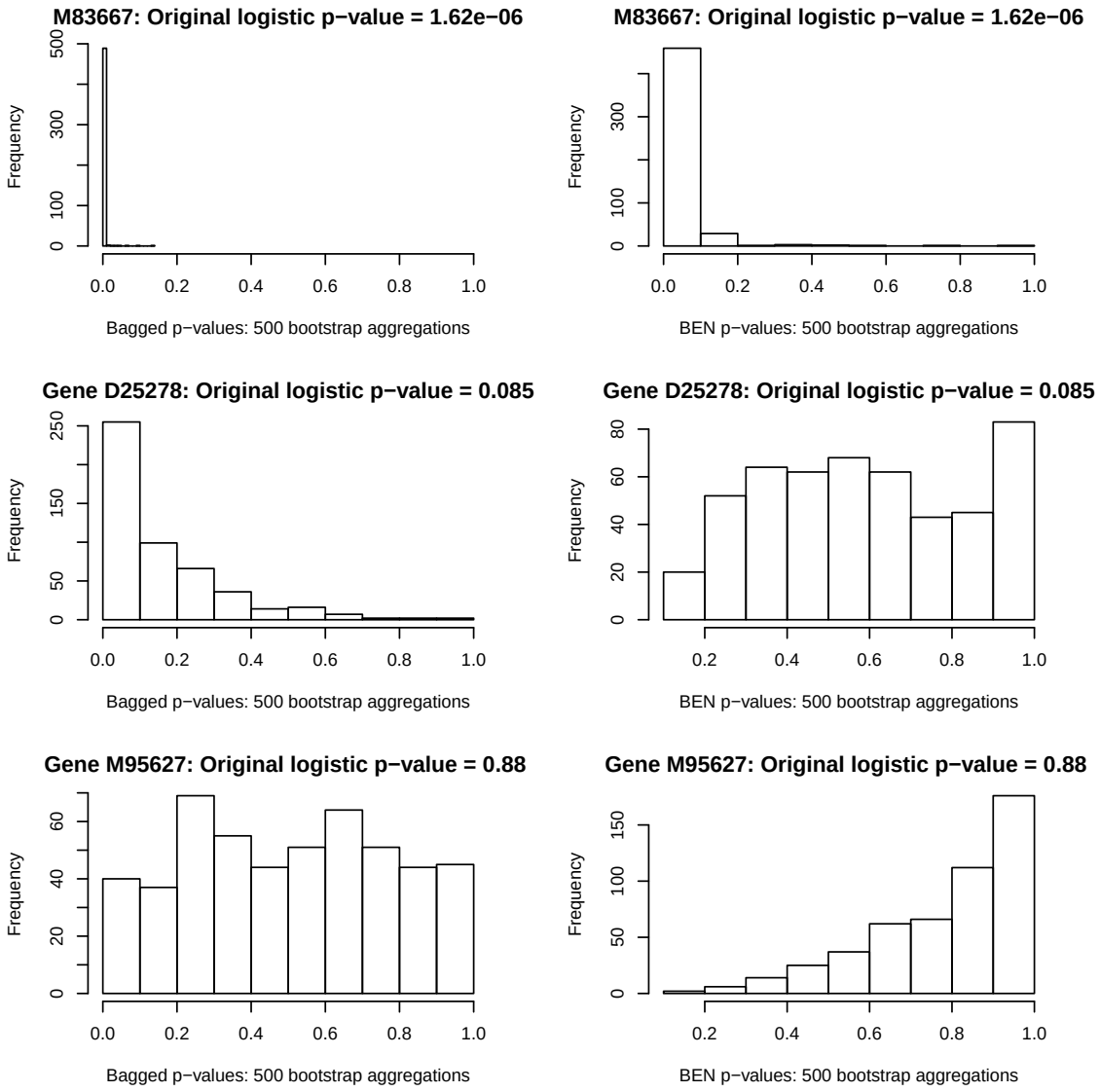



Figure 3.9: Histograms of 3 different genes of their bagged $p$-values and BEN $p$-values. The $p$-values from the univariate logistic regressions are included for comparison.

### 3.7.6 Remark F: Bootstrapping $p$-values

Even if model misspecification is not a concern, we recommend that $p$-values are bootstrapped. We have bootstrapped the $p$-values of the leukemia data assuming the univariate model is true, and include the results below. For this specific example, the bootstrapped $p$-value results are similar to the bagged B-H $p$-value results.

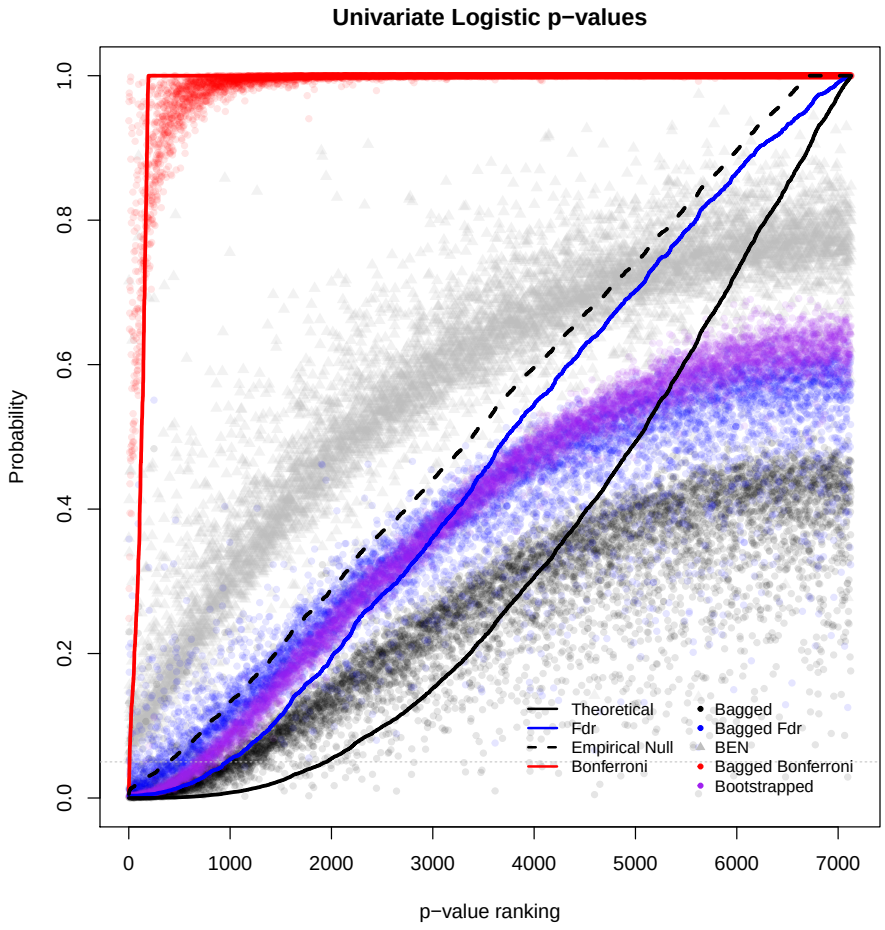



Figure 3.10: Left Figure: The black solid line represents the unadjusted $p$-values associated with the gene expression coefficient from each of the 7128 univariate logistic regression models. The $p$-values are sorted from smallest to largest. The black dashed line represents the p-values recalibrated using the empirical $N(0.13, 1.70)$ null derived from the maximum likelihood method. The solid blue line represents The Benjamini-Hochberg (B-H) $p$-values, equivalent to the two-tailed global false discovery rate (Fdr) calculated under the theoretical null. The solid red line represents the Bonferroni corrected p-values under the theoretical null. Right Figure: The black dots are the bagged $p$-values calculated under the theoretical null. The gray triangles are the bagged empirical null (BEN) $p$-values. The blue dots are the bagged B-H $p$-values calculated under the theoretical null. The purple dots are the bootstrapped $p$-values. The red dots are the bagged Bonferroni $p$-values calculated under the theoretical null. All points are sorted based on the original unadjusted $p$-values.

## 3.8 Appendix B. Code for Psuedo-Simulation and Application to Leukemia Data

```
#Install the packages
library(Hmisc)
library(rms)
library(mice)
library(locfdr)

#Intall the Golub leukemia results
#source("http://bioconductor.org/biocLite.R")
#biocLite("golubEsets")
library(golubEsets)

#################
# Functions Used
# 1) AUC Function
# 2) Expit
# 3) Global FDR/BH p-value correction code
# 4) Bonferroni p-value correction code
# 5)
#################

#AUC function
auc=function(score,status){
  ######################
  ## auc version 1.0
  ## Compute Area under Rmpirical ROC curve by Trapezoidal Rule
  ## Author: J. Blume
  ## Date:   July 2014
  ######################

  pos=score[status==1]
  neg=score[status==0]

  ct1=sum(outer(pos,neg,">"))
  ct2=sum(outer(pos,neg,"=="))
  den=length(pos)*length(neg)
  auc=(ct1+0.5*ct2)/den
  auc=max(auc,1-auc)
  auc

}

#Inverse Logit function
expit <- function(z){
  exp(z)/(1+exp(z))
}

#Logit function
logit <- function(z){
  (z)/(1-z)
}


#Two tailed Global Fdr
Global.fdr <- function(z, na.rm = FALSE){
    rawp <- z
    m<-length(rawp)
    if(na.rm){
      mgood<-sum(!is.na(rawp))
    }else{
      mgood<-m
    }
    index<-order(rawp)
    spval<-rawp[index]
    tmp<-spval
    for(i in (m-1):1){
      tmp[i]<-min(tmp[i+1],min((mgood/i)*spval[i],1,na.rm=TRUE),na.rm=TRUE)
      #if(is.na(spval[i])) tmp[i]<-NA
    }
    FDR <-tmp[order(index)]
    FDR
}
```

```
#Bonferroni Correction
Bonferroni <- function(z, na.rm = FALSE){
  rawp <- z
  m<-length(rawp)
  if(na.rm){
    mgood<-sum(!is.na(rawp))
  }else{
    mgood<-m
  }
  index<-order(rawp)
  spval<-rawp[index]
  tmp<-mgood*spval
  tmp[tmp>1]<-1
  Bonferroni <-tmp[order(index)]
  Bonferroni
}



##########################################
# Start Analysis of the Original Leukemia Data
##########################################

#Read in Leukemia data

data(Golub_Merge)
exprsDat <- exprs(Golub_Merge)
N.tmp <- nrow(exprsDat)

#Normalizing the gene expression data as described by efron
NormalizedGeneDat <-  apply(exprsDat, 2, function(z) qnorm((rank(z)-0.5)/N.tmp))

#Efron dropped the most extreme value so we will too
NormalizedGeneDat <- NormalizedGeneDat[-6777,]

NormalizedGeneDat_t <- t(NormalizedGeneDat)
N <- nrow(NormalizedGeneDat)

set.seed(sample(1:100000000,size=1))

#Impute Covariate Data
imputations <- mice(pData(Golub_Merge)[,c('ALL.AML','BM.PB','T.B.cell','Gender','PS','Source')], m=1)
imp.dat <- complete(imputations)

#Combining the normalized gene expression with the covariate information
LeukDat <- cbind(NormalizedGeneDat_t,imp.dat)


############################################################
# BEN Algorithm
#
# 1) Bootstrap individuals,
# 2) Fit different models, save z values for every gene
# 3) Fit Empirical Null for each model
# 4) Calculate all statistics
# 5) Choose set of statistics based on model with lowest AIC
# 6) Average Results
############################################################

##########################################################
# The following function takes in a seed and the number
# of simulations you with to run and returns a list of
# length N (where N is the number of genes being evaluated)
##########################################################

BEN.logistic <- function(SEED,NUMSIM){

set.seed(SEED)
n.sim = NUMSIM


bagged.res <- lapply(1:N, function(i) data.frame(z.gene = rep(NA,n.sim),
                                                 aic = rep(NA,n.sim),
                                                 auc = rep(NA,n.sim),
```

```
                                            emp.loc.fdr = rep(NA,n.sim),
                                            emp.p = rep(NA,n.sim),
                                            emp.var.p = rep(NA, n.sim),
                                            null.loc.fdr = rep(NA,n.sim),
                                            null.p = rep(NA,n.sim),
                                            emp.Fdr = rep(NA,n.sim),
                                            null.Fdr = rep(NA,n.sim),
                                            emp.Bon = rep(NA,n.sim),
                                            null.Bon = rep(NA,n.sim),
                                            emp.mean = rep(NA, n.sim),
                                            emp.sd = rep(NA,n.sim),
                                            which.mod = rep(NA,n.sim)))

for(i in 1:n.sim){
  #data frame of statistics
  all.stats <- lapply(1:6,function(i) data.frame(z.gene = rep(NA,N),
                                                 aic = rep(NA,N),
                                                 auc = rep(NA,N),
                                                 emp.loc.fdr = rep(NA,N),
                                                 emp.p = rep(NA,N),
                                                 emp.var.p = rep(NA,N),
                                                 null.loc.fdr = rep(NA,N),
                                                 null.p = rep(NA,N),
                                                 emp.Fdr = rep(NA,N),
                                                 null.Fdr = rep(NA,N),
                                                 emp.Bon = rep(NA,N),
                                                 null.Bon = rep(NA,N),
                                                 emp.mean = rep(NA, N),
                                                 emp.sd = rep(NA,N)))

  #Bootstrap the same proportion of people with ALL and AML as is in the original sample

  boot.samp.ALL <- sample(which(LeukDat[,'ALL.AML']=='ALL'),replace = TRUE)
  boot.samp.AML <- sample(which(LeukDat[,'ALL.AML']=='AML'),replace = TRUE)
  tmp.dat <- rbind(LeukDat[boot.samp.ALL,],LeukDat[boot.samp.AML,])

  for(xx in 1:N){

    #Univariate Model
    m.1 <- tryCatch(glm( (as.numeric(ALL.AML) - 1) ~ tmp.dat[,xx],
     data = tmp.dat, family = binomial), error=function(e) {NA})

    #Single Covariates
    m.2 <-tryCatch( glm( (as.numeric(ALL.AML) - 1) ~ tmp.dat[,xx] + BM.PB ,
     data = tmp.dat, family = binomial), error=function(e) {NA})
    m.3 <- tryCatch(glm( (as.numeric(ALL.AML) - 1) ~ tmp.dat[,xx] + Gender,
     data = tmp.dat, family = binomial), error=function(e) {NA})

    #Two Covariates
    m.4 <- tryCatch(glm( (as.numeric(ALL.AML) - 1) ~ tmp.dat[,xx] + BM.PB + Gender,
      data = tmp.dat, family = binomial), error=function(e) {NA})

    #Splined Gene Expression
   m.5 <- tryCatch(lrm( (as.numeric(ALL.AML) - 1) ~ rcs(tmp.dat[,xx],3),
     data = tmp.dat), error=function(e) {NA})
    m.5.glm <- tryCatch(glm( (as.numeric(ALL.AML) - 1) ~ rcs(tmp.dat[,xx],3),
      data = tmp.dat, family = 'binomial'), error=function(e) {NA})

  #Splined Gene Expression with covariates
   m.6 <- tryCatch(lrm( (as.numeric(ALL.AML) - 1) ~ rcs(tmp.dat[,xx],3) + BM.PB + Gender,
     data = tmp.dat), error=function(e) {NA})
    m.6.glm <- tryCatch(glm( (as.numeric(ALL.AML) - 1) ~ rcs(tmp.dat[,xx],3) + BM.PB + Gender,
      data = tmp.dat, family = 'binomial'), error=function(e) {NA})


    for(j in 1:4){
      m.tmp <- get(paste('m.',j,sep = ""))

      if(is.na(m.tmp[[1]][1])){
        all.stats[[j]][xx,c('z.gene')] <-NA
        all.stats[[j]][xx,c('aic')] <- NA
        all.stats[[j]][xx,c('auc')] <- NA
      }
      if(!is.na(m.tmp[[1]][1])){
```

```
      all.stats[[j]][xx,c('z.gene')] <- summary(m.tmp)$coefficients['tmp.dat[, xx]','z value']
      all.stats[[j]][xx,c('aic')] <- AIC(m.tmp)
      all.stats[[j]][xx,c('auc')] <-auc(m.tmp$fitted,as.numeric(tmp.dat$ALL.AML)-1)
    }
  }

  if(!is.na(m.5[[1]][1])){
    #Chunk test of all coefficients associated with gene expression
    all.stats[[5]][xx,c('z.gene')] <- qnorm(pchisq(anova(m.5)[1,1],anova(m.5)[1,2],
     lower.tail = FALSE),lower.tail = FALSE)
    all.stats[[5]][xx,c('aic')] <- m.5.glm$aic
    all.stats[[5]][xx,c('auc')] <- auc(m.5.glm$fitted,as.numeric(tmp.dat$ALL.AML)-1)
  }
  if(is.na(m.5[[1]][1])){
    all.stats[[5]][xx,c('z.gene')] <- NA
    all.stats[[5]][xx,c('aic')] <- NA
    all.stats[[5]][xx,c('auc')] <- NA
  }

  if(!is.na(m.6[[1]][1])){
    #Chunk test of all coefficients associated with gene expression
    all.stats[[6]][xx,c('z.gene')] <- qnorm(pchisq(anova(m.6)[1,1],anova(m.6)[1,2],
     lower.tail = FALSE),lower.tail = FALSE)
    all.stats[[6]][xx,c('aic')] <- m.6.glm$aic
    all.stats[[6]][xx,c('auc')] <- auc(m.6.glm$fitted,as.numeric(tmp.dat$ALL.AML)-1)
  }
  if(is.na(m.6[[1]][1])){
    all.stats[[6]][xx,c('z.gene')] <- NA
    all.stats[[6]][xx,c('aic')] <- NA
    all.stats[[6]][xx,c('auc')] <- NA
  }


}

#For each model find the p-value, and local fdr under the empirical null
BEN.fdr.p <- for(qq in 1:6) {
  tmp.mod <- all.stats[[qq]]
  tmp.dat <- tmp.mod[complete.cases(tmp.mod[,'z.gene']),'z.gene']

  #Empirical Null estimation using maximum likelihood methods
  tmp.fdr.emp <- tryCatch(locfdr(tmp.dat,nulltype=1,plot=0),
                          error=function(e) {NA})
  tmp.fdr.null <- tryCatch(locfdr(tmp.dat,nulltype=0,plot=0),
                           error=function(e) {NA})
  if(length(tmp.fdr.emp)>1){
    emp.loc.fdr <- tmp.fdr.emp$fdr

    #recalibration of p-values under the empirical null


    if{qq == 5|qq == 6}{
    emp.p <- min(2*(1- pnorm(tmp.dat,
                      mean=tmp.fdr.emp$fp0['mlest','delta'],
                      sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)

    #recalibration of p-values fixing the mean at 0 and taking the variance from the empirical null
    emp.var.p <- min(2*(1- pnorm(tmp.dat,
                      mean=0,
                      sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)
    }
    else{emp.p <- min(2*(1- pnorm(abs(tmp.dat),
                      mean=tmp.fdr.emp$fp0['mlest','delta'],
                      sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)

    #recalibration of p-values fixing the mean at 0 and taking the variance from the empirical null
    emp.var.p <- min(2*(1- pnorm(abs(tmp.dat),
                      mean=0,
                      sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)
    }
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.loc.fdr'] <- emp.loc.fdr

    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.p'] <- emp.p
```

```
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.var.p'] <- emp.var.p

    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Fdr'] <- Global.fdr(emp.p)

    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Bon'] <- Bonferroni(emp.p)

all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.mean'] <- tmp.fdr.emp$fp0['mlest','delta']

all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.sd'] <- tmp.fdr.emp$fp0['mlest','sigma']

  }
  if(is.na(tmp.fdr.emp)){
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.loc.fdr'] <- NA
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.p'] <- NA
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.var.p'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Fdr'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Bon'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.mean'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.sd'] <- NA
  }
  if(length(tmp.fdr.null)>1){
    null.loc.fdr <- tmp.fdr.null$fdr
    null.p <- min(2*(1- pnorm(abs(tmp.dat), mean=0,sd=1)),1)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.loc.fdr'] <- null.loc.fdr
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.p'] <- null.p
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Fdr'] <- Global.fdr(null.p)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Bon'] <- Bonferroni(null.p)
  }
  if(is.na(tmp.fdr.null)){
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.loc.fdr'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.p'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Fdr'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Bon'] <- NA
  }


}

tmp.dat <- do.call(rbind, all.stats) # dim(X2) = 7128*17 x ncol
grp <- rep(1:N, 6)
split.by.genes <- split(tmp.dat, grp)

#Now store the BEN p-values, Bagged p-values, and AUC for the selected Model
#For gene 1, find the model
for(j in 1:N){
 # Choose the model with the lowest AIC
  best.mod <- which(split.by.genes[[j]][,'aic']==min(split.by.genes[[j]][,'aic'], na.rm = TRUE))
  bagged.res[[j]][i,c('z.gene',
                      'aic',
                      'auc',
                      'emp.loc.fdr',
                      'emp.p',
                      'emp.var.p',
                      'null.loc.fdr',
                      'null.p',
                      'emp.Fdr',
                      'null.Fdr',
                      'emp.Bon',
                      'null.Bon',
                      'emp.mean',
                      'emp.sd',
                      'which.mod')] <- unlist(c(split.by.genes[[j]][best.mod,c('z.gene',
                      'aic',
                      'auc',
                      'emp.loc.fdr',
                      'emp.p',
                      'emp.var.p',
                      'null.loc.fdr',
                      'null.p',
                      'emp.Fdr',
                      'null.Fdr',
                      'emp.Bon',
                      'null.Bon',
                      'emp.mean',
```

```
                        'emp.sd')],best.mod))
  }

}
bagged.res
}


#####################################
#Bagged Emp Null from the linear models
#####################################

BEN.linear <- function(SEED,NUMSIM){

  set.seed(SEED)
  n.sim = NUMSIM

bagged.res.linear <- lapply(1:N, function(i) data.frame(z.gene = rep(NA,n.sim),
                                                        aic = rep(NA,n.sim),
                                                        r2 = rep(NA,n.sim),
                                                        r2.adjusted = rep(NA,n.sim),
                                                        emp.loc.fdr = rep(NA,n.sim),
                                                        emp.p = rep(NA,n.sim),
                                                        emp.var.p = rep(NA, n.sim),
                                                        null.loc.fdr = rep(NA,n.sim),
                                                        null.p = rep(NA,n.sim),
                                                     emp.Fdr = rep(NA,n.sim),
                                                     null.Fdr = rep(NA,n.sim),
                                                    emp.Bon = rep(NA,n.sim),
                                                     null.Bon = rep(NA,n.sim),
                                                        emp.mean = rep(NA, n.sim),
                                                        emp.sd = rep(NA,n.sim),
                                                        which.mod = rep(NA,n.sim)))

for(i in 1:n.sim){
  #data frame of statistics
  all.stats <- lapply(1:4,function(i) data.frame(z.gene = rep(NA,N),
                                                 aic = rep(NA,N),
                                                 r2 = rep(NA,N),
                                                 r2.adjusted = rep(NA,N),
                                                 emp.loc.fdr = rep(NA,N),
                                                 emp.p = rep(NA,N),
                                                 emp.var.p = rep(NA,N),
                                                 null.loc.fdr = rep(NA,N),
                                                 null.p = rep(NA,N),
                                                 emp.Fdr = rep(NA,N),
                                                 null.Fdr = rep(NA,N),
                                                 emp.Bon = rep(NA,N),
                                                 null.Bon = rep(NA,N),
                                                 emp.mean = rep(NA, N),
                                                 emp.sd = rep(NA,N)))

#Bootstrap the same proportion of people with ALL and AML as is in the original sample
 boot.samp.ALL <- sample(which(LeukDat[,'ALL.AML']=='ALL'),replace = TRUE)
 boot.samp.AML <- sample(which(LeukDat[,'ALL.AML']=='AML'),replace = TRUE)
 tmp.dat <- rbind(LeukDat[boot.samp.ALL,],LeukDat[boot.samp.AML,])

  for(xx in 1:N){

    #Univariate Model
    m.1 <- tryCatch(lm(tmp.dat[,xx] ~ ALL.AML, data = tmp.dat), error=function(e) {NA})

    #Single Covariates
    m.2 <-tryCatch( lm( tmp.dat[,xx] ~ ALL.AML + BM.PB , data = tmp.dat), error=function(e) {NA})
    m.3 <- tryCatch(lm( tmp.dat[,xx] ~ ALL.AML + Gender, data = tmp.dat), error=function(e) {NA})

    #Two Covariates
    m.4 <- tryCatch(lm( tmp.dat[,xx] ~ ALL.AML + BM.PB + Gender, data = tmp.dat), error=function(e) {NA})


    for(j in 1:4){
      m.tmp <- get(paste('m.',j,sep = ""))

      if(is.na(m.tmp)){
```

```
      all.stats[[j]][xx,c('z.gene')] <-NA
      all.stats[[j]][xx,c('aic')] <- NA
      all.stats[[j]][xx,c('r2')] <- NA
      all.stats[[j]][xx,c('r2.adjusted')] <- NA

    }
    if(!is.na(m.tmp)){
      z.sign <- ifelse(summary(m.tmp)$coefficients['ALL.AMLAML','t value']>=0,1,-1)
      all.stats[[j]][xx,c('z.gene')] <- abs(qnorm(summary(m.tmp)$coefficients['ALL.AMLAML','Pr(>|t|)']/2))*z.sign
      all.stats[[j]][xx,c('aic')] <- AIC(m.tmp)
      all.stats[[j]][xx,c('r2')] <-summary(m.tmp)$r.squared
      all.stats[[j]][xx,c('r2.adjusted')] <-summary(m.tmp)$adj.r.squared
    }
  }
}

#For each model find the p-value, and local fdr under the empirical null
BEN.fdr.p <- for(qq in 1:4) {
  tmp.mod <- all.stats[[qq]]
  tmp.dat <- tmp.mod[complete.cases(tmp.mod[,'z.gene']),'z.gene']
  tmp.fdr.emp <- tryCatch(locfdr(tmp.dat,nulltype=1,plot=0),
                          error=function(e) {NA})
  tmp.fdr.null <- tryCatch(locfdr(tmp.dat,nulltype=0,plot=0),
                           error=function(e) {NA})
  if(length(tmp.fdr.emp)>1){
    #Empirical Null calculated using the maximum likelihood method
    emp.loc.fdr <- tmp.fdr.emp$fdr

    #recalibration of p-values under the empirical null (BEN pvals)
    emp.p <- min(2*(1- pnorm(abs(tmp.dat),
                      mean=tmp.fdr.emp$fp0['mlest','delta'],
                      sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)

    #BEN pvalues fixing mean at 0, and using variance from empirical null
    emp.var.p <- min(2*(1- pnorm(abs(tmp.dat),
                      mean=0,
                      sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)

    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.loc.fdr'] <- emp.loc.fdr
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.p'] <- emp.p
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.var.p'] <- emp.var.p
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Fdr'] <- Global.fdr(emp.p)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Bon'] <- Bonferroni(emp.p)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.mean'] <- tmp.fdr.emp$fp0['mlest','delta']
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.sd'] <- tmp.fdr.emp$fp0['mlest','sigma']

  }
  if(is.na(tmp.fdr.emp)){
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.loc.fdr'] <- NA
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.p'] <- NA
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.var.p'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Fdr'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Bon'] <- NA
     all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.mean'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.sd'] <- NA


  }
  if(length(tmp.fdr.null)>1){
    null.loc.fdr <- tmp.fdr.null$fdr
    null.p <- 2*(1- pnorm(abs(tmp.dat), mean=0,sd=1))
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.loc.fdr'] <- null.loc.fdr
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.p'] <- null.p
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Fdr'] <- Global.fdr(null.p)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Bon'] <- Bonferroni(null.p)

  }
  if(is.na(tmp.fdr.null)){
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.loc.fdr'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.p'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Fdr'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Bon'] <- NA

  }
```

```
  }


  tmp.dat <- do.call(rbind, all.stats) # dim(X2) = 7128*17 x ncol
  grp <- rep(1:N, 4)
  split.by.genes <- split(tmp.dat, grp)

  #Now store the BEN statistics and bagged AUC for the selected Model
  #For gene 1, find the model
  for(j in 1:N){
   #Choose model with the smalled AIC
    best.mod <- which(split.by.genes[[j]][,'aic']==min(split.by.genes[[j]][,'aic'], na.rm = TRUE))
    bagged.res.linear[[j]][i,c('z.gene',
                               'aic',
                               'r2',
                               'r2.adjusted',
                               'emp.loc.fdr',
                               'emp.p',
                               'emp.var.p',
                               'null.loc.fdr',
                               'null.p',
                             'emp.Fdr',
                               'null.Fdr',
                               'emp.Bon',
                               'null.Bon',
                               'which.mod',
                               'emp.mean',
                               'emp.sd')] <- unlist(c(split.by.genes[[j]][best.mod,c('z.gene',
                               'aic',
                               'r2',
                               'r2.adjusted',
                               'emp.loc.fdr',
                               'emp.p',
                               'emp.var.p',
                               'null.loc.fdr',
                               'null.p',
                            'emp.Fdr',
                               'null.Fdr',
                               'emp.Bon',
                               'null.Bon',
                               'emp.mean',
                               'emp.sd')],best.mod))
  }

}

bagged.res.linear

}

#########################################################################################################################
#####################################
#Adjustments to Original leukemia data
#####################################
# load("/Users/SFM/Dropbox/GraduateSchool/LSIGroup/Chapter 6/leukz.Rda")

LeukStats <- data.frame(logistic.z = rep(NA, 7128),
                        logistic.p = rep(NA, 7128),
                        logistic.auc = rep(NA,7128),
                        logistic.aic = rep(NA,7128),
                        linear.p = rep(NA, 7128),
                        linear.p.z = rep(NA, 7128),
                        linear.aic = rep(NA,7128),
                        linear.r2 = rep(NA,7128),
                        linear.r2.adjusted = rep(NA,7128))
for(xx in 1:7128){
 #LOGISTIC UNIVARIATE MODELS
logistic.mod <-  glm( (as.numeric(ALL.AML) - 1) ~ LeukDat[,xx],
                           data = LeukDat,
                           family = binomial)
tmp.auc <- auc(logistic.mod$fitted,as.numeric(LeukDat$ALL.AML)-1)
LeukStats[xx,'logistic.auc'] <- ifelse(tmp.auc>1,1,tmp.auc)
```

```
LeukStats[xx,'logistic.aic'] <- logistic.mod$aic
LeukStats[xx,'logistic.z'] <- summary(logistic.mod)$coefficients['LeukDat[, xx]','z value']
LeukStats[xx,'logistic.p'] <- summary(logistic.mod)$coefficients['LeukDat[, xx]','Pr(>|z|)']

 #LINEAR UNIVARIATE MODELS
linear.mod <- lm(LeukDat[,xx] ~ ALL.AML, data = LeukDat)
    z.sign <- ifelse(summary.lm(linear.mod)$coefficients['ALL.AML2','t value']>=0,1,-1)
    z.tmp <- abs(qnorm(summary.lm(linear.mod)$coefficients['ALL.AML2','Pr(>|t|)']/2))*z.sign
    LeukStats[xx,c('linear.z')] <-  z.tmp
    LeukStats[xx,'linear.p'] <- summary(linear.mod)$coefficients['ALL.AML2','Pr(>|t|)']
    LeukStats[xx,'linear.p.z'] <-  min(2*(1- pnorm(abs(z.tmp), mean=0,sd=1)),1)
    LeukStats[xx,'linear.aic'] <- AIC(linear.mod)
LeukStats[xx,'linear.r2'] <- summary(linear.mod)$r.squared
LeukStats[xx,'linear.r2.adjusted'] <-summary(linear.mod)$adj.r.squared
}

################################
# Empirical Null with Linear z
################################

tmp.fdr.empirical.null <- locfdr(LeukStats[,c('linear.z')],nulltype=1,plot=0)

null.p <- min(2*(1- pnorm(abs(LeukStats[,c('linear.z')]),
                    mean=0,
                    sd=1)),1)
#Calculate p value from the empirical mean and variance
emp.p <- min(2*(1- pnorm(abs(LeukStats[,c('linear.z')]),
                    mean=tmp.fdr.emp$fp0['mlest','delta'],
                    sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)
emp.p <- ifelse(emp.p >1,1,emp.p)

LeukStats[,'null.p.linear.bonf'] <- Bonferroni(null.p)
LeukStats[,'null.p.linear.fdr'] <- Global.fdr(null.p)
LeukStats[,'emp.p.linear.bonf'] <- Bonferroni(emp.p)
LeukStats[,'emp.p.linear.fdr'] <- Global.fdr(emp.p)

################################
# Empirical Null with Logistic z
################################

leukz.logistic <- LeukStats[,'logistic.z']
tmp.fdr.emp.logistic <- locfdr(leukz.logistic,nulltype=1,plot=0)

null.p.logistic <- min(2*(1- pnorm(abs(leukz.logistic),
                    mean=0,
                    sd=1)),1)
#Calculate p value from the empirical mean and variance
emp.p.logistic <- 2*(1- pnorm(abs(leukz.logistic),
                    mean=tmp.fdr.emp.logistic$fp0['mlest','delta'],
                    sd=tmp.fdr.emp.logistic$fp0['mlest','sigma']))
emp.p.logistic <- ifelse(emp.p.logistic >1,1,emp.p.logistic)


LeukStats[,'null.p.logistic.bonf'] <- Bonferroni(null.p.logistic)
LeukStats[,'null.p.logistic.fdr'] <- Global.fdr(null.p.logistic)
LeukStats[,'emp.p.logistic.bonf'] <- Bonferroni(emp.p.logistic)
LeukStats[,'emp.p.logistic.fdr'] <- Global.fdr(emp.p.logistic)

#########################################################################################################

#Genes from 5 papers we reviewed

#Golub Paper: top 25 overexpressed in ALL
U22376 <- grep("U22376",rownames(NormalizedGeneDat),ignore.case=TRUE)
X59417 <- grep("X59417",rownames(NormalizedGeneDat),ignore.case=TRUE)
U05259 <- grep("U05259",rownames(NormalizedGeneDat),ignore.case=TRUE)
M92287 <- grep("M92287",rownames(NormalizedGeneDat),ignore.case=TRUE)
M31211 <- grep("M31211",rownames(NormalizedGeneDat),ignore.case=TRUE)
X74262 <- grep("X74262",rownames(NormalizedGeneDat),ignore.case=TRUE)
D26156 <- grep("D26156",rownames(NormalizedGeneDat),ignore.case=TRUE)
S50223 <- grep("S50223",rownames(NormalizedGeneDat),ignore.case=TRUE)
M31523 <- grep("M31523",rownames(NormalizedGeneDat),ignore.case=TRUE)
L47738 <- grep("L47738",rownames(NormalizedGeneDat),ignore.case=TRUE)
U32944 <- grep("U32944",rownames(NormalizedGeneDat),ignore.case=TRUE)
```

```
Z15115 <- grep("Z15115",rownames(NormalizedGeneDat),ignore.case=TRUE)
X15949 <- grep("X15949",rownames(NormalizedGeneDat),ignore.case=TRUE)
X63469 <- grep("X63469",rownames(NormalizedGeneDat),ignore.case=TRUE)
M91432 <- grep("M91432",rownames(NormalizedGeneDat),ignore.case=TRUE)
U29175 <- grep("U29175",rownames(NormalizedGeneDat),ignore.case=TRUE)
Z69881 <- grep("Z69881",rownames(NormalizedGeneDat),ignore.case=TRUE)
U20998 <- grep("U20998",rownames(NormalizedGeneDat),ignore.case=TRUE)
D38073 <- grep("D38073",rownames(NormalizedGeneDat),ignore.case=TRUE)
U26266 <- grep("U26266",rownames(NormalizedGeneDat),ignore.case=TRUE)
M31303 <- grep("M31303",rownames(NormalizedGeneDat),ignore.case=TRUE)
Y08612 <- grep("Y08612",rownames(NormalizedGeneDat),ignore.case=TRUE)
U35451 <- grep("U35451",rownames(NormalizedGeneDat),ignore.case=TRUE)
M29696 <- grep("M29696",rownames(NormalizedGeneDat),ignore.case=TRUE)
M13792 <- grep("M13792",rownames(NormalizedGeneDat),ignore.case=TRUE)

#Golub paper: top 25 overexpressed in AML
M55150 <- grep("M55150",rownames(NormalizedGeneDat),ignore.case=TRUE)
X95735 <- grep("X95735",rownames(NormalizedGeneDat),ignore.case=TRUE)
U50136 <- grep("U50136",rownames(NormalizedGeneDat),ignore.case=TRUE)
M16038 <- grep("M16038",rownames(NormalizedGeneDat),ignore.case=TRUE)
U82759 <- grep("U82759",rownames(NormalizedGeneDat),ignore.case=TRUE) #****Single most highly correlated gene
M23197 <- grep("M23197",rownames(NormalizedGeneDat),ignore.case=TRUE)
M84526 <- grep("M84526",rownames(NormalizedGeneDat),ignore.case=TRUE)
Y12670 <- grep("Y12670",rownames(NormalizedGeneDat),ignore.case=TRUE)
M27891 <- grep("M27891",rownames(NormalizedGeneDat),ignore.case=TRUE)
X17042 <- grep("X17042",rownames(NormalizedGeneDat),ignore.case=TRUE)
Y00787 <- grep("Y00787",rownames(NormalizedGeneDat),ignore.case=TRUE)
M96326 <- grep("M96326",rownames(NormalizedGeneDat),ignore.case=TRUE)
U46751 <- grep("U46751",rownames(NormalizedGeneDat),ignore.case=TRUE)
M80254 <- grep("M80254",rownames(NormalizedGeneDat),ignore.case=TRUE)
L08246 <- grep("L08246",rownames(NormalizedGeneDat),ignore.case=TRUE)
M62762 <- grep("M62762",rownames(NormalizedGeneDat),ignore.case=TRUE)
M28130 <- grep("M28130",rownames(NormalizedGeneDat),ignore.case=TRUE)
M63138 <- grep("M63138",rownames(NormalizedGeneDat),ignore.case=TRUE)
M57710 <- grep("M57710",rownames(NormalizedGeneDat),ignore.case=TRUE)
M69043 <- grep("M69043",rownames(NormalizedGeneDat),ignore.case=TRUE)
M81695 <- grep("M81695",rownames(NormalizedGeneDat),ignore.case=TRUE)
X85116 <- grep("X85116",rownames(NormalizedGeneDat),ignore.case=TRUE)
M19045 <- grep("M19045",rownames(NormalizedGeneDat),ignore.case=TRUE)
M83652 <- grep("M83652",rownames(NormalizedGeneDat),ignore.case=TRUE)
X04085 <- grep("X04085",rownames(NormalizedGeneDat),ignore.case=TRUE)

GolubPaper <- c(U22376,X59417,U05259,M92287,M31211,
                X74262,D26156,S50223,M31523,L47738,
                U32944,Z15115,X15949,X63469,M91432,
                U29175,Z69881,U20998,D38073,U26266,
                M31303,Y08612,U35451,M29696,M13792,
                M55150,X95735,U50136,M16038,U82759,
                M23197,M84526,Y12670,M27891,X17042,
                Y00787,M96326,U46751,M80254,L08246,
                M62762,M28130,M63138,M57710,M69043,
                M81695,X85116,M19045,M83652,X04085)

#From Gene Selection - A Bayesian Variable Selection Approach
LeePaper <- c(1882,760,2288,4847,1144,1120,4535,6218,6200,1834,1630,5772,1745,
  804,2354,3252,6201,1685,6041,1779,6855-1,173,2642,1829,4107,697,229)


#Bo Paper
M84526 <- grep("M84526",rownames(NormalizedGeneDat),ignore.case=TRUE)
M92287 <- grep("M92287",rownames(NormalizedGeneDat),ignore.case=TRUE)
M23197 <- grep("M23197",rownames(NormalizedGeneDat),ignore.case=TRUE)
M31523 <- grep("M31523",rownames(NormalizedGeneDat),ignore.case=TRUE)
U46499 <- grep("U46499",rownames(NormalizedGeneDat),ignore.case=TRUE)
M31303 <- grep("M31303",rownames(NormalizedGeneDat),ignore.case=TRUE)
M63138 <- grep("M63138",rownames(NormalizedGeneDat),ignore.case=TRUE)
HG1612 <- grep("HG1612",rownames(NormalizedGeneDat),ignore.case=TRUE)
X62320 <- grep("X62320",rownames(NormalizedGeneDat),ignore.case=TRUE)
Z14982 <- grep("Z14982",rownames(NormalizedGeneDat),ignore.case=TRUE)
M31211 <- grep("M31211",rownames(NormalizedGeneDat),ignore.case=TRUE)
X62654 <- grep("X62654",rownames(NormalizedGeneDat),ignore.case=TRUE)
M27891 <- grep("M27891",rownames(NormalizedGeneDat),ignore.case=TRUE)
U89922 <- grep("U89922",rownames(NormalizedGeneDat),ignore.case=TRUE)
X59417 <- grep("X59417",rownames(NormalizedGeneDat),ignore.case=TRUE)
```

```
X52056 <- grep("X52056",rownames(NormalizedGeneDat),ignore.case=TRUE)
M19507 <- grep("M19507",rownames(NormalizedGeneDat),ignore.case=TRUE)
M89957 <- grep("M89957",rownames(NormalizedGeneDat),ignore.case=TRUE)
M84371 <- grep("M84371",rownames(NormalizedGeneDat),ignore.case=TRUE)
U16954 <- grep("U16954",rownames(NormalizedGeneDat),ignore.case=TRUE)
M63379 <- grep("M63379",rownames(NormalizedGeneDat),ignore.case=TRUE)
M83667 <- grep("M83667",rownames(NormalizedGeneDat),ignore.case=TRUE)
M16038 <- grep("M16038",rownames(NormalizedGeneDat),ignore.case=TRUE)
Y08612 <- grep("Y08612",rownames(NormalizedGeneDat),ignore.case=TRUE)
D88422 <- grep("D88422",rownames(NormalizedGeneDat),ignore.case=TRUE)
M11722 <- grep("M11722",rownames(NormalizedGeneDat),ignore.case=TRUE)
X66401 <- grep("X66401",rownames(NormalizedGeneDat),ignore.case=TRUE)
Y00433 <- grep("Y00433",rownames(NormalizedGeneDat),ignore.case=TRUE)
M63959 <- grep("M63959",rownames(NormalizedGeneDat),ignore.case=TRUE)
X51521 <- grep("X51521",rownames(NormalizedGeneDat),ignore.case=TRUE)
Z15115 <- grep("Z15115",rownames(NormalizedGeneDat),ignore.case=TRUE)
U10868 <- grep("U10868",rownames(NormalizedGeneDat),ignore.case=TRUE)
Y12670 <- grep("Y12670",rownames(NormalizedGeneDat),ignore.case=TRUE)
U77948 <- grep("U77948",rownames(NormalizedGeneDat),ignore.case=TRUE)
U46751 <- grep("U46751",rownames(NormalizedGeneDat),ignore.case=TRUE)
L06797 <- grep("L06797",rownames(NormalizedGeneDat),ignore.case=TRUE)
M95678 <- grep("M95678",rownames(NormalizedGeneDat),ignore.case=TRUE)
U72936 <- grep("U72936",rownames(NormalizedGeneDat),ignore.case=TRUE)
S76617 <- grep("S76617",rownames(NormalizedGeneDat),ignore.case=TRUE)
L09209 <- grep("L09209",rownames(NormalizedGeneDat),ignore.case=TRUE)
M55150 <- grep("M55150",rownames(NormalizedGeneDat),ignore.case=TRUE)
M96803 <- grep("M96803",rownames(NormalizedGeneDat),ignore.case=TRUE)
X17042 <- grep("X17042",rownames(NormalizedGeneDat),ignore.case=TRUE)
X99920 <- grep("X99920",rownames(NormalizedGeneDat),ignore.case=TRUE)
S50223 <- grep("S50223",rownames(NormalizedGeneDat),ignore.case=TRUE)
U82759 <- grep("U82759",rownames(NormalizedGeneDat),ignore.case=TRUE)
J03589 <- grep("J03589",rownames(NormalizedGeneDat),ignore.case=TRUE)
X12447 <- grep("X12447",rownames(NormalizedGeneDat),ignore.case=TRUE)
X74262 <- grep("X74262",rownames(NormalizedGeneDat),ignore.case=TRUE)
L19437 <- grep("L19437",rownames(NormalizedGeneDat),ignore.case=TRUE)

BoPaper <- c(M84526,M92287,M23197,
  M31523,U46499,M31303,M63138,HG1612,
  X62320,Z14982,M31211,X62654,M27891,
  U89922,X59417,X52056,M19507,M89957,M84371,U16954,M63379,
  M83667,M16038,Y08612,D88422,M11722,X66401,
         Y00433,M63959,X51521,Z15115,U10868,Y12670,
  U77948,U46751,L06797,M95678,U72936,S76617,L09209,
  M55150,M96803,X17042,X99920,S50223,U82759,
  J03589,X12447,X74262,L19437)

ZhouPaper <- c(4211, 5772, 2354, 1144, 1928, 4167, 804, 6281, 4398, 1630,
               1882, 1834, 5501, 2348, 1120, 5039, 6855-1, 6279, 3258, 1704)

TabusPaper <- c(1882, 2642, 3252, 758, 4847, 1834, 2288, 2335, 1685, 760, 6376,
                6855-1, 2354, 6041, 4680, 4377, 3469, 6510, 6225, 4328)


#############################################################################################################
###################
# Pseudo-Simulation
####################
#Read in Leukemia data

## Replicating the normalized expression levels from the raw data
data(Golub_Merge)
exprsDat <- exprs(Golub_Merge)

#I'm going to take all the p-values > 0.3 from the univariate linear model,
#and call those genes that are probably not differentially expressed
non.sig <- which(LeukStats$linear.p>0.3)


exprsDat <- exprsDat[non.sig,]
N.tmp <- nrow(exprsDat)

#This allows the imputed data to vary with the simulations, so a weird imputed data set doesn't make weird results
set.seed(MYSEED)
#Impute Covariate Data
```

```
imputations <- mice(pData(Golub_Merge)[,c('ALL.AML','BM.PB','T.B.cell','Gender','PS','Source')], m=1)
imp.dat <- complete(imputations)


exprsDat2 <- exprsDat

#Function to make chosen genes significant
makeSignificant <- function(gene,beta.int,
                                beta.leuk,
                                beta.gender,
                                beta.sample,
                                beta.leukxgender,
                                beta.leukxsample,
                                beta.genderxsample){

#Fit full linear model
m.gene <- glm(exprsDat2[gene,] ~ imp.dat$ALL.AML*imp.dat$Gender
                               + imp.dat$ALL.AML*imp.dat$BM.PB
                               + imp.dat$Gender*imp.dat$BM.PB)

X = cbind(1,as.numeric(imp.dat$ALL.AML)-1,
            as.numeric(imp.dat$Gender)-1,
            as.numeric(imp.dat$BM.PB)-1,
           (as.numeric(imp.dat$ALL.AML)-1)*(as.numeric(imp.dat$Gender)-1),
           (as.numeric(imp.dat$ALL.AML)-1)*(as.numeric(imp.dat$BM.PB)-1),
           (as.numeric(imp.dat$BM.PB)-1)*(as.numeric(imp.dat$Gender)-1))

#Update Beta so that the new gene has a specified relationship
Beta = c(beta.int*m.gene$coefficients[1],
         beta.leuk*m.gene$coefficients[2],
         beta.gender*m.gene$coefficients[3],
         beta.sample*m.gene$coefficients[4],
         beta.leukxgender*m.gene$coefficients[5],
         beta.leukxsample*m.gene$coefficients[6],
         beta.genderxsample*m.gene$coefficients[7])

#new fitted values
new.fitted <- X%*%t(t(Beta))

#Add the new fitted value to the original residuals
gene.new <- new.fitted + m.gene$residuals
gene.new

}


strongaffect = 7
modaffect = 4
weakaffect = 2

#Begin Strong
exprsDat2[gene.sample[1],] <- makeSignificant(gene.sample[1], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[2],] <- makeSignificant(gene.sample[2], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[3],] <- makeSignificant(gene.sample[3], beta.int = 1,
beta.leuk = 1,
beta.gender=strongaffect,
beta.sample=1,
beta.leukxgender=strongaffect,
beta.leukxsample=1,
beta.genderxsample=1)
```

```
exprsDat2[gene.sample[4],] <- makeSignificant(gene.sample[4], beta.int = 1,
beta.leuk = 1,
beta.gender=strongaffect,
beta.sample=1,
beta.leukxgender=strongaffect,
beta.leukxsample=1,
beta.genderxsample=1)


exprsDat2[gene.sample[5],] <- makeSignificant(gene.sample[5], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=strongaffect,
beta.genderxsample=1)

exprsDat2[gene.sample[6],] <- makeSignificant(gene.sample[6], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=strongaffect,
beta.genderxsample=1)


exprsDat2[gene.sample[7],] <- makeSignificant(gene.sample[7], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=strongaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[8],] <- makeSignificant(gene.sample[8], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=strongaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[9],] <- makeSignificant(gene.sample[9], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=strongaffect,
beta.leukxsample=strongaffect,
beta.genderxsample=1)

exprsDat2[gene.sample[10],] <- makeSignificant(gene.sample[10], beta.int = 1,
beta.leuk = strongaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=strongaffect,
beta.leukxsample=strongaffect,
beta.genderxsample=1)


#Begin Moderate
exprsDat2[gene.sample[11],] <- makeSignificant(gene.sample[11], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[12],] <- makeSignificant(gene.sample[12], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
```

```
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[13],] <- makeSignificant(gene.sample[13], beta.int = 1,
beta.leuk = 1,
beta.gender=modaffect,
beta.sample=1,
beta.leukxgender=modaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[14],] <- makeSignificant(gene.sample[14], beta.int = 1,
beta.leuk = 1,
beta.gender=modaffect,
beta.sample=1,
beta.leukxgender=modaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[15],] <- makeSignificant(gene.sample[15], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=modaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[16],] <- makeSignificant(gene.sample[16], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=modaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[17],] <- makeSignificant(gene.sample[17], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=modaffect,
beta.genderxsample=1)

exprsDat2[gene.sample[18],] <- makeSignificant(gene.sample[18], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=modaffect,
beta.genderxsample=1)


exprsDat2[gene.sample[19],] <- makeSignificant(gene.sample[19], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=modaffect,
beta.leukxsample=modaffect,
beta.genderxsample=1)

exprsDat2[gene.sample[20],] <- makeSignificant(gene.sample[20], beta.int = 1,
beta.leuk = modaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=modaffect,
beta.leukxsample=modaffect,
beta.genderxsample=1)


#Begin Weak Associations
exprsDat2[gene.sample[21],] <- makeSignificant(gene.sample[21], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
```

```
beta.leukxgender=1,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[22],] <- makeSignificant(gene.sample[22], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[23],] <- makeSignificant(gene.sample[23], beta.int = 1,
beta.leuk = 1,
beta.gender=weakaffect,
beta.sample=1,
beta.leukxgender=weakaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[24],] <- makeSignificant(gene.sample[24], beta.int = 1,
beta.leuk = 1,
beta.gender=weakaffect,
beta.sample=1,
beta.leukxgender=weakaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[25],] <- makeSignificant(gene.sample[25], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=weakaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[26],] <- makeSignificant(gene.sample[26], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=weakaffect,
beta.leukxsample=1,
beta.genderxsample=1)

exprsDat2[gene.sample[27],] <- makeSignificant(gene.sample[27], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=weakaffect,
beta.genderxsample=1)

exprsDat2[gene.sample[28],] <- makeSignificant(gene.sample[28], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=1,
beta.leukxsample=weakaffect,
beta.genderxsample=1)


exprsDat2[gene.sample[29],] <- makeSignificant(gene.sample[29], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=weakaffect,
beta.leukxsample=weakaffect,
beta.genderxsample=1)

exprsDat2[gene.sample[30],] <- makeSignificant(gene.sample[30], beta.int = 1,
beta.leuk = weakaffect,
beta.gender=1,
beta.sample=1,
beta.leukxgender=weakaffect,
```

```
beta.leukxsample=weakaffect,
beta.genderxsample=1)


#Normalizing the gene expression data as described by efron
NormalizedGeneDat <-  apply(exprsDat2, 2, function(z) qnorm((rank(z)-0.5)/nrow(exprsDat2)))


NormalizedGeneDat <- exprsDat2

#Efron dropped the most extreme value so we will too
#which(which(LeukStats$linear.p > 0.3)==6777))
#[1] 4781
#NormalizedGeneDat <- NormalizedGeneDat[-4781,]

NormalizedGeneDat_t <- t(NormalizedGeneDat)
N <- nrow(NormalizedGeneDat)


#Combining the normalized gene expression with the covariate information
LeukDat <- cbind(NormalizedGeneDat_t,imp.dat)
################################################################################

#######################################
#Save Statistics from traditional method
#######################################

LeukStats.ps <- data.frame(
linear.p = rep(NA, nrow(NormalizedGeneDat)),
linear.bonf = rep(NA, nrow(NormalizedGeneDat)),
linear.fdr = rep(NA, nrow(NormalizedGeneDat)),
linear.z = rep(NA, nrow(NormalizedGeneDat)),
linear.aic = rep(NA,nrow(NormalizedGeneDat)),
linear.r2 = rep(NA,nrow(NormalizedGeneDat)),
linear.r2.adjusted = rep(NA,nrow(NormalizedGeneDat)),
linear.emp.p = rep(NA, nrow(NormalizedGeneDat)),
linear.emp.bonf = rep(NA, nrow(NormalizedGeneDat)),
linear.emp.fdr = rep(NA, nrow(NormalizedGeneDat)))

for(xx in 1:nrow(NormalizedGeneDat)){
    linear.mod <- lm(LeukDat[,xx] ~ ALL.AML, data = LeukDat)
    z.sign <- ifelse(summary.lm(linear.mod)$coefficients['ALL.AML2','t value']>=0,1,-1)
    z.tmp <- abs(qnorm(summary.lm(linear.mod)$coefficients['ALL.AML2','Pr(>|t|)']/2))*z.sign
    z.tmp <- ifelse(z.tmp == 'Inf', 10, z.tmp)
    z.tmp <- ifelse(z.tmp == '-Inf', -10, z.tmp)
    LeukStats.ps[xx,c('linear.z')] <-  z.tmp
    LeukStats.ps[xx,'linear.p'] <- summary(linear.mod)$coefficients['ALL.AML2','Pr(>|t|)']
    #LeukStats.ps[xx,'linear.p'] <-  2*(1- pnorm(abs(z.tmp), mean=0,sd=1))
    LeukStats.ps[xx,'linear.aic'] <- AIC(linear.mod)
    LeukStats.ps[xx,'linear.r2'] <- summary(linear.mod)$r.squared
    LeukStats.ps[xx,'linear.r2.adjusted'] <-summary(linear.mod)$adj.r.squared
}
LeukStats.ps[,'linear.bonf'] <- Bonferroni(LeukStats.ps[,'linear.p'])
LeukStats.ps[,'linear.fdr'] <- Global.fdr(LeukStats.ps[,'linear.p'])

#Calculate p value from the empirical mean and variance
tmp.fdr.emp <- locfdr(LeukStats.ps[,c('linear.z')],nulltype=1,plot=0)
emp.p <- 2*(1- pnorm(abs(LeukStats.ps[,c('linear.z')]),
               mean=tmp.fdr.emp$fp0['mlest','delta'],
               sd=tmp.fdr.emp$fp0['mlest','sigma'])
           )
emp.p <- ifelse(emp.p >1,1,emp.p)

LeukStats.ps[,'linear.emp.p'] <- emp.p
LeukStats.ps[,'linear.emp.bonf'] <- Bonferroni(LeukStats.ps[,'linear.emp.p'])
LeukStats.ps[,'linear.emp.fdr'] <- Global.fdr(LeukStats.ps[,'linear.emp.p'])

################################################################################
set.seed(MYSEED)
n.sim = 20

bagged.res.linear <- lapply(1:N, function(i) data.frame(z.gene = rep(NA,n.sim),
                                                        aic = rep(NA,n.sim),
                                                        r2 = rep(NA,n.sim),
```

```
                                                r2.adjusted = rep(NA,n.sim),
                                                emp.loc.fdr = rep(NA,n.sim),
                                                emp.p = rep(NA,n.sim),
                                                emp.var.p = rep(NA, n.sim),
                                                null.loc.fdr = rep(NA,n.sim),
                                                null.p = rep(NA,n.sim),
                                                emp.Fdr = rep(NA,n.sim),
                                                null.Fdr = rep(NA,n.sim),
                                                emp.Bon = rep(NA,n.sim),
                                                null.Bon = rep(NA,n.sim),
                                                emp.mean = rep(NA, n.sim),
                                                emp.sd = rep(NA,n.sim),
                                                which.mod = rep(NA,n.sim)))

for(i in 1:n.sim){
  #data frame of z statistics to get a single local fdr
  all.stats <- lapply(1:8,function(i) data.frame(z.gene = rep(NA,N),
                                                 aic = rep(NA,N),
                                                 r2 = rep(NA,N),
                                                 r2.adjusted = rep(NA,N),
                                                 emp.loc.fdr = rep(NA,N),
                                                 emp.p = rep(NA,N),
                                                 emp.var.p = rep(NA,N),
                                                 null.loc.fdr = rep(NA,N),
                                                 null.p = rep(NA,N),
                                                 emp.Fdr = rep(NA,N),
                                                 null.Fdr = rep(NA,N),
                                                 emp.Bon = rep(NA,N),
                                                 null.Bon = rep(NA,N),
                                                 emp.mean = rep(NA, N),
                                                 emp.sd = rep(NA,N)))

  boot.samp.ALL <- sample(which(LeukDat[,'ALL.AML']=='ALL'),replace = TRUE)
  boot.samp.AML <- sample(which(LeukDat[,'ALL.AML']=='AML'),replace = TRUE)
  tmp.dat <- rbind(LeukDat[boot.samp.ALL,],LeukDat[boot.samp.AML,])

  d <- datadist(tmp.dat)
  options(datadist="d")

  for(xx in 1:N){

    m.1 <- tryCatch(ols(tmp.dat[,xx] ~ ALL.AML, data = tmp.dat), error=function(e) {NA})

    #Single Covariates
    m.2 <-tryCatch( ols( tmp.dat[,xx] ~ ALL.AML + BM.PB , data = tmp.dat), error=function(e) {NA})
    m.3 <- tryCatch(ols( tmp.dat[,xx] ~ ALL.AML + Gender, data = tmp.dat), error=function(e) {NA})

    #Two Covariates
    m.4 <- tryCatch(ols( tmp.dat[,xx] ~ ALL.AML + BM.PB + Gender, data = tmp.dat), error=function(e) {NA})

    #Interactions
    m.5 <- tryCatch(ols( tmp.dat[,xx] ~ ALL.AML*BM.PB + Gender, data = tmp.dat), error=function(e) {NA})
    m.6 <- tryCatch(ols( tmp.dat[,xx] ~ ALL.AML*Gender + BM.PB, data = tmp.dat), error=function(e) {NA})

    m.7 <- tryCatch(ols( tmp.dat[,xx] ~ ALL.AML + Gender*BM.PB, data = tmp.dat), error=function(e) {NA})

    m.8 <- tryCatch(ols( tmp.dat[,xx] ~ ALL.AML*Gender + ALL.AML*BM.PB, data = tmp.dat), error=function(e) {NA})



    for(j in 1:4){
      #m.tmp <- get(paste('m.',j,sep = ""))
      m.tmp <- eval(parse(text=paste('m.',j,sep = "")))
      if(all(is.na(m.tmp))) {
        all.stats[[j]][xx,c('z.gene')] <- NA
        all.stats[[j]][xx,c('aic')] <- NA
        all.stats[[j]][xx,c('r2')] <- NA
        all.stats[[j]][xx,c('r2.adjusted')] <- NA

      } else {
        z.sign <- ifelse(summary.lm(m.tmp)$coefficients['ALL.AML=AML','t value']>=0,1,-1)
        z.tmp <- abs(qnorm(summary.lm(m.tmp)$coefficients['ALL.AML=AML','Pr(>|t|)']/2))*z.sign
        z.tmp <- ifelse(z.tmp == 'Inf', 10, z.tmp)
```

```
      z.tmp <- ifelse(z.tmp == '-Inf', -10, z.tmp)
      all.stats[[j]][xx,c('z.gene')] <-  z.tmp
      all.stats[[j]][xx,c('aic')] <- AIC(m.tmp)
      all.stats[[j]][xx,c('r2')] <- m.tmp$stats['R2']
      all.stats[[j]][xx,c('r2.adjusted')] <- summary.lm(m.tmp)[['adj.r.squared']]
    }
  }
  for(k in 5:8) {
    m.tmp <- get(paste('m.',k,sep = ""))

    if(all(is.na(m.tmp))) {
      all.stats[[k]][xx,c('z.gene')] <-NA
      all.stats[[k]][xx,c('aic')] <- NA
      all.stats[[k]][xx,c('r2')] <- NA
      all.stats[[k]][xx,c('r2.adjusted')] <- NA
    } else {
      all.stats[[k]][xx,c('z.gene')] <- tryCatch(qnorm(anova(m.tmp)[1,'P'],lower.tail = FALSE), error=function(e) {NA})
      all.stats[[k]][xx,c('z.gene')] <- ifelse(all.stats[[k]][xx,c('z.gene')]=='Inf',10,
                                               ifelse(all.stats[[k]][xx,c('z.gene')]=='-Inf',-10,
                                                      all.stats[[k]][xx,c('z.gene')]))
      all.stats[[k]][xx,c('aic')] <- AIC(m.tmp)
      all.stats[[k]][xx,c('r2')] <- m.tmp$stats['R2']
      all.stats[[k]][xx,c('r2.adjusted')] <- summary.lm(m.tmp)[['adj.r.squared']]
    }
  }
}

BEN.fdr.p <- for(qq in 1:8) {

  tmp.mod <- all.stats[[qq]]

  tmp.dat.cc <- tmp.mod[complete.cases(tmp.mod[,'z.gene']),'z.gene']

  tmp.fdr.emp <- tryCatch(locfdr(tmp.dat.cc,nulltype=1,plot=0),
                          error=function(e) {NA})

  tmp.fdr.null <- tryCatch(locfdr(tmp.dat.cc,nulltype=0,plot=0),
                           error=function(e) {NA})

  if(length(tmp.fdr.emp)>1){
    emp.loc.fdr <- tmp.fdr.emp$fdr

    emp.p <- min(2*(1- pnorm(abs(tmp.dat.cc),
                         mean=tmp.fdr.emp$fp0['mlest','delta'],
                         sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)

    emp.var.p <- min(2*(1- pnorm(abs(tmp.dat.cc),
                           mean=0,
                           sd=tmp.fdr.emp$fp0['mlest','sigma'])),1)

    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.loc.fdr'] <- emp.loc.fdr
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.p'] <- emp.p
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.var.p'] <- emp.var.p
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Fdr'] <- Global.fdr(emp.p)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Bon'] <- Bonferroni(emp.p)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.mean'] <- tmp.fdr.emp$fp0['mlest','delta']
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.sd'] <- tmp.fdr.emp$fp0['mlest','sigma']
  }
  if(all(is.na(tmp.fdr.emp))){
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.loc.fdr'] <- NA
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.p'] <- NA
    all.stats[[qq]][complete.cases( all.stats[[qq]][,'z.gene']),'emp.var.p'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Fdr'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.Bon'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.mean'] <- NA
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'emp.sd'] <- NA
  }
  if(length(tmp.fdr.null)>1){
    null.loc.fdr <- tmp.fdr.null$fdr
    null.p <- 2*(1- pnorm(abs(tmp.dat.cc), mean=0,sd=1))
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.loc.fdr'] <- null.loc.fdr
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.p'] <- null.p
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Fdr'] <- Global.fdr(null.p)
    all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Bon'] <- Bonferroni(null.p)
```

```
    }
    if(is.na(tmp.fdr.null)){
      all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.loc.fdr'] <- NA
      all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.p'] <- NA
      all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Fdr'] <- NA
      all.stats[[qq]][complete.cases(all.stats[[qq]][,'z.gene']),'null.Bon'] <- NA
    }
  }

  tmp.all.data <- do.call(rbind, all.stats) # dim(X2) = 7000*17 x ncol

  grp <- rep(1:N, 8)
  split.by.genes <- split(tmp.all.data, grp)

  #Now store the BEN statistics
  for(j in 1:N){
    best.mod <- which(split.by.genes[[j]][,'aic']==min(split.by.genes[[j]][,'aic'], na.rm = TRUE))
    if(length(best.mod)>1){best.mod <- sample(best.mod, 1)}
    stats <- unlist(c(split.by.genes[[j]][best.mod,c('z.gene',
                                                     'aic',
                                                     'r2',
                                                     'r2.adjusted',
                                                     'emp.loc.fdr',
                                                     'emp.p',
                                                     'emp.var.p',
                                                     'null.loc.fdr',
                                                     'null.p',
                                                     'emp.Fdr',
                                                     'null.Fdr',
                                                     'emp.Bon',
                                                     'null.Bon',
                                                     'emp.mean',
                                                     'emp.sd')],best.mod))

    bagged.res.linear[[j]][i,c('z.gene',
                               'aic',
                               'r2',
                               'r2.adjusted',
                               'emp.loc.fdr',
                               'emp.p',
                               'emp.var.p',
                               'null.loc.fdr',
                               'null.p',
                               'emp.Fdr',
                               'null.Fdr',
                               'emp.Bon',
                               'null.Bon',
                               'emp.mean',
                               'emp.sd',
                               'which.mod')] <- stats
  }
}
```

CHAPTER 4

# AN EMPIRICAL STUDY OF MODEL PREDICTION WHEN USING THE RESPONSE FOR IMPUTATION OF MISSING COVARIATES: RECOMMENDATIONS FOR INFERENCE AND VALIDATION

## 4.1 Abstract

Missing data are a common problem for the construction, validation, and implementation of a prediction model. Multiple Imputation (MI) is the standard algorithm used to fill in missing covariate information for inference of a regression model. Inclusion of the outcome ($Y$) in the imputation model is known to produce unbiased and efficient parameter estimates during the model construction. Here we show that using the outcome in the imputation model when data are missing in the out-of-sample validation set, leads to imputations whose corresponding risk predictions will result in artificially increased model discrimination statistics. Specifically, we show that for a logistic prediction model, the validated AUC, Brier and Logarithmic scores are optimistically biased when the outcome is assumed to be known during imputation of covariates. The outcome would not be available during model application, and the imputation algorithm used during model validation should take this into consideration. Simulations and an analysis of the TREAT lung cancer prediction model, confirm the need to more carefully consider how the outcome is used for imputation during each stage of model development. Model fit statistics are validated within each missingness pattern, so that we can more clearly see how a prediction model marginalized over missing data pattern would perform for a new out-of-sample individual missing specific model covariates. We find that the biased model fit statistics in the validation and implementation phases of model development are impervious to missing data mechanism. We provide practical recommendations for the circumstances where it is appropriate to include or exclude the outcome in the multiple imputation algorithm.

## 4.2 Introduction

Statistical prediction models are often used in clinical practice to evaluate a patient's unknown probability of risk of having a particular disease or outcome. We will discuss three main stages in the development of a prediction model: (1) The construction of a prediction model using in-sample data (covariates known or missing, and outcome known), (2) the validation of a prediction model using a different out-of-sample population from which the model was initially constructed (covariates known or missing, and outcomes known), and (3) the application of a risk prediction model to a new person in the clinic (covariates known or missing, and outcome unknown). It is often the case in biomedical data that

model covariates are missing, and these missing values need to be dealt with uniquely for each of these three stages of model development.

Imputation is the practice of filling in missing values with a 'best guess'. Imputation has been widely discussed in the prediction model literature, with the consensus advocating that multiple imputation is necessary for accurate effect estimates during the first model building stage (Little and Rubin, 2014; Little, 1992; Rubin, 1996; Harrell, 2013; Janssen et al., 2010; Schafer, 1997; Meyer and Windeler, 2009; Vergouwe et al., 2010; Schafer and Graham, 2002; Clark and Altman, 2003; Vach, 2012; Vach and Blettner, 1991; Greenland and Finkle, 1995; Schafer, 1999; Ibrahim, 2012; Barnard and Meng, 2016; Bartlett et al., 2012; Rubin and Schenker, 1991).

Specifically, Moons et al. (2006) detail the advantages of including the outcome, $Y$, in the imputation model. Using $Y$ in the imputation model during model construction leads to unbiased estimates of regression coefficients (Moons et al., 2006). Whereas this may be a fine approach during the model building (in-sample) population, if $Y$ is assumed to be known and then is included in the multiple imputation model for the validation (out-of-sample) population, model fit statistics (AUC, Brier Score, $R^2$, MSE) will be artificially inflated. Furthermore, including the outcome in an imputation model when applying a prediction model to a new patient in the clinic where their outcome is unknown, is not practical. The focus of this paper is on imputation with relation to performance of a prediction model, and best practice for inclusion and exclusion of model parameters is not explored. We will show through simulation and example that careful consideration of the imputation model with regards to the outcome needs not to be overlooked.

### 4.2.1 Missing Data and Multiple Imputation

Missing data can occur according to three mechanisms. If missing data is ignorable, i.e. Missing Completely at Random (MCAR), then imputation is not necessary. However non-ignorable missingness such as Missing at Random (MAR) and Missing Not at Random (MNAR) requires more thoughtful consideration during the data imputation process. MCAR implies that the missing data pattern is completely random and is not conditional on the observed data. MAR assumes that the missing data mechanism is conditional on another observed variable but not on the missing variable itself. MNAR random implies that the missing value is dependent on the missing observation itself. For example- if a patient is missing the variable weight it could be because (1) The scale happened to be broken the day she arrived for her appointment (MCAR), (2) the patient is a child and the scale is for adults (MAR), or (3) the scale often fails when a weight is above or below some threshold (MNAR). The missing data mechanism may also be conditional on the observed response. If the missing covariate information is not missing in connection with the outcome, either directly or indirectly, complete case analysis or mean imputation may still

yield valid results (Little and Rubin, 2014; Vach, 2012; Vach and Blettner, 1991). However, it is often the case in biomedical data that missing covariate information is missing related to the outcome, and these scenarios will be discussed in Section 4.4.3.

When data are MAR or MNAR simple techniques to impute missing values often result in biased estimates of model parameters and an underestimate of standard errors (Rubin, 1996). Therefore, more sophisticated methods of imputations are required to fill-in missing covariates. Conditional Imputation (CI), based on available patient characteristics has be promoted to fill in missing values. Multiple Imputation (MI), although more computationally intensive, has the added benefit of incorporating the uncertainty of the imputed values (Schafer and Graham, 2002). MI is an imputation method that uses conditional distributions to draw multiple replacement values that are most likely given the observed data (van Buuren, 2012; Harrell, 2013; Janssen et al., 2010). To do this, a missing observation is iteratively imputed $m$ times (there are several available algorithms) based on the observed values for a given individual, and then $m$ complete data sets are produced. Each complete dataset is subjected to the same statistical analysis, and results are pooled using Rubin's rules (Rubin, 1976). MI procedures, particularly Multivariate Imputation by Chained Equations (MICE), are highly flexible and can be used in a wide range of settings. The MICE algorithm calculates a series of regression models where each variable with missing data is modeled conditional upon the other variables in the data (Azur et al., 2011).

The MICE method can use several methods of imputation, however, predictive mean matching (PMM) is the imputation method we use in our examples and simulations. PMM is a semi-parametric imputation approach, that fills in a value randomly from among observed donor values from an observation whose regression-predicted values are closest to the regression-predicted value for the missing value from the simulated regression model (Heitjan and Little, 1991; Schenker and Taylor, 1996). The advantage of PMM, compared to other regression based imputation strategies (which assumes a joint multivariate normal distribution), is that PMM ensures the missing values are filled in with plausible observations (Horton and Lipsitz, 2012).

### 4.2.2 Imputation During Model Development, Validation, and Application

These MI methods are advocated for prediction model derivation and application (Harrell, 2013; Janssen et al., 2009). MI is currently the gold standard for imputation methods because it is more efficient than other methods by using all the information in the incomplete cases. Moons et al. (2006) suggests using the outcome for the imputation of missing predictor values, showing regression coefficients based on MI including the outcome were close to the true parameter values, and MI excluding the outcome resulted in underestimated model coefficients. Using the outcome in the imputation algorithm is considered the best imputation practice during model construction, and we similarly use this imputation

procedure for the construction of all prediction models considered in this paper.

When validating a risk prediction model, either by using cross-validation or a new out-of-sample population, missing data is also likely to occur. Since many prediction models are developed for use in the clinic on a new patient, model validation is an important step in order to assess how a statistical model will generalize to an independent set (Geisser, 1993; Kohavi, 1995). Since a risk model is predicting future events or outcomes, using the outcome for imputation of missing covariates is circular. For example, a person's blood pressure may be a predictor in a model which estimates the risk of cardiovascular disease. If blood pressure is missing, and cardiovascular disease is used in the imputation model of blood pressure, the resulting fill-in value for blood pressure will not be independent of the outcome. This imputed blood pressure will then in turn be used to predict the probability of cardiovascular disease, thus creating a 'too perfect' prediction and will lead to bias in the validation metrics of the prediction model. Although this seems obvious, in a validation sample where the outcome is collected and known, it is tempting to want to include it in the imputation algorithm without considering the effects on the 'Pragmatic Model Performance', what Wood et al. (2015) defines as a model's performance in a future clinical setting where some individuals may have partly missing predictors.

Janssen et al. (2009) and Wood et al. (2015) have previously explored missing data during model application. In this setting for MI to be an option, the model user to have access to the original data, a similar conditional distribution to make multiple draws, and a computer program to combine prediction results. To obtain the best predictions from a model that was derived using multiple imputation,Vergouwe et al. (2010) and Wood et al. (2015) recommend obtaining predictions from every model fitted to each imputed dataset, and then combining those predictions using Rubin's Rules. This may be unfeasible when using a prediction model in the clinic, where the only thing known are the pooled model covariates and standard errors.

The computational burden of MI, coupled with the required imputation algorithm to obtain accurate imputations or previously saved imputed data sets, often make this exact procedure impossible to perform in the clinic. Therefore, after model construction we consider our pooled prediction model final and fixed for both the validation set and the application stage. We also assume both our validation sample would have one-by-one enrollment, that is to say, an entire new cohort would not be available for a new out-of-sample imputation algorithm. This is not an important assumption if the validation set comes from the same population as the in-sample population.

The occurrence of missing data must be carefully considered in every phase of the development, testing, and application of a prediction model, and the imputation strategy may need to be adjusted to most accurately reflect the model's true performance. Through simulations and a re-analysis of empirical lung cancer study data, we quantified the effect

on discrimination and validation metrics, when the outcome was included and excluded as part of the imputation model and validation sample.

## 4.3 Methods

## 4.4 Motivating Clinical Example

The TREAT model for predicting lung cancer was developed in 492 individuals who were evaluated for surgery with known or suspected lung cancer (Deppen et al., 2014). The TREAT model depends on a physician having access to the following predictors: age (`age`), BMI (`bmi`), gender (`gender`), smoking pack years (`pack_years`), pre-operative lesion maximum diameter (`ct_size`), spiculation (`spicul`), lesion growth (`growthcat`), previous cancer (`prev_cancer`), any symptoms (`anysympt`), FEV1 predicted (`fev1_pred`), and FDG-PET avidity (`petpos34`). The model was developed using a pre-specified set of candidate variables derived from previously published and validated models, as well as other covariates chosen by thoracic surgeons commonly encountered in the at-risk population.

In this cohort, only 264 individuals had complete data. BMI was missing for 2 individuals, pack years was missing for 8 individuals, spiculation was missing for 19 individuals, lesion growth was missing for 65 individuals, any symptoms was missing for 33 individuals, FEV1 predicted was missing for 50 individuals, and FDG-PET avidity was missing for 109 individuals. The most significant predictor of lung cancer observed in the TREAT model was FDG-PET avidity, the variable with the highest amount of missing data in this population. Using the original data, we applied MI and PMM (10 multiple imputations), fit a multivariable logistic regression model with the same set of variables as the published TREAT model, and performed 5-fold cross validation to calculate prediction model metrics. This process was bootstrapped 500 times and statistics were averaged.

### 4.4.1 Imputation scenarios in model validation

To evaluate the impact the imputation strategy has on the prediction metrics, we consider four scenarios that are possible when missing data occurs in the out-of-sample validation population, or during model application in the clinic. In every case we assume that the imputation model used for developing the clinical prediction model includes the outcome ($Y$) in order to obtain unbiased parameter estimates.

1. The imputation model includes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation.

2. The imputation model includes the outcome ($Y$), and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample.

3. The imputation model excludes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation. Here, since the imputation model does not include the outcome, assuming the outcome is available does not affect imputation of missing covariates.

4. The imputation model excludes the outcome ($Y$), and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample.

Note that scenarios (3) and (4) should produce the similar estimates for the missing covariate values, and both are included for completeness. We estimate a logistic risk prediction model, and investigate the prediction model performance for each of these four scenarios comparing the resulting AUCs, Brier Scores, and Logarithmic Scoring Rules by missing data pattern and overall.

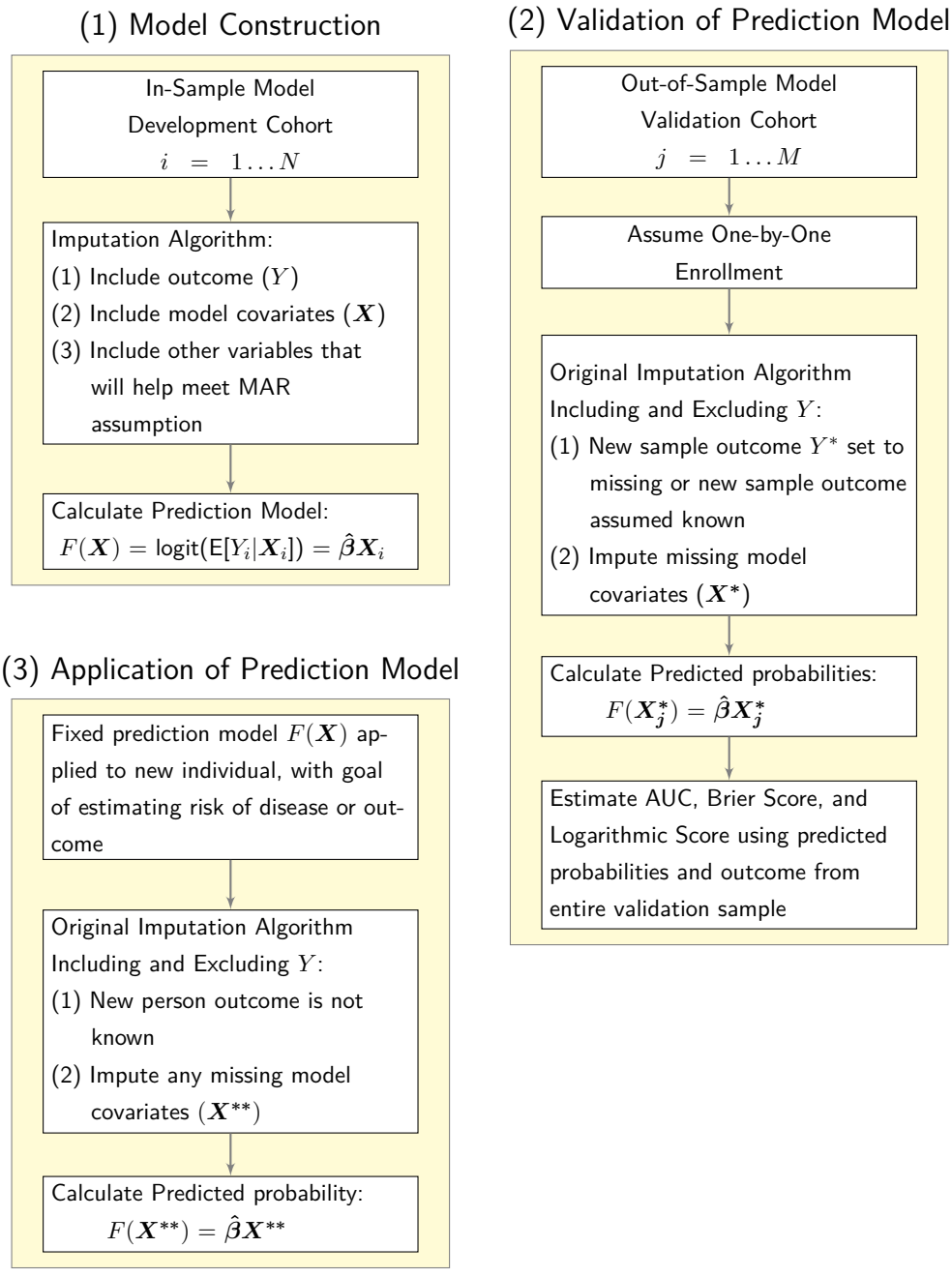
#### 4.4.2 AUC, Brier Score, and Logarithmic scoring rule

The loss function for a logistic regression model requires more thoughtful consideration. The Area Under the Receiver Operating Curve (AUC), is a measure of the models discrimination, however it cannot be collapsed or averaged over each pattern. Therefore, the AUCs need to be compared to the other methods of interest within a missing data pattern. Furthermore, the AUC is a desirable measure because it is not influenced by the number of people within a pattern, however it cannot be calculated for a pattern where all members of the pattern exhibit the same outcome.

Both the Brier score and Logarithmic scoring rules are proper scoring rules used to estimate the accuracy of a risk prediction models, and both scores define arbitrary labels of 0 and 1 for the outcome. The predicted probability is a continuous density for the distribution of possible outcomes, and it is compared to a discrete label. The Brier score is the average squared difference between the labeled outcome and the predicted probability of risk $\text{BS} = \frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$, and the Logarithmic score is defined as $\text{LS} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i \ln(p_i) + (1-y_i)\ln(1-p_i)\right)$, where $y_i$ and $p_i$ are the true outcome and predicted probability respectively, for individual $i$. It is known that the Brier score does not penalize predictions that give very small probabilities when they should be giving larger probabilities, and therefore the Brier score does not necessarily make the right decision about which method of two forecasts is better (Jewson, 2004). The Logarithmic scoring rule (Log score) rewards more extreme predictions that are in the right direction. This score can be grossly inflated by a single prediction of probability of 0 or 1 that is in the wrong direction, and heavily penalizes classifiers that are confident about an incorrect classification. The logarithmic scoring rule is a rescaling of the gold standard optimization criteria and so in a sense it is the best accuracy score to use for binary outcome.

Figure 4.1 depicts the three stages of model development and highlights the places where missing data can occur.

Figure 4.1: Construction, Validation and Application of a Clinical Risk Prediction Model



### 4.4.3 Simulations

We generated $n$ multivariate normal predictor vectors according to $\binom{x_1}{x_2} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = (0, 0)$ and $\boldsymbol{\Sigma} = \left(\begin{smallmatrix} 1 & 0.25 \\ 0.25 & 1 \end{smallmatrix}\right)$, for example, are set to provide certain predictor profiles in terms of their correlation. Simulated outcomes $Y$ are generated from various combinations of $x_1$ and $x_2$. Outcome probabilities are simulated such that {Probability of Outcome} $= p = 1/(1 + \exp(\boldsymbol{X\beta}))$ using the marginal model $\boldsymbol{X\beta} = \beta_0 + \beta_1 X_1 + \beta_2 X_2$. The outcome $Y$ is drawn from the distribution $Y \sim$ Bernoulli$(p)$. Missing data indictors are then induced according to the desired mechanism. A more complex model can always be reduced to a linear combination of non-missing variables, and missing variables, and so this simple example is representative of more complicated situations.

We simulated the following three missing data mechanisms:

1. MCAR: $P(M) = expit(\nu_0)$
2. MAR: $P(M) = expit(\nu_0 + \nu_2 X_2)$
3. MNAR: $P(M) = expit(\nu_0 + \nu_1 X_1)$

We forced the missingness data mechanism to be consistent between the in-sample and out-of-sample populations, and $\nu_0$ is empirically calculated to maintain the desired probability of missingness.

### 4.4.4 Parameters

Parameter profiles explored were $\beta_1 = 1, 3, 5$, $\rho = 0, 0.25, 0.5$, $P(M_1 = 1) = 0.20, 0.50, 0.75$, and $n = 50, 200, 500, 1000$. We present here only one case that was largely representative of our findings: $\beta_1 = 3$, $\rho = 0.25$, $P(M_1 = 1) = 0.50$, and $n = 200$. For the validation and application populations we assumed one-by-one enrollment. Missing data was imputed by multiple imputation (predictive mean matching, 5 imputations). We did not necessarily fix the imputation engine based on the in-sample population. For scenarios (3) and (4) the imputation model was refit from the original in-sample population excluding the outcome $Y$.

### 4.4.5 Simulation procedure

The prediction model was always fit using 'best' methods described by Moons et al. (2006), that is, the imputation engine included the outcome $Y$ in order to obtain unbiased regression coefficients. The full simulation procedure was as follows: (1) data are generated and missing data indicators are generated according to the desired missing data mechanism, as described in Section 4.4.3; (2) missing data are multiply imputed using an imputation algorithm that includes the outcome; (3) the risk prediction model is fit; (4) step 1 is

repeated to obtain a new validation out-of-sample population; (5) individuals are imputed one by one, using one of the four imputation procedures described in Section 4.4.1; (6) individual predictions and performance measures are computed; (7) steps 1 through 6 are repeated 1000 times.

AUCs, Brier scores, and Logarithmic scores were compared for the four imputation scenarios. Validation statistics were compared within a missingness pattern as well as overall missingness patterns.

## 4.5 Results

### 4.5.1 Simulations

Results are presented for the following set of parameters: $\beta_0 = 0, \beta_1 = 3, \beta_2 = 0.5, P(M_1 = 1) = 0.5, \nu_1 = 0.75, \nu_2 = 0.75, \nu_{1,Y} = 0.75, \nu_{2,Y} = 0.75$. Figure 4.2, 4.3 and 4.4, show the 1000 simulation estimates for the simulation procedure. For pattern 2 (green points) where $X_1$ was missing, we considered the 'Best' metric the prediction from the model only including $X_2$: $\text{logit}(Y) = \hat{\beta}_0 + \hat{\beta}_2 X_2$. The average prediction metric for all plots, is the statistic for each pattern weighted by the observed probability of missingness for that pattern.

Including the outcome in the imputation model and assuming the outcome known in the validation sample resulted in severely biased (overestimated) model fit statistics (inflated AUC, and underestimated brier and log scores). In fact, each prediction model metric for this scenario (1), had almost near perfect predictions in the pattern with missing data, and always had better results than the pattern with complete data.The bias was present regardless of missingness type, as well as severity of missingness, correlation between $X_1$ and $X_2$, and proportion of missingness (not shown). Dropping the outcome from the imputation model used in the validation sample produced realistic estimates of the pragmatic prediction model performance. For the Brier and Logarithmic scores the MNAR missing data mechanism has slightly worse prediction model metrics, compared to MCAR and MAR mechanisms. This trend is not seen in the AUC plot results where all missing data mechanisms perform almost identically. As both the strength of the missingness mechanism and the beta coefficient associated with the missing variable increase, the magnitude of the differences in imputation scenarios become more apparent.

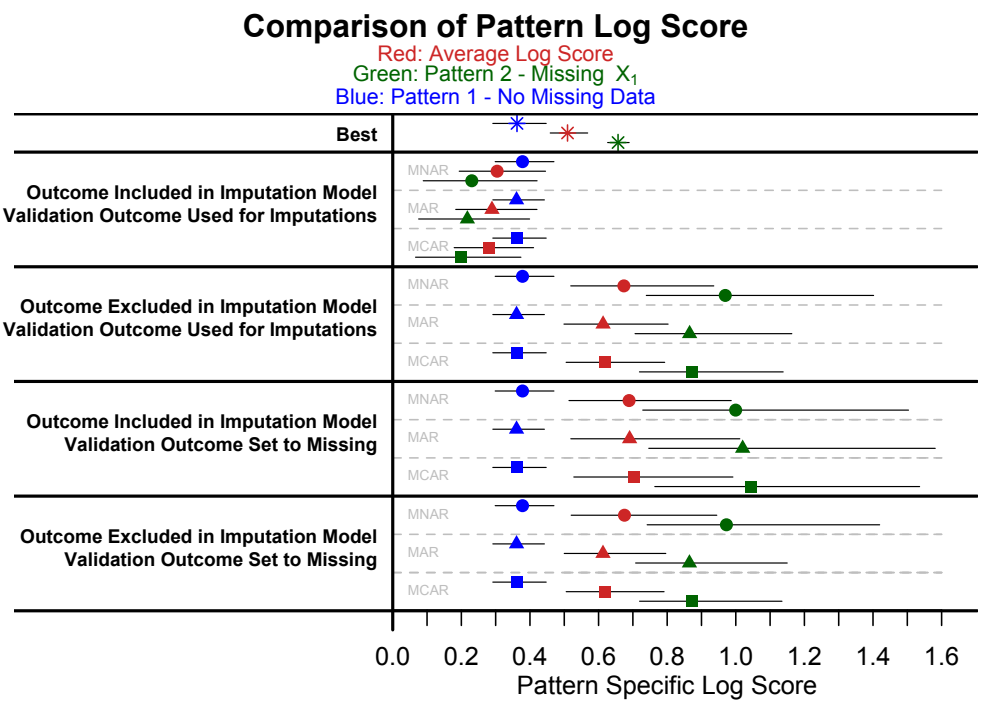



Figure 4.2: Simulation Results set of parameters: 1000 simulations, $\beta_0 = 0, \beta_1 = 3, \beta_2 = 0.5, P(M_1 = 1) = 0.5, \nu_1 = 0.75, \nu_2 = 0.75, \nu_{1,Y} = 0.75, \nu_{2,Y} = 0.75$. The Logarithmic (Log) score is defined as $\text{LS} = \frac{1}{N}\sum_{i=1}^{N}(y_i \ln(p_i) + (1 - y_i)\ln(1 - p_i))$. The missing data mechanisms Missing Completely at Random (MCAR, square points), Missing at Random (MAR, triangular points), and Missing Not at Random (MNAR, circular points) were generated according to section 4.4.3. Red points represent the Total Logarithmic (Log) Score, averaged over all missing data patterns. Blue points represent the Log Score for Pattern 1 where there is no missing data. Green points represent the Log Score for Pattern 2 in which $X_1$ is missing. The black bars represent the bootstrapped 95 percentile interval for each point.

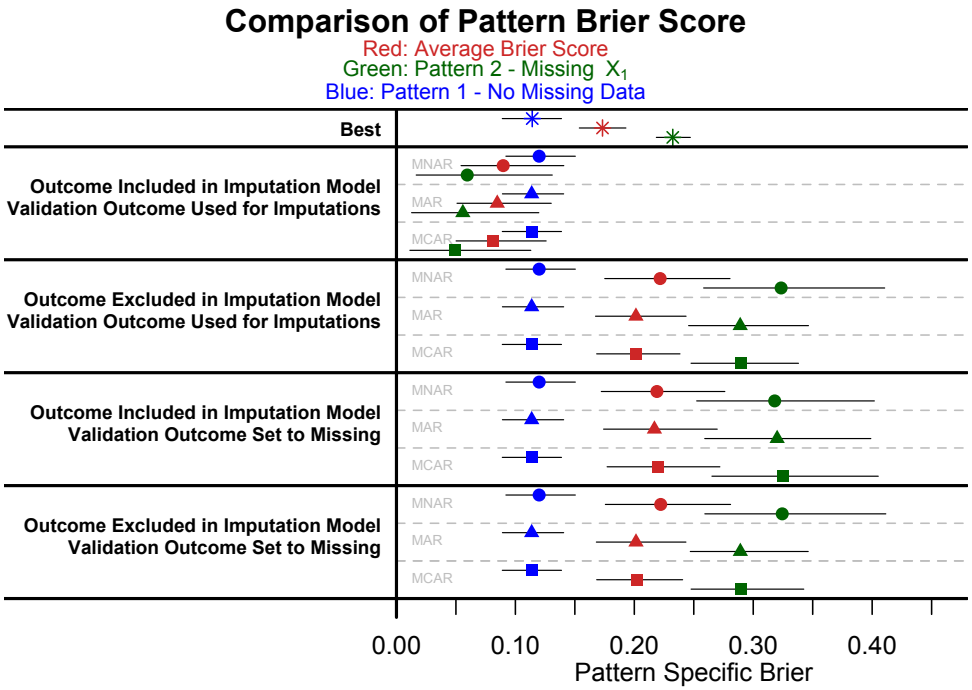



Figure 4.3: Simulation Results set of parameters: 1000 simulations, $\beta_0 = 0, \beta_1 = 3, \beta_2 = 0.5, P(M_1 = 1) = 0.5, \nu_1 = 0.75, \nu_2 = 0.75, \nu_{1,Y} = 0.75, \nu_{2,Y} = 0.75$. The Brier score is defined as $\text{BS} = \frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$. The missing data mechanisms Missing Completely at Random (MCAR, square points), Missing at Random (MAR, triangular points), and Missing Not at Random (MNAR, circular points) were generated according to section 4.4.3. Red points represent the Total Brier Score, averaged over all missing data patterns. Blue points represent the Brier Score for Pattern 1 where there is no missing data. Green points represent the Brier Score for Pattern 2 in which $X_1$ is missing. The black bars represent the bootstrapped 95 percentile interval for each point.

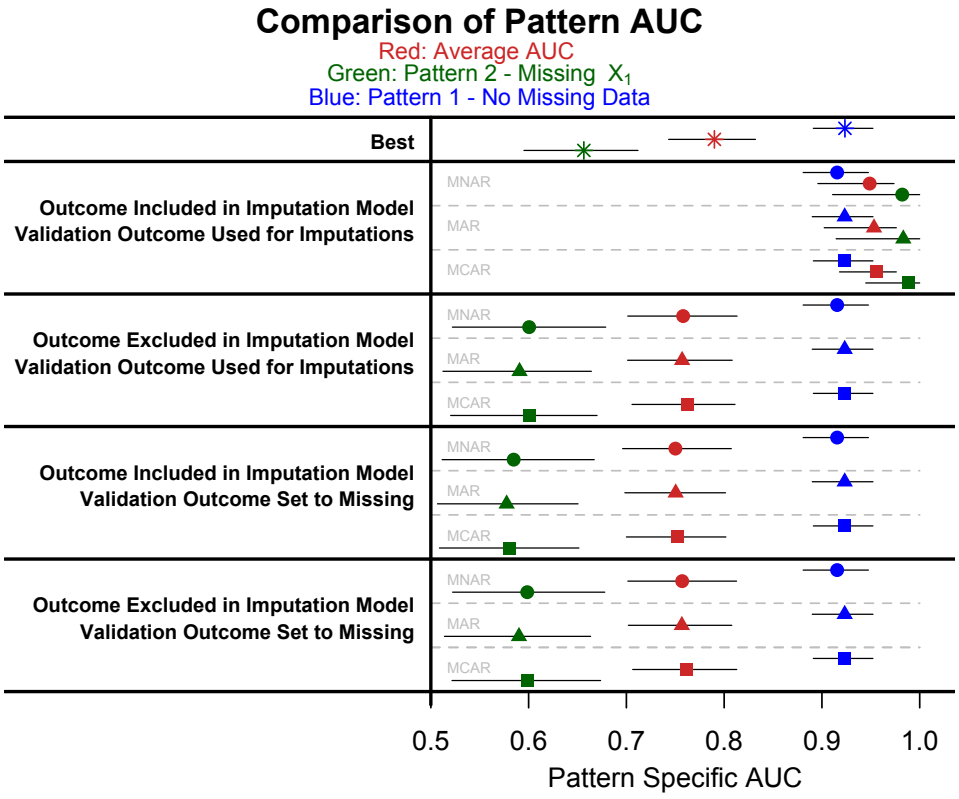



Figure 4.4: Simulation Results set of parameters: 1000 simulations, $\beta_0 = 0, \beta_1 = 3, \beta_2 = 0.5, P(M_1 = 1) = 0.5, \nu_1 = 0.75, \nu_2 = 0.75, \nu_{1,Y} = 0.75, \nu_{2,Y} = 0.75$. The AUC is defined as the Area Under the Receiver Operating Characteristic Curve. The missing data mechanisms Missing Completely at Random (MCAR, square points), Missing at Random (MAR, triangular points), and Missing Not at Random (MNAR, circular points) were generated according to section 4.4.3. Red points represent the Total AUC, averaged over all missing data patterns. Blue points represent the AUC for Pattern 1 where there is no missing data. Green points represent the AUC for Pattern 2 in which $X_1$ is missing. The black bars represent the bootstrapped 95 percentile interval for each point.

### 4.5.2 Imputation Error of $X_1$ in the Validation Sample

In our simulations, using the outcome in the imputation model during model construction leads to unbiased estimates of regression coefficients. However, when the outcome remained in the imputation model, and the outcome was assumed to be missing in the validation sample (scenario 2), the chained equations imputation model can lead to biased imputations for the next missing covariates of the chain (in these simulations $X_1$). In this scenario, since the outcome is missing its imputed value, is also used to help impute the missing value of $X_1$, and therefore adds additional variability to the imputed value of $X_1$.

Table 4.1: Squared Imputation Error of the true out-of-sample $X_1$ compared to the imputed $X_1$ under different imputation methods and missing data mechanisms: Imputation Error of $X_1 = \frac{1}{N}\sum_i (X_{1i} - \hat{X}_{1i})^2$. Multiple Imputation was done the usual way using predictive mean matching and chained equations, where the variable with the least amount of missing data is the first variable imputed ($Y$), and the variable with the next least amount of missing data is imputed second (in this case $X_1$).

| | MCAR | MAR | MNAR |
|---|---|---|---|
| MI (y), Validation Sample (y) | 0.33 (0.04) | 0.34 (0.05) | 0.35 (0.06) |
| MI (y), Validation Sample (no y) | 0.68 (0.07) | 0.67 (0.08) | 0.70 (0.10) |
| MI (no y), Validation Sample (y) | 0.58 (0.05) | 0.59 (0.08) | 0.70 (0.08) |
| MI (no y), Validation Sample (no y) | 0.58 (0.05) | 0.57 (0.08) | 0.70 (0.08) |

This is why we recommend the scenario in which the outcome is used in the in-sample imputation model to produce unbiased regression estimates, but not included in the out-of-sample imputation model - a combination which would as the most appropriate pragmatic model performance statistics.

Table 4.1 of out-of-sample imputations of $X_1$ provides insight into how $X_1$ is imputed for each of the four imputation scenarios. For scenario 1, when $Y$ is included in the imputation model during model construction and validation, the predictions of $X_1$ are the most accurate. For all other imputation scenarios the prediction of $X_1$ are similar. Although the apparent bias in imputations for missing covariates may seem small, their total contribution over all individuals can be quite significant. These results show that downward 'bias' in using the outcome to improve imputations of missing predictors, leads to superior downstream predictions and overly optimistic AUC, Brier, and Log scores.

### 4.5.3 TREAT Model Results

The results from 500 bootstrapped samples, are presented in Figures 4.5, 4.6, and 4.7. For each figure, the subplot showing the observed missing data pattern is presented with an 'X' denoting a variable that is missing, and the number of individuals within that pattern from the original sample is given. Note in Figure 4.7, we could not estimate the AUC in every pattern of missing data because some patterns had only individuals with one outcome.

For each imputation method, five-fold cross validation of the prediction models was implemented. For the TREAT data, all imputation strategies performed similarly, and there was no statistical difference. However, both within patterns and overall, the first imputation strategy where the imputation model includes the outcome (Lung Cancer), and the validation sample assumes that the outcome is available to be used for missing covariate imputation, resulted in inflated AUC, Brier and Logarithmic scores in every case.

In the TREAT model, FDG-PET avidity (petpos34) was the most significant predictor of Lung Cancer. FDG-PET avidity is a dichotomous variable, and we know it's value was often not collected for individuals who were FDG-PET avid, a probable case of MNAR or MNARY. Since this variable, when imputed, could be perfectly accurate, the magnitude of inflation of the prediction metrics is not as exaggerated as in the simulated data. For every pattern where FDG-PET avidity was missing, we saw the most extreme difference between imputation scenario 1 and the other 3 imputation scenarios, and had this variable been continuous these dissimilarities would probably be even more apparent. We induced selective missingness in FDG-PET and the variable lesion diameter size (results not shown), following the same process as described above, and found the same trends as in the simulated data.

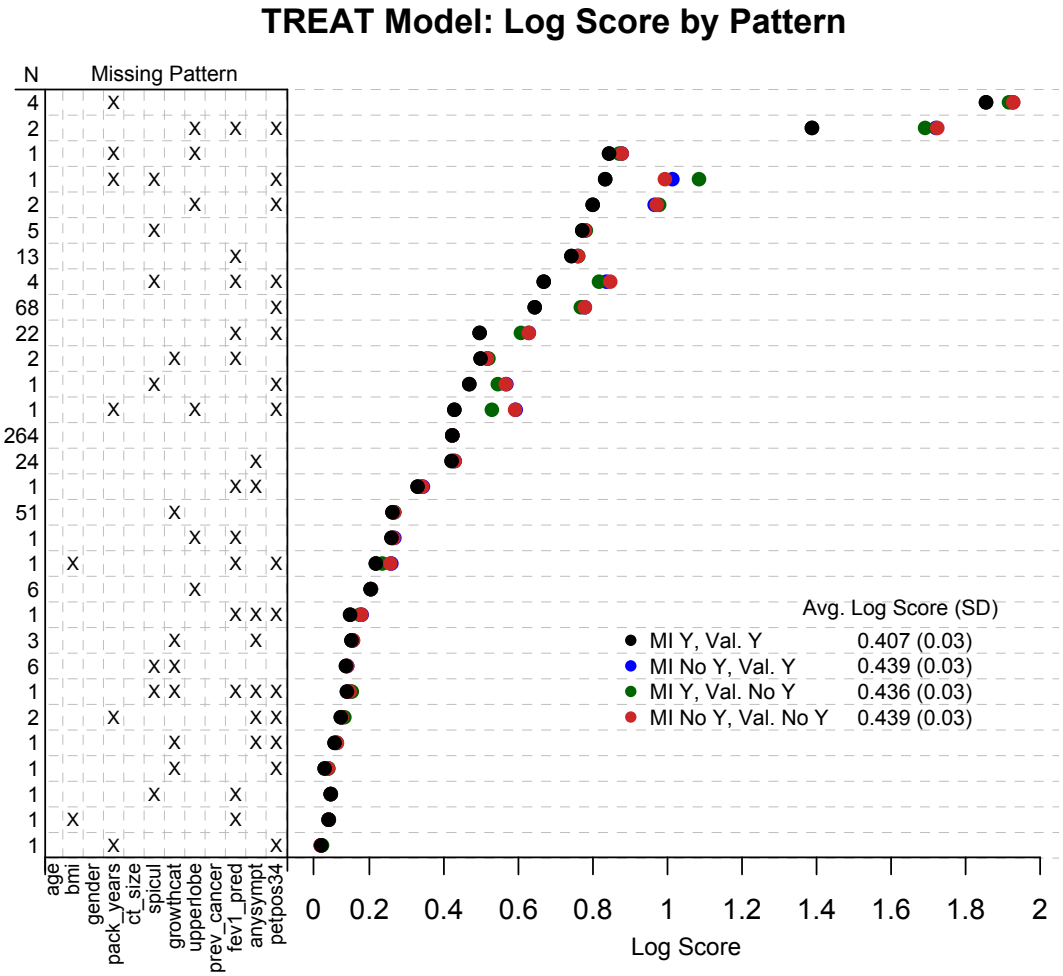



Figure 4.5: The covariates included in the TREAT prediction model include age (`age`), BMI (`bmi`), gender (`gender`), pack years (`pack_years`), pre-operative lesion maximum diameter (`ct_size`), spiculation (`spicul`), lesion growth (`growthcat`), previous cancer (`prev_cancer`), any symptoms (`anysympt`), FEV1 predicted (`fev1_pred`), and FDG-PET avidity (`petpos34`). There are 30 patterns present in the TREAT data, and missing covariates are denoted with 'X'. $N$ is the total number of subjects in each missing data pattern. The Logarithmic (Log) score is defined as $\text{LS} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i \ln(p_i) + (1-y_i)\ln(1-p_i)\right)$. MI Y, Val. Y is scenario 1 where the imputation model includes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation. MI Y, Val No Y is scenario 2 where the imputation model includes the outcome ($Y$), and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample. MI No Y, Val.Y is scenario 3 where the imputation model excludes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation. MI No Y, Val No Y is scenario 4 where the imputation model does not include the outcome, and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample.

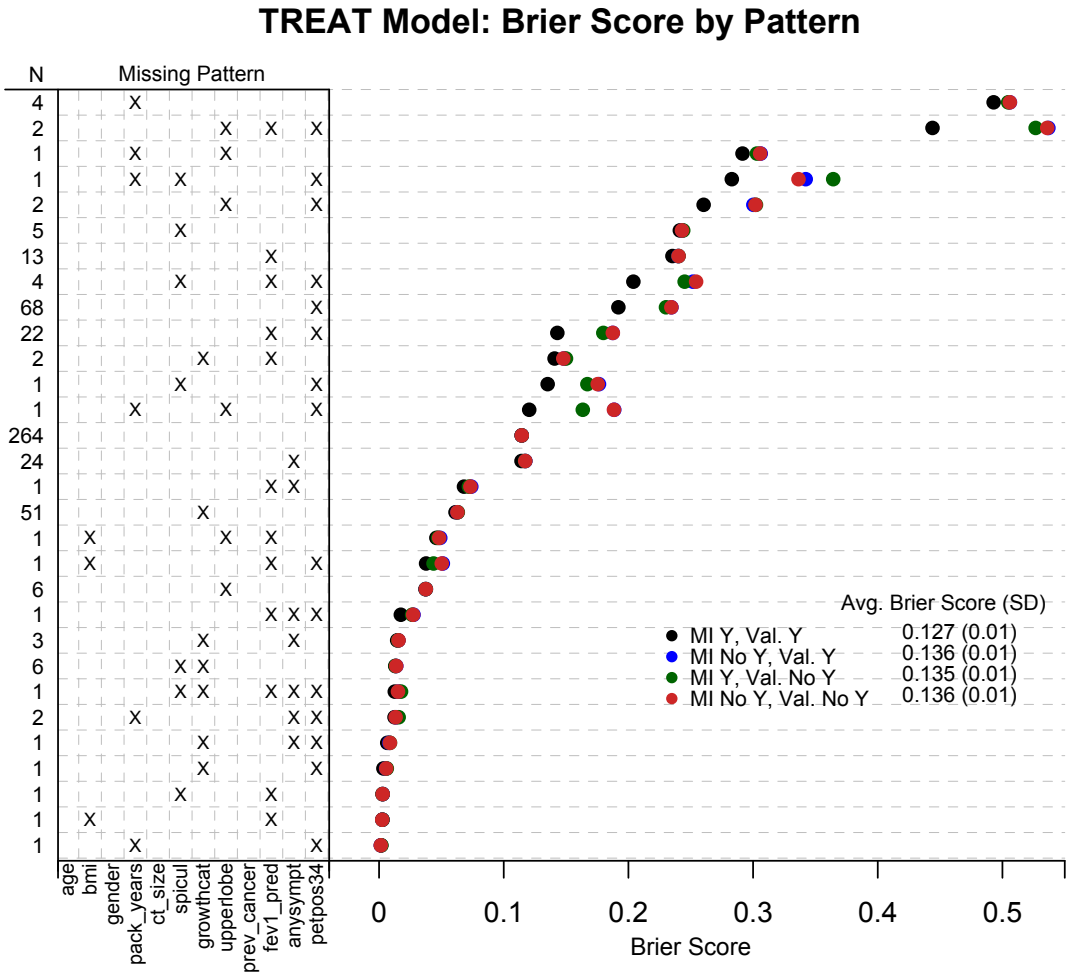



Figure 4.6: The covariates included in the TREAT prediction model include age (`age`), BMI (`bmi`), gender (`gender`), pack years (`pack_years`), pre-operative lesion maximum diameter (`ct_size`), spiculation (`spicul`), lesion growth (`growthcat`), previous cancer (`prev_cancer`), any symptoms (`anysympt`), FEV1 predicted (`fev1_pred`), and FDG-PET avidity (`petpos34`). There are 30 patterns present in the TREAT data, and missing covariates are denoted with 'X'. $N$ is the total number of subjects in each missing data pattern. The Brier score is defined as $\text{BS} = \frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$. MI Y, Val Y is scenario 1 where the imputation model includes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation. MI Y, Val No Y is scenario 2 where the imputation model includes the outcome ($Y$), and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample. MI No Y, Val.Y is scenario 3 where the imputation model excludes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation. MI No Y, Val No Y is scenario 4 where the imputation model does not include the outcome, and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample.

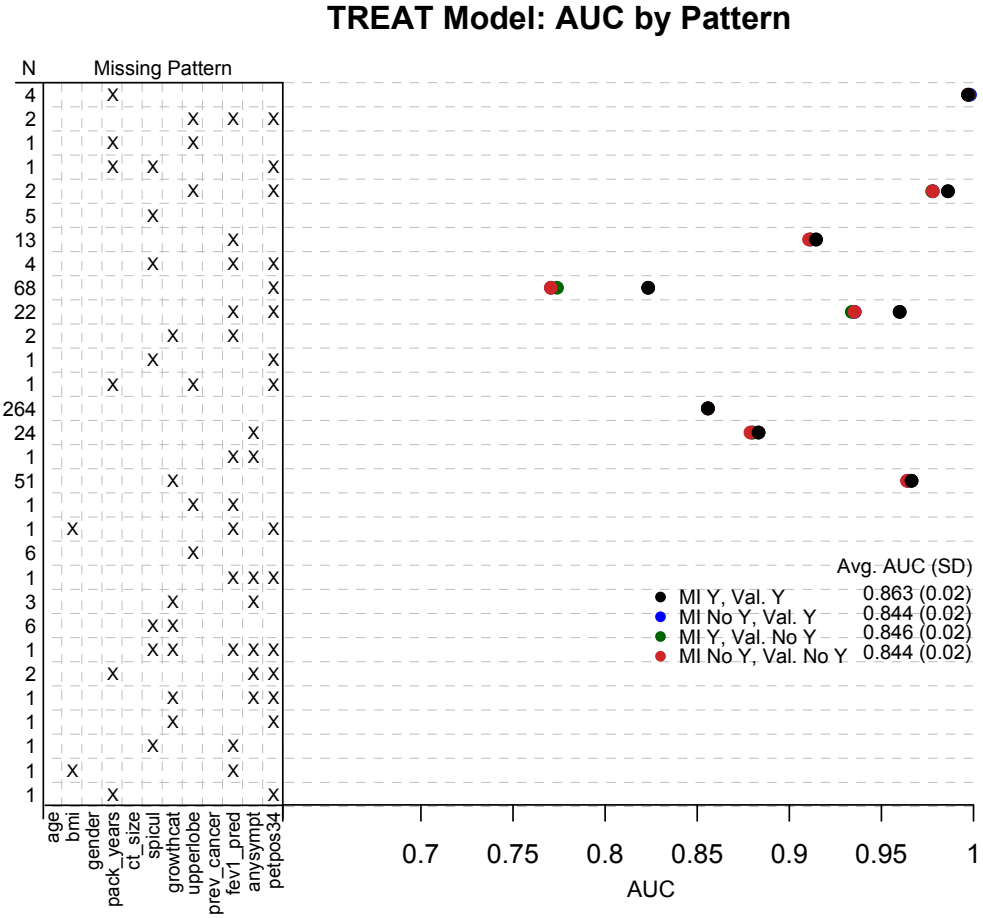



Figure 4.7: The covariates included in the TREAT prediction model include age (`age`), BMI (`bmi`), gender (`gender`), pack years (`pack_years`), pre-operative lesion maximum diameter (`ct_size`), spiculation (`spicul`), lesion growth (`growthcat`), previous cancer (`prev_cancer`), any symptoms (`anysympt`), FEV1 predicted (`fev1_pred`), and FDG-PET avidity (`petpos34`). There are 30 patterns present in the TREAT data, and missing covariates are denoted with 'X'. $N$ is the total number of subjects in each missing data pattern. The AUC is defined as the Area Under the Receiver Operating Characteristic Curve. MI Y, Val. Y is scenario 1 where the imputation model includes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation. MI Y, Val No Y is scenario 2 where the imputation model includes the outcome ($Y$), and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample. MI No Y, Val.Y is scenario 3 where the imputation model excludes the outcome ($Y$), and the validation sample assumes that the outcome is available to be used for missing covariate imputation. MI No Y, Val No Y is scenario 4 where the imputation model does not include the outcome, and although outcome is available in the validation sample, the outcome is excluded to mimic how the model will be used out-of-sample.

## 4.6 Discussion

We have evaluated four imputation strategies both including and excluding the outcome when imputing missing covariate values in the validation set. Based on our simulations and examples, assuming the outcome is known, and including this information in the imputation

model gives overly optimistic prediction model AUCs, Brier scores, and Logarithmic scores. Instead, the imputation techniques used during model validation, should be reflective of the information that would be available during model application. Since the outcome is not known for the model application stage, it should not be considered as a useful variable in the imputation model during model validation.

Furthermore, prediction model metrics should be evaluated both within a missing data pattern and overall. This gives a better representation of the future prediction model performance for an out-of-sample individual with specific missing covariates.

Our motivation comes from the TREAT lung cancer prediction model, where a large amount of missing data was present, and inclusion of the outcome during the validation set imputation gave overly accurate risk predictions that would not have been reflective of data imputation during use of this model in the clinic. We have applied our investigation to simulated data, and have demonstrated that the AUC, Brier score, and Log scores are all optimistic when the outcome is used assumed to be known for the imputation model during validation.

In this setting, where the prediction model will be used on patients with a high probability of missing data, it is highly recommended that the outcome only be in the imputation of missing predictor values during model construction to provide unbiased model parameters. Another imputation algorithm should be in place excluding the outcome, for the imputation of missing covariates in the validation sample. These recommendations extend beyond logistic risk models to other types of prediction models.

## 4.7 Appendix A: Code for TREAT Model Case Study

```
library(rms)

########################
# Functions to use
######################

createFolds <- function (y, k = 10, list = TRUE, returnTrain = FALSE)
{
    if (class(y)[1] == "Surv")
    y <- y[, "time"]
    if (is.numeric(y)) {
        cuts <- floor(length(y)/k)
        if (cuts < 2)
        cuts <- 2
        if (cuts > 5)
        cuts <- 5
        breaks <- unique(quantile(y, probs = seq(0, 1, length = cuts)))
        y <- cut(y, breaks, include.lowest = TRUE)
    }
    if (k < length(y)) {
        y <- factor(as.character(y))
        numInClass <- table(y)
        foldVector <- vector(mode = "integer", length(y))
        for (i in 1:length(numInClass)) {
            min_reps <- numInClass[i]%/%k
            if (min_reps > 0) {
                spares <- numInClass[i]%%k
                seqVector <- rep(1:k, min_reps)
                if (spares > 0)
                seqVector <- c(seqVector, sample(1:k, spares))
                foldVector[which(y == names(numInClass)[i])] <- sample(seqVector)
            }
            else {
                foldVector[which(y == names(numInClass)[i])] <- sample(1:k,
                size = numInClass[i])
            }
        }
    }
    else foldVector <- seq(along = y)
    if (list) {
        out <- split(seq(along = y), foldVector)
        names(out) <- paste("Fold", gsub(" ", "0", format(seq(along = out))),
        sep = "")
        if (returnTrain)
        out <- lapply(out, function(data, y) y[-data], y = seq(along = y))
    }
    else out <- foldVector
    out
}



#This function adds a missing data indicator to every variable
#1 if the variable is missing, and 0 if it is present
create.miss.ind <- function(DATA){
         tmp.dat <- as.data.frame(is.na(DATA)*1)
         names(tmp.dat) <- paste('m.',names(tmp.dat),sep="")
    cbind(DATA,tmp.dat)
}

#This function is the inverse logit function for
#obtaining predited values from linear predictors from a logistic regression
expit <- function(z){
  exp(z)/(1+exp(z))
}

#This function takes two data frames and binds them together based on their column names
rbind.match.columns <- function(input1, input2) {
    n.input1 <- ncol(input1)
    n.input2 <- ncol(input2)
    if (n.input2 < n.input1) {
        TF.names <- which(names(input2) %in% names(input1))
```

```
        column.names <- names(input2[, TF.names])
    } else {
        TF.names <- which(names(input1) %in% names(input2))
        column.names <- names(input1[, TF.names])
    }
    return(rbind(input1[, column.names], input2[, column.names]))
}



auc=function(score,status){
######################
## auc version 1.0
## Compute Area under Rmpirical ROC curve by Trapezoidal Rule
## Author: J. Blume
## Date:   July 2014
######################

        pos=score[status==1]
        neg=score[status==0]

        ct1=sum(outer(pos,neg,">"))
        ct2=sum(outer(pos,neg,"=="))
        den=length(pos)*length(neg)
        auc=(ct1+0.5*ct2)/den
        auc=max(auc,1-auc)
        auc

}



brier.score <- function(pred, outcome){
        mean((pred - outcome)^2)
}

logarithmic.scoring.rule <- function(pred, outcome){

mean(outcome*log(pred) + (1-outcome)*log(1-pred))

}


#Function to find all the observed missing data patterns
which.pattern <- function(DATA, model){

  mod.DATA        <- get_all_vars(as.formula(model), data=DATA)
  SDATA                       <- mod.DATA[,-1] #remove the outcome
  tmp.dat         <- as.data.frame(is.na(SDATA)*1)
  tmp.pattern     <- factor(apply(tmp.dat,1,function(z) paste(z,collapse="")))

   pattern <- tmp.pattern
   pattern
}

#Get metrics by pattern for a logistic model
get_metric <- function(DATA) {
  colnames(DATA) <- c('pred','true','pattern')
  do.call(rbind, by(DATA, DATA['pattern'],function(xx) {
    data.frame('auc'=auc(xx$pred,xx$true),
               'brier'=brier.score(xx$pred,xx$true),
               'log'=logarithmic.scoring.rule(xx$pred,xx$true),
               'prop.pattern' = (length(xx$true)/nrow(DATA)))}))
}

logistic_metrics <- function(DATA) {
  colnames(DATA) <- c('pred','true')
  data.frame('auc'=auc(DATA$pred,DATA$true),
             'brier'=brier.score(DATA$pred,DATA$true),
             'log'=logarithmic.scoring.rule(DATA$pred,DATA$true))}
```

```
#MSE from a fitted model
mse <- function(sm) {
  mse <- mean(sm$residuals^2)
  return(mse)
}

#logit function
logit <- function(p) log(p/(1-p))



############################
# #Application to TREAT data
# ######################################################
# ## get data from shared directory 496 observations and 30 variables

treatdat.orig <- read.delim("~/ImputationWithY/TREATEx/predict out 10162012.txt")


treat.fit = 'cancer ~ age + bmi + gender +
rcs(pack_years,3) + ct_size + spicul +
upperlobe + prev_cancer + fev1_pred + anysympt +
petpos34 + growthcat'

#treatdat.orig$pattern <- which.pattern(DATA=treatdat.orig, model=treat.fit, logistic=TRUE)
treatdat.I <- create.miss.ind(treatdat.orig)
dd <- datadist(treatdat.I)
options(datadist='dd')


########################################
#Bootstrap entire process
########################################
simulation <- function(){
n.boot <- 1
all.results <- vector('list',5)
boot.res <-  data.frame(auc.MI.y        = rep(NA, n.boot),
                        auc.MI.noy      = rep(NA, n.boot),
                        auc.MI.y.testnoy       = rep(NA, n.boot),
                        auc.MI.noy.testnoy      = rep(NA, n.boot),
                        brier.MI.y      = rep(NA, n.boot),
                        brier.MI.noy    = rep(NA, n.boot),
                        brier.MI.y.testnoy      = rep(NA, n.boot),
                        brier.MI.noy.testnoy    = rep(NA, n.boot),
                        log.MI.y                = rep(NA, n.boot),
                        log.MI.noy      = rep(NA, n.boot),
                        log.MI.y.testnoy                = rep(NA, n.boot),
                        log.MI.noy.testnoy      = rep(NA, n.boot)
                        )
boot.res.MIy <- vector('list',n.boot)
boot.res.MInoy <- vector('list',n.boot)
boot.res.MIy.testnoy <- vector('list',n.boot)
boot.res.MInoy.testnoy <- vector('list',n.boot)


for(boot.samp in seq(n.boot)){
boot.nocancer <- sample(which(treatdat.orig$cancer == 0), sum(treatdat.orig$cancer == 0), replace = TRUE)
boot.cancer <- sample(which(treatdat.orig$cancer == 1), sum(treatdat.orig$cancer == 1), replace = TRUE)
treatdat.boot <- rbind(treatdat.orig[boot.nocancer,],treatdat.orig[boot.cancer,])
treatdat.boot.I <- create.miss.ind(treatdat.boot)
dd <- datadist(treatdat.boot.I)
options(datadist='dd')

#####################################################
# Cross Validate the MI and the MIMI models
#####################################################
set.seed(boot.samp)
k=5
flds <- createFolds(treatdat.boot$cancer, k=k, list=TRUE, returnTrain=FALSE)

nimp = 10  #number of imputations

results <- data.frame(auc.MI.y        = rep(NA, k),
```

```
                                    auc.MI.noy      = rep(NA, k),
                                    auc.MI.y.testnoy       = rep(NA, k),
                                    auc.MI.noy.testnoy      = rep(NA, k),
                                    brier.MI.y      = rep(NA, k),
                                    brier.MI.noy    = rep(NA, k),
                                    brier.MI.y.testnoy      = rep(NA, k),
                                    brier.MI.noy.testnoy    = rep(NA, k),
                                    log.MI.y                = rep(NA, k),
                                    log.MI.noy      = rep(NA, k),
                                    log.MI.y.testnoy                = rep(NA, k),
                                    log.MI.noy.testnoy      = rep(NA, k)
                                    )


#Creating a progress bar to know the status of CV
#progress.bar <- create_progress_bar("text")
#progress.bar$init(k)

results.MIy <- NULL
results.MInoy <- NULL
results.MIy.testnoy <- NULL
results.MInoy.testnoy <- NULL


for( i in 1:k){
        # remove rows with id i from dataframe to create training set
        # select rows with id i to create test set

        trainingset <- treatdat.boot.I[-flds[[i]],]
        testset <- treatdat.boot.I[flds[[i]],]

        #Imputation engine including outcome
        TREATimps  <- aregImpute(~ cancer + age + wandw + bmi + gender +
                                   smokeyn +    pack_years + ct_size + spicul + upperlobe + petpos34 + prev_cancer +
        fev1_pred + anysympt + growthcat + weight, n.impute=nimp, x=TRUE, nk=0,
        tlinear=F, data=trainingset, pr=FALSE)


        TREAT <- fit.mult.impute(cancer~ age + bmi + gender + rcs(pack_years,3)
        + ct_size + spicul + growthcat + upperlobe +  prev_cancer +
        fev1_pred + anysympt + petpos34, lrm, TREATimps,
        data=trainingset, pr=FALSE)




############################
# Start Using the test data
############################
imputed.datasets.withy <- replicate(nimp, testset, simplify=FALSE)
imputed.datasets.noy   <- replicate(nimp, testset, simplify=FALSE)
imputed.datasets.withy.testsetnoy <- replicate(nimp, testset, simplify=FALSE)
imputed.datasets.noy.testsetnoy   <- replicate(nimp, testset, simplify=FALSE)

testset.noy <- testset
testset.noy[,'cancer'] <- NA


for(cc in which(!complete.cases(testset))){
        addNewPatient <- rbind.match.columns(trainingset, testset[cc,])
        addNewPatient.noy <- rbind.match.columns(trainingset, testset.noy[cc,])

        Validateimps_y  <- aregImpute(~ cancer + age + bmi +
         gender + pack_years + ct_size
                                                                + spicul + upperlobe + petpos34 +
                                                        prev_cancer + fev1_pred + anysympt
                                                        + growthcat, n.impute=nimp,
                                                        x=TRUE, nk=3,
                                                        tlinear=F, data=addNewPatient, pr=FALSE)

        #Impute Validation Set without Y
        Validateimps_noy  <- aregImpute(~ age + bmi + gender + pack_years +
        ct_size + spicul + upperlobe + petpos34 +
                                                                prev_cancer +  fev1_pred + anysympt +
```

```
                                                        growthcat, n.impute=nimp, x=TRUE, nk=3,
                                                        tlinear=F, data=addNewPatient, pr=FALSE)

       Validateimps_y.testnoy  <- aregImpute(~ cancer + age + bmi +
        gender + pack_years + ct_size
                                                                    + spicul + upperlobe + petpos34 +
                                                    prev_cancer + fev1_pred + anysympt
                                                    + growthcat, n.impute=nimp,
                                                    x=TRUE, nk=3,
                                                    tlinear=F, data=addNewPatient.noy, pr=FALSE)

       #Impute Validation Set without Y
       Validateimps_noy.testnoy  <- aregImpute(~ age + bmi + gender + pack_years +
        ct_size + spicul + upperlobe + petpos34 +
                                                        prev_cancer +  fev1_pred + anysympt +
                                                        growthcat, n.impute=nimp, x=TRUE, nk=3,
                                                        tlinear=F, data=addNewPatient.noy, pr=FALSE)




       for (j in 1:nimp){

           tmp.imp.y <- as.data.frame(impute.transcan(Validateimps_y,  imputation=j,   data=addNewPatient,
              list.out=TRUE,pr=FALSE,  check=TRUE),stringsAsFactors=FALSE)[nrow(addNewPatient),]

        imputed.datasets.withy[[j]][cc,names(tmp.imp.y) ] <- tmp.imp.y

        tmp.imp.noy <- as.data.frame(impute.transcan(Validateimps_noy, imputation=j, data=addNewPatient,
                                          list.out=TRUE,pr=FALSE, check=TRUE),
                                  stringsAsFactos=FALSE)[nrow(addNewPatient),]

               imputed.datasets.noy[[j]][cc,names(tmp.imp.noy)] <-   tmp.imp.noy

               ############################
               #Test Set Does Not Include Y
               ############################

                 tmp.imp.y.testnoy <- as.data.frame(impute.transcan(Validateimps_y.testnoy,
  imputation=j,
  data=addNewPatient.noy,
  list.out=TRUE,pr=FALSE,
  check=TRUE),stringsAsFactors=FALSE)[nrow(addNewPatient.noy),]

        imputed.datasets.withy.testsetnoy[[j]][cc,names(tmp.imp.y.testnoy) ] <- tmp.imp.y.testnoy

        tmp.imp.noy.testnoy <- as.data.frame(impute.transcan(Validateimps_noy.testnoy,
  imputation=j,
  data=addNewPatient.noy,
  list.out=TRUE,pr=FALSE,
  check=TRUE), stringsAsFactos=FALSE)[nrow(addNewPatient.noy),]

               imputed.datasets.noy.testsetnoy[[j]][cc,names(tmp.imp.noy.testnoy)] <-   tmp.imp.noy.testnoy




                   }
        }


pred.TREAT.withy      <- lapply(imputed.datasets.withy, function(z) predict(TREAT, newdata=z))

pred.TREAT.noy        <- lapply(imputed.datasets.noy,function(z) predict(TREAT, newdata=z))

pred.TREAT.withy.testnoy      <- lapply(imputed.datasets.withy.testsetnoy,      function(z) predict(TREAT, newdata=z))

pred.TREAT.noy.testnoy        <- lapply(imputed.datasets.noy.testsetnoy,        function(z) predict(TREAT, newdata=z))


#Combine prediction estimates
pred.TREAT.withy.avg      <- expit(Reduce("+", pred.TREAT.withy) / length(pred.TREAT.withy))
pred.TREAT.noy.avg            <- expit(Reduce("+", pred.TREAT.noy) / length(pred.TREAT.noy))
pred.TREAT.withy.testnoy.avg      <- expit(Reduce("+", pred.TREAT.withy.testnoy) / length(pred.TREAT.withy.testnoy))
```

```
pred.TREAT.noy.testnoy.avg            <- expit(Reduce("+", pred.TREAT.noy.testnoy) / length(pred.TREAT.noy.testnoy))



 ##############################################
 # Metrics by Pattern
 ##############################################
 ex.fit <- "cancer ~ age + bmi + gender + rcs(pack_years,3) + ct_size + spicul + growthcat +
upperlobe + prev_cancer + fev1_pred + anysympt + petpos34"

  pattern <- which.pattern(testset, model=ex.fit)

 imputed.datasets.withy.pattern = data.frame(Reduce("+",lapply(imputed.datasets.withy, function(z)
   {get_metric(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer'],pattern))}))/length(imputed.datasets.withy),
   pattern = levels(pattern))


 imputed.datasets.noy.pattern = data.frame(Reduce("+",lapply(imputed.datasets.noy,function(z)
   {get_metric(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer'],pattern))}))/length(imputed.datasets.noy),
                                                                    pattern = levels(pattern))



imputed.datasets.withy.testsetnoy.pattern = data.frame(Reduce("+", lapply(imputed.datasets.withy.testsetnoy,function(z)
    {get_metric(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer'],
                           pattern))}))/length(imputed.datasets.withy.testsetnoy),
                  pattern = levels(pattern))



imputed.datasets.noy.testsetnoy.pattern = data.frame(Reduce("+",lapply(imputed.datasets.noy.testsetnoy,
                                                                    function(z)
{get_metric(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer'],pattern))}))/length(imputed.datasets.noy.testsetnoy),
                                                                    pattern = levels(pattern))




 results.MIy <- rbind(results.MIy, imputed.datasets.withy.pattern)
 results.MInoy <- rbind(results.MInoy, imputed.datasets.noy.pattern)
 results.MIy.testnoy <- rbind(results.MIy.testnoy, imputed.datasets.withy.testsetnoy.pattern)
 results.MInoy.testnoy <- rbind(results.MInoy.testnoy, imputed.datasets.noy.testsetnoy.pattern)


 results.MIy[results.MIy == 'NaN'] <- NA
results.MIy.avg <- do.call(rbind,lapply(split(results.MIy,results.MIy$pattern),function(z)
  data.frame(mean(z$auc,na.rm = TRUE),mean(z$brier),mean(z$log)) ))

results.MInoy[results.MInoy == 'NaN'] <- NA
results.MInoy.avg <- do.call(rbind,lapply(split(results.MInoy,results.MInoy$pattern),function(z)
  data.frame(mean(z$auc,na.rm = TRUE),mean(z$brier),mean(z$log)) ))

results.MIy.testnoy[results.MIy.testnoy == 'NaN'] <- NA
results.MIy.testnoy.avg <- do.call(rbind,lapply(split(results.MIy.testnoy,results.MIy.testnoy$pattern),function(z)
  data.frame(mean(z$auc,na.rm = TRUE),mean(z$brier),mean(z$log)) ))

results.MInoy.testnoy[results.MInoy.testnoy == 'NaN'] <- NA
results.MInoy.testnoy.avg <- do.call(rbind,lapply(split(results.MInoy.testnoy,results.MInoy.testnoy$pattern),function(z)
  data.frame(mean(z$auc,na.rm = TRUE),mean(z$brier),mean(z$log)) ))


imputed.datasets.withy.avg = colMeans(do.call(rbind,lapply(imputed.datasets.withy,function(z)
  {logistic_metrics(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer']))})))

 imputed.datasets.noy.avg = colMeans(do.call(rbind,lapply(imputed.datasets.noy,function(z)
   {logistic_metrics(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer']))})))

 imputed.datasets.withy.testsetnoy.avg = colMeans(do.call(rbind,lapply(imputed.datasets.withy.testsetnoy, function(z)
   {logistic_metrics(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer']))})))

 imputed.datasets.noy.testsetnoy.avg = colMeans(do.call(rbind,lapply(imputed.datasets.noy.testsetnoy,function(z)
   {logistic_metrics(data.frame(predict(TREAT, z, type = "fitted"),testset[,'cancer']))})))
```

```
 results[i,c('auc.MI.y', 'brier.MI.y', 'log.MI.y',
                         'auc.MI.noy',  'brier.MI.noy', 'log.MI.noy',
                         'auc.MI.y.testnoy','brier.MI.y.testnoy','log.MI.y.testnoy',
                         'auc.MI.noy.testnoy','brier.MI.noy.testnoy','log.MI.noy.testnoy')] <-
   c(imputed.datasets.withy.avg[c('auc','brier','log')],
     imputed.datasets.noy.avg[c('auc','brier','log')],
     imputed.datasets.withy.testsetnoy.avg[c('auc','brier','log')], imputed.datasets.noy.testsetnoy.avg [c('auc','brier','log')]  )



#progress.bar$step()

}


boot.res[boot.samp,] <-         colMeans(results)
boot.res.MIy[[boot.samp]] <-  results.MIy.avg
boot.res.MInoy[[boot.samp]] <-  results.MInoy.avg
boot.res.MIy.testnoy[[boot.samp]] <-  results.MIy.testnoy.avg
boot.res.MInoy.testnoy[[boot.samp]] <-  results.MInoy.testnoy.avg
}

all.results[[1]] <- boot.res
all.results[[2]] <- boot.res.MIy
all.results[[3]] <- boot.res.MInoy
all.results[[4]] <- boot.res.MIy.testnoy
all.results[[5]] <- boot.res.MInoy.testnoy
all.results

}

#save(boot.res, file = "/Users/SFM/Desktop/BootResults.Rda")
```

# CHAPTER 5

# CONCLUSION

This dissertation has examined several topics related to the application of prediction models in the presence of missing covariate estimation, bagged empirical null methods for large scale data, and best imputation practices for model construction and validation. Chapter 2 defined the problem of obtaining predictions from an existing clinical risk prediction model when covariates are missing, summarized existing imputation techniques are currently used to fill-in data, introduced the Pattern Mixture Kernel Submodel (PMKS), and illustrated the use of these techniques in a simulation and case study. Chapter 3 presents the Bagged Empirical Null (BEN) $p$-value, a new algorithm that combines existing methodology of bagging and empirical null techniques, and applied this procedure to pseudo-simulated data and a famous leukemia gene study. Chapter 4 discusses imputation of model covariates during model construction, validation, and prediction, and gives recommendations for when and when not to use the outcome in each of these developmental stages.

Chapter 2 highlights the common fallacy that imputation for an out-of-sample individual add information that make better predictions. Instead, imputation simply allows use of the fixed risk prediction model, and the best predictions for any missing data type (MCAR, MAR, MNAR) come from submodels fit within each missing data pattern (PMKS). While Chapter 2 provides a method to obtain instantaneous prediction for a new individual with missing data, its connection with the Multiple Imputation with Missingness Indicators Model (MIMI Model), reaches beyond the scope of risk prediction. Furthermore, the PMKS could implement any machine learning algorithm within a missing data pattern, and could extend beyond generalized linear models.

Chapter 3 introduces BEN $p$-values, which implements both bootstrap aggregation and empirical null techniques, producing $p$-values which significantly alter the ranking of results. We applied these ideas to a famous leukemia example, and uncovered findings that had previously been backed by published benchwork. Although empirical null methodology has roots in false discovery rate inference, applying these ideas to $p$-value calculations provides a more intuitive metric for selecting genes (in our example) for future research.

Chapter 4 gives recommendations for including the outcome in the imputation model, and suggests including the outcome for imputation of missing covariate values during model construction to obtain unbiased parameter estimates. When the outcome is used in the imputation algorithm during the validation step, we show through simulation, the model prediction metrics are artificially inflated. When used in the clinic on an individual with missing covariate values, the actual pragmatic model performance would be inferior to the validated results.

While the three papers presented here provide foundations for missing data and large scale inferential techniques, these ideas are applicable to a wide range of biomedical settings.

Allison, P. D. (2001), *Missing Data*, SAGE Publications.

Azur, M. J., Stuart, E. A., Frangakis, C. and Leaf, P. J. (2011), Multiple imputation by chained equations: what is it and how does it work?, *International Journal of Methods in Psychiatric Research* **20**(1), 40–49.

Barnard, J. and Meng, X.-L. (2016), Applications of multiple imputation in medical studies: from AIDS to NHANES, *Statistical methods in medical research* **8**(1), 17–36.

Bartlett, J. W., Carpenter, J. R., Tilling, K. and Vansteelandt, S. (2014), Improving upon the efficiency of complete case analysis when covariates are MNAR, *Biostatistics* **15**(4), 719–730.

Bartlett, J. W., Seaman, S. R., White, I. R. and Carpenter, J. R. (2012), Multiple imputation of covariates by fully conditional specification: accommodating the substantive model, *arXiv.org* (4), 462–487.

Benjamini, Y. and Hochberg, Y. (1995), Controlling the false discovery rate: a practical and powerful approach to multiple testing, *Journal of the Royal Statistical Society. Series B. Methodological* **57**(1), 289–300.

Blume, J. D. (2011), *Likelihood and its Evidential Framework*, Elsevier B.V.

Bø, T. and Jonassen, I. (2002), New feature subset selection procedures for classification of expression profiles | Genome Biology | Full Text, *Genome biology* **3**(4), research0017.1.

Breiman, L. (1996), Bagging predictors, *Machine learning* .

Clark, T. G. and Altman, D. G. (2003), Developing a prognostic model in the presence of missing data, *Journal of clinical epidemiology* **56**(1), 28–37.

Dardanoni, V., De Luca, G., Modica, S. and Peracchi, F. (2015), Model averaging estimation of generalized linear models with imputed covariates, *Journal of Econometrics* **184**(2), 452–463.

Dardanoni, V., Modica, S. and Peracchi, F. (2011), Regression with imputed covariates: A generalized missing-indicator approach, *Journal of Econometrics* **162**(2), 362–368.

Deppen, S. A., Blume, J. D., Aldrich, M. C., Fletcher, S. A., Massion, P. P., Walker, R. C., Chen, H. C., Speroff, T., Degesys, C. A., Pinkerman, R., Lambright, E. S., Nesbitt,

J. C., Putnam, J. B. and Grogan, E. L. (2014), Predicting Lung Cancer Prior to Surgical Resection in Patients with Lung Nodules, *Journal of Thoracic Oncology* **9**(10), 1477–1484.

Dunn, O. J. (1959), Estimation of the Medians for Dependent Variables, *The Annals of Mathematical Statistics* **30**(1), 192–197.

Efron, B. (2012*a*), *Large-Scale Inference*, Empirical Bayes Methods for Estimation, Testing, and Prediction, Cambridge University Press.

Efron, B. (2012*b*), 'Model Selection Estimation and Bootstrap Smoothing'.

Geisser, S. (1993), *Predictive Inference*, CRC Press.

Golub, T. R., Slonim, D. K., Tamayo, P. and Huard, C. (1999), Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, ... .

Greenland, S. and Finkle, W. D. (1995), A Critical Look at Methods for Handling Missing Covariates in Epidemiologic Regression Analyses, *American Journal of Epidemiology* **142**(12), 1255–1264.

Groenwold, R., White, I. R. and Donders, A. (2012), Missing covariate data in clinical research: when and when not to use the missing-indicator method for analysis, *Canadian Medical ...* .

Harrell, F. E. (2013), *Regression Modeling Strategies*, With Applications to Linear Models, Logistic Regression, and Survival Analysis, Springer Science & Business Media.

Hastie, T., Tibshirani, R. and Friedman, J. (2009), *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York, New York, NY.

Heitjan, D. F. and Little, R. J. A. (1991), Multiple Imputation for the Fatal Accident Reporting System, *Applied Statistics* **40**(1), 13.

Horton, N. J. and Lipsitz, S. R. (2012), Multiple Imputation in Practice, *The American Statistician* .

Ibrahim, J. G. (2012), Incomplete Data in Generalized Linear Models, *Journal of the American Statistical Association* **85**(411), 765–769.

Jaffe, A. E., Storey, J. D., Ji, H. and Leek, J. T. (2013), Gene set bagging for estimating the probability a statistically significant result will replicate, *BMC Bioinformatics* **14**(1), 360.

Janssen, K. J. M., Donders, A. R. T., Harrell Jr., F. E., Vergouwe, Y., Chen, Q., Grobbee, D. E. and Moons, K. G. M. (2010), Missing covariate data in medical research: to impute is better than to ignore., *Journal of clinical epidemiology* **63**(7), 721–727.

Janssen, K. J. M., Vergouwe, Y., Donders, A. R. T., Harrell, F. E., Chen, Q., Grobbee, D. E. and Moons, K. G. M. (2009), Dealing with missing predictor values when applying clinical prediction models., *Clinical Chemistry* **55**(5), 994–1001.

Jewson, S. (2004), The problem with the Brier score, *arXiv.org* .

Jones, M. P. (1996), Indicator and stratification methods for missing explanatory variables in multiple linear regression, *Journal of the American Statistical Association* .

Knol, M. J., Janssen, K. J., Donders, A. R. T., Egberts, A. C., Heerdink, E. R., Grobbee, D. E., Moons, K. G. and Geerlings, M. I. (2010), Unpredictable bias when using the missing indicator method or complete case analysis for missing confounder values: an empirical example, *Journal of clinical epidemiology* **63**(7), 728–736.

Kohavi, R. (1995), Vanderbilt Library, *Ijcai* .

Lee, K. E., Sha, N., Dougherty, E. R. and Vannucci, M. (2003), Gene selection: a Bayesian variable selection approach, ... .

Little, R. J. A. (1992), Regression With Missing X's: A Review, *Journal of the American Statistical Association* **87**(420), 1227–1237.

Little, R. J. A. (1993), Pattern-Mixture Models for Multivariate Incomplete Data, *Journal of the American Statistical Association* **88**(421), 125.

Little, R. J. A. and Rubin, D. B. (2014), *Statistical Analysis with Missing Data*, John Wiley & Sons.

Little, R. J. and Wang, Y. (1996), Pattern-mixture models for multivariate incomplete data with covariates., *Biometrics* **52**(1), 98–111.

Mark E Glickman, P. D., Sowmya R Rao, P. D. and Mark R Schultz, P. D. (2015), Response to: "False discovery rate control is not always a replacement for Bonferroni-type adjustment", *Journal of clinical epidemiology* 1–5.

Meyer, K. and Windeler, J. (2009), A New Suggestion for the Classification of Missing Values in the Outcome of Clinical Trials, *Clinical Research and Regulatory Affairs* .

Miller, R. G. J. (2012), *Simultaneous Statistical Inference*, Springer Science & Business Media.

Molenberghs, G., Beunckens, C., Sotto, C. and Kenward, M. G. (2008), Every missingness not at random model has a missingness at random counterpart with equal fit, *Journal of the Royal Statistical Society. Series B. Statistical Methodology* **70**(2), 371–388.

Moons, K. G. M., Donders, R. A. R. T., Stijnen, T. and Harrell, F. E. (2006), Using the outcome for imputation of missing predictor values was preferred., *Journal of clinical epidemiology* **59**(10), 1092–1101.

Natsuka, S., Akira, S., Nishio, Y., Hashimoto, S. and Sugita, T. (1992), Macrophage differentiation-specific expression of NF-IL6, a transcription factor for interleukin-6, *Blood* .

Pe'er, I., Yelensky, R., Altshuler, D. and Daly, M. J. (2008), Estimation of the multiple testing burden for genomewide association studies of nearly all common variants, *Genetic epidemiology* **32**(4), 381–385.

Perneger, T. V. (1998), What's wrong with Bonferroni adjustments - ProQuest, *British Medical Journal* .

Phillips, R. S., Connors Jr, A. P. and Dawson, N. V. (1995), The SUPPORT Prognostic Model, *Ann Intern Med* .

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

Rubin, D. B. (1996), Multiple Imputation After 18+ Years, *Journal of the American Statistical Association* **91**(434), 473–489.

Rubin, D. B. (2009), *Multiple Imputation for Nonresponse in Surveys*, Wiley Series in Probability and Statistics, John Wiley & Sons, Hoboken, NJ, USA.

Rubin, D. B. and Schenker, N. (1991), Multiple imputation in health-are databases: An overview and some applications, *Statistics in Medicine* **10**(4), 585–598.

Schafer, J. L. (1997), *Analysis of Incomplete Multivariate Data*, CRC Press.

Schafer, J. L. (1999), Multiple imputation: a primer., *Statistical methods in medical research* **8**(1), 3–15.

Schafer, J. L. and Graham, J. W. (2002), Missing data: our view of the state of the art., *Psychological methods* **7**(2), 147–177.

Schenker, N. and Taylor, J. M. G. (1996), Partially parametric techniques for multiple imputation, *Computational Statistics and Data Analysis* **22**(4), 425–446.

Shaffer, J. P. (1995), Multiple hypothesis testing, *Annual review of psychology* .

Sham, P. C. and Purcell, S. M. (2014), Statistical power and significance testing in large-scale genetic studies, *Nature Reviews Genetics* **15**(5), 335–346.

Shmueli, G. (2010), To Explain or to Predict?, *Statistical Science* **25**(3), 289–310.

Storey, J. D. (2002), A direct approach to false discovery rates, *Journal of the Royal Statistical Society. Series B. Statistical Methodology* **64**(3), 479–498.

Tong, D. L. and Schierz, A. C. (2011), Hybrid genetic algorithm-neural network: Feature extraction for unpreprocessed microarray data, *Artificial intelligence in medicine* **53**(1), 47–56.

Vach, W. (2012), *Logistic Regression with Missing Values in the Covariates*, Springer Science & Business Media.

Vach, W. and Blettner, M. (1991), Biased Estimation of the Odds Ratio in Case-Control Studies due to the Use of Ad Hoc Methods of Correcting for Missing Values for Confounding Variables, *American Journal of Epidemiology* **134**(8), 895–907.

van Buuren, S. (2012), *Flexible Imputation of Missing Data*, Vol. 20125245 of *Chapman & Hall/CRC Interdisciplinary Statistics Series*, CRC Press.

Vergouwe, Y., Royston, P., Moons, K. G. M. and Altman, D. G. (2010), Development and validation of a prediction model with missing predictor data: a practical approach., *Journal of clinical epidemiology* **63**(2), 205–214.

White, I. R. and Carlin, J. B. (2010), Bias and efficiency of multiple imputation compared with complete-case analysis for missing covariate values., *Statistics in Medicine* **29**(28), 2920–2931.

Wood, A. M., Royston, P. and White, I. R. (2015), The estimation and use of predictions for the assessment of model performance using large samples with multiply imputed data, *Biometrical Journal* **57**(4), 614–632.

Wright, S. P. (1992), Adjusted P-Values for Simultaneous Inference, *Biometrics* **48**(4), 1005.

Zhou, X., Liu, K.-Y. and Wong, S. T. C. (2004), Cancer classification and prediction using logistic regression with Bayesian gene selection, *Journal of Biomedical Informatics* **37**(4), 249–259.