CLUSTERING OF BULK RNA-SEQ DATA AND MISSING DATA METHODS IN DEEP LEARNING

David K. Lim

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Biostatistics in the Gillings School of Global Public Health.

Chapel Hill
2022

Approved by:

Naim U. Rashid

Joseph G. Ibrahim

Michael I. Love

Junier B. Oliva

Wei Sun

Di Wu

ABSTRACT

David K. Lim : Clustering of Bulk RNA-Seq Data and Missing Data Methods in Deep Learning
(Under the direction of Naim U. Rashid and Joseph G. Ibrahim)

Clustering is a form of unsupervised learning that aims to uncover latent groups within data based on similarity across a set of features. A common application of this in biomedical research is in delineating novel cancer subtypes from patient gene expression data, given a set of informative genes. However, it is typically unknown a priori what genes may be informative in discriminating between clusters, and what the optimal number of clusters is. In addition, few methods exist for unsupervised clustering of bulk RNA-seq samples, and no method exists that can do so while simultaneously adjusting for between-sample global normalization factors, accounting for potential confounding variables, and selecting cluster-discriminatory genes.

In Chapter 2, we present FSCseq (**F**eature **S**election and **C**lustering of RNA-**seq**): a model-based clustering algorithm that utilizes a finite mixture of regression (FMR) model and employs a quadratic penalty method with a SCAD penalty. The maximization is done by a penalized EM algorithm, allowing us to include normalization factors and confounders in our modeling framework. Given the fitted model, our framework allows for subtype prediction in new patients via posterior probabilities of cluster membership.

The field of deep learning has also boomed in popularity in recent years, fueled initially by its performance in the classification and manipulation of image data, and, more recently, in areas of public health, medicine, and biology. However, the presence of missing data in these latter areas is very common, and involves more complicated mechanisms of missingness than the former. While a rich statistical literature exists regarding the characterization and treatment of missing data in traditional statistical models, it is unclear how such methods may extend to deep learning methods.

In Chapter 3, we present NIMIWAE (**N**on-**I**gnorably **M**issing **I**mportance **W**eighted **A**uto**E**ncoder), an unsupervised learning algorithm which provides a formal treatment of missing data in the context of Importance Weighted Autoencoders (IWAEs), an unsupervised Bayesian deep learning architecture, in order to perform single and multiple imputation of missing data. We review existing methods that handle up to the missing at random (MAR) missingness, and propose methods to handle the more difficult missing not at random (MNAR) scenario. We show that this extension is critical to ensure the performance of data imputation, as well as downstream coefficient estimation. We utilize simulation examples to illustrate the impact of missingness on such tasks, and compare the performance of several proposed methods in handling missing data. We applied our proposed methods to a large electronic healthcare record dataset, and illustrated its utility through a qualitative look at the downstream fitted models after imputation.

Finally, in Chapter 4, we present dlglm (**d**eeply-**l**earned **g**eneralized **l**inear **m**odel), a supervised learning algorithm that extends the missing data methods from Chapter 3 directly to supervised learning tasks such as classification and regression. We show that dlglm can be trained in the presence of missing data in both the predictors and the response, and under the MCAR, MAR, and MNAR missing data settings. We also demonstrate that the trained dlglm model can directly predict response on partially-observed samples in the prediction or test set, drawing from the learned variational posterior distribution of the missing values conditional on the observed values during model training. We utilize statistical simulation and real-world datasets to show the impact of our method in increasing accuracy of coefficient estimation and prediction under different mechanisms of missingness.

This thesis, as with all that I have done and will ever do, is dedicated to Jesus Christ, my Lord and Savior. He has dealt bountifully with me.

To Him be the glory.

ACKNOWLEDGEMENTS

I would like to first and foremost acknowledge my lovely wife, Grace, for sticking by my side through the most difficult times of life, and being my constant source of help and support. I would also like to acknowledge my loving father, mother, and brother. They have always loved and encouraged me, and they have inspired me to challenge myself and pursue a higher education.

Next, I would like to acknowledge my co-advisors Naim Rashid and Joseph Ibrahim. They have been the most patient, thoughtful, helpful, and supportive mentors I could have ever asked for. Thank you for helping to develop me into a better student, researcher, and overall person.

Finally, I would like to acknowledge my dear friends who have shared life with me these past several years. Thank you for your many words of encouragement and prayer, for the coffee breaks and food, sports, and everything else that brought me joy throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AE	Autoencoder
ARI	Adjusted Rand Index
BIC	Bayesian Information Criterion
CDA	Coordinate-wise Descent Algorithm
DL	Deep Learning
EHR	Electronic Health Records
ELBO	Evidence Lower Bound
EM	Expectation Maximization
FFNN	Feed-Forward Neural Network
FPR	False Positive Rate
GLM	Generalized Linear Model
IRLS	Iteratively Reweighted Least Squares
IWAE	Importance-Weighted Autoencoder
LFC	Logarithmic (base 2) Fold Change
MAR	Missing at Random
MCAR	Missing Completely at Random
MCMC	Markov Chain Monte Carlo
MNAR	Missing Not at Random
PB	Percent Bias
RNA-seq	RNA sequencing
SGD	Stochastic Gradient Descent
TPR	True Positive Rate
VAE	Variational Autoencoder

CHAPTER 1: LITERATURE REVIEW

1.1 Clustering of RNA-Seq Gene Expression Samples

RNA-sequencing (RNA-seq) is the direct sequencing of transcripts using high-throughput technologies (Zhao et al., 2014), and is a common platform for measuring gene expression. Gene expression is extremely important in determining the unique biology of organisms, playing vital roles in almost every biological process (Alberts et al., 2002; Tomancak et al., 2007; Wittkopp, 2007; O'Connor et al., 2010; Romero et al., 2012; Zhang et al., 2014; Chappell et al., 2015). Importantly, RNA-seq gives a global view of the transcriptome and its organization for different species and cell types (Wang et al., 2009), and RNA-seq has been shown to boast prominent advantages over the previously popular DNA microarray technologies while still being able to replicate microarray-based results (Fu et al., 2009; McGettigan, 2013; Trost et al., 2015; Wolff et al., 2018; Rao et al., 2019).

In the context of cancer, RNA-seq has been studied extensively. Applications include discovering mutations and how they manifest in targeted genes, and detecting gene fusion that results from genome rearrangement in cancer (Mardis and Wilson, 2009; Robison, 2010; Li et al., 2011; McPherson et al., 2011). Additionally, differential expression analysis can be done using RNA-seq (Robinson et al., 2009; Ramsköld et al., 2011; Love et al., 2014), which has countless applications in the study of cancer, like analyzing the response of lung cancer to external stimuli like smoking (Beane et al., 2011), detecting gene-level differences for breast cancers exhibiting tamoxifen resistance (Huber-Keener et al., 2012), identification of biomarkers for specific tumors (Seyednasrollah et al., 2015; Cui et al., 2015; Liang et al., 2015), and more (Young et al., 2010; McCarthy et al., 2012; Jardim-Perassi et al., 2019).

A specific application of RNA-seq in cancer research is in clustering tumors based upon gene expression. Clustering is a form of unsupervised learning that aims to uncover latent groups across subjects based on their similarity with respect to a common set of features. A common application of clustering is in the discovery of novel molecular subtypes in cancer based upon patient tumor gene expression data. Resulting clusters based on gene expression have been shown to delineate samples of different tumor subtypes, which exhibit different prognoses, clinical outcomes, and even treatment responses (van de Vijver et al., 2002; Sorlie et al., 2001; Perou et al., 2000; Brannon et al., 2010; Chia et al., 2012; Mao et al., 2017). The discovery of such subtypes therefore enable precision medicine approaches for treatment selection, where, given a patient's subtype information, an optimal treatment may be recommended to maximize patient clinical outcomes based upon prior data. More generally, clustering of gene expression data has been proven to be useful for understanding cellular processes and gene regulation (Yeung et al., 2001b), and has been used to understand cellular processes in cancer (Liu et al., 2014; Best et al., 2015; Cima et al., 2016; Tirosh et al., 2016; Chung et al., 2017; Murakami et al., 2018) as well as to comprehensively profile many cancer types (McLendon et al., 2008; Muzny et al., 2012; Koboldt et al., 2012; Hammerman et al., 2012; Levine et al., 2013; Creighton et al., 2013; Weinstein et al., 2014; Collisson et al., 2014; Bass et al., 2014; Lawrence et al., 2015; Brat et al., 2015; Ciriello et al., 2015; Linehan et al., 2016; Zheng et al., 2016; Fishbein et al., 2017; Robertson et al., 2017; Ricketts et al., 2018).

However, analyses of RNA-seq datasets can be confounded by technical factors, such as differences in sequencing depth (Zyprych-Walczak et al., 2015) or batch effects (Hicks et al., 2017). Between-sample variation due to differences in sequencing depth can cause global shifts in sample-specific read count distributions, necessitating the use of between-sample correction methods (Li et al., 2015). Batch effects have been known to be very strong confounders of gene expression analyses (Li et al., 2009b), requiring some method of correction to account for these batch effects (Leek, 2014; Peixoto et al., 2015)

Gene expression via RNA-seq is quantified by aligning known genes to millions of short strings of input RNAs called 'reads' (Finotello and Camillo, 2014), and quantification is done using some software (Li et al., 2009a; Anders et al., 2014; Patro et al., 2017) to enumerate the number of reads that map to specific protein-coding regions of the genome. In particular, software like `RSEM` and `Salmon` can handle multimapped reads, or sequences that map to multiple regions of the genome, by estimating the uncertainty of transcript abundance with conditional probabilities of a read being multimapped (Li et al., 2011; Patro et al., 2017). The resulting measures from this quantification process often results in overdispersed counts, in contrast to the Gaussian distributional assumptions made by earlier methods designed for clustering subjects with microarray-based gene expression measurements.

This shift in distributional assumption also has practical implications. For example, due to the count nature of RNA-seq data, the correction factors for differences in sequencing depth are often directly incorporated into statistical modeling procedures, for example, as offset terms in negative binomial regression-based differential expression methods (Robinson et al., 2009; Love et al., 2014). Correction for other confounders, such as batch effects (Leek et al., 2010) or specific clinical variables, is done by including the relevant factors as regression covariates. However, to the best of our knowledge no analogous approach exists to correct for such factors in RNA-seq read count-based clustering analysis. Therefore, such an approach that can correct for such factors may have significant utility in improving clustering performance.

Currently, few methods exist that can cluster samples whose genes (features) are measured via RNA-seq. Also, no methods currently exist that are able to allow for direct prediction of subtype for new samples. A method that allows for such prediction may be convenient, as one may utilize an already-trained model to predict the subtype of new samples, rather than having to retrain the entire model. Additionally, few methods exist that are able to select for features that discriminate in expression across subtypes while clustering. Selecting such features may aid in the identification of driver genes that delineate the subtypes of a particular cancer type. It can also help lead to the development of gene lists such as the PAM50 genes (Nielsen et al., 2010), which are used to guide treatment decisions based upon subtype of cancer. We go over these existing methods in detail in Section 1.2

1.2 Existing Clustering Methods

Current methods that seek to cluster RNA-seq data can be divided into three general classes: transformation based, count based, and nonparameteric clustering methods.

Transformation-based methods

Previous studies on microarray gene expression has assumed the data to be normally distributed (Lee et al., 2000; Allison et al., 2002; Chu et al., 2005). As a result, many parametric clustering algorithms designed for application to microarray-based gene expression profiles are based upon the Gaussian mixture model (Pan et al., 2003; Ouyang et al., 2004; Qu and Xu, 2004; McNicholas and Murphy, 2010).

The `mclust` package is an example of one such algorithm (Fraley and Raftery, 1998; Yeung et al., 2001a; Scrucca et al., 2016). This algorithm uses a finite mixture model such that the conditional distribution pertaining to each of the K clusters in the model is assumed to follow a multivariate Gaussian distribution. Maximum likelihood estimation is performed using the EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008). The covariance matrix $\mathbf{\Sigma}_k$ for each cluster k, $k = 1, \ldots, K$, is specified by utilizing an eigenvalue decomposition such that $\mathbf{\Sigma}_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T$, where $\mathbf{A}_k$ is a diagonal matrix that specifies the shape of the density contours and $\mathbf{D}_k$ is an orthogonal matrix that determines the orientation of the corresponding ellipsoid (Banfield and Raftery, 1993; Celeux and Govaert, 1995), and the number of estimated parameters in the covariance is constrained by various specifications of $\mathbf{A}_k$ and $\mathbf{D}_k$ (Browne and McNicholas, 2013; Scrucca et al., 2016), performing model selection using the Bayesian Information Criterion (Schwarz, 1978), or "BIC".

It is clear that discrete count data pertaining to RNA-seq read counts does not fit the typical Gaussian mixture model assumption. It has been seen that non-normal count data can be made more approximately Gaussian through the use of a transformation on the data that stabilizes the variance (Rohlf and Sokal, 1981; Zar, 1999). One such transformation that is very simple and intuitive is the $\log$ transform. The variance-stabilizing transform is a monotone mapping such that the variance of transformed values is approximately independent of the mean, and it has been shown to have a straightforward application in clustering of RNA-seq read counts (Anders and Huber, 2010). Alternatively, the regularized logarithm (rlog) transformation is performed by fitting an intercept-only GLM on each gene, shrinking the log fold changes (LFCs) with respect to the baseline using an empirical Bayes method (Love et al., 2014). The rlog transform additionally directly accounts for variation in sequencing depth, although it may not preserve the ordering of genes within a sample under large amounts of shrinkage (Love et al., 2014).

Once an appropriate transformation is applied, the application of methods developed for microarray data (Zwiener et al., 2014) is feasible by assuming the transformed data are distributed approximately Gaussian. However, prior work has shown that the choice of transformation can greatly influence clustering performance, as the quality of this Gaussian approximation varies by condition, and the optimal choice may not be readily apparent (Noel-MacDonnell et al., 2018). Additionally, studies have shown that such transformations may not work very well for count data (O'Hara and Kotze, 2010), especially for overdispersed counts (Solomon and Sawilowsky, 2009; Zwiener et al., 2014). From a practical standpoint, methods that directly model the data rather than relying on transformations may also be more intuitive for inference and interpretation. Also, these transformation-based methods are not able to directly adjust for technical variation like differences in sequencing depth, and for confounders like batch. Additionally, in the RNA-seq context, they do not perform automatic feature selection nor predict cluster membership for a new subject based on a fitted model.

Count-based methods

The second class of methods directly models gene-level read counts, assuming a specific distribution on the read counts. There have been developed a number of RNA-seq clustering methods that directly model gene level read counts (Wang et al., 2013; Si et al., 2013; Silva et al., 2019), but these methods cluster genes rather than samples.

`iCluster+` (Mo et al., 2013) is an integrative method that can perform clustering analysis using multiple datasets of different distributional assumptions. Estimation of parameters is done using a modified Monte Carlo Newton-Raphson algorithm (Friedman et al., 2010; McCulloch, 1997). `iCluster+` writes the form of a joint log-likelihood that incorporates the information across each genomic dataset, incorporating the L1 penalty for regularization (Tibshirani, 1996) to induce sparsity. Also, a set of multivariate standard normal latent variables is utilized to denote underlying driving factors, and samples are drawn from the joint posterior distribution of these latent variables using the random-walk Metropolis Hasting algorithm (Tanner and Wong, 2010; Casella and Robert, 2010; Liu, 2008). Using K-means on the latent variables, samples are clustered together, using a guided procedure based upon the BIC and deviance ratio for selecting the optimal number of clusters and the optimal value for the penalty parameter (Mo and Shen, 2019). Although `iCluster+` performs feature selection and directly models counts, `iCluster+` also assumes a Poisson distribution on count data, thus ignoring potential overdispersion in counts. Additionally, the `iCluster+` framework does not allow for adjustment for sequencing depth or covariates while clustering, and does not perform prediction on new samples. Moreover, although `iCluster+` provides a mode for automatic feature selection, the manual (Mo and Shen, 2019) suggests that gene selection must be done using arbitrary cutoffs based on quantiles of estimated parameters.

`NBMB` (Li et al., 2018) is a recently developed clustering method that models RNA-seq read counts using the negative-binomial distribution. Specifically, `NBMB` assumes a finite mixture model on the read counts, with a mixture of K negative binomial distributions, corresponding to the K clusters. A stochastic form of the EM algorithm with deterministic annealing (Rose, 1998) is utilized, similar to a form proposed by Si et al. (2013). The `NBMB` algorithm is initialized with equal mixing proportions and MLE estimates for the mean and dispersion, and the algorithm iteratively estimates posterior probabilities of cluster membership (E-step), and parameter updates (M-step). Also, separate mean and dispersion parameters are estimated for each cluster of each gene, taking into account potentially different levels of overdispersion of gene read counts across samples in each cluster. Although `NBMB` can account for overdispersed RNA-seq counts, `NBMB` cannot perform feature selection like `iCluster+`. In addition, like `iCluster+`, `NBMB` is not able to adjust for factors such as batch effects or differences in sequencing depth, and also does not perform prediction on new samples.

Nonparametric methods

The third class of methods includes non-parameteric clustering approaches such as the average-linkage hierarchical clustering (HC) and K-medoids (KM) (Jaskowiak et al., 2018). These algorithms have been used heavily in practice due to their speed and simplicity, as well as ease of implementation. These clustering methods involve using some dissimilarity measure to guide clustering, grouping together samples that show low dissimilarity, and separating samples that show high dissimilarity.

Hierarchical agglomerative clustering (Johnson, 1967; Zhao et al., 2005; Manning et al., 2008) works in a bottom-up approach, such that pairs of clusters are merged together iteratively based upon dissimilarity. Typically, a linkage is specified for the particular hierarchical clustering algorithm. Single-linkage clustering takes the dissimilarity between two clusters to be the distance between the most similar members, average-linkage clustering takes the dissimilarity to be the average distance between all members, and complete-linkage clustering takes the dissimilarity to be the distance between the most dissimilar members (Patel et al., 2015). Often, the choice of dissimilarity measures and linkages is arbitrary, with the clustering performance based upon each choice depending heavily on the dataset in question (Jaskowiak et al., 2014). Out of these methods, Jaskowiak et al. (2018) suggests that the average-linkage hierarchical clustering algorithm provides the best performance in clustering RNA-seq.

K-medoids (Bishop, 2006) is another nonparametric clustering algorithm, and works by partitioning samples in a manner that is similar to the well-known K-means algorithm (Wu et al., 2008). In the K-medoids algorithm, "medoids" are selected as objects of a cluster whose average dissimilarity to all objects in the cluster is minimal, and partitioning is done by minimizing the average L1 distance between the each point of a cluster and its corresponding medoid (Cao and Yang, 2010). In the Partitioning Around Medoids (PAM) algorithm (Han and Pei, 2017), the medoids are iteratively re-assigned to improve clustering results, and each point is assigned to clusters based on the minimum dissimilarity with the corresponding medoids. Jaskowiak et al. (2018) showed that K-medoids, like average-linkage hierarchical clustering, performs well in recovering clusters in RNA-seq compared to other nonparametric methods.

Although distribution-free, these nonparameteric methods are very dependent on the choices of model specifications, like the measure of dissimilarity, and the type of linkage in hierarchical clustering. In practice, there are numerous variants to these nonparametric methods, and their performance is highly dependent on the context in which they are employed (Zhao et al., 2005; Müllner, 2011; Mingoti and Lima, 2006; Nazari et al., 2015; Schubert and Rousseeuw, 2018). Also, none of these algorithms described can perform feature selection to determine cluster-discriminatory genes, nor can they adjust for batch or differences in sequencing depth. They also cannot perform prediction of subtype in new samples, and can only cluster the initial training samples. These limit the applicability of such methods to translational settings such as cancer subtyping based upon tumor RNA-seq data.

1.3 Deep Learning

Neural networks are networks comprised of “layers” of so-called “neurons” that are connected such that some transformation of the values from the previous layer is applied to output values for the subsequent layer (Taylor, 1996; Lecun et al., 1998). By combining many layers of neurons, a neural network can perform a series of transformations such that the input data is mapped to a highly nonlinear and complicated transformation of the input (Schmidhuber, 2015). The structure of the connections between neurons are remarkably simple, allowing for intricate architectures that provide very powerful tools for learning even very complicated tasks involving massive amounts of input data (Najafabadi et al., 2015).

Deep learning refers to a broad class of algorithms that utilizes neural networks to learn complex relationships within data. These deep learning methods initially gained mass interest through their excellent performance in classifying images (Krizhevsky et al., 2012; Szegedy et al., 2014; He et al., 2015). Such methods took in millions of labeled images, and performed the supervised task of classifying new images into pre-trained categories. Seeing the promising performance of such algorithms, researchers quickly began to generalize deep learning methods to be used in various analyses in other fields, such as recommender systems (Zhang et al., 2017), AI gaming (Silver et al., 2017), video editing (Podlesnyy, 2019), and many more (Vankayala and Rao, 1993; Pumsirirat and Yan, 2018; Cavalcante et al., 2016; Basak et al., 2018; Suk et al., 2013; Voulodimos et al., 2018; Almalaq and Edwards, 2017; Rawat and Wang, 2017; Ozdag, 2018; Carrio et al., 2017; Xie et al., 2018; Zhang et al., 2018).

In particular, there has been significant interest in the application of deep learning methods to problems in the biomedical sciences, for example to detect tumors from X-ray or MRI scans (Razzak et al., 2017; Ausawalaithong et al., 2018; Shen and Gao, 2019). Similarly, the field of genomics has seen numerous advances in deep learning methods for analyses of single-cell RNA-seq datasets (Way and Greene, 2018; Lopez et al., 2018; Lotfollahi et al., 2019). In the realm of health informatics, with applications such as cancer diagnostics, tissue classification, anomaly detection, human activity recognition, prediction of disease, and air pollutant prediction (Ravi et al., 2017; Fakoor et al., 2013; Brosch et al., 2013; Sun et al., 2014; Nie et al., 2015; Zou et al., 2016).

We introduce some common deep learning architectures for supervised and unsupervised deep learning tasks in the next section.

1.3.1 Unsupervised Deep Learning Architectures

An autoencoder (AE) is one particular deep learning architecture that was first developed for and used in dimension reduction (Goodfellow et al., 2016). The structure of an AE involves an encoder, which "encodes" the data into a lower-dimensional space that captures the salient features of the data, and a decoder, which reconstructs the original data from the lower-dimensional space (Tschannen et al., 2018). In this way, an autoencoder simultaneously learns a lower-dimensional representation of the input data, which may potentially be very large, and learns to reconstruct the input data from the learned representation. Such a dimensionality reduction technique can be useful in clustering (Song et al., 2013) or for visualization of data (Wang et al., 2016), and reconstruction of the data may be useful in de-noising or imputation of data (Vincent et al., 2008).

The variational autoencoder (VAE) is built upon a similar encoder-decoder structure and resembles the AE, but the VAE additionally imposes a probabilistic assumption on the lower-dimensional (or "latent") space (Doersch, 2016). This assumption essentially casts the autoencoder in a Bayesian framework, where the goal is to compute the posterior of the latent variable conditioned on the observed data, given a specified prior distribution. Traditionally in this framework, the most widely-used methods for approximate posterior inference have been Markov Chain Monte Carlo (MCMC) sampling methods (Gelfand and Smith, 1990), such as the Metropolis-Hastings (Metropolis et al., 1953; Hastings, 1970) and Gibbs sampling (Geman and Geman, 1984) algorithms; however, these methods may be too slow for larger and more complex datasets (Blei et al., 2016), which may be the case for many deep learning applications.

To address this, the VAE instead utilizes variational inference, where a family of distributions is posited to approximate the posterior, and the optimal approximation within this family is learned by minimizing the Kullback-Leibler divergence between the true and approximate posteriors (Kingma and Welling, 2013; Rezende et al., 2014; Blei et al., 2016). In particular, the encoder of a VAE learns the approximate posterior of this latent variable given the input data using amortized variational inference (Gershman and Goodman, 2014), and the decoder of a VAE learns the generative distribution of the data conditioned on the latent space. The VAE is typically trained by utilizing some computationally efficient and straightforward stochastic optimization technique (Rezende et al., 2014; Kingma and Ba, 2014).

The presence of complex interactions among high-dimensional features in modern biomedical data has motivated the use of VAEs, commonly for unsupervised learning tasks such as dimension reduction, representational learning, and generation of synthetic data mimicking real input data, which may be unavailable, especially in the healthcare setting, due to patient confidentiality (Shickel et al., 2018). In prior evaluations, VAEs have shown tremendous performance in data generation and representation learning (Kingma and Welling, 2019).

1.3.2 Supervised Deep Learning Architectures

While a number of deep learning architectures have been proposed for supervised learning, the feed forward neural network (FFNN) is very commonly used in most architectures. In a FFNN, sequential non-linear transformations are applied to the values of the input layer. Each value in the subsequent layer of the FFNN is computed by applying a non-linear (or "activation") function to the linear transformation of the values in the previous layer, outputting a complex non-linear transformation of the input (Svozil et al., 1997). Such networks may contain a large number of parameters and take a long time to train, so optimization is often done via stochastic gradient descent, for scalability (Guo and Gelfand, 1990).

Such FFNNs have been utilized for tasks like regression (Tran et al., 2019) and classification (Krizhevsky et al., 2012; Russakovsky et al., 2015), and are highly utilized for their ability to capture complex nonlinearity between the features, as well as model nonlinear relationships between the input and the output. Additionally, prediction models using FFNNs have generally shown very excellent performance (Szegedy et al., 2014; He et al., 2015).

1.4 Missing Data in Deep Learning

Historically, statistical literature characterizes missingness into one of three main categories: missing completely at random (MCAR), missing at random (MAR), or missing not at random (MNAR) (Little and Rubin, 2002). MCAR refers to the case where the probability of missingness is completely independent of the observed and missing values, MAR refers to the case where the probability of missingness is dependent on the observed values but independent of the missing values, and MNAR refers to the case where the probability of missingness is dependent on the unobserved values. There is a wide array of methods on how to treat each type of missingness in traditional statistical frameworks (Ibrahim et al., 2005; Ibrahim and Molenberghs, 2009). However, it is not clear how these methods translate into deep learning. Specifically, missing data methods involving computationally expensive sampling steps may be intractable for adaptation into deep learning, which typically utilizes very large volumes of data (Chen et al., 2019).

One particular application of existing deep learning methods, like AEs or VAEs, is in analyzing medical data, such as the Electronic Health Records (EHR). Missingness of data in this domain is very common, yielding over 80% missingness of features in some datasets (Luo et al., 2018). In addition, the missingness in these kinds of data is typically informative, or non-ignorable (Beaulieu-Jones and and, 2016; Venugopalan et al., 2019; Sharafoddini et al., 2019), causing methods that don't explicitly account for this type of missingness to yield biased estimates (Yuan and Yin, 2009).

In practice, there is a vast array of methods on handling and imputing missing values in datasets (Horton and Kleinman, 2007), with some methods utilizing state-of-the-art machine learning techniques (Stekhoven and Buhlmann, 2011). In deep learning, some AE methods have been used to attempt to handle missingness in incomplete medical datasets (Beaulieu-Jones and and, 2016; Miotto et al., 2016). However, it has been shown that VAEs may outperform AEs in learning representations on larger datasets, like EHR data (Sadati et al., 2019). Consequently, a number of methods have been developed recently for handling missing data in the context of VAEs (McCoy et al., 2018; Nazabal et al., 2018; Ivanov et al., 2019; Mattei and Frellsen, 2019). In Section 1.5, we review some of these existing methods of dealing with missing data, with emphasis on the VAE methods.

Additionally, there have been some recent attempts to train supervised deep learning models for regression and classification tasks in the presence of missing features (Ipsen et al., 2021), however such methods typically assume either MCAR or MAR missingness. Also, commonly used methods such as mean imputation or complete case analysis in training such models have historically yielded biased results (Ibrahim and Molenberghs, 2009). Multiple Imputation by Chained Equations (*mice*) (Van Buuren and Groothuis-Oudshoorn, 2011) has also been widely employed to handle missing data in a supervised learning setting, but *mice* is unable to generalize a trained imputation model to handle missingness that may exist in a separate test set (Hoogland et al., 2020). Also, using multiple imputation approaches may not be feasible to apply when the downstream model is computationally intensive, such as in the setting of training a deep learning neural network, since one must train the model separately for each imputed dataset. Moreover, existing approaches to handle MAR or MCAR missingness when training deep learning models for supervised learning tasks are currently limited, and have not been sufficiently explored in the literature.

1.5 Existing Missing Data Methods

Non-deep learning methods. Traditionally, there have been several naïve methods of handling missing data. In deep learning applications, naïve approaches to handling missing data, such as listwise deletion, mean substitution, regression imputation, and last observation carried forward (Kang, 2013) have been commonly employed.

In listwise deletion, missing observations are simply omitted from analysis, keeping just the fully-observed cases. This is typically valid only in the MCAR setting, and results in larger standard errors, wider confidence intervals, and lower power in testing hypotheses (Allison et al., 2010). In mean substitution, the missing values for each feature are imputed by the average of the observed values for that feature. This is also only valid in a strictly MCAR setting, and has been shown to give biased estimates (Haitovsky, 1968) and to underestimate the standard errors by inflating the sample size (Kang, 2013). In regression imputation, a linear regression is fit with observed data to impute the missing entries. Regression imputation is an improvement from mean imputation, but most imputation-based methods of missing data tend to underestimate standard errors and overinflate the confidence of results, since there is no way to distinguish between imputed values and observed values (Allison et al., 2010).

Last observation carried forward (LOCF) is typically employed in the longitudinal setting, where repeated measures for subjects are correlated. LOCF simply carries over the immediately previous measurement of a particular feature to impute a missing measurement for that feature for the same subject (Hamer and Simpson, 2009). LOCF makes the strong and unrealistic assumption that no change in the measurement occurs across the two time points.

`missForest` (Stekhoven and Buhlmann, 2011) is a regression tree-based method that has yielded state-of-the-art results in imputation. It is related to regression imputation, utilizing a random forest on the observed values. In particular, for one fixed variable, a random forest is fit on the observed entries, using the observed values of the other variables pertaining to each non-missing observation in the fixed variable. Then, after training on these non-missing observations, the trained random forest is used to predict missing observations of the fixed variable, using the corresponding observed values of all other variables. This process is repeated for all variables in the dataset, until all missing entries have been imputed. Due to the use of regression trees, `missForest` can be used for datasets of mixed data types, and can handle up to MAR missingness. Also, `missForest` is computationally efficient and can handle high-dimensional data (Stekhoven and Buhlmann, 2011).

`mice` (Van Buuren and Groothuis-Oudshoorn, 2011) is a popular state-of-the-art method that performs multiple imputation, and utilizes Rubin's rules to pool coefficient estimates based upon models fit on the multiply-imputed datasets. It uses a fully-conditional model to iteratively update the missing entries, using chained equations. `mice` has been shown to perform very well under MCAR and MAR missingness in both imputing missing values, and in analyzing downstream models using multiply-imputed datasets. However, it cannot generalize the learned imputation model to impute missing values that may exist at test time. Furthermore, it may not perform well when complex nonlinear interactions exist between variables of a dataset, as it typically assumes a linear relationship between features.

Deep learning VAE methods. Deep learning VAE methods of handling missing data typically take advantage of the encoder-decoder architecture to generate data pertaining to missing entries. One of the first VAE methods of imputation to handle MAR missingness was proposed by Rezende et al. (2014), in which they first train a VAE using the subset of the observations that are completely observed, or nonmissing. Upon training the VAE, the variational posterior is assumed to be a good approximation of the true posterior. In this setting, an imputation procedure was proposed, where missing values are initially replaced with random values and fed into the VAE. Then, missing features are sampled from the resulting generative model. The sampled missing features and fixed observed features are then fed into the VAE again, and this process is repeated until the reconstruction error on observed values is small (McCoy et al., 2018). This procedure has the advantage of being simple and straightforward, but it requires the existence of a sufficiently large set of completely-nonmissing observations with which a VAE can be properly trained such that a good approximate posterior can be learned. Especially in a high-dimensional setting, such a set is not guaranteed to exist, limiting the utility of this scheme. Also, the iterative process of imputation after training the VAE can be very time-consuming, making it impractical for larger datasets.

`VAEAC` (Ivanov et al., 2019) utilizes the Conditional VAE (Sohn et al., 2015), which allows for conditioning on an arbitrary subset of the features. Like the scheme by Rezende et al. (2014), `VAEAC` is similarly designed for datasets with a completely-observed set of observations. Specifically, they induce random missingness on the completely-nonmissing observations to train the VAE to learn to impute these missing values. They additionally condition missing values on the observed in their lower bound to impute values of up to MAR missingness. Compared to other methods, the architecture of `VAEAC` is very intricate, as it was mainly designed to process pixel data. Thus, the VAE is trained on a much fewer number of epochs than some other VAE missing data methods.

`HI-VAE` (Nazabal et al., 2018) is a VAE-based method that, like `VAEAC`, does not require a fully-observed training set. Instead, missing features are replaced with 0's, and the terms in the ELBO are calculated with respect to just the observed values. This is possible by assuming that missingness is MCAR or MAR, such that the missingness can be considered ignorable and the missingness mechanism need not be modelled explicitly (Rubin, 1976). Additionally `HI-VAE` is able to deal with heterogeneous data types, including categorical and binary data, by casting different assumptions on the generative model for each data type. Additionally, they include an additional latent variable to denote cluster membership, such that one can identify underlying subgroups within data without relying upon heuristic clustering methods on the lower-dimensional latent space.

`MIWAE` (Mattei and Frellsen, 2019), like `HI-VAE` and `VAEAC`, does not require a fully-observed training set. However, `MIWAE` utilizes the Importance-weighted AE (IWAE) architecture (Burda et al., 2015), which more tightly lower bounds the $\log$ marginal distribution than the traditional ELBO. Similar to `HI-VAE`, `MIWAE` also computes the corresponding lower bound on just the observed features, thereby allowing up to ignorable MCAR or MAR missingness. Missing features are imputed after training, using an approximate posterior with respect to the observed values. In particular, `MIWAE` utilizes an importance sampling scheme to compute the expectation of the missing features, conditional on the observed features, and uses Monte Carlo integration to approximate the expectation using samples from the latent space.

Although the deep learning methods described in this section can perform imputation, which can aid in downstream modelling procedures such as prediction, and can learn complex interactions between features which may exist in real data, none of these existing methods are able to properly handle missingness that is MNAR, and thus may yield biased results in imputation and downstream models.

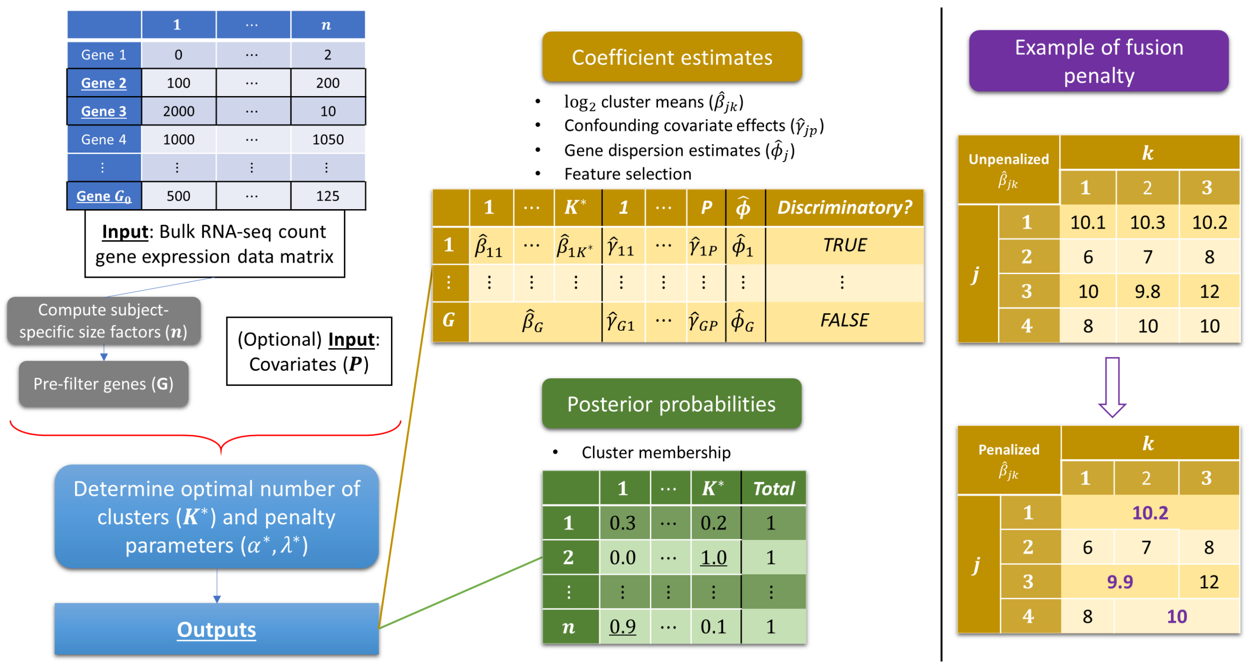
Additionally, these methods are all unsupervised learning methods for imputation of missing values, and do not contain a mode to perform supervised learning, such as prediction. Utilizing a method to impute the training set to perform impute-then-regress modelling may not work when missingness exists in the test set. A multiple imputation method like `mice` may also be impractical if the downstream model is very complex, as this model must be fit on each of the multiply-imputed datasets.

CHAPTER 2: FSCSEQ: SIMULTANEOUS FEATURE SELECTION AND CLUSTERING OF BULK RNA-SEQ DATA

2.1 Introduction

To address the issues with respect to current methods for clustering samples based on RNA-seq, we propose FSCseq (simultaneous Feature Selection and Clustering of RNA-seq): a penalized finite mixture of regression models that assumes negative-binomially distributed mixture components. In addition to directly clustering samples with respect to RNA-seq gene counts, FSCseq also performs simultaneous feature selection to select cluster-discriminatory genes during the clustering process. We achieve this via a coordinate-wise descent within an iteratively-reweighted least squares algorithm, imposing a fusion penalty on the differences in estimated cluster $\log_2$ mean expression on each individual gene, simultaneously shrinking the estimates closer together while inducing sparsity on the differences across clusters. By modeling cluster means with weighted negative binomial regression models, FSCseq can directly adjust for differences in sequencing depth through the inclusion of size factors in a manner similar to common differential gene expression methods. It can also correct for potential confounders such as batch effects, by including them as regression covariates. We demonstate the utility of these features in improving clustering performance via simulation studies and a real data application to a TCGA breast cancer RNA-seq dataset. Lastly, we illustrate how FSCseq performs subtype prediction in new samples, given a new set of gene expression profiles and a prior fitted FSCseq object. In all, we address critically lacking features in clustering samples whose gene expression profiles are measured via RNA-seq, and demonstrate improved performance with respect to existing methods over a range of conditions.

The workflow for FSCseq and an example visualization of the penalty incorporated in our framework are given in Figure 1. Details of these are explained in Section 2.2.



Input: Bulk RNA-seq count gene expression data matrix
Compute subject-specific size factors (n)
Pre-filter genes (G)
(Optional) Input: Covariates (P)
Determine optimal number of clusters (K^*) and penalty parameters (α^*, λ^*)
Outputs
Coefficient estimates
log$_2$ cluster means ($\hat{\beta}_{jk}$)
Confounding covariate effects ($\hat{\gamma}_{jp}$)
Gene dispersion estimates ($\hat{\phi}_j$)
Feature selection
Posterior probabilities
Cluster membership
Example of fusion penalty

	1	⋯	n
Gene 1	0	⋯	2
Gene 2	100	⋯	200
Gene 3	2000	⋯	10
Gene 4	1000	⋯	1050
⋮	⋮	⋮	⋮
Gene G_0	500	⋯	125

	1	⋯	K^*	1	⋯	P	$\hat{\phi}$	Discriminatory?
1	$\hat{\beta}_{11}$	⋯	$\hat{\beta}_{1K^*}$	$\hat{\gamma}_{11}$	⋯	$\hat{\gamma}_{1P}$	$\hat{\phi}_1$	TRUE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
G	$\hat{\beta}_G$			$\hat{\gamma}_{G1}$	⋯	$\hat{\gamma}_{GP}$	$\hat{\phi}_G$	FALSE

	1	⋯	K^*	Total
1	0.3	⋯	0.2	1
2	0.0	⋯	1.0	1
⋮	⋮	⋮	⋮	⋮
n	0.9	⋯	0.1	1

Unpenalized $\hat{\beta}_{jk}$		k		
		1	2	3
j	1	10.1	10.3	10.2
	2	6	7	8
	3	10	9.8	12
	4	8	10	10

Penalized $\hat{\beta}_{jk}$		k		
		1	2	3
j	1	10.2		
	2	6	7	8
	3	9.9		12
	4	8	10	

Figure 1: Example workflow of FSCseq (left) and visualization of the mechanics of the fusion SCAD penalty (right) imposed on $\hat{\beta}_{jk}$ with $j = 1, \dots, 4$ and $k = 1, \dots, 3$. For prediction, the coefficient estimates and the list of cluster-discriminatory genes obtained from FSCseq can be used to compute posterior probabilities of cluster membership on new samples. Importantly, the fusion penalty is imposed on each gene separately, and clusters are fused together within each given gene separately. When all clusters are fused together for a particular gene ($j = 1$, right), that gene is determined to be nondiscriminatory across clusters. Otherwise, the gene is considered to discriminate in expression across clusters.

2.2 Methods

The main utility of FSCseq is two-fold: unsupervised clustering of subjects based on subject-specific RNA-seq gene expression profiles, and simultaneous selection of cluster-discriminatory genes. Our modeling approach can correct for sources of technical variation like sequencing depth, adjust for effects of potential confounders during clustering such as batch effects, and directly predict discovered subtypes in new patients based upon a previously fitted FSCseq model. We describe our model implementation below.

2.2.1 Model likelihood

Gene-level RNA-seq read counts are utilized as the basis of our model, which can be obtained from RNA-seq gene expression quantification software such as `Salmon` (Patro et al., 2017). Such approaches account for read multi-mapping and other biases and, as a result, often provide non-integer “expected read counts” adjusting for these factors. We round these expected count values to integers, similar to previous approaches (Love et al., 2014). Let $\mathbf{y}$ denote an $n \times G$ matrix of n subjects (or “samples”) and G genes (or “features”), with each element y_{ij} denoting the read count of the j^{th} gene for the i^{th} subject, where $i = 1, \dots, n$ and $j = 1, \dots, G$.

We assume that the RNA-seq gene expression profile of subject i, $\boldsymbol{y}_i = (y_{i1}, \dots, y_{iG})$, can be modeled by a mixture of K G-dimensional multivariate negative binomial regression models. The likelihood of this mixture model may be written as

$$L(\boldsymbol{\mu}, \boldsymbol{\phi}|\mathbf{y}) = f(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\phi}) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{y}_i; \boldsymbol{\mu}_{ik}, \boldsymbol{\phi}), \tag{2.1}$$

where K is the “order” of the mixture model (i.e. the number of assumed clusters in the data), π_k is the mixture proportion pertaining to cluster k, and $\sum_{k=1}^{K} \pi_k = 1$. We denote μ_{ijk} as the mean of the j^{th} gene for the i^{th} subject in the k^{th} cluster, after adjusting for subject-specific sequencing depth, and we denote ϕ_j as the gene-level dispersion parameter pertaining to gene j. Then, $\boldsymbol{\mu}_{ik} = (\mu_{i1k}, \dots, \mu_{iGk})$ is the mean expression profile pertaining to sample i and cluster k, and $\boldsymbol{\phi} = (\phi_1, \dots, \phi_G)$ is a vector of the G gene-level dispersion parameters.

We utilize a gene-gene independence assumption, which has been successfully implemented in prior RNA-seq read count-based clustering methods (Si et al., 2013; Mo et al., 2013; Li et al., 2018), as a reasonable approximation to simplify (2.1). Under this assumption, $f_k(\boldsymbol{y}_i; \boldsymbol{\mu}_{ik}, \boldsymbol{\phi}) = \prod_{j=1}^{G} f_{jk}(y_{ij}; \mu_{ijk}, \phi_j)$, where $f_{jk}(\cdot)$ is the negative binomial probability mass function, given by

$$f_{jk}(y_{ij}; \mu_{ijk}, \phi_j) = \binom{y_{ij} + \phi_j^{-1} - 1}{\phi_j^{-1} - 1} \left[\frac{1}{1 + \phi_j \mu_{ijk}}\right]^{\phi_j^{-1}} \left[\frac{\phi_j \mu_{ijk}}{1 + \phi_j \mu_{ijk}}\right]^{y_{ij}},$$

equivalent to the “NB2” parameterization by Hilbe (2009). Although the multivariate log-normal Poisson or multivariate negative binomial model may alternatively account for potential correlation between genes, the specification and estimation of such a covariance structure across genes is intractable in very high dimensions (Inouye et al., 2017).

For the j^{th} gene, we may specify the link and variance functions with respect to μ_{ijk} as

$$\log_2(\mu_{ijk}) = \beta_{jk} + \sum_{p=1}^{P} x_{ip}\gamma_{jp} + s_i \qquad V(\mu_{ijk}) = \mu_{ijk} + \phi_j \mu_{ijk}^2, \tag{2.2}$$

where β_{jk} is the log base 2 ($\log_2$) mean read count for gene j and cluster k, γ_{jp} is the effect of covariate p on the expression of gene j, and s_i is the $\log_2$ scaled size factor of subject i calculated by `DESeq2` to correct for subject-specific differences in sequencing depth. In other words, we model the $\log_2$ mean of the j^{th} gene for subjects in the k^{th} cluster via a cluster-specific mean parameter (on the $\log_2$ scale) and a set of P subject-level covariates $(x_{i1}, \ldots, x_{iP})$, while also accounting for sequencing depth. We allow covariate effects to be distinct across genes, similar to prior batch correction methods (Johnson et al., 2006; Leek, 2014), as, biologically, certain sets of genes may be affected differently by a particular covariate. Examples of such covariates may include technical factors such as batch, or potential demographic variables such as age or tumor grade. Through the inclusion of covariates in the model, we will later demonstrate FSCseq's ability to adjust cluster mean estimates, thus improving clustering performance in the presence of such factors.

We note that in the given parametrization, we specify gene-level dispersion parameters ϕ_j that are shared across the K clusters within gene j (Robinson et al., 2009; Love et al., 2014). In contrast, NBMB (Li et al., 2018) specifies cluster-specific dispersion parameters $\boldsymbol{\phi}_j = (\phi_{j1} \ldots, \phi_{jK})$ within each gene j. Conceptually, when n is small, the estimate of each ϕ_{jk} may not be accurate due to the smaller average per-cluster sample size (Piegorsch, 1990; Al-Khasawn, 2010). In FSCseq, we utilize gene-level dispersions as the default in order to avoid this issue, and later show that the cluster-specific dispersion scheme by NBMB performs poorly in clustering analyses.

FSCseq identifies cluster-discriminatory genes simultaneously during clustering. To do this, within a given gene j, we impose the SCAD penalty (Fan and Li, 2001) on the pairwise differences between $\log_2$ cluster means, in order to induce shrinkage on these differences and enforce equality of similar group means within gene j. That is, we introduce the penalty term

$$p_\lambda(\boldsymbol{\beta}_j) = \sum_{k<l}^{K} SCAD_\lambda(|\beta_{jk} - \beta_{jl}|), \tag{2.3}$$

where $\boldsymbol{\beta}_j = (\beta_{j1}, \ldots, \beta_{jK})$, λ is a tuning parameter to control the amount of penalty that is introduced, and k and l are cluster indices such that $1 \leq k < l \leq K$. The explicit form of the SCAD penalty is most commonly given by its first derivative: $SCAD'_{\lambda^*}(\theta) = \lambda^*\{I(\theta \leq \lambda^*) + [(a\lambda^* - \theta)_+/(a\lambda^* - \lambda^*)]I(\theta > \lambda^*)\}$ for $a > 2$, $\theta > 0$, and $\lambda^* = \lambda\alpha$. As suggested by Fan and Li (2001), we set $a = 3.7$.

Then, we aim to maximize the penalized log-likelihood, which may be written as

$$\begin{aligned}\log[L_p(\boldsymbol{\Psi})] &= \log[L(\boldsymbol{\mu}, \boldsymbol{\phi}|\mathbf{y})] + Penalty \\ &= \sum_{i=1}^{n} \log\left\{\sum_{k=1}^{K} \pi_k \prod_{j=1}^{G} f_{jk}(y_{ij}; s_i, \beta_{jk}, \boldsymbol{\gamma}_j, \phi_j)\right\} \\ &\qquad + \sum_{j=1}^{G} p_\lambda(\boldsymbol{\beta}_j),\end{aligned} \tag{2.4}$$

after incorporating the link function in (2.2) and the SCAD penalty. Here, $\boldsymbol{\Psi} = (\boldsymbol{\pi}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\phi})$ pertains to the vector of model parameters, with $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_G)$, and $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_G)$. Here, $\boldsymbol{\gamma}_j = (\gamma_{j1}, \ldots, \gamma_{jP})$ denotes the regression coefficients pertaining to the P covariates in gene j, $j = 1, \ldots, G$. The first term of (2.4) is the $\log$ of the model likelihood in (2.1), with μ_{ijk} replaced by $(s_i, \beta_{jk}, \boldsymbol{\gamma}_j)$, since μ_{ijk} is a function of $(s_i, \beta_{jk}, \boldsymbol{\gamma}_j)$ by (2.2).

However, because the penalty is imposed on the differences in cluster $\log_2$ mean parameters in gene j, it is not separable in $\boldsymbol{\beta}_j$. As a result, the penalized log-likelihood may not converge to a stationary point during maximization (Friedman et al., 2007; Wu and Lange, 2008). Similar to Pan et al. (2013), we address this by introducing a new variable $\theta_{j,kl} = \beta_{jk} - \beta_{jl}$ for each gene $j = 1, \ldots, G$ and each pair of clusters k and l such that $1 \leq k < l \leq K$. We then apply the quadratic penalty method (Nocedal and Wright, 1999) leveraging this new constraint. In particular, for a given gene j, let $\boldsymbol{\theta}_j$ be a $K \times K$ upper triangular matrix denoting the pairwise differences in $\boldsymbol{\beta}_j$, with entries $(\boldsymbol{\theta}_j)_{kl} = \theta_{j,kl}$. Then, under the quadratic penalty method, (2.3) can be rewritten as

$$p_{\lambda,\alpha}(\boldsymbol{\beta}_j) = \frac{\lambda(1-\alpha)}{2} \sum_{k<l} (\beta_{jk} - \beta_{jl} - \theta_{j,kl})^2 + \sum_{k<l} SCAD_{\lambda\alpha}(|\theta_{j,kl}|), \tag{2.5}$$

where the first term is the added quadratic penalty term as described by Nocedal and Wright (1999), pertaining to the constraint $\theta_{j,kl} = \beta_{jk} - \beta_{jl}$. This first term in (2.5) is convex and differentiable, while the second term is non-smooth, but separable and convex, and so a coordinate-wise descent algorithm is guaranteed to converge to the global optimum (Tseng, 2001). In FSCseq, we utilize this property to implement a coordinate-wise descent algorithm (CDA) to estimate cluster $\log_2$ means and covariate effects within each given gene $j = 1, \ldots, G$. We also introduce the hyperparameter α to control the balance between these two terms. Although similar in form, we note that this is not the traditional elastic net penalty (Zou and Hastie, 2005). Rather, this penalty is similar to the L1 penalty with the quadratic penalty method proposed by (Pan et al., 2013), with the L1 replaced by the SCAD penalty.

We also note that $\beta_{jk} - \beta_{jl} \to 0$ as $\theta_{j,kl} \to 0$. However, similar to Pan et al. (2013), we observe that $\beta_{jk} - \beta_{jl} - \theta_{j,kl} \to 0$ as $\lambda(1-\alpha) \to \infty$, but $\beta_{jk} - \beta_{jl} = 0$ cannot be guaranteed. In particular, let $\hat{\theta}_{j,kl}$ and $\hat{\beta}_{jk}$ denote the estimates of $\theta_{j,kl}$ and β_{jk}, respectively. Then, although $\hat{\theta}_{j,kl} = 0$ can be attained by the SCAD penalty, $\hat{\beta}_{jk} = \hat{\beta}_{jl}$ cannot be guaranteed for two distinct clusters k and l. This is because the term introduced by the quadratic penalty method (Nocedal and Wright, 1999) to enforce the constraint $\theta_{j,kl} = \beta_{jk} - \beta_{jl}$ is a ridge-like term, and thus does not perform strict thresholding, much like the classic L2 penalty (Hoerl and Kennard, 1970). Increasing the first term by increasing $\lambda(1-\alpha)$ more closely approximates equality in the constraint, but this introduces bias in the estimation of $\boldsymbol{\beta}_j$ (Pan et al., 2013).

In FSCseq, we address this by fusing clusters k and l together into one joint cluster for gene j if $\hat{\theta}_{j,kl} = 0$, denoting the index of this new cluster as k_*. Specifically, we estimate $\hat{\beta}_{jk_*}$ using data from both clusters k and l in gene j, and we strictly enforce the quadratic penalty method constraint by setting $\hat{\beta}_{jk} = \hat{\beta}_{jl} = \hat{\beta}_{jk_*}$. If all clusters are fused together in this way for a particular gene j, then gene j is determined to be "nondiscriminatory" across clusters. Otherwise, gene j is considered "cluster-discriminatory." We call this penalty the "SCAD fusion penalty".

We select optimal values of the three tuning parameters (K, α, and λ) by searching over a grid of possible values for α and λ with K fixed, and then repeat this procedure for every candidate value of K. We utilize the Bayesian Information Criterion (BIC) for selection (Schwarz, 1978). Details of this tuning parameter selection procedure is given in Section 2.2.2.

2.2.2 Computation

We maximize the penalized log-likelihood from (2.4) via a Classification Expectation-Maximization (CEM) algorithm with simulated annealing (Celeux and Govaert, 1992) to reduce the likelihood of converging to a local maximum. To reduce computational overhead per iteration, we also apply "mini-batching" of genes. That is, we update the estimates of parameters pertaining to a random subset (or a 'mini-batch') of the genes in the M-step, and then perform the E-step after each 'mini-batched' M-step. This is in contrast to the standard EM algorithm, where we would only perform the E-step after estimating parameters for all genes in the M-step. Thus, mini-batching allows the E-step to be updated more quickly with M-step updates from a subset of the parameter space, similar to the multicycle ECM algorithm (Meng and Rubin, 1993) which has been shown to decrease overall computation time (Meng, 1994). We show later that this mini-batching approach works well in our simulation and real data analyses.

The complete data penalized log-likelihood, which corresponds to the penalized log-likelihood in (2.4), can be written as

$$\begin{aligned}\log[L_{cp}(\mathbf{\Psi})] = \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\left\{\log(\pi_k) + \log\prod_{j=1}^{G} f_{jk}(y_{ij}; s_i, \beta_{jk}, \boldsymbol{\gamma}_j, \phi_j)\right\} \\ + \sum_{j=1}^{G} p_{\lambda,\alpha}(\boldsymbol{\beta}_j),\end{aligned} \tag{2.6}$$

where $z_{ik} = I(z_i = k)$ is the indicator of whether subject i belongs to cluster k. In this context, z_{ik} is a latent variable denoting cluster membership of sample i in cluster k. In the standard EM algorithm, the so-called "Q-function" is then calculated as the conditional expectation of $\log[L_{cp}(\mathbf{\Psi})]$, given the current parameter estimates and observed data (McLachlan and Krishnan, 2008; Garcia et al., 2009). This expectation is calculated in the E-step, and parameter estimates are updated in the M-step by maximizing the Q-function.

The rest of this section is organized as follows. To simplify the presentation of our method, we first describe the general E and M-steps for our algorithm without mini-batching, given some fixed values of K, α, and λ. Then, we describe the modifications of these steps to support gene mini-batching. We then show how the FSCseq algorithm is initialized, outline convergence criteria, and describe our procedure for selecting the optimal values of K, α and λ. We conclude this section by describing a method for subtype prediction using the converged FSCseq model.

We first describe the E-step with no mini-batching of genes. Let m denote the current EM iteration, $\hat{\mathbf{\Psi}}^{(m)} = (\hat{\boldsymbol{\pi}}^{(m)}, \hat{\boldsymbol{\beta}}^{(m)}, \hat{\boldsymbol{\gamma}}^{(m)}, \hat{\boldsymbol{\phi}}^{(m)})$ denote the estimates of $(\boldsymbol{\pi}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\phi})$ in the current iteration, and $\hat{z}_{ik}^{(m)} = E[z_{ik} \mid \mathbf{y}, \mathbf{s}, \hat{\mathbf{\Psi}}^{(m)}]$ denote the conditional expectation of z_{ik}, given the observed data and the current parameter estimates, where $\mathbf{s} = (s_1, \ldots, s_n)$ is the vector of fixed size factor offsets calculated *a priori* by `DESeq2` (Love et al., 2014). Given the form of (2.6), evaluation of the Q-function requires only the calculation of $\hat{z}_{ik}^{(m)}$, which can be interpreted as the posterior probability of subject i belonging to cluster k given the parameter estimates at the m^{th} EM iteration. Thus, in the standard EM algorithm, the m^{th} E-step reduces to computing $\hat{z}_{ik}^{(m)}$ for $i = 1, \ldots, n$ and $k = 1, \ldots, K$ by the following expression:

$$\hat{z}_{ik}^{(m)} = \frac{\hat{\pi}_k^{(m)} \prod_{j=1}^{G} f_{jk}(y_{ij}; s_i, \hat{\beta}_{jk}^{(m)}, \hat{\boldsymbol{\gamma}}_j^{(m)}, \hat{\phi}_j^{(m)})}{\sum_{k'=1}^{K} \hat{\pi}_{k'}^{(m)} \prod_{j=1}^{G} f_{jk'}(y_{ij}; s_i, \hat{\beta}_{jk'}^{(m)}, \hat{\boldsymbol{\gamma}}_j^{(m)}, \hat{\phi}_j^{(m)})}. \tag{2.7}$$

These posterior probabilities may then be used as cluster-specific observation weights in the M-step. Following convergence, it is also common to assign cluster membership to a subject by determining the cluster with the maximum posterior probability in that subject. That is, we can assign subject i to the cluster corresponding to $\arg\max_k(\hat{z}_{ik})$ at convergence. We note from (2.7) that the contribution of cluster-nondiscriminatory genes cancel out in the posterior probability calculations.

However, the standard EM algorithm is known for its susceptibility to converge to a local optimum, rather than the global optimum (Dellaert, 2002). To reduce the likelihood of convergence to a local optimum, we modify the E-step in a manner similar to the Classification EM (CEM) algorithm with simulated annealing (Celeux and Govaert, 1990; Si et al., 2013). In the original CEM with simulated annealing algorithm (Celeux and Govaert, 1990), the E-step update in (2.7) is modified (denoted as the "AE-step"), and an additional sampling step (denoted as the "C-step") is added after the AE-step.

In the AE-step, a "temperature" term is incorporated into the E-step update, and is gradually decreased (or "annealed") such that the values of $\hat{z}_{ik}$ approach 0 or 1. The AE-step update corresponding to (2.7) is written as

$$\hat{z}_{ik}^{(m)} = \frac{[\hat{\pi}_k^{(m)} \prod_{j=1}^{G} f_{jk}(y_{ij}; s_i, \hat{\beta}_{jk}^{(m)}, \hat{\boldsymbol{\gamma}}_j^{(m)}, \hat{\phi}_j^{(m)})]^{1/\tau^{(m)}}}{\sum_{k'=1}^{K} [\hat{\pi}_{k'}^{(m)} \prod_{j=1}^{G} f_{jk'}(y_{ij}; s_i, \hat{\beta}_{jk'}^{(m)}, \hat{\boldsymbol{\gamma}}_j^{(m)}, \hat{\phi}_j^{(m)})]^{1/\tau^{(m)}}}, \tag{2.8}$$

where $\tau^{(m)}$ is the temperature of annealing in the m^{th} AE-step iteration. As $\tau^{(m)} \to \infty$, $\hat{z}_{ik}^{(m)} \to 1/K$, maximizing the entropy of the partition. As $\tau^{(m)} \to 0$, we approach a hard partition for cluster membership (Rose, 1998) because the elements of the vector of posterior probabilities $\hat{\mathbf{z}}_i^{(m)} = (\hat{z}_{i1}^{(m)}, \ldots, \hat{z}_{iK}^{(m)})$ tend to 0 or 1, and $\hat{z}_{ik} \in (0, 1)$ with $\sum_{k=1}^{K} \hat{z}_{ik}^{(m)} = 1$ for all subjects $i = 1, \ldots, n$. Traditionally, one allows $\tau^{(m)} \to 0$ as $m \to \infty$ at a pre-specified annealing rate r such that $0 < r < 1$ and $\tau^{(m+1)} = r\tau^{(m)}$. In this way, the AE-step modification allows the CEM to avoid a local optimum by gradually reducing the stochasticity introduced by the sampling in the C-step.

In the C-step, a sample $\mathbf{c}_i = (c_{i1}, \ldots, c_{iK})$ is drawn from the multinomial distribution with probabilities equal to $\hat{\mathbf{z}}_i^{(m)}$ from (2.8). Then, we set $\hat{\mathbf{z}}_i^{(m)} \leftarrow \mathbf{c}_i$, such that the $\hat{\mathbf{z}}_i^{(m)}$ now specify a hard classification of sample i, i.e. for a given sample i, $\hat{z}_{ik}^{(m)}$ is now exactly 1 for one value of k, and 0 for all other values of k. This is repeated for all samples $i = 1, \ldots, n$, yielding a hard classification partition of samples.

The combination of sampling in the C-step and the inclusion of annealing in the AE-step together helps the algorithm escape potential local optima. This is achieved by initializing $\tau^{(m)}$ to be large in the first iterations, such that the $\hat{z}_{ik}^{(m)}$ from the AE-step are close to $1/K$ for all i and k. Then, the probability of yielding a random partition of subjects in the C-step is high in the first iterations. By decreasing the value of $\tau^{(m)}$, the probability of random partitioning approaches 0. Thus, a transition corresponding to a decrease in the objective function can be accepted with non-zero probability, and this probability approaches 0 as the algorithm proceeds (Celeux and Govaert, 1992). This results in the useful property for the CEM of not terminating when reaching the first local optimum.

Following the guidelines by Celeux and Govaert (1992) and Rose (1998), we set the annealing rate $r = 0.9$, similar to other methods (Si et al., 2013; Li et al., 2018). It is suggested that the entropy should be sufficiently large in early iterations, eventually shrinking to 0 in later iterations (Klein and Dubes, 1989; van Laarhoven and Aarts, 1987; Rose, 1998). Additionally, if the initial temperature $\tau^{(1)}$ is too small, annealing may terminate at a suboptimal solution (Celeux and Govaert, 1992). We address this by setting the value of $\tau^{(1)} = G$. We find that this works well in our simulations and real data analyses.

In the M-step, we maximize the Q-function, given the current E-step values $\hat{\mathbf{z}}^{(m)} = (\hat{\mathbf{z}}_1^{(m)}, \ldots, \hat{\mathbf{z}}_n^{(m)})$ and some specified values of α and λ, updating the current estimates of the model parameters. From (2.6), the estimation of π_k is separable from the remaining model parameters, and the corresponding updates for $\hat{\boldsymbol{\pi}} = (\hat{\pi}_1, \ldots, \hat{\pi}_K)$ are given by $\hat{\pi}_k^{(m+1)} = \sum_{i=1}^{n} \hat{z}_{ik}^{(m)}/n$ for $k = 1, \ldots, K$. We also observe that the updates of the remaining model parameters are separable across genes. That is, we may obtain corresponding estimates $(\hat{\boldsymbol{\beta}}_j^{(m+1)}, \hat{\boldsymbol{\gamma}}_j^{(m+1)}, \hat{\phi}_j^{(m+1)})$ for each gene j separately of all other genes at each M-step, as these parameters are not shared across genes. This observation also facilitates the mini-batching approach described later in this section. First, we outline the M-step without mini-batching.

Specifically, for a given gene j, we utilize Iteratively Reweighted Least Squares (IRLS) with an embedded Coordinate Descent Algorithm (CDA) that updates $(\hat{\boldsymbol{\beta}}_j^{(m+1)}, \hat{\boldsymbol{\gamma}}_j^{(m+1)})$, given specified values of α, λ, $\hat{\mathbf{z}}^{(m)}$, and $\hat{\boldsymbol{\Psi}}^{(m)}$. The update scheme for a particular gene j consists of two nested loops: an inner loop corresponding to the CDA and an outer loop corresponding to the IRLS. The IRLS weights and working responses are updated in the outer loop, and fed into the inner loop to update the parameter estimates in a coordinate-descent fashion. After convergence of IRLS, we estimate $\hat{\phi}_j^{(m+1)}$ using Newton-Raphson. The entire M-step procedure is outlined in Algorithm 1.

Algorithm 1 M-step

```
 1: Input $\hat{\mathbf{z}}^{(m)}$
 2: Update $\hat{\boldsymbol{\pi}}^{(m+1)}$
 3: for $j \in \{1,\ldots,G\}$ do                                          ▷ Cycle through all genes
 4:     Input $(\hat{\boldsymbol{\beta}}_j^{(m)}, \hat{\boldsymbol{\gamma}}_j^{(m)}, \hat{\phi}_j^{(m)})$
 5:     while IRLS not converged do                                         ▷ IRLS loop
 6:         Compute IRLS weights and working response
 7:         while CDA not converged do                                      ▷ CDA loop
 8:             Update $\hat{\boldsymbol{\theta}}_j$ using previous CDA updates of $\hat{\boldsymbol{\beta}}_j$
 9:             for $k \in \{1,\ldots,K\}$ do
10:                 Update $\hat{\beta}_{jk}$
11:             for $p \in \{1,\ldots,P\}$ do
12:                 Update $\hat{\gamma}_{jp}$
13:     Replace $(\hat{\boldsymbol{\beta}}_j^{(m+1)}, \hat{\boldsymbol{\gamma}}_j^{(m+1)}) \leftarrow (\hat{\boldsymbol{\beta}}_j, \hat{\boldsymbol{\gamma}}_j)$     ▷ Final CDA updates
14:     Update $\hat{\phi}_j^{(m+1)}$                                       ▷ via Newton-Raphson
15: Output $(\hat{\boldsymbol{\pi}}^{(m+1)}, \hat{\boldsymbol{\beta}}^{(m+1)}, \hat{\boldsymbol{\gamma}}^{(m+1)}, \hat{\boldsymbol{\phi}}^{(m+1)})$
```

$\hat{\beta}_{jk}$, $\hat{\gamma}_{jp}$, and $\hat{\boldsymbol{\theta}}_j$ denote the CDA updates of β_{jk}, γ_{jp}, and $\boldsymbol{\theta}_j$, respectively, and $\hat{\boldsymbol{\theta}}_j$ facilitates coordinate-wise descent updates of $\hat{\boldsymbol{\beta}}_j$. Notably, the fusion SCAD penalty is imposed on the updates of $\hat{\boldsymbol{\beta}}_j$, but updates of all other parameters are not penalized.

The coordinate-wise descent algorithm (CDA) updates for $(\theta_{j,kl}, \beta_{jk}, \gamma_{jp})$ for each gene $j = 1, \ldots, G$, clusters $1 \le k < l \le K$, and covariates $p = 1, \ldots, P$ are given in this section. Current estimates of parameters are denoted by tilde, e.g. $\tilde{\beta}_{jk}$, and the new CDA update is denoted by a hat, e.g. $\hat{\beta}_{jk}$. Let $(m+1)$ be the current iteration index of the M-step in the EM/CEM algorithm. Then, in the first IRLS iteration, the values of the current estimates from the previous M step update are initialized in CDA as $(\tilde{\beta}_{jk}, \tilde{\gamma}_{jp}) \leftarrow (\hat{\beta}_{jk}^{(m)}, \hat{\gamma}_{jp}^{(m)})$. Subsequent CDA loops are initialized with the most recent CDA updates. For clarity of notation, let v_{ij} denote the i^{th} element of the diagonal entries of $\mathbf{W}_j$ for all $i = 1, \ldots, N$ with $N = n \cdot K$. Then, the CDA update equations for fixed gene j are given as follows:

$$\hat{\theta}_{j,kl} = \begin{cases} sgn(\tilde{\theta}_{j,kl}) \left(\left|\tilde{\theta}_{j,kl}\right| - \dfrac{\alpha}{1-\alpha} \right)_+, & \left|\tilde{\theta}_{j,kl}\right| \leq \lambda^* \\ sgn(\tilde{\omega}_{j,kl}) \left(\left|\tilde{\omega}_{j,kl}\right| - \dfrac{a\frac{\alpha}{1-\alpha}}{a - 1 - \frac{1}{\lambda(1-\alpha)}} \right)_+, & \lambda^* < \left|\tilde{\theta}_{j,kl}\right| \leq a\lambda\alpha \\ \tilde{\theta}_{j,kl}, & \left|\tilde{\theta}_{j,kl}\right| > a\lambda\alpha \end{cases}$$

$$\hat{\beta}_{jk} = \frac{\frac{1}{n_k}\sum_{i=1}^N v_{ij}x_{ik}\tilde{r}_{ijk} + \lambda(1-\alpha)[\sum_{l>k}(\tilde{\beta}_{jl} + \tilde{\theta}_{j,kl}) + \sum_{l<k}(\tilde{\beta}_{jl} - \tilde{\theta}_{j,lk})]}{\lambda(1-\alpha)(K-1) + \frac{1}{n_k}\sum_{i=1}^N v_{ij}x_{ik}^2} \tag{2.9}$$

$$\hat{\gamma}_{jp} = \frac{\sum_{i=1}^N v_{ij}x_{ip}\tilde{r}_{ijp}}{\sum_{i=1}^N v_{ij}x_{ip}^2}$$

where $(a)_+$ is equal to a if $a > 0$ and is equal to 0 otherwise, and x_{ik} is the i^{th} row and k^{th} column entry of the design matrix $\mathbf{X}$ as defined in Section A2. The effective number of observations used for estimation of parameters is $N = n \cdot K$, denoting each of the IRLS-weighted n samples in each of the K clusters. The estimated number of samples in cluster k (at the current M-step) used for estimation of β_{jk} is $n_k = \hat{\pi}_k^{(m+1)} n$. Also, $\tilde{r}_{ijk} = \tilde{y}_{ijk} - \sum_{c \neq k}^K x_{ic}\tilde{\beta}_{jc} - \sum_{p=1}^P x_{ip}\tilde{\gamma}_{jp}$; $\tilde{r}_{ijp} = \tilde{y}_{ijk} - \sum_{c=1}^K x_{ic}\tilde{\beta}_{jc} - \sum_{q \neq p}^P x_{iq}\tilde{\gamma}_{jq}$; $\tilde{\omega}_{j,kl} = \frac{(a-1)\tilde{\theta}_{j,kl}}{a-1-\frac{1}{\lambda(1-\alpha)}}$; and $\lambda^* = \dfrac{\alpha}{1-\alpha} + \lambda\alpha$.

We iteratively update these parameters until convergence of CDA (inner loop) for fixed gene j. Then, we recompute the IRLS weights and repeat, until the IRLS (outer loop) converges. The dispersion estimate $\hat{\phi}_j$ is attained upon convergence of IRLS, as described in Section A2. Then, we output all of the final updates as the estimates from the current $(m+1)^{th}$ M step, i.e. $(\hat{\beta}_{jk}^{(m+1)}, \hat{\gamma}_{jp}^{(m+1)}, \hat{\pi}_k^{(m+1)}, \hat{\phi}_j^{(m+1)}) \leftarrow (\hat{\beta}_{jk}, \hat{\gamma}_{jp}, \hat{\pi}_k, \hat{\phi}_j)$. This entire procedure is repeated for each gene $j = 1, \ldots, G$, or if mini-batching, $j \in minibatch^{(m+1)}$, where $minibatch^{(m+1)}$ is the mini-batch of genes that were selected at the current M step. We note that estimation of $\hat{\pi}_k$ is done before the IRLS, as described in Algorithm 1 of the main text.

In order to make our algorithm more scalable to large datasets and to speed up the computation time, we mini-batch genes in the M-step, such that the parameters pertaining to only a subset of genes are estimated during a particular M-step iteration (Neal and Hinton, 1998). Mini-batching of genes is done by randomly drawing a prespecified proportion of the genes whose parameters are updated in that M-step iteration. By default, we set the mini-batch size at each iteration to $G/5$, or 20% of the total genes. Specifically, in the

M-step, we run Algorithm 1 on the pre-specified subset of genes corresponding to the current mini-batch. That is, we update $\hat{\boldsymbol{\pi}}^{(m+1)}$ as before, but we update $(\hat{\boldsymbol{\beta}}_j^{(m+1)}, \hat{\boldsymbol{\gamma}}_j^{(m+1)}, \hat{\boldsymbol{\phi}}_j^{(m+1)})$ only if gene j is in the mini-batch for that M step iteration. This is straightforward to implement due to the separability across genes in parameter updates. If gene j is not in the mini-batch for the current M-step, current estimates are replaced with the previous iteration's estimates, i.e. $(\hat{\boldsymbol{\beta}}_j^{(m+1)}, \hat{\boldsymbol{\gamma}}_j^{(m+1)}, \hat{\boldsymbol{\phi}}_j^{(m+1)}) \leftarrow (\hat{\boldsymbol{\beta}}_j^{(m)}, \hat{\boldsymbol{\gamma}}_j^{(m)}, \hat{\boldsymbol{\phi}}_j^{(m)})$.

The E-step calculations are then done as before using the parameter estimates of all genes, with the updated values pertaining to the included genes of the mini-batch. Although mini-batching requires more CEM iterations to converge, we found that it yields slightly faster convergence than updating all genes at every M-step.

For each candidate value of K, we initialize our algorithm with very small penalty (small values of α and λ) from two sets of informative initial cluster labels (from naïve K-means and hierarchical clustering), and one set of random initial cluster labels drawn from the multinomial distribution with equal probabilities for the K clusters. We run the CEM with mini-batching on these 3 initializations until convergence, and select the converged model that yields the lowest BIC. Then, we run the standard EM algorithm from this selected starting point to convergence. This is similar to the "xCEM-EM" strategy proposed by Biernacki et al. (2003), using the BIC for selection rather than the log-likelihood. Then, we use the estimates and posterior probabilities of the converged model, and perform selection of α and λ using EM. We repeat this initialization procedure for each candidate value of K, tuning α and λ jointly given K via warm starts. This procedure is described in more detail later in this section

The stopping criterion for both the CEM and EM algorithms is based upon a threshold on the relative change in the Q function. Convergence is determined when $\left|(Q^{(m)} - Q^{(m-n_{mb})})/Q^{(m-n_{mb})}\right| < \epsilon_1$ where $Q^{(m)}$ is the value of the total Q-function conditional on current estimates at the m^{th} EM or CEM iteration, and n_{mb} is the minimum number of mini-batches that the data can be divided into (rounded up to the nearest integer). By default, $n_{mb} = 5$ (corresponding to mini-batching 20% of genes), and the left hand side (LHS) would be interpreted as the relative change in the Q-function across 5 iterations. Without mini-batching ($n_{mb} = 1$), the LHS simplifies to the relative change in the Q-function across one iteration.

Convergence of the IRLS and CDA are determined by a threshold on the mean absolute relative change of parameter estimates across one IRLS/CDA iteration. In particular, let (r) index the current IRLS/CDA iteration, such that $\hat{\boldsymbol{\Theta}}_j^{(r)} = (\hat{\boldsymbol{\beta}}_j^{(r)}, \hat{\boldsymbol{\gamma}}_j^{(r)}) = (\hat{\beta}_{j1}^{(r)}, \ldots, \hat{\beta}_{jK}^{(r)}, \hat{\gamma}_{j1}^{(r)}, \ldots, \hat{\gamma}_{jP}^{(r)})$ denote the estimates of the parameters $\boldsymbol{\beta}_j$ and $\boldsymbol{\gamma}_j$ at the end of the current r^{th} IRLS or CDA iteration. Then, for Θ_{jt} denoting the t^{th}

element of $\mathbf{\Theta}_j$, convergence of the IRLS/CDA is attained when:

$$\frac{1}{K+P}\sum_{t=1}^{K+P}\left|\frac{\hat{\Theta}_{jt}^{(r)}-\hat{\Theta}_{jt}^{(r-1)}}{\hat{\Theta}_{jt}^{(r-1)}}\right| < \epsilon_2.$$

In FSCseq, we set $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-4}$ as the default convergence thresholds.

In this section, we outline how optimal values of tuning parameters are selected in FSCseq via "warm starts" (Friedman et al., 2007, 2010). Let $(K^*, \alpha^*, \lambda^*)$ denote the optimal values of $(K$, α, $\lambda)$, respectively. Denote candidate values of α as $(\alpha_1, \ldots, \alpha_A)$ with $\alpha_1 < \cdots < \alpha_A$, and candidate values of λ as $(\lambda_1, \ldots, \lambda_L)$ with $\lambda_1 < \cdots < \lambda_L$, such that there are $A \cdot L$ possible combinations of values for (α, λ). Also, let $\hat{\mathbf{\Psi}}_{a,\ell}$ denote the final estimates of the converged results pertaining to $\alpha = \alpha_a$ and $\lambda = \lambda_\ell$, for a given candidate value of K. Then, the tuning parameter selection proceeds as follows. First, initialize a model with fixed candidate value of K as described earlier, to obtain $\hat{\mathbf{\Psi}}_{1,1}$. Then, leaving α fixed, increase λ to the next smallest value, λ_2. Next, we run the penalized EM algorithm to convergence, utilizing the prior converged result $\hat{\mathbf{\Psi}}_{1,1}$ as the initial value for $\mathbf{\Psi}$, to obtain $\hat{\mathbf{\Psi}}_{1,2}$. Then, we increase λ again and repeat the previous procedure, utilizing $\hat{\mathbf{\Psi}}_{1,2}$ as the initial value for $\mathbf{\Psi}$, to obtain $\hat{\mathbf{\Psi}}_{1,3}$. Repeat until all L candidate values of λ are searched. Then, set $\alpha = \alpha_2$ and $\lambda = \lambda_1$, and run EM to convergence from $\hat{\mathbf{\Psi}}_{1,1}$ to obtain $\hat{\mathbf{\Psi}}_{2,1}$. Repeat this process for each value of α, until all $A \cdot L$ combinations of λ and α are searched. Finally, we repeat this entire procedure for each candidate value of K.

For a given combination of order K and penalty parameters $(\alpha$, $\lambda)$, we may calculate the Bayesian Information Criterion (BIC) pertaining to the converged result, which has the form:

$$BIC = -2\log\hat{L} + q\log(n\cdot G). \tag{2.10}$$

Here, $\hat{L}$ is the marginal likelihood given in (2.1) calculated with the estimates $\hat{\mathbf{\Psi}}$ of the converged model, and q is the total number of estimated parameters. Let K_j denote the number of distinct cluster mean parameters for gene j, with each fused cluster considered as one distinct cluster. Then, $q = G + (K-1) + \sum_{j=1}^{G} K_j$, corresponding to G gene-level dispersion parameters, $(K-1)$ free mixing proportions (with $\pi_K = 1 - \sum_{k=1}^{K-1}\pi_k$), and a total of $\sum_{j=1}^{G} K_j$ distinct $\log_2$ cluster means. It is clear that $1 \leq K_j \leq K$ for all j, since $K_j = 1$ when all clusters are fused together in gene j (gene is determined to be cluster-nondiscriminatory), and $K_j = K$ when no pair of clusters are fused together in gene j. Additionally, given the gene-gene independence assumption from Section 2.2.1, we have an effective sample size of $n \cdot G$, corresponding to the number of entries in the gene expression matrix.

To select the optimal tuning parameter values, we calculate the BIC of each converged model using (2.10), and select $(K^*, \alpha^*, \lambda^*)$ as the unique combination of input values of (K, α, λ) corresponding to the converged model that yielded the lowest BIC.

FSCseq is able to perform prediction on new samples based on the converged model with input values for $(K^*, \alpha^*, \lambda^*)$. Given the RNA-seq read counts of cluster-discriminatory genes in a new sample, we may derive the posterior probabilities of subtype (cluster) membership for this new sample based on (2.7), where we assign subtypes as described previously. Specifically, we calculate the posterior probability of cluster k membership in this new sample, $\hat{z}_{new,k}$, utilizing the estimated parameters from the converged FSCseq model based upon the original dataset (training set). This can be useful for predicting the subtype of new samples (prediction set) without having to re-cluster patients in future studies. We see that the contribution of cluster-nondiscriminatory genes would cancel out in the posterior probability calculations of new samples.

However, correcting for sequencing depth in the new samples can be nontrivial, since the size factors are specific to the training set samples used to fit the FSCseq model. To calculate the size factor of a new sample, we compare the counts of the new sample to the geometric mean of the counts of the training set samples, using the information in the training set as a pseudo-reference (Anders and Huber, 2010). Then, the estimated size factor of a new sample ($\hat{s}_{new}$) is given by:

$$\hat{s}_{new} = \underset{j}{median} \left[\frac{y_{new,j}}{(\prod_{v=1}^{n_{train}} y_{vj}^{train})^{1/n_{train}}} \right], \tag{2.11}$$

where n_{train} denotes the training set sample size, and $y_{new,j}$ and y_{vj}^{train} denote the count of gene j for the new sample and for training set sample v, respectively. As before, the size factors are calculated with the `DESeq2` package.

2.3 Numerical Examples

2.3.1 Simulations

To evaluate our proposed method, we simulated data across an extensive set of conditions, varying the following factors: number of true underlying clusters (K_{true}), sample size (n), $\log_2$ fold change between cluster means (LFC), proportion of cluster-discriminatory genes (p_{DE}), baseline $\log_2$ mean count (β_0) of genes, and overdispersion (ϕ_0) of counts across samples for a given gene. We fix LFC, β_0, and ϕ_0 to be the same across all simulated genes in order to demonstrate the performance of FSCseq in a controlled setting with respect to these factors. We also fixed the number of simulated genes at $G = 10000$. Of these, we specified the first $(p_{DE} \cdot G)$ genes to be cluster-discriminatory, and the rest of the genes to be nondiscriminatory, and we denote the set of cluster-discriminatory genes as G_d. In cancer subtyping data, it is common to see specific genes upregulated or downregulated for just one subtype (Yersal, 2014). Thus, for each cluster-discriminatory gene $j \in G_d$, we randomly select one cluster k_*, $1 \leq k_* \leq K_{true}$, and randomly up-regulate ($\beta_{jk_*} = \beta_0 + LFC$) or down-regulate ($\beta_{jk_*} = \beta_0 - LFC$) the $\log_2$ mean for that cluster, while keeping the expression of the remaining clusters the same as the baseline ($\beta_{jk'} = \beta_0$, for all $k' \neq k_*$), where β_{jk} denotes the $\log_2$ mean of gene j for a sample in cluster k. For each nondiscriminatory gene $j \notin G_d$, we set $\beta_{jk} = \beta_0$ for all $k = 1, \ldots, K_{true}$. In this way, each cluster-discriminatory gene is expected to have one cluster with $\log_2$ mean different from β_0, while each nondiscriminatory gene is expected to have $\log_2$ mean equal to β_0 across all clusters.

We simulated datasets with $K_{true} = (2, 4)$ underlying groups with 25 and 50 samples per cluster, such that $n = (50, 100)$ for $K_{true} = 2$, and $n = (100, 200)$ for $K_{true} = 4$. Additionally, we considered $p_{DE} = (0.025, 0.050)$, $\beta_0 = (8, 12)$, $\phi_0 = (0.15, 0.35, 0.50)$ and $LFC = (1, 2)$ in our simulations. These values were determined using RNA-seq data from the TCGA Breast Cancer project (Grossman et al., 2016). Specifically, after removing outlier genes, we fit a negative binomial regression model to expression counts from each gene one-by-one, utilizing $\log_2$ size factors calculated from `DESeq2` as offsets and the 5 annotated PAM50 subtypes as covariates, such that a cluster mean is estimated for each subtype. The corresponding $n \times 5$ design matrix (via cell-means coding) has i^{th} row and k^{th} column entry $x_{ik} = 1$ if tumor sample i has annotated PAM50 subtype k, and $x_{ik} = 0$ otherwise, for $i = 1, \ldots, n$ and $k = 1, \ldots, 5$. Based upon the results from these models, the 50^{th} and 75^{th} quantiles of the estimated LFCs in the TCGA Breast Cancer dataset were 0.875 and 1.72, and the median estimated $\hat{\beta}_{jk}$ and $\hat{\phi}_j$ were 10.2 and 0.354, respectively. These values informed our simulation parameters above. We also simulated realistic between-sample technical variation due to sequencing depth by simulating size factors for each sample $i = 1, \ldots, n$ from $s_i \sim N(1, 0.25)$, based upon the estimates acquired by `DESeq2` on the same TCGA samples.

To prevent our simulation model from being identical to our model framework, we also introduced Gaussian noise to the $\log_2$ expression of each count, drawn from $\sigma_{ij} \sim N(0, 0.1)$ for each sample i and gene

j. Then, the expression of each gene $j = 1, \dots, G$ for subjects $i = 1, \dots, n$ in cluster $k = 1, \dots, K_{true}$ was simulated from the negative binomial distribution $NB(\mu_{ijk}, \phi_0)$ with mean μ_{ijk}, such that

$$\log_2(\mu_{ijk}) = s_i + \beta_{jk} + \sigma_{ij}, \qquad (2.12)$$

and dispersion parameter ϕ_0. We simulated 100 datasets for the main simulation analyses below, and 25 datasets for the rest of the analyses. In order to mimic what is done in real data in each simulated dataset, we used thresholds on the median count and the MAD value to pre-filter low-count and low-variable genes, respectively, keeping features that yielded median count > 100 and MAD score in the top 50^{th} quantile. We ran FSCseq on each dataset after pre-filtering. Optimal values for tuning parameters $(K^*, \alpha^*, \lambda^*)$ were found by searching candidate values of $K = \{2, \dots, 6\}$; $\alpha = \{0.01, 0.05, 0.10, \dots, 0.50\}$; and $\lambda = \{0.25, 0.50, \dots, 5.00\}$, as described in Section 2.2.2.

n	**LFC**	$\mathbf{p_{DE}}$	ϕ_0	$\bar{K}^*$	$\overline{ARI}$	$\overline{pARI}$	$\overline{TPR}$	$\overline{FPR}$
100	1.0	0.025	0.15	4.05	0.994	0.998	0.703	0.0007
			0.35	3.78	0.926	0.922	0.554	0.0012
			0.50	2.28	0.233	0.196	0.077	0.0003
		0.050	0.15	4.02	0.998	1.000	0.685	0.0006
			0.35	4.01	0.994	0.994	0.608	0.0015
			0.50	3.80	0.941	0.923	0.432	0.0013
	2.0	0.025	0.15	4.06	0.996	0.999	0.780	0.0004
			0.35	4.01	0.999	1.000	0.757	0.0004
			0.50	4.02	0.998	1.000	0.726	0.0014
		0.050	0.15	4.05	0.995	0.999	0.781	0.0012
			0.35	4.04	0.997	0.999	0.747	0.0010
			0.50	4.01	0.999	1.000	0.725	0.0026
200	1.0	0.025	0.15	4.17	0.988	0.993	0.741	0.0014
			0.35	4.07	0.989	0.992	0.700	0.0018
			0.50	3.97	0.975	0.974	0.645	0.0012
		0.050	0.15	4.15	0.992	0.996	0.727	0.0017
			0.35	4.04	0.997	0.997	0.700	0.0034
			0.50	4.00	1.000	1.000	0.662	0.0014
	2.0	0.025	0.15	4.11	0.992	0.999	0.803	0.0019
			0.35	4.07	0.998	1.000	0.798	0.0004
			0.50	4.09	0.995	0.999	0.782	0.0007
		0.050	0.15	4.05	0.996	0.999	0.809	0.0019
			0.35	4.04	0.997	0.999	0.778	0.0007
			0.50	4.04	0.998	0.998	0.767	0.0005

Table 1: Results of FSCseq clustering, feature selection, and prediction on subset of combinations of simulation conditions with $K_{true} = 4$ true number of clusters and $\beta_0 = 12$ baseline $\log_2$ *mean*. Clustering is measured by average optimal order ($\bar{K}^*$) and adjusted rand index ($\overline{ARI}$). Prediction is done on an independently simulated dataset with $n_{pred} = 25$ samples, and is measured by the average ARI between predicted and simulated cluster labels, and is denoted $\overline{pARI}$. Feature selection performance is measured by mean true positive rate ($\overline{TPR}$) and false positive rate ($\overline{FPR}$) of cluster-discriminatory gene discovery.

In this section, we evaluate the clustering, feature selection, and prediction performance of FSCseq under a subset of simulated conditions that most closely reflected the estimates from the TCGA data. Thus, this subset represents conditions that are most similar to what one may expect from real data.

Accuracy of derived clustering results was measured by two metrics: optimal order obtained (K^*) and concordance (or agreement) in cluster assignment with the truth, measured by the Adjusted Rand Index (ARI). Feature selection performance was measured by the true positive rate (TPR) and false positive rate (FPR) of discovering true simulated cluster-discriminatory genes. Specifically, TPR is the proportion of true simulated cluster-discriminatory genes ($\{j : j \in G_d\}$) that were correctly determined to be discriminatory by FSCseq, while FPR is the proportion of true simulated nondiscriminatory genes ($\{j : j \notin G_d\}$) that were incorrectly determined to be discriminatory. To assess the prediction performance of FSCseq in new data, we also independently simulated a prediction set of $n_{pred} = 25$ samples for each simulated dataset, based upon the same set of simulated conditions. We then took the estimates obtained from the converged FSCseq model on the original simulated dataset (training set), and then applied the procedure from Section 2.2.2 to the corresponding test set samples to obtain the predicted cluster labels for these samples. We again measured concordance of predicted clusters with the true simulated cluster labels in the test set by ARI, and we denote this "prediction" ARI, or $pARI$. In Table 1, all performance metrics are averaged across the 100 simulated datasets, with each row denoting a unique combination of conditions. The averaged value is denoted by bar notation, e.g. $\bar{K}^*$ is the average K^* across the 100 datasets corresponding to a particular set of simulated conditions.

We found that the clustering performance of FSCseq was generally robust to the magnitude of ϕ_0, even when n is small (Table 1). One exception was found when $n = 100$, $p_{DE} = 0.025$ and $LFC = 1$ with $K_{true} = 4$, i.e. when both the proportion of discriminatory genes and the LFC across these discriminatory genes were small. In such a case, the clusters may be poorly separated and clustering results may be confounded by higher ϕ_0, compounded by the small average per-cluster sample size (25 per cluster). We note that none of the compared methods perform well in this case, although FSCseq yields higher mean ARI.

As n, LFC, or p_{DE} is increased, we also found that that the sample clusters become more distinct from one another, yielding better clustering results. Increasing n and LFC also tended to result in an increase in TPR, reflecting the increasing sensitvity of our variable selection method under larger sample size and fold change differences between clusters. However as ϕ_0 increased, we found a decrease in the TPR, where the higher overdispersion confounded the true differences in expression of the cluster-discriminatory genes and reduced our sensitivity to select cluster discriminatory genes. Interestingly, the FPR did not vary significantly across changes in these conditions, showing the reliability of cluster-discriminatory features discovered by FSCseq, and the FPR was relatively low in general. Figure 2 shows a scatterplot of TPR vs FPR from each individual simulated dataset from Table 1, stratified by simulated LFC and n. In addition to showing that larger n or LFC and a smaller value of ϕ_0 yielded higher TPR, we also show that selecting the correct order was important in attaining both high TPR and low FPR. This is because FSCseq more accurately selected cluster-discriminatory genes if the correct number of groups is identified.

FSCseq's prediction performance was also found to be highly dependent on its initial clustering performance during model training, i.e., $pARI$ on simulated test subjects was correlated to ARI during FSCseq training. This reflects the fact that FSCseq models that correctly clustered samples also tended to yield more accurate parameter estimates, which allowed for more accurate prediction of cluster identity in new samples. Since FSCseq generally yielded high ARI, we also generally found high average $pARI$ for simulated conditions, reflecting the accuracy of predictions via FSCseq. Overall, we found favorable performance of FSCseq over a variable series of conditions.

We compared performance of FSCseq with 7 competing methods: `iCluster+` (iCl), average-linkage hierarchical clustering (HC), K-medoids (KM), `NB.MClust` (NBMB), and `mclust` on log, variance stabilizing, and rlog transformations of the normalized data (lMC, vMC, and rMC). We note that KM was also run on log-transformed data (Jaskowiak et al., 2018) as in lMC. Because there was no default method of choosing K^* for HC, we chose to utilize the gap statistic, as proposed by Tibshirani et al. (2001), and implemented the order selection in this setting via the `NbClust` R package (Charrad et al., 2014). For KM, we selected K^* to be the value of K that maximized the average silhouette width (Reynolds et al., 2004). For NBMB and the transformed MC methods, we used default selection procedures for K^* directly available from the respective `NB.MClust` and `mclust` R packages.

For `iCluster+`, we used the following automated procedure for selecting K^*. The `iCluster+` paper and manual suggest searching graphically for a plateau of the deviance ratio (% Variability Explained) as the optimal number of clusters. Mo et al. (2013) elaborates in the manual of the `iClusterPlus` package that for increased noise in the dataset, the deviance ratio will continue to increase with higher order. We observed this pattern even at very low levels of noise, causing the highest deviance ratio to often be at the maximum number of clusters. In searching extensive sets of simulations, it becomes infeasible to heuristically or visually search for optimal parameters. Thus, to systemize the procedure across numerous simulation

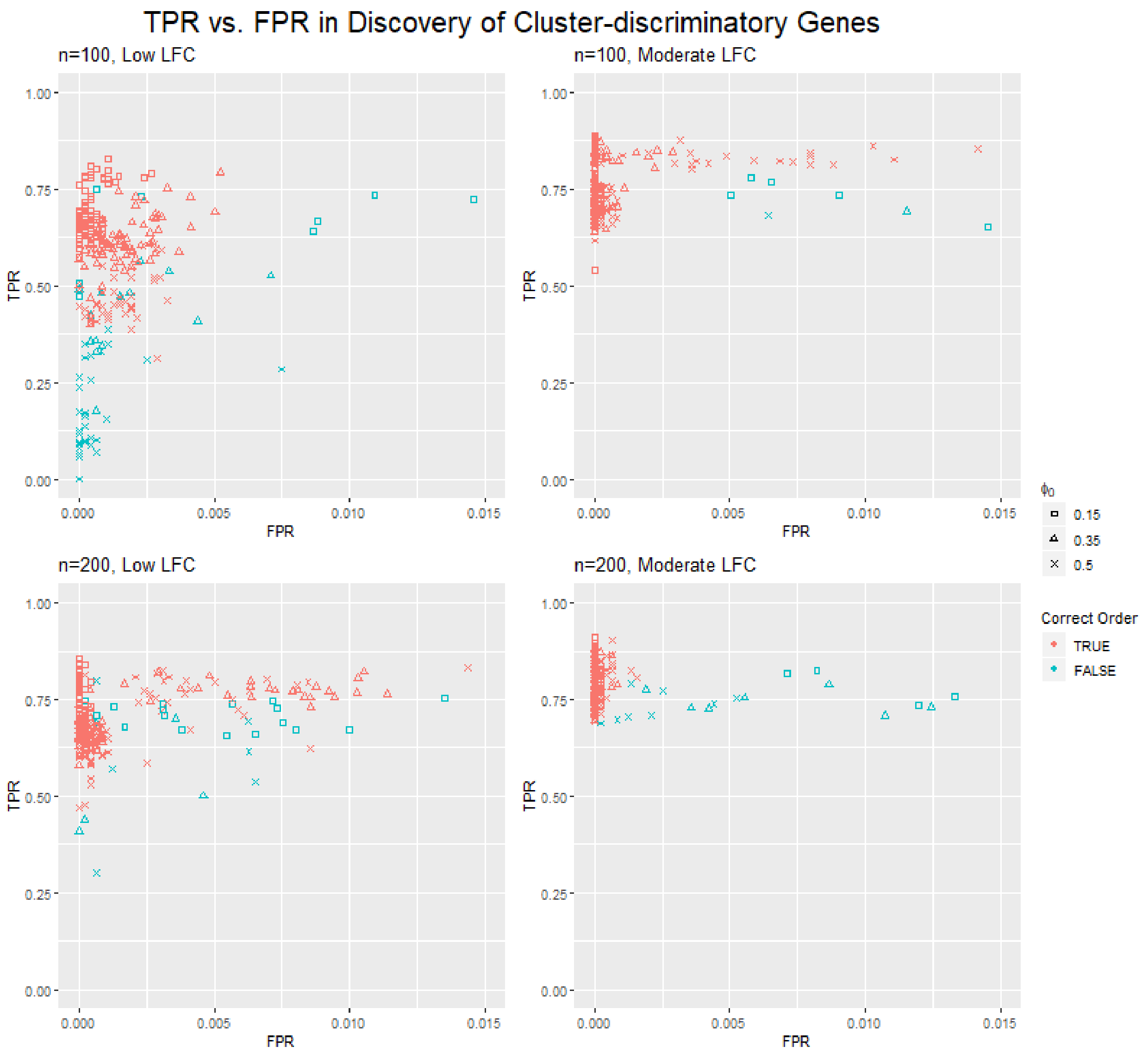


Figure 2: Scatterplot of true positive rate (TPR) vs false positive rate (FPR) in discovering cluster-discriminatory genes in simulated datasets via FSCseq. Displayed points correspond to simulated datasets with $K_{true} = 4$ underlying clusters, with $n = 100$ (top) or 200 (bottom) and simulated $LFC = 1$ ('Low', left) or 2 ('Moderate', right). Red points indicate that the correct order was uncovered ($K^* = K_{true}$), and blue points indicate that an incorrect order was uncovered. Squares, circles, and X's indicate low ($\phi_0 = 0.15$), moderate ($\phi_0 = 0.35$), and high ($\phi_0 = 0.50$) levels of overdispersion, respectively. In general, TPR is higher for larger n and larger LFC. TPR is higher and FPR is lower for results that yielded the correct order, or for smaller values of ϕ_0. For ease of visualization, a total of 15 outlier points (of 2400 total) were removed in this figure with $0.015 < FPR < 0.060$.

cases, we selected an arbitrary threshold based on the manual, such that if the percent increase in variability explained is less than 0.05 with an added cluster, the optimal K^* is selected as the immediately previous value of K.

Variable selection was not compared across these methods as only FSCseq had an automated procedure for feature (gene) selection. We note that `iCluster+` does perform automatic gene selection via the L1 penalty; however, the `iCluster+` manual recommends thresholding an arbitrary proportion of genes for selection rather than utilizing its automatic feature selection process (Mo and Shen, 2019). One could similarly perform thresholding on the estimated coefficients from the `mclust` or `NB.MClust` results, but this would also depend on an arbitrary threshold.

We compared performance of these methods under different numbers of underlying groups $K_{true} = (2, 4)$ with 25 samples per cluster. We also varied ϕ_0 to test each method's robustness to the level of overdispersion. Here, we define order accuracy (OA) as the proportion of simulated datasets (within a given simulated condition) that yielded the correct order, such that $K^* = K_{true}$.

Figure 3 shows violin plots of order accuracy OA and ARI. Here, we fixed $LFC = 1$, $p_{DE} = 0.05$, $n/K_{true} = 25$, and $\beta_0 = 12$, and we varied $\phi_0 = \{0.15, 0.35, 0.50\}$. Generally, FSCseq yielded clusters that had the highest concordance with the simulated clusters. Many of the methods performed competitively in clustering when $K_{true} = 2$, but FSCseq and `iCluster+` performed markedly better than competing methods when $K_{true} = 4$. Throughout all conditions, `iCluster+` tended to select a larger optimal order $K^* > K_{true}$, which caused performance to be significantly better under $K_{true} = 4$ than $K_{true} = 2$. KM performed sporadically throughout, suggesting lack of robustness in clustering performance to the magnitude of ϕ_0. Generally, NBMB yielded the lowest ARI compared to all other methods. We postulate that this is because NBMB utilizes cluster-specific dispersion parameters for each gene in their model, which may yield unstable estimates due to smaller sample sizes in estimating each cluster-level dispersion parameter.

In general, we observed poorer clustering performance of most methods in $K_{true} = 4$ compared to $K_{true} = 2$, with markedly lower average ARI for HC and the transformed MC runs (Figure 3). When K_{true} is larger, there is a smaller proportion of samples within each cluster-discriminatory gene with $\log_2$ mean different from β_0, since only one cluster is differentially regulated by LFC for these genes. Thus, we generally observe a significant decrease in performance for larger K_{true}, although FSCseq's performance is most robust to this effect. In addition, similarly sporadic performance of HC in RNA-seq was also observed previously (Vidman et al., 2019), suggesting unreliability of HC for clustering RNA-seq gene expression.

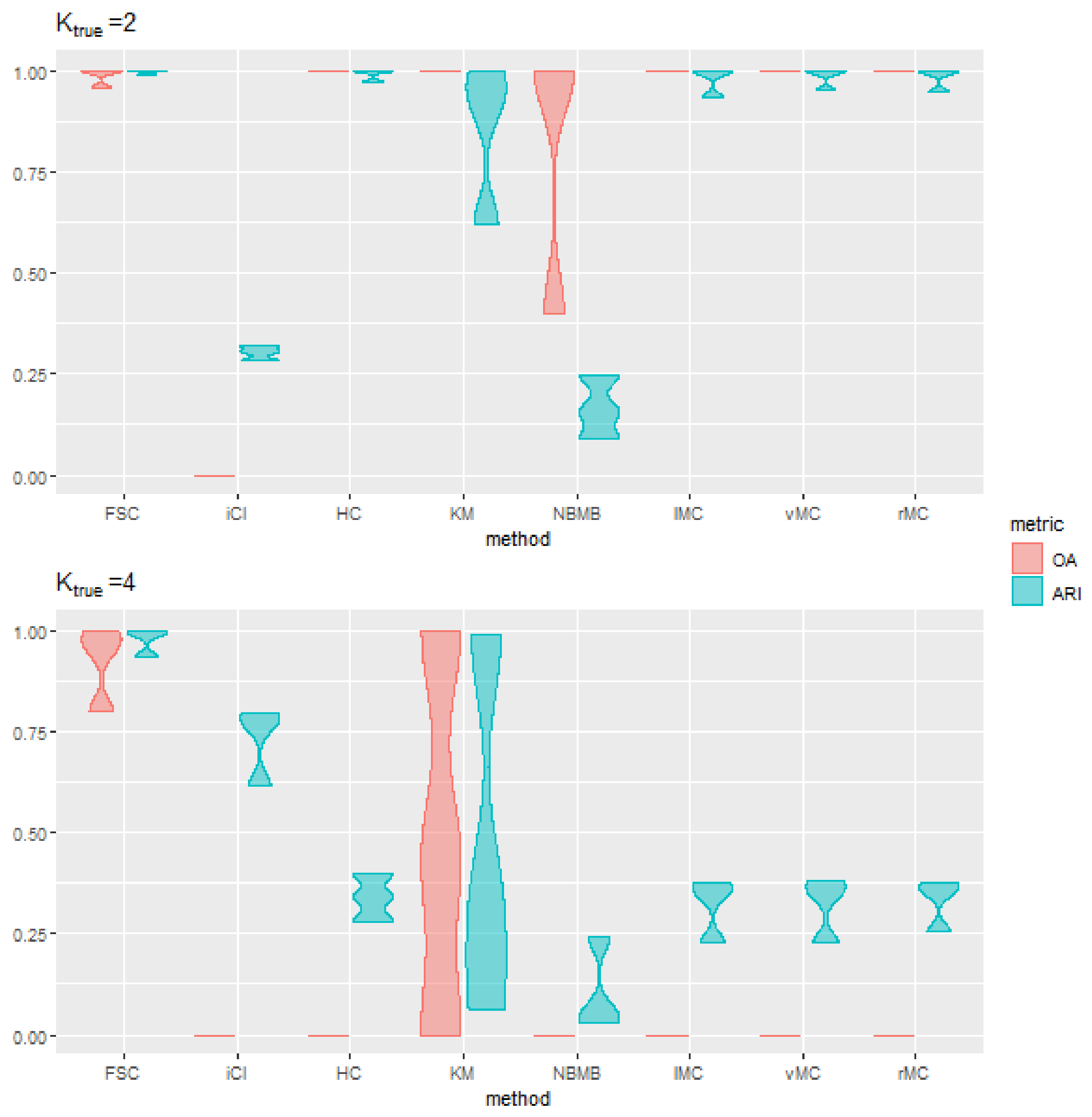


Figure 3: Violin plots of Order Accuracy (OA, in red) and average cluster concordance with the truth (ARI, in blue) from simulation results with $K_{true} = 2$ and $K_{true} = 4$. We compared performance of FSCseq (FSC), `iCluster+` (iCl), hierarchical clustering (HC), K-medoids (KM), `NB.MClust` (NBMB), and `mclust` on log/variance-stabilizing/rlog transformed data (lMC/vMC/rMC). Simulated overdispersion was varied with $\phi_0 = (0.15, 0.35, 0.50)$ to test each method's robustness to the magnitude of overdispersion. Other simulation parameters were fixed at $LFC = 1$, $p_{DE} = 0.05$, $\beta_0 = 12$, and $n/K_{true} = 25$ for both $K_{true} = 2$ (top) and $K_{true} = 4$ (bottom). We fix n/K_{true} here to show the effect of varying K_{true} on performance, with a fixed number of samples per cluster. Many methods perform competitively when $K_{true} = 2$, but FSCseq attains the highest average ARI overall, and yields high performance that is very robust to the magnitude of ϕ_0.

Finally, although the MC methods performed similarly on average, the best performing method between these three transformations varied for each set of simulation conditions, reflecting the fact that the optimal transformation may not be known in advance.

	γ_0	$\bar{K}_2^*$	$\overline{ARI}_2$	$\bar{K}_4^*$	$\overline{ARI}_4$
FSC	2	2.00	1.000	4.16	0.971
	3	2.00	1.000	4.00	0.965
iCl	2	4.68	0.450	5.00	0.779
	3	3.00	0.287	4.00	0.332
HC	2	2.00	0.000	2.00	0.001
	3	2.00	-0.002	2.00	-0.001
KM	2	2.00	0.989	2.00	0.002
	3	2.00	-0.002	2.00	-0.001
NBMB	2	2.28	0.055	2.32	0.004
	3	2.32	0.043	2.04	-0.001
lMC	2	3.92	0.498	2.00	0.001
	3	4.00	0.499	2.00	-0.001
vMC	2	3.92	0.498	2.00	0.001
	3	4.00	0.499	2.00	-0.001
rMC	2	2.04	0.024	2.00	0.001
	3	3.36	0.368	2.00	-0.001

Table 2: Average obtained order $\bar{K}^*$ and average ARI for competing methods with $K_{true} = 2$ and $K_{true} = 4$ underlying groups, in the presence of simulated batch effects γ_0. We shorten notation in this table by denoting the $\bar{K}^*$ value as $\bar{K}_2^*$ for $K_{true} = 2$, and as $\bar{K}_4^*$ for $K_{true} = 4$. Similarly, we denote ARI as ARI_2 for $K_{true} = 2$, and as ARI_4 for $K_{true} = 4$. Tabulated results are from datasets with $n/K_{true} = 50$ simulated samples per cluster, with fixed $LFC = 2$, $p_{DE} = 0.05$, and $\beta_0 = 12$, $\phi_0 = 0.35$. Simulated effect across batches was varied, such that $\gamma_0 = 2, 3$.

We also evaluated the clustering and order selection performance of these methods in the presence of batch effects. Here we simulate two batches in a subset of simulation conditions, where batch effects were imposed on a subset of genes for each simulated dataset. Simulated RNA-seq read counts were generated in a manner similar to (2.12), except now we assume $\log_2(\mu_{ijk}) = s_i + \beta_{jk} + \sigma_{ij} + \{I(j \in G_b) \cdot \gamma_0 \cdot Batch_i\}$, where G_b is the set of genes that are simulated to be batch-affected, γ_0 is a fixed batch effect on the $\log_2$ scale, and $Batch_i = \{-0.5, 0.5\}$. We randomly assign $Batch_i = -0.5$ or $Batch_i = 0.5$ for $i = 1, \ldots n$ with equal probability, and G_b is comprised of a randomly selected 50% subset of the G genes. For each simulation we performed FSCseq with the batch variable as a covariate, and compared performance via each method on simulation conditions with $n/K_{true} = 50$, $LFC = 2$, $p_{DE} = 0.05$, $\beta_0 = 12$, and $\phi_0 = 0.35$. We chose this subset to isolate the confounding effect of batch, since these simulated conditions yielded good clustering performance when batch effects were not simulated.

Results on batch-simulated datasets are shown in Table 2. FSCseq is the only method that performs robustly to the magnitude of γ_0, showing its ability to properly adjust for the confounding batch effects.

Although KM does not perform any correction for batch, it surprisingly performed well under $K_{true} = 2$ and $\gamma_0 = 2$. However, because KM does not actually correct for batch, KM performs expectedly poorly under larger simulated batch effects $\gamma_0 = 3$.

2.3.2 Application to TCGA Breast Cancer RNA-seq dataset

In this section, we compared performance of these clustering methods on the TCGA Breast Cancer (BRCA) dataset. The dataset and annotations were obtained from the National Cancer Institute GDC Portal (Grossman et al., 2016). Gene expression RNA-seq data was sequenced via the Illumina Hiseq 2000 platform, and alignment and quantification were done using the GDC's mRNA Analysis Pipeline, which can be found in the "GDC Data User's Guide" online (Grossman et al., 2016). The raw read counts were downloaded using the `TCGAbiolinks` R package (Colaprico et al., 2015).

Subtyping was done previously using the PAM50 classifier: a set of 50 genes that has been heavily investigated as driving genes of breast cancer subtypes (Koboldt et al., 2012). These genes are also known to be primarily expressed in tumor cells. However, tumor bulk samples are typically comprised of heterogeneous mixtures of cell types, and may poorly reflect the expression profile of the tumor if the sample is of low tumor purity, i.e. if it is composed of a small proportion of tumor cells. Attempts at unsupervised clustering of low-purity samples based on subtype may emphasize genes that are associated with purity and confound results (Aran et al., 2015). Thus, we first included in our analysis only samples whose estimated ABSOLUTE purity (Carter et al., 2012) was greater than 0.9. This is done in order to (1) properly cluster samples via each compared method based upon tumor subtypes, and (2) compare each clustering method fairly with the annotated subtypes, which are based upon tumor-intrinsic PAM50 classifier genes. We also performed our analyses on all samples without purity filtering, and compared our results with those from Koboldt et al. (2012), which performed similar clustering analyses on all samples.

After filtering samples by purity, we found very low incidence of the HER2-enriched and normal-like subtypes, and thus removed these subtypes from our analysis. Then, we pre-filtered low read count genes by low median normalized count, and low-variable genes by low MAD scores. After all filtering steps, the dataset contained 123 samples from 3 distinct subtypes, and 4038 genes. We measured concordance with annotated sample groupings (tumor subtypes in the TCGA BRCA dataset) by four clustering metrics: ARI, normalized mutual information (NMI) (Strehl and Ghosh, 2003), normalized variation of information (NVI) (Reichart and Rappoport, 2009), and normalized information distance (NID) (Vinh et al., 2010).

The RNA-seq experiments were performed on separate wells of multiwell plates, with multiple samples sequenced on the same plate at different times. Therefore, samples across plates may express variability similar to batch effects (Reese et al., 2013). We performed FSCseq analysis with (FSC_{adj}) and without (FSC) adjustment for this confounding effect, and compared the results with competing methods. For proper analysis via FSC_{adj}, we agglomerated singleton plates (plates with just one sample) into one joint plate.

For this data, we expanded the grid of candidate values of K to $\{2, \ldots, 8\}$. All clustering metrics were calculated with respect to the mRNA PAM50 subtype annotations (‘$anno$’) by the GDC. Potential confounders like age, ethnicity, and tumor grade were not included here because they were not of major interest in our analysis, but they can be incorporated as additional covariates in FSCseq. Results are given in Table 3.

	K^* (3)	ARI (1)	NMI (1)	NVI (0)	NID (0)
anno	3				
FSC	5	0.375	0.422	0.667	0.578
FSC_{adj}	3	0.594	0.642	0.523	0.358
iCl	8	0.204	0.32	0.737	0.68
HC	2	0.481	0.476	0.559	0.524
KM	2	0.486	0.489	0.539	0.511
NBMB	4	0.447	0.461	0.658	0.539
lMC	4	0.437	0.464	0.649	0.536
vMC	4	0.428	0.465	0.65	0.535
rMC	4	0.436	0.464	0.649	0.536

Table 3: Selected order (K^*) and clustering concordance between compared methods and annotated TCGA Breast Cancer subtypes. FSCseq was run with adjustment (FSC_{adj}) and without adjustment (FSC) for plate effect, and each of the clustering labels were compared to annotated subtypes (anno). For each column, the value of the best performing metric is colored in red. The values in parentheses in the column headings represent the optimal value for that metric. For ARI and NMI, values closer to 1 indicate better clustering, and values closer to 0 indicate worse clustering. For NVI and NID, values closer to 0 indicate better clustering, and values closer to 1 indicate worse clustering.

FSC selected order $K^* = 5$, but after adjusting for plate effects, FSC_{adj} uncovered the true order of $K^* = 3$. Similar to the trends seen in simulations, `iCluster+` selected a larger number of clusters $K^* = 8$, while HC and KM selected a smaller $K^* = 2$. The transformed MC methods and NBMB selected a slightly larger $K^* = 4$. Of the competing methods, FSC_{adj} yielded the clusters with the highest concordance with the annotations, with an ARI of 0.594, followed by KM and HC with ARIs of 0.486 and 0.481, respectively. FSC_{adj} additionally yielded the best NMI, NVI, and NID.

Feature selection via FSC and FSC_{adj} respectively determined a total of 2693 and 2238 genes to be cluster-discriminatory, while the automatic feature selection by the L1 penalty in `iCluster+` determined significantly more genes (4035) to be cluster-discriminatory. Of the 41 PAM50 genes that were included in our analysis after pre-filtering, FSC determined all 41 likely to be cluster-discriminatory, and FSC_{adj} determined 38 of them to be likely. This shows FSCseq's ability to identify subtype-discriminating genes in real cancer data, as the relevance of PAM50 genes in RNA-seq has been shown by previous studies (Picornell et al., 2019; Raj-Kumar et al., 2019). Figure 4 shows a heatmap of the PAM50 genes with notations on their inclusion/exclusion through the pre-filtering step, and through FSCseq's simultaneous feature selection. Column (samples) ordering is based on annotated subtypes, and samples are ordered within subtypes by decreasing order of maximum posterior probability from FSC_{adj} results. KM performs well in terms of ARI, but it only grouped samples by basal vs. non-basal subtypes and didn't distinguish between the Luminal A and Luminal B subtypes. Similarly, lMC performs well in grouping basal samples, but overselected the order ($K^* = 4$) and was not able to distinguish between Luminal A and B samples. Overall, the cluster labels from FSC_{adj} most accurately clustered the samples according to the 3 underlying subtypes.

In addition to the PAM50 genes, other significant cluster-discriminatory genes found by both FSC and FSC_{adj} included genes like CDH1, which has been linked to lobular or ER+ tumor carcinomas (Yang et al., 2015), and CHEK2, which has been studied for its association with the luminal subtypes (Huszno and Kolosza, 2019). Additionally, we performed gene ontology analysis on the discovered sets of cluster-discriminatory genes from both FSC and FSC_{adj}. Enriched pathway analysis showed that genes from both sets were involved in known key pathways associated with cancer. Such pathways included signaling pathways like EIF2 and AHR, as well as pathways involved in mitosis and other molecular mechanisms of cancer. Both sets of genes also contained many gene ontology (GO) biological processes pertaining to the cell cycle, consistent with previous studies that found such enrichment in basal-like subtypes of breast cancer (Yang et al., 2017, 2019). We further validated our results by testing for overlaps with known gene sets via GSEA analysis (Mootha et al., 2003; Subramanian et al., 2005). We first grouped cluster-discriminatory genes from FSC_{adj} using $MBCluster.Seq$ (Si et al., 2013), then separately analyzed the subset of genes that were upregulated ($basalUP$), and those that were downregulated ($basalDOWN$) for the most distinct basal subtype. Significantly overlapping gene sets with $basalUP$ and $basalDOWN$ were sets of genes known to differentiate subtypes of breast cancer. Additionally, the top overlapping gene set for $basalUP$ and $basalDOWN$ corresponded to the specific collections of genes known to be upregulated and downregulated, respectively, in the basal subtype of breast cancer (Smid et al., 2008). Results of these gene ontology analyses are given in Figures 5 and 6.

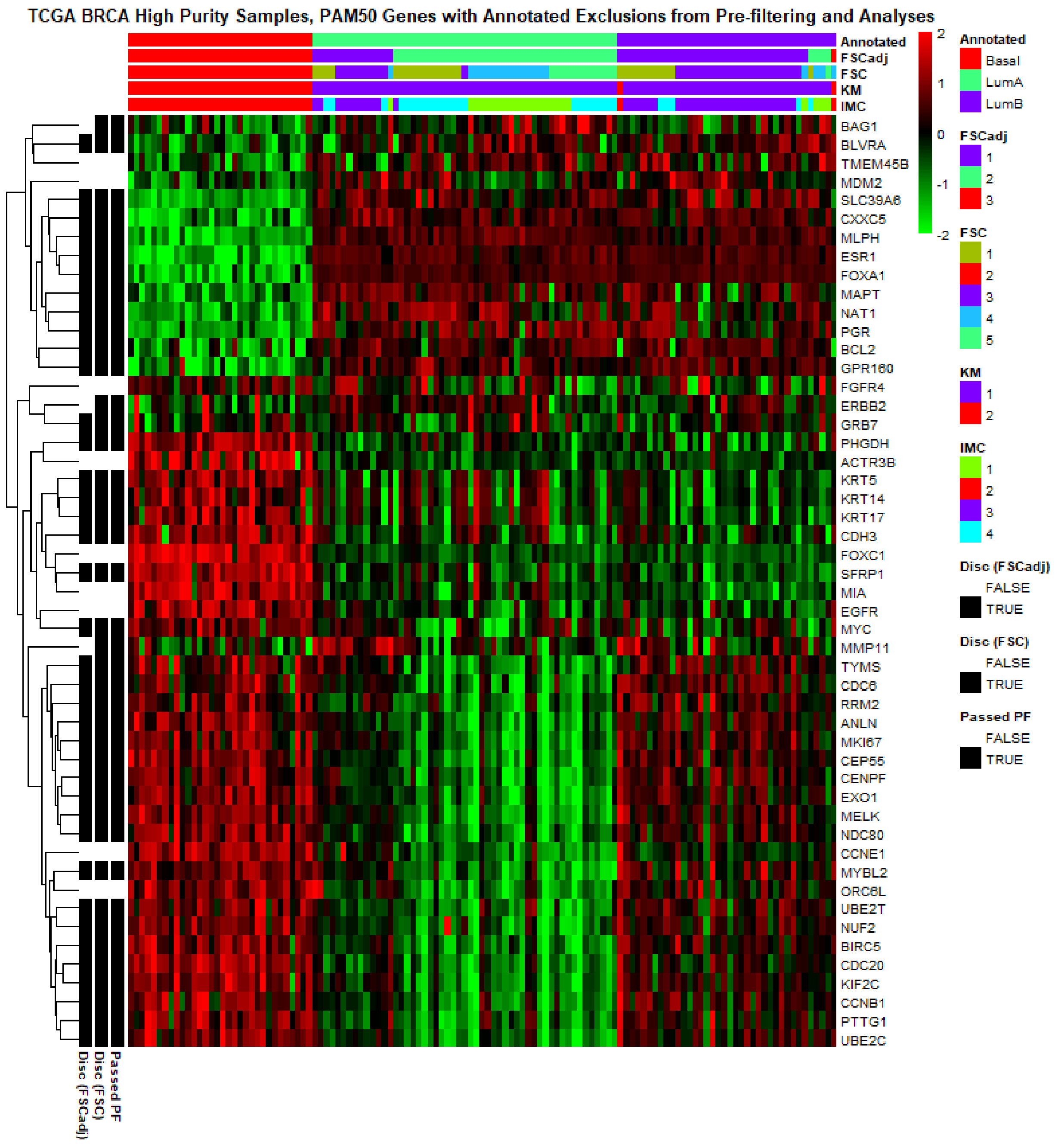


Figure 4: Heatmap of the PAM50 genes included in FSCseq analyses, with row annotations for feature selection and pre-filtering (left) and column annotations for clustering labels (top). Column ordering is based on annotated subtypes, and samples are ordered within subtypes by decreasing order of maximum posterior probability from FSC_{adj} results. 9 PAM50 genes did not pass the pre-filtering (PF) threshold: TMEM45B, MDM2, FGFR4, ACTR3B, FOXC1, MIA, EGFR, CCNE1, and ORC6L. Of the 41 remaining PAM50 genes, FSC and FSC_{adj} found 41 and 38 of the PAM50 genes were cluster-discriminatory, respectively. Additionally, all clustering labels distinguish well between Basal and Luminal subtypes, but FSC_{adj} best distinguishes between Luminal A and Luminal B samples.

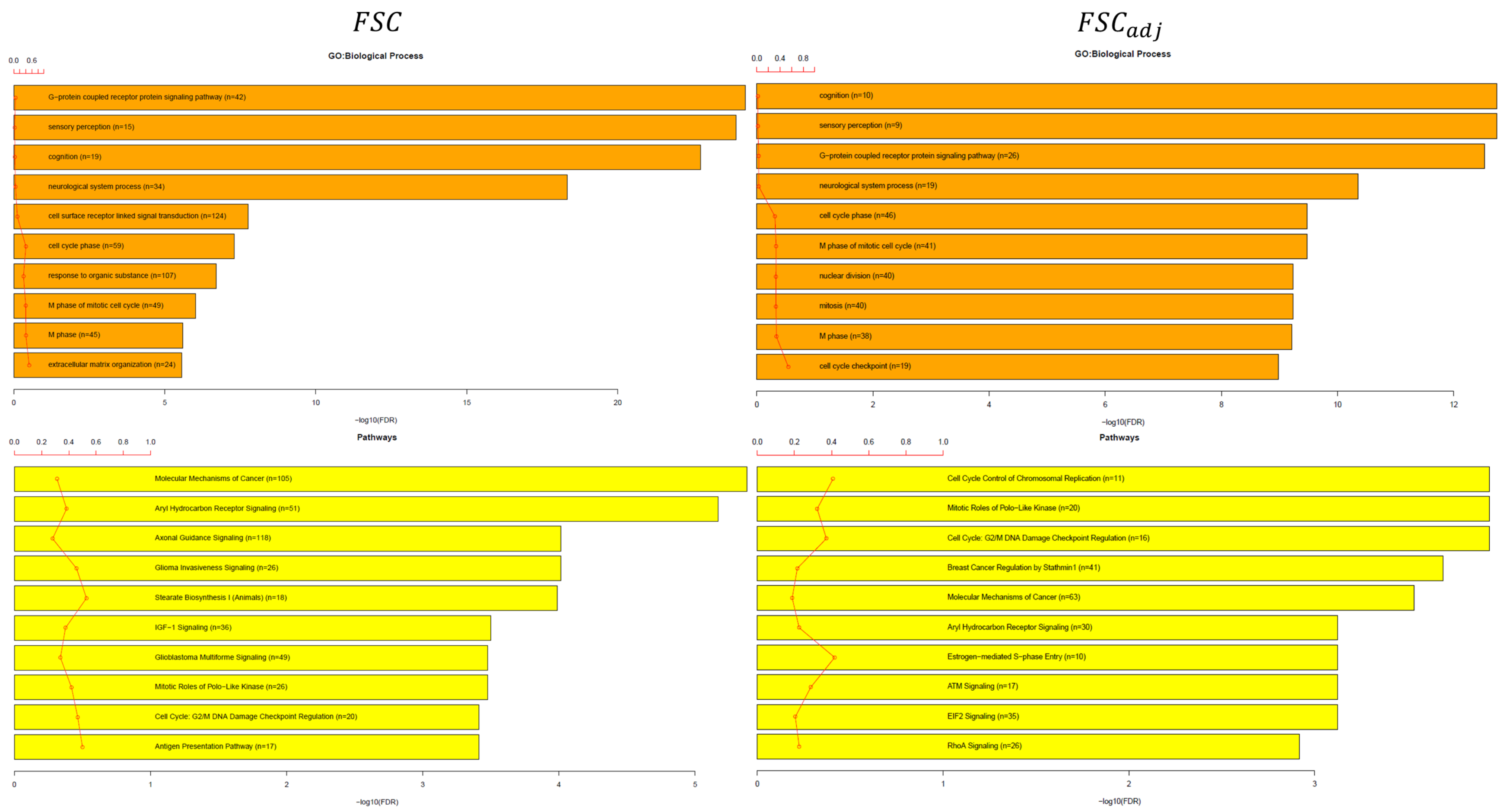


Figure 5: Gene ontology biological processes and enriched pathways of cluster discriminating genes found by FSC (left) and FSC_{adj} (right). The bars represent p-value corrected FDR (in $-\log$ scale), and the red lines represent ratios of list genes found in each pathway over the total number of genes in that pathway.

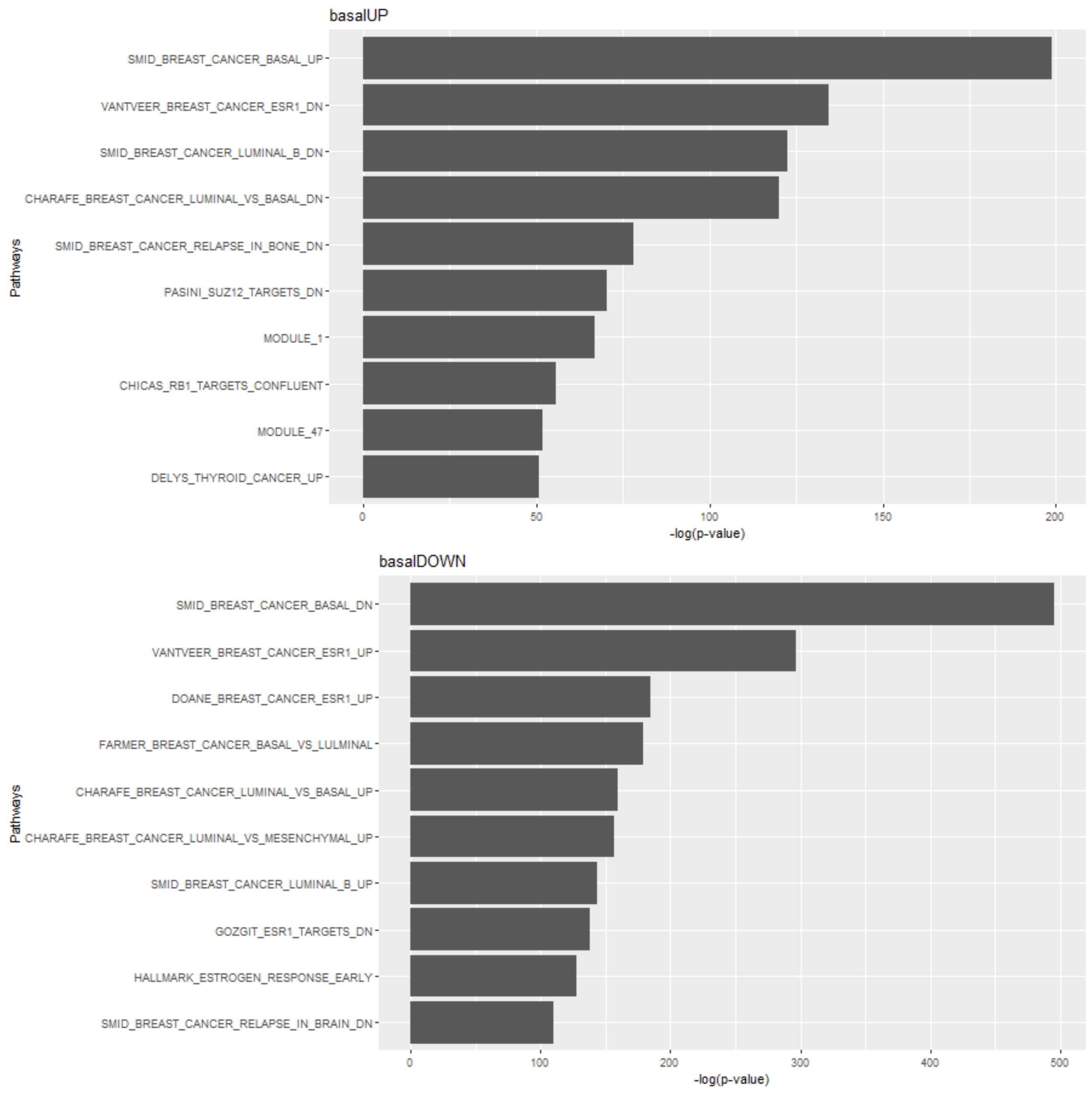


Figure 6: GSEA analyses on $basalUP$ (basal up-regulated, top) and $basalDOWN$ (basal down-regulated, bottom) subsets of cluster-discriminatory genes discovered from FSC_{adj}. Only the top five overlapping gene sets shown.

We measured performance in prediction via leave-one-out cross-validation by training a model on cluster-discriminatory genes from FSC with one sample held out, and using this model to predict the cluster label of the held-out sample. In this way, prediction was performed on each sample once. We observed very high overall accuracy of 0.951, showing robustness of our prediction framework in real data settings.

Finally, we also clustered the TCGA samples without pre-filtering samples for purity and including all 5 original subtypes ($K_{true} = 5$), as done previously by Koboldt et al. (2012). We anticipated that there would exist a larger number of underlying groups in this dataset due to heterogeneity caused by low purity samples, which is also reflected by the results in Koboldt et al. (2012). Thus, we expanded the search range of K to $K = \{2, \ldots, 15\}$. As expected, all methods yielded clusters of poor concordance with the annotated PAM50 subtypes, with FSC and FSC_{adj} yielding larger orders of $K^* = 7$ and $K^* = 8$, respectively, and ARIs of 0.316 and 0.245, respectively. Koboldt et al. (2012) performed unsupervised clustering on the same samples and found 13 clusters, and they performed semi-supervised clustering with an intrinsic list of significant genes that similarly yielded 14 clusters. Their results were similarly poor in agreement with the annotated subtypes (ARI of 0.272 and 0.258 for unsupervised and semi-supervised clusters, respectively). Compared to these results, FSCseq was able to select K^* that is closer to the true number of subtypes, however the potential confounding in expression due to variable sample purity appears to cloud the ARI performance of all methods in this setting. The results from these analyses can be found in Table 4

	K^* (5)	ARI (1)	NMI (1)	NVI (0)	NID (0)
anno	5				
FSC	7	0.316	0.369	0.732	0.631
FSC_{adj}	8	0.245	0.295	0.78	0.705
iCl	6	0.257	0.28	0.805	0.72
HC	2	0.004	0.003	0.997	0.997
KM	2	0.348	0.293	0.731	0.707
NBMB	15	-0.008	0.097	0.932	0.903
lMC	15	0.157	0.259	0.788	0.741
vMC	12	0.164	0.268	0.786	0.732
rMC	12	0.169	0.274	0.781	0.726
SigI	14	0.258	0.333	0.73	0.667
SigU	13	0.272	0.35	0.736	0.65

Table 4: Selected order (K^*) and clustering concordance between compared methods and annotated TCGA Breast Cancer subtypes. FSCseq was run with adjustment (FSC_{adj}) and without adjustment (FSC) for plate effect, and each of the clustering labels were compared to annotated subtypes (anno). Results from unsupervised (SigU) and semi-supervised (SigI) clustering from Koboldt et al. (2012) are also shown. For each column, the best performing metric is colored in red. The value in parentheses in the column headings represent optimal values. For ARI and NMI, values closer to 1 indicate better clustering, and values closer to 0 indicate worse clustering. For NVI and NID, values closer to 0 indicate better clustering, and values closer to 1 indicate worse clustering.

2.4 Discussion

Our findings from our simulations and real data applications give evidence to the utility of our method across a varied set of conditions. In our simulations, we found very good feature selection performance throughout simulated conditions, and FSCseq outperformed existing clustering methods for RNA-seq data. In addition, the markedly low FPRs in gene discovery may help researchers attain a higher degree of confidence in the discovered list of genes, while limiting expended resources on validation studies in clinical settings. In the TCGA BRCA dataset, FSCseq clusters aligned best to previously discovered subtypes when correcting for batch effects (plate), although differing levels of heterogeneity in samples confounded the results. The annotated subtypes were discovered by analyzing just the genes that were clinically validated to be significant in discriminating across breast cancer subtypes. We showed that FSCseq is able to perform comparably, despite using no such *a priori* knowledge of significant genes. Moreover, of the PAM50 genes included in our TCGA BRCA analysis, the FSCseq workflow identified most of these genes as discriminatory across the resulting clusters. As these genes have been validated to be significant on both microarray and RNA-seq platforms, this emphasizes FSCseq's ability to uncover significant genes in real data settings.

CHAPTER 3: UNSUPERVISED DEEP LEARNING WITH MISSING DATA

Although there exist some methods to handle up to MAR missingness in the VAE setting, there is a lack of discussion on the principled theory of missingness in conjunction with such methods. Additionally, these methods are not able to handle MNAR missingness, where a parametric model is typically necessary for the missingness. To address these issues, we compare performance of state-of-the-art deep learning methods under different mechanisms of missingness, unifying these deep learning methods with a proper statistical framework of missing data. We then propose an extension of the VAE to handle MNAR data, and show that proper modelling of the missingness increases performance in tasks such as imputation of missing data, and clustering of sub-populations. We demonstrate the utility of our method through imputation and clustering of synthetic datasets with simulated missingness, and downstream prediction analysis on real datasets.

3.1 Methods

In this section, we first show the formulation of variational autoencoders (VAEs). Then, we explore the different mechanisms of missingness as discussed thoroughly by statistical literature (Little and Rubin, 2002). We expand on the special case of MNAR missingness, where the missingness can be dependent on the missing values themselves, and present the models that are typically employed under the MNAR case. Finally, we look at VAEs in the context of the discussed mechanisms of missingness, and propose a novel method using VAEs in the presence of MNAR missingness.

3.1.1 Variational Autoencoder

Let $\mathbf{X}$ be an $n \times p$ data matrix, where $\mathbf{x}_i$ denotes the observation vector pertaining to the i^{th} observation, $i = 1, \ldots, n$, and x_{ij} denotes the value of the j^{th} feature in this vector, $j = 1, \ldots, p$. In a VAE, we assume $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are i.i.d. samples from a multivariate p.d.f or "generative model" $p_\psi(\mathbf{X}|\mathbf{Z})$. Here, $\mathbf{Z}$ is an $n \times d$ matrix, such that $\mathbf{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$ and $\mathbf{z}_i$ is a latent vector of length d pertaining to the i^{th} sample (Kingma and Welling, 2019). Typically it is assumed that $d \leq p$, such that $\mathbf{Z}$ constitutes a lower-dimensional representation of the original data $\mathbf{X}$. The parameters ψ and conditional distribution $p_\psi(\mathbf{X}|\mathbf{Z})$ indicate how the observed data $\mathbf{X}$ may be generated from $\mathbf{Z}$. In this manner, a VAE aims to learn accurate representations of high-dimensional data, and may be used to generate synthetic data with similar qualities as its training data. These aspects are also aided through the use of embedded deep learning neural networks, for example within $p_\psi(\mathbf{X}|\mathbf{Z})$, which also facilitate its applicability to larger dimensions and complex datasets.

3.1.1.1 Objective Function

Since ψ is unknown, learning is performed by maximizing the marginal log-likelihood of $\mathbf{X}$ with respect to ψ, where we denote this marginal log-likelihood as

$$\log p_\psi(\mathbf{X}) = \log \int p_\psi(\mathbf{X}, \mathbf{Z}) d\mathbf{Z} = \log \int p_\psi(\mathbf{X}|\mathbf{Z}) p(\mathbf{Z}) d\mathbf{Z}.$$

However, due to the integral involved, this quantity is often intractable and is difficult to maximize directly. Therefore, VAE's alternatively optimize the so-called "Evidence Lower Bound" (ELBO), which has the following form (Kingma and Welling, 2019):

$$\mathcal{L}^{ELBO}(\theta, \psi) = \mathbb{E}_{\mathbf{Z} \sim q_\theta(\mathbf{Z}|\mathbf{X})} \log \left[\frac{p_\psi(\mathbf{X}|\mathbf{Z}) p(\mathbf{Z})}{q_\theta(\mathbf{Z}|\mathbf{X})} \right] \tag{3.13}$$

$$\hat{\mathcal{L}}_K^{ELBO}(\theta, \psi) = \frac{1}{K} \sum_{k=1}^{K} \log \left[\frac{p_\psi(\mathbf{X}|\tilde{\mathbf{Z}}_k) p(\tilde{\mathbf{Z}}_k)}{q_\theta(\tilde{\mathbf{Z}}_k|\mathbf{X})} \right]. \tag{3.14}$$

Here, $\mathcal{L}^{ELBO}(\theta, \psi)$ denotes the ELBO such that $\mathcal{L}^{ELBO}(\theta, \psi) \leq \log p_\psi(\mathbf{X})$. Also let $\hat{\mathcal{L}}_K^{ELBO}(\theta, \psi)$ denote the empirical approximation to Eq. (3.13) computed by Monte Carlo integration, such that $\mathcal{L}^{ELBO}(\theta, \psi) \approx \hat{\mathcal{L}}_K^{ELBO}(\theta, \psi)$ and $\tilde{\mathbf{Z}}_1, \ldots, \tilde{\mathbf{Z}}_K$ are K samples drawn from $q_\theta(\mathbf{Z}|\mathbf{X})$, the variational approximation of the true but intractable posterior $p_\psi(\mathbf{Z}|\mathbf{X})$, also called the "recognition model". Furthermore, denote $f_\psi(\mathbf{Z})$ and $g_\theta(\mathbf{X})$ as the decoder and encoder feed forward neural networks of the VAE, where ψ and θ are the sets of weights and biases pertaining to each of these neural networks, respectively. Given $\mathbf{Z}$, $f_\psi(\mathbf{Z})$ outputs the distributional parameters pertaining to $p_\psi(\mathbf{X}|\mathbf{Z})$. Given $\mathbf{X}$, $g_\theta(\mathbf{X})$ outputs the distributional parameters for $q_\theta(\mathbf{Z}|\mathbf{X})$.

In variational inference, $q_\theta(\mathbf{Z}|\mathbf{X})$ is constrained to be from a class of simple distributions, or "variational family", to obtain the best candidate from within that class to approximate $p_\psi(\mathbf{Z}|\mathbf{X})$. Variational inference is usually used in tandem with amortization of the parameters where the neural network parameters are shared across observations (Gershman and Goodman, 2014), allowing for stochastic gradient descent (SGD) to be used for optimization of Eq. (3.14) (Kingma and Welling, 2019). In practice, both $q_\theta(\mathbf{Z}|\mathbf{X})$ and $p(\mathbf{Z})$ are typically assumed to have simple forms, such as multivariate Gaussians with diagonal covariance structures, and $q_\theta(\mathbf{Z}|\mathbf{X})$ is commonly assumed to be factorizable, such that $q_\theta(\mathbf{Z}|\mathbf{X}) = \prod_{i=1}^{n} q_\theta(\mathbf{z}_i|\mathbf{x}_i)$ (Kingma and Welling, 2019).

3.1.1.2 Estimation Procedure and Use Cases

Let $(\hat{\theta}^{(t)}, \hat{\psi}^{(t)})$ be the estimates of (θ, ψ) at update (or iteration) t. For $t = 0$, these values are often initialized to small values centered around 0, although other initialization schemes may be used (Saxe et al., 2014; Murphy, 2016). Each subsequent update $t \geq 1$ consists of two general steps to maximize $\mathcal{L}(\theta, \psi)$. First, K samples are drawn from $q_{\hat{\theta}^{(t)}}(\mathbf{Z}|\mathbf{X})$ to compute the quantity in Eq. (3.14), conditional on $(\hat{\theta}^{(t)}, \hat{\psi}^{(t)})$. Then, the so-called "reparametrization trick" is utilized to facilitate the calculation of gradients of this approximation to obtain $(\hat{\theta}^{t+1}, \hat{\psi}^{t+1})$ using stochastic gradient descent (Kingma and Welling, 2019). This procedure may be repeated for a fixed number of iterations, or may be terminated early via pre-specified early stop criteria (Prechelt, 1998). Kingma and Welling (2019) provides additional details on the maximization procedure for VAEs. The networks $f_\psi(\mathbf{Z})$ and $g_\theta(\mathbf{X})$ also allow the VAE to capture complex and non-linear relationships between features when outputting the distributional parameters for the generative and recognition models, respectively, through the inclusion of hidden layers in each network. The number of hidden layers and nodes per layer for each network are commonly determined via hyperparameter tuning.

After model fitting, the VAE has several useful features. First, synthetic data can be generated by sampling from the learned generative model $p_{\hat{\psi}}(\mathbf{X}|\mathbf{Z})$ after drawing a sample from $q_{\hat{\theta}}(\mathbf{Z}|\mathbf{X})$ (Mattei and Frellsen, 2019; Nazabal et al., 2018). Second, the posterior modes in the latent space $\mathbf{Z}$ can be determined from $q_{\hat{\theta}}(\mathbf{Z}|\mathbf{X})$. These posterior modes may be used for purposes such as clustering or substructure discovery (Lim et al., 2020). In some applications, $p_{\hat{\psi}}(\mathbf{X}|\mathbf{Z})$ may also be used to directly perform imputation (Nazabal et al., 2018), however the statistical properties of this procedure have not been thoroughly discussed in prior work.

3.1.2 Importance-Weighted Autoencoder

The IWAE (Burda et al., 2015) is a generalization of the standard VAE, where the resulting IWAE bound, corresponding to the ELBO in Eq. (3.13), can be written as

$$\mathcal{L}_K^{IWAE}(\theta,\psi) = \mathbb{E}_{\mathbf{Z}_k \sim q_\theta(\mathbf{Z}|\mathbf{X})} \log \left[\frac{1}{K} \sum_{k=1}^{K} \frac{p_\psi(\mathbf{X}|\mathbf{Z}_k)p(\mathbf{Z}_k)}{q_\theta(\mathbf{Z}_k|\mathbf{X})} \right] \tag{3.15}$$

$$\hat{\mathcal{L}}_K^{IWAE}(\theta,\psi) = \log \left[\frac{1}{K} \sum_{k=1}^{K} \frac{p_\psi(\mathbf{X}|\tilde{\mathbf{Z}}_k)p(\tilde{\mathbf{Z}}_k)}{q_\theta(\tilde{\mathbf{Z}}_k|\mathbf{X})} \right]. \tag{3.16}$$

An important distinction in Eq. (3.15) from Eq. (3.13) is that a VAE assumes a single latent variable $\mathbf{Z}$ in Eq. (3.13) that is sampled K times in the ELBO approximation from Eq. (3.14). In contrast, an IWAE assumes K i.i.d. latent variables in the expression for its lower bound, where $\mathbf{Z}_1, \ldots, \mathbf{Z}_K \overset{i.i.d}{\sim} q_\theta(\mathbf{Z}|\mathbf{X})$. The contribution of each $\mathbf{Z}_k$ in Eq. (3.15) is weighted by $\frac{p(\mathbf{Z}_k)}{q_\theta(\mathbf{Z}_k|\mathbf{X})}$. Then, to compute its empirical approximation $\hat{\mathcal{L}}_K^{IWAE}$, typically only one sample is drawn for each $\mathbf{Z}_k$ in Eq. (3.16). For $K > 1$, Burda et al. (2015) showed that $\log p(\mathbf{X}) \geq \mathcal{L}_{K+1}^{IWAE} \geq \mathcal{L}_K^{IWAE}$, such that $\mathcal{L}_K^{IWAE} \to \log p(\mathbf{X})$ as $K \to \infty$ if $p_\psi(\mathbf{X},\mathbf{Z})/q_\theta(\mathbf{Z}|\mathbf{X})$ is bounded. Thus, the IWAE bound more closely approximates the true marginal log likelihood when $K > 1$ (Cremer et al., 2017), at the cost of greater computational burden. If $K = 1$, $\mathcal{L}_1^{IWAE} = \mathcal{L}^{VAE}$, and the IWAE corresponds exactly to the standard VAE. In this way, the IWAE can be considered to be part of the VAE family, and we refer to methods that use either VAEs or IWAEs broadly as "VAE methods". A visualization of the workflow for an IWAE (without missing data) can be found in Figure 7.

3.1.3 IWAE Architecture

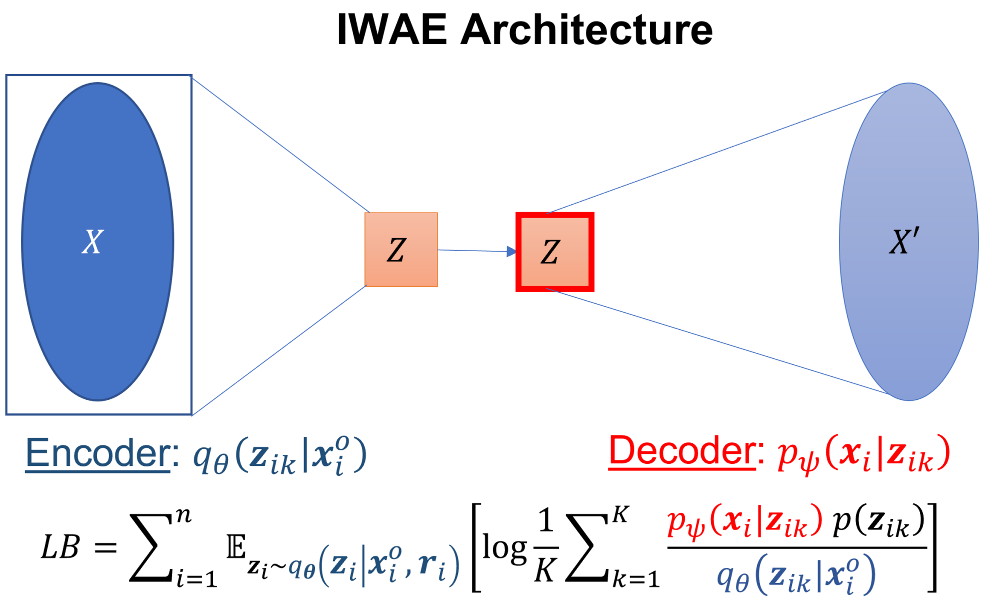

IWAE Architecture
X
Z
Z
X'
Encoder: $q_\theta(\mathbf{z}_{ik}|\mathbf{x}_i^o)$
Decoder: $p_\psi(\mathbf{x}_i|\mathbf{z}_{ik})$
$LB = \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}_i \sim q_\theta(\mathbf{z}_i|\mathbf{x}_i^o, \mathbf{r}_i)} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\psi(\mathbf{x}_i|\mathbf{z}_{ik})\, p(\mathbf{z}_{ik})}{q_\theta(\mathbf{z}_{ik}|\mathbf{x}_i^o)} \right]$

Figure 7: Architecture of an importance weighted autoencoder (IWAE) in the absence of missing data. Darkly colored nodes represent deterministic values, lightly colored nodes represent learned distributional parameters, and outlined (in red) nodes represent sampled values from learned distributions. Orange cells correspond to latent variables $\mathbf{Z}$. $\mathbf{Z}_1, \ldots, \mathbf{Z}_K$ is sampled 1 time each from the variational posterior posterior distribution $q(\mathbf{Z}|\mathbf{X})$. Below is the lower bound (LB), which is optimized via stochastic gradient descent.

VAE methods have shown excellent performance in representation learning on many types of data. However, the presence of missingness in $\mathbf{X}$ presents significant challenges to the above modeling procedures and the application of VAE methods in general.

3.1.4 Missing Data

In this section, we first introduce notation for missing data and review the different mechanisms of missingness, as described in the statistical literature. Let the data be factored such that $\mathbf{X} = \{\mathbf{X}^o, \mathbf{X}^m\}$, with $\mathbf{X}^o$ denoting the observed values and $\mathbf{X}^m$ denoting the missing values. For each observation vector $\mathbf{x}_i$, denote $\mathbf{x}_i^o$ and $\mathbf{x}_i^m$ respectively to be the observed and missing features of $\mathbf{x}_i$. Also, let $\mathbf{R}$ be a matrix of the same dimension as $\mathbf{X}$, with entries $r_{ij} = I(x_{ij}$ is observed) for the i^{th} observation and j^{th} feature, where $I(\cdot)$ denotes the indicator function. In this way, $\mathbf{R}$ is the "mask" matrix pertaining to $\mathbf{X}$, such that $\mathbf{x}_i^o = \{x_{ij} : r_{ij} = 1\}$ and $\mathbf{x}_i^m = \{x_{ij} : r_{ij} = 0\}$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, p$.

Missingness was classified into three major categories, or mechanisms, in the seminal work by Little and Rubin (2002). These mechanisms are missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR), and they satisfy the following relations: (1) MCAR: $p(\mathbf{r}_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\phi}) = p(\mathbf{r}_i|\boldsymbol{\phi})$, (2) MAR: $p(\mathbf{r}_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\phi}) = p(\mathbf{r}_i|\mathbf{x}_i^o, \boldsymbol{\phi})$, and (3) MNAR: $p(\mathbf{r}_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\phi}) = p(\mathbf{r}_i|\mathbf{x}_i^o, \mathbf{x}_i^m, \mathbf{z}_i, \boldsymbol{\phi})$. Here, $\boldsymbol{\phi}$ denotes the unknown parameters pertaining to the missingess model $p(\mathbf{r}_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\phi})$, where $\mathbf{r}_i = \{r_{i1}, \dots, r_{ip}\}$. We discuss forms of this model in Section 3.1.4.3. In the presence of missingness mask $\mathbf{R}$, the marginal log-likelihood may be written as

$$\log p_{\psi,\phi}(\mathbf{X}^o, \mathbf{R}) = \log \iint p_{\psi,\phi}(\mathbf{X}^o, \mathbf{X}^m, \mathbf{Z}, \mathbf{R}) d\mathbf{X}^m d\mathbf{Z}. \tag{3.17}$$

We may factor $p_{\psi,\phi}(\mathbf{X}^o, \mathbf{X}^m, \mathbf{Z}, \mathbf{R})$ using the selection model factorization (Diggle and Kenward, 1994), which is written as

$$p_{\psi,\phi}(\mathbf{X}^o, \mathbf{X}^m, \mathbf{Z}, \mathbf{R}) = p_\psi(\mathbf{X}^o, \mathbf{X}^m|\mathbf{Z}) p(\mathbf{Z}) p(\mathbf{R}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\phi}),$$

where $p(\mathbf{R}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\phi}) = \prod_{i=1}^n p(\mathbf{r}_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\phi})$.

3.1.4.1 Ignorable Missingness

In a likelihood-based analysis under either MCAR or MAR, the missingness mechanism is considered to be "ignorable" such that the missingness mechanism need not be explicitly modelled in these cases (Rubin, 1976; Little and Rubin, 2002). Under ignorable missingness, the left hand side of Eq. (3.17) can be separated into $\log p_\psi(\mathbf{X}^o) + \log p_\phi(\mathbf{R}|\mathbf{X}^o)$, where $p_\psi(\mathbf{X}^o)$ is the marginal distribution of $\mathbf{X}^o$. Therefore, $p_\phi(\mathbf{R}|\mathbf{X}^o)$ need not be specified because inference on the parameters of interest pertaining to $p_\psi(\mathbf{X}^o)$ is independent of $p_\phi(\mathbf{R}|\mathbf{X}^o)$. Then, one aims to maximize the quantity

$$\log p_\psi(\mathbf{X}^o) = \log \iint p_\psi(\mathbf{X}^o, \mathbf{X}^m, \mathbf{Z}) d\mathbf{X}^m d\mathbf{Z} = \log \int p_\psi(\mathbf{X}^o, \mathbf{Z}) d\mathbf{Z}. \tag{3.18}$$

This quantity can be bounded below exactly as in Section 3.1.1.1, conditioning on just the observed data $\mathbf{X}^o$, rather than the full data $\mathbf{X}$. Existing methods typically take advantage of this simplification, and are shown to perform well under ignorable missingness. Details for these methods are given in the next section.

3.1.4.2 VAEs and IWAEs with Ignorable Missingness

There are a number of VAE/IWAE methods that have been developed to handle ignorably missing data.

VAEAC (Ivanov et al., 2019) is a method catered for datasets with complete training sets. VAEAC can handle different types of data (continuous, categorical, or count), and additionally conditions on the missingness mask. In training time, additional MCAR missingness is imposed on the fully-observed training set by masking a fixed proportion of values. This trains the neural network to learn to impute missing values accurately during testing. Missing data is imputed by values that are sampled from the generative distribution $p(\mathbf{X}^m|\mathbf{Z}, \mathbf{X}^o, \mathbf{R})$. However, inducing additional missingness may be problematic for situations where there already exists a high level of inherent missingness in the data.

HI-VAE (Nazabal et al., 2018) takes advantage of the fact that the marginal log-likelihood simplifies to Equation (3) of the main text under ignorable missingness. Under this simplification, the new ELBO corresponding to Equation (1) of the main text is given by

$$\mathcal{L}(\theta, \psi) = \mathbb{E}_{\mathbf{Z}\sim q_\theta(\mathbf{Z}|\mathbf{X}^o)} \log \left[\frac{p_\psi(\mathbf{X}^o|\mathbf{Z})p(\mathbf{Z})}{q_\theta(\mathbf{Z}|\mathbf{X}^o)} \right], \tag{3.19}$$

such that the training of the VAE depends only on the observed values. To preserve the length of each observation vector, the missing features are pre-imputed by some value, typically using zero imputation. In this setting, the standard variational approximation $q_\theta(\mathbf{Z}|\mathbf{X})$ is replaced by $q_\theta(\mathbf{Z}|\mathbf{X}^o)$. This quantity approximates $p(\mathbf{Z}|\mathbf{X}^o)$ instead of $p(\mathbf{Z}|\mathbf{X})$, thus allowing the VAE to be trained with partially-observed input data. In HI-VAE, the VAE framework is adapted to be able to deal with heterogenous data, i.e. both discrete and continuous data. Missing data is imputed by either sampling from the generative model $p(\mathbf{X}^m|\mathbf{Z})$, or by setting missing values equal to the corresponding means of the posterior distribution. In contrast, the MIWAE method (Mattei and Frellsen, 2019) does not take heterogeneous data as input, but uses the IWAE framework to create a tighter lower bound on the marginal log-likelihood. They also use a principled imputation scheme that works up to MAR missingness, utilizing importance-weighted samples to calculate $\mathbb{E}[\mathbf{X}^m|\mathbf{X}^o, \mathbf{Z}]$.

3.1.4.3 Non-ignorable Missingness

In contrast, non-ignorable (or MNAR) missingness refers to the case where the missingness can be dependent on any unobserved values, including the missing entries $\mathbf{x}_i^m$. MNAR missingness can also be dependent on $\mathbf{x}_i^o$ as well as latent values like $\mathbf{Z}$, and thus MNAR represents the most general and difficult case of missingness in practice. Here, we assume that $\mathbf{R}$ is independent of $\mathbf{Z}$, as conditioning on such latent factors may be computationally redundant based on the assumed data generating process (Ibrahim, 2001). In this setting, the missingness typically requires specification of a model for the missingness $p(\mathbf{R}|\mathbf{X}, \boldsymbol{\phi})$ (Stubbendick and Ibrahim, 2003). Current VAE methods are only able to handle MCAR or MAR missingness, and there is no method to properly deal with the more difficult MNAR case. This issue is especially problematic because missingness in many real world applications have been posited to be non-ignorable (Beaulieu-Jones and and, 2016; O'Shea, 2019).

There have been a number of ways to specify $p(\mathbf{R}|\mathbf{X}, \boldsymbol{\phi})$ in statistical literature. For example, Diggle and Kenward (1994) proposes a binomial model for the missing data mechanism:

$$p(\mathbf{R}|\mathbf{X}, \boldsymbol{\phi}) = \prod_{i=1}^{n} \prod_{j_m=1}^{p} \left[p(r_{ij_m} = 1|\mathbf{x}_i, \boldsymbol{\phi}_{j_m})\right]^{r_{ij_m}} \left[1 - p(r_{ij_m} = 1|\mathbf{x}_i, \boldsymbol{\phi}_{j_m})\right]^{1-r_{ij_m}},$$

where $j_m = 1, \ldots, p_{miss}$ indexes the p_{miss} features in $\mathbf{X}$ that contain missingness, $\boldsymbol{\phi}_{j_m}$ is the sets of coefficients pertaining to j_m^{th} missingness model, and $p(r_{ij_m} = 1|\mathbf{x}_i, \boldsymbol{\phi}_{j_m})$ can be modeled straightforwardly by a logistic regression model, such that

$$logit[p(r_{ij_m} = 1|\mathbf{x}_i, \boldsymbol{\phi}_{j_m})] = \phi_{0j_m} + \mathbf{x}_i^o \boldsymbol{\phi}_{1j_m} + \mathbf{x}_i^m \boldsymbol{\phi}_{2j_m}, \tag{3.20}$$

where ϕ_{0j_m} is the intercept of the j_m^{th} missingness model, $\boldsymbol{\phi}_{1j_m} = \{\phi_{1,j_m,1}, \ldots, \phi_{1,j_m,p_{obs}}\}^T$ is a $p_{obs} \times 1$ vector of coefficients pertaining to the fully-observed features, and $\boldsymbol{\phi}_{2j_m} = \{\phi_{2,j_m,1}, \ldots, \phi_{2,j_m,p_{miss}}\}^T$ is a $p_{miss} \times 1$ vector of coefficients pertaining to the missing features.

3.1.5 NIMIWAE: IWAE with Nonignorable Missingness

We now propose a novel method to perform statistical learning and imputation using an IWAE in the presence of missing data (NIMIWAE), assuming missingness is nonignorable. We later show how this model can be simplified when missingness is assumed to be ignorable (IMIWAE). First, we specify a general form of the lower bound, in which we utilize the general IWAE framework to form a tighter bound on the marginal log-likelihood than the VAE ELBO. Let us define $q_\theta(\mathbf{Z}, \mathbf{X}^m)$ as the variational joint posterior pertaining to $(\mathbf{Z}, \mathbf{X}^m)$. We can factorize this variational joint posterior as $q_\theta(\mathbf{Z}, \mathbf{X}^m) = q_{\theta_1}(\mathbf{Z}|\mathbf{X}^o) q_{\theta_2}(\mathbf{X}^m|\mathbf{Z}, \mathbf{X}^o, \mathbf{R})$.

Here, for $k = 1, \dots, K$, we assume $\mathbf{Z}_k \overset{i.i.d}{\sim} q_{\theta_1}(\mathbf{Z}|\mathbf{X}^o)$ similar to the traditional IWAE. We additionally assume i.i.d. latent variables $\mathbf{X}_k^m \overset{i.i.d}{\sim} q_{\theta_2}(\mathbf{X}^m|\mathbf{Z}, \mathbf{X}^o, \mathbf{R})$ pertaining to the missing features, where each $\mathbf{X}_k^m$ has dimensionality p_{miss}. Similar to traditional VAEs, we utilize the class of factorized variational posteriors, except now $q_\theta(\mathbf{Z}, \mathbf{X}^m) = \prod_{i=1}^n q_\theta(\mathbf{z}_i, \mathbf{x}_i^m)$ and $q_\theta(\mathbf{z}_i, \mathbf{x}_i^m) = q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o) q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i)$. Then, denoting $\mathbf{z}_{ik}$ and $\mathbf{x}_{ik}^m$ as the i^{th} observation vectors of $\mathbf{Z}_k$ and $\mathbf{X}_k^m$, respectively, we have $\mathbf{z}_{i1}, \dots, \mathbf{z}_{iK} \overset{i.i.d}{\sim} q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)$ and $\mathbf{x}_{i1}^m, \dots, \mathbf{x}_{iK}^m \overset{i.i.d}{\sim} q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i)$. The form of the lower bound, which we call the **N**on**I**gnorably **M**issing **I**mportance-**W**eighted **A**uto **E**ncoder bound, or "**NIMIWAE** bound", is derived as follows:

$$\begin{aligned}
\log p_{\psi,\phi}(\mathbf{X}^o, \mathbf{R}) &= \sum_{i=1}^n \log p_{\psi,\phi}(\mathbf{x}_i^o, \mathbf{r}_i) \\
&= \sum_{i=1}^n \log \left[\iint p_{\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_i^m, \mathbf{r}_i, \mathbf{z}_i) d\mathbf{z}_i d\mathbf{x}_i^m \right] \\
&= \sum_{i=1}^n \log \mathbb{E}_{(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m) \sim q_\theta(\mathbf{z}_i, \mathbf{x}_i^m)} \left[\frac{1}{K} \sum_{k=1}^K \frac{p_{\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_{ik}^m, \mathbf{r}_i, \mathbf{z}_{ik})}{q_\theta(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m)} \right] \\
&\geq \sum_{i=1}^n \mathbb{E}_{(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m) \sim q_\theta(\mathbf{z}_i, \mathbf{x}_i^m)} \log \left[\frac{1}{K} \sum_{k=1}^K \frac{p_{\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_{ik}^m, \mathbf{r}_i, \mathbf{z}_{ik})}{q_\theta(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m)} \right] = \mathcal{L}_K^{NIMIWAE}. \quad (3.21)
\end{aligned}$$

As explained in Section 3.1.4, we use the selection model factorization of the joint distribution of $\{\mathbf{x}_i, \mathbf{r}_i, \mathbf{z}_i\}$, such that $p_{\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_i^m, \mathbf{r}_i, \mathbf{z}_i) = p_\psi(\mathbf{x}_i^o, \mathbf{x}_i^m|\mathbf{z}_i) p(\mathbf{z}_i) p_\phi(\mathbf{r}_i|\mathbf{x}_i^o, \mathbf{x}_i^m)$. Here, ψ denotes the weights and biases of the encoder and decoder neural networks, and ϕ denotes the weights and biases of the missingness network that learns the parameters of the missingness model.

Applying the above factorizations to Eq. (3.21), and estimating the expectations in Eq. (3.21) by sampling from $q_\theta(\mathbf{z}_i, \mathbf{x}_i^m)$ we obtain the estimate of the NIMIWAE bound:

$$\hat{\mathcal{L}}_K^{NIMIWAE} = \sum_{i=1}^n \log \left[\frac{1}{K} \sum_{k=1}^K \frac{p_\psi(\mathbf{x}_i|\tilde{\mathbf{z}}_{ik}) p(\tilde{\mathbf{z}}_{ik}) p_\phi(\mathbf{r}_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o) q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, \mathbf{r}_i)} \right], \quad (3.22)$$

where $\{\tilde{\mathbf{z}}_{ik}, \tilde{\mathbf{x}}_{ik}^m\}$ are samples of $\{\mathbf{z}_i, \mathbf{x}_i^m\}$ that are drawn via ancestral sampling (Bishop, 2006) from the variational posteriors $q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o)$ and $q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, \mathbf{r}_i)$, respectively, and the NIMIWAE bound is optimized using the Adam optimizer (Kingma and Ba, 2014). In NIMIWAE, we have four neural networks $f_\psi(\mathbf{z}_i)$, $g_{\theta_1}(\mathbf{x}_i^o)$, $g_{\theta_2}(\mathbf{x}_i^o, \mathbf{r}_i, \mathbf{z}_i)$, and $h_\phi(\boldsymbol{x}_i)$, which respectively output the parameters pertaining to $p_\psi(\mathbf{x}_i|\mathbf{z}_{ik})$, $q_{\theta_1}(\mathbf{z}_{ik}|\mathbf{x}_i^o)$, $q_{\theta_2}(\mathbf{x}_{ik}^m|\mathbf{z}_{ik}, \mathbf{x}_i^o, \mathbf{r}_i)$, and $p_\phi(\mathbf{r}_i|\mathbf{x}_i^o, \mathbf{x}_{ik}^m)$.

The quantity $h_\phi(\mathbf{x}_i)$, which we call the "missingness network", outputs the distributional parameters to the missingness model $p_\phi(\mathbf{r}_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)$, explicitly included in Eq. (3.22). Furthermore, by omitting this network and its contribution to the NIMIWAE bound altogether, one can attain an ignorably-missing version of the NIMIWAE method (IMIWAE), which would be more suitable for use under MCAR and MAR missingness. We explore the empirical performance of each of these models under misspecification of the missingness mechanism in Section 3.2. An illustration of $h_\phi(\mathbf{x}_i)$ is given in Figure 8.

Historically, the set of features for the p_{miss} logistic regression models from Eq. (3.20) need to be carefully pre-specified, usually based upon prior information (Little and Rubin, 2002). Prior work has shown that overparameterization of the missingness model can lead to identifiability issues and divergence in EM-based maximization procedures (Ibrahim and Molenberghs, 2009). Our proposed method allows users to similarly pre-specify a subset of features in the missingness network. Alternatively, when such information is not available, we show empirically in Section 3.2 that using all p features in the missingness network can yield reasonable performance, especially when the number of samples is large. We postulate that this may be due to the fact that the missingness model utilizes a neural network, which has been shown to generalize well despite severe overparameterization (Poggio et al., 2020). Still, the specification of a smaller model that is closer to the truth may improve the accuracy of imputations, especially under smaller sample sizes (Du et al., 2021).

3.1.5.1 NIMIWAE Training Algorithm

The training of the NIMIWAE architecture proceeds as follows:

1. The missing entries are pre-imputed to zero and appended with observed entries, and fed into the encoder, or $g_{\theta_1}(\mathbf{x}_i^o)$ with the missingness mask, to learn parameters of $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)$. One can also specify a mean pre-imputation, but the choice of the pre-imputation method has been shown to not significantly affect performance in an IWAE network (Mattei and Frellsen, 2019).

2. K samples are drawn from $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)$.

3. Samples from (2) are used as input for the decoder, or $f_\psi(\mathbf{z}_i)$, to learn the parameters of $p_\psi(\mathbf{x}_i|\mathbf{z}_i)$.

4. The samples from (2) are used again as input for the missing data posterior network, or $g_{\theta_2}(\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i)$, concatenated with the observed data entries (with missing entries pre-imputed to 0) and the missingness mask, to learn parameters of $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i)$.

5. We draw samples of $\mathbf{x}_i^m$ from $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i)$, and use them as input, concatenated with the fixed observed entries $\mathbf{x}_i^o$, into missingness network, or $h_\phi(\mathbf{x}_i)$ to learn the parameters associated with the model of the missingness mask $p_\phi(\mathbf{r}_i|\mathbf{x}_i)$.

In NIMIWAE, the neural network that models the missingness, or $h_\phi(\boldsymbol{x}_i)$, contains p_{miss} output nodes, and applies the Sigmoid activation function to the output node to yield probabilities of each partially-observed feature j_m being observed for the i^{th} sample. By default, this network takes all p features as input, and the number of hidden layers and nodes per hidden layer are tuned separately from the rest of the architecture. Each output node in $h_\phi(\boldsymbol{x}_i)$ is exactly equivalent to the logistic regression model proposed by Diggle and Kenward (1994) in Section 3.1.4.3 when $h_\phi(\mathbf{x}_i)$ contains no hidden layers, and the selection of additional hidden layers can allow for the capturing of more complex effects.

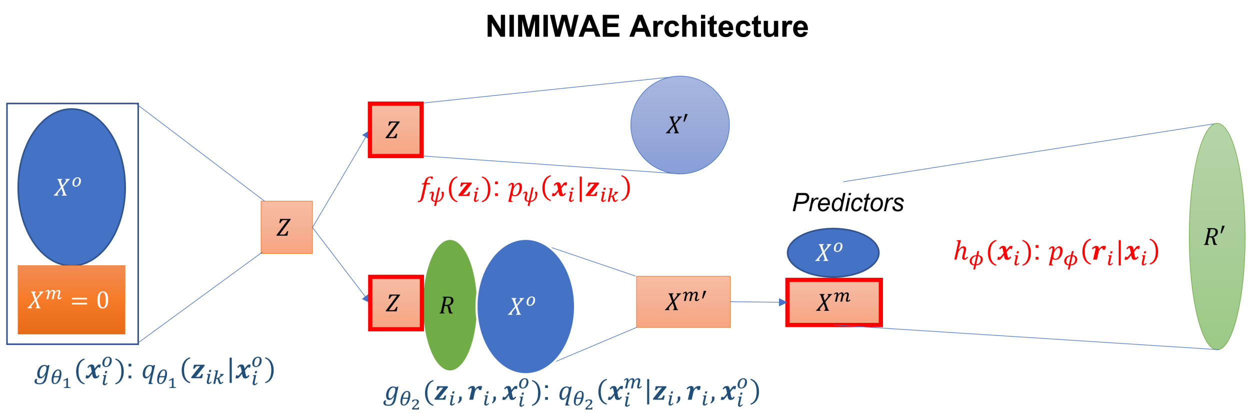
Under simple distributional assumptions of $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{r}_i, \mathbf{x}_i^o)$, the sampling step in Step (4) is similar to the sampling of the latent variable $\mathbf{z}_i$ in Step (2), and both can be accomplished using the reparametrization trick (Kingma and Welling, 2013).

3.1.5.2 Initialization, Early Stop, and Hyperparameter Tuning

Initialization of weights and biases in deep learning architectures can significantly affect the trained model, especially in datasets with smaller sample sizes. By default, NIMIWAE uses the semi-orthogonal matrix initialization (Saxe et al., 2014) for ψ, θ, and ϕ. Alternatively, we initialize weights pertaining to the missing features in the input layer of the missingness network using values drawn from a Uniform(-2,2) distribution. This is done in order to draw larger initial values of the effects of missing variables on the missingness, in order to encourage the network to learn nonzero effects of these missing features. Empirically, we found that this alternative initialization helps the network impute more accurate values in smaller sample size settings, particularly under MNAR missingness, while maintaining similar performance in the MCAR and MAR cases.

We also incorporate an early stop criterion (Prechelt, 1998) in order to prevent overfitting on the training set, and to reduce computation time. Specifically, let $L^{(\tau)} \equiv \hat{\mathcal{L}}_{K,valid}^{NIMIWAE,(\tau)}$ denote the estimated NIMIWAE bound on a held-out validation set at each epoch (τ), and initialize $L_{opt} = L^{(0)}$ and $E^{(0)} = 0$. Each epoch consists of $\lceil n_{train}/bs \rceil$ updates, where n_{train} is the number of observations in the training set and bs is the mini-batch size hyperparameter, and $\lceil x \rceil$ is the smallest integer greater than or equal to x. At each epoch, the n_{train} observations are divided into approximately equal size mini-batches of at most bs observations each, and updates of the neural network parameters are done on by estimating the NIMIWAE bound on each mini-batch, cycling through each mini-batch such that all observations are involved in the updates for during each epoch. For $\tau \geq 1$, if $L^{(\tau)} - L_{opt} > 0$, we replace $L_{opt} = L^{(\tau)}$. Also, if $L^{(\tau)} - L_{opt} \leq \varepsilon L_{opt}$, we set $E^{(\tau)} = E^{(\tau-1)} + 1$. In this way, training continues as long as the improvement in the estimated lower bound in the validation set is greater than εL_{opt}, but if not, training is allowed to run for a set number of epochs before early stopping. This leeway (called "$patience$") is allowed due to the properties of SGD, which may cause $L^{(\tau)}$ to fluctuate, especially for smaller τ. If $E^{(\tau)} = patience$, we save the optimal model pertaining to L_{opt}, and terminate training. Here, we set $patience = 50$ and $\varepsilon = 0.0001$.

In NIMIWAE, we have several hyperparameters that need to be tuned (default values in parentheses): 1) learning rate ($\{0.001, 0.01\}$), 2) number of hidden layers ($\{0, 1, 2\}$) and nodes per hidden layer ($\{64, 128\}$), 3) dimensionality of **Z** ($\{\lfloor p/12 \rfloor, \lfloor p/4 \rfloor, \lfloor p/2 \rfloor, \lfloor 3p/4 \rfloor\}$), where $\lfloor x \rfloor$ is the largest integer less than or equal to x. We also tune (2) separately for the missingness network ($\{0, 1\}$ and $\{16, 32\}$) ($h_{\phi}(\cdot)$) vs. the rest of the networks ($f_{\psi}(\cdot), g_{\theta_1}(\cdot), g_{\theta_2}(\cdot)$). In order to find the optimal combination of values for these hyperparameters, we perform an extensive grid search of all combinations of prespecified values for these quantities. One may also choose to tune the number of latent variables K. In this chapter, we fix $K = 5$ during training for computational ease.


NIMIWAE Architecture
X^o
$X^m = 0$
$g_{\theta_1}(\boldsymbol{x}_i^o)$: $q_{\theta_1}(\boldsymbol{z}_{ik}|\boldsymbol{x}_i^o)$
Z
Z
X'
$f_\psi(\boldsymbol{z}_i)$: $p_\psi(\boldsymbol{x}_i|\boldsymbol{z}_{ik})$
Z
R
X^o
$X^{m\prime}$
$g_{\theta_2}(\boldsymbol{z}_i, \boldsymbol{r}_i, \boldsymbol{x}_i^o)$: $q_{\theta_2}(\boldsymbol{x}_i^m|\boldsymbol{z}_i, \boldsymbol{r}_i, \boldsymbol{x}_i^o)$
Predictors
X^o
X^m
$h_\phi(\boldsymbol{x}_i)$: $p_\phi(\boldsymbol{r}_i|\boldsymbol{x}_i)$
R'

$$\mathcal{L}_K^{NIMIWAE} = \sum_{i=1}^{n} \mathbb{E}_{(\boldsymbol{z}_{ik}, \boldsymbol{x}_{ik}^m) \sim q_\theta(\boldsymbol{z}_i, \boldsymbol{x}_i^m | \boldsymbol{x}_i^o, \boldsymbol{r}_i)} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\psi(\boldsymbol{x}_i^o, \boldsymbol{x}_{ik}^m | \boldsymbol{z}_{ik})\, p_\phi(\boldsymbol{r}_i | \boldsymbol{x}_i^o, \boldsymbol{x}_{ik}^m)\, p(\boldsymbol{z}_{ik})}{q_{\theta_1}(\boldsymbol{z}_{ik}|\boldsymbol{x}_i^o)\, q_{\theta_2}(\boldsymbol{x}_{ik}^m | \boldsymbol{x}_i^o, \boldsymbol{z}_{ik}, \boldsymbol{r}_i)} \right]$$

$$\hat{\mathcal{L}}_K^{NIMIWAE} = \sum_{i=1}^{n} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\psi(\boldsymbol{x}_i^o, \tilde{\boldsymbol{x}}_{ik}^m | \tilde{\boldsymbol{z}}_{ik})\, p_\phi(\boldsymbol{r}_i | \boldsymbol{x}_i^o, \tilde{\boldsymbol{x}}_{ik}^m)\, p(\tilde{\boldsymbol{z}}_{ik})}{q_{\theta_1}(\tilde{\boldsymbol{z}}_{ik}|\boldsymbol{x}_i^o)\, q_{\theta_2}(\tilde{\boldsymbol{x}}_{ik}^m | \boldsymbol{x}_i^o, \tilde{\boldsymbol{z}}_{ik}, \boldsymbol{r}_i)} \right]$$

Figure 8: Architecture of proposed NIMIWAE method. Dark colored nodes represent deterministic values, lightly colored nodes represent learned distributional parameters, and outlined (in red) nodes represent sampled values. Orange cells correspond to latent variables $\mathbf{Z}$ and $\mathbf{X}^m$. $\mathbf{Z}_1, \ldots, \mathbf{Z}_K$ and $\mathbf{X}_1^m, \ldots, \mathbf{X}_K^m$ are sampled from their respective variational posteriors. Below is the NIMIWAE bound ($\mathcal{L}_K^{NIMIWAE}$) and the estimate of the NIMIWAE bound ($\hat{\mathcal{L}}_K^{NIMIWAE}$), which is optimized via SGD.

3.1.5.3 Multiple Imputation

Following training, NIMIWAE can provide point estimates for $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, \mathbf{r}_i]$, defined as the expected value of the missing features given the observed data and the mask for the i^{th} observation under MNAR. The same NIMIWAE model can also perform multiple imputation of the incomplete training dataset to facilitate fitting of downstream statistical models. We first note that

$$\begin{aligned}
\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, \mathbf{r}_i] &= \int \mathbf{x}_i^m p_{\psi,\phi}(\mathbf{x}_i^m|\mathbf{x}_i^o, \mathbf{r}_i) d\mathbf{x}_i^m \\
&= \iint \mathbf{x}_i^m p_{\psi,\phi}(\mathbf{x}_i^m, \mathbf{z}_i|\mathbf{x}_i^o, \mathbf{r}_i) d\mathbf{z}_i d\mathbf{x}_i^m \\
&= \iint \mathbf{x}_i^m \frac{p_{\psi,\phi}(\mathbf{x}_i^m, \mathbf{x}_i^o, \mathbf{z}_i, \mathbf{r}_i)}{p_{\psi,\phi}(\mathbf{x}_i^o, \mathbf{r}_i)} d\mathbf{z}_i d\mathbf{x}_i^m.
\end{aligned}$$

Then, we may estimate this integral by self-normalized importance sampling. We utilize the proposal density $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o) q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i)$, and define the quantities

$$w_{ik} = \frac{s_{ik}}{s_{i1} + \ldots + s_{iK}}, \text{ and } s_{ik} = \frac{p_\psi(\mathbf{x}_i|\tilde{\mathbf{z}}_{ik}) p(\tilde{\mathbf{z}}_{ik}) p_\phi(\mathbf{r}_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o) q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, \mathbf{r}_i)}$$

for $k = 1, \dots, K$, with 1 sample drawn from the variational posterior of each latent variable $\mathbf{z}_{ik}$ and $\mathbf{x}_{ik}^m$ to compute s_{ik}, where w_{ik} is defined as standardized "importance weights" (Mattei and Frellsen, 2019). Using these weights we may compute $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, \mathbf{r}_i] \approx \sum_{k=1}^K w_{ik}\tilde{\mathbf{x}}_{ik}^m$. Then, the process can be repeated for each observation $i = 1, \dots, n$.

In the MCAR or MAR case, one can similarly estimate $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o]$ using the fitted IMIWAE model. By following a similar derivation using the proposal density $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o)$, we obtain the same approximation $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o] \approx \sum_{k=1}^K w_{ik}\tilde{\mathbf{x}}_{ik}^m$, with w_{ik} defined as before, but with a slightly different form for s_{ik}:

$$s_{ik} = \frac{p_\psi(\mathbf{x}_i|\tilde{\mathbf{z}}_{ik})p(\tilde{\mathbf{z}}_{ik})}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o)q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o)}.$$

Given these weights w_{ik}, we may now also produce Q multiply-imputed datasets using the sampling importance resampling (SIR) algorithm (Smith and Gelfand, 1992). We first construct a set of K candidate draws and corresponding weights using the procedure described above (default $K = 10 \times Q$ draws), and then perform a weighted resample of size Q with replacement from this set of draws to obtain approximate draws from $p_\psi(\mathbf{x}_i^m|\mathbf{x}_i^o, \mathbf{r}_i)$, for each observation $i = 1, \dots, n$. We may then use techniques such as "Rubin's rules" (Rubin, 2004) to pool the estimates obtained from a candidate regression model fit on each imputed dataset, and obtain pooled point estimates and standard errors that account for the uncertainty due to the imputation. In our analyses, we used NIMIWAE to construct $Q = 50$ multiply-imputed datasets by drawing $K = 500$ times from $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i)$ for each $i = 1, \dots, n$ after the model was trained.

3.2 Numerical Results

3.2.1 Simulated Data

We utilize statistical simulation to evaluate the imputation performance of our proposed NIMIWAE method under the assumption of MCAR, MAR, and MNAR missingness. We also compared this performance to state-of-the-art missing data methods in machine learning that claim to handle ignorable missingness patterns: HIVAE (Nazabal et al., 2018), VAEAC (Ivanov et al., 2019), MIWAE (Mattei and Frellsen, 2019), in addition to the popular MICE method (Van Buuren and Groothuis-Oudshoorn, 2011) and a naïve mean imputation method. We also included MissForest (Stekhoven and Buhlmann, 2011) in analyses where the model could be fit in a CPU with 32 GB of memory. For all simulations, we divided the full data into training and validation sets with ratio 8:2. For methods that require hyperparameter tuning, each method was trained on the training set for a given set of hyperparameters, the performance of this model was then evaluated on the validation set, and finally the training set was imputed using the model pertaining to the optimal set of hyperparameters (based up on validation set performnance). For methods that required no hyperparameter tuning, we directly imputed just the training set, for consistency across all methods. In Sections 3.2.1.1-3.2.1.2, we evaluate performance on fully synthetic data. Then, in Section 3.2.1.3, we simulate missingness into existing UCI datasets to preserve non-linearity and interactions between features previously observed. The simulation setup and performance criteria are described in the subsequent sections.

3.2.1.1 Simulation Setup

We first evaluate the performance of each method on completely synthetic data. Here we assume $\mathbf{X}$ is generated such that $\mathbf{X} = \mathbf{ZW} + \mathbf{B}$, where $\mathbf{W}$ and $\mathbf{B}$ and are matrices of dimensions $d \times p$ and $n \times p$, respectively, $\mathbf{Z} \sim N_d(\mathbf{0}, \mathbf{I})$, $W_{lj} \sim N(0, 0.5)$, and $B_{ij} \sim N(0, 1)$ for $i = 1, \ldots, n$, $j = 1, \ldots, p$, and $l = 1, \ldots, d$.

We then simulate the missingness mask matrix $\mathbf{R}$ such that 30% of features are partially observed, and 50% of the observations for each of these features are missing. We generate r_{ij} from the Bernoulli distribution with probability equal to $p(r_{ij_m} = 1|\mathbf{x}_i, \boldsymbol{\phi})$, such that logit$[p(r_{ij_m} = 1|\mathbf{x}_i, \boldsymbol{\phi})] = \phi_0 + \boldsymbol{\phi}_1 \mathbf{x}_i^o + \boldsymbol{\phi}_2 \mathbf{x}_i^m$. Here, we assume that $j_m = 1, \ldots, p_{miss}$ index the missing features, $\boldsymbol{\phi}_1 = \{\phi_{11}, \ldots, \phi_{1,p_{obs}}\}$ are the coefficients pertaining to the fully observed features, and $\boldsymbol{\phi}_2 = \{\phi_{21}, \ldots, \phi_{2,p_{miss}}\}$ are those pertaining to the partially observed features, and p_{obs} and p_{miss} are the total number of features that are fully and partially observed, respectively. We set $p_{miss} = \lfloor 0.3 * p \rfloor$ and $p_{obs} = p - p_{miss}$. We drew nonzero values of $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ from the log-normal distribution with mean $\mu_\phi = 5$, with log standard deviation $\sigma_\phi = 0.2$.

To evaluate the impact of the misspecification of the missingness mechanism on model performance, r_{ij_m} was simulated under each mechanism as follows: (1) MCAR: $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2\} = 0$, (2) MAR: Same as MCAR except $\phi_{1j_o} \neq 0$ for one randomly selected completely-observed feature j_o where $j_o = p_{miss} + 1, \ldots, p$, and (3) MNAR: Same as MCAR except $\phi_{2j_m} \neq 0$. In this way, for each MAR or MNAR feature, the missingness is dependent on just one feature. In each case, we used ϕ_0 to control for an expected rate of missingness of 50% in each feature. We note that for each these simulations, we utilize all features in NIMIWAE's missingness network, although only one feature is involved under the true missingness model.

Lastly, we simulated a binary response variable assuming $\Pr(\mathbf{y} = 1|\mathbf{X}) = Sigmoid(\beta_0 + \mathbf{X}\boldsymbol{\beta})$, where $\mathbf{y}$ is a binary response variable, $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_p\}$ are the set of regression coefficients, and β_0 is the intercept. In many applications, it is of interest to use the features in $\boldsymbol{X}$ to predict some outcome variable $\mathbf{y}$ when $\boldsymbol{X}$ is only partially observed. Therefore, the ability accurately estimate $\boldsymbol{\beta}$ is also of importance in the presence of missingness. Multiple imputation methods like *mice* can be used to perform coefficient estimation by using Rubin's rules to pool the coefficient estimates from logistic regression models fitted on each individual multiply-imputed dataset. We similarly perform coefficient estimation using multiply-imputed datasets from the SIR algorithm within NIMIWAE, allowing for direct comparisons with MICE in estimating $\boldsymbol{\beta}$.

We vary n, p, and d such that $n = \{10,000, 100,000\}$, $p = \{25, 100\}$ features, and $d = \{2, 8\}$. We simulated 5 datasets per simulation condition, spanning various missingness mechanisms and values for $\{n, p, d\}$. We fix the values of $\boldsymbol{\beta}$ at 0.25 for each feature, and adjusted β_0 to ensure equal proportions for the two binary classes in $\mathbf{Y}$. We measured imputation performance by calculating the average L1 distance between true and imputed masked values in $\mathbf{X}$. Letting $\hat{\mathbf{X}}^m$ denote the imputed masked values of the true $\mathbf{X}^m$ values of the missing entries, we denote the average L1 distance is simply $\frac{|\hat{\mathbf{X}}^m - \mathbf{X}^m|}{N_{miss}}$, where N_{miss} is the total number of missing entries in the dataset. To assess the ability of multiple imputation methods in performing coefficient estimation, we reported the percent bias (PB) of these pooled estimates compared to the truth, averaged across the p features, i.e. $PB = \frac{1}{p}\sum_{j=1}^{p} \frac{|\beta_j - \hat{\beta}_j|}{|\beta_j|}$.

Below, we summarize the combinations of hyperparameter values that were searched. For each method and dataset, we searched over combinations of two different values for each hyperparameter that was tuned. We tuned several hyperparameters for each method: number of nodes per hidden layer (h), number of nodes per hidden layer in the missingness network (h_r) , number of hidden layers (nhl), number of hidden layers in the missingness network (nhl_r), dimensionality of latent space (dz), and learning rate (lr). For HI-VAE, nhl was not able to be varied. Therefore, we fixed nhl and tuned the dimensionality of the y latent variable (dy) instead. Hyperparameters nhl_r and h_r were applicable only to NIMIWAE, so they were tuned only for NIMIWAE. Hyperparameters batch size (bs) and maximum number of epochs ($epochs_{max}$) were fixed in our analyses. Below are the searched values of the hyperparameters for all of the datasets we analyzed in Section 3.2.1:

- $p = (25, 100)$ Simulated data with $n = (10,000; 100,000)$
  - $h = \{128, 64\}$, $h_r = \{16, 32\}$

- $lr = \{0.001, 0.01\}$
- $dz = \{\lfloor 3 * p/4 \rfloor, \lfloor p/2 \rfloor, \lfloor p/4 \rfloor, 8\}$
- $nhl = \{0, 1, 2\}$
- $nhl_r = \{0, 1\}$
- $bs = 1,000$ for $n = 10,000$ and $bs = 10,000$ for $n = 100,000$, $epochs_{max} = 2002$

- banknote
  - $h = \{128, 64\}$, $h_r = \{16, 32\}$
  - $lr = \{0.001, 0.01\}$
  - $dz = \{1, 2, 3\}$
  - $nhl = \{0, 1, 2\}$
  - $nhl_r = \{0, 1\}$
  - $bs = 200$, $epochs = 2002$
- concrete
  - $h = \{128, 64\}$, $h_r = \{16, 32\}$
  - $lr = \{0.001, 0.01\}$
  - $dz = \{2, 4, 6\}$
  - $nhl = \{0, 1, 2\}$
  - $nhl_r = \{0, 1\}$
  - $bs = 200$, $epochs = 2002$
- hepmass
  - $h = \{128, 64\}$, $h_r = \{16, 32\}$
  - $lr = \{0.001, 0.01\}$
  - $dz = \{5, 10, 15\}$
  - $nhl = \{0, 1, 2\}$

- $nhl_r = \{0, 1\}$
- $bs = 10,000$, $epochs_{max} = 2002$

• power

- $h = \{128, 64\}$, $h_r = \{16, 32\}$
- $lr = \{0.001, 0.01\}$
- $dz = \{1, 3, 4\}$
- $nhl = \{0, 1, 2\}$
- $nhl_r = \{0, 1\}$
- $bs = 10,000$, $epochs_{max} = 2002$

• red

- $h = \{128, 64\}$, $h_r = \{16, 32\}$
- $lr = \{0.001, 0.01\}$
- $dz = \{2, 5, 8\}$
- $nhl = \{0, 1, 2\}$
- $nhl_r = \{0, 1\}$
- $bs = 200$, $epochs_{max} = 2002$

• white

- $h = \{128, 64\}$, $h_r = \{16, 32\}$
- $lr = \{0.001, 0.01\}$
- $dz = \{2, 5, 8\}$
- $nhl = \{0, 1, 2\}$
- $nhl_r = \{0, 1\}$
- $bs = 200$, $epochs_{max} = 2002$

• Physionet 2012 Challenge data

- $h = \{128, 64\}$, $h_r = \{16, 32\}$
- $lr = \{0.001, 0.01\}$
- $dz = \{8, 28, 57, 85\}$
- $nhl = \{0, 1, 2\}$
- $nhl_r = \{0, 1, 2\}$
- $bs = 1,000$, $epochs_{max} = 2002$

3.2.1.2 Simulation Results

Figure 9 shows the results pertaining to $n = 100,000$. Error bars represented the variability in performance across 5 simulations. Despite the overparameterized missingness model, NIMIWAE consistently yields improved imputation performance compared to other methods under MNAR missingness, while yielding an average L1 that is comparable to other methods in the MCAR and MAR cases, under these conditions assuming large sample sizes. This can be exceptionally useful for many real data applications, where the true covariates of the missingness model may not be known *a priori*. In estimating $\boldsymbol{\beta}$, we see that NIMIWAE yields a lower average percent bias under MNAR missingness, while MICE, IMIWAE, and NIMIWAE perform similarly under MCAR or MAR missingness.

Figure 10 shows the results pertaining to $n = 10,000$ utilizing the alternative initialization method described in Section 3.1.5, due to the smaller sample size in this setting. We see that NIMIWAE performs best in imputing MNAR values, and in estimating the coefficients under MNAR missingness, and still performs comparably well to other methods in the MCAR and MAR cases. We found that the default method initialized weights for missing features too small for the network to recover in the MNAR case, under the lower sample size and high dimensionality setting. Based upon these results, we recommend the alternative weight initialization for smaller sample sizes ($n \leq 10,000$) with large dimensionality ($p \geq 100$), while the default initialization may suffice when the sample size is large, or if the dimensionality of the data is smaller. Alternatively, one may greatly improve imputation performance by narrowing down the features that are input into the missingness network using some prior knowledge or assumptions on the mechanism of missingness. In practical applications, it is unknown which mechanism may truly underly the data, and, as usual, sensitivity analyses, where one varies the assumed mechanism, is still important. For example, imputing data via IMIWAE may be helpful and be more efficient under MCAR or MAR than via NIMIWAE.

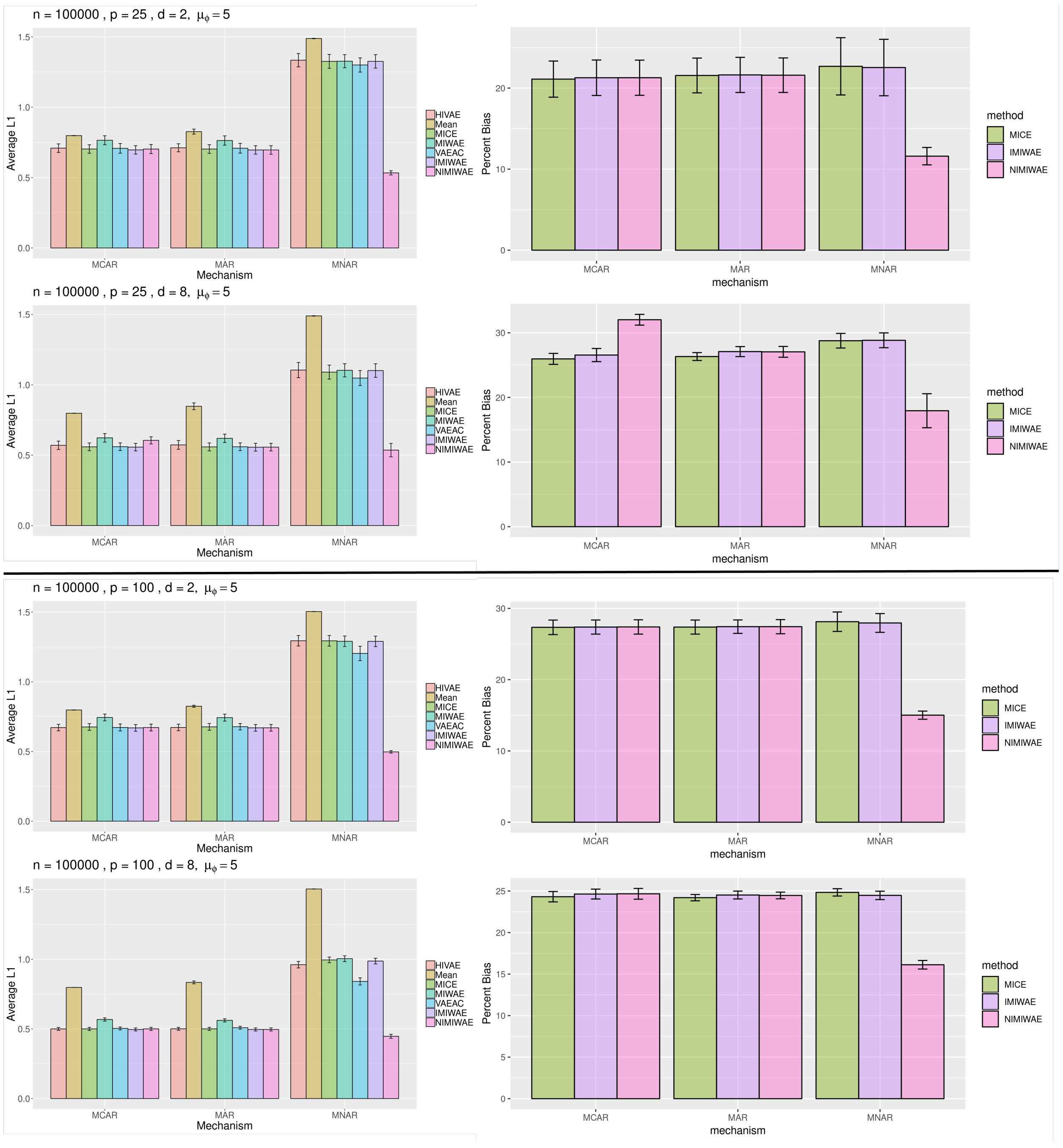


Figure 9: Average L1 distance between true and imputed values for missing entries (left) and percent bias of pooled coefficient estimates (right) for $p = 25$ (top 4) and $p = 100$ (bottom 4) features, stratified by d. NIMIWAE outperforms all methods in imputing MNAR missing values, while performing comparably to other methods in imputing MCAR and MAR values. Here, $n = 100,000$, $\mu_\phi = 5$, and error bars show the variability of each metric across 5 reps. Weights and biases were initialized by using the default semi-orthogonal matrix method.

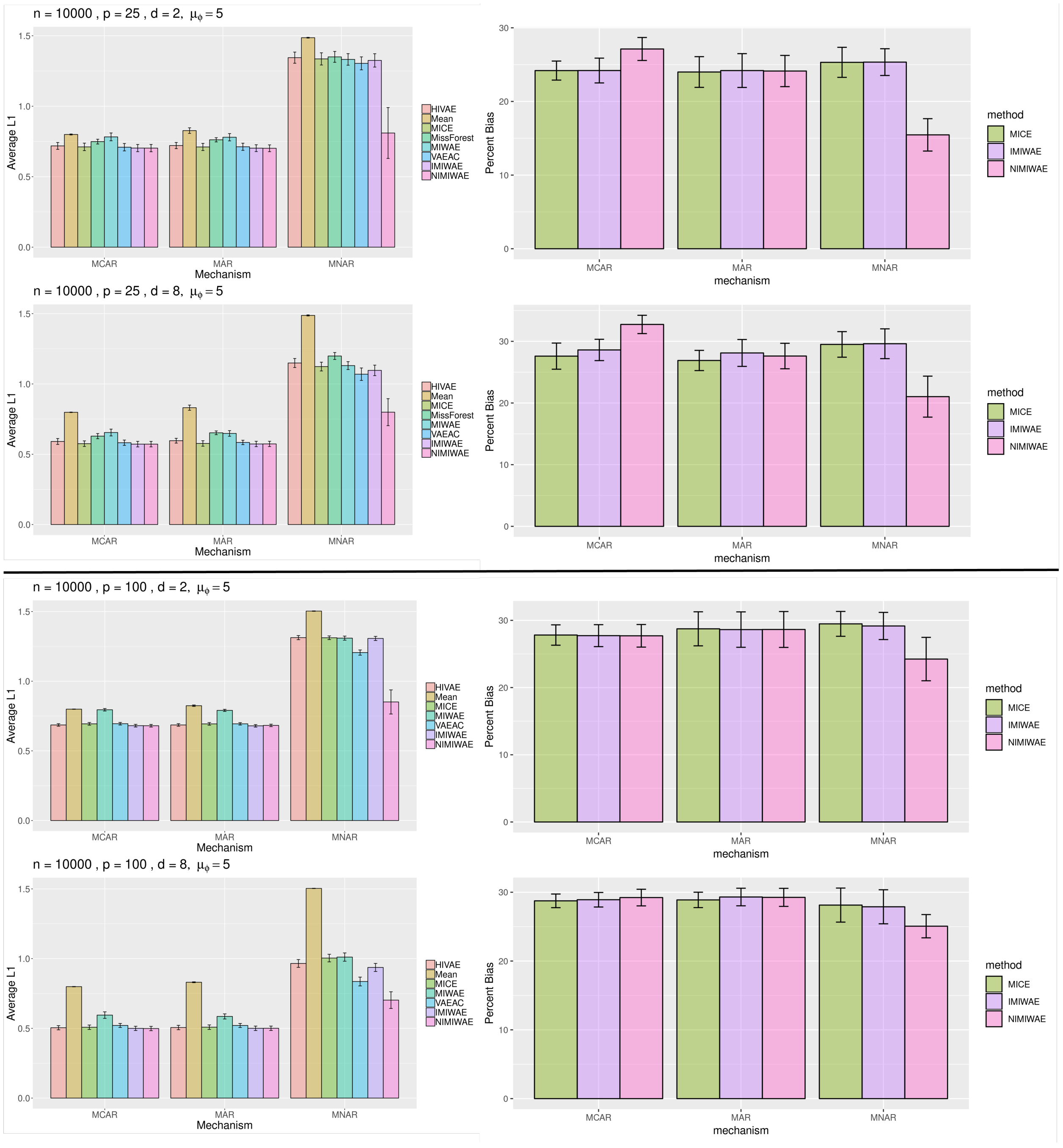


Figure 10: Average L1 distance between true and imputed values for missing entries (left) and percent bias of pooled coefficient estimates (right) for $p = 25$ (top 4) and $p = 100$ (bottom 4) features, stratified by d. NIMIWAE outperforms all methods in imputing MNAR missing values, while performing comparably to other methods in imputing MCAR and MAR values. Here, $n = 10,000$, $\mu_\phi = 5$, and error bars show the variability of each metric across 5 reps. Weights and biases were initialized by using the alternative method, as described in Section 3.1.5

Overall, these simulations confirm that existing methods can impute MCAR and MAR missingness with a reasonable degree of accuracy, but they break down under MNAR missingness. Our NIMIWAE method is able to adequately impute MNAR missing values, while still imputing MCAR and MAR missing values with a comparable degree of accuracy as existing methods geared specifically towards the ignorable mechanisms of missingness. Additionally, we note that MissForest was able to be run to the $n = 10,000$ and $p = 25$ simulation case only due to memory constraints, and yielded very poor imputation performance in the MNAR case, like the other ignorably-missing methods.

Figure 11 shows results from the simulation setup with $n = 10,000$, and Figure 12 shows results from the alternate setup with $n = 100,000$, varying $p = \{25, 100\}$ and $d = \{2, 8\}$ in each case. As in all deep learning algorithms, NIMIWAE performs best under larger sample sizes, and this aspect is especially important given the additional task in NIMIWAE of narrowing down relevant features in a potentially overspecified missingness model. When a large number of samples is not readily available, and the dimensionality of the data is very large, one can either utilize an alternative method of weight initialization for the missing features in the missingness network, as seen in Section 3.1 of the main text, or narrow down the number of features that are included in NIMIWAE's missingness network using some prior knowledge on the dependencies of the features with the missingness. We found that under the large sample size setting ($n = 100,000$), imputation performance was generally robust to the method of initialization, as shown in Figure 12. We additionally found that having fewer unnecessary features in the missingness model greatly increased the performance of both imputation and coefficient estimation under MNAR, as one would expect. However, omitting a feature that is truly relevant to the missingness may cause very poor performance in both imputation and downstream coefficient estimation, and in practice, determining which features are relevant to the missingness may be nontrivial. One may utilize *a priori* knowledge about the data in order to inform a decision on which features are significant in the missingness model, but such decisions may not be able to be made with a high levels of certainty.

3.2.1.3 UCI Machine Learning Datasets

Next, we analyzed how accurately these methods can impute missing values that may be present in real data, preserving non-linearity and interactions that may exist in these datasets, while controlling the mechanism of simulated missingness. Since we are unable to obtain the true missing values in real datasets, we took 6 different completely-observed datasets from the UCI Machine Learning Repository, and simulated

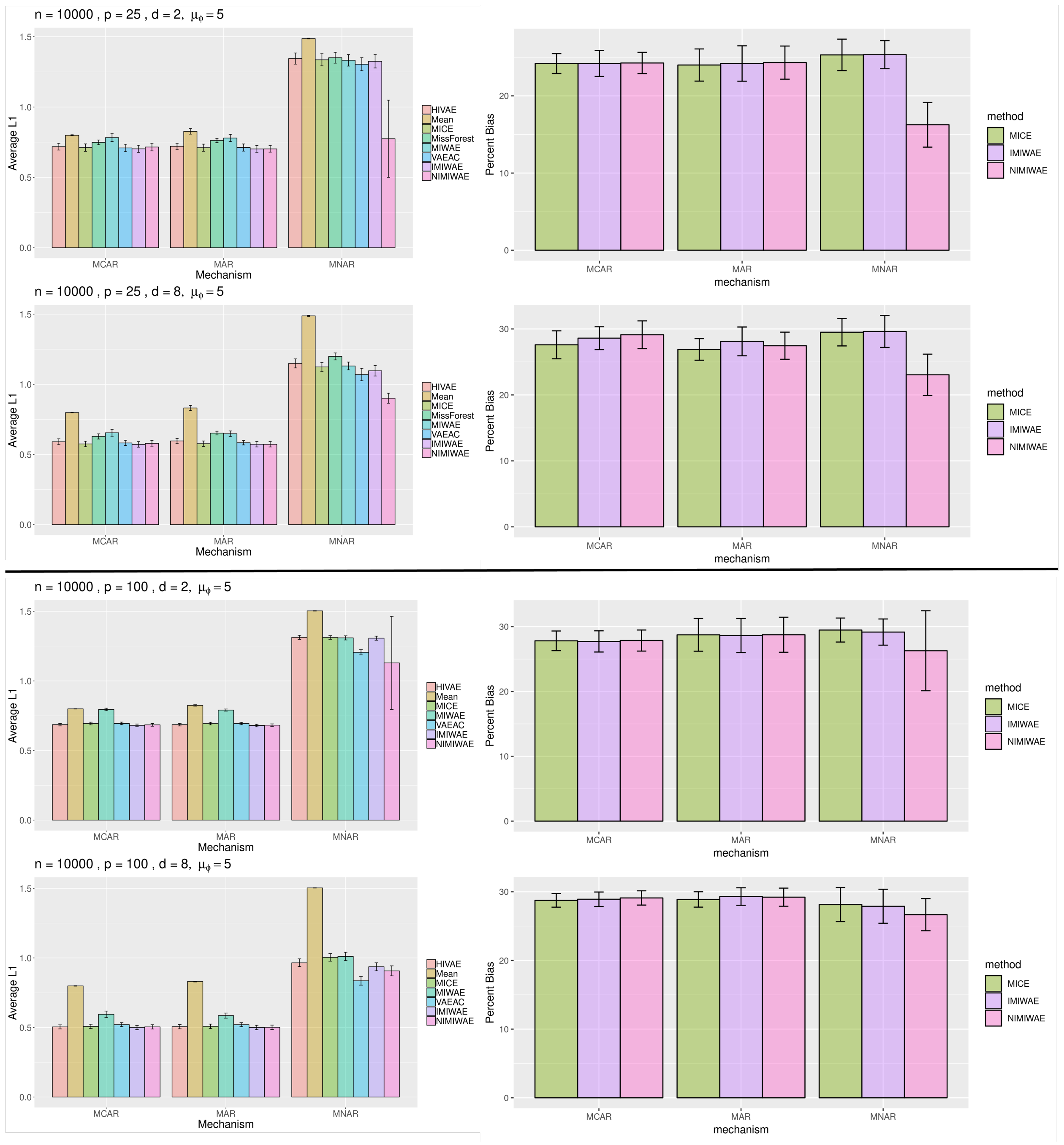


Figure 11: Average L1 distance between true and imputed values for missing entries stratified by d (left), and percent bias of pooled coefficient estimates. NIMIWAE outperforms all methods in imputing missing values that were simulated to be MNAR when $p = 25$, but performs poorly under $p = 100$. Here, $n = 10,000$, $\mu_\phi = 5$, and error bars show the variability of each metric across the 5 reps. NIMIWAE struggles to handle the difficult MNAR missingness pattern when the sample size is smaller, while the dimensionality of the data is large.

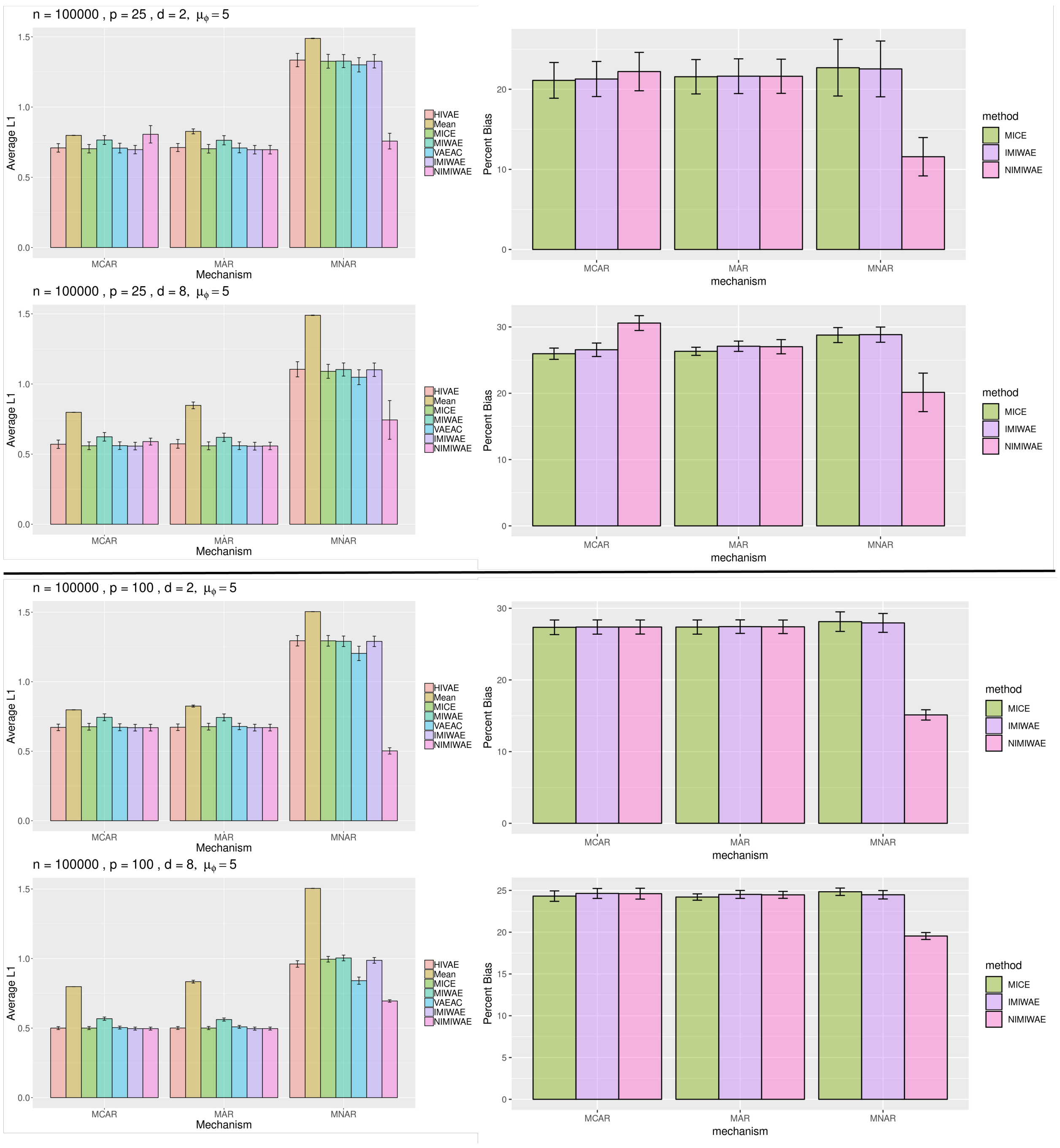


Figure 12: Average L1 distance between true and imputed values for missing entries stratified by d (left), and percent bias of pooled coefficient estimates. NIMIWAE was run using the alternative initialization method, as described in Section 2.3 of the main text. Again, NIMIWAE outperforms all methods in imputing missing values that were simulated to be MNAR. Here, $n = 100,000$, $\mu_\phi = 5$, and error bars show the variability of each metric across the 5 reps.

missingness according to each mechanism as in the fully synthetic datasets. Here, we masked half of the features, such that $p_{miss} = floor(0.5 * p)$.

Dataset		HIVAE	Mean	MICE	MissForest	MIWAE	VAEAC	NIMIWAE	IMIWAE
banknote	*MCAR*	1.62	2.00	1.63	0.95	1.37	1.32	1.49	1.19
$n = 1372$	*MAR*	2.00	2.27	2.43	1.88	2.90	1.76	1.89	1.96
	MNAR	3.39	3.92	3.78	3.18	3.10	3.46	1.46	3.30
concrete	*MCAR*	47.54	51.28	29.97	25.57	40.53	33.03	42.12	33.69
$n = 1030$	*MAR*	67.37	60.00	44.77	53.19	66.35	61.09	55.82	57.56
	MNAR	59.48	95.85	68.24	79.87	76.79	70.12	47.46	74.44
hepmass	*MCAR*	0.75	0.82	0.74	NA	0.80	0.68	0.78	0.69
$n = 525,123$	*MAR*	0.76	0.84	0.75	NA	0.84	0.72	0.72	0.71
	MNAR	1.41	1.54	1.40	NA	1.36	1.30	0.99	1.36
power	*MCAR*	0.54	0.66	0.50	NA	0.59	0.48	0.56	0.49
$n = 1,000,000$	*MAR*	0.61	0.75	0.56	NA	0.67	0.54	0.78	0.57
	MNAR	0.75	1.14	0.84	NA	0.86	0.79	0.73	0.79
red	*MCAR*	1.15	1.64	1.04	0.86	1.10	1.10	1.08	0.98
$n = 1599$	*MAR*	1.23	1.67	1.16	0.98	1.30	1.29	1.09	1.06
	MNAR	2.12	3.24	2.46	2.06	1.87	2.73	0.90	1.74
white	*MCAR*	2.14	2.61	1.97	1.60	2.18	1.93	2.16	1.88
$n = 4898$	*MAR*	2.28	2.63	1.99	1.69	2.15	2.11	1.89	1.89
	MNAR	4.42	5.36	4.21	4.27	4.46	4.10	3.47	4.70

Table 5: Average L1 distance between true and imputed values for the masked entries in various datasets, under different mechanisms of simulated missingness. Best imputation performance (lowest average L1) in each row is highlighted in red. Proportion of missing entries was fixed at 50% per feature, with 50% of the features containing missingness. We see that NIMIWAE consistently performs best in imputing MNAR missingness, while performance of the "Ignorable" IMIWAE model is comparable to other methods under MCAR and MAR. Although MissForest claims superiority in MCAR and some MAR cases in the smaller datasets, it was not scalable to larger datasets like hepmass and power.

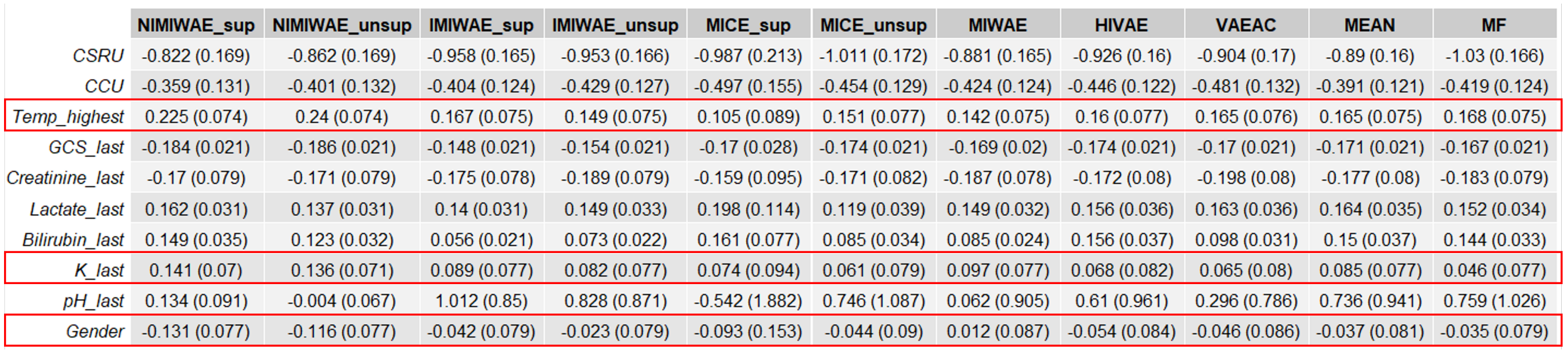
Table 5 shows the average L1 distance between true and imputed missing values for each of the 6 UCI datasets, with each simulated mechanism of missingness. We see that NIMIWAE again performs best across all methods in accurately imputing MNAR missing values. Overall, MissForest performed best in imputing MCAR missing values in the smaller UCI datasets, but this method was too memory-consuming to be trained on the significantly larger *hepmass* and *power* datasets. As in the simulated data, mean imputation is consistently one of the least accurate methods of imputation. Also, whereas MICE performed very well under MCAR and MAR in the simulated data, deep learning methods like VAEAC and IMIWAE consistently yield more accurate imputations here. This may be due to the fact that MICE uses a fully-conditional linear model for imputation, it may therefore perform suboptimally when the true relationships between features are non-linear. Also, we see that NIMIWAE imputes values slightly less accurately than IMIWAE when the missingness is MCAR or MAR, since in these cases NIMIWAE explicitly estimates the missingness network when it is not necessary.

3.2.2 Physionet 2012 Challenge Dataset

Finally, we analyzed the Physionet 2012 Challenge data using each of the compared methods. Namely, we perform a qualitative analysis of each imputed dataset, highlighting differences between results from assuming non-ignorable (NIMIWAE) and ignorable (IMIWAE) missingness (Ibrahim et al., 2005; Ibrahim and Molenberghs, 2009). This is because, in contrast to our simulations, the true values of the missing entries are not available to directly assess imputation performance. Additionally, the missingness mechanism itself is generally not "testable" by the observed data in practice (Ibrahim et al., 1999). Here, we used the alternative initialization scheme of NIMIWAE, due to the limited sample size and large dimension of the data. Additionally, we added a "supervised" and "unsupervised" version of NIMIWAE, IMIWAE, and MICE, so that these multiple imputation methods can also leverage the response variable in imputing missing entries (supervised setting), and also perform Rubin's rules to pool the estimates, and compute standard errors across imputations.

Based on the imputed datasets from these methods, we fit a logistic regression model with post-baseline mortality as the binary response, with the baseline features of this dataset as the covariates. We report details of the covariates with the top 10 largest effects on mortality when using the multiply imputed datasets from the supervised non-ignorably missing NIMIWAE model in Table 13. We found some significant differences in the results of the logistic regression based on the different imputed datasets. For example, we found that $NIMIWAE_{sup}$ uncovered a stronger effect of the variables $Temp\_highest$, K_last and $Gender$ compared to other methods. These factors have been studied for their association with mortality in ICU patients. Specifically, gender has been studied for having a potentially significant effect on mortality in critically-ill patients (Mahmood et al., 2012; Larsson et al., 2019). Additionally, body temperature (Schell-Chaple et al., 2015) and irregular potassium levels (Tongyoo et al., 2018) have both been known to be associated with an increased risk of in-hospital mortality.

Figure 14 shows the imputed values of two example variables that had significantly different missingness rates in surviving vs deceased patients, $FiO2\_last$ and $RespRate\_last$. Imputed values from each of the methods are shown in the boxplots on the right, and the rates of missingness in the deceased vs. surviving patients for each variable are shown on the left. We found that a larger proportion of entries of $FiO2\_last$ and a smaller proportion of entries of $RespRate\_last$ were observed in the surviving patients than in the deceased patients. Of the methods we used to impute values of $FiO2\_last$, we found that $NIMIWAE_{sup}$



	NIMIWAE_sup	NIMIWAE_unsup	IMIWAE_sup	IMIWAE_unsup	MICE_sup	MICE_unsup	MIWAE	HIVAE	VAEAC	MEAN	MF
CSRU	-0.822 (0.169)	-0.862 (0.169)	-0.958 (0.165)	-0.953 (0.166)	-0.987 (0.213)	-1.011 (0.172)	-0.881 (0.165)	-0.926 (0.16)	-0.904 (0.17)	-0.89 (0.16)	-1.03 (0.166)
CCU	-0.359 (0.131)	-0.401 (0.132)	-0.404 (0.124)	-0.429 (0.127)	-0.497 (0.155)	-0.454 (0.129)	-0.424 (0.124)	-0.446 (0.122)	-0.481 (0.132)	-0.391 (0.121)	-0.419 (0.124)
Temp_highest	0.225 (0.074)	0.24 (0.074)	0.167 (0.075)	0.149 (0.075)	0.105 (0.089)	0.151 (0.077)	0.142 (0.075)	0.16 (0.077)	0.165 (0.076)	0.165 (0.075)	0.168 (0.075)
GCS_last	-0.184 (0.021)	-0.186 (0.021)	-0.148 (0.021)	-0.154 (0.021)	-0.17 (0.028)	-0.174 (0.021)	-0.169 (0.02)	-0.174 (0.021)	-0.17 (0.021)	-0.171 (0.021)	-0.167 (0.021)
Creatinine_last	-0.17 (0.079)	-0.171 (0.079)	-0.175 (0.078)	-0.189 (0.079)	-0.159 (0.095)	-0.171 (0.082)	-0.187 (0.078)	-0.172 (0.08)	-0.198 (0.08)	-0.177 (0.08)	-0.183 (0.079)
Lactate_last	0.162 (0.031)	0.137 (0.031)	0.14 (0.031)	0.149 (0.033)	0.198 (0.114)	0.119 (0.039)	0.149 (0.032)	0.156 (0.036)	0.163 (0.036)	0.164 (0.035)	0.152 (0.034)
Bilirubin_last	0.149 (0.035)	0.123 (0.032)	0.056 (0.021)	0.073 (0.022)	0.161 (0.077)	0.085 (0.034)	0.085 (0.024)	0.156 (0.037)	0.098 (0.031)	0.15 (0.037)	0.144 (0.033)
K_last	0.141 (0.07)	0.136 (0.071)	0.089 (0.077)	0.082 (0.077)	0.074 (0.094)	0.061 (0.079)	0.097 (0.077)	0.068 (0.082)	0.065 (0.08)	0.085 (0.077)	0.046 (0.077)
pH_last	0.134 (0.091)	-0.004 (0.067)	1.012 (0.85)	0.828 (0.871)	-0.542 (1.882)	0.746 (1.087)	0.062 (0.905)	0.61 (0.961)	0.296 (0.786)	0.736 (0.941)	0.759 (1.026)
Gender	-0.131 (0.077)	-0.116 (0.077)	-0.042 (0.079)	-0.023 (0.079)	-0.093 (0.153)	-0.044 (0.09)	0.012 (0.087)	-0.054 (0.084)	-0.046 (0.086)	-0.037 (0.081)	-0.035 (0.079)

Figure 13: Table of coefficient estimates (and standard errors) of covariates with the top 10 magnitudes of estimates via $NIMIWAE_{sup}$, from fitting a logistic regression model with imputed datasets from each method. Results from multiple imputation methods NIMIWAE, IMIWAE, and MICE (first 6 columns) are based on 50 multiply imputed datasets, and reflect pooled coefficient estimates and standard errors using Rubin's rules. For fair comparison, we also included results from single imputation methods (last 5 columns). Here, IMIWAE is the ignorable version of NIMIWAE.

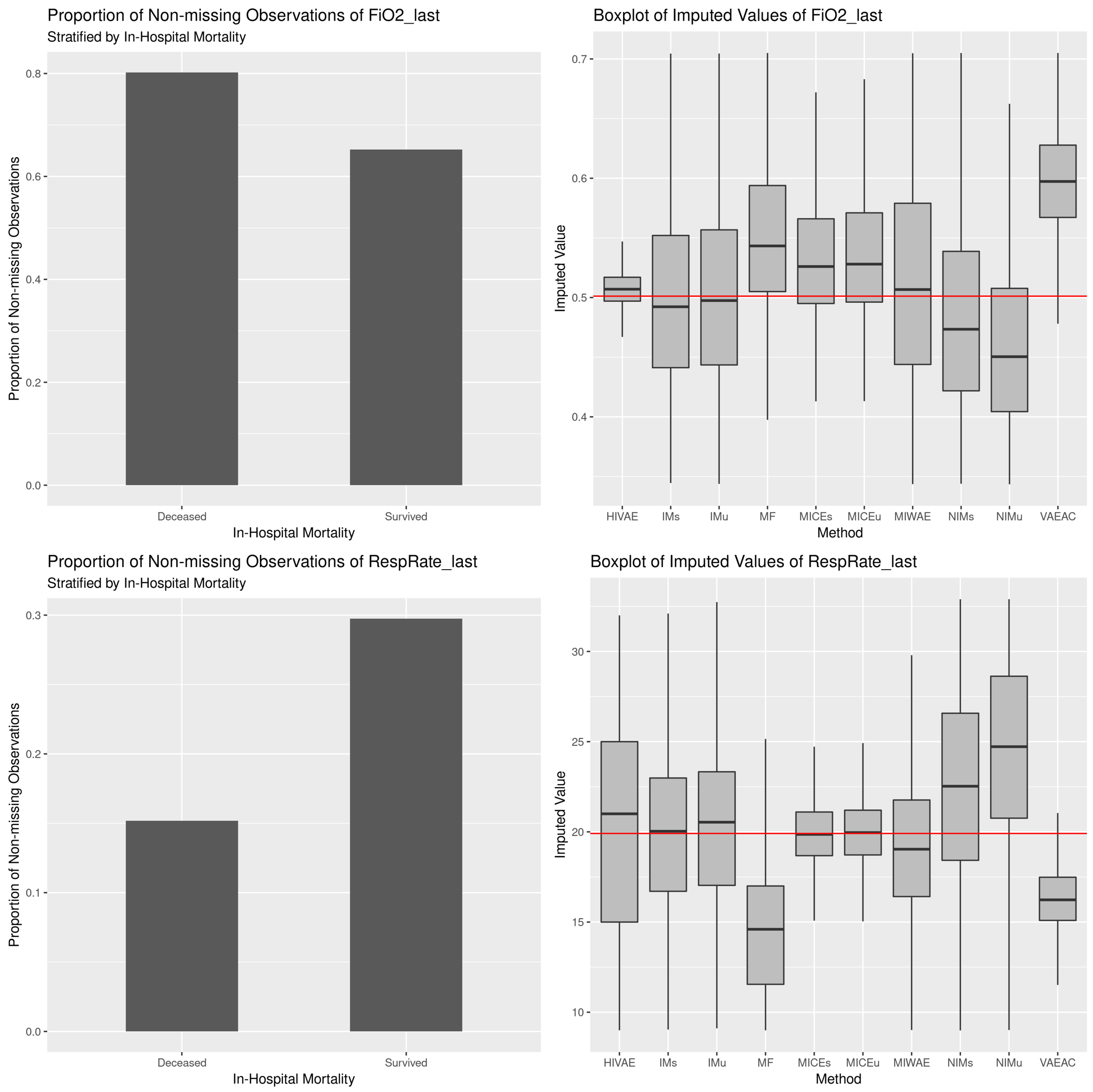


Figure 14: (left) Proportion of non-missing observations of last measurements of $FiO2$ (top) and $RespRate$ (bottom) in surviving and deceased ICU patients, and (right) imputed values of non-missing entries by HIVAE, supervised and unsupervised versions of the ignorable NIMIWAE (IMs, IMu), MissForest (MF), supervised and unsupervised versions of MICE (MICEs, MICEu), MIWAE, supervised and unsupervised versions of NIMIWAE (NIMs, NIMu), and VAEAC. The mean of the observed values is given by the red horizontal line.

and $NIMIWAE_{unsup}$ imputed values were generally smaller than those from other methods. We also found that the supervised and unsupervised $IMIWAE$ models yielded similar values to $HIVAE$, $MIWAE$, and $MICE$. These are all ignorably-missing methods, and may not impute accurate values when the missingness is MNAR. MissForest and VAEAC imputed values of FiO2 that were significantly higher than the observed mean, and values of RespRate that were significantly lower than the observed mean. Some studies have shown that abnormally high or low respiratory rates may be associated with higher mortality in critically-ill patients (Strauß et al., 2014; Ljunggren et al., 2016), suggesting that the missing values of $RespRate\_last$, which were more prevalent in deceased patients, may have truly been further away from the normal range of 12-20. Additionally, Esteban (2002) found that higher levels of administered FiO2 were associated linearly with higher mortality. Thus, the missing values of $FiO2\_last$, which were more prevalent in surviving patients, may have been lower than the observed values. This suggests that the mechanism of missingness of $FiO2\_last$ and $RespRate\_last$ may be non-ignorable, since $NIMIWAE$ imputed more realistic values for these variables according to the mortality rates of patients with missingness in these variables.

3.3 Discussion

In this chapter we introduce NIMIWAE, one of the first methods to handle up to MNAR patterns of missingness in the VAE/IWAE class of methods, to address complex patterns of missingess observed in the Physionet EHR data. Using statistical simulations, we show that NIMIWAE performs well in imputing missing features under MNAR, and has reasonable performance under the MCAR or MAR settings. Performance in imputing MCAR and MAR missingness can be further improved in NIMIWAE by using the ignorable version of this model (IMIWAE), where we omit the missingness network. We also found that the results of the analysis on the Physionet data are highly dependent on the choice of missingness model, which specifies the assumption of the missingness mechanism. However, NIMIWAE is able to impute missing values well in simulations regardless of the underlying missingness mechanism, flexibly modelling the mechanism using a deeply-learned neural network. The NIMIWAE-imputed dataset resulted in more realistic imputed values with respect to what we may expect in the Physionet Challenge patients, since NIMIWAE takes into account possible non-ignorable missingness in the data. Additionally, the IWAE architecture learns a lower-dimensional representation of the data, which can be used for further tasks, such as patient subgroup identification or visualization of data.

Learning algorithms that can be applied to EHR data like the Physionet Challenge dataset can be valuable tools that clinicians can use to aid decisions in hospital settings and understand patterns within these health records. For example, properly handling missingness in EHRs when imputing the missing entries can improve the performance of prediction algorithms that can assess risk of death or other outcomes of interest, like disease. Informative missingness is a common problem in analyzing EHR data, and accounting for such missingness can be helpful in obtaining accurate, unbiased estimates of the true missing values. We note that although we have used our NIMIWAE method primarily to analyze the Physionet 2012 Challenge dataset, it can more generally be applied to settings where one wishes to train a VAE when missingness is present among input features.

CHAPTER 4: SUPERVISED DEEP LEARNING WITH MISSING DATA

In the previous chapter, we discussed *NIMIWAE*, a novel method to handle different mechanisms of missing data for the purposes of unsupervised learning, like imputation, dimension reduction, and representation learning using an IWAE. In this chapter, we present *dlglm*: a deep generalized linear model (GLM) for probabilistic supervised learning in the presence of missing input features and/or response variables across a variety of missingness patterns. Our proposed method utilizes variational inference to learn approximate posterior distributions for the missing variables, and replaces missing entries with samples from these distributions during maximization. In this way, *dlglm* can perform regression and classification tasks in the presence of missingness in both the features and the response of interest among the training samples. We also incorporate a model for the missingness, which can take into account MNAR patterns of missingness, even at training time. Through neural networks, *dlglm* is able to model complex non-linear relationships between the input features and the response, and is scalable to large quantities and dimensionality of data. After training the *dlglm* architecture, prediction can be perfomed on fully- or partially-observed test samples using the trained model, without requiring separate imputation of the missing values.

4.1 Methods

Here we first discuss the formulation of the generalized linear model (GLM) in Section 4.1.1, and then introduce the deeply-learned GLM in Section 4.1.2. We then discuss missingness in the context of GLMs in Section 4.1.3, and lastly propose a novel deep learning architecture *dlglm* in Section 4.1.4 to fit deeply learned GLMs in the presence of missingness.

4.1.1 Generalized Linear Models (GLMs)

Let $\mathbf{X}$ be the $n \times p$ matrix of covariates (input features) with observation vectors $\mathbf{x}_i$, where each corresponding entry x_{ij} denotes the value of the i^{th} observation of the j^{th} feature for $i = 1, \ldots, n$ and $j = 1, \ldots, p$. Also, let $\mathbf{Y} = \{y_1, \ldots, y_n\}$ be the vector of univariate responses where y_i is the response pertaining to the i^{th} observation. We note that y_i may also be assumed to be multivariate; however, we focus specifically on the case of univariate response to simplify the discussion, and discuss extensions to the setting of multivariate response in Section 4.3. Then, denote $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is a vector of regression coefficients and $\boldsymbol{\eta}$ is the linear predictor. Also define $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_n\}$ with $\mu_i = E(y_i|\mathbf{x}_i)$ and link function $g(\cdot)$ such that $g(\mu_i) = \eta_i = \mathbf{x}_i\boldsymbol{\beta}$. We assume that the conditional distribution $p(y_i|\mathbf{x}_i)$ is a member of the exponential family of distributions (McCullagh and Nelder, 2019), such that $p(y_i|\mathbf{x}_i)$ can be written as

$$p(y_i|\mathbf{x}_i) = \exp\left[\frac{y_i\Theta_i - b(\Theta_i)}{a(\alpha)} + c(y_i, \alpha)\right],$$

with canonical parameter Θ_i, dispersion parameter α, and some functions $a(\cdot)$, $b(\cdot)$, and $c(\cdot)$. Here, we further assume $g(\cdot)$ is a canonical link function such that $g(\mu_i) = \Theta_i$. With the appropriate specification of the canonical link $g(\cdot)$ and variance function $V_\alpha(\cdot)$, we obtain the formulation of a GLM.

GLMs were first motivated by the limitations of the traditional linear model, which imposed strict assumptions of linearity between $\boldsymbol{\mu}$ and $\mathbf{X}$ and of normality of errors with fixed variance. GLMs instead utilize specific link and variance functions, allowing for model fitting on types of response data that may violate these assumptions, such as count or categorical outcomes, without having to rely on heuristic transformations of the data (Nelder and Wedderburn, 1972). Typically, GLMs are estimated by utilizing iteratively re-weighted least squares in lower dimensions (Holland and Welsch, 1977), with extensions to the higher dimensional case via penalized likelihood (Friedman et al., 2010).

4.1.2 Deeply Learned GLMs

The traditional GLM assumes $g(\mu_i)$ is a linear function of $\mathbf{x}_i$, i.e. $g(\mu_i) = \mathbf{x}_i\boldsymbol{\beta}$. In many modern applications, one may wish to model $g(\mu_i)$ as a non-linear function of $\mathbf{x}_i$ or capture complex interactions between features to predict response (Qi and Wu, 2003). In such cases, we may generalize the GLM to a deeply-learned GLM (Tran et al., 2019) with the following expression: $g(\mu_i) = \eta_i = h_\pi(\mathbf{x}_i)\boldsymbol{\beta}$, where $h_\pi(\cdot)$ denotes the output of a series of non-linear transformations applied to the input $\mathbf{X}$ by a neural network, with weights and bias parameters denoted by π. In addition, η_i can alternatively be expressed $\eta_i = s_{\pi,\boldsymbol{\beta}}(\mathbf{x}_i)$, where $s_{\pi,\boldsymbol{\beta}}(\cdot)$ is a neural network where $\boldsymbol{\beta}$ denotes the weights and bias associated with the output (last) layer of $s_{\pi,\boldsymbol{\beta}}(\cdot)$. This formulation allows for the traditional interpretation of $\boldsymbol{\beta}$ as the coefficients pertaining to a transformed version of the input covariates. Figure 15 shows an illustration of this architecture.

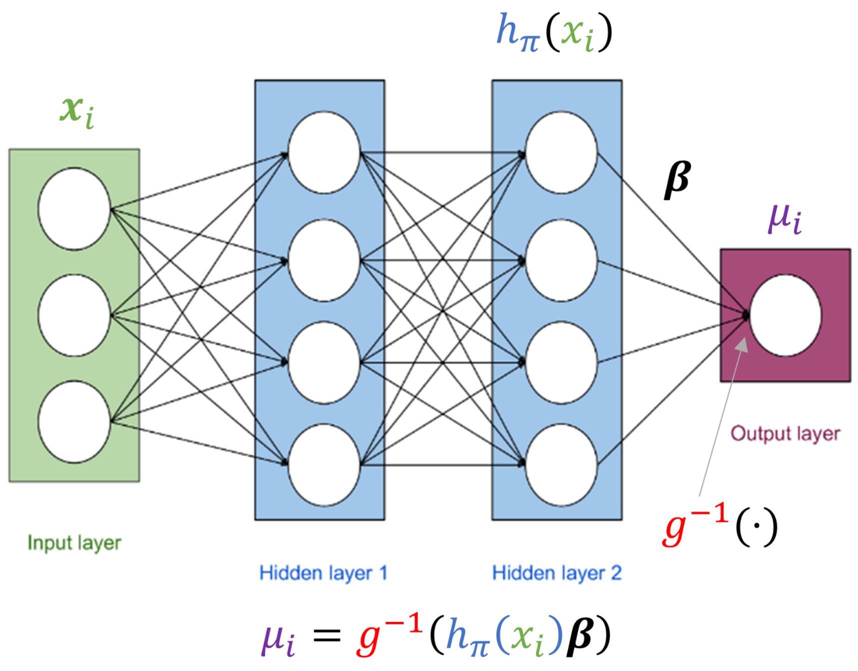


Figure 15: Visualization of a sample deeply-learned GLM architecture $s_{\pi,\boldsymbol{\beta}}(\mathbf{x}_i)$. Here, π denotes the set of weights and biases pertaining to the portion of the architecture from the input layer to the second to last layer (hidden layer 2). $h_\pi(\mathbf{x}_i)$ is a subset of the entire architecture, such that $s_{\pi,\boldsymbol{\beta}}(\mathbf{x}_i) = h_\pi(\mathbf{x}_i)\boldsymbol{\beta}$. Original artwork of a FFNN (Dormehl, 2019) was modified to show deeply-learned GLM architecture.

Let n_{HL} denote the number of hidden layers in $s_{\pi,\boldsymbol{\beta}}(\cdot)$. We note that if $n_{HL} = 0$, then $h_\pi(\mathbf{x}_i) = \mathbf{x}_i$ and $s_{\boldsymbol{\beta}}(\mathbf{x}_i) = \mathbf{x}_i\boldsymbol{\beta}$, reducing to the traditional GLM. In this setting, $\boldsymbol{\beta}$ pertains to the intercept and regression coefficients. Deeply learned GLMs and other neural networks are often maximized using stochastic gradient descent (Bottou, 2012).

In most deep learning architectures, stochastic gradient descent (SGD) is the favored algorithm of optimization, as it is very scalable to higher dimensions. A typical SGD algorithm proceeds as follows: let Q be an objective function to be maximized and let Ω denote the collection of all parameters one wishes to maximize over. Also let $Q^{(t)}$ denote the value of the objective function and $\hat{\Omega}^{(t)}$ denoting the estimate of Ω at update step t. Then, one can optimize Ω with respect to Q by some update rule $\hat{\Omega}^{(t+1)} = \hat{\Omega}^{(t)} + \delta\nabla_\Omega Q^{(t)}$, where δ is the step size which controls how large of a change is applied to the update, and the gradient ∇ is taken with respect to each parameter in Ω. In this way, Ω is updated at each step t such that $Q^{(t)}$ is increased, and the magnitude of the change in $\hat{\Omega}^{(t)}$ is mediated by δ. There are many variants of SGD, including ADAM (Kingma and Ba, 2014), ADMM (Boyd et al., 2011), Adagrad (Lydia and Francis, 2019), and more, as well as a natural gradient variational approximation with factor covariance method as discussed by Tran et al. (2019). In this chapter, we utilize ADAM as the default optimizer.

4.1.3 Missingness in GLMs

Modern biomedical datasets often contain complex forms of missingness (Ghorbani and Zou, 2018). In GLMs, missingness can exist in either $\mathbf{X}$ or $\mathbf{Y}$. Therefore, we specify three cases of missingness in this context: missing covariates with fully-observed response (*Case x*), missing response with fully-observed covariates (*Case y*), and missing covariates and missing response (*Case xy*). Let $\mathbf{R} = \{\mathbf{R}^X, \mathbf{R}^Y\}$, such that $\mathbf{R}^X$ and $\mathbf{R}^Y$ denote the missingness mask of $\mathbf{X}$ and $\mathbf{Y}$. Additionally, let $\mathbf{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$ with $\mathbf{r}_i = \{\mathbf{r}_i^X, r_i^Y\} = \{r_{i1}^X, \ldots, r_{ip}^X, r_i^Y\}$, $\mathbf{R}^X = \{\mathbf{r}_1^X, \ldots, \mathbf{r}_n^X\}$, and $\mathbf{R}^Y = \{r_1^Y, \ldots, r_n^Y\}$ with elements $\mathbf{r}_i^X$ and r_i^Y pertaining to the missingness of the i^{th} observation of $\mathbf{X}$ and $\mathbf{Y}$, respectively. Then, $\mathbf{X}$ and $\mathbf{Y}$ can be factored into the unobserved and observed entries $\{\mathbf{X}^m, \mathbf{X}^o\}$ and $\{\mathbf{Y}^m, \mathbf{Y}^o\}$, respectively, such that $\mathbf{X}^m = \{\mathbf{X} : \mathbf{R}^X = 0\}$ with $\mathbf{x}_i^m = \{\mathbf{x}_i : \mathbf{r}_i^X = 0\}$, $\mathbf{X}^o = \{\mathbf{X} : \mathbf{R}^X = 1\}$ with $\mathbf{x}_i^o = \{\mathbf{x}_i : \mathbf{r}_i^X = 1\}$, and $\mathbf{Y}^m = \{\mathbf{Y} : \mathbf{R}^Y = 0\}$ and $\mathbf{Y}^o = \{\mathbf{Y} : \mathbf{R}^Y = 1\}$, with $y_i^m = \{y_i : r_i^Y = 0\}$ and $y_i^o = \{y_i : r_i^Y = 1\}$.

Missingness was classified into three primary mechanisms in the seminal work by Little and Rubin (2002): missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). They satisfy similar relations to the missingness in Section 3.1.4:

- MCAR: $p(\mathbf{r}_i|\mathbf{x}_i, y_i, \boldsymbol{\phi}) = p(\mathbf{r}_i|\boldsymbol{\phi})$
- MAR: $p(\mathbf{r}_i|\mathbf{x}_i, y_i, \boldsymbol{\phi}) = p(\mathbf{r}_i|\mathbf{x}_i^o, y_i^o, \boldsymbol{\phi})$
- MNAR: $p(\mathbf{r}_i|\mathbf{x}_i, y_i, \boldsymbol{\phi}) = p(\mathbf{r}_i|\mathbf{x}_i^o, \mathbf{x}_i^m, y_i^o, y_i^m, \boldsymbol{\phi})$,

where $\boldsymbol{\phi}$ denotes the collection of parameters for the model of the missingness mask $\mathbf{r}_i$. In the presence of missingness, the marginal log-likelihood can generally be written as

$$\begin{aligned}\log p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{X}^o, \mathbf{Y}^o, \mathbf{R}) &= \log \iint p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{X}, \mathbf{Y}, \mathbf{R}) d\mathbf{X}^m d\mathbf{Y}^m \\ &= \log \iint p_{\alpha,\boldsymbol{\beta},\pi}(\mathbf{Y}|\mathbf{X}) p_\psi(\mathbf{X}) p_\phi(\mathbf{R}|\mathbf{X}, \mathbf{Y}) d\mathbf{X}^m d\mathbf{Y}^m, \end{aligned} \tag{4.23}$$

Under MNAR, it is not possible to remove $p_\phi(\mathbf{R}|\mathbf{X}, \mathbf{Y})$ from the integral, since $\mathbf{R}$ can depend on $\{\mathbf{X}^m, \mathbf{Y}^m\}$. The binomial model for this missing data mechanism discussed in Section 3.1.4.3 can be updated in the GLM context as

$$p(\mathbf{R}|\mathbf{X},\mathbf{Y},\boldsymbol{\phi}_{j_m}) = \prod_{i=1}^{n}\prod_{j_m=1}^{p_{miss}} [p(r_{ij_m}=1|\mathbf{x}_i,y_i,\boldsymbol{\phi}_{j_m})]^{r_{ij_m}} \left[1-p(r_{ij_m}=1|\mathbf{x}_i,y_i,\boldsymbol{\phi}_{j_m})\right]^{1-r_{ij_m}},$$

where $j_m = 1,\ldots,p_{miss}$ indexes the p_{miss} features in $\{\mathbf{X},\mathbf{Y}\}$ that contain missingness. Here $p_{miss} = p_{miss}^X + p_{miss}^Y$, where p_{miss}^X is the total number of features containing missingness in $\mathbf{X}$, and p_{miss}^Y is 1 if $\mathbf{Y}$ contains missingness (0 otherwise). Also, $\boldsymbol{\phi}_{j_m}$ is the set of coefficients pertaining to the missingness model of the j_m^{th} missing variable, and $p(r_{ij_m}=1|\mathbf{x}_i,y_i,\boldsymbol{\phi}_{j_m})$ can be modeled straightforwardly by a logistic regression model, such that

$$\text{logit}[p(r_{ij_m}=1|\mathbf{x}_i,y_i,\boldsymbol{\phi}_{j_m})] = \phi_{0j_m} + y_i\phi_{1j_m} + \mathbf{x}_i^o\boldsymbol{\phi}_{2j_m} + \mathbf{x}_i^m\boldsymbol{\phi}_{3j_m},$$

where ϕ_{0j_m} is the intercept of the j_m^{th} missingness model, ϕ_{1j_m} is the coefficient pertaining to the response variable $\mathbf{Y}$, and $\boldsymbol{\phi}_{2j_m} = \{\phi_{2,j_m,1},\ldots,\phi_{2,j_m,p_{obs}^X}\}^T$ and $\boldsymbol{\phi}_{3j_m} = \{\phi_{3,j_m,1},\ldots,\phi_{3,j_m,p_{miss}^X}\}^T$ are the sets of coefficients of the j_m^{th} variable's missingness model pertaining to the effects of the observed and missing features on the missingness, respectively, with p_{obs}^X and p_{miss}^X denoting the number of completely-observed and partially observed features in $\mathbf{X}$, respectively.

When the missingness is ignorable, the marginal log-likelihood can again be factored similarly as before, such that $\log p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{X}^o,\mathbf{Y}^o,\mathbf{R}) = \log p_{\alpha,\boldsymbol{\beta},\pi,\psi}(\mathbf{X}^o,\mathbf{Y}^o) + \log p_\phi(\mathbf{R}|\mathbf{X}^o,\mathbf{Y^o})$. Equation (4.23) can then be simplified as

$$\log p_{\alpha,\boldsymbol{\beta},\pi,\psi}(\mathbf{X}^o,\mathbf{Y}^o) = \log \iint p_{\alpha,\boldsymbol{\beta},\pi,\psi}(\mathbf{X},\mathbf{Y})d\mathbf{X}^m d\mathbf{Y}^m = \log p_{\alpha,\boldsymbol{\beta},\pi,\psi}(\mathbf{X}^o,\mathbf{Y}^o). \quad (4.24)$$

This simpler form does not involve $\mathbf{R}$, as in the ignorable version in 3.1.4.1, since the missingness is ignorable and the mask need need not be explicitly modelled.

4.1.4 Deeply-learned GLM with Missingness (*dlglm*)

In this section, we propose an algorithm for training deeply-learned GLMs in the presence of MCAR, MAR, and MNAR missingness. Before discussing this model, we first discuss the specification of the so-called covariate distribution $p_\psi(\mathbf{X})$ introduced in Equations 4.23 and 4.24, which is critical for maximizing the marginal log-likelihood in either setting. In Sections 4.1.4.1-4.1.4.2, we discuss two different models for

$p_\psi(\mathbf{X})$, and then in Section 4.1.4.3 we propose a novel method to handle missingness using a deeply-learned GLM architecture with an Importance-Weighted Autoencoder (IWAE) covariate structure. To simplify the discussion, we narrow the scope of our discussion to the *Case x* setting, where only $\mathbf{X}$ contains missingness, but note that the proposed methodology naturally extends to *Case y* and *Case xy* settings as well.

4.1.4.1 Modeling $p_\psi(\mathbf{X})$ with known distribution

Given Eq. 4.23, we must model $\mathbf{X}$ with some assumed covariate distribution $p_\psi(\mathbf{X})$. Care must be taken in specifying this distribution, as improper specification may reduce the accuracy of estimation of the parameters of interest $\boldsymbol{\beta}$ (Lipsitz and Ibrahim, 1996). For example, we may assume $p_\psi(\mathbf{X})$ follows some known multivariate distribution such as the multivariate normal, where $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\psi = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. Here, ψ can be optimized jointly with the rest of the parameters $\{\alpha, \boldsymbol{\beta}, \pi, \phi\}$ that are involved in the marginal log-likelihood. However, this assumption may not be applicable in many instances such as in the case when $\mathbf{X}$ contains mixed data types, where both continuous and discrete features may be correlated and a joint distribution may be difficult to specify in closed form. In certain cases, it may be beneficial to model $p_\psi(\mathbf{X})$ flexibly, such that no strong prior assumptions need to be made on the form of this distribution. To address this, a sequence of 1-D conditionals have previously been proposed to model the covariate distribution (Lipsitz and Ibrahim, 1996), but such a model may be computationally intractable when the number of covariates is very large. Still, if an explicit form for the covariate distribution can be specified, a lower bound of the marginal log-likelihood in the presence of missingness, as introduced in Section 4.1.3, can be derived. In this special case, a lower bound on the marginal log-likelihood in the presence of missingness can be computed under MNAR. This quantity, which we call $\mathcal{L}_K^{dlglmX}$, can be derived as

$$
\begin{aligned}
&\log p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{X}^o, \mathbf{Y}, \mathbf{R}^X) = \sum_{i=1}^{n} \log p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^o, y_i, \mathbf{r}_i^X) \\
&= \sum_{i=1}^{n} \log \left[\int p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_i^m, y_i, \mathbf{r}_i^X) d\mathbf{x}_i^m \right] \\
&= \sum_{i=1}^{n} \log \mathbb{E}_{\mathbf{x}_{ik}^m \sim q_\theta(\mathbf{x}_i^m|\mathbf{x}_i^o,\mathbf{r}_i^X,y_i)} \left[\frac{1}{K} \sum_{k=1}^{K} \frac{p_\psi(\mathbf{x}_i^o, \mathbf{x}_{ik}^m) p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \mathbf{x}_{ik}^m) p_\phi(\mathbf{r}_i^X|\mathbf{x}_i^o, \mathbf{x}_{ik}^m, y_i)}{q_\theta(\mathbf{x}_{ik}^m|\mathbf{x}_i^o, \mathbf{r}_i^X, y_i)} \right] \\
&\geq \sum_{i=1}^{n} \mathbb{E}_{\mathbf{x}_{ik}^m \sim q_\theta(\mathbf{x}_i^m|\mathbf{x}_i^o,\mathbf{r}_i^X,y_i)} \log \left[\frac{1}{K} \sum_{k=1}^{K} \frac{p_\psi(\mathbf{x}_i^o, \mathbf{x}_{ik}^m) p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \mathbf{x}_{ik}^m) p_\phi(\mathbf{r}_i^X|\mathbf{x}_i^o, \mathbf{x}_{ik}^m, y_i)}{q_\theta(\mathbf{x}_{ik}^m|\mathbf{x}_i^o, \mathbf{r}_i^X, y_i)} \right] = \mathcal{L}_K^{dlglmX}.
\end{aligned}
$$

Here, we assume $\mathbf{x}_{i1}^m, \ldots, \mathbf{x}_{iK}^m \overset{i.i.d}{\sim} q_\theta(\mathbf{x}_i^m|\mathbf{x}_i^o, \mathbf{r}_i^X, y_i)$, except now we have just the latent variables pertaining to the missing data, the weights and biases of the neural network that learns the variational posterior is now $\{\theta_1, \theta_2\} \to \theta$, and we need not specify a form for either the prior $p(\mathbf{z}_{ik})$ or the variational joint posterior which includes the the latent variables pertaining to the lower-dimensional representation $\mathbf{z}_{i1}, \ldots, \mathbf{z}_{iK}$. Instead, we assume an explicit covariate distribution $p_\psi(\mathbf{x}_i^o, \mathbf{x}_{ik}^m)$, which is indexed by some parameters ψ. For example, if $p_\psi(\mathbf{x}_i^o, \mathbf{x}_{ik}^m)$ is assumed to be distributed multivariate normal with an independent covariance structure, then $\psi = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, where $\boldsymbol{\mu}$ is a mean vector of length p, and $\boldsymbol{\Sigma}$ is a $p \times p$ covariance matrix. One can then optimize values of ψ in conjunction with the weights and biases of the neural network architecture via stochastic gradient descent during training.

Under ignorable missingness, the analogous lower bound can be obtained by assuming independence between the missing values $\mathbf{X}^m$ and the missingness mask $\mathbf{R}^X$:

$$\mathcal{L}_K^{idlglmX} = \sum_{i=1}^n \mathbb{E}_{\mathbf{x}_{ik}^m \sim q_\theta(\mathbf{x}_i^m|\mathbf{x}_i^o, y_i)} \log \left[\frac{1}{K} \sum_{k=1}^K \frac{p_\psi(\mathbf{x}_i^o, \mathbf{x}_{ik}^m) p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \mathbf{x}_{ik}^m) p_\phi(\mathbf{r}_i^X|\mathbf{x}_i^o, y_i)}{q_\theta(\mathbf{x}_{ik}^m|\mathbf{x}_i^o, y_i)} \right].$$

4.1.4.2 Modelling $p_\psi(\mathbf{X})$ with VAEs and IWAEs

Alternatively, one can approximately learn $p_\psi(\mathbf{X})$ from the training data by using an IWAE neural network architecture. In Section 3.1, we discussed in-depth the general form of the variational autoencoder (VAE) and IWAE in the case of completely-observed data $\mathbf{X}$. By having a VAE/IWAE structure for the covariate distribution, we allow for flexible modelling of this distribution, as no closed-form assumption is imposed on $p(\mathbf{X})$. Additionally, we allow for $\mathbf{X}$ to contain features of mixed data types, capturing the correlation across the different features via the learned latent structure in a VAE/IWAE. We apply the IWAE covariate structure to the deeply-learned GLM setting in Section 4.1.4.3 and show how this representation naturally extends to the case where MCAR, MAR, or MNAR missingness is observed in $\mathbf{X}$ when training deeply-learned GLMs.

4.1.4.3 dlglm: Modeling X in the presence of missingness

Now, we extend the above framework to the deeply-learned GLM framework, where features within $\mathbf{X}$ are partially observed during training. We formally introduce the *dlglm* model to handle MNAR missingness in the context of deeply-learned GLMs, as well as a variant of *dlglm* to specifically handle MCAR and MAR missingness.

Let us define $q_\theta(\mathbf{Z}, \mathbf{X}^m)$ as the variational joint posterior pertaining to $(\mathbf{Z}, \mathbf{X}^m)$. We can factor this variational joint posterior as $q_\theta(\mathbf{Z}, \mathbf{X}^m) = q_{\theta_1}(\mathbf{Z}|\mathbf{X}^o)q_{\theta_2}(\mathbf{X}^m|\mathbf{Z}, \mathbf{X}^o, \mathbf{R})$. Here, for $k = 1, \ldots, K$, we assume $\mathbf{Z}_k \overset{i.i.d}{\sim} q_{\theta_1}(\mathbf{Z}|\mathbf{X}^o)$ similar to an IWAE, and additionally assume $\mathbf{X}_k^m \overset{i.i.d}{\sim} q_{\theta_2}(\mathbf{X}^m|\mathbf{Z}, \mathbf{X}^o, \mathbf{R})$, where each $\mathbf{X}_k^m$ has dimensionality p_{miss}^X. We utilize the class of factored variational posteriors, such that $q_\theta(\mathbf{Z}, \mathbf{X}^m) = \prod_{i=1}^n q_\theta(\mathbf{z}_i, \mathbf{x}_i^m)$ and $q_\theta(\mathbf{z}_i, \mathbf{x}_i^m) = q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)q_{\theta_2}(\mathbf{x}_i^m|y_i, \mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i^X)$, with $\theta = \{\theta_1, \theta_2\}$. Then, denoting $\mathbf{z}_{ik}$ and $\mathbf{x}_{ik}^m$ as the i^{th} observation vectors of $\mathbf{Z}_k$ and $\mathbf{X}_k^m$, respectively, we have $\mathbf{z}_{i1}, \ldots, \mathbf{z}_{iK} \overset{i.i.d}{\sim} q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)$ and $\mathbf{x}_{i1}^m, \ldots, \mathbf{x}_{iK}^m \overset{i.i.d}{\sim} q_{\theta_2}(\mathbf{x}_i^m|y_i, \mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i^X)$. In this case, the lower bound, which we call the "dlglm bound", can be derived as follows:

$$\begin{aligned}
\log p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{X}^o, \mathbf{Y}, \mathbf{R}^X) &= \sum_{i=1}^n \log p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^o, y_i, \mathbf{r}_i^X) \\
&= \sum_{i=1}^n \log \left[\iint p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_i^m, y_i, \mathbf{r}_i^X, \mathbf{z}_i) d\mathbf{z}_i d\mathbf{x}_i^m \right] \\
&= \sum_{i=1}^n \log \mathbb{E}_{(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m) \sim q_\theta(\mathbf{z}_i, \mathbf{x}_i^m)} \left[\frac{1}{K} \sum_{k=1}^K \frac{p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_{ik}^m, y_i, \mathbf{r}_i^X, \mathbf{z}_{ik})}{q_\theta(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m)} \right] \\
&\geq \sum_{i=1}^n \mathbb{E}_{(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m) \sim q_\theta(\mathbf{z}_i, \mathbf{x}_i^m)} \log \left[\frac{1}{K} \sum_{k=1}^K \frac{p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_{ik}^m, y_i, \mathbf{r}_i^X, \mathbf{z}_{ik})}{q_\theta(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m)} \right] = \mathcal{L}_K^{dlglm}, \qquad (4.25)
\end{aligned}$$

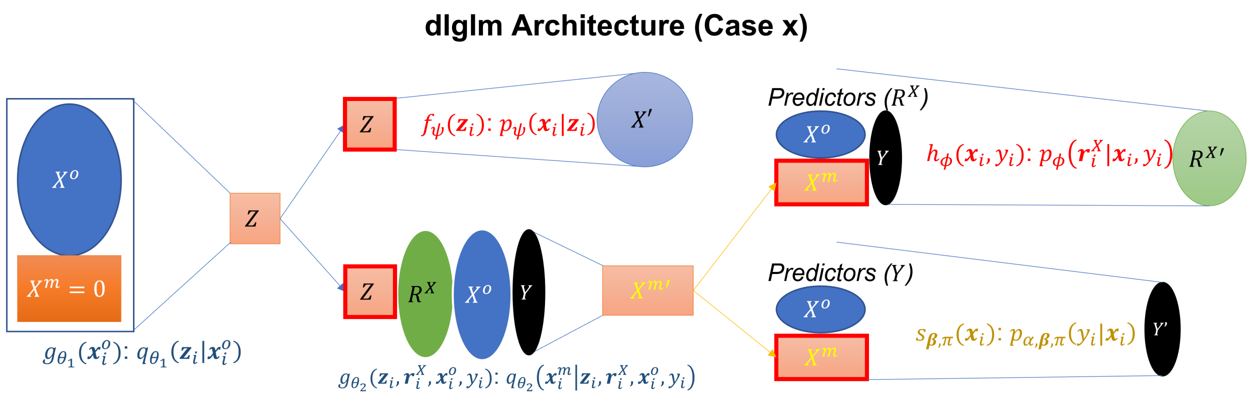
Here, $\{\psi, \boldsymbol{\beta}, \pi, \phi, \theta\}$ are the weights and biases associated with the neural networks that output the parameters of the distributions that are involved, α is the dispersion parameter associated with the variance function of $\mathbf{Y}$, and $\tilde{\mathbf{z}}_{ik}$ and $\tilde{\mathbf{x}}_{ik}^m$ are the samples drawn from $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)$, and $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, y_i, \mathbf{r}_i^X)$, respectively.

Again, we use the selection model factorization of the complete data log-likelihood, such that

$$p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_i^m, y_i, \mathbf{r}_i^X, \mathbf{z}_i) = p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i)p_\psi(\mathbf{x}_i|\mathbf{z}_i)p(\mathbf{z}_i)p_\phi(\mathbf{r}_i^X|\mathbf{x}_i, y_i).$$

Applying this factorization to (4.25), we obtain the form of the estimate of the "dlglm bound", where the integral is estimated via Monte Carlo integration:

$$\hat{\mathcal{L}}_K^{dlglm} = \sum_{i=1}^n \log \left[\frac{1}{K} \sum_{k=1}^K \frac{p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)p_\psi(\mathbf{x}_i|\tilde{\mathbf{z}}_{ik})p(\tilde{\mathbf{z}}_{ik})p_\phi(\mathbf{r}_i^X|y_i, \mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o)q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, y_i, \mathbf{r}_i^X)} \right], \qquad (4.26)$$


dlglm Architecture (Case x)
X^o
$X^m = 0$
$g_{\theta_1}(x_i^o)$: $q_{\theta_1}(z_i|x_i^o)$
Z
Z
$f_\psi(z_i)$: $p_\psi(x_i|z_i)$
X'
Z
R^X
X^o
Y
$X^{m\prime}$
$g_{\theta_2}(z_i, r_i^X, x_i^o, y_i)$: $q_{\theta_2}(x_i^m|z_i, r_i^X, x_i^o, y_i)$
Predictors (R^X)
X^o
X^m
Y
$h_\phi(x_i, y_i)$: $p_\phi(r_i^X|x_i, y_i)$
$R^{X\prime}$
Predictors (Y)
X^o
X^m
$s_{\beta,\pi}(x_i)$: $p_{\alpha,\beta,\pi}(y_i|x_i)$
Y'

$$\mathcal{L}_K^{dlglm} = \sum_{i=1}^{n} \mathbb{E}_{(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m) \sim q_\theta(\mathbf{z}_i, \mathbf{x}_i^m | y_i, \mathbf{x}_i^o, \mathbf{r}_i^X)} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \mathbf{x}_{ik}^m)\, p_\psi(\mathbf{x}_i^o, \mathbf{x}_{ik}^m|\mathbf{z}_{ik})\, p_\phi(\mathbf{r}_i^X|y_i, \mathbf{x}_i^o, \mathbf{x}_{ik}^m)\, p(\mathbf{z}_{ik})}{q_{\theta_1}(\mathbf{z}_{ik}|\mathbf{x}_i^o)\, q_{\theta_2}(\mathbf{x}_{ik}^m|y_i, \mathbf{x}_i^o, \mathbf{z}_{ik}, \mathbf{r}_i^X)} \right]$$

$$\hat{\mathcal{L}}_K^{dlglm} = \sum_{i=1}^{n} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)\, p_\psi(\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik})\, p_\phi(\mathbf{r}_i^X|y_i, \mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)\, p(\tilde{\mathbf{z}}_{ik})}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o)\, q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|y_i, \mathbf{x}_i^o, \mathbf{z}_{ik}, \mathbf{r}_i^X)} \right]$$

Figure 16: Architecture of proposed dlglm method (*Case x*). Dark colored nodes represent deterministic values, lightly colored nodes represent learned distributional parameters, and outlined (in red) nodes represent sampled values. Orange cells correspond to latent variables $\mathbf{Z}$ and $\mathbf{X}^m$. $\mathbf{Z}_1, \dots, \mathbf{Z}_K$ and $\mathbf{X}_1^m, \dots, \mathbf{X}_K^m$ are sampled from their respective variational posteriors.

We see that this quantity closely resembles the lower bound of an IWAE, and, similar to traditional VAEs, we utilize neural networks $f_\psi(\mathbf{z}_i)$, $g_{\theta_1}(\mathbf{x}_i^o)$, $g_{\theta_2}(\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i^X, y_i)$, $s_{\boldsymbol{\beta},\pi}(\mathbf{x}_i)$, and $h_\phi(\mathbf{x}_i, y_i)$ to learn the values of the parameters of $p_\psi(\mathbf{x}_i|\mathbf{z}_i)$, $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)$, $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i^X, y_i)$, $p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i)$, and $p_\phi(\mathbf{r}_i^X|\mathbf{x}_i, y_i)$. The associated weighs and biases of the neural networks $\{\boldsymbol{\beta}, \pi, \psi, \phi\}$, as well as the dispersion parameter α pertaining to $p_{\alpha,\boldsymbol{\beta},\pi}(\mathbf{Y}|\mathbf{X})$ are updated using stochastic gradient descent via the ADAM optimizer (Kingma and Ba, 2014). The architecture of *dlglm* can be found in Figure 16.

The training of the *dlglm* architecture proceeds as follows:

1. The missing entries are pre-imputed to zero and appended with observed entries, and fed into $g_{\theta_1}(\mathbf{x}_i^o)$, to learn parameters of $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)$.

2. K samples are drawn from $q_{\theta_1}(\mathbf{Z}|\mathbf{X}^o, \mathbf{R})$.

3. Samples from (2) are used as input for $f_\psi(\mathbf{z}_i)$, to learn the parameters of $p_\psi(\mathbf{x}_i|\mathbf{z}_i)$.

4. The samples from (2) are used again as input for $g_{\theta_2}(\mathbf{z}_i, \mathbf{r}_i^X, \mathbf{x}_i^o, y_i)$, concatenated with the observed data entries (with pre-imputed missing entries) the missingness mask, and the response to learn parameters of $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i^X, y_i)$.

5. We draw samples of $\mathbf{x}_i^m$ from $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, \mathbf{r}_i^X, y_i)$, and use them as input, concatenated with the response y_i and the fixed observed entries $\mathbf{x}_i^o$, into $h_\phi(\mathbf{x}_i, y_i)$ (or the mask decoder network) to learn the parameters associated with the model of the missingness mask $p_\phi(\mathbf{r}_i^X|\mathbf{x}_i, y_i)$.

6. We use the samples of $\mathbf{x}_i^m$ from (5), concatenated with the observed variables $\mathbf{x}_i^o$ as input into $s_{\boldsymbol{\beta},\pi}(\mathbf{x}_i)$ to output parameters of $p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i)$. The dispersion parameter α is additionally learned via stochastic gradient descent, along with the all of the weights and biases of the entire architecture.

Under simple distributional assumptions of $q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{r}_i^X, \mathbf{x}_i^o, y_i)$, the sampling step in Step (5) is similar to the sampling of the latent variable $\mathbf{Z}$ in Step (2), and both can be accomplished using the reparametrization trick (Kingma and Welling, 2013). These steps outline the case where $\mathbf{R}^X$ is independent on $\mathbf{Z}$.

We can obtain a variant of this method, which we call ignorably-missing dlglm (*idlglm*), by assuming independence between $\mathbf{X}^m$ and $\mathbf{R}$ by omitting $\mathbf{r}_i^X$ from Equation 4.25, and removing $p_\phi(\mathbf{r}_i^X|y_i, \mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)$ and letting $p_\phi(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, y_i, \mathbf{r}_i^X) \to p_\phi(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, y_i)$ in Equation 4.26. Whereas *dlglm* is better suited to handle MNAR, *idlglm* may be more appropriate for the MCAR or MAR settings, where a missingness model need not be specified. This is analogous to IMIWAE and NIMIWAE from Section 3.1.

We limited our discussion in this chapter to *Case x*, where missingness exists only in $\mathbf{X}$ but not in $\mathbf{Y}$; however, the lower bound for *dlglm* can similarly be derived for the more general *Case xy* as well. In this case, one must additionally learn an approximate posterior distribution of the missing response $\mathbf{Y}^m$. The dlglm bound can be derived similarly to the Case x missingness as follows:

$$\begin{aligned}
\log p_{\alpha,\pi,\psi,\phi}(\mathbf{X}^o, \mathbf{Y}^o, \mathbf{R}) &= \sum_{i=1}^n \log p_{\alpha,\pi,\psi,\phi}(\mathbf{x}_i^o, y_i^o, \mathbf{r}_i) \\
&= \sum_{i=1}^n \log \left[\iiint p_{\alpha,\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_i^m, y_i^o, y_i^m, \mathbf{r}_i^X, r_i^Y, \mathbf{z}_i) d\mathbf{z}_i d\mathbf{x}_i^m dy_i^m \right] \\
&= \sum_{i=1}^n \log \mathbb{E}_{(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m, y_{ik}^m) \sim q_\theta(\mathbf{z}_i, \mathbf{x}_i^m, y_i^m)} \left[\frac{1}{K} \sum_{k=1}^K \frac{p_{\alpha,\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_{ik}^m, y_i^o, y_{ik}^m, \mathbf{r}_i^X, r_i^Y, \mathbf{z}_{ik})}{q_\theta(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m, y_{ik}^m)} \right] \\
&\geq \sum_{i=1}^n \mathbb{E}_{(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m, y_{ik}^m) \sim q_\theta(\mathbf{z}_i, \mathbf{x}_i^m, y_i^m)} \log \left[\frac{1}{K} \sum_{k=1}^K \frac{p_{\alpha,\pi,\psi,\phi}(\mathbf{x}_i^o, \mathbf{x}_{ik}^m, y_i^o, y_{ik}^m, \mathbf{r}_i^X, r_i^Y, \mathbf{z}_{ik})}{q_\theta(\mathbf{z}_{ik}, \mathbf{x}_{ik}^m, y_{ik}^m)} \right] \\
&= \mathcal{L}_K^{dlglm},
\end{aligned} \tag{4.27}$$

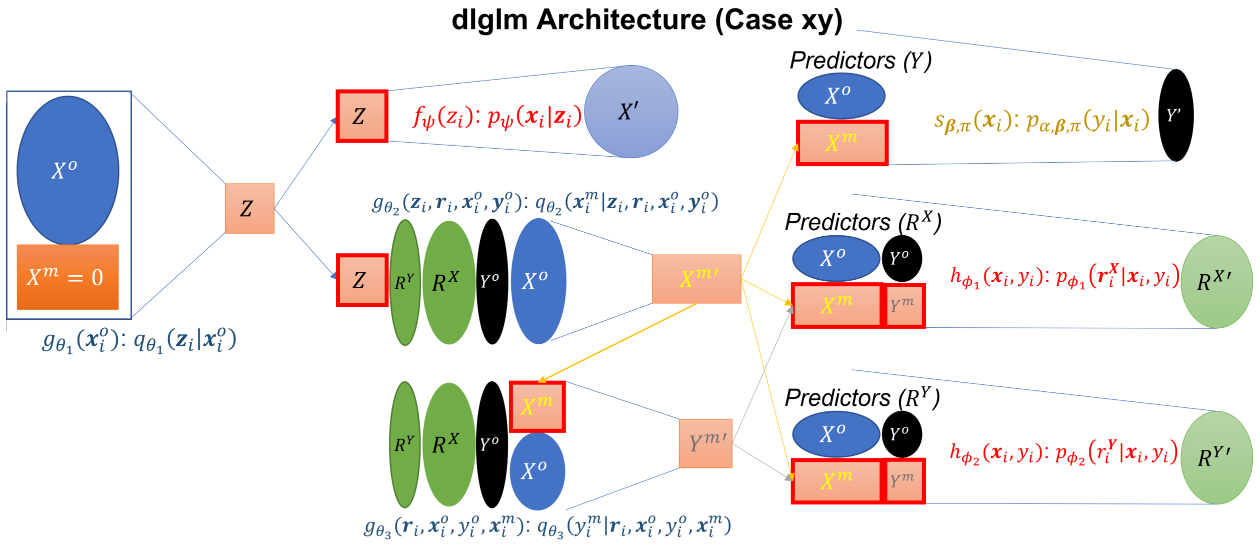
where now, we modify the factorization of the joint posterior as

$$q_\theta(\mathbf{Z}, \mathbf{X}^m, \mathbf{Y}^m) = q_{\theta_1}(\mathbf{Z}|\mathbf{X}^o) q_{\theta_2}(\mathbf{X}^m|\mathbf{Z}, \mathbf{X}^o, \mathbf{R}) q_{\theta_3}(\mathbf{Y}^m|\mathbf{X}^o, \mathbf{X}^m, \mathbf{R}).$$

Then, the estimated dlglm bound is

$$\hat{\mathcal{L}}_K^{dlglm} = \sum_{i=1}^{n} \log \left[\frac{1}{K} \sum_{k=1}^{K} \frac{p_{\alpha,\psi}(y_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m) p_\pi(\mathbf{x}_i|\tilde{\mathbf{z}}_{ik}) p(\tilde{\mathbf{z}}_{ik}) p_{\phi_1}(\mathbf{r}_i|y_i^o, \tilde{y}_{ik}^m, \mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m)}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o) q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, y_i^o, \mathbf{r}_i) q_{\theta_3}(\tilde{y}_{ik}^m|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m, y_i^o, \mathbf{r}_i)} \right]. \quad (4.28)$$

A visualization of the *dlglm* architecture in Case xy can be found in Figure 17.



dlglm Architecture (Case xy)
X^o
$X^m = 0$
$g_{\theta_1}(x_i^o)$: $q_{\theta_1}(z_i|x_i^o)$
Z
$f_\psi(z_i)$: $p_\psi(x_i|z_i)$
X'
$g_{\theta_2}(z_i, r_i, x_i^o, y_i^o)$: $q_{\theta_2}(x_i^m|z_i, r_i, x_i^o, y_i^o)$
R^Y R^X Y^o X^o
$X^{m\prime}$
X^m
$Y^{m\prime}$
$g_{\theta_3}(r_i, x_i^o, y_i^o, x_i^m)$: $q_{\theta_3}(y_i^m|r_i, x_i^o, y_i^o, x_i^m)$
Predictors (Y)
$s_{\beta,\pi}(x_i)$: $p_{\alpha,\beta,\pi}(y_i|x_i)$
Y'
Predictors (R^X)
Y^m
$h_{\phi_1}(x_i, y_i)$: $p_{\phi_1}(r_i^X|x_i, y_i)$
$R^{X\prime}$
Predictors (R^Y)
$h_{\phi_2}(x_i, y_i)$: $p_{\phi_2}(r_i^Y|x_i, y_i)$
$R^{Y\prime}$

$$\mathcal{L}_K^{dlglm} = \sum_{i=1}^{n} \mathbb{E}_{(z_{ik}, x_{ik}^m, y_{ik}^m) \sim q_\theta(z_i, x_i^m, y_i^m | y_i^o, x_i^o, r_i)} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\alpha,\beta,\pi}(y_i|x_i^o, x_{ik}^m)\, p_\psi(x_i^o, x_{ik}^m|z_{ik})\, p_\phi(r_i^X, r_i^Y|y_i^o, \tilde{y}_{ik}^m, x_i^o, x_{ik}^m)\, p(z_{ik})}{q_{\theta_1}(z_{ik}|x_i^o)\, q_{\theta_2}(x_{ik}^m|y_i^o, x_i^o, z_{ik}, r_i)\, q_{\theta_3}(y_{ik}^m|y_i^o, x_i^o, x_{ik}^m, r_i)} \right]$$

$$\hat{\mathcal{L}}_K^{dlglm} = \sum_{i=1}^{n} \left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\alpha,\beta,\pi}(y_i|x_i^o, \tilde{x}_{ik}^m)\, p_\psi(x_i^o, \tilde{x}_{ik}^m|\tilde{z}_{ik})\, p_\phi(r_i^X, r_i^Y|y_i^o, \tilde{y}_{ik}^m, x_i^o, \tilde{x}_{ik}^m)\, p(\tilde{z}_{ik})}{q_{\theta_1}(\tilde{z}_{ik}|x_i^o)\, q_{\theta_2}(\tilde{x}_{ik}^m|y_i^o, x_i^o, \tilde{z}_{ik}, r_i) q_{\theta_3}(\tilde{y}_{ik}^m|y_i^o, x_i^o, \tilde{x}_{ik}^m, r_i)} \right]$$

Figure 17: Architecture of proposed dlglm method (*Case xy*). Dark colored nodes represent deterministic values, lightly colored nodes represent learned distributional parameters, and outlined (in red) nodes represent sampled values. Orange cells correspond to latent variables $\mathbf{Z}$, $\mathbf{X}^m$, and $\mathbf{Y}^m$. $\mathbf{Z}_k$, $\mathbf{X}_k^m$, and $\mathbf{Y}_k^m$ are sampled from their respective variational posteriors for $k = 1, \ldots, K$. Below is the dlglm bound ($\mathcal{L}_K^{dlglm}$), and the estimated dlglm bound ($\hat{\mathcal{L}}_K^{dlglm}$), which is optimized via stochastic gradient descent.

In this chapter, we are primarily interested in supervised learning. However, following training, *dlglm* and *idlglm* can also perform imputation as in the unsupervised learning architecture for handling missingness, as proposed in Section 3.1, although such imputation is not necessary for training, coefficient estimation, or prediction. Following training, *dlglm* can perform single imputation of missing values by obtaining point estimates for $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, y_i, \mathbf{r}_i^X]$, defined as the expected value of the missing features given the observed data and the mask for the i^{th} observation under MNAR. We note that imputation is not the main focus of this chapter, as it is primarily an unsupervised task, but it may be of interest in some settings to obtain these imputed values for downstream analyses. To do this, we first note that

$$\begin{aligned}
\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, y_i, \mathbf{r}_i^X] &= \int \mathbf{x}_i^m p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^m|\mathbf{x}_i^o, y_i, \mathbf{r}_i^X) d\mathbf{x}_i^m \\
&= \iint \mathbf{x}_i^m p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^m, \mathbf{z}_i|\mathbf{x}_i^o, y_i, \mathbf{r}_i^X) d\mathbf{z}_i d\mathbf{x}_i^m \\
&= \iint \mathbf{x}_i^m \frac{p_{\alpha,\boldsymbol{\beta},\pi,\psi,\phi}(\mathbf{x}_i^m, \mathbf{x}_i^o, \mathbf{z}_i, y_i, \mathbf{r}_i^X)}{p_{\psi,\phi}(\mathbf{x}_i^o, y_i, \mathbf{r}_i^X)} d\mathbf{z}_i d\mathbf{x}_i^m.
\end{aligned}$$

Then, we can estimate this integral by self-normalized importance sampling. We utilize the proposal density $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, y_i, \mathbf{r}_i^X)$, and define the quantities

$$w_{ik} = \frac{s_{ik}}{s_{i1} + \ldots + s_{iK}}, \text{ and } s_{ik} = \frac{p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_i^m)p_\psi(\mathbf{x}_i|\tilde{\mathbf{z}}_{ik})p(\tilde{\mathbf{z}}_{ik})p_\phi(\mathbf{r}_i^X|\mathbf{x}_i^o, \tilde{\mathbf{x}}_{ik}^m, y_i)}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o)q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, \mathbf{r}_i^X, y_i)}$$

for $k = 1, \ldots, K$, with 1 sample drawn from the variational posterior of each latent variable $\mathbf{z}_{ik}$ and $\mathbf{x}_{ik}^m$ to compute s_{ik}, where w_{ik} is defined as standardized "importance weights" (Mattei and Frellsen, 2019). Using these weights we may estimate $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, y_i, \mathbf{r}_i^X] \approx \sum_{k=1}^K w_{ik}\tilde{\mathbf{x}}_{ik}^m$. Then, the process can be repeated for each observation $i = 1, \ldots, n$.

In the MCAR or MAR case, one can similarly estimate $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, y_i]$ using the fitted *idlglm* model. By following a similar derivation using the proposal density $q_{\theta_1}(\mathbf{z}_i|\mathbf{x}_i^o)q_{\theta_2}(\mathbf{x}_i^m|\mathbf{z}_i, \mathbf{x}_i^o, y_i)$, we obtain the same approximation $\mathbb{E}[\mathbf{x}_i^m|\mathbf{x}_i^o, y_i] \approx \sum_{k=1}^K w_{ik}\tilde{\mathbf{x}}_{ik}^m$, with w_{ik} defined as before, but with a slightly different form for s_{ik}:

$$s_{ik} = \frac{p_{\alpha,\boldsymbol{\beta},\pi}(y_i|\mathbf{x}_i^o, \tilde{\mathbf{x}}_i^m)p_\psi(\mathbf{x}_i|\tilde{\mathbf{z}}_{ik})p(\tilde{\mathbf{z}}_{ik})}{q_{\theta_1}(\tilde{\mathbf{z}}_{ik}|\mathbf{x}_i^o)q_{\theta_2}(\tilde{\mathbf{x}}_{ik}^m|\tilde{\mathbf{z}}_{ik}, \mathbf{x}_i^o, y_i)}.$$

4.2 Numerical Examples

In this section, we evaluate the performance of *dlglm* and *idlglm* to analyze each method's performance in imputation, coefficient estimation, and prediction tasks on simulated datasets under MCAR, MAR, and MNAR missingness in Section 4.2.1. We also compare our methods to two commonly used approaches for modeling missing data in the supervised setting, mean imputation and the *mice* method for multiple imputation (Van Buuren and Groothuis-Oudshoorn, 2011). To account for potential non-linearity and complex relationships between features, in Section 4.2.2, we mask completely-observed datasets obtained from the UCI Machine Learning Repository with varying mechanisms of missingness on the predictors. Finally, in Section 4.2.3, we perform prediction on the Bank Marketing dataset, which inherently contains missingness in the predictors.

Below, we summarize the combinations of hyperparameter values that were searched by *dlglm* and *idlglm*. For each dataset, we searched over combinations of two different values for each hyperparameter that was tuned. We tuned 4 different hyperparameters for each method: number of nodes per hidden layer (h), number of nodes per hidden layer in the missingness network (h_r) , number of hidden layers (nhl), number of hidden layers in the $s_{\boldsymbol{\beta},\pi}(\mathbf{x}_i)$ network (nhl_y), number of hidden layers in the missingness network (nhl_r), dimensionality of latent space (dz), and learning rate (lr). For *idlglm*, $nhl_r = 0$ and $h_r = 0$ were fixed (since these are omitted in the *idlglm* architecture). Below are the searched values of the hyperparameters for each dataset in our analyses:

- Simulated Data ($p = \{25, 50\}$ features)
  - $h = \{128, 64\}$, $h_r = \{16, 32\}$
  - $lr = \{0.001, 0.01\}$
  - $dz = \{\lfloor 3 * p/4 \rfloor, \lfloor p/2 \rfloor, \lfloor p/4 \rfloor, \lfloor p/12 \rfloor\}$
  - $nhl = \{0, 1, 2\}$
  - $nhl_y = 0$
  - $nhl_r = \{0, 1\}$
  - $bs = 1,000$, $epochs_{max} = 2002$
- All UCI (including Banknote) datasets (p features)
  - $h = \{128, 64\}$, $h_r = \{16, 32\}$
  - $lr = \{0.001, 0.01\}$
  - $dz = \{\lfloor 3 * p/4 \rfloor, \lfloor p/2 \rfloor, \lfloor p/4 \rfloor, 8\}$
  - $nhl = \{0, 1, 2\}$
  - $nhl_y = \{0, 1, 2\}$

- $nhl_r = \{0, 1\}$
- $bs = 1000$, $epochs_{max} = 2002$

4.2.1 Simulated Data

4.2.1.1 Simulation Setup

We first utilized completely synthetic data to evaluate the performance of each. Here, $\mathbf{X}$ is generated such that $\mathbf{X} = normalize(\mathbf{ZW} + \mathbf{B}) + B_0$, where $normalize(\cdot)$ takes an input matrix and standardizes each column to mean 0 and standard deviation 1, and $\mathbf{W}$ and $\mathbf{B}$ and are matrices of dimensions $d \times p$ and $n \times p$, respectively, $\mathbf{Z} \sim N_d(\mathbf{0}, \mathbf{I})$, and $W_{lj} \sim N(0, 0.5)$ and $B_{ij} \sim N(0, 1)$ for $i = 1, \ldots, n$, $p = 1, \ldots, p$, and $l = 1, \ldots, d$, and $B_0 = 2$ is fixed. We also generated a binary response variable $\mathbf{Y}$ such that Pr($\mathbf{Y} = 1|\mathbf{X}) = \beta_0 + \boldsymbol{\beta}\mathbf{X}$, where β are drawn randomly from $\{-\frac{1}{4}, \frac{1}{4}\}$, and β_0 is chosen such that approximately half of the sample are in either class. Values of $\mathbf{Y}$ are drawn from Bernoulli(Pr($\mathbf{Y} = 1|\mathbf{X}$)).

We then simulate the missingness mask matrix $\mathbf{R}^X$ such that 50% of features in $\mathbf{X}$ are partially observed, and 30% of the observations for each of these features are missing. We generate r_{ij} from the Bernoulli distribution with probability equal to $p(r_{ij_m} = 1|\mathbf{x}_i, y_i, \boldsymbol{\phi})$, such that

$$\text{logit}[p(r_{ij_m} = 1|\mathbf{x}_i, y_i, \boldsymbol{\phi})] = \phi_0 + \phi_1 y_i + \boldsymbol{\phi}_2 \mathbf{x}_i^o + \boldsymbol{\phi}_3 \mathbf{x}_i^m,$$

where $j_m = 1, \ldots, p^X_{miss}$ index the missing features, ϕ_1 is the coefficient pertaining to the response, $\boldsymbol{\phi}_2 = \{\phi_{21}, \ldots, \phi_{2,p^X_{obs}}\}$ are the coefficients pertaining to the observed features, and $\boldsymbol{\phi}_3 = \{\phi_{31}, \ldots, \phi_{3,p^X_{miss}}\}$ are those pertaining to the missing features, where p^X_{obs} and p^X_{miss} are the total number of features that are observed and missing, respectively, with $p^X_{miss} = floor(0.5 * p)$ and $p^X_{obs} = p - p^X_{miss}$. Here, we fixed $\phi_1 = 0$, and drew nonzero values of $\{\boldsymbol{\phi}_2, \boldsymbol{\phi}_3\}$ from the log-normal distribution with mean $\mu_\phi = 5$, with log standard deviation $\sigma_\phi = 0.2$.

To evaluate the impact of the misspecification of the missingness mechanism on model performance, r_{ij_m} was simulated under each mechanism as follows: (1) MCAR: $\{\phi_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3\} = 0$ (2) MAR: Same as MCAR except $\phi_{2j_o} \neq 0$ for one completely-observed feature j_o (3) MNAR: Same as MCAR except $\phi_{3j_m} \neq 0$ for one missing feature j_m. In this way, for each MAR or MNAR feature, the missingness is dependent on just one feature. In each case, we used ϕ_0 to control for an expected rate of missingness of 30% in each partially-observed feature. We note that for each these simulations, we utilize all features in $\mathbf{X}$ as well as the response $\mathbf{Y}$ as input into *dlglm*'s missingness network, although only one feature is involved under the true missingness model.

We vary n and d such that $n = \{10,000, 100,000\}$ and $d = \{2, 8\}$, and fix $p = 50$. We simulated 5 datasets per simulation condition, spanning various missingness mechanisms and values for $\{n, d\}$. We fix the values of $\boldsymbol{\beta}$ at 0.25 for each feature, and adjusted β_0 to ensure equal proportions for the binary class response $\mathbf{Y}$. For each simulation case, we partitioned the data into training, validation, and test sets with ratio 8:1:1. For *mice* imputation, we averaged across 500 multiply-imputed datasets to obtain a single imputed dataset. We note that we generated $\mathbf{Y}$ by a linear transformation of $\mathbf{X}$ in these simulations in order to facilitate fair comparisons with *mice*, which cannot account for non-linear relationships between the features and the response. Because no hyperparameter tuning is required, the validation set is not utilized for *mice* and mean imputation.

We measured the performance of each method with respect to three different tasks: imputation of missing values, coefficient estimation, and prediction. Imputation performance was measured with respect to the truth on a single imputed dataset by mean, *dlglm* and *idlglm* imputation, and on an average of multiply-imputed datasets by *mice*. Coefficient estimation for mean and *mice* were based on downstream fitted GLM(s) on these imputed dataset(s), where estimates were pooled using Rubin's rules (Rubin, 2004) for *mice*. For *dlglm* and *idlglm*, we estimated the coefficients by the weights and bias $\boldsymbol{\beta}$ of the last layer of the $s_{\boldsymbol{\beta},\pi}(\cdot)$ trained neural network. Here, we fixed the number of hidden layers in $s_{\boldsymbol{\beta},\pi}(\cdot)$ to 0 to allow for direct comparison with the other methods. A more complex prediction model via a neural network can be learned by simply incorporating additional hidden layers in $s_{\boldsymbol{\beta},\pi}(\cdot)$. We note that *dlglm* and *idlglm* can estimate $\boldsymbol{\beta}$ without having to perform multiple imputation and downstream modelling like *mice*, where fitting complex methods such as neural networks each of the multiply-imputed datasets separately may be computationally prohibitive.

After obtaining the coefficient estimates and trained models, we performed prediction on the test set in two ways: 1) using the incomplete (predI) test set, where the true values of $\mathbf{X}^m$ are not known at prediction time, and 2) using the complete (predC) test set, where the true simulated values of $\mathbf{X}^m$ are known at prediction time. These two ways reflect the two realistic cases in which (1) missingness is present during training time but complete data is available at prediction time, and (2) missingness is present during both training and prediction time. For predI, *mice* and mean imputation require an additional imputation step on the test set before predicting $\mathbf{Y}$; for *dlglm* and *idlglm*, we simply input the incomplete test set into the trained model without needing to separately impute the test set, and we predict using the trained model. That is, *mice* and mean imputation cannot generalize the trained model to impute the test set, *dlglm* and *idlglm* provide a seamless framework to utilize the already-trained model to impute and predict on a held-out test set. For predC, we use the underlying true values of $\mathbf{X}^m$ to predict on the test dataset.

Imputation error was measured by the average L1 distance between true and imputed masked values in $\mathbf{X}$. Letting $\hat{\mathbf{X}}^m$ denote the imputed masked values of the true $\mathbf{X}^m$ values of the missing entries, we denote the average L1 distance is simply $\frac{|\hat{\mathbf{X}}^m - \mathbf{X}^m|}{N_{miss}}$, where N_{miss} is the total number of missing entries in the dataset. Performance in coefficient estimation was measured by the average percent bias (PB) of the coefficient estimates compared to the truth, averaged across the p features, i.e. $PB = \frac{1}{p}\sum_{j=1}^{p} \frac{|\beta_j - \hat{\beta}_j|}{|\beta_j|}$. Finally, predC and predI prediction error was measured by the average L1 distance between predicted and true values of the probabilities of class membership Pr$(\mathbf{Y} = 1|\mathbf{X})$ in the test set.

4.2.1.2 Simulation Results

Figures 18 and 19 illustrate the simulation results pertaining to imputation accuracy, coefficient estimation, and prediction accuracy for the condition $p = 50$. We see that across all combinations of $\{n, d\}$ and mechanisms of missingness, mean imputation consistently performs poorly in imputation, coefficient estimation, and prediction, while *mice* and *idlglm* perform comparably in these metrics. Also, we note that under MNAR missingness, *dlglm* generally yields the lowest imputation and prediction error, as well as percent bias across all simulation cases. Under MAR missingness, *dlglm* performs comparably to *idlglm* and *mice*. This shows the ability of *dlglm* to learn an accurate model of the missingness, even under severe overparametrization of the missingness model (model need not be specified for ignorable missingness). However, due to the complexity of the model, we see that *dlglm* does generally perform poorly compared to *idlglm* and *mice* under MCAR missingness, when $n = 10,000$, although it still performs comparably to other methods when the sample size is very large ($n = 100,000$). As one may expect, prediction performance using the incomplete data (predI) was poorer than prediction performance using the complete data (predC) for all methods.

Figures 20 and 21 show the results of the imputation, coefficient estimation, and prediction performance on simulated data with $p = 25$ fixed. Overall, we found that *dlglm* performed best under MNAR missingness, and comparably to *idlglm* and *mice* under MCAR and MAR missingness.

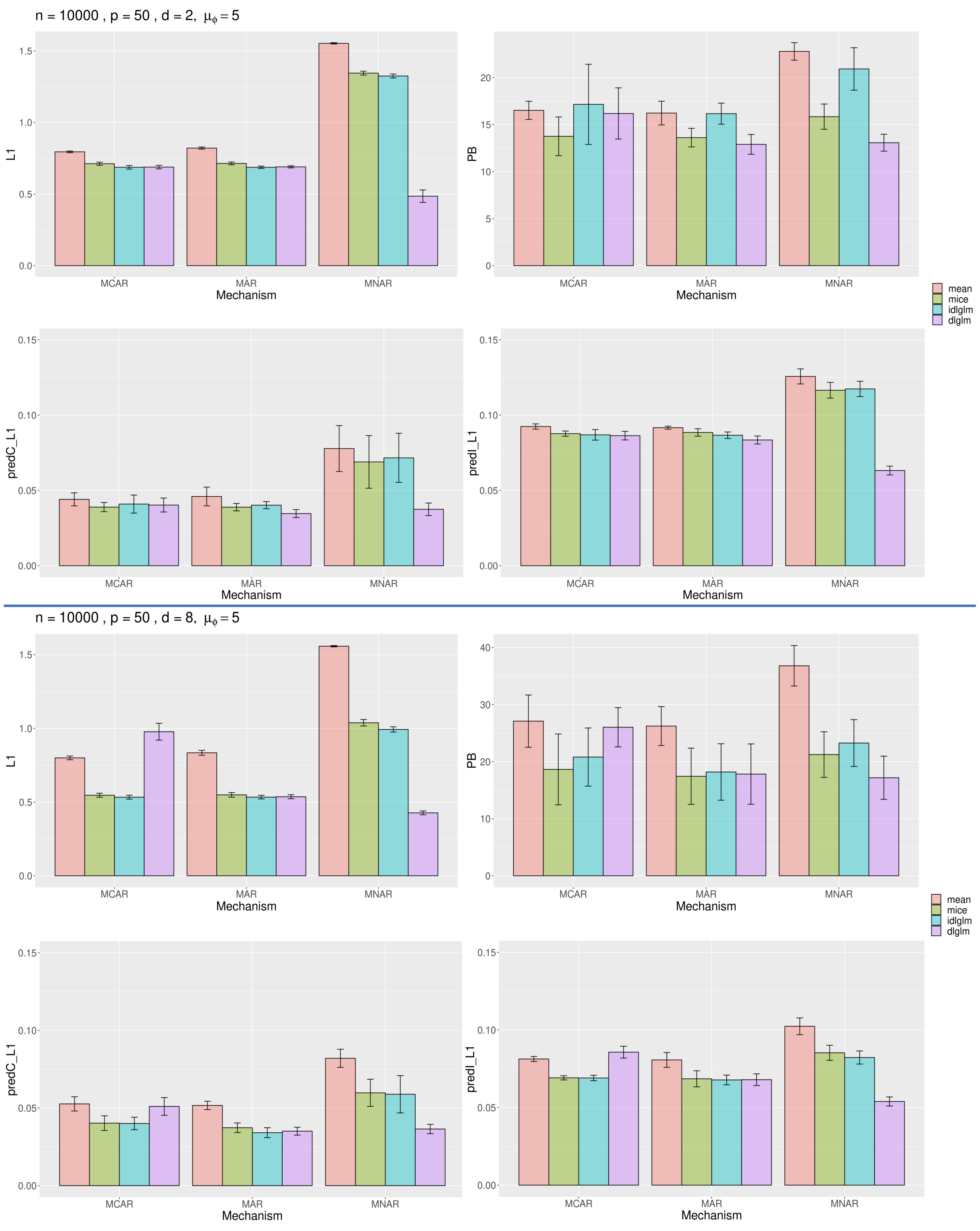


Figure 18: Simulation results with $n = 10,000$ and $p = 50$, varying $d = 2$ (top 4) and $d = 8$ (bottom 4). In each quadrant, we measure imputation accuracy by the average L1 distance between imputed vs true values in $\mathbf{X}$ (top-left), coefficient estimation accuracy by the average percent bias (PB) of the estimates $\hat{\boldsymbol{\beta}}$ compared to the truth (top-right), and prediction accuracy by the average L1 distance between the predicted and true probabilities of class 1 membership of $\mathbf{Y}$ using the true unmasked test set (predC, bottom-left) and the incomplete test set (predI, bottom-right).

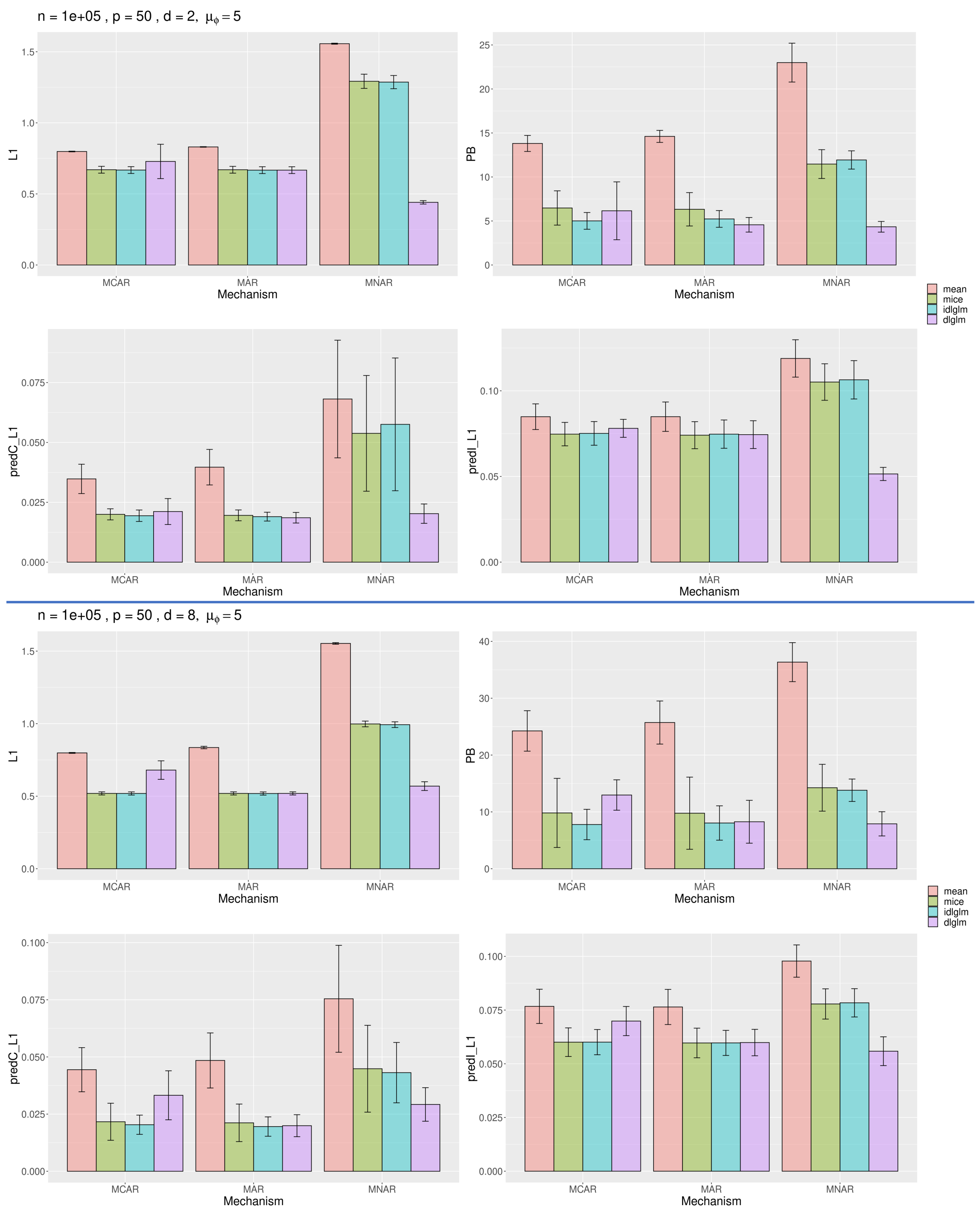
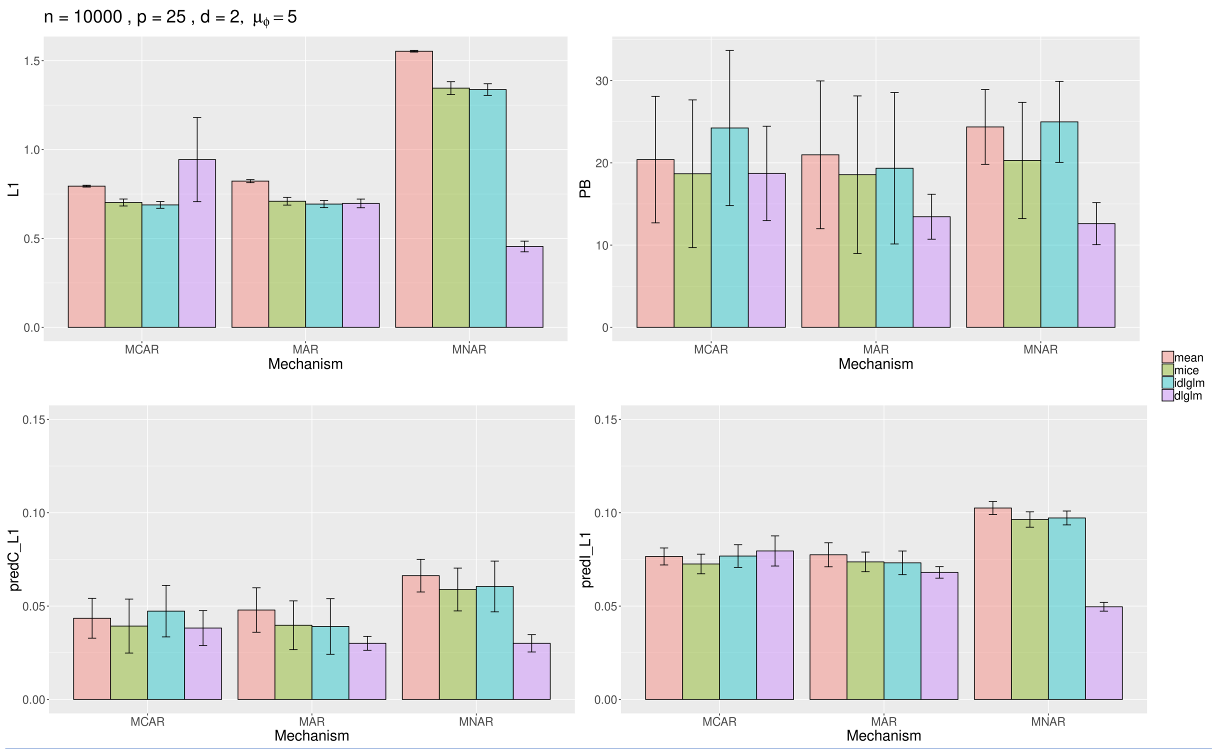


Figure 19: Simulation results with $n = 100,000$ and $p = 50$, varying $d = 2$ (top 4) and $d = 8$ (bottom 4). In each quadrant, we measure imputation accuracy by the average L1 distance between imputed vs true values in $\mathbf{X}$ (top-left), coefficient estimation accuracy by the average percent bias (PB) of the estimates $\hat{\boldsymbol{\beta}}$ compared to the truth (top-right), and prediction accuracy by the average L1 distance between the predicted and true probabilities of class 1 membership of $\mathbf{Y}$ using the true unmasked test set (predC, bottom-left) and the incomplete test set (predI, bottom-right).



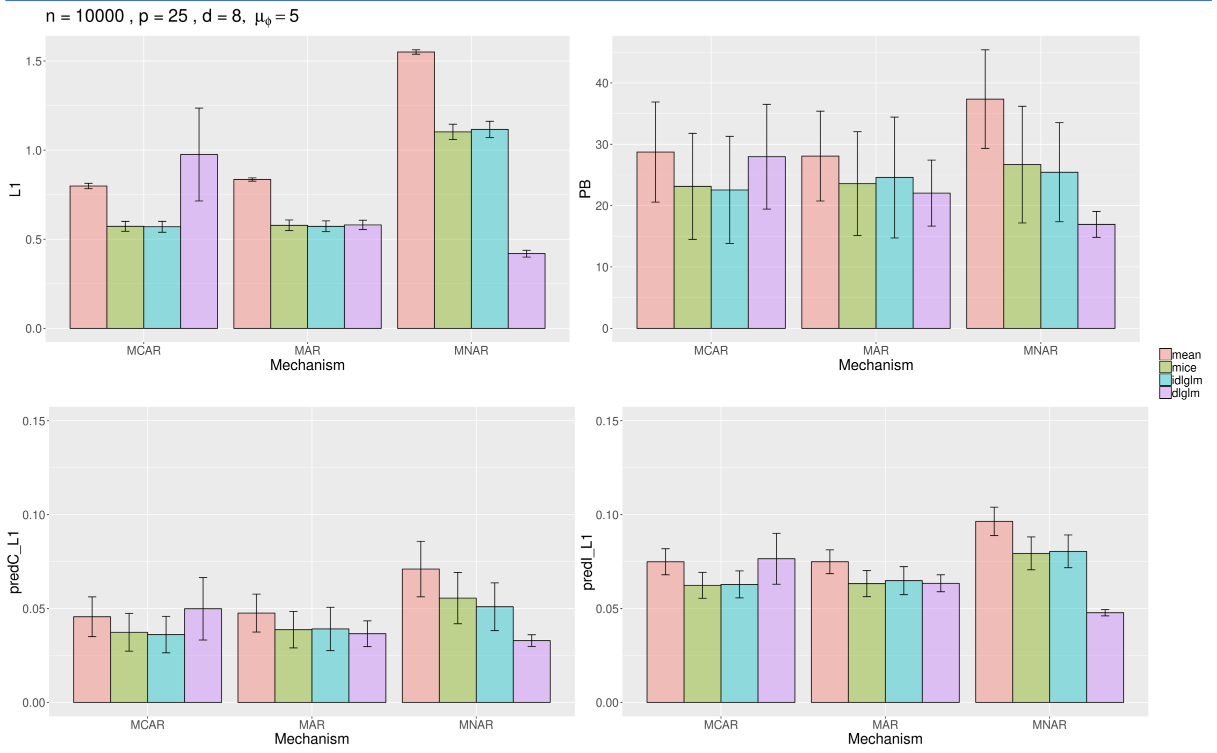


Figure 20: Simulation results with $n = 10,000$ and $p = 25$, varying $d = 2$ (top 4) and $d = 8$ (bottom 4). In each quadrant, we measure imputation accuracy by the average L1 distance between imputed vs true values in $\mathbf{X}$ (top-left), coefficient estimation accuracy by the average percent bias (PB) of the estimates $\hat{\boldsymbol{\beta}}$ compared to the truth (top-right), and prediction accuracy by the average L1 distance between the predicted and true probabilities of class 1 membership of $\mathbf{Y}$ using the true unmasked test set (predC, bottom-left) and the incomplete test set (predI, bottom-right).

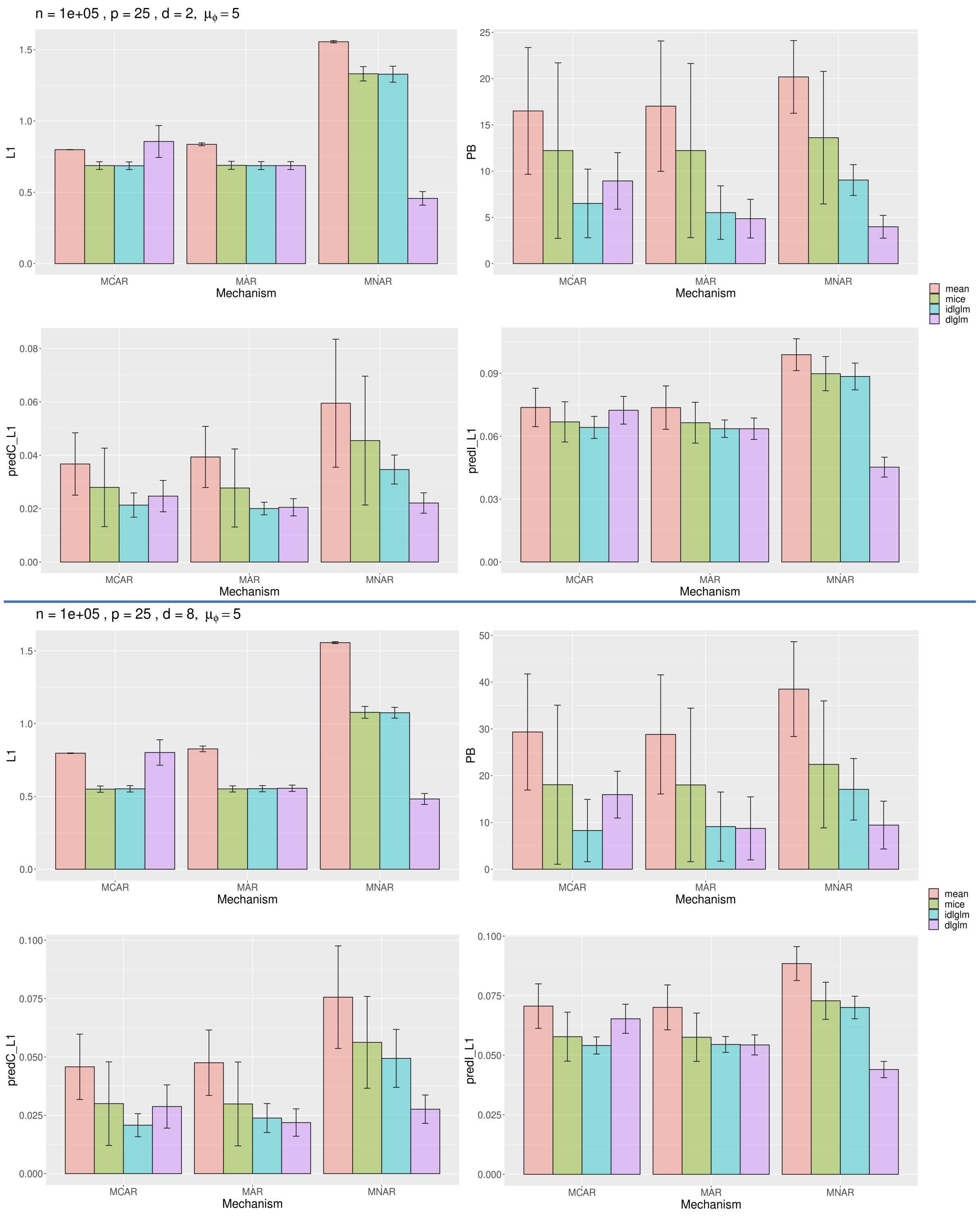


Figure 21: Simulation results with $n = 100,000$ and $p = 25$, varying $d = 2$ (top 4) and $d = 8$ (bottom 4). In each quadrant, we measure imputation accuracy by the average L1 distance between imputed vs true values in $\mathbf{X}$ (top-left), coefficient estimation accuracy by the average percent bias (PB) of the estimates $\hat{\boldsymbol{\beta}}$ compared to the truth (top-right), and prediction accuracy by the average L1 distance between the predicted and true probabilities of class 1 membership of $\mathbf{Y}$ using the true unmasked test set (predC, bottom-left) and the incomplete test set (predI, bottom-right).

4.2.2 Real Data with Simulated Missingness

Next, we analyzed 3 completely-observed, large datasets from the UCI Machine Learning Repository (Dua and Graff, 2017) that contained a specific response variable of interest, in order to preserve non-linearity and interactions between observed features. The DRYBEAN dataset contains 16 features describing 13,611 images of dry beans taken with a high-resolution camera, and the response variable of interest was the type of dry bean each image represents, with 7 different possible types of beans. The LETTER dataset contains 16 attributes of 20,000 black-and-white pixel images, each displaying one English letter (A to Z). Finally, the SHUTTLE dataset contains 9 numerical attributes pertaining to 58,000 shuttle stat logs (observations), which are classified into 7 different categories. Due to a low sample size in 4 of the 7 categories, we pre-filtered the observations pertaining to these categories out of the dataset, and the resulting dataset contained 57,756 observations of 3 categories. In each of these datasets, the response variable was multi-category.

We then simulated the missingness mask $\mathbf{R}^X$ with MCAR, MAR, and MNAR patterns of missingness in the manner described in Section 4.2.1.1. We split the samples in each dataset by a similar 8:1:1 ratio of training/validation/test samples. In the test set samples, we then imputed the missing values and predicted the response variables with each method in a manner similar to Section 4.2.1.1. For *dlglm* and *idlglm*, We account for potential nonlinear relationships between the covariates and response by allowing the number of hidden layers in $s_{\boldsymbol{\beta},\pi}(\cdot)$ to be nonzero in hyperparameter tuning. Then, we compared imputation and prediction accuracy on each dataset, under each mechanism of missingness. Since the underlying true probabilities of class membership were unavailable, we measured prediction accuracy by the Cohen's kappa metric on the complete (kappaC) and incomplete (kappaI) test set. This metric measures how accurately a binary class variable was predicted, with a value of -1 indicating worst possible performance, and a value of 1 indicating perfect concordance with the truth.

Results from the imputation and prediction analyses on these datasets can be found in Figure 22. We found that, as in the simulations, mean imputation performed most poorly in both imputation and downstream prediction, while *dlglm* tended to perform best in the MNAR cases, and performed comparably to *mice* and *idlglm* under the MCAR and MAR cases. This further validates our claims under a more realistic setting, where the true data generation mechanism may be unknown. We also see that under both MCAR and MAR missingness, *mice* performed worse than *idlglm* in prediction on the LETTER and SHUTTLE datasets. This is particularly interesting since *mice* is a widely-used algorithm that has been shown to perform well under

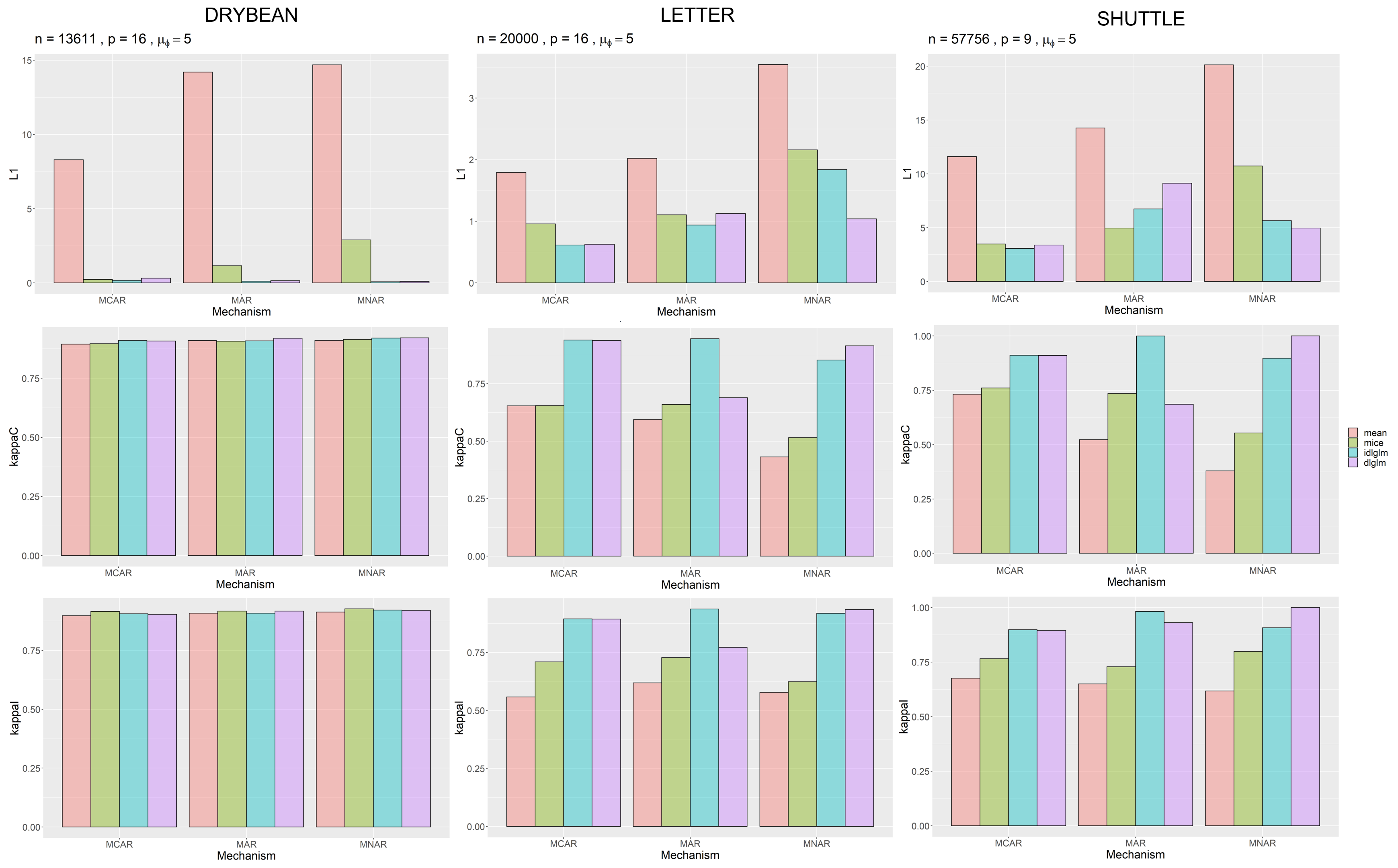


Figure 22: Imputation (top row) and prediction results from predC (middle row) and predI (bottom row) from comparative methods run on 3 large datasets from the UCI Machine Learning Repository: DRYBEAN, LETTER, and SHUTTLE (columns, left to right). Imputation error was measured by the average L1 distance between true and imputed entries, with lower values indicating better performance, and prediction performance was measured by the Cohen's kappa metric for both predC (kappaC) and predI (kappaI), with higher values indicating better performance.

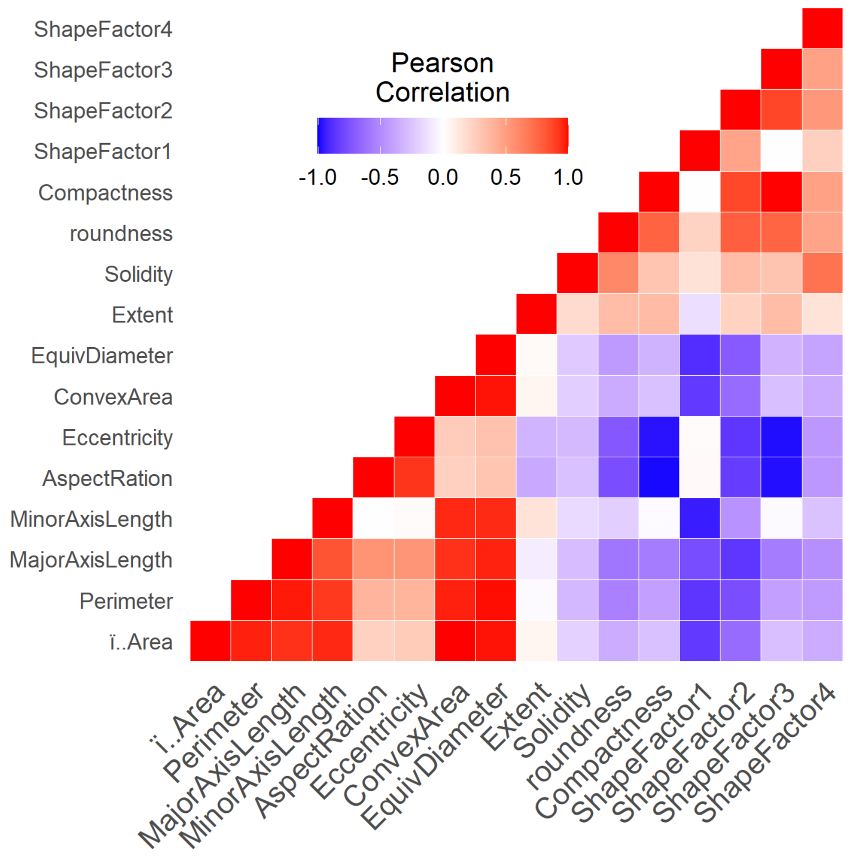


Figure 23: Correlation heatmap matrix of features in the DRYBEAN dataset. Many features in this dataset were highly correlated.

ignorable missingness (Van Buuren and Groothuis-Oudshoorn, 2011). However, the *mice* model has been known to break down under nonlinear relationships between the features (Van Buuren, 2018), as may be the case in many real-world datasets like these. Using neural networks to model the data generation process allows *idlglm* to better model potential nonlinear relationships between features, allowing for more accurate prediction.

Interestingly, all of the algorithms performed similarly in prediction on the DRYBEAN dataset. However, we found that this dataset contained extremely high levels of correlation between the variables, as shown in Figure 23. Highly-correlated features may cause some missing features to be redundant in the prediction model, thus reducing the importance of an accurate method to properly account for the missingness. Intuitively, when features containing missingness are highly correlated to other fully-observed features, such missingness may not truly reflect the MNAR scenario (Hapfelmeier et al., 2012). This is because there exist fully-observed features that are highly correlated with the missing features, and ignorably-missing data methods like *idlglm* may gather information about the missing entries from these correlated, fully-observed features without having to explicitly model the mechanism of missingness. Still, *idlglm* and *dlglm* imputed missing entries much more accurately than *mean* and *mice* under MAR and MNAR. Interestingly, we also see that *idlglm* performed similarly to *dlglm* under MNAR in this dataset.

We additionally performed similar analyses on 5 other smaller UCI datasets (BANKNOTE, CAREVALUATION, SPAM, RED, WHITE), and these results can be found in Figures 24 and 25. In general, we see that *dlglm* still performs best in imputation and prediction under MNAR missingness, although *idlglm* performs slightly better in the BANKNOTE dataset. In lower sample size settings, *dlglm* may not perform optimally, due to the more complex architecture and the additional task of learning a missingness model. We also note that for these missingness models in *dlglm*, as in the main text, we included all of the features in $\mathbf{X}$ as well as the response variable $\mathbf{Y}$ as covariates, although only one feature in $\mathbf{X}$ is involved in the simulation of missingness in each incomplete feature. Selecting important features from an overparameterized missingness model may be very difficult, and thus we recommend *dlglm* to be used with a sample size of at least $n = 10000$.

4.2.3 Bank Marketing Dataset

Finally, we performed prediction on the Bank Marketing dataset from the UCI Machine Learning Repository. This dataset contained 41,188 observations of 20 different attributes that were obtained based on direct phone calls from a Portuguese banking institution as part of a promotion campaign for a term deposit subscription (Moro et al., 2014). The response variable of interest was a fully-observed binary measure of whether the client subscribed a term deposit. Of the 20 attributes, we removed 1 attribute as directed from the manual due to perfect correlation with the response variable, and removed 3 other attributes that were deemed irrelevant to the prediction task: month of contact, day of contact, and communication type (cell phone or telephone).

Missingness was present in 8 of the 16 attributes: type of job, marital status, level of education, whether the client had a credit in default, whether the client had a housing loan, whether the client had a personal loan, number of days since the client was contacted in a previous campaign, and outcome of the previous campaign. The remaining 8 attributes were fully-observed: age of client, number of contacts during this campaign, number of contacts before this campaign, employment variation rate, consumer price index, consumer confidence index, euribor 3 month rate, and employee number. The global rate of missingness

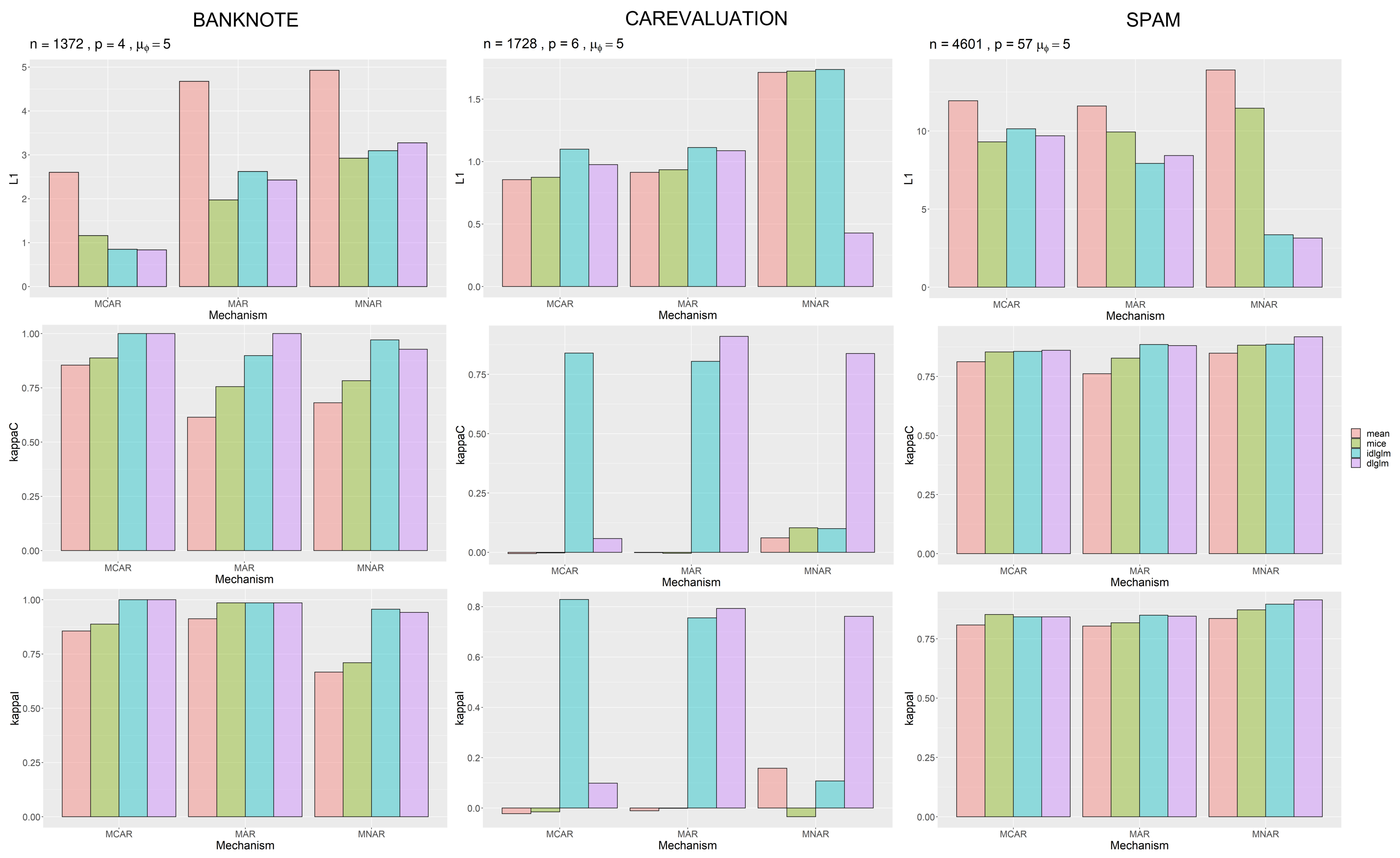


Figure 24: Imputation (top row) and prediction results from predC (middle row) and predI (bottom row) from comparative methods run on 3 small datasets from the UCI Machine Learning Repository: BANKNOTE, CAREVALUATION, and SPAM (columns, left to right). Imputation error was measured by the average L1 distance between true and imputed entries, with lower values indicating better performance, and prediction performance was measured by the Cohen's kappa metric for both predC (kappaC) and predI (kappaI), with higher values indicating better performance.

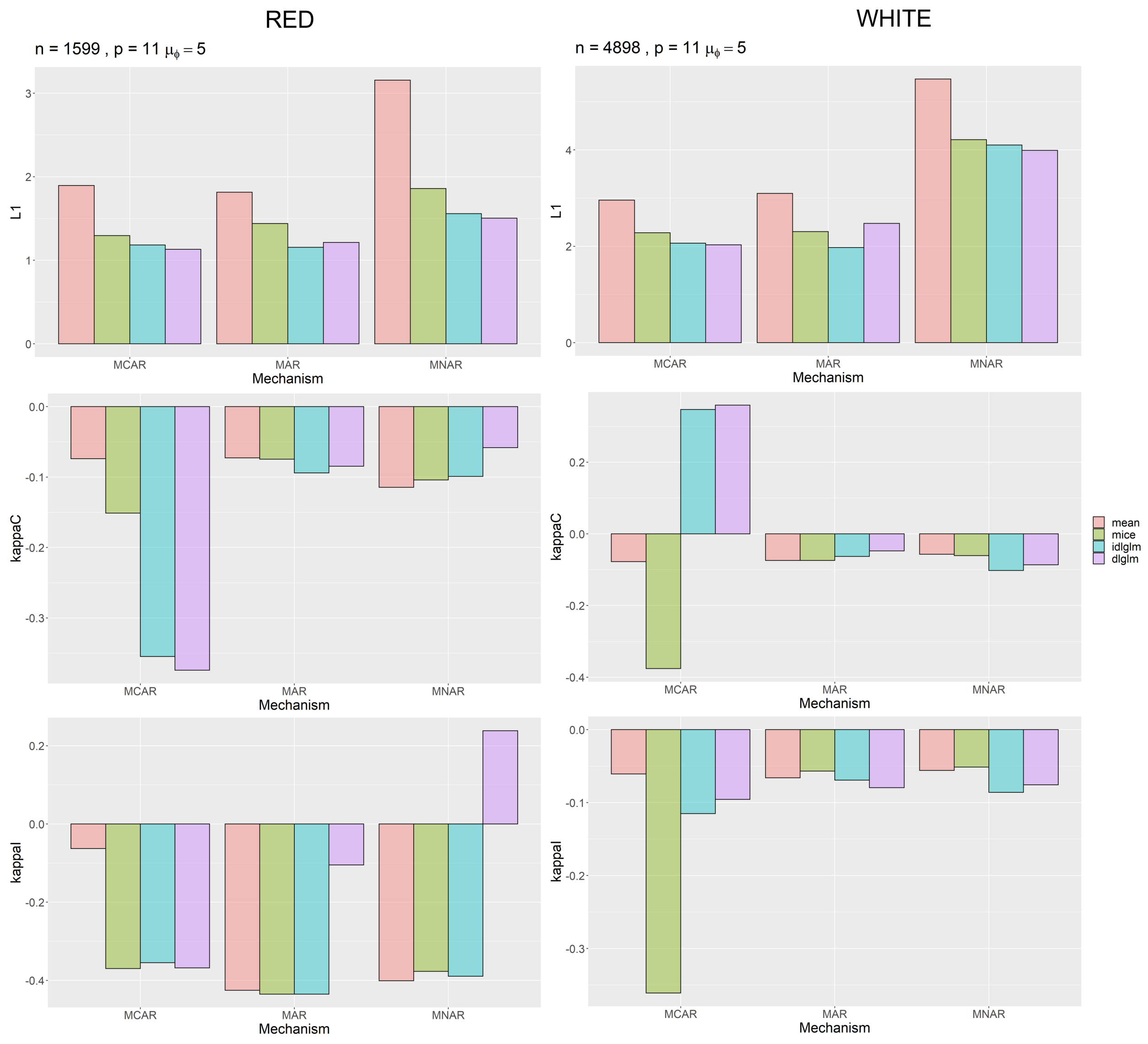


Figure 25: Imputation (top row) and prediction results from predC (middle row) and predI (bottom row) from comparative methods run on 2 small datasets from the UCI Machine Learning Repository: RED and WHITE (columns, left and right). Imputation error was measured by the average L1 distance between true and imputed entries, with lower values indicating better performance, and prediction performance was measured by the Cohen's kappa metric for both predC (kappaC) and predI (kappaI), with higher values indicating better performance.

was about 13.3%. The response variable of interest was collected by additional follow-up calls to confirm whether the client subscribed to the product.

This type of dataset reflects the most realistic situation in practice, where missingness exists in a dataset and one has no prior knowledge of either the relationships between the features and the response, or the underlying mechanism of the missingness. We divided the dataset into the 8:1:1 training, validation, and test set ratio, and performed prediction as before. Because neither the data nor the missingness was simulated, we compared just the predI prediction performance across the methods.

	AUC	PPV	kappa	F1
dlglm	0.88	0.481	0.445	0.516
idlglm	0.779	0.446	0.411	0.488
mean	0.769	0.448	0.385	0.46
mice	0.771	0.455	0.396	0.471

Table 6: Results from prediction analyses on the Bank Marketing dataset from the UCI Machine Learning Repository. We measured concordance between the true and predicted binary response by 4 metrics: Area Under ROC Curve (AUC), Positive Predictivity (PPV), Cohen's kappa (kappa), and F1 score.

Table 6 shows the results from these prediction analyses. We measured prediction performance of the binary response variable by 4 metrics: Area Under the ROC Curve (AUC), Positive Predictivity (PPV), Cohen's kappa (kappa), and the F1 metric. For each metric, a larger value represents greater concordance between the true and predicted response. We see that *dlglm* yielded a significantly greater performance in prediction via all metrics, which suggests that the missingness present in this dataset may be MNAR, as *dlglm* is the only method that accounts for MNAR missingness. This is in line with previous studies on missingness in survey data, which researchers have oftentimes considered MNAR (de Leeuw, 2001)

Also, we note that predictions via *idlglm* yielded slightly greater AUC, F1, and Cohen's kappa than *mice* imputation, but slightly lower PPV. This is reflective of the properties seen in the simulations, where *idlglm* performed comparably to *mice* in each simulated mechanism of missingness. Since these are both methods catered to handle ignorably-missing data, they would likely yield biased models under MNAR missingness, as may be the case in this dataset. Additionally, we see that *mice* and mean imputation yielded similar performance via all metrics. This may be due to the fact that these methods are impute-then-regress methods, unlike *dlglm* and *idlglm*, and thus downstream prediction performance may be less affected by the method of imputation employed (Le Morvan et al., 2021).

4.3 Discussion

In this chapter, we introduced a novel deep learning method called Deeply-learned Generalized Linear Model with Missing Data (*dlglm*), which is able to perform coefficient estimation and prediction in the presence of missing not at random (MNAR) data. *dlglm* utilizes a deep learning neural network architecture to model the generation of the data matrix $\mathbf{X}$, as well as the relationships between the response variable $\mathbf{Y}$ and $\mathbf{X}$ and between the missingness mask $\mathbf{R}$ and $\mathbf{X}$. In this way, we are able to (1) generalize the traditional GLM to account for complex nonlinear interactions between the features, and (2) account for ignorable and non-ignorable forms of missingness in the data. We also showed through simulations and real data analyses that *dlglm* performs better in coefficient estimation and prediction in the presence of MNAR missingness than other impute-then-regress methods, like mean and mice imputation. Furthermore, we found that *dlglm* was generally robust to the mechanism of missingness, performing comparably well to *mice* and *idlglm* under MCAR and MAR settings. Still, it is recommended to utilize *idlglm* when the missingness is ignorable, as the missingness model that is learned in *dlglm* is not necessary in the ignorable missingness setting.

A supervised learning algorithms such as *dlglm* and *idlglm* can be particularly useful in analyzing real-life data in the presence of missingness. In reality, the mechanism underlying missing values cannot be known or tested, however a flexible algorithm like *dlglm* that can be relatively robust to severe overparameterization of the missingness model can be pivotal to unbiased results. Furthermore, whereas impute-then-regress methods may typically require fully-observed observations at test time for prediction, *dlglm* and *idlglm* can predict the response of interest using partially-observed observations. This provides a convenient workflow, where a user need not separately re-impute the prediction set at test time.

In this chapter, we focused specifically on the case of univariate response $\mathbf{Y}$. *dlglm* and *idlglm* can be generalized to the multivariate $\mathbf{Y}$ case by (1) including $\mathbf{Y}$ in the existing IWAE structure and (2) expanding the neural network $s_{\boldsymbol{\beta},\pi}(\mathbf{x}_i)$ to account for all q responses in $\mathbf{Y}$, and utilizing samples of $\mathbf{Z}$ as additional input into this network such that $s_{\boldsymbol{\beta},\pi}(\mathbf{x}_i) \rightarrow s_{\boldsymbol{\beta},\pi}(\mathbf{x}_i, \mathbf{z}_i)$. By doing this, we account for multivariate $\mathbf{Y}$, outputting additional parameters pertaining to the newly-specified distribution of $p_{\boldsymbol{\beta},\pi}(\mathbf{y}_i|\mathbf{x}_i, \mathbf{z}_i)$ and modelling correlation of $\mathbf{Y}$ by the learned latent structure. We leave this as an extension of our method.

BIBLIOGRAPHY

Al-Khasawn, M. F. (2010). Estimating the negative binomial dispersion parameter. *Asian Journal of Mathematics & Statistics*, 3(1):1–15.

Alberts, B., Bray, D., and Lewis, J. (2002). *Molecular Biology of the Cell*. Taylor & Francis.

Allison, D. B., Gadbury, G. L., Heo, M., Fernández, J. R., Lee, C.-K., Prolla, T. A., and Weindruch, R. (2002). A mixture model approach for the analysis of microarray gene expression data. *Computational Statistics & Data Analysis*, 39:1–20.

Allison, P. D. et al. (2010). *Missing data*, volume 200210. Sage Thousand Oaks, CA.

Almalaq, A. and Edwards, G. (2017). A review of deep learning methods applied on load forecasting. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 511–516.

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106.

Anders, S., Pyl, P. T., and Huber, W. (2014). HTSeq–a python framework to work with high-throughput sequencing data. *Bioinformatics*, 31(2):166–169.

Aran, D., Sirota, M., and Butte, A. J. (2015). Systematic pan-cancer analysis of tumour purity. *Nature Communications*, 6(1).

Ausawalaithong, W., Marukatat, S., Thirach, A., and Wilaiprasitporn, T. (2018). Automatic Lung Cancer Prediction from Chest X-ray Images Using Deep Learning Approach. *arXiv e-prints*, page arXiv:1808.10858.

Banfield, J. D. and Raftery, A. E. (1993). Model-based gaussian and non-gaussian clustering. *Biometrics*, 49(3):803.

Basak, S., Sengupta, S., and Dubey, A. (2018). Mechanisms for Integrated Feature Normalization and Remaining Useful Life Estimation Using LSTMs Applied to Hard-Disks. *arXiv e-prints*, page arXiv:1810.08985.

Bass, A., Thorsson, V., Shmulevich, I., and et al. (2014). Comprehensive molecular characterization of gastric adenocarcinoma. *Nature*, 513(7517):202–209.

Beane, J., Vick, J., Schembri, F., Anderlind, C., Gower, A., Campbell, J., Luo, L., Zhang, X. H., Xiao, J., Alekseyev, Y. O., Wang, S., Levy, S., Massion, P. P., Lenburg, M., and Spira, A. (2011). Characterizing the impact of smoking and lung cancer on the airway transcriptome using RNA-seq. *Cancer Prevention Research*, 4(6):803–817.

Beaulieu-Jones, B. K. and and, J. H. M. (2016). Missing Data Imputation in the Electronic Health Record Using Deeply Learned Autoencoders. In *Biocomputing 2017*. WORLD SCIENTIFIC.

Best, M. G., Sol, N., Kooi, I., Tannous, J., Westerman, B. A., Rustenburg, F., Schellen, P., Verschueren, H., Post, E., Koster, J., Ylstra, B., Ameziane, N., Dorsman, J., Smit, E. F., Verheul, H. M., Noske, D. P., Reijneveld, J. C., Nilsson, R. J. A., Tannous, B. A., Wesseling, P., and Wurdinger, T. (2015). RNA-seq of tumor-educated platelets enables blood-based pan-cancer, multiclass, and molecular pathway cancer diagnostics. *Cancer Cell*, 28(5):666–676.

Biernacki, C., Celeux, G., and Govaert, G. (2003). Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computational Statistics & Data Analysis*, 41(3-4):561–575.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2016). Variational Inference: A Review for Statisticians. *arXiv e-prints*, page arXiv:1601.00670.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122.

Brannon, A. R., Reddy, A., Seiler, M., Arreola, A., Moore, D. T., Pruthi, R. S., and et al. (2010). Molecular stratification of clear cell renal cell carcinoma by consensus clustering reveals distinct subtypes and survival patterns. *Genes & Cancer*, 1(2):152–163.

Brat, D. J., Verhaak, R. G., Aldape, K. D., and et al. (2015). Comprehensive, integrative genomic analysis of diffuse lower-grade gliomas. *New England Journal of Medicine*, 372(26):2481–2498.

Brosch, T., , and Tam, R. (2013). Manifold learning of brain MRIs by deep learning. In *Advanced Information Systems Engineering*, pages 633–640. Springer Berlin Heidelberg.

Browne, R. P. and McNicholas, P. D. (2013). Estimating common principal components in high dimensions. *Advances in Data Analysis and Classification*, 8(2):217–226.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance Weighted Autoencoders. *arXiv e-prints*, page arXiv:1509.00519.

Cao, D. and Yang, B. (2010). An improved k-medoids clustering algorithm. In *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*. IEEE.

Carrio, A., Sampedro, C., Rodriguez-Ramos, A., and Campoy, P. (2017). A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, 2017:1–13.

Carter, S. L., Cibulskis, K., Helman, E., McKenna, A., Shen, H., Zack, T., Laird, P. W., Onofrio, R. C., Winckler, W., Weir, B. A., Beroukhim, R., Pellman, D., Levine, D. A., Lander, E. S., Meyerson, M., and Getz, G. (2012). Absolute quantification of somatic DNA alterations in human cancer. *Nature Biotechnology*, 30(5):413–421.

Casella, G. and Robert, C. (2010). *Monte Carlo Statistical Methods*. Springer New York.

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., and Oliveira, A. L. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55:194–211.

Celeux, G. and Govaert, G. (1990). Stochastic algorithms for clustering. In Momirović, K. and Mildner, V., editors, *Compstat*, pages 3–8, Heidelberg. Physica-Verlag HD.

Celeux, G. and Govaert, G. (1992). A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14(3):315–332.

Celeux, G. and Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781–793.

Chappell, J., Watters, K. E., Takahashi, M. K., and Lucks, J. B. (2015). A renaissance in RNA synthetic biology: new mechanisms, applications and tools for the future. *Current Opinion in Chemical Biology*, 28:47–56.

Charrad, M., Ghazzali, N., Boiteau, V., and Niknafs, A. (2014). NbClust: AnRPackage for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, 61(6).

Chen, D., Liu, S., Kingsbury, P., Sohn, S., Storlie, C. B., Habermann, E. B., and et al. (2019). Deep learning and alternative learning strategies for retrospective real-world clinical data. *npj Digital Medicine*, 2(1).

Chia, S. K., Bramwell, V. H., Tu, D., Shepherd, L. E., Jiang, S., and et al., T. V. (2012). A 50-gene intrinsic subtype classifier for prognosis and prediction of benefit from adjuvant tamoxifen. *Clinical Cancer Research*, 18(16):4465–4472.

Chu, W., Ghahramani, Z., Falciani, F., and Wild, D. L. (2005). Biomarker discovery in microarray gene expression data with gaussian processes. *Bioinformatics*, 21(16):3385–3393.

Chung, W., Eum, H. H., Lee, H.-O., Lee, K.-M., Lee, H.-B., Kim, K.-T., and et al. (2017). Single-cell RNA-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. *Nature Communications*, 8(1).

Cima, I., Kong, S. L., Sengupta, D., Tan, I. B., Phyo, W. M., Lee, D., and et al. (2016). Tumor-derived circulating endothelial cell clusters in colorectal cancer. *Science Translational Medicine*, 8(345):345ra89–345ra89.

Ciriello, G., Gatza, M. L., Beck, A. H., Wilkerson, M. D., Rhie, S. K., Pastore, A., and et al. (2015). Comprehensive molecular portraits of invasive lobular breast cancer. *Cell*, 163(2):506–519.

Colaprico, A., Silva, T. C., Olsen, C., Garofano, L., Cava, C., Garolini, D., and et al. (2015). TCGAbiolinks: an r/bioconductor package for integrative analysis of TCGA data. *Nucleic Acids Research*, 44(8):e71–e71.

Collisson, E., Campbell, J., Brooks, A., and et al. (2014). Comprehensive molecular profiling of lung adenocarcinoma. *Nature*, 511(7511):543–550.

Creighton, C., Morgan, M., Gunaratne, P., and et al. (2013). Comprehensive molecular characterization of clear cell renal cell carcinoma. *Nature*, 499(7456):43–49.

Cremer, C., Morris, Q., and Duvenaud, D. (2017). Reinterpreting Importance-Weighted Autoencoders. *arXiv e-prints*, page arXiv:1704.02916.

Cui, W., Qian, Y., Zhou, X., Lin, Y., Jiang, J., Chen, J., and et al. (2015). Discovery and characterization of long intergenic non-coding RNAs (lincRNA) module biomarkers in prostate cancer: an integrative analysis of RNA-seq data. *BMC Genomics*, 16(Suppl 7):S3.

de Leeuw, E. D. (2001). Reducing missing data in surveys: An overview of methods. *Quality and Quantity*, 35(2):147–160.

Dellaert, F. (2002). The expectation maximization algorithm. Technical report, Georgia Institute of Technology.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via theEMAlgorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Diggle, P. and Kenward, M. G. (1994). Informative drop-out in longitudinal data analysis. *Applied Statistics*, 43(1):49.

Doersch, C. (2016). Tutorial on Variational Autoencoders. *arXiv e-prints*, page arXiv:1606.05908.

Dormehl, L. (2019). What is an artificial neural network? here's everything you need to know.

Du, H., Enders, C., Keller, B. T., Bradbury, T. N., and Karney, B. R. (2021). A bayesian latent variable selection model for nonignorable missingness. *Multivariate Behavioral Research*, pages 1–49.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Esteban, A. (2002). Characteristics and outcomes in adult patients receiving mechanical ventilation. a 28-day international study. *JAMA*, 287(3):345.

Fakoor, R., Ladhak, F., Nazi, A., and Huber, M. (2013). Using deep learning to enhance cancer diagnosis and classification. In *Proceedings of the international conference on machine learning*, volume 28, pages 3937–3949. ACM, New York, USA.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360.

Finotello, F. and Camillo, B. D. (2014). Measuring differential gene expression with RNA-seq: challenges and strategies for data analysis. *Briefings in Functional Genomics*, 14(2):130–142.

Fishbein, L., Leshchiner, I., Walter, V., Danilova, L., Robertson, A. G., Johnson, A. R., and et al. (2017). Comprehensive molecular characterization of pheochromocytoma and paraganglioma. *Cancer Cell*, 31(2):181–193.

Fraley, C. and Raftery, A. E. (1998). How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1).

Fu, X., Fu, N., Guo, S., Yan, Z., Xu, Y., Hu, H., and et al. (2009). Estimating accuracy of RNA-seq and microarrays with proteomics. *BMC Genomics*, 10(1):161.

Garcia, R. I., Ibrahim, J. G., and Zhu, H. (2009). Variable selection in the cox regression model with covariates missing at random. *Biometrics*, 66(1):97–104.

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

Gershman, S. J. and Goodman, N. D. (2014). Amortized inference in probabilistic reasoning. In *CogSci*.

Ghorbani, A. and Zou, J. Y. (2018). Embedding for informative missingness: Deep learning with incomplete data. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 437–445. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Grossman, R. L., Heath, A. P., Ferretti, V., Varmus, H. E., Lowy, D. R., Kibbe, W. A., and Staudt, L. M. (2016). Toward a shared vision for cancer genomic data. *New England Journal of Medicine*, 375(12):1109–1112.

Guo, H. and Gelfand, S. B. (1990). Analysis of gradient descent learning algorithms for multilayer feedforward neural networks. In *29th IEEE Conference on Decision and Control*, pages 1751–1756. IEEE.

Haitovsky, Y. (1968). Missing data in regression analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, 30(1):67–82.

Hamer, R. M. and Simpson, P. M. (2009). Last observation carried forward versus mixed models in the analysis of psychiatric clinical trials.

Hammerman, P., Lawrence, M., Voet, D., and et al. (2012). Comprehensive genomic characterization of squamous cell lung cancers. *Nature*, 489(7417):519–525.

Han, J. and Pei, M. K. J. (2017). *Data Mining: Concepts and Techniques*. Elsevier LTD, Oxford.

Hapfelmeier, A., Hothorn, T., Ulm, K., and Strobl, C. (2012). A new variable importance measure for random forests with missing data. *Statistics and Computing*, 24(1):21–34.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385.

Hicks, S. C., Townes, F. W., Teng, M., and Irizarry, R. A. (2017). Missing data and technical variability in single-cell RNA-sequencing experiments. *Biostatistics*, 19(4):562–578.

Hilbe, J. M. (2009). *Modeling Count Data*. Cambridge University Press.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Holland, P. W. and Welsch, R. E. (1977). Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827.

Hoogland, J., Barreveld, M., Debray, T. P. A., Reitsma, J. B., Verstraelen, T. E., Dijkgraaf, M. G. W., and Zwinderman, A. H. (2020). Handling missing predictor values when validating and applying a prediction model to new patients. *Statistics in Medicine*, 39(25):3591–3607.

Horton, N. J. and Kleinman, K. P. (2007). Much ado about nothing. *The American Statistician*, 61(1):79–90.

Huber-Keener, K. J., Liu, X., Wang, Z., Wang, Y., Freeman, W., Wu, S., and et al. (2012). Differential gene expression in tamoxifen-resistant breast cancer cells revealed by a new analytical model of RNA-seq data. *PLoS ONE*, 7(7):e41333.

Huszno, J. and Kolosza, Z. (2019). Molecular characteristics of breast cancer according to clinicopathological factors. *Molecular and Clinical Oncology*.

Ibrahim, J. G. (2001). Missing responses in generalised linear mixed models when the missing data mechanism is nonignorable. *Biometrika*, 88(2):551–564.

Ibrahim, J. G., Chen, M.-H., Lipsitz, S. R., and Herring, A. H. (2005). Missing-data methods for generalized linear models. *Journal of the American Statistical Association*, 100(469):332–346.

Ibrahim, J. G., Lipsitz, S. R., and Chen, M.-H. (1999). Missing covariates in generalized linear models when the missing data mechanism is non-ignorable. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(1):173–190.

Ibrahim, J. G. and Molenberghs, G. (2009). Missing data methods in longitudinal studies: a review. *TEST*, 18(1):1–43.

Inouye, D. I., Yang, E., Allen, G. I., and Ravikumar, P. (2017). A review of multivariate distributions for count data derived from the poisson distribution. *Wiley Interdisciplinary Reviews: Computational Statistics*, 9(3):e1398.

Ipsen, N. B., Mattei, P.-A., and Frellsen, J. (2021). not-miwae: Deep generative modelling with missing not at random data.

Ivanov, O., Figurnov, M., and Vetrov, D. (2019). Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*.

Jardim-Perassi, B. V., Alexandre, P. A., Sonehara, N. M., de Paula-Junior, R., Júnior, O. R., Fukumasu, H., and et al. (2019). RNA-seq transcriptome analysis shows anti-tumor actions of melatonin in a breast cancer xenograft model. *Scientific Reports*, 9(1).

Jaskowiak, P. A., Campello, R. J., and Costa, I. G. (2014). On the selection of appropriate distances for gene expression data clustering. *BMC Bioinformatics*, 15(S2).

Jaskowiak, P. A., Costa, I. G., and Campello, R. J. (2018). Clustering of RNA-seq samples: Comparison study on cancer data. *Methods*, 132:42–49.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

Johnson, W. E., Li, C., and Rabinovic, A. (2006). Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127.

Kang, H. (2013). The prevention and handling of the missing data. *Korean Journal of Anesthesiology*, 64(5):402.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114.

Kingma, D. P. and Welling, M. (2019). An Introduction to Variational Autoencoders. *arXiv e-prints*, page arXiv:1906.02691.

Klein, R. W. and Dubes, R. C. (1989). Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22(2):213–220.

Koboldt, D. C., Fulton, R. S., McLellan, M. D., Schmidt, H., KalickiVeizer, J., and et al., J. F. M. (2012). Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61–70.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Larsson, E., Lindström, A.-C., Eriksson, M., and Oldner, A. (2019). Impact of gender on post- traumatic intensive care and outcomes. *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine*, 27(1).

Lawrence, M., Sougnez, C., Lichtenstein, L., and et al. (2015). Comprehensive genomic characterization of head and neck squamous cell carcinomas. *Nature*, 517(7536):576–582.

Le Morvan, M., Josse, J., Scornet, E., and Varoquaux, G. (2021). What'sa good imputation to predict with missing values? *Advances in Neural Information Processing Systems*, 34.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, M.-L. T., Kuo, F. C., Whitmore, G. A., and Sklar, J. (2000). Importance of replication in microarray gene expression studies: Statistical methods and evidence from repetitive cDNA hybridizations. *Proceedings of the National Academy of Sciences*, 97(18):9834–9839.

Leek, J. T. (2014). svaseq: removing batch effects and other unwanted noise from sequencing data. *Nucleic Acids Research*, 42(21):e161–e161.

Leek, J. T., Scharpf, R. B., Bravo, H. C., Simcha, D., Langmead, B., Johnson, W. E., and et al. (2010). Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews Genetics*, 11(10):733–739.

Levine, D. A., Getz, G., Gabriel, S., and et al. (2013). Integrated genomic characterization of endometrial carcinoma. *Nature*, 497(7447):67–73.

Li, B., Ruotti, V., Stewart, R. M., Thomson, J. A., and Dewey, C. N. (2009a). RNA-seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493–500.

Li, J., Bushel, P. R., Chu, T.-M., and Wolfinger, R. D. (2009b). Principal variance components analysis: Estimating batch effects in microarray gene expression data. In *Batch Effects and Noise in Microarray Experiments*, pages 141–154. John Wiley & Sons, Ltd.

Li, P., Piao, Y., Shon, H. S., and Ryu, K. H. (2015). Comparing the normalization methods for the differential analysis of illumina high-throughput RNA-seq data. *BMC Bioinformatics*, 16(1).

Li, Q., Noel-MacDonnell, J. R., Koestler, D. C., Goode, E. L., and Fridley, B. L. (2018). Subject level clustering using a negative binomial model for small transcriptomic studies. *BMC Bioinformatics*, 19(1).

Li, Y., Chien, J., Smith, D. I., and Ma, J. (2011). FusionHunter: identifying fusion transcripts in cancer using paired-end RNA-seq. *Bioinformatics*, 27(12):1708–1710.

Liang, J., Lv, J., and Liu, Z. (2015). Identification of stage-specific biomarkers in lung adenocarcinoma based on RNA-seq data. *Tumor Biology*, 36(8):6391–6399.

Lim, K.-L., Jiang, X., and Yi, C. (2020). Deep clustering with variational autoencoder. *IEEE Signal Processing Letters*, 27:231–235.

Linehan, W. M., Spellman, P. T., Ricketts, C. J., and et al. (2016). Comprehensive molecular characterization of papillary renal-cell carcinoma. *New England Journal of Medicine*, 374(2):135–145.

Lipsitz, S. R. and Ibrahim, J. G. (1996). A conditional model for incomplete covariates in parametric regression models. *Biometrika*, 83(4):916–922.

Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc.

Liu, J. S. (2008). *Monte Carlo Strategies in Scientific Computing*. Springer.

Liu, Y., Noon, A. P., Cabeza, E. A., Shen, J., Kuk, C., Ilczynski, C., and et al. (2014). Next-generation RNA sequencing of archival formalin-fixed paraffin-embedded urothelial bladder cancer. *European Urology*, 66(6):982–986.

Ljunggren, M., Castrén, M., Nordberg, M., and Kurland, L. (2016). The association between vital signs and mortality in a retrospective cohort study of an unselected emergency department population. *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine*, 24(1).

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058.

Lotfollahi, M., Wolf, F. A., and Theis, F. J. (2019). scGen predicts single-cell perturbation responses. *Nature Methods*, 16(8):715–721.

Love, M. I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12).

Luo, Y., Cai, X., ZHANG, Y., Xu, J., and xiaojie, Y. (2018). Multivariate time series imputation with generative adversarial networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 1596–1607. Curran Associates, Inc.

Lydia, A. and Francis, S. (2019). Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, 6(5):566–568.

Mahmood, K., Eldeirawi, K., and Wahidi, M. M. (2012). Association of gender with outcomes in critically ill patients. *Critical Care*, 16(3):R92.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge university press.

Mao, J.-H., van Diest, P. J., Perez-Losada, J., and Snijders, A. M. (2017). Revisiting the impact of age and molecular subtype on overall survival after radiotherapy in breast cancer patients. *Scientific Reports*, 7(1).

Mardis, E. R. and Wilson, R. K. (2009). Cancer genome sequencing: a review. *Human Molecular Genetics*, 18(R2):R163–R168.

Mattei, P.-A. and Frellsen, J. (2019). MIWAE: Deep generative modelling and imputation of incomplete data sets. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4413–4423, Long Beach, California, USA. PMLR.

McCarthy, D. J., Chen, Y., and Smyth, G. K. (2012). Differential expression analysis of multifactor RNA-seq experiments with respect to biological variation. *Nucleic Acids Research*, 40(10):4288–4297.

McCoy, J. T., Kroon, S., and Auret, L. (2018). Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine*, 51(21):141–146.

McCullagh, P. and Nelder, J. A. (2019). *Generalized linear models*. Routledge.

McCulloch, C. E. (1997). Maximum likelihood algorithms for generalized linear mixed models. *Journal of the American Statistical Association*, 92(437):162–170.

McGettigan, P. A. (2013). Transcriptomics in the RNA-seq era. *Current Opinion in Chemical Biology*, 17(1):4–11.

McLachlan, G. J. and Krishnan, T. (2008). *The EM Algorithm and Extensions, 2E*. John Wiley & Sons, Inc.

McLendon, R., Friedman, A., Bigner, D., and et al (2008). Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216):1061–1068.

McNicholas, P. D. and Murphy, T. B. (2010). Model-based clustering of microarray expression data via latent gaussian mixture models. *Bioinformatics*, 26(21):2705–2712.

McPherson, A., Hormozdiari, F., Zayed, A., Giuliany, R., Ha, G., Sun, M. G. F., and et al. (2011). deFuse: An algorithm for gene fusion discovery in tumor RNA-seq data. *PLoS Computational Biology*, 7(5):e1001138.

Meng, X.-L. (1994). On the rate of convergence of the ecm algorithm. *The Annals of Statistics*, 22(1):326–339.

Meng, X.-L. and Rubin, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267–278.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Mingoti, S. A. and Lima, J. O. (2006). Comparing SOM neural network with fuzzy c-means, k-means and traditional hierarchical clustering algorithms. *European Journal of Operational Research*, 174(3):1742–1759.

Miotto, R., Li, L., Kidd, B. A., and Dudley, J. T. (2016). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6(1).

Mo, Q. and Shen, R. (2019). *iClusterPlus: Integrative clustering of multi-type genomic data*. R package version 1.20.0.

Mo, Q., Wang, S., Seshan, V. E., Olshen, A. B., Schultz, N., and et al., C. S. (2013). Pattern discovery and cancer gene identification in integrated cancer genomic data. *Proceedings of the National Academy of Sciences*, 110(11):4245–4250.

Mootha, V. K., Lindgren, C. M., Eriksson, K.-F., Subramanian, A., Sihag, S., Lehar, J., Puigserver, P., Carlsson, E., Ridderstråle, M., Laurila, E., Houstis, N., Daly, M. J., Patterson, N., Mesirov, J. P., Golub, T. R., Tamayo, P., Spiegelman, B., Lander, E. S., Hirschhorn, J. N., Altshuler, D., and Groop, L. C. (2003). PGC-1-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nature Genetics*, 34(3):267–273.

Moro, S., Cortez, P., and Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv e-prints*, page arXiv:1109.2378.

Murakami, N., Okuno, Y., Yoshida, K., Shiraishi, Y., Nagae, G., Suzuki, K., and et al. (2018). Integrated molecular profiling of juvenile myelomonocytic leukemia. *Blood*, 131(14):1576–1586.

Murphy, J. (2016). An overview of convolutional neural network architectures for deep learning. *Microway Inc*, pages 1–22.

Muzny, D., Bainbridge, M., Chang, K., and et al. (2012). Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330–337.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1).

Nazabal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. (2018). Handling Incomplete Heterogeneous Data using VAEs. *arXiv e-prints*, page arXiv:1807.03653.

Nazari, Z., Kang, D., Asharif, M. R., Sung, Y., and Ogawa, S. (2015). A new hierarchical clustering algorithm. In *2015 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*. IEEE.

Neal, R. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers.

Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384.

Nie, L., Wang, M., Zhang, L., Yan, S., Zhang, B., and Chua, T.-S. (2015). Disease inference from health-related questions via sparse deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2107–2119.

Nielsen, T. O., Parker, J. S., Leung, S., Voduc, D., Ebbert, M., Vickery, T., Davies, S. R., Snider, J., Stijleman, I. J., Reed, J., et al. (2010). A comparison of pam50 intrinsic subtyping with immunohistochemistry and clinical prognostic factors in tamoxifen-treated estrogen receptor–positive breast cancerpam50 in er-positive breast cancer. *Clinical cancer research*, 16(21):5222–5232.

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer.

Noel-MacDonnell, J. R., Usset, J., Goode, E. L., and Fridley, B. L. (2018). Assessment of data transformations for model-based clustering of RNA-seq data. *PLOS ONE*, 13(2):e0191758.

O'Hara, R. B. and Kotze, D. J. (2010). Do not log-transform count data. *Methods in Ecology and Evolution*, 1(2):118–122.

O'Shea, R. (2019). Interpreting Missing Data Patterns in the ICU. *arXiv e-prints*, page arXiv:1912.08612.

Ouyang, M., Welsh, W. J., and Georgopoulos, P. (2004). Gaussian mixture clustering and imputation of microarray data. *Bioinformatics*, 20(6):917–923.

Ozdag, M. (2018). Adversarial attacks and defenses against deep neural networks: A survey. *Procedia Computer Science*, 140:152–161.

O'Connor, C. M., Adams, J. U., and Fairman, J. (2010). Essentials of cell biology. *Cambridge, MA: NPG Education*, 1:54.

Pan, W., Lin, J., and Le, C. T. (2003). A mixture model approach to detecting differentially expressed genes with microarray data. *Functional & integrative genomics*, 3(3):117–124.

Pan, W., Shen, X., and Liu, B. (2013). Cluster analysis: Unsupervised learning via supervised learning with a non-convex penalty. *Journal of Machine Learning Research*, 14.

Patel, S., Sihmar, S., and Jatain, A. (2015). A study of hierarchical clustering algorithms. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 537–541.

Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., and Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 14(4):417–419.

Peixoto, L., Risso, D., Poplawski, S. G., Wimmer, M. E., Speed, T. P., Wood, M. A., and Abel, T. (2015). How data analysis affects power, reproducibility and biological insight of RNA-seq studies in complex datasets. *Nucleic Acids Research*, 43(16):7664–7674.

Perou, C. M., Sørlie, T., Eisen, M. B., van de Rijn, M., Jeffrey, S. S., and et al., C. A. R. (2000). Molecular portraits of human breast tumours. *Nature*, 406(6797):747–752.

Picornell, A. C., Echavarria, I., Alvarez, E., López-Tarruella, S., Jerez, Y., Hoadley, K., Parker, J. S., del Monte-Millán, M., Ramos-Medina, R., Gayarre, J., Ocaña, I., Cebollero, M., Massarrah, T., Moreno, F., Saenz, J. A. G., Moreno, H. G., Ballesteros, A., Borrego, M. R., Perou, C. M., and Martin, M. (2019). Breast cancer PAM50 signature: correlation and concordance between RNA-seq and digital multiplexed gene expression technologies in a triple negative breast cancer series. *BMC Genomics*, 20(1).

Piegorsch, W. W. (1990). Maximum likelihood estimation for the negative binomial dispersion parameter. *Biometrics*, 46(3):863.

Podlesnyy, S. (2019). Towards Data-Driven Automatic Video Editing. *arXiv e-prints*, page arXiv:1907.07345.

Poggio, T., Banburski, A., and Liao, Q. (2020). Theoretical issues in deep networks. *Proceedings of the National Academy of Sciences*, 117(48):30039–30045.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Pumsirirat, A. and Yan, L. (2018). Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. *International Journal of Advanced Computer Science and Applications*, 9(1).

Qi, M. and Wu, Y. (2003). Nonlinear prediction of exchange rates with monetary fundamentals. *Journal of Empirical Finance*, 10(5):623–640.

Qu, Y. and Xu, S. (2004). Supervised cluster analysis for microarray data based on multivariate gaussian mixture. *Bioinformatics*, 20(12):1905–1913.

Raj-Kumar, P.-K., Liu, J., Hooke, J. A., Kovatich, A. J., Kvecher, L., Shriver, C. D., and Hu, H. (2019). PCA-PAM50 improves consistency between breast cancer intrinsic and clinical subtyping reclassifying a subset of luminal a tumors as luminal b. *Scientific Reports*, 9(1).

Ramsköld, D., Kavak, E., and Sandberg, R. (2011). How to analyze gene expression using RNA-sequencing data. In *Next Generation Microarray Bioinformatics*, pages 259–274. Humana Press.

Rao, M. S., Vleet, T. R. V., Ciurlionis, R., Buck, W. R., Mittelstadt, S. W., Blomme, E. A. G., and Liguori, M. J. (2019). Comparison of RNA-seq and microarray gene expression platforms for the toxicogenomic evaluation of liver from short-term rat toxicity studies. *Frontiers in Genetics*, 9.

Ravi, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., and Yang, G.-Z. (2017). Deep learning for health informatics. *IEEE Journal of Biomedical and Health Informatics*, 21(1):4–21.

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449.

Razzak, M. I., Naz, S., and Zaib, A. (2017). Deep learning for medical image processing: Overview, challenges and the future. In *Lecture Notes in Computational Vision and Biomechanics*, pages 323–350. Springer International Publishing.

Reese, S. E., Archer, K. J., Therneau, T. M., Atkinson, E. J., Vachon, C. M., de Andrade, M., Kocher, J.-P. A., and Eckel-Passow, J. E. (2013). A new statistic for identifying batch effects in high-throughput genomic data that uses guided principal component analysis. *Bioinformatics*, 29(22):2877–2883.

Reichart, R. and Rappoport, A. (2009). The nvi clustering evaluation measure. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 165–173, Stroudsburg, PA, USA. Association for Computational Linguistics.

Reynolds, A. P., Richards, G., and Rayward-Smith, V. J. (2004). The application of k-medoids and pam to the clustering of rules. In *Intelligent Data Engineering and Automated Learning - IDEAL 2004, 5th International Conference*, pages 173–178.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv e-prints*, page arXiv:1401.4082.

Ricketts, C. J., Cubas, A. A. D., Fan, H., Smith, C. C., Lang, M., Reznik, E., and et al. (2018). The cancer genome atlas comprehensive molecular characterization of renal cell carcinoma. *Cell Reports*, 23(1):313–326.e5.

Robertson, A. G., Kim, J., Al-Ahmadie, H., Bellmunt, J., Guo, G., and et al., A. D. C. (2017). Comprehensive molecular characterization of muscle-invasive bladder cancer. *Cell*, 171(3):540–556.e25.

Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2009). edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140.

Robison, K. (2010). Application of second-generation sequencing to cancer genomics. *Briefings in Bioinformatics*, 11(5):524–534.

Rohlf, F. J. and Sokal, R. R. (1981). *Biometry: the principles and practice of statistics in biological research*. Freeman New York.

Romero, I. G., Ruvinsky, I., and Gilad, Y. (2012). Comparative studies of gene expression and the evolution of gene regulation. *Nature Reviews Genetics*, 13(7):505–516.

Rose, K. (1998). Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239.

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

Rubin, D. B. (2004). *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Sadati, N., Zafar Nezhad, M., Babu Chinnam, R., and Zhu, D. (2019). Representation Learning with Autoencoders for Electronic Health Records: A Comparative Study. *arXiv e-prints*, page arXiv:1908.09174.

Saxe, A. M., Mcclelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *In International Conference on Learning Representations*.

Schell-Chaple, H. M., Puntillo, K. A., Matthay, M. A., and and, K. D. L. (2015). Body temperature and mortality in patients with acute respiratory distress syndrome. *American Journal of Critical Care*, 24(1):15–23.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schubert, E. and Rousseeuw, P. J. (2018). Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms. *arXiv e-prints*, page arXiv:1810.05691.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2016). mclust 5: Clustering, classification and density estimation using gaussian finite mixture models. *The R journal*, 8:289–317.

Seyednasrollah, F., Rantanen, K., Jaakkola, P., and Elo, L. L. (2015). ROTS: reproducible RNA-seq biomarker detector—prognostic markers for clear cell renal cell cancer. *Nucleic Acids Research*, 44(1):e1–e1.

Sharafoddini, A., Dubin, J. A., Maslove, D. M., and Lee, J. (2019). A new insight into missing data in intensive care unit patient profiles: Observational study. *JMIR Medical Informatics*, 7(1):e11605.

Shen, Y. and Gao, M. (2019). Brain Tumor Segmentation on MRI with Missing Modalities. *arXiv e-prints*, page arXiv:1904.07290.

Shickel, B., Tighe, P. J., Bihorac, A., and Rashidi, P. (2018). Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE Journal of Biomedical and Health Informatics*, 22(5):1589–1604.

Si, Y., Liu, P., Li, P., and Brutnell, T. P. (2013). Model-based clustering for RNA-seq data. *Bioinformatics*, 30(2):197–205.

Silva, A., Rothstein, S. J., McNicholas, P. D., and Subedi, S. (2019). A multivariate poisson-log normal mixture model for clustering transcriptome sequencing data. *BMC Bioinformatics*, 20(1).

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

Smid, M., Wang, Y., Zhang, Y., Sieuwerts, A. M., Yu, J., Klijn, J. G., Foekens, J. A., and Martens, J. W. (2008). Subtypes of breast cancer show preferential site of relapse. *Cancer Research*, 68(9):3108–3114.

Smith, A. F. M. and Gelfand, A. E. (1992). Bayesian statistics without tears: A sampling–resampling perspective. *The American Statistician*, 46(2):84–88.

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491.

Solomon, S. R. and Sawilowsky, S. S. (2009). Impact of rank-based normalizing transformations on the accuracy of test scores. *Journal of Modern Applied Statistical Methods*, 8(2):448–462.

Song, C., Liu, F., Huang, Y., Wang, L., and Tan, T. (2013). Auto-encoder based data clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 117–124. Springer Berlin Heidelberg.

Sorlie, T., Perou, C. M., Tibshirani, R., Aas, T., Geisler, S., Johnsen, H., and et al. (2001). Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, 98(19):10869–10874.

Stekhoven, D. J. and Buhlmann, P. (2011). MissForest–non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118.

Strauß, R., Ewig, S., Richter, K., König, T., Heller, G., and Bauer, T. T. (2014). The prognostic significance of respiratory rate in patients with pneumonia. *Deutsches Ärzteblatt international*.

Strehl, A. and Ghosh, J. (2003). Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617.

Stubbendick, A. L. and Ibrahim, J. G. (2003). Maximum likelihood methods for nonignorable missing responses and covariates in random effects models. *Biometrics*, 59(4):1140–1150.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., and et al. (2005). Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550.

Suk, H.-I., , Lee, S.-W., and Shen, D. (2013). Latent feature representation with stacked auto-encoder for AD/MCI diagnosis. *Brain Structure and Function*, 220(2):841–859.

Sun, L., Jia, K., Chan, T.-H., Fang, Y., Wang, G., and Yan, S. (2014). DL-SFA: Deeply-learned slow feature analysis for action recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., and et al. (2014). Going Deeper with Convolutions. *arXiv e-prints*, page arXiv:1409.4842.

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

Taylor, G. J. (1996). *Neural Networks and Their Applications*. John Wiley & Sons, Inc., USA, 1st edition.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423.

Tirosh, I., Izar, B., Prakadan, S. M., Wadsworth, M. H., Treacy, D., Trombetta, J. J., and et al. (2016). Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. *Science*, 352(6282):189–196.

Tomancak, P., Berman, B. P., Beaton, A., Weiszmann, R., Kwan, E., Hartenstein, V., and et al. (2007). Global analysis of patterns of gene expression during drosophila embryogenesis. *Genome Biology*, 8(7):R145.

Tongyoo, S., Viarasilpa, T., and Permpikul, C. (2018). Serum potassium levels and outcomes in critically ill patients in the medical intensive care unit. *Journal of International Medical Research*, 46(3):1254–1262.

Tran, M.-N., Nguyen, N., Nott, D., and Kohn, R. (2019). Bayesian deep net GLM and GLMM. *Journal of Computational and Graphical Statistics*, 29(1):97–113.

Trost, B., Moir, C. A., Gillespie, Z. E., Kusalik, A., Mitchell, J. A., and Eskiw, C. H. (2015). Concordance between RNA-sequencing data and DNA microarray data in transcriptome analysis of proliferative and quiescent fibroblasts. *Royal Society Open Science*, 2(9):150402.

Tschannen, M., Bachem, O., and Lucic, M. (2018). Recent Advances in Autoencoder-Based Representation Learning. *arXiv e-prints*, page arXiv:1812.05069.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494.

Van Buuren, S. (2018). *Flexible imputation of missing data*. CRC press.

Van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(1):1–67.

van de Vijver, M. J., He, Y. D., van 't Veer, L. J., Dai, H., Hart, A. A., Voskuil, D. W., and et al. (2002). A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999–2009.

van Laarhoven, P. J. M. and Aarts, E. H. L. (1987). Simulated annealing. In *Simulated Annealing: Theory and Applications*, pages 7–15. Springer Netherlands.

Vankayala, V. S. S. and Rao, N. D. (1993). Artificial neural networks and their applications to power systems—a bibliographical survey. *Electric Power Systems Research*, 28(1):67–79.

Venugopalan, J., Chanani, N., Maher, K., and Wang, M. D. (2019). Novel data imputation for multiple types of missing data in intensive care units. *IEEE Journal of Biomedical and Health Informatics*, 23(3):1243–1250.

Vidman, L., Källberg, D., and Rydén, P. (2019). Cluster analysis on high dimensional RNA-seq data with applications to cancer research - an evaluation study. *PLOS ONE*, 14(12):e0219102.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018:1–13.

Wang, N., Wang, Y., Hao, H., Wang, L., Wang, Z., Wang, J., and Wu, R. (2013). A bi-poisson model for clustering gene expression profiles by RNA-seq. *Briefings in Bioinformatics*, 15(4):534–541.

Wang, Y., Yao, H., and Zhao, S. (2016). Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242.

Wang, Z., Gerstein, M., and Snyder, M. (2009). RNA-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63.

Way, G. P. and Greene, C. S. (2018). Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 23:80–91.

Weinstein, J., Akbani, R., Broom, B., and et al. (2014). Comprehensive molecular characterization of urothelial bladder carcinoma. *Nature*, 507(7492):315–322.

Wittkopp, P. J. (2007). Variable gene expression in eukaryotes: a network perspective. *Journal of Experimental Biology*, 210(9):1567–1575.

Wolff, A., Bayerlová, M., Gaedcke, J., Kube, D., and Beißbarth, T. (2018). A comparative study of RNA-seq and microarray data analysis on the two examples of rectal-cancer patients and burkitt lymphoma cells. *PLOS ONE*, 13(5):e0197162.

Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., and et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37.

Xie, Y., Le, L., Zhou, Y., and Raghavan, V. V. (2018). Deep learning for natural language processing. In *Handbook of Statistics*, pages 317–328. Elsevier.

Yang, F., Ding, P., and Huang, R. (2015). Clinicopathological significance and potential drug target of CDH1 in breast cancer: a meta-analysis and literature review. *Drug Design, Development and Therapy*, page 5277.

Yang, K., Gao, J., and Luo, M. (2019). Identification of key pathways and hub genes in basal-like breast cancer using bioinformatics analysis. *OncoTargets and Therapy*, Volume 12:1319–1331.

Yang, L., Shen, Y., Yuan, X., Zhang, J., and Wei, J. (2017). Analysis of breast cancer subtypes by AP-ISA biclustering. *BMC Bioinformatics*, 18(1).

Yersal, O. (2014). Biological subtypes of breast cancer: Prognostic and therapeutic implications. *World Journal of Clinical Oncology*, 5(3):412.

Yeung, K. Y., Fraley, C., Murua, A., Raftery, A. E., and Ruzzo, W. L. (2001a). Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987.

Yeung, K. Y., Haynor, D. R., and Ruzzo, W. L. (2001b). Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318.

Young, M. D., Wakefield, M. J., Smyth, G. K., and Oshlack, A. (2010). Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11(2):R14.

Yuan, Y. and Yin, G. (2009). Bayesian quantile regression for longitudinal studies with nonignorable missing data. *Biometrics*, 66(1):105–114.

Zar, J. H. (1999). *Biostatistical analysis*. Pearson Education India.

Zhang, Q., Yang, L. T., Chen, Z., and Li, P. (2018). A survey on deep learning for big data. *Information Fusion*, 42:146–157.

Zhang, R., Lahens, N. F., Ballance, H. I., Hughes, M. E., and Hogenesch, J. B. (2014). A circadian gene expression atlas in mammals: Implications for biology and medicine. *Proceedings of the National Academy of Sciences*, 111(45):16219–16224.

Zhang, S., Yao, L., Sun, A., and Tay, Y. (2017). Deep Learning based Recommender System: A Survey and New Perspectives. *arXiv e-prints*, page arXiv:1707.07435.

Zhao, S., Fung-Leung, W.-P., Bittner, A., Ngo, K., and Liu, X. (2014). Comparison of RNA-seq and microarray in transcriptome profiling of activated t cells. *PLoS ONE*, 9(1):e78644.

Zhao, Y., Karypis, G., and Fayyad, U. (2005). Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery*, 10(2):141–168.

Zheng, S., Cherniack, A. D., Dewal, N., Moffitt, R. A., Danilova, L., Murray, B. A., and et al. (2016). Comprehensive pan-genomic characterization of adrenocortical carcinoma. *Cancer Cell*, 29(5):723–736.

Zou, B., Lampos, V., Gorton, R., and Cox, I. J. (2016). On infectious intestinal disease surveillance using social media content. In *Proceedings of the 6th International Conference on Digital Health Conference - DH '16*. ACM Press.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Zwiener, I., Frisch, B., and Binder, H. (2014). Transforming RNA-seq data to improve the performance of prognostic gene signatures. *PLoS ONE*, 9(1):e85150.

Zyprych-Walczak, J., Szabelska, A., Handschuh, L., Górczak, K., Klamecka, K., Figlerowicz, M., and Siatkowski, I. (2015). The impact of normalization methods on RNA-seq data analysis. *BioMed Research International*, 2015:1–10.