2017

Machine Learning Methods To Identify Hidden Phenotypes In The Electronic Health Record

Brett Kreigh Beaulieu-Jones
University of Pennsylvania, BRETTBJ@GMAIL.COM

Machine Learning Methods To Identify Hidden Phenotypes In The Electronic Health Record

Abstract
The widespread adoption of Electronic Health Records (EHRs) means an unprecedented amount of patient treatment and outcome data is available to researchers. Research is a tertiary priority in the EHR, where the priorities are patient care and billing. Because of this, the data is not standardized or formatted in a manner easily adapted to machine learning approaches. Data may be missing for a large variety of reasons ranging from individual input styles to differences in clinical decision making, for example, which lab tests to issue. Few patients are annotated at a research quality, limiting sample size and presenting a moving gold standard. Patient progression over time is key to understanding many diseases but many machine learning algorithms require a snapshot, at a single time point, to create a usable vector form. In this dissertation, we develop new machine learning methods and computational workflows to extract hidden phenotypes from the Electronic Health Record (EHR). In Part 1, we use a semi-supervised deep learning approach to compensate for the low number of research quality labels present in the EHR. In Part 2, we examine and provide recommendations for characterizing and managing the large amount of missing data inherent to EHR data. In Part 3, we present an adversarial approach to generate synthetic data that closely resembles the original data while protecting subject privacy. We also introduce a workflow to enable reproducible research even when data cannot be shared. In Part 4, we introduce a novel strategy to first extract sequential data from the EHR and then demonstrate the ability to model these sequences with deep learning.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Genomics & Computational Biology

First Advisor
Jason H. Moore

Second Advisor
Casey S. Greene

Keywords
deep learning, electronic health record, electronic phenotyping, machine learning, semi-supervised learning

Subject Categories
Bioinformatics | Genetics

MACHINE LEARNING METHODS TO IDENTIFY HIDDEN PHENOTYPES IN THE ELECTRONIC HEALTH RECORD

Brett Kreigh Beaulieu-Jones

A DISSERTATION

in

Genomics and Computational Biology

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2017

Supervisor of Dissertation

_____

Jason H. Moore, Ph.D.

Edward Rose Professor of Informatics

Co-Supervisor of Dissertation

_____

Casey S. Greene, Ph. D.

Assistant Professor of Pharmacology

Graduate Group Chairperson

_____

Li-San Wang, Ph.D.

Associate Professor of Pathology and Laboratory Medicine

Dissertation Committee:

John H. Holmes, Ph.D., Professor of Medical Informatics

Lyle H. Ungar, Ph.D., Professor of Computer and Information Science

Suchi Saria, Ph.D., Assistant Professor of Computer Science, Applied Math & Statistics, and Health Policy & Management

MACHINE LEARNING METHODS TO IDENTIFY HIDDEN PHENOTYPES IN THE ELECTRONIC HEALTH RECORD

Dedicated to Robyn Beaulieu, Stuart Jones, William Beaulieu and Elizabeth O'Neill

ACKNOWLEDGMENTS

I would like to thank Jason Moore and Casey Greene for being incredible mentors and advisors. I am grateful of my entire committee – John Holmes, Lyle Ungar, and Suchi Saria for providing guidance, expertise and advice along the way. The entire community at the University of Pennsylvania, and especially GCB created a stimulating and encouraging environment. I would like to especially thank my cohort of GCB students – Katie Siewart, Ian Mellis, Alex Amelie-Wolf, Lucy Shan and Salika Dunatunga for their help adapting to Penn and preparing for the prelim. I would also like to thank Sallie Ellison, Maureen Kirsch and Hannah Chervitz for their organizational help and assistance transitioning to Penn.

Both the Greene and Moore labs provided ideal environments for learning and personal development. I would like to specifically highlight Elizabeth Piette, Christian Darabos, and Ryan Uranowicz for their support and friendship especially during the Moore lab's move to Penn. In the Greene lab, Jie Tan and Gregory Way provided invaluable methods discussions and brainstorming. I would like to thank many others from both labs for their support and feedback along the way – Jaclyn Taroni, Daniel Himmelstein, Rene Zelaya, Kathy Chen, William LaCava, Randall Olson, Brian Cole, Patryk Orzechowski, Alicia Cutillo, Elisabetta Manduchi, Nadia Penrod, Molly Hall and Dan Herman.

I am grateful for all my collaborators at Penn and Geisinger Health System. Steven Wu, Chris Bauer, and Sarah Pendergrass, especially, provided great inspiration, expertise and were a pleasure to work with. I thank my mentee, Chris Williams, for taking on a challenging exploratory project and getting enough traction to develop a full-fledged solution and manuscript.

I would like to thank my family and friends for being an incredible support system. My parents, Robyn and Stuart for giving me both goals and the path to achieve them. My grandfather, William, for instilling the intellectual curiosity that has led to a love of science. My siblings – Kyle, Megan and Brendin for challenging, teaching and encouraging me every step of the way. My co-founders and colleagues – Alex LoVerde, Jeff Impey, Dan Stefanis, Andrew Stephan, Chris Sullivan and Anthony Russo for enabling me to make the transition to graduate school. Finally, Elizabeth O'Neill, for her great support and encouragement throughout this process.

ABSTRACT

MACHINE LEARNING METHODS TO IDENTIFY HIDDEN PHENOTYPES IN THE ELECTRONIC HEALTH RECORD

Brett Kreigh Beaulieu-Jones

Jason H. Moore

Casey S. Greene

The widespread adoption of Electronic Health Records (EHRs) means an unprecedented amount of patient treatment and outcome data is available to researchers. Research is a tertiary priority in the EHR, where the priorities are patient care and billing. Because of this, the data is not standardized or formatted in a manner easily adapted to machine learning approaches. Data may be missing for a large variety of reasons ranging from individual input styles to differences in clinical decision making, for example, which lab tests to issue. Few patients are annotated at a research quality, limiting sample size and presenting a moving gold standard. Patient progression over time is key to understanding many diseases but many machine learning algorithms require a snapshot, at a single time point, to create a usable vector form. In this dissertation, we develop new machine learning methods and computational workflows to extract hidden phenotypes from the Electronic Health Record (EHR). In Part 1, we use a semi-supervised deep learning approach to compensate for the low number of research quality labels present in the EHR. In Part 2, we examine and provide recommendations for characterizing and managing the large amount of missing data inherent to EHR data. In Part 3, we present an adversarial approach to generate synthetic data that closely resembles the original data while protecting subject privacy. We also introduce a workflow to enable reproducible

research even when data cannot be shared. In Part 4, we introduce a novel strategy to first extract sequential data from the EHR and then demonstrate the ability to model these sequences with deep learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Chapter 1. An introduction to extracting phenotypes using machine learning in the Electronic Health Record.

Portions of this chapter were adapted from: Beaulieu-Jones, Brett K. "Machine Learning for Structured Clinical Data." To appear in Advances in Biomedical Informatics (Book Editors: Dawn Holmes and Lakhmi Jain), Springer. Preprint: https://arxiv.org/abs/1707.06997

1.1. *Motivation for using machine learning on the structured Electronic Health Record*

Precision medicine has the potential to substantially change the way patients are treated in many facets of health care. Precision medicine is the idea of delivering personalized treatment and prevention strategies by considering the holistic patient, including their genetics, environment, and lifestyle. Machine learning using structured clinical data will likely play a large role in the success or failure of precision medicine. Specifically, machine learning using structure data can help in finding associations between a patient's genotype and phenotype, identifying similar patients and evaluating and predicting the efficacy of different clinical treatment strategies on a personalized level.

The amount of data collected in the clinic has rapidly expanded, the first EHRs are now more than 20 years old and the United States federal government mandated meaningful use of EHRs by 2014. According to the American Hospital Association, by

2015, 96% of acute care hospitals had implemented a certified EHR. Correspondingly, several top research institutions across the country have established departments or institutes in biomedical informatics using the EHR as a major data source in the past 5 years.

Smartphones, wearable devices and in-clinic diagnostic tools offer the ability to stream accurate measurements in real time. AliveCor received FDA approval in 2012 for its iPhone-based heart monitor using machine learning to detect Atrial Fibrillation in seconds. Billions of dollars in venture capital are currently being invested in companies, such as Grail, Foundation Medicine, and Guardant health, promising less invasive biopsies, or liquid biopsies, using machine learning to classify patients from circulating tumor cells in the bloodstream. Preventative wellness clinics, such as Forward, are emerging to characterize and track what it means to be healthy.

These are only a few examples of the many opportunities centered on patient data. Data for both evidence-based clinical decision making and computational research is becoming increasingly available and we must now develop new methods to preprocess and analyze this data at a matching rate.

1.2. *Common Uses of Machine Learning for Structured Clinical Data*

Each time a patient interacts with a health system, actions, notes, and measurements are recorded in the EHR. This wealth of data has made the EHR the primary source of structure clinical data. Three promising research applications of EHR data are:

1.) Patient clustering and disease stratification.

2.) Electronic phenotyping for genetic studies.

3.) Advising clinical treatment strategies.

These tasks can be performed using machine learning, but each task requires careful preprocessing of data and appropriate phrasing of the problem to utilize traditional machine learning methods. The nature of EHR data places emphasis on unsupervised clustering and semi-supervised classification. In this section, we discuss these common tasks and show examples where researchers have utilized machine learning effectively to guide discovery. There exist many great resources for understanding machine learning approaches as applied to general problems *(1)*. We concentrate on how to position relevant clinical questions and the challenges specific to the EHR that need to be solved in order to apply these powerful techniques.

1.2.1. *Patient Clustering and Disease Stratification*

As we learn more about the mechanisms and etiology of a disease, our diagnoses can become more precise, leading to the creation of disease subtypes. Historically, cancers were diagnosed based on their occurrence location and their reaction to different treatments. As the mechanisms of cancer are better understood, they are further categorized by their physiological nature. The progression of subtypes in lung cancer illustrates the increases in resolution over time for a previously poorly defined disease *(2)*. Beginning with a single diagnosis based on occurrence in the lung, it was later differentiated as small cell lung cancer and non-small cell lung cancer *(3, 4)*. Non-small cell lung cancer was then broken up into squamous cell carcinoma, adenocarcinoma, and large cell carcinoma. Today these subtypes continue to be broken up based on the genetic locations and pathways of associated risk variants.

What happens when physiological differences cannot easily be used to subtype disease? This is true with several metabolic disorders, for example, metabolic syndrome

has been redefined numerous times. It is associated with a wide range of comorbidities and presents in a clinically heterogeneous manner. These comorbidities, including coronary heart disease, diabetes, and stroke, represent an oversized risk to public health and increasingly unwieldy burden on the health care system. Despite this, metabolic syndrome's predictive value for cardiovascular events, disease prediction and progression is disputed and may not outperform the individual components that define it *(5)*. While the concept of identifying patients at high risk of developing diseases such as heart disease and diabetes for early intervention is an important one, metabolic syndrome in its current form fails to do this effectively.

Li et al. demonstrated the ability to identify disease subtypes of patients with a metabolic disorder, type 2 diabetes *(6)*. To do this they performed a topological analysis of 11,210 patients with type 2 diabetes at Mount Sinai Medical Center in New York. This topological analysis constructed a network of patients by connecting those most similar to each other. Using this they found three unique subtypes. Subtype 1 demonstrated the traditional observations of type 2 diabetes, hyperglycemia, obesity, and eye and kidney diseases. Subtype 2's main comorbidity was cancer, and subtype 3's unique comorbidities were neurological diseases. These subtypes are likely enriched for etiological differences; the disease likely operates differently in someone who develops cancer than someone who develops kidney disease. By developing a machine learning classifier to identify which subtype a patient is in as early as possible, clinicians may be able personalize treatment to reduce the odds of developing these more serious comorbidities.

Multiple sclerosis illustrates an area machine learning for disease stratification could be particularly useful. Multiple sclerosis was traditionally subtyped into Relapsing-Remitting MS and Progressive MS. In 2014, it was recommended that these subtypes be further divided into six total subtypes *(7)*. Unfortunately, the current strategies for determining subtype and thus treatment strategy require looking at the progression of the disease. This is essentially a retrospective diagnosis and means personalized treatment plans cannot be started until progression has been observed. Could unsupervised clustering be used to identify subtypes earlier on?

1.2.2. *Electronic phenotypes for Genetic Associations*

Genetic associations examine whether a genetic variant is associated with a specific trait. This specific trait, a phenotype, can be a moving target when dealing with the complexity of human disease. The trait is often a human defined disease. Those with the disease are labeled the case and those without the disease are considered controls. Early genetic associations using the electronic health record were performed with raw International Classification of Diseases (ICD) codes. ICD codes are recorded by physicians when diagnosing a patient with a condition, and are used to ensure proper billing and insurance reimbursement. ICD codes are published and updated by the World Health Organization and are primarily used for clinical billing purposes. Despite ICD-10 being initially published in 1994, ICD-9 codes are still commonly used in both clinical and research settings.

While ICD codes provide a clear, discrete endpoint for genetic associations, the use of billing codes can introduce unintentional biases to analyses. An ICD code may be added to an EHR in order to issue and receive insurance reimbursement for a test to

screen for the disease the ICD code represents. In this case, not only is the timing of diagnosis difficult to determine, but solely looking at the ICD codes for a patient is likely to introduce false positives. In addition, certain ICD codes are more easily reimbursed than others. When a clinician determines that a patient requires a treatment or test to increase their odds of a successful outcome, the clinician is incentivized to choose the ICD code most likely to allow them to effectively treat their patient.

Phenotype algorithms can be developed using the structured EHR to leverage both ICD codes and the rest of the of a patient's record. The eMERGE project is a national network which has deployed phenotype algorithms for over 40 diseases, over 500,000 EHRs and 55,000 patients with genetic data. Many of the phenotype algorithms are simple rules-based systems, for example: Type 2 Diabetes (Figure 1.1).

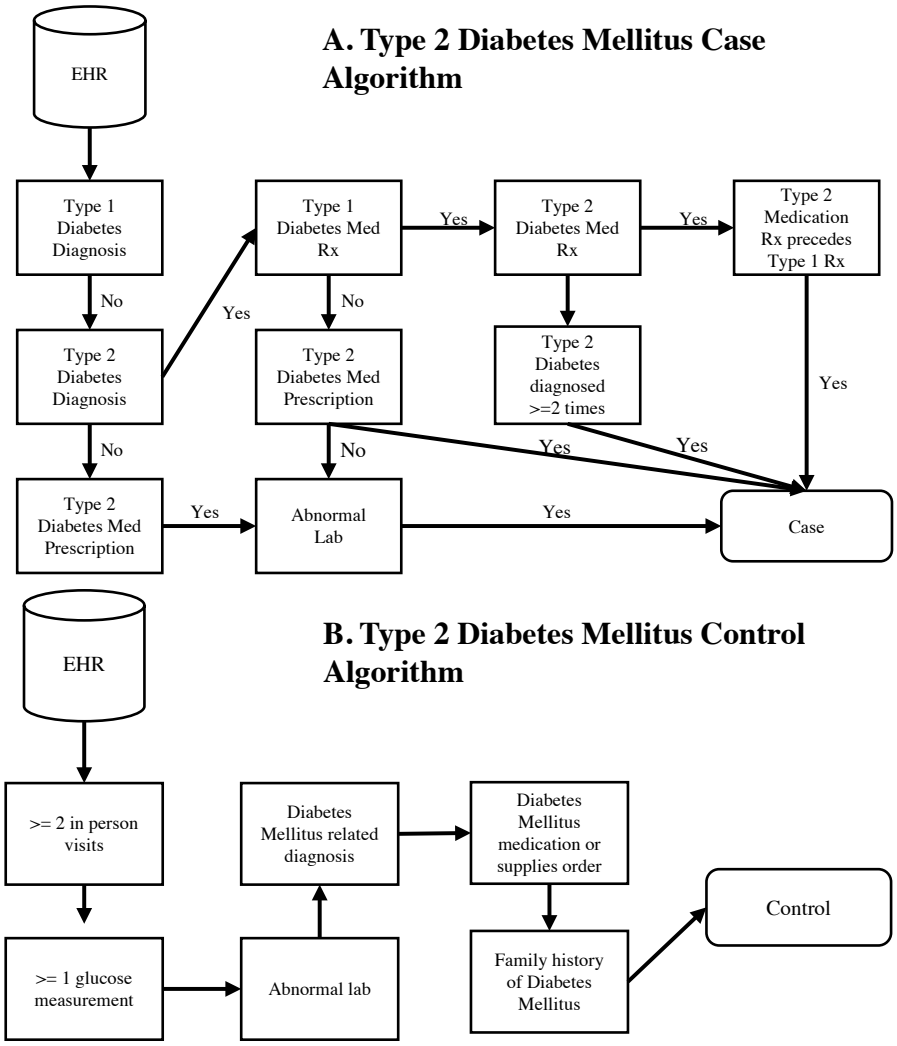


Figure 1.1: Phenotype Algorithms for Type 2 Diabetes Mellitus.
A.) Case selection from the EHR. B.) Control Selection from the EHR. Adapted from: (8).

An approach to study phenotype-genotype associations from the EHR are Phenome-wide association studies or PheWAS *(8)*. PheWAS use EHRs to define a phenome that can be linked back to individual genetic variants. The approach can discover gene-disease associations while identifying pleiotropic effects of individual SNPs. PheWAS generally uses the ICD9 codes to construct a phenotype. While primarily used for billing, these codes provide a set of discrete variables that can represent many phenotypes for a patient at the same time, providing greater resolution. Besides the repurposing of billing codes, a

major challenge of PheWAS is in understanding the functional mechanisms at work behind GWAS SNP matches. Stratification by the 4,841 different codes creates wide data, presenting statistical challenges in achieving adequate power. This challenge of achieving adequate power will be exacerbated by the transition to ICD10, with less historical data built up and the potential for over 16,000 codes. Continuing to increase open data access will allow researchers to utilize a more accurate phenotypic representation while lessening the burden of statistical challenges. Coding systems, unlike patient notes or genomic data should be easier to anonymize, aggregate, and distribute.

ICD billing codes can be biased, as evidenced by phenotype algorithms having multiple steps to catch errors for both case and control status, using billing codes alone may cause misclassification of phenotypes. The misclassification of phenotypes substantially reduces the power to detect linkage in case-control studies. With 1% phenotypic misclassification up to 10% of the power is lost, and with 5% phenotypic misclassification, the power is reduced by approximately two-thirds *(9–11)*. Misclassification can occur for a variety of reasons including misdiagnosis/clinical error, clerical error, or lack of scientific knowledge about the disease in question.

Labbe et al. showed increased linkage by clustering lifetime symptoms in schizophrenia and bipolar disease to form more homogenous phenotypes. Separating cases by the symptoms of psychiatric diseases compensates for the inability to subtype these diseases by physical properties *(12)*. This is important due to the deficit of physiological understanding for these diseases. Labbe et al. also included familial information to understand the heritability of these diseases. When looking at subtypes

that show a strong familial aggregation they observed higher linkage scores. By looking at ancestral histories for subtypes, the expected heritability could be better estimated resulting in a reduction of "missing heritability."

Phenotypic subtyping was also used successfully in the analysis of genetic variants responsible for the severe development regression and stereotypical hand movements of Rett syndrome. Causal mutations were found in the FOXG1 and MECP2 genes and deletions at the 22q11.2 locus *(13)*.

Each of these examples point towards the promise of using machine learning to cluster patients based on their EHRs to identify disease subtypes or more homogenous groups of patients for use in association studies.

1.2.3. *Clinical Recommendations*

The availability of data and advances in biomedical informatics have helped to make medicine increasingly evidence based and in some cases entirely data driven. Clinicians and researchers now have the ability to leverage millions of data points when designing and determining treatment best practices. The New England Journal of Medicine recently held the SPRINT data analysis to "use the data underlying a recent article to identify a novel clinical finding that advances medical science." The original clinical trial sought to see whether intensive management of systolic blood pressure (<120 mm Hg) was more effective than standard management (<140 mm Hg). The original trial was stopped early due to the success of the intensive management strategy in reducing cardiovascular events. The data from the trial was released as a challenge where teams used machine learning approaches (primarily rules based) to provide personalized recommendations.

More personalized treatment strategies are a popular use of machine learning in the EHR. This can be driven by genomics (pharmacogenomics), or simply by sub setting patients based of attributes (race, BMI, etc.). Wiley et al. demonstrate the importance of training an algorithm on a population similar to the application population *(14)*. In their case it was necessary to extract the percent African ancestry from the genome instead of self reported race in order to improve the model fit.

Due to the inherent risk of adjusting clinical treatment strategies, many of the early applications of machine learning in health systems have been seen in academic research (retrospective analysis, drug development, pharmacogenomics) and for things like resource usage. For example, how likely is a patient coming into the ER to need an ICU bed? Increasingly machine learning methods are likely to be applied to clinical decisions including providing prognosis information for shared decision making strategies. Deep learning, in particular, is becoming an increasingly tool for drug discovery and development *(15)*.

1.3. *Challenges to Using Machine Learning on Structured Clinical Data*

1.3.1. *Limited research labeled "gold-standards"*

Large institutions and health care systems can have EHRs containing millions of patients and billions of measurements. Despite the size of these data, electronic phenotyping requires a gold standard to validate accuracy. This gold standard often requires time consuming, manual clinician review and is thus expensive.

In addition, the selection of cases and controls can unintentionally create biases in downstream algorithms and analyses. It is often easiest to select the most severe cases

and the healthiest controls. In these circumstances researchers can have the greatest confidence they are accurately selecting a true case or control. Unfortunately, this creates a biased training set where it is difficult to differentiate between less severe cases and less healthy controls. Figure 1.2 shows an illustration of a simulated dataset where the first two principal components happen to represent the degree of the case phenotype. If the most severe cases are selected, a classifier trained to distinguish between cases and controls is unlikely to generalize well. If less severe cases are chosen, there may be issues with mislabeled cases.

Figure 1.2: Example of case vs. control selection.

Simulated disease severity plot where the 2 principal components stratify patients according to severity.

An example of another bias can be seen in the Type 2 Diabetes Mellitus algorithm, controls must have at least 1 glucose measurement (Figure 1.1 B). For a young patient, this means that a clinician must have had reason to suspect that the patient's glucose could be abnormal and thus could bias controls to patients who "look" like they are at a high risk for developing Type 2 Diabetes.

Because patients move between health systems, a patient may not be diagnosed in the system they are treated in and may only have a partial history. Some methods for controlling for incomplete histories can result in smaller sample sizes. It is common to include only patients who have a visit in the system prior to the diagnosis of the phenotype of interest. While this can help to determine the diagnosis date for a disease, it excludes anyone who was diagnosed on their first visit to a particular health system.

1.3.2. *Missing Data*

The average patient is unlikely to have measurements for the clear majority of fields in the EHR. It does not make sense logistically or economically to administer every test to a seemingly healthy patient. There are three primary types of missing data:

1.) Missing Completely at Random (MCAR) – when data is missing in a completely unrelated way to the values of both the observed and the unobserved data.

2.) Missing at Random (MAR) – when the data is missing based on the observed data, when other fields in the EHR indicate whether the value will be present or absent.

3.) Missing Not at Random (MNAR) – when the data is missing based on the values of the unobserved data.

Figure 1.3 shows the ability to use a random forest to predict whether a lab value will be present or absent based on other lab values. Unsurprisingly data that is MCAR cannot

be predicted (Figure 1.3 A), and data that is either MAR or MNAR (Figure 1.3 B, 1.3 C) can be predicted with an accuracy significantly greater than random. In practice, most missing data in the EHR tends to be of the MAR variety. Clinicians must decide which measurements are relevant and fiscally responsible, irrelevant tests are wasteful and it does not make sense to subject patients to unnecessary discomfort. The clinician is making these decisions based on the observations they make, so when data is missing it is related to the observed data.

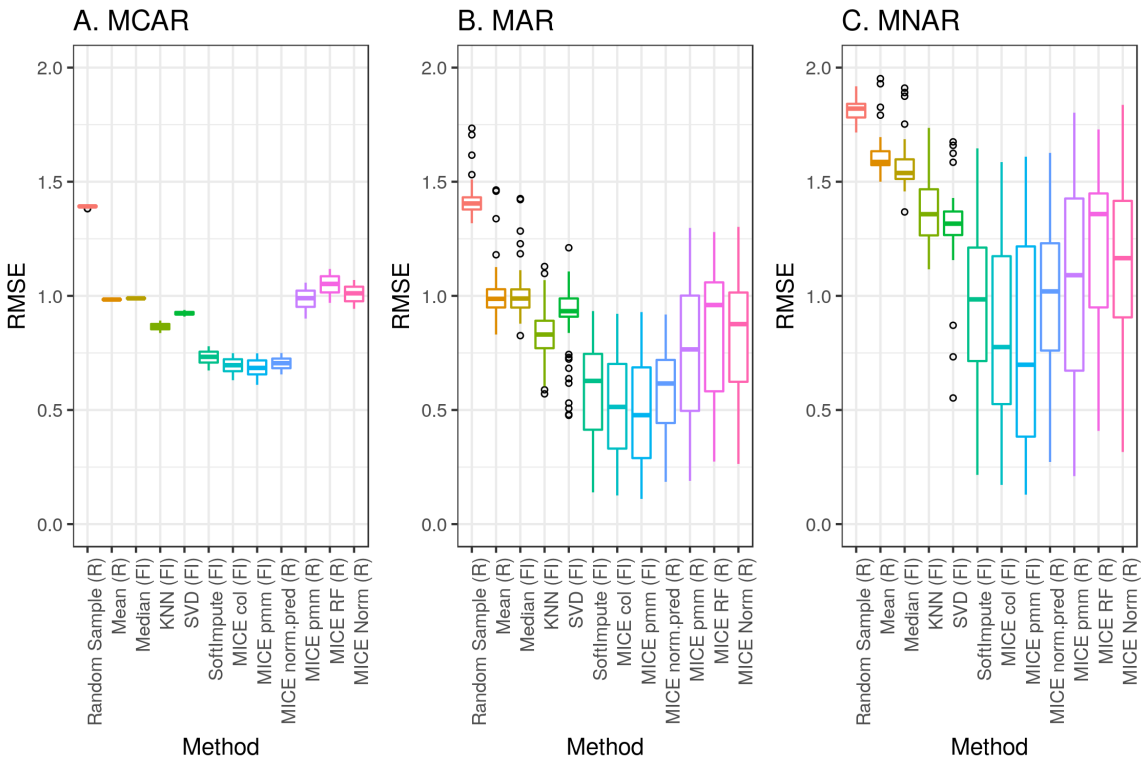


Figure 1.3: Predicting the presence of data under different missing data mechanisms.

A.) Data that is missing completely at random cannot be predicted. **B & C.)** Data that is missing at random and missing not at random can be predicted at better than random accuracy (to appear *(16)*).

MCAR data is less likely to present issues to downstream analyses than data that is either MAR or MNAR, but if not handled correctly all three types of missing data can

introduce unintentional biases to all sorts of downstream analyses including machine learning. Machine learning algorithms often expect a complete matrix as input and are not designed to handle null values. This often leads to researchers performing one of three options:

1.) Perform feature selection of relevant features and use only complete cases, or patients that have values for all features.

2.) Modify the algorithm to accept null inputs (often by ignoring them) or

3.) Perform imputation to predict what the value for a feature would be.

Each of these options have several pros and cons and can have unintended effects on machine learning. When performing complete case analysis after feature selection, the features included can lead to including either more severe cases or cases that were harder to diagnose. Imagine a disease that is diagnosed by a laboratory measurement where values over 10 conclusively indicate you have the disease but values between 8 and 10 require an additional test. If the additional test is included in the features selected, the complete cases are now only the patients that were harder to diagnose. When modifying an algorithm to accept null inputs, the researcher needs to be careful that the algorithm does not disproportionately learn to depend on patients that have all of the measurements or only the most complete measurements. If the algorithm relies on patients with all of the measurements, many of the same issues that arise in complete case analysis repeat. If the algorithm learns to ignore rare measurements it can miss signal. For example, in an analysis of different treatment options, say 40 of 10,000 patients suffered a fairly rare but severe adverse event. Without careful monitoring the algorithm may not place enough importance on this feature despite the fact this outcome is disproportionately important.

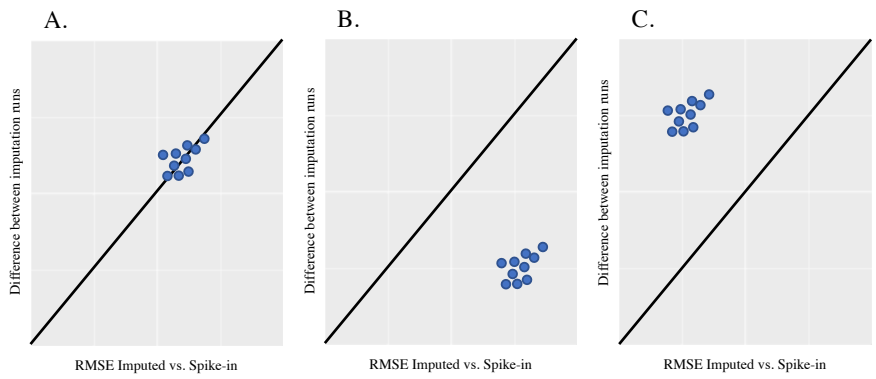


Figure 1.4: Comparison between spike-in accuracy and variation between imputation runs.

Imputation can be effective in the EHR because many missing values can be inferred by omission and just knowing whether a value was present or absent can be useful. If a patient has never had a chest x-ray, it is unlikely that their physician suspects a broken rib cage. This information can be provided to downstream machine learning algorithms by performing imputation. It is, however, very important to carefully analyze the results of imputation. Oftentimes much can be learned simply by looking at which methods are the most accurate. Direct accuracy can be measured by spiking in missing values to replace known values, imputing these spiked-in values and measuring their difference from the real values. Despite this direct accuracy should only be used as a benchmark, and it is important to analyze the effect of imputation on the downstream analyses you are performing (Figure 1.4).

For example, mean imputation may perform strongly in an analysis using spiked-in missingness but remove all variance from the imputed values for a feature (Figure 1.4 C). Other evaluation criteria, such as, comparing the variance between imputed values of different imputation runs and the difference between imputed values and real values. Ideally these values would be highly correlated in order to maintain the variance

structure. Popular imputation methods for EHRs include K-Nearest Neighbors, Singular Value Decomposition and Multiple Imputation by Chained Equations.

All three of these strategies for handling missing data may introduce bias when performing EHR-based analyses. It is important to consider potential effects and ideally to utilize multiple strategies and examine the differences.

1.3.3. *Privacy, Reproducibility, and Data Sharing*

Patient privacy needs to be a focus of any secondary use of EHRs. Because a patients EHR is 'de-identified' does not mean that it is anonymous. Latanya Sweeney demonstrated this emphatically when the Massachusetts Group Insurance Commission released de-identified data on state employees *(17)*. These records included each hospital visit and Sweeney re-identified several patients including the former Governor of Massachusetts. Sweeney did this from his birth date, zip code and sex alone, and to prove a point mailed the Governor a copy of his personal records. The task of re-identification has been shown possible in several other cases where data holders attempted to share their data, including the Netflix challenge. Narayana and Shmatikov were able to de-anonymize users in the Netflix challenge by linking their viewing histories with popular movie review sites. For users who had rated more than 6 movies, they were able to do this with greater than 90% accuracy *(18)*. This, in part, led to Netflix canceling the second iteration of its popular recommendation contest following a privacy lawsuit.

Caution needs to be taken even when the actual data is not released. Deep learning models can have many millions of parameters, allowing adversaries to perform membership inference attacks in order to determine whether a user was a member of the study or not. Shokri et al. *(19)* demonstrate greater than 90% precision even with 30,000

examples in the training set. They do this by examining the trained parameters of a deep neural network trained on the CIFAR-100 dataset. Even without the model, enterprising adversaries performed membership inference attacks with only black-box access to the target model through an API. Shokri et al. again demonstrated this on various purchase history datasets made available through Amazon and Google APIs.

One approach to adding privacy protection is called "Differential Privacy" *(20)*. Differential Privacy is a robust, meaningful and mathematical rigorous definition of privacy which operates under the knowledge that data cannot be fully anonymized and remain useful. If you remove all of the signal in a dataset to anonymize, machine learning methods are fruitless. If you keep any signal at all, there is a chance an adversary will able to discover information about the members of the dataset. The goal of differential privacy is to find a balance between an acceptable risk, the privacy budget, and usefulness of the data. It attempts to minimize the likelihood an adversary can perform a membership inference attack to determine if a subject is in a dataset. It works by adding a plausible deniability of any outcome by inserting random noise into the information made available. If balanced, meaningful answers can be interrogated from the data while greatly reducing the risk that any member of a study is harmed by de-identification. A classic example and simple way to think about differential privacy is to imagine a study where participants are told to answer a question. Before answering the question they flip a coin, if the coin lands heads, they give the real answer, the truth. If the coin lands tails, they answer randomly by flipping an additional coin and responding yes if it lands heads and no if it lands tails.

Simmons et al. used a variant of differential privacy to enable privacy preserving genome wide association studies even when there is significant population stratification. Genomic data has a high dimensionality and relatively low signal to noise ratio making de-identification or other attempts at masking individual records impractical. They demonstrate the ability to allow users to query summary statistics while minimizing privacy risks. This is a particularly interesting application because while genome sequencing prices have rapidly decreased, the combined costs of recruitment and sequencing are a major barrier to this type of research. This method allows for increased sharing of valuable, difficult to obtain datasets. Differential privacy is a rapidly growing area, we suggest "The Algorithmic Foundations of Differential Privacy" *(21)* as a starting point if interested in implementing differential privacy.

Privacy challenges can make sharing data prohibitively difficult. This in turn presents challenges in reproducing work from other researchers. Even if source code is shared, researchers attempting to reproduce original research generally can only compare final results. This means that even if a protocol of a paper is well written and described, if it has 100 steps, a researcher attempting to reproduce cannot be sure where their results diverged. Because of this challenge we strongly advocate publishing intermediate results. This can help narrow down divergences to a few steps, was it the data? The preprocessing? The actual analysis? The plotting into charts? One way to release intermediate results without adding a large amount of additional work is to use continuous integration to run the analysis and export the log file *(22)*.

1.3.4. *Longitudinal Data*

A key attribute and potential strength of EHRs is the ability to track the way a patient progresses over time. Early moving caregivers such as Geisinger Health System implemented initial EHRs over twenty years ago but fully utilizing this longitudinal presents challenges to researchers.

Longitudinal EHR data are often irregular time series. Measurements are recorded at irregular times, can be mixed type (continuous, ordinal, categorical), require feature extraction (images, free text). It is common for researchers to take a single time point (i.e. current time, set time after diagnosis etc.) and use this as the single end point or label for machine learning analyses. This can be problematic when patients have arrived at that point through very different routes. For example, if using a systolic blood pressure as an end point, one patient may be on an intensive blood pressure management protocol while another with the same blood pressure may have never taken medication. In the SPRINT clinical trial there were patients on as many as seven medications to manage blood pressure *(23)*, if unmedicated these patients would almost definitely have significantly higher measurements. One method researchers use to remediate this issue is to derive statistics to represent the time series, such as taking the median value. This can be insufficient when the way clinicians choose to observe and treat patients based on data either not recorded in the electronic health record, in the unstructured data or in fields not selected for inclusion can also bias the labels. For example, if patient A has a single normal white blood cell count, and patient B has had a monthly count every month for the past 5 years. A clinician could have been checking to see if patient A showed an increased white blood cell count after a surgery suspecting a possible infection. In

contrast, the repeated measurements for patient B indicate the clinician may have a reason to believe patient B is immunocompromised or may become immunocompromised due to a virus or adverse reaction to a medication and is using the white blood cell count to monitor this. Despite the patients having relatively equal white blood cell counts, using this single value as a label is clearly inadequate to represent the complete state of the patient. For this specific case deriving a panel of statistics including features such as the count and variance of the measurement could help to better represent the current state of a patient. Recent work takes this further to calculate disease and patient trajectories by generating networks of the way a patient or disease progresses over time. Jensen et al. demonstrated this using 6.2 million patients from Danish National Patient Registry to cluster patients based on time dependent disease diagnoses *(24)*. These disease diagnoses were extracted from patterns of ICD-9 codes on patient's EHRs. This method creates a visualization of patient trajectories and allows for analyses of co-morbidities observed in health systems in order to identify important patterns that indicate the potential for more severe outcomes. Further work in this field could move beyond billing codes to allow for increased resolution of patient trajectories.

Chapter 2. Semi-Supervised Learning of the Electronic Health Record with limited "gold-standard" labels.

This chapter was originally published as: Beaulieu-Jones, Brett K., Casey S. Greene, and the Pooled Resource Open-Access ALS Clinical Trials Consortium. "Semi-supervised learning of the electronic health record for phenotype stratification." *Journal of biomedical informatics* 64 (2016): 168-178. doi: 10.1016/j.jbi.2016.10.007

B.K.B.-J. and C.S.G. conceived the study and designed the solution. B.K.B.-J. implemented and performed the analysis. B.K.B.-J. and C.S.G. wrote, revised and approved the manuscript.

Data used in the preparation of this article were obtained from the Pooled Resource Open-Access ALS Clinical Trials (PRO-ACT) Database. As such, the following organizations and individuals within the PRO-ACT Consortium contributed to the design and implementation of the PRO-ACT Database and/or provided data, but did not participate in the analysis of the data or the writing of this report: Neurological Clinical Research Institute, MGH, Northeast ALS Consortium, Novartis, Prize4Life, Regeneron Pharmaceuticals, Inc., Sanofi, Teva Pharmaceutical Industries, Ltd.

2.1. *Abstract*

Patient interactions with health care providers result in entries to electronic health records (EHRs). EHRs were built for clinical and billing purposes but contain many data points about an individual. Mining these records provides opportunities to extract electronic phenotypes, which can be paired with genetic data to identify genes underlying common human diseases. This task remains challenging: high quality phenotyping is costly and requires physician review; many fields in the records are sparsely filled; and our definitions of diseases are continuing to improve over time. Here we develop and

evaluate a semi-supervised learning method for EHR phenotype extraction using denoising autoencoders for phenotype stratification. By combining denoising autoencoders with random forests we find classification improvements across multiple simulation models and improved survival prediction in ALS clinical trial data. This is particularly evident in cases where only a small number of patients have high quality phenotypes, a common scenario in EHR-based research. Denoising autoencoders perform dimensionality reduction enabling visualization and clustering for the discovery of new subtypes of disease. This method represents a promising approach to clarify disease subtypes and improve genotype-phenotype association studies that leverage EHRs.

2.2. *Introduction*

Biomedical research often considers diseases as fixed phenotypes, but many have evolving definitions and are difficult to classify. The electronic health record (EHR) is a popular source for electronic phenotyping to augment traditional genetic association studies, but there is a relative scarcity of research quality annotated patients *(25)*. Electronic phenotyping relies on either codes designed for billing or time intensive manual clinician review. This is an ideal environment for semi-supervised algorithms, performing unsupervised learning on many patients followed by supervised learning on a smaller, annotated, subset. Denoising autoencoders (DAs) are a powerful tool to perform unsupervised learning *(26)*. DAs are a type of artificial neural network trained to reconstruct an original input from an intentionally corrupted input. Through this training they learn higher-level representations modeling the structure of the underlying data. We sought to determine whether applying DAs to the EHR could reduce the number of

annotated patients required, construct non-billing code based phenotypes and elucidate disease subtypes for fine-tuned genetic association.

The United States federal government mandated meaningful use of EHRs by 2014 to improve patient care quality, secure and communicate patient information, and clarify patient billing *(27, 28)*. Despite not being designed specifically for research, EHRs have already proven an effective source of phenotypes in genetic association studies *(29, 30)*. Initially, phenotypes were hand designed based on manual clinician review of patient records. These studies were limited by the time and cost inherent in manual review *(31, 32)*, but DAs can make use of unlabeled data. After unsupervised pre-training, the trained DA's hidden layer can be used as input to a traditional classifier to create a semi-supervised learner. This allows the DA to learn from all samples, even those without labels, and requires only a small subset to be annotated. Today, phenome-wide association studies (PheWAS) are the most prevalent example of EHR phenotyping, proving particularly effective at identifying pleiotropic genetic variants *(33)*. PheWASs often use algorithms based on the International Classification of Disease (ICD) codes to construct a phenotype. This coding system was designed for billing, not to capture research phenotypes. DA constructed features are combinations of many components of clinical data and may provide a more holistic view of a patient than billing codes alone.

Through extensive study, disease diagnoses can become more precise over time *(2–4, 34, 35)*. Cancers, for example, were historically typed by occurrence location and the efficacy of different treatments. As the mechanisms of cancer are better understood, they are further categorized by their physiological nature. The progression of subtypes in lung cancer illustrates this increased understanding over time *(34)*. Beginning with a single

diagnosis based on occurrence in the lung, lung cancer has been divided into dozens of subtypes over several decades based on histological analysis, and genetic markers *(2–4, 35)*. The unsupervised nature of DAs means that even if the definitions of a disease change, they would not need to be retrained. The ability to identify more homogenous phenotypes showed increased genotype to phenotype linkage in schizophrenia, bipolar disease *(12)*, and Rett Syndrome *(13, 36–38)*. Furthermore, type 2 diabetes subtypes have been discovered using topological analysis of EHR patient similarity *(6)*. The dimensionality reduction possible with a DA makes clustering and visualization more feasible. Subtyping exposes disease heterogeneity and may contribute to additional physiological understanding.

Previous work in semi-supervised learning of the EHR relies on closed source commercial software *(6)*, and natural language processing of free text fields to match clinical diagnosis *(39, 40)*. We are not aware of any previous work performing semi-supervised classification and clustering from quantitative structured patient data.

2.3. *Methods*

We developed an approach, entitled "Denoising Autoencoders for Phenotype Stratification (DAPS)," that constructs phenotypes through unsupervised learning. This generalized phenotype construction can be used to classify whether patients have a particular disease or to search for disease subtypes in patient populations. To evaluate DAPS, we created a simulation framework with multiple hidden factors influencing potentially overlapping observed variables. We evaluated the reduced DA models against feature-complete representations with popular supervised learning algorithms. These evaluations covered both complete datasets, as well as the more realistic cases of

incompletely labeled and missing data. We developed a technique that uses the reduced feature-space of the DA to visualize potential subtypes. Finally, we evaluate DAs ability to predict ALS patient survival in both classification and clustering tasks. Each of these is fully described below and full parameters included in sweeps are available in the supplementary materials.

Source code to reproduce each analysis is included in our repository (https://github.com/greenelab/DAPS) *(41)* and is provided under a permissive open source license (3-clause BSD). A docker build is included with the repository to provide a common environment to easily reproduce results without installing dependencies *(42)*. In addition, Shippable, a continuous integration platform, is used to reanalyze results in a clean environment and generate figures after each commit *(43)*.

2.3.1. *Unsupervised Training with Denoising Autoencoders*

DAs were initially introduced as a component in constructing the deep networks used in deep learning *(44)*. Deep learning algorithms have become the dominant performers in many domains including image recognition, speech recognition and natural language processing *(45–50)*. Recently they have also been used to solve biological problems including tumor classification, predicting chromatin structure and protein binding *(26, 51, 52)*. DAs showed strong performance early in the deep learning revolution but have been surpassed in most domains by convolutional neural networks or recurrent neural networks *(44)*. While these complex deep networks have surpassed the performance of DAs in these areas, they rely on strictly structured relationships such as the relative positions of pixels within an image *(47, 53)*. This structure is unlikely to exist in the EHR. In addition, complex deep networks are notoriously hard to interpret. DAs

are easily generalizable, benefit from both linear and nonlinear correlation structure in the data, and contain accessible, interpretable, internal nodes *(26)*. Oftentimes the hidden layer is a "bottle-neck", a much smaller size than the input layer, in order to force the autoencoder to learn the most important patterns in the data *(53)*.

We used the Theano library *(54, 55)* to construct a DA consisting of three layers, an input layer *x*, a single hidden layer *y*, and a reconstructed layer *z (44)* (Figure 2.1A). Noise was added to the input layer through a stochastic corruption process, which masks 20% of the input values, selected at random, to zero.

The hidden layer y was calculated by multiplying the input layer by a weight vector *W*, adding a bias vector *b* and computing the sigmoid (Formula 1). The reconstructed layer *z* was similarly computed using tied weights, the transpose of *W* and *b* (Formula 2). The cost function is the cross-entropy of the reconstruction, a measure of distance between the reconstructed layer and the input layer (Formula 3).

$$y = s(Wx + b) \quad \text{(Formula 1)}$$

$$z = s(W'y + b') \quad \text{(Formula 2)}$$

$$cost = -\sum_{k=1}^{d}[x_k \log(z_k) + (1 - x_k)\log(1 - z_k)] \quad \text{(Formula 3)}$$

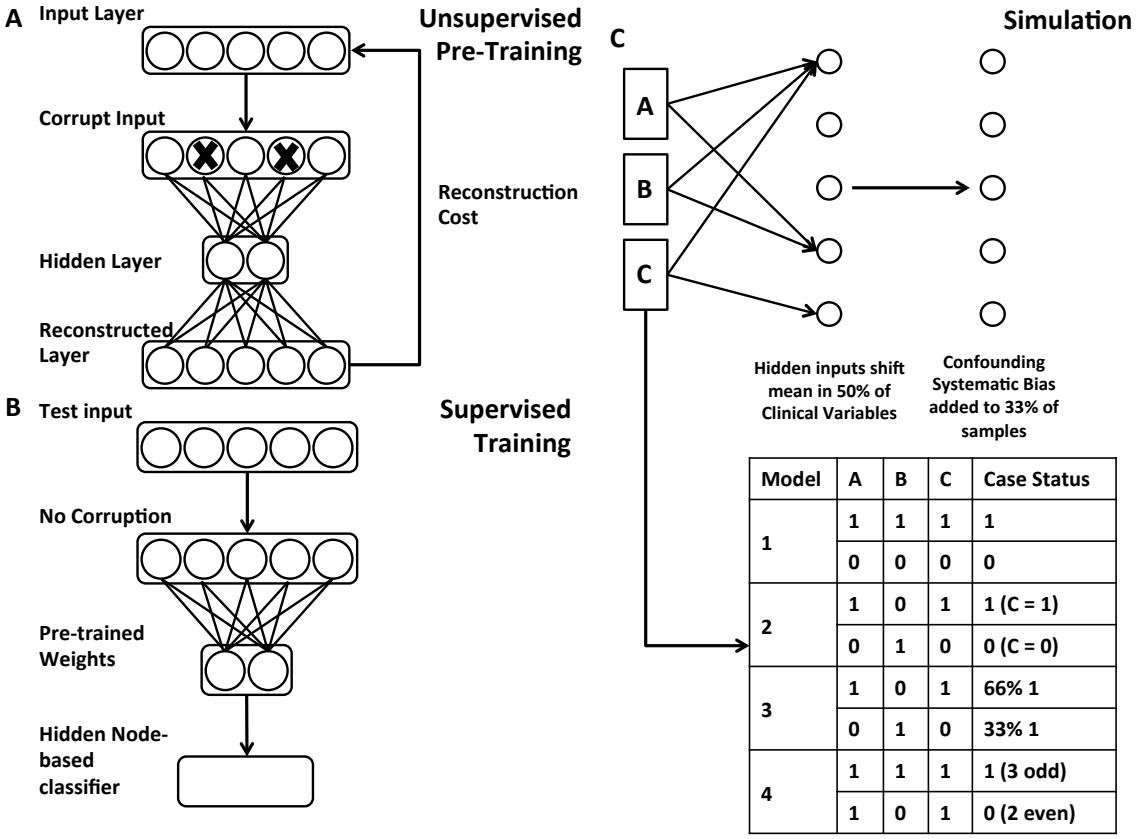
Stochastic gradient descent was performed for 1000 training epochs, at a learning rate of 0.1. Hidden layers of two, four, eight and sixteen hidden nodes were included in the parameter sweep with a 20% input corruption level. Vincent et al. *(44)* provide a through explanation of training for DAs without missing data.

In the event of missing data, the cost calculation was modified to exclude missing data from contributing to the reconstruction cost. A missingness vector *m* was created for each input vector, with a value of 1 where the data is present and 0 when the data is

missing. Both the input sample *x* and reconstruction *z* were multiplied by *m* and the cross-entropy error was divided by the sum of the *m*, the number of non-missing features to get the average cost per feature present (Formula 4). This allowed the DA to learn the structure of the data from present features rather than imputation.

$$cost = -\sum_{k=1}^{d}[x_k \log(z_k)\ m_k + (1 - x_k)\log(1 - z_k)\, m_k] \,/\, count(m)$$

(Formula 4)



Model	A	B	C	Case Status
1	1	1	1	1
	0	0	0	0
2	1	0	1	1 (C = 1)
	0	1	0	0 (C = 0)
3	1	0	1	66% 1
	0	1	0	33% 1
4	1	1	1	1 (3 odd)
	1	0	1	0 (2 even)

Figure 2.1: Diagram of Denoising Autoencoder and Simulation Procedure.

A.) Network diagram of DAs used for unsupervised pre-training. Input data is intentionally corrupted and then weights and biases are learned to minimize reconstruction cost when mapping the data to a hidden layer and back to a reconstructed layer. **B.)** Supervised classification occurs using the pre-trained DA hidden nodes as input to a traditional classifier. **C.)** Simulation model with example cases and controls under each rule set.

2.3.2. *Supervised Denoising Autoencoder Classifier*

To convert the DA to a supervised classifier, we first trained the DA in an unsupervised fashion (pre-training) (Fig 2.1 A). We then applied a variety of traditional machine learning classifiers including, decision trees, random forests, logistic regression, nearest neighbors and support vector machines to the pre-trained unsupervised hidden layer values, y, of the DA (Figure 2.1 B). Random forests applied to DA hidden nodes (DA+RF) were shown for all comparisons. Predictive performance was measured by comparing the AUROC using stratified 10-fold cross validation. The Scikit-learn library was used for the traditional classifiers *(56)*. The Support Vector Machine uses a radial basis function kernel, with a penalty parameter of 1. The nearest neighbors classifier uses a k-value of 5 and the random forest uses 10 estimators. These parameters achieved optimal performance in a preliminary parameter sweep.

2.3.3. *Simulation Framework*

We designed four simulation models to evaluate algorithmic performance. These simulations were not designed to perfectly recapitulate EHR data. Instead they are designed to capture a variety of complexity in order to identify algorithmic strengths and weaknesses with known underlying models.

To simulate patients, first clinical observations were generated by first drawing random samples from a normal distribution. Next hidden input effects were generated in accordance with one of four simulation models. When turned on these hidden input effects shift 1 to N observed clinical variables with replacement (Figure 2.1 C). Shifted clinical features were chosen at random, but consistent for all patients. Case-control status was determined by rules applied to the hidden input effects, where 1 represents the

effect being on and 0 represents the effect being off. Next, a confounding systematic bias was added to a random subset (33%) of the patients as a source of additional noise to simulate the variance accompanying data created by physicians, labs, hospitals or other spurious effects.

There are four models defining hidden input effect rules to determine case-control status:

1. **All together/all relevant.** Individuals have the same value (0 or 1) for all hidden input effects. Controls have all hidden effects set to 0. Cases have all hidden effects set to 1. A model capturing any hidden input will be able to predict case/control status in this scenario. This is a test of whether each method can recognize any of the hidden effects.
2. **All independent /single effect relevant.** Individuals have 0 to N (specified per simulation) hidden input effects chosen at random. One arbitrary effect (the last one) is used to determine case-control status. In controls, this is 0. In cases, this is 1. A model capturing the relevant hidden input will be able to predict case/control status in this scenario. This is a test of whether each method can recognize the important hidden effect when there are multiple shifted distributions.
3. **All independent/percentage based.** Individuals have 0 to N (specified per simulation) of hidden input effects chosen at random set to 1. The percentage of hidden input effects on represents the probability of the patient being a case. A model capturing more hidden effects will be able to more accurately predict case/control in this scenario. This is a test of whether each method can perform effectively without a hard-rule based model, and could represent a disease with incomplete penetrance.
4. **All independent/complex rule based.** Individuals have 0 to N (specified per simulation) of hidden inputs chosen at random set to 1. The sum of hidden effects determines case-control status (cases are even, controls are odd). A model must capture all hidden effects to successfully predict case/control in

this scenario. This is a test to identify the complexity limitations of each of the methods.

Simulation model 3 could reflect a disease with incomplete penetrance. The probabilistic manner of this simulation means that the optimum binary classifier will have an expected accuracy limited by the role of stochasticity in the model. The amount of stochasticity is a function of the number of hidden effects. In this model, case control odds were equal to the percentage of hidden input effects on. If there are 4 hidden input effects and 2 are on, the patient has a 50% chance of being a case and a 50% chance of being a control. A binary classifier can perfectly model this simulation and still have error due to the probabilistic nature. The maximum expected accuracy was calculated from a binomial distribution multiplied by the minority percentage as the best a binary classifier could do is choose the more likely class. For example, in the case of 4 hidden effects the maximum expected accuracy is 68.75%.

An example of a hypothetical condition a hidden input effect could represent is the familial hypercholesterolemia genotype. For a patient with the familial hypercholesterolemia genotype, the simulated clinical observations could represent increases in levels of total and low-density lipoprotein cholesterol, the deposition of cholesterol in extravascular tissues, corneal arcus and elevated triglyceride levels *(57)*. Some factors such as elevated triglyceride levels are not solely the result of the genetic predisposition and are related to environmental factors. Hypothetically additional hidden input effects on the same observed variable would represent these other factors. Because our goal is to evaluate methods for their ability to broadly capture these types of patterns,

we generate randomized relationships between hidden and observed variables. This avoids overfitting our evaluation to specific phenotypes.

2.3.4. *Supervised Classification Comparison*

If successfully trained, the hidden layer of a DA, *y*, captures the first *n* factors of variation in the data, where *n* is the number of nodes in the hidden layer. To test whether the DA constructed useful features by learning the main factors of variation in the data we used the trained hidden layer as an input to a shallow classifier.

To do this, we first completed unsupervised pre-training of the DA with all of the simulated samples. The hidden layer values, *y*, were calculated for all samples using the trained DA without any corruption and fed in as the features to a random forest to form a supervised classifier.

Classification performance between DAs plus random forests (DA+RF) were compared against decision trees, random forests, nearest neighbors and support vector machines in a parameter sweep under each model (Table 2.1). Additional model parameters included in sweeps are included in the supplementary materials. All traditional classifiers were implemented with Scikit-learn *(56)*. Classification performance was compared using the AUROC with 10-fold cross validation across 10 independent replicates for each set of parameters.

PARAMETER	VALUES
OBSERVED VARIABLES	50, 100, 200, 400
EFFECT MAGNITUDE (X VARIANCE)	1, 2, 4
HIDDEN INPUT EFFECTS	1, 2, 4, 8, 16
EFFECTED OBSERVED VARIABLES PER	5, 10
UNLABELED PATIENTS	10,000

LABELED PATIENTS	100, 200, 500, 1,000,
SYSTEMATIC BIAS	0.1 applied to 0.33 of
DA HIDDEN NODES	1, 2, 4, 8

Table 2.1 : Simulation Model 1 Parameter Sweep Specifications.

2.3.5. *Semi-Supervised Classification Comparison*

The supervised classification comparison was repeated but with additional patients simulated and utilized during the unsupervised pre-training of the DA. The additional patients were simulated at the same 50% case, 50% control ratio but their labels were discarded after simulation. These additional patients were mixed with the original labeled patients and included in the unsupervised pre-training of the DA. The unlabeled samples were then discarded and the DA+RF was then provided the same, labeled, patient groups as the traditional classifiers. The labeled patient samples were run through the trained DA in the same manner as the unsupervised pre-training but without any corruption added to the data. The DA+RF and traditional classifiers were evaluated in a parameter sweep under each model using 10-fold cross validation.

2.3.6. *Missing Data Comparison*

The semi-supervised classification comparison was repeated five times with, 0%, 10%, 20%, 30% and 40% of the data missing. Missing data was added at random per sample, depending on the specified percentage missing.

Throughout these trials, the cost calculation was modified to exclude missing data from the cost and allow the DA to learn without imputing values (Formula 4). The traditional classifiers were trained using mean imputation for missing data. Mean

imputation is particularly well suited for the simulation models because the observations were drawn from normal distributions, potentially giving an advantage to the non-DA algorithms that would not be available in many real datasets.

As in the semi-supervised classification comparison trial, the DA+RF and traditional classifiers were evaluated under each model using 10-fold cross validation.

2.3.7. *Clustering and Visualization*

To interpret and visualize results, patient populations were clustered using principal components analysis (PCA) and t-Stochastic Neighbor Embedding (t-SNE) of the trained DA's hidden nodes *(58, 59)*. PCA and t-SNE were implemented with the Sci-kit learn library *(56)*.

Ten thousand patients (5,000 cases, 5,000 controls) with four hidden effects were simulated under model 1. PCA followed by t-SNE was performed initially on the raw input for comparison and then on the hidden nodes of the DA after every 10 training epochs.

To test the ability to identify subtypes, we simulated 15,000 patients, 5,000 cases under model 1, 5,000 cases under model 2, and 5,000 controls. Input observations were compared to two, three and four-node DAs using PCA followed by t-SNE.

2.3.8. *ALS Survival Analysis*

Data used in the ALS Survival portion of this article were obtained from the Pooled Resource Open-Access ALS Clinical Trials (PRO-ACT) Database. In 2011, Prize4Life, in collaboration with the Northeast ALS Consortium, and with funding from the ALS Therapy Alliance, formed the Pooled Resource Open-Access ALS Clinical Trials (PRO-ACT) Consortium. The data available in the PRO-ACT Database has been volunteered

by PRO-ACT Consortium members. The PRO-ACT dataset includes 23 clinical trials covering 10,723 patients. We limit our survival analysis to the 3,398 patients with known death information, but perform unsupervised pre-training of the DA with all 10,723 patients.

Patient data includes quantitative features consisting of demographic information, diagnosis history, family history, treatment history, vital sign readings, concomitant medications and laboratory tests. Categorical variables were converted to one-hot encoding. Repeated or temporal measurements were encoded as the mean, minimum, maximum, count, standard deviation and slope across each repeat. Measurement scales were standardized and input features were normalized to be between 0 and 1. No imputation was performed on the input to the DA; K-nearest neighbors imputation (K=15) was performed for the raw comparison. This preprocessing resulted in an input layer of 6,812 numerical features per patient.

Patient survival was predicted as the number of days from disease onset. Random Forest Regression using Scikit-learn with 1,000 estimators was performed on the raw data and the hidden layer of a 250 node DA trained for 1,000 epochs. Performance was evaluated using 10-fold cross validation. Cluster analysis using t-SNE was compared between PCA (2, 4, 8, and 16 components) on the raw input with the hidden layer of the DA.

2.4. *Results*

2.4.1. *Case-Control DA Training Visualization*

We trained a DA and visualized the training process using PCA and t-SNE. These visualization techniques offer intuition and the ability to examine the sub-clusters. Given

5,000 cases and 5,000 controls under simulation model 1, PCA and t-SNE alone did not yield defined clusters (Figure 2.2 A). Figures 2.2 B-F show the separation of cases from controls as the DA is trained. One thousand epochs of training via stochastic gradient descent were found to be sufficient for the convergence of reconstruction cost and stabilization of visualizations within simulated data (Figure 2.2 E and F, Supplemental Figure 1).

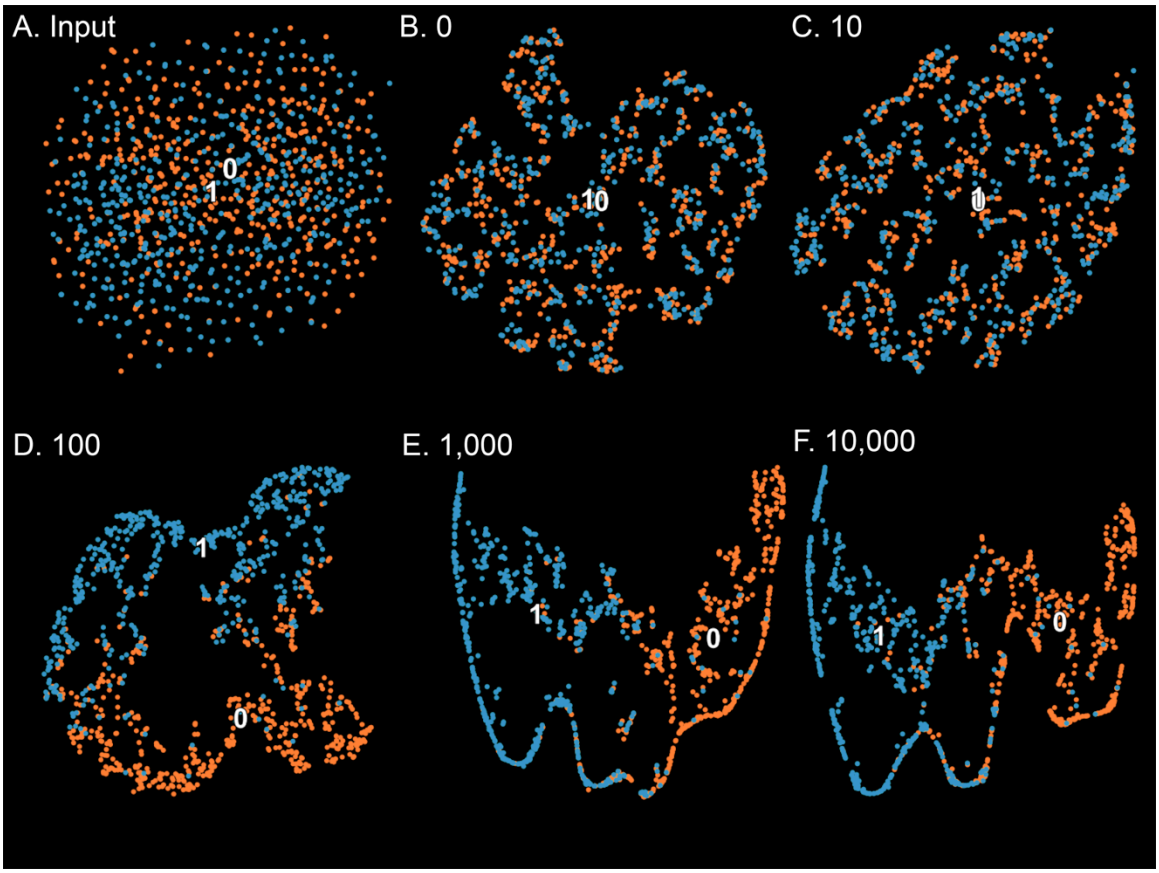


Figure 2.2: Visualization of DA training over time.

Case vs. Control clustering via principal components analysis and t-distributed stochastic neighbor embedding throughout the training of the DA. Controls are shown in yellow, cases are shown in red. **A.)** Raw input **B.)** 0 training epochs **C.)** 10 training epochs **D.)** 100 training epochs **E.)** 1,000 training epochs **F.)** 10,000 training epochs.

2.4.2. *Fully Supervised Comparison*

To examine the ability of DAs to learn the structure of the data we compare the predictive ability of classification algorithms applied to the DA constructed through

unsupervised training. Random forests demonstrated a strong balance of performance and stability, and were used for all comparisons. We then compare the DA plus a random forest classifier (DA+RF) to the top performing classifiers on raw input data (Table 2.2).

PATIENTS	DA+RF	RANDOM FOREST	SUPPORT VECTOR MACHINE	DECISION TREE	NEAREST NEIGHBORS
100	0.618	0.653	0.504	0.599	0.635
200	0.637	0.610	0.449	0.589	0.608
500	0.677	0.690	0.663	0.617	0.642
1000	0.774	0.717	0.776	0.634	0.651
2000	0.755	0.736	0.862	0.643	0.658
MEAN	0.692	0.681	0.651	0.616	0.639

Table 2.2: Mean Receiver Operating Curve Area Under Curve by method under simulation model 1. (10 Replicate, 10-fold cross validation)

Key trends emerged under each model; with few patients SVMs had AUCs indistinguishable from those expected from a random classifier. As one would expect, SVMs were top performers at when the number of patients was high. Random forest classification performance scaled steadily with patient count. The DA+RF performed similarly to the random forest, showing that a 2-node DA is able to capture at least one of the input hidden effects. Capturing any signal is sufficient to accurately classify simulation model 1.

2.4.3. *Semi-Supervised Comparison*

The full potential of the DA+RF is reflected in semi-supervised parameter sweep comparison for simulation model 1 (Figure 2.3 A). Each set of parameters was evaluated with 10 replicates and 10-fold cross validation for each replicate. With sufficient unlabeled examples, the DA method's performance is high, even with very few labeled

examples. Because of the extreme feature reduction, the traditional classifier on top of the DA is able to reach its learning capacity with very few labeled patients (Figure 2.3 A).

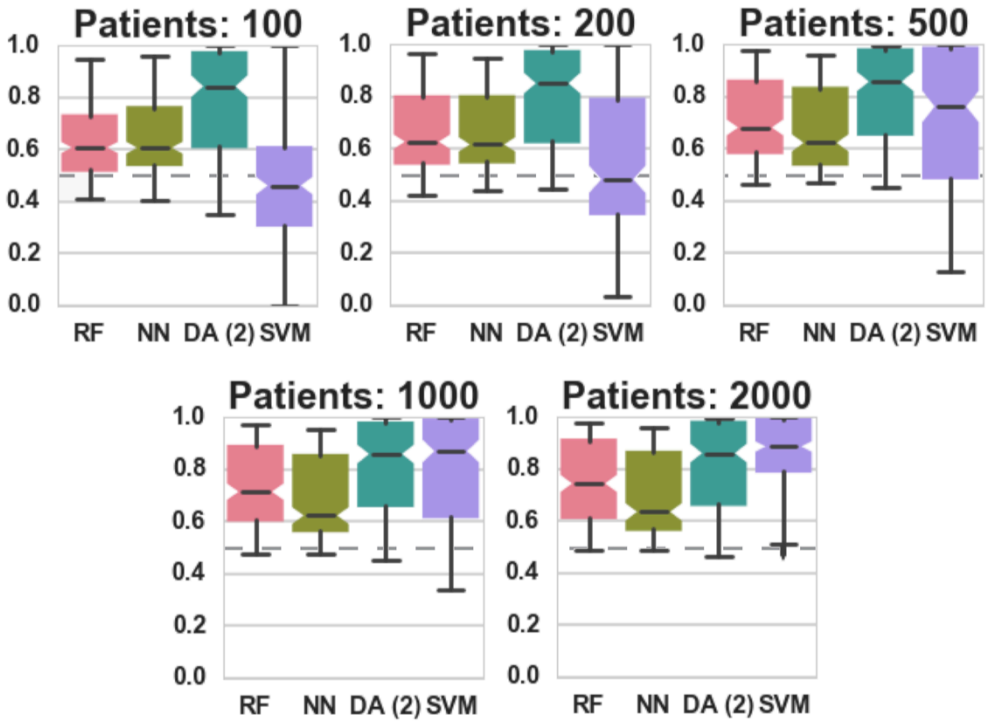


Figure 2.3: Classification AUC in relation to the number of labeled patients under simulation model 1.

(RF – Random Forest, NN – Nearest Neighbors, DA – 2-node DA + Random Forest, SVM – Support Vector Machine). Unsupervised pre-training of the 2-node DA was performed with 10,000 patients. Notch indicates 95% confidence interval for the median. Whiskers extend 1.5 times past the low and high quartiles. Points outside this range are denoted as dots and represent outliers. Gray dashed line indicates random choice expected performance (0.5).

Efficient learning from labeled examples is critical in practical use cases because there are often few well-annotated cases due to the expense of clinician manual review. The 2-node DA plus random forest also showed strong performance in relation to an SVM when there were many observed clinical variables (Supp. Fig 3) and when there were many hidden effects. The SVM again showed the highest performance at very high numbers (1000 or more) of labeled patients. The advantages of semi-supervised learning diminish as the number of labeled patients gets closer to the number of total patients. In

addition, at high patient counts, a DA with more than 2 hidden nodes is required to capture the structure of the data with higher resolution.

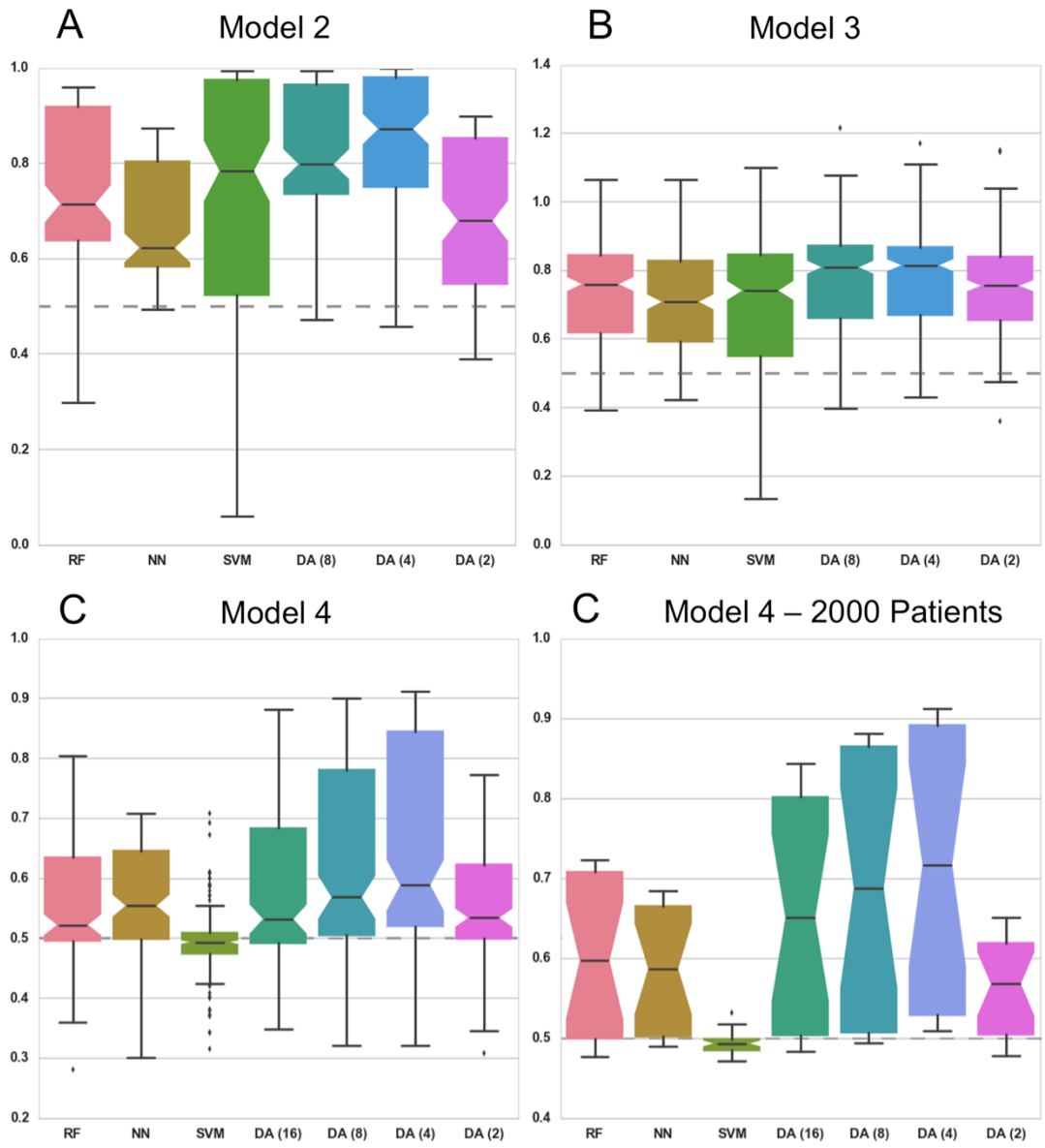


Figure 2.4: Classification accuracy comparisons for models 2-4.

Notch indicates 95% confidence interval for the median. Whiskers extend 1.5 times past the low and high quartiles. Gray dashed line indicates random choice expected performance (50%). **A)** Classification Accuracy of model 2 (1, 2, 4 and 8 effects). **B)** Classification AUC normalized to simulation model 2 expected max predictive accuracy (1, 2, 4 and 8 effects). **C)** Classification AUC of model 4 (1, 2, 4 and 8 effects). **D.)** Classification AUC of model 4 (parameter sweep results for 1, 2, 4 and 8 effects using only the parameter sets with 2,000 labeled patients).

These patterns repeat across the other simulation models, with more complex models requiring more hidden nodes to adequately model the structure of the data. In simulation model 2 (Figure 2.4 A), both 4 and 8 node DAs outperform the 2-node DA. Under Model 3, the 4-node DA is the strongest performing, with median performance 5% better than the next best traditional classifier. The 4-node DA's median 95% confidence interval was above any of the compared methods. Model 4 (Figure 2.4 C-D) was the most difficult to classify as the classifier had to capture all of the hidden effects to be accurate. In several cases, no classifier did better than the expected performance of a random classifier. In fact, the SVM's average AUC over the entire sweep was statistically indistinguishable from random performance. As expected, the 2-node performs worse than the 4 and 8-node DAs on model 4. The 2-node DA lacks sufficient dimensionality to capture more than 4 hidden input effects.

Clinical records often have empty fields, so algorithms must be robust to missing data. We evaluated the DA's robustness in this situation. The DA is robust to missing data maintaining near-max classification performance across the missingness proportions tested (Supp. Fig 4). For these simulation models, the mean imputation used for non-DA approaches is an ideal strategy. DAs and SVMs show consistent performance even as the percent of data missing increases, suggesting that the DA is at least as robust as the ideal imputation method.

2.4.4. *Simulated Subtype Clustering Visualization*

We evaluated the DAPS' ability to cluster patients for subtype identification. To perform this analysis, we simulated 5,000 cases from each of two different models (1 and 2) to represent a disease with two subtypes. An additional 5,000 controls were simulated.

We then visualized the DA constructed from this set of patients using PCA followed by t-SNE. In the input data, the subtypes are relatively overlapping (Figure 2.5 A). A DA with two nodes was also unable to separate this number of subtypes (Figure 2.5 B). Visualizations constructed from DAs with three (Figure 2.5 C) or four (Figure 2.5 D) nodes were able to effectively separate both subtypes of cases from each other and from controls.

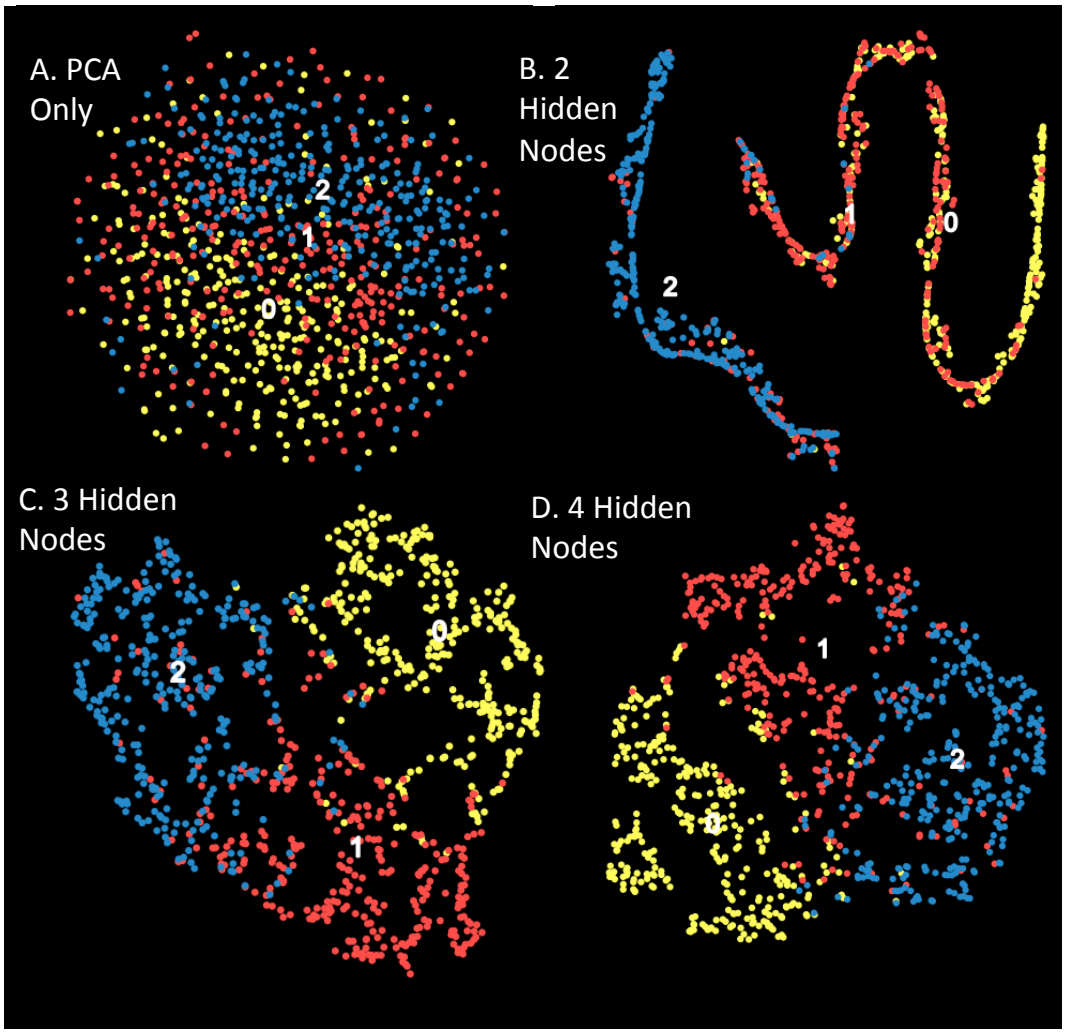


Figure 2.5: Case vs. Control clustering of 2 disease states and 1 healthy state.

Clustering via principal components analysis and t-distributed stochastic neighbor embedding after training the DA for controls and cases generated from a combination of models 1 and 2. Controls are shown in yellow, subtype 1 (model 1) is shown in red, subtype 2 (model 2) is shown

in blue. **A)**. Raw input. **B.)** 1,000 training epochs with 2 hidden nodes. **C.)** 1,000 training epochs with 3 hidden nodes. **D.)** 1,000 training epochs with 4 hidden nodes.

2.4.5. *ALS Survival Analysis*

We evaluated the DAPS' ability to quantitatively predict ALS patient survival (Figure 6) with ten-fold cross validation. Both models used a random forest regressor to predict survival to compare predictions between the raw imputed data and the hidden layer of a 250 node DA.

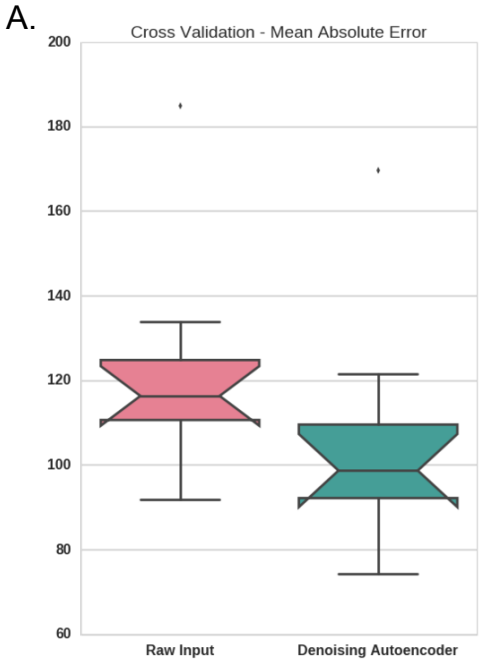


Figure 2.6: Prediction comparison of ALS survival between DA constructed and raw features. **A.)** Ten-fold cross validation survival prediction quantification. Mean absolute error in days. Notch indicates 95% confidence interval for the median. Whiskers extend 1.5 times past the low and high quartiles. Points outside this range are denoted as dots and represent outliers.

In order to visualize the basis for improved prediction, we performed t-SNE clustering to compare PCA with the hidden layer of the DA (Figure 7 A-B, Supp. Fig. 7). The visualization constructed from the hidden layer of the DA shows space defined by

the DA separated several clear clusters with low patient survival as well as a more heterogeneous cluster with longer survival. PCA applied to the raw input produced some patterns but did not produce any clear clustering by survival.

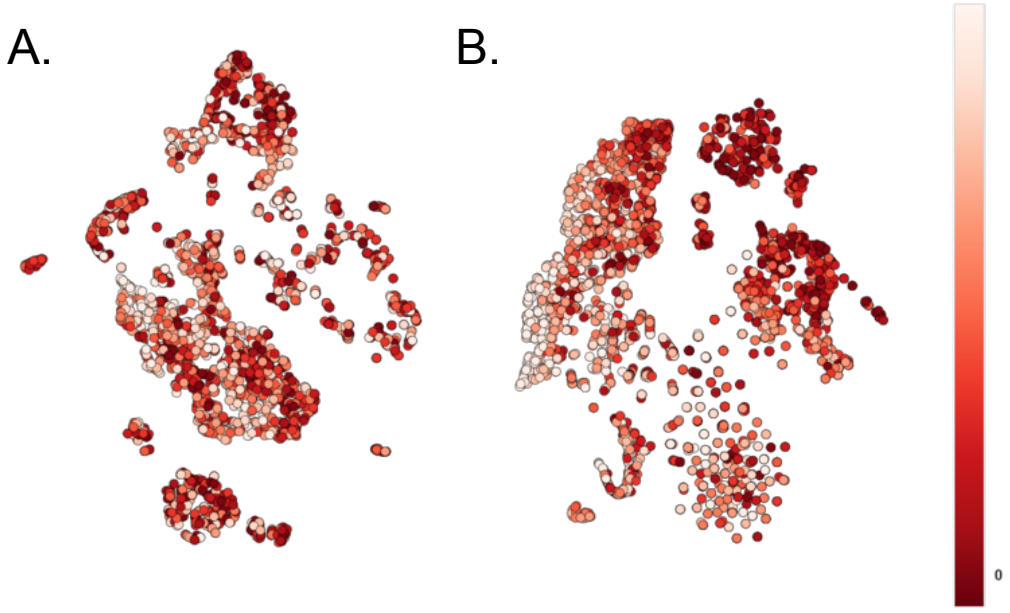


Figure 2.7: ALS Survival Visualization (Raw vs. Constructed Features).
A.) PCA (2 components) followed by t-SNE. **B.)** t-SNE of the DA (250 nodes) hidden layer. All cluster coloring was determined by rank of days survived. Light colors indicate longer survival.

2.5. *Summary and Future Directions.*

We presented a semi-supervised learning approach using DAs to model patients in the EHR. The benefits of the method presented in this work are; 1) It is relatively inexpensive to perform analysis on large amounts of unlabeled EHR data; 2) It is expensive to have data labeled at a research quality by a clinician; 3) The DA+RF has strong performance when there are many unlabeled samples and few labeled samples.

Competitive supervised classification accuracy with a large degree of feature reduction indicates the DA successfully learned the structure of the high-dimensional

EHR data. DAs are particularly well suited to the EHR because their unsupervised nature allows the formation of a semi-supervised classifier and the ability to utilize large un-annotated patient populations to improve classification accuracy. The dimensionality reduction of DAs allows clustering of the reduced feature set for the visualization and determination of subtypes. These clusters may reveal disease subtypes, fine-tuned targets for genotype-phenotype association. The DA models are easily de-constructible because they use a simple model for the traditional classifier with transparent node compositions that can be traced back to inputs. In addition, our method proposes a straightforward modification to the DA to enable it to process missing data without imputation.

PheWASs are a powerful tool to leverage the vast clinical data contained in the electronic health record but currently suffer from the reliance on billing codes or manual clinician annotation. Denny et al. *(25)* call out the need for increased accuracy in phenotype definition in the original PheWAS publication, particularly for rare phenotypes or phenotypes that do not directly correspond with a billing code. In addition, several studies have found increased genetic linkage via subtyping *(9, 12, 13, 36–38)*. Li et al. *(6)* presented a powerful example of EHR subtyping of patients with type 2 diabetes using a similar methodology, but they utilized Ayasdi, a commercial, closed source topology data analysis software tool. Our method is built on free, open source libraries that will continue to be improved and our software is accessible for the research community.

DA nodes and clusters of nodes provide composite variables that may better approximate and represent the condition of the subject. These additional phenotype targets may provide more homogeneous targets for genotype associations. Beyond

genotype to phenotype association, these visualizations may also help clinicians to understand the level of heterogeneity for a specific disease and to make treatment associations among sub-clusters of patients. While further work is required to analyze the makeup and meaning of the ALS survival clusters recognized by DAPS, they suggest a helpful starting point for investigation.

Our work provides an important contribution but additional analysis and challenges remain. The transition from simulated data and relatively homogenous clinical trial data to diverse multi-disease real world clinical data will likely require additional steps. In addition, in our simulations we assume a preprocessing step has already been performed to handle the compound structure present in the EHR. This step is necessary to transform categorical, free text, images and temporal data to suitable input for the DA. The PRO-ACT ALS clinical trial data does not currently include any free text or images. Raw EHR data will not be as complete or clean as either clinical trial or simulated data. Despite these challenges, autoencoders have been shown to effectively denoise data *(60, 61)*, and may also be well suited to noisy EHR-derived data.

Future work will focus on developing tools to examine and interpret constructed phenotypes (hidden nodes) and clusters. In addition, we will develop a framework for evaluating the significance of constructed clusters for genotype to phenotype association. We anticipate high weights indicate important contributors to node construction revealing relevant combinations of input features. Finally, we will construct a scheme for determining optimal hyper parameter (i.e. hidden node count) selection.

2.6.*Acknowledgements*

This work is supported by the Commonwealth Universal Research Enhancement (CURE) Program grant from the Pennsylvania Department of Health as well as the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4552 to C.S.G. The authors would like to thank Dr. Jason H. Moore for helpful discussions. The authors acknowledge the support of the NVIDIA Corporation for the donation of a TitanX GPU used for this research.

Chapter 3. Characterizing and managing missing data in the Electronic Health Record.

This chapter was adapted from:

Beaulieu-Jones, Brett K., Daniel R. Lavage, John W. Snyder, Jason H. Moore, Sarah A Pendergrass, Christopher R. Bauer. "Characterizing and Managing Missing Structured Data in Electronic Health Records." In Review. Preprint doi: 10.1101/167858

B.K.B.-J., J.H.M., S.A.P. and C.R.B. and conceived of the study. D.L. and J.S. performed data processing. B.K.B.-J. and C.R.B. performed analyses. B.K.B.-J., S.A.P. and C.R.B. wrote the manuscript and all authors revised and approved the final manuscript. Supplemental figures are available at: doi.org/10.1101/167858

And:

Beaulieu-Jones, Brett K., Jason H. Moore, and the Pooled Resource Open-Access ALS Clinical Trials Consortium. "Missing Data imputation in the Electronic Health Record Using Deeply Learned Autoencoders." *Pac Symp Biocomput*. 2016 ; 22: 207–218.

B.K.B.-J. and J.H.M. conceived of the study. B.K.B.-J. performed the analysis and wrote the manuscript. B.K.B.-J. and J.H.M. revised and approved the final manuscript.

Data used in the preparation of this article were obtained from the Pooled Resource Open-Access ALS Clinical Trials (PRO-ACT) Database. As such, the following organizations and individuals within the PRO-ACT Consortium contributed to the design and implementation of the PRO-ACT Database and/or provided data, but did not participate in the analysis of the data or the writing of this report: Neurological

Clinical Research Institute, MGH, Northeast ALS Consortium, Novartis, Prize4Life, Regeneron Pharmaceuticals, Sanofi, Teva Pharmaceutical Industries, Ltd.

3.1. *Motivation and Introduction*

Missing data present a challenge to researchers in many fields and this challenge is growing as datasets increase in size and scope. This is especially problematic for electronic health records (EHRs), where missing values frequently outnumber observed values, and the absence of an observation can be caused by a variety of mechanisms that may or may not be informative. EHRs were designed to record and improve patient care and streamline billing, and not as resources for research*(28)* thus there are significant challenges using these data to gain a better understanding of human health. As EHR data become an increasingly utilized as a source of phenotypic information for biomedical research*(62)* it is crucial to develop strategies for coping with missing data.

Clinical laboratory assays are a particularly rich data source within the EHR, but they also tend to have large amounts missing data. These data may be missing for many different reasons. Some tests are used for routine screening but screening may be biased. Other tests are only conducted if they are clinically relevant to very specific ailments. Patients may also receive care at multiple healthcare systems, resulting in information gaps at each institution. Age, sex, socioeconomic status, access to care, and medical conditions can all impact how comprehensive the data is for a given patient.

Aside from the uncertainty associated with a variable that is not observed, many analytical methods, such as regression or principal components analysis are designed to operate on a complete dataset. The easiest way to implement these procedures is to remove variables with missing values or remove individuals with missing values. Eliminating variables is justifiable in many situations, especially if a given variable has a

large proportion of missing values, but doing so may restrict the scope and power of a study. Removing individuals with missing data is another option known as complete case analysis. This is generally not recommended unless the fraction of individuals that will be removed is small enough to be considered trivial, or there is good reason to believe that the absence of a value is due to random chance. If there are any differences between individuals with and without observations, complete case analysis will be biased. For example, if only patients with severe symptoms receive a certain test, removing patients with missing values is equivalent to removing the healthy patients.

An alternative is to fill the missing fields with estimates. This process, called imputation, requires a model that generally makes assumptions about why only some values are observed. These missingness mechanisms fall somewhere in a spectrum between three scenarios (Figure 3.1). When data is missing in a manner completely unrelated to both the observed and unobserved values, it is considered to be missing completely at random (MCAR) *(63, 64)*. When data are MCAR, the observed data represent a random, unbiased sample of the population, but this is rarely encountered in practice. Conversely, data missing not at random (MNAR) refers to a situation where the probability of observing a data point depends on the value of that data point *(65)*. In this case, the mechanism responsible for the missing data is biased and should not be considered ignorable *(66)*. For example, rheumatoid factor is an antibody detectable in blood, and the concentration of this antibody is correlated with the presence and severity of rheumatoid arthritis. This test is typically performed only on patients with some indication of rheumatoid arthritis. Thus, patients with high rheumatoid factor levels are more likely to have rheumatoid factor measures.

A more complicated scenario can arise when multiple variables are available. If the probability of observing a data point does not depend on the value of that data point, after conditioning on one or more additional variables, then that data is said to be missing at random (MAR) *(65)*. For example, a variable, *X,* may be MNAR if considered in isolation. However, if we observe another variable, *Y*, that explains some of the variation in *X* such that after controlling for the effect of *Y*, the probability of observing *X* is no longer related to its value, *X* is said to be MAR. In this way, *Y* can transform *X* from MNAR to MAR. There is no way to prove that *X* is randomly sampled after conditioning on covariates unless we measure some of the unobserved values, but strong correlations, ability to explain missingness, and domain knowledge may provide evidence of that the data are MAR.

Imputation methods assume specific mechanisms of missingness and assumption violations can lead to bias in the results of downstream analyses that can be difficult to predict *(67, 68)*. Variances of imputed values are often underestimated causing artificially low *p*-values *(69)*. Additionally, for data MNAR, the observed values have a different distribution than the missing values. To cope with this, a model can be specified to represent the missing data mechanism, but these models can be difficult to evaluate and may have a large impact on results. Great caution should be taken when handling missing data, particularly data that are MNAR. Most imputation methods assume that data are MAR or MCAR, but it is worth reiterating that these are all idealized states and real data invariably fall somewhere in between (Figure 3.1).

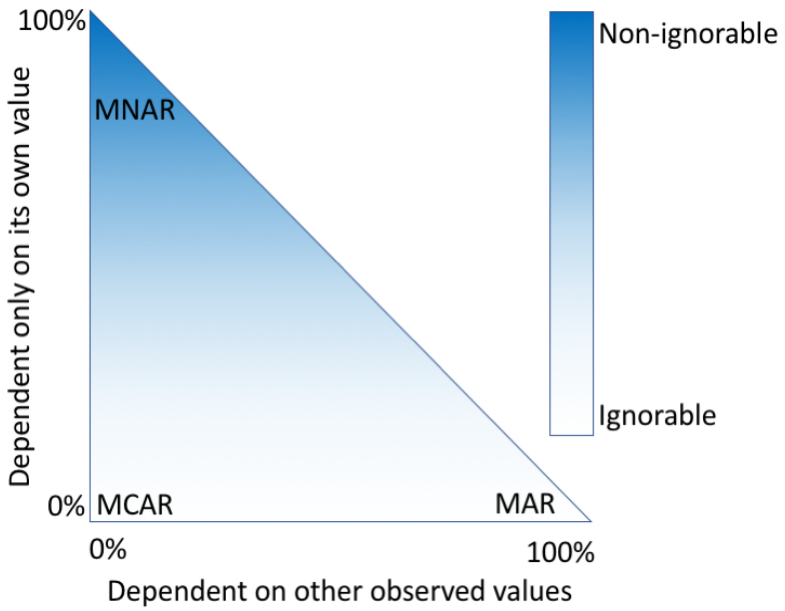


Figure 3.1: Mechanisms of missing data.

Two general paradigms are commonly used to describe missing data. Missing data are considered ignorable if the probability of observing a variable has no relation to the value of the observed variable, and are considered non-ignorable otherwise. The second paradigm divides missingness into three categories: missing completely at random (MCAR: the probability of observing a variable is not dependent on its value or other observed values), missing at random (MAR: the probability of observing a variable is not dependent on its own value after conditioning on other observed variables), and missing not at random (MNAR: the probability of observing a variable is dependent on its value, even after conditioning on other observed variables). The X-axis indicates the extent to which a given value being observed depends upon other the values of other observed variables. The Y-axis indicates the extent to which a given value being observed depends upon its own value.

3.2. *Characterizing and Managing Missing Structured Data in Electronic Health Records.*

3.2.1. *Abstract*

Missing data is a challenge for all studies; however, this is especially true for electronic health record (EHR) based analyses. Failure to appropriately consider missing data can lead to biased results. Here, we provide detailed procedures for when and how to

conduct imputation of EHR data. We demonstrate how the mechanism of missingness can be assessed, evaluate the performance of a variety of imputation methods, and describe some of the most frequent problems that can be encountered. We analyzed clinical lab measures from 602,366 patients in the Geisinger Health System EHR. Using these data, we constructed a representative set of complete cases and assessed the performance of 12 different imputation methods for missing data that was simulated based on 4 mechanisms of missingness. Our results show that several methods including variations of Multivariate Imputation by Chained Equations (MICE) and softImpute consistently imputed missing values with low error; however, only a subset of the MICE methods were suitable for multiple imputation. The analyses described provide an outline of considerations for dealing with missing EHR data, steps that researchers can perform to characterize missingness within their own data, and an evaluation of methods that can be applied to impute clinical data. While the performance of methods may vary between datasets, the process we describe can be generalized to the majority of structured data types that exist in EHRs and all of our methods and code are publicly available.

3.2.2. *Materials and Methods*

3.2.2.1. Source Code

Source code to reproduce the analyses in this work are provided in our repository (https://github.com/EpistasisLab/imputation) under a permissive open source license. In addition, Continuous Analysis*(70)* was used to generate Docker images matching the environment of the original analysis, and to create intermediate results and logs. These artifacts are freely available (https://hub.docker.com/r/brettbj/ehr-imputation/ and archive version 10.6084/m9.figshare.5165653).

3.2.2.2. EHR data processing

All clinical laboratory assays were mapped to LOINC (Logical Observation Identifiers Names and Codes). We restricted our analysis only to outpatient lab results to minimize the effects of extreme results from inpatient and emergency department data. We used all laboratory results between August 8th, 1996 and March 3rd, 2016. We excluded any codes for which less than 0.5% of patients had a result. The resulting dataset consisted of 669,212 individuals and 143 laboratory assays.

We next removed any lab results that occurred prior to patient's 18th birthday or after their 90th. In cases where a date of death was present, we also removed any lab results that occurred within one year of death as we found that the frequency of observations often spiked during this period and the values for certain labs were altered for patients near death. For each patient's clinical lab measures, a median date of observation was then calculated based on all of their remaining lab results. We defined a temporal window of observation by removing any lab results that were recorded more than 5 years from the median date. We then calculated the median result of the remaining labs for each patient. Finally, we calculated the mean BMI of each patient, and dropped any patients whose sex was unknown. As each variable had a different scale and many deviated from normality, we applied Box-Cox and Z-transformations to all variables. The final dataset used for all downstream analyses contained 602,366 patients and 146 variables (age, sex, BMI, and 143 laboratory measures).

3.2.2.3. Variable selection

We first ranked the labs by total missingness. At each rank we calculated the percent of complete cases for the set including all lower ranked labs. We also built a random

forest classifier to predict the missingness of each variable. Based on these results, in conjunction with domain knowledge, we selected 28 variables that provided a reasonable trade-off between quantity and completeness and were deemed to be largely MAR.

3.2.2.4. Predicting the presence of data

For each clinical lab, we used the default scikit-learn *(71)* random forest classifier, to predict whether a specific value would be present or absent. Each lab measure was converted to a binary label vector based on whether the measure was recorded or not. The values of all other labs, excluding co-members of a panel, were used as the training matrix input to the random forest. This process was repeated for each lab test using 10-fold cross validation. Prediction accuracy was assessed by the area under the receiver operator characteristic (AUC ROC), and the results for each variable relate to the degree of MNAR, MAR, or MCAR present.

3.2.2.5. Sampling of complete cases

To generate a set of complete cases that resembled the whole population, we randomly sampled 100,000 patients without replacement. We then matched each of these individuals to the most similar patient who had a value for each of the 28 most common labs by matching sex and finding the minimal Euclidean distance of age and BMI.

3.2.2.6. Simulation of missing data

Within the sampled complete cases, we selected the data for removal by four mechanisms:

Simulation 1: Missing Completely at Random (MCAR)

We replaced values with NAN at random. This procedure was repeated 10 times each for 10%, 20%, 30%, 40%, and 50% missingness yielding 50 simulated datasets.

Simulation 2: Missing at Random (MAR)

We selected two columns (*A* and *B*). When column *A* had a value in the highest quartile of values for this lab, we replaced 50% of the values selected at random from column *B* with NAN. The procedure was repeated for each quartile and each lab combination yielding 3024 simulated datasets.

Simulation 3: Missing not at Random (MNAR)

We selected a column and a quartile. When the column's value was in the quartile we replaced it with NAN 50% of the time. This procedure was repeated for each of the 4 quartiles of each of the 28 labs generating a total 112 total simulations of missingness.

Simulation 4: Missingness based on real data observations

From our sampled complete cases dataset, we took each patient and matched them to the nearest neighbor, excluding self matches, in the entire set of observed data based on their sex, age, and BMI. We then replaced any lab value with NAN if it was absent for the matched patient in the original data.

3.2.2.7. Imputation of Missing Data

Using our simulated datasets (Simulations 1 - 4) we compared 18 common imputation methods (representative 12 methods are shown in primary figures) from the *fancyimpute (72)* and the Multivariate Imputation by Chained Equations (MICE) *(73)*

packages. A full list of imputation methods and the parameters used for each are in Supplementary Table 1.

3.2.3. *Results*

Our first step was to select a subset of the 143 lab measures for which imputation would be a reasonable approach. We began by ranking the clinical lab measures in descending order by the number of patients who had an observed value for that lab. At each rank, we plotted the percent of individuals missing a value for that lab as well as the percent of complete cases when that given lab was joined with all of the labs of lower rank. These plots showed that the best tradeoff between quantity of data and completeness would fall between 20 and 30 variables (Figure 3.2 A).

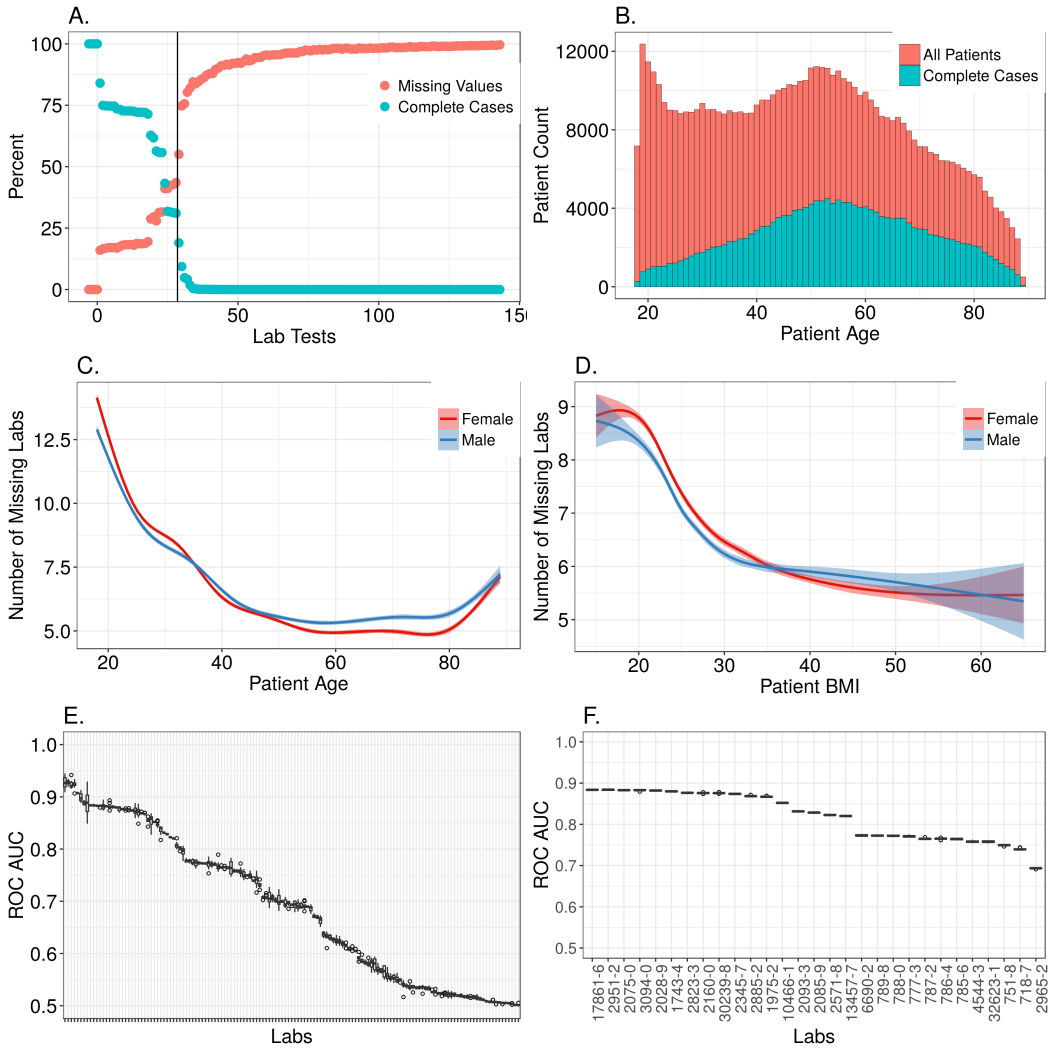


Figure 3.2: Summary of missing data for 143 clinical lab measures in the Geisinger Health System EHR.

A.) After ranking the clinical laboratory measures by the number of total results, the percent of patients missing a result for each lab was plotted (red points). At each rank, the percent of complete cases for all labs of equal or lower rank were also plotted (blue points). Only variables with a rank of 75 or less are shown. The vertical bar indicates the 28 labs that were selected for further analysis. **B.)** The full distribution of patient median ages is shown in blue and the fraction of individuals in each age group that had a complete set of observations for labs 1-28 are shown in red. **C.)** Within the 28 labs that were selected for imputation analyses, the mean number of missing labs is depicted as a function of age. **D.)** Within the 28 labs that were selected for imputation, the mean number of missing labs is depicted as a function of BMI. **E.)** Accuracy of a random forest predicting the presence or absence of all 143 laboratory tests. **F.)** Accuracy of a random forest predicting the presence or absence of the top 28 laboratory tests.

LOINC	Description
718-7	Hemoglobin [Mass/volume] in Blood
4544-3	Hematocrit [Volume Fraction] of Blood by Automated count
787-2	Erythrocyte mean corpuscular volume [Entitic volume] by Automated count
786-4	Erythrocyte mean corpuscular hemoglobin concentration [Mass/volume] by Automated count
785-6	Erythrocyte mean corpuscular hemoglobin [Entitic mass] by Automated count
6690-2	Leukocytes [#/volume] in Blood by Automated count
789-8	Erythrocytes [#/volume] in Blood by Automated count
788-0	Erythrocyte distribution width [Ratio] by Automated count
32623-1	Platelet mean volume [Entitic volume] in Blood by Automated count
777-3	Platelets [#/volume] in Blood by Automated count
2345-7	Glucose [Mass/volume] in Serum or Plasma
2160-0	Creatinine [Mass/volume] in Serum or Plasma
2823-3	Potassium [Moles/volume] in Serum or Plasma
3094-0	Urea nitrogen [Mass/volume] in Serum or Plasma
2951-2	Sodium [Moles/volume] in Serum or Plasma
2075-0	Chloride [Moles/volume] in Serum or Plasma
2028-9	Carbon dioxide, total [Moles/volume] in Serum or Plasma
17861-6	Calcium [Mass/volume] in Serum or Plasma
1743-4	Alanine aminotransferase [Enzymatic activity/volume] in Serum or Plasma by With P-5'-P
30239-8	Aspartate aminotransferase [Enzymatic activity/volume] in Serum or Plasma by With P-5'-P
1975-2	Bilirubin.total [Mass/volume] in Serum or Plasma
2885-2	Protein [Mass/volume] in Serum or Plasma
10466-1	Anion gap 3 in Serum or Plasma
751-8	Neutrophils [#/volume] in Blood by Automated count
2093-3	Cholesterol [Mass/volume] in Serum or Plasma
2571-8	Triglyceride [Mass/volume] in Serum or Plasma
2085-9	Cholesterol in HDL [Mass/volume] in Serum or Plasma
13457-7	Cholesterol in LDL [Mass/volume] in Serum or Plasma by calculation

Table 3.1: LOINC codes and descriptions of the most frequently ordered clinical laboratory measurements. The assays are ranked from the most common to the least.

As age, sex, and BMI have a considerable impact on what clinical lab measures are collected, we evaluated the relationship between missingness and these covariates (Figure 3.2 B-D). We also used a random forest approach to predicted the presence or absence of each measure based on the values of the other observed measures. MCAR data is not predictable, resulting in ROC AUC near 0.5. Only 38 of the 143 labs had ROC AUCs less than 0.55 (Figure 3.2 E). Very high ROC AUC are most consistent with data that are MAR. For the top 30 candidate labs based on the number of complete cases, the mean ROC AUC was 0.82. This suggested that the observed data could explain much of the mechanism responsible for the missing data within this set. We ultimately decided not to

include the 29th ranked lab, specific gravity of urine (2965-2), since it had a ROC AUC of only 0.69 and is typically used for screening only within urology or nephrology departments (R. Levy MD, personal communication). We did include the lipid measures (the 25th-28th ranked labs) since they had ROC AUC values near 0.82 and they are recommended for screening of all patients depending on age, sex, and BMI*(74)*. Our data confirm that age, sex, and BMI are all predictive of the presence of lipid measures (Supplementary Figure 1 A-B).

To assess the accuracy of imputation methods, we required known values to compare with imputed values. Thus, we generated a set of complete cases from the 28 variables we selected based on our characterization of data missingness. Since the set of complete cases differed from the broader population (Figure 3.2 B-D), we used sampling and K-nearest neighbors matching to generate a sample of the complete cases that resembled the entire population. We then simulated missing data within this set based on 4 mechanisms: MCAR, MAR, MNAR, and realistic patterns based on the original data.

We evaluated our ability to predict the presence of each value in the simulated datasets so that we could compare patterns with the real data. These simulations confirmed that our MCAR simulation had low ROC AUC (Figure 3.3 A). The MAR data (Figure 3.3 B) and MNAR data (Figure 3.3 C) were often well predicted, particularly for the MAR data, and when data were missing from the tails of distributions. The AUCs rarely exceeded 0.75 in the MNAR simulations while values above 0.75 were common in the MAR simulations. This provided additional support to our decision to include the top 28 lab measures, since they all have AUCs between 0.9 and 0.75, which was outside the range of nearly all MNAR simulations (Figure 3.2 F and Figure 3.3 C).

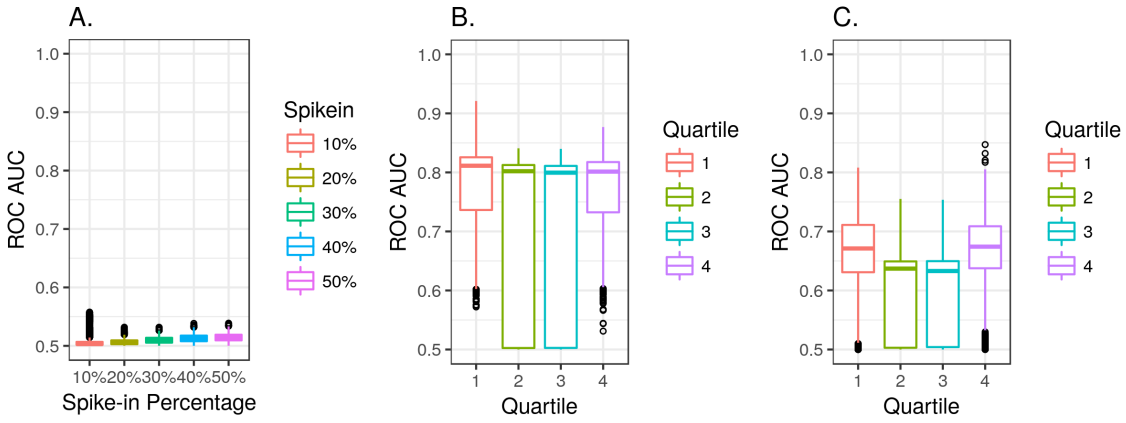


Figure 3.3: Ability to predict variable presence for each of the three types of missing data. **A.)** MCAR simulation **B.)** MAR simulation **C.**) MNAR simulation.

We chose to test the accuracy of several methods from two popular and freely available libraries: the Multivariate Imputation by Chained Equations (MICE) package for R and the *fancyimpute* library for python. We first applied each of these methods across simulations 1-3. For each combination, the overall root mean squared errors are depicted in Figure 3.4. A breakdown of all the methods and parameters are shown in Supplementary Table 1 with results in Supplementary Figures 3-21.

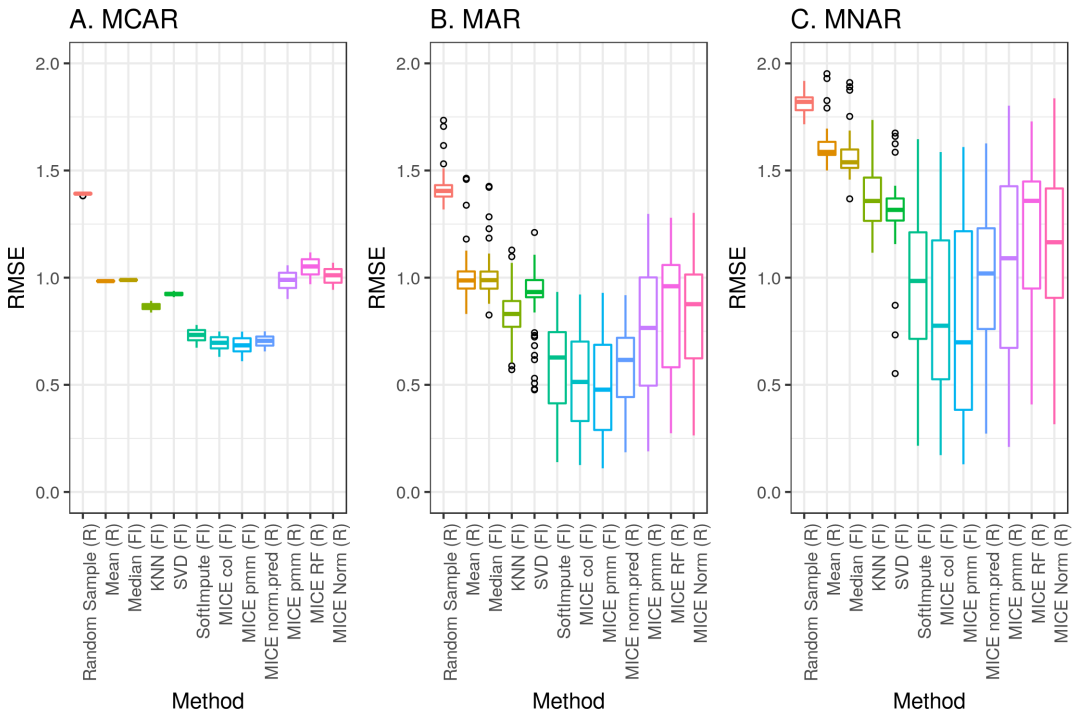


Figure 3.4: Imputation accuracy measured by RMSE across simulations 1-3.

Imputation accuracy measured by RMSE across simulations 1-3. **A.)** Missing Completely at Random (MCAR) **B.)** Missing at Random (MAR) **C.)** Missing Not at Random (MNAR)

We next measured imputation accuracy based on the patterns of missingness that were observed in the real data (Figure 3.5). The main difference compared to simulations 1-3 was lower error for some of the deterministic methods (Mean, Median, and KNN). It is worth mentioning that the error was highly dependent upon the variable that was being imputed. Specifically for the fancyimpute MICE PMM method, multicollinearity within some of the variables caused convergence failures that led to extremely large errors (Figure 3.5, method 8). These factors were relatively easy to address in the R package MICE-PMM method by adjusting the predictor matrix *(73)*.

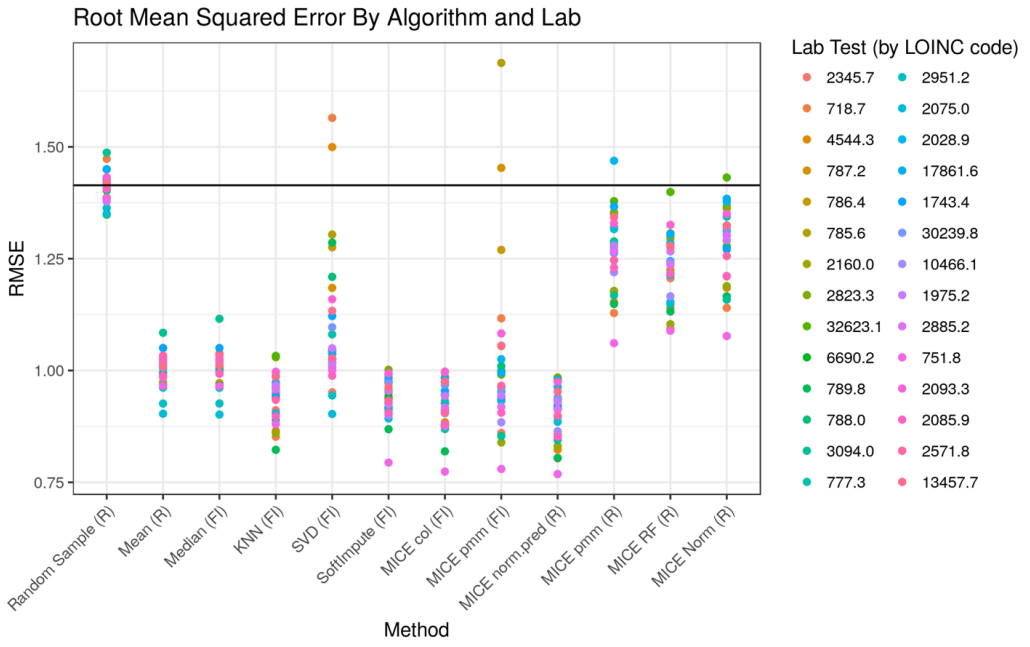


Figure 3.5: Imputation error (RMSE) for a subset of 10,000 patients from simulation 4. Twelve imputation methods were tested (X-axis) and colors indicate how the error varied between different lab measures (LOINC codes). The black line shows the theoretical error from random sampling.

In addition to evaluating the accuracy of imputation, it is also important to estimate the uncertainty associated imputation. One approach to address this is multiple imputation, where each data point is imputed multiple times using a nondeterministic method. This allows for the calculation of a confidence interval for any downstream result of interest. To determine if each method properly captured the true uncertainty of the data, we compared the error between an imputed dataset and the observed data with the error between two sets of imputed values for each method (Figure 3.6). If these errors are equal, then multiple imputation is likely producing good estimates of uncertainty. If, however, the error between two imputed datasets is less than that between each imputed dataset and the known values, then the imputation method is likely underestimating the variance.

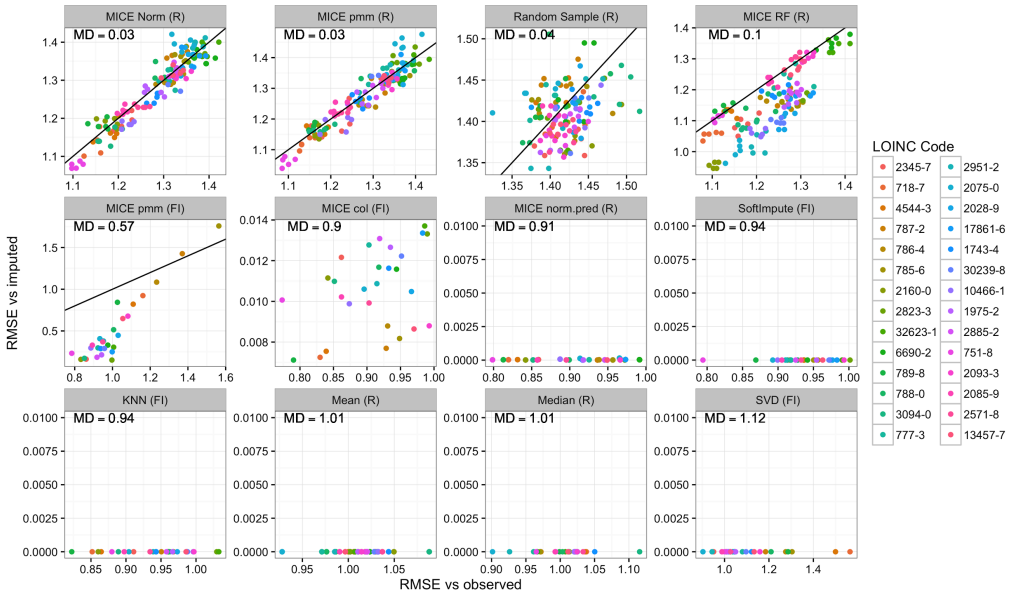


Figure 3.6: Assessment of multiple imputation for each method.

Using simulation 4, we imputed missing values multiple times with each method. The RMSEs between each imputed dataset and the observed values are shown on the X-axis and the RMSEs between two sets of imputed data are shown on the Y-axis. The axis scales vary between panels to better show the range of variation. Each lab test (LOINC) is indicated by the color of the points. The diagonal line represents unity. Panels are ordered by each method's mean deviation (MD) with unity, indicated in the top left corner of each panel.

Our results (Figure 3.6) demonstrate that many of the imputation methods are not suitable for multiple imputation. Three of the methods that had the lowest error in the idealized MCAR, MAR, and MNAR simulations (soft impute, MICE col (FI), MICE norm.pred (R)) were found to have minimal variation between imputations. This was also true of KNN, SVD, mean, and median imputation. Only three methods (random sampling, MICE norm (R), and MICE pmm (R)) seemed to have similar error between the multiple imputations and the observed data and thus appear to be unbiased. The latter two had very similar performance and are the best candidates for multiple imputation. Two methods had intermediate performance. The MICE RF (R) was similar to several other MICE methods in terms of error relative to the observed data but it produced

slightly less variation between each imputed dataset. This seemed to affect some variables more than other but there was no obvious pattern. The MICE pmm (FI) was not deterministic but it did seem to achieve low error at the expense of increased bias. In this case, the variables that could be imputed with the lowest error also seemed to have the most bias. Since this method claims to be a reimplementation of the MICE pmm (R) method, this may be due to multicolinearity among the variables that could not easily be accounted for as there was no simple way to alter the predictor matrix.

3.2.1. *Discussion*

It is not possible, or even desirable, to choose "the best" imputation method. There are many considerations that may not be generalizable between different sets of data; however, we can draw some general conclusions about how different methods compare in terms of error, bias, complexity, and difficulty of implementation. Based on our results, there seem to be three broad categories of methods.

The first category are the simple, deterministic, methods. These include mean or median imputation and K-nearest neighbors. The idea behind these methods is that the central tendency of a distribution will be good guess for any unobserved data point. Imputing mean or median values is very easy to implement this but may lead to severe bias and large errors if the unobserved data are more likely to come from the tails of the observed distribution (Figure 3.5 A-C, methods 2-5). It will also cause the variance of the distribution to be underestimated if more than a small fraction of the data are missing. Since these methods are deterministic, they are also not suitable for multiple imputation since no estimate of the uncertainty in the results of any downstream analyses can be made (Figure 3.6, bottom row).

KNN is similar to mean imputation but based on the idea that there may be groups of individuals that are similar to each other. The value of a missing data point can be estimated by identifying other individuals who have that measure and appear similar to the unmeasured individual based on values of variables observed in both. A group of similar individuals are thus identified and their values are averaged to provide an estimate of the missing value. This generally provides lower error than taking the mean of all individuals, but the choice of K can be difficult to specify. Our simulations suggest that the optimal value can range from less than 1% of the population to more than 50% the population depending on the mechanism of missingness.

KNN is a popular choice for imputation and has been shown to perform very well in some types of data *(75, 76)* but it was not particularly well suited for our data, regardless of the choice of K. This may due to issues of data dimensionality *(77)* or it may be that human beings do not fall into well separated groups based on their clinical lab results. This method is also not currently suitable for large datasets. The first step is to build a distance matrix for all pairs of individuals that is stored in RAM, and the size of the distance matrix scales with n^2.

There are also many different methods for calculating distance and the optimal choice may vary widely from one type of data to another. The method that we implemented uses the mean squared linear distance across all pairs of shared observations. While this is probably the best choice when the number of shared features varies between individuals, it assumes that all variables are equal in their ability to capture similarities between individuals regardless of what variable is being imputed. This is certainly not a realistic assumption for our dataset which further points to the fact

that imputation is not plug and play and analysis must be done before handling missing data.

The second broad category of algorithms could be called the sophisticated, deterministic methods. These include SVD, soft impute, and MICE col/norm.predict. They tend to rely on either multivariate regression and/or projection of the data into a space of lower dimension. SVD performed poorly compared to its counterparts and sometimes produced errors greater than simple random sampling (Figure 3.5, method 5). The reasons for this are not clear, but we cannot currently recommend this method. Soft impute and MICE col/norm.predict were among the lowest error methods in all of our simulations (Figure 3.5, methods 6-7). The main limitation of these methods are that they cannot be used for multiple imputation (Figure 3.6, middle row).

The third broad category of algorithms were the stochastic methods which included random sampling and most of the remaining methods in the MICE library. The random sampling method almost always produced the highest error (Figure 3.4-3.5, method 1) but it has the advantage of being easy to implement and it requires no parameter selection. The MICE methods based on predictive mean matching, random forests, and bayesian linear regression tended to perform similarly in terms of error in most of our simulations (Figure 3.4-3.5, methods 10-12).

Imputation methods that involve some type of stochastic sampling allow for a fundamentally different type of analysis called multiple imputation. In this paradigm, multiple imputed datasets (a minimum of 3 and often 10-20 depending of the percentage of missing data)*(78–80)* are generated and each is analyzed in the same way. At the end of all downstream analyses, the results are then compared. Typically, the ultimate result

of interest is supported by a p-value, a regression coefficient, an odds ratio, etc. In the case of a multiply imputed dataset, the researcher will have several test statistics that can be used to estimate a confidence interval for the result.

Multiple imputation has been gaining traction over the years and the MICE (multiple imputation by chained equations) package has become one of the most popular choices for implementing this procedure. This package is very powerful and very well documented*(73)* but like all methods for imputation, caution must be exercised. There also seems to be some confusion surrounding the concept of MICE. It is not a single algorithm but rather a framework for applying a variety of algorithms. Each missing value for a variable of interest is imputed by considering the other variables that were observed for that individual, the observed values of the variable of interest in other individuals, and/or the relationships between the variables. This procedure is applied for each missing value in one variable, and then to each subsequent variable. This entire process is then repeated for a number of iterations such that the values imputed in the first iteration can update the estimates for the second iteration. The result is a chain of imputed datasets and this entire process is typically performed in parallel so that multiple chains are generated.

In MICE, there are a number of choices that must be made and care should taken to evaluate the results. The first obvious choice is the method (i.e. equation). Many methods are available in the base package, additional methods can be added from other packages [https://cran.r-project.org/web/packages/miceadds/index.html], and users can even define their own. These methods could be extended in theory to include any of the previously described algorithms and the base package already includes random sampling

and mean imputation. We thoroughly evaluated three methods in the context of our dataset: predictive mean matching (pmm), bayesian linear regression (norm), and random forest (rf).

PMM is the default choice and popular since it can be used on a mixture of numeric and categorical variables. We found PMM to have a good trade-off between error and bias, but for our dataset it was critical to remove several variables from the predictor matrix due to strong correlations and multicolinearity. Bayesian regression performed similarly but was less sensitive to these issues. If a dataset contains only numeric values, bayesian regression may be a safer option. Random forest tended to produce results that were slightly biased for a subset of the variables without an appreciable reduction in error. Aside from random sampling, none of the other methods we evaluated were suitable for multiple imputation (Figure 3.6).

While the minimization of error in the imputed data is the primary goal, a singular focus on this objective is likely to lead to bias. For each missing value, it is also important to estimate the uncertainty associated with it. This can be achieved by multiple imputation using an algorithm that incorporates stochastic processes. Multiple imputation has become the field standard because it provide confidence intervals for the results of downstream analyses. One should not naively assume that any stochastic process is free of bias. It is important to check that multiple imputation is providing variability that corresponds to the actual uncertainty of the imputed values using a set of simulated data.

3.2.1. *Acknowledgments*

We thank Casey S. Greene (University of Pennsylvania) for his helpful discussions. This work was supported by the Commonwealth Universal Research Enhancement (CURE) Program grant from the Pennsylvania Department of Health. B.K.B.-J. and J.H.M. were also supported by by US National Institutes of Health grants AI116794 and LM010098 to J.H.M.

3.3. *Missing Data imputation in the Electronic Health Record Using Deeply Learned Autoencoders.*

3.3.1. *Abstract*

Electronic health records (EHRs) have become a vital source of patient outcome data but the widespread prevalence of missing data presents a major challenge. Different causes of missing data in the EHR data may introduce unintentional bias. Here, we compare the effectiveness of popular multiple imputation strategies with a deeply learned autoencoder using the Pooled Resource Open-Access ALS Clinical Trials Database (PRO-ACT). PRO-ACT is particularly interesting because it includes data across 23 different clinical trials. To evaluate performance, we examined imputation accuracy for known values simulated to be either missing completely at random or missing not at random. We also compared ALS disease progression prediction across different imputation models. Autoencoders showed strong performance for imputation accuracy and contributed to the strongest disease progression predictor. Finally, we show that despite clinical heterogeneity, ALS disease progression appears homogenous with time from onset being the most important predictor.

3.3.2. *ALS and the Pooled Resource Open-access Clinical Trials*

We evaluate each of the imputation methods on the ALS Pooled Resource Open-access Clinical Trials (PRO-ACT). Pooled clinical trial datasets present an ideal option for evaluating EHR imputation strategies because they include patients from differing environments with potential systematic biases. In addition, clinical trials represent the

gold standard for data collection making it possible to spike-in missing data while maintaining enough signal to evaluate imputation techniques.

Prize4Life and the Neurological Clinical Research Institute (NCRI) at Massachusetts General Hospital created the Pooled Resource Open-Access ALS Clincial Trials (PRO-ACT) platform with funding from the ALS Therapy Alliance in and in partnership with the Northeast ALS Consortium. The PRO-ACT project was designed to empower translational ALS research and includes data from 23 clinical trials and 10,723 patients. In this work, we use the subset of 1,824 patients included in the Prize4Life challenges*(81)*.

ALS is a progressive neurodegenerative disorder affecting both the upper and lower motor neurons causing muscle weakness, paralysis and leading to death*(81)*. ALS patients typically survive only 3 to 5 years from disease onset and show large degrees of clinical heterogeneity*(82–85)*.

A common measure used to monitor an ALS patient's condition is the ALS functional rating scale (ALSFRS)*(86, 87)*. The ALSFRS consists of 10 tests scored from 0-4 assessing patients' self-sufficiency in categories including: feeding, grooming, ambulation and communication. The change over time, or slope, is commonly used as a statistic to represent ALS progression.

3.3.3. *Methods*

We compare and evaluate a variety of methods to impute missing data in the EHR. We spiked-in missing data to the PRO-ACT dataset, and evaluated each approach's performance imputing known data. We also evaluated prediction accuracy using each of

the imputation methods on the ALSFRS. Each of these is described in detail below and all analysis was run using freely available open source library packages, DAPS*(88)*, FancyImpute*(72)*, Keras*(89)* and Scikit-learn*(56)*.

3.3.3.1. Data preparation and standardization

The PRO-ACT dataset includes patient demographic data, family history, concomitant medications, vital sign measurements, laboratory results, and patient history (disease onset etc.). PRO-ACT performed an initial data cleaning and quality assurance process. This process included extracting quantitative variables, merging laboratory tests with different names across trials, removable of indecipherable records and converting units. After processing the PRO-ACT dataset includes only quantitative values (continuous, binary, ordinal and categorical).

Our analysis encoded categorical variables using Sci-kit learn's OneHotEncoder*(56)*. Temporal or repeated measurements were encoded as the mean, minimum, maximum, count, standard deviation and slope across all measurements, creating 572 features for each samples. Additional measurements were standardized across scales (i.e. inches to cm). Non-numeric values in numeric measurements were coerced to numeric values. Where coercion failed they were replaced by NaN. Input features were normalized and scaled to be between 0 and 1, with missing features remaining as NaN.

3.3.4. *Imputation Strategies and Evaluations*

Autoencoders are a variation of artificial neural networks that learn a distributed representation of their input*(45)*. They learn parameters to transform the data to a hidden layer and then reconstruct the original input. By using a hidden layer smaller than the number of input features, or "bottle-neck" layer the autoencoder is forced to learn the

most important patterns in the data*(53)*. To prevent overreliance on specific features two techniques are commonly used. In a denoising autoencoder, noise is added to corrupt a portion of the inputs*(44, 53)*. Alternatively, a technique called dropout in which random units and connections are removed from the network forcing it to learn generalizations*(90)*. Autoencoders were shown to generate useful higher representations in both simulated and real EHR data. Because autoencoders learn by reconstructing the original input from a corrupted version, imputation is a natural extension*(88, 91)*.

3.3.4.1. Imputing missing data with Autoencoders

We constructed an autoencoder with a modified binary cross entropy cost between the reconstructed layer *z* and the input data *x* to better handle missing data as in Beaulieu-Jones and Greene (2016) (Formula 1)*(88)*. The modified function takes into account missing data, with *m* representing a "missingness" vector; *m* has a value of 1 where the data is present and 0 when the data is missing. By multiplying by *m* and dividing by the count of present features (sum of *m*) the result represents the average cost per present feature. The weights and biases of the autoencoder are trained only on present features and imputation does not need to be performed prior to training the autoencoder.

$$cost = -\sum_{k=1}^{d}[x_k \log(z_k)\ m_k + (1 - x_k)\log(1 - z_k)\, m_k] \,/\, count(m) \qquad \text{(Formula 1)}$$

With the exception of the modified cost function autoencoders were trained as described by Vincent et al. with a 100 training epoch patience*(44)*. If a new minimum cost was not reached in 100 epochs, training was stopped. The autoencoder with dropout was implemented using the FancyImpute*(72)* and Keras*(89)* libraries with a Theano*(54, 55)* backend.

We performed a parameter sweep to determine the hyperparameters for the autoencoder. In the sweep we included autoencoders of one to three hidden layers and each combination of 2, 4, 10, 100, 200, 500 and 1000 (over-complete representation) hidden nodes per layer. Autoencoders with two hidden layers made up of 500 nodes each are shown for all comparisons (Figure 3.7). Dropout levels of 5, 10, 20, 30, 40 and 50% were evaluated with 20% being shown for all comparisons.

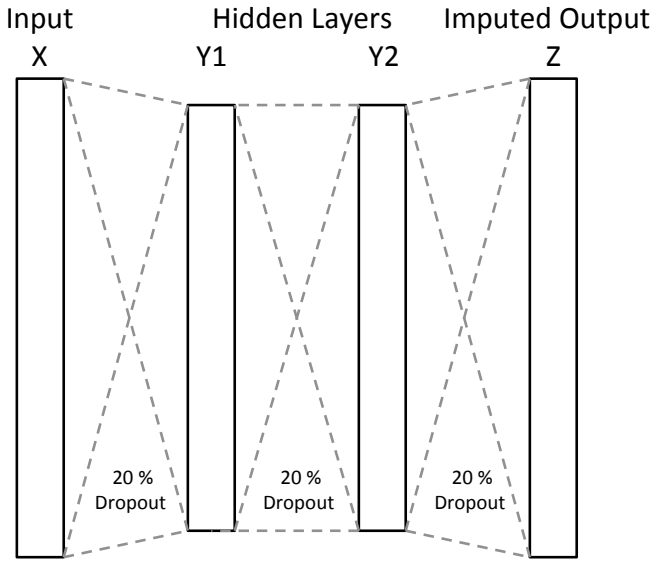


Figure 3.7 : Schematic structure of the autoencoder used for evaluations.
Autoencoder includes two hidden layers and 20% dropout between each layer.

Binary cross entropy was used for training because it tends to be a better evaluator of quality when training neural networks*(44, 53, 92, 93)*. We use a root mean squared error for comparison to other methods to prevent a bias in favor of autoencoders, as most other methods are not trained with cross entropy.

3.3.4.2. Comparative imputation strategies

We used the FancyImpute*(72)* libraries implementations for each of the other imputation strategies:

1.) IterativeSVD, matrix completion by low rank singular value decomposition based on SVDImpute*(76)*.
2.) K-nearest neighbors imputation (KNNimpute), matrix completion by choosing the mean values of the K closest samples for features where both samples are present.
3.) SoftImpute*(94)*, matrix completion by iterative replacement of missing values with values from a soft-thresholded singular value decomposition.
4.) Column mean filling.
5.) Column median filling

The standard implementations of the remaining algorithms in the FancyImpute library, MICE, Matrix Factorization and Nuclear Norm Minimization are known to be slow on large matrices and were impractically slow on this dataset*(95–99)*.

We performed a parameter sweep for SVDimpute analyzing ranks of 5, 10, 20, 40 and 80. Ranks of 40 showed the strongest performance with this dataset and are shown for all comparisons. The parameter sweep for KNNimpute included 1, 3, 5, 7, 15 and 30 neighbors, k of 7 showed the strongest performance of the parameter sweep and is used for all comparisons.

3.3.4.3. Missing Completely at Random Imputation Evaluation

To evaluate imputation accuracy in a missing completely at random environment we performed trials replacing 10, 20, 30, 40 and 50% of known features at random with

NaN. We performed each imputation strategy on the data with spiked-in missingness and evaluated the root mean squared error between the imputed estimates and the original data. We performed five trials for each amount of spiked-in data (Figure 3.8). Performance was evaluated using the root mean squared error between the known value before spiking in missingness and the imputed value.

3.3.4.4. Missing Not at Random Imputation Evaluation

To perform a basic imputation simulation where data was missing not at random, varying percentages (10, 20, 30, 40, and 50%) of features were chosen at random. Half of the highest or lowest (randomly selected) quartile of values was replaced by NaN at random. Each imputation strategy was evaluated on five independent spike-in trails. Performance was evaluated using the root mean squared error between the imputed values and original values. This type of imputation could occur when the highest or lowest values represent the normal range and the clinician is able to determine a patient is normal through other factors. Alternatively, the extreme values could represent a clear result where an additional is not needed to determine the result. Performance was evaluated using the root mean squared error between the known value before spiking in missingness and the predicted value.

3.3.4.5. Progression Prediction Evaluation

To predict disease progression as represented by the ALSFRS score slope, we first imputed the missing data using column mean averaging, column median averaging, SVDImpute, SoftImpute, KNNimpute, and an autoencoder with dropout. For prediction purposes, we excluded all ALSFRS score and Forced Vital Capacity-related features.

We then used the scikit-learn implementation of a random forest regressor*(56)* to predict the ALSFRS score slope. The random forest regressor was chosen because four of the top six teams in the DREAM-Phil Bowen ALS Prediction Prize4Life challenge used variants of random forest regressors*(81)*. We also compare a random forest regressor modified to predict progression from the raw data without imputation*(100)*. Ten-fold cross validation was performed and the root mean squared error between the predicted slope and actual slope was calculated. We then extracted the top 10 most important features used in the trained model for analysis (Figure 3.8).

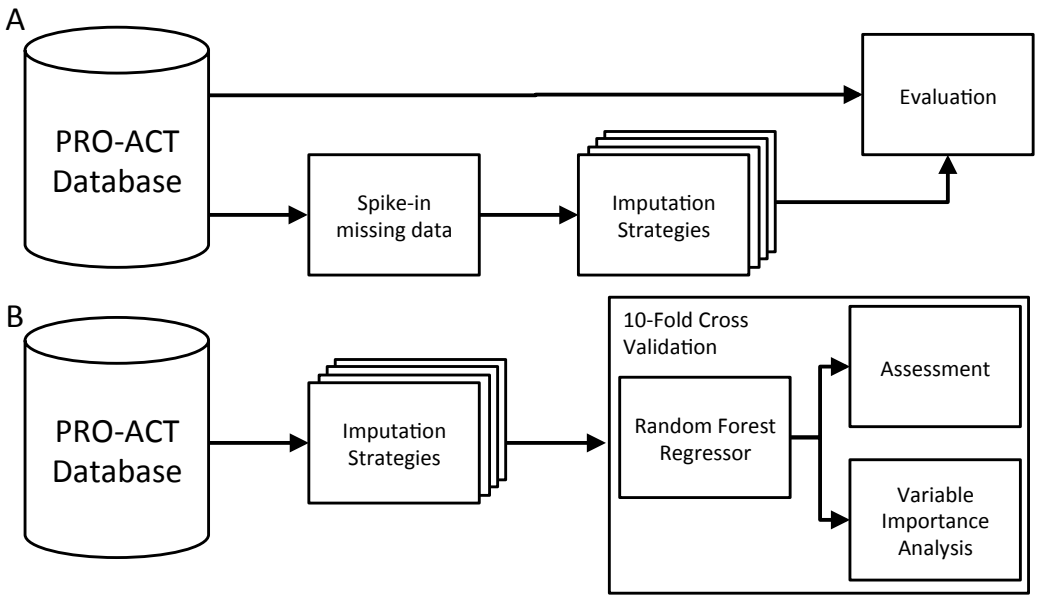


Figure 3.8: Imputation Evaluation Outline.

A.) Imputation Evaluation. PRO-ACT patient data of 10,723 subjects has known data masked with spiked in missing data. Imputation strategies are performed in parallel and the RMSE is calculated between the masked input data and each strategies imputations. **B.)** Progression Prediction. PRO-ACT patients are imputed using each strategy. Ten-fold cross validation of a random forest regressor is performed on imputed patients.

3.3.5. *Results*

Most patients were missing approximately half of the features we extracted from the EHR (Figure 3.9 A). The pooled aspect of the PRO-ACT data is particularly evident in

the distribution of missing features as different clinical trials collected different amounts of data. Features tended to be observed in either less than 25% or in greater than 75% of patients (Figure 3.9 B). Lab tests, in particular, demonstrated high variability of missingness among patients, with many present in small numbers of patients. It is impossible to determine the level of each type of missing data that exists, but it is clear that at least some of data missing is due to clinical factors (trial group etc.). The most complete features are demographics and family history information, information likely collected before entry into any of the clinical trials.

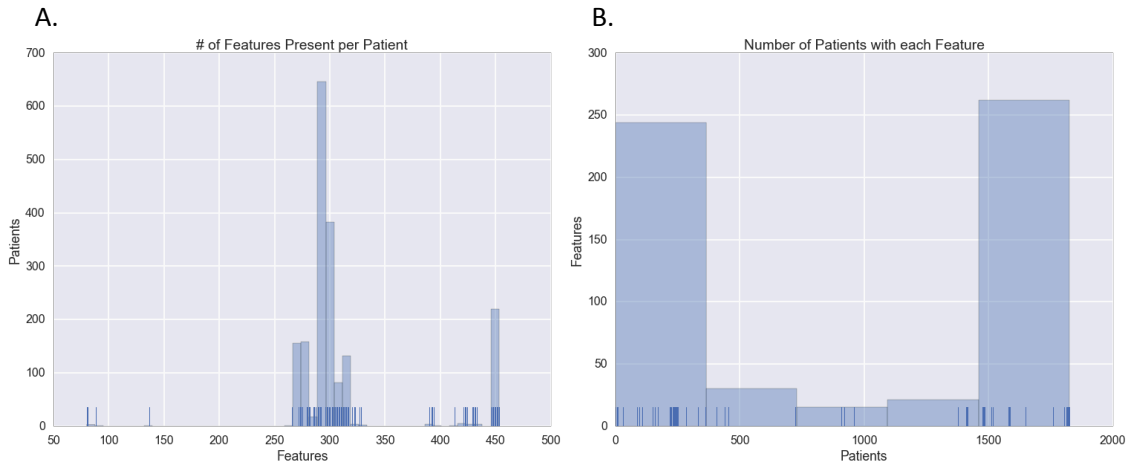


Figure 3.9: Histogram distribution and rug plot showing the number of patients each feature is present in.

A.) The number of features each patient has. Ticks at the bottom indicate one patient with the count of features, bins indicate the number of patients in a range. **B.)** The number of patients having a recorded value for each feature. Ticks at the bottom indicate the number of patients a feature is present in, bins indicate the number of features in a range.

3.3.6. *Missing completely at random spike-in results*

Mean, Median and Singular Value Decomposition imputation perform poorly when data is missing completely at random. However, they do not appear to degrade as the spike-in ratio increase (Figure 3.10). This is not surprising for mean and median imputation because missing data is chosen completely at random and is unlikely to have a

large effect on statistical averages. The autoencoder had the highest imputation performance despite increasing as the spike-in ratio increased.

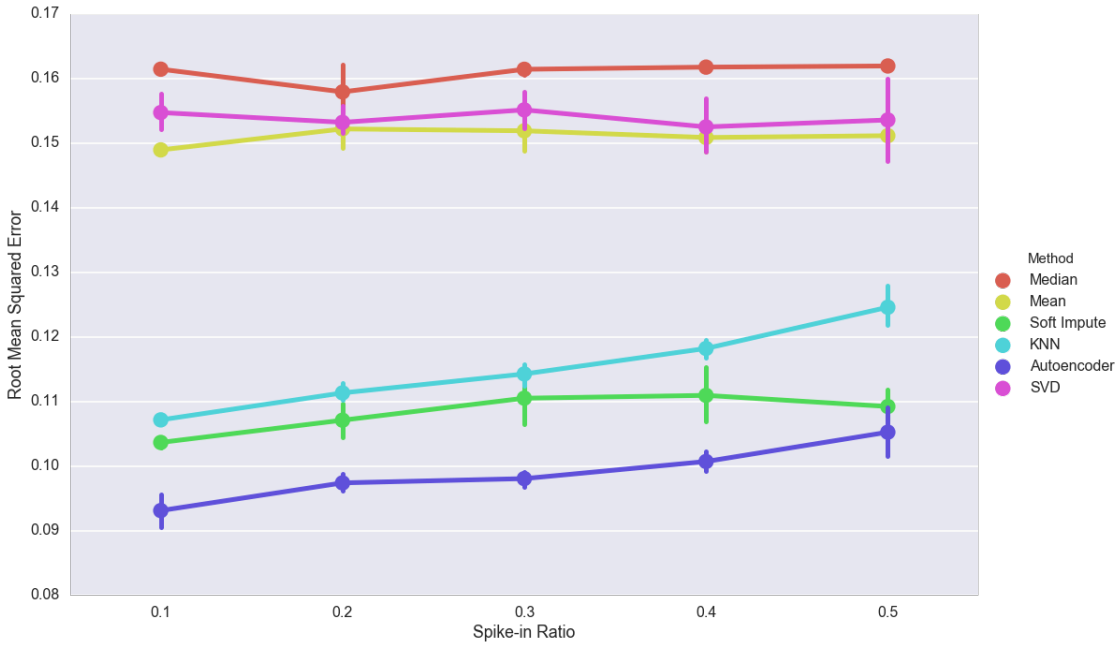


Figure 3.10: Effect of the amount of spiked-in missing data on imputation. Error bars indicate 5-fold cross validation score ranges.

3.3.7. *Not missing at random spike-in results*

The trends seen in the missing completely at random experiment largely repeat when the data is missing not at random. The autoencoder approach shows strong performance but is closely followed by the KNN, Softimpute and SVD approaches (Figure 3.11). KNN works by finding the k-nearest neighbors for shared values and taking the mean for the missing feature. Autoencoders work by learning the optimal network for reconstruction. Similar input values will have similar hidden node values. This similarity could explain the relatively even performance between the two methods. In addition to recognizing similar samples, autoencoders have been shown to perform well when there

is dependency or correlation between variables*(60)*; this is the scenario when data is missing not at random. When spike-in ratios increase to high levels the methods begin to converge to the performance of mean and median imputation. This is likely because too much of the signal is lost as missing data to learn the correlation structure.

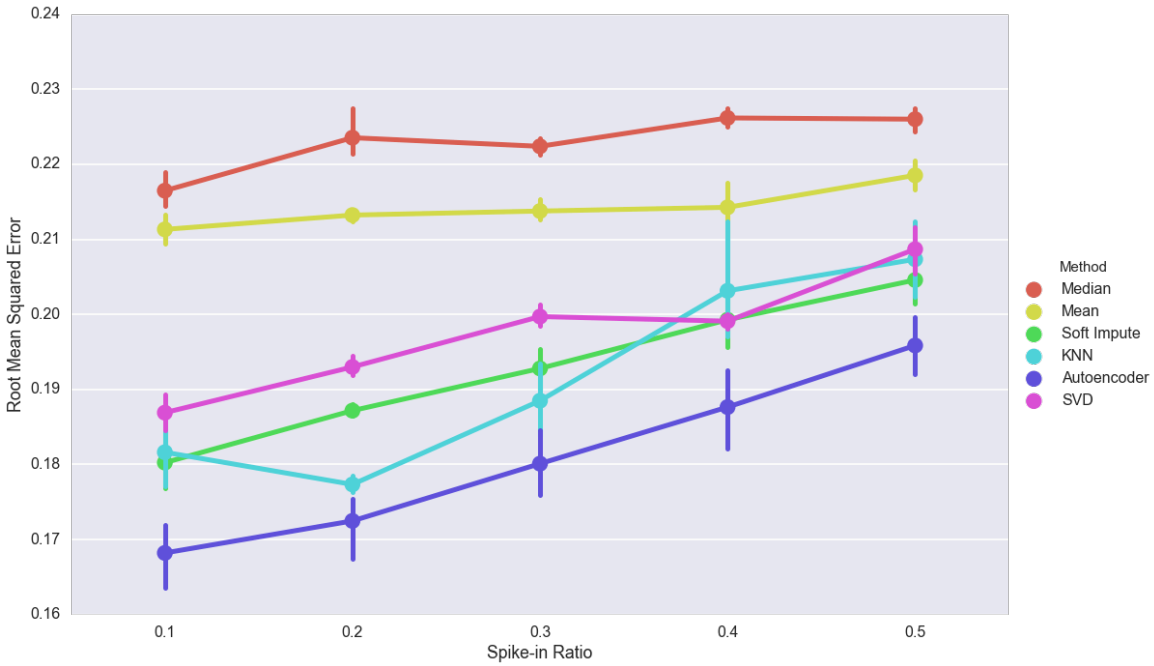


Figure 3.11: Effect of non-random spiked-in missing data on imputation (measured in RMSE). Autoencoder w/Dropout (2 layer 500 nodes each), SVD – SVDImpute with rank of 40, KNN - KNNimpute with 7 neighbors, Mean – Column Mean Averaging, Median – column median averaging, SI – SoftImpute.

3.3.8. *ALS disease progression*

Imputation strategy has a modest but statistically significant impact on the root mean squared error of ALS disease progression prediction, but the autoencoder approach is the strongest performing (Figure 3.12). Despite showing poor performance in the imputation accuracy exercises Singular Value Decomposition does approximately as well as k-nearest neighbors and SoftImpute in this experiment. A random forest regressor applied to the raw data is the worst performing, but is not significantly worse than any of the

methods other than the Autoencoder. In terms of ALS disease progression, imputation does not appear to have a large effect on prediction, but can be vital to allow the use of other algorithms (prediction, clustering etc.) without modification.

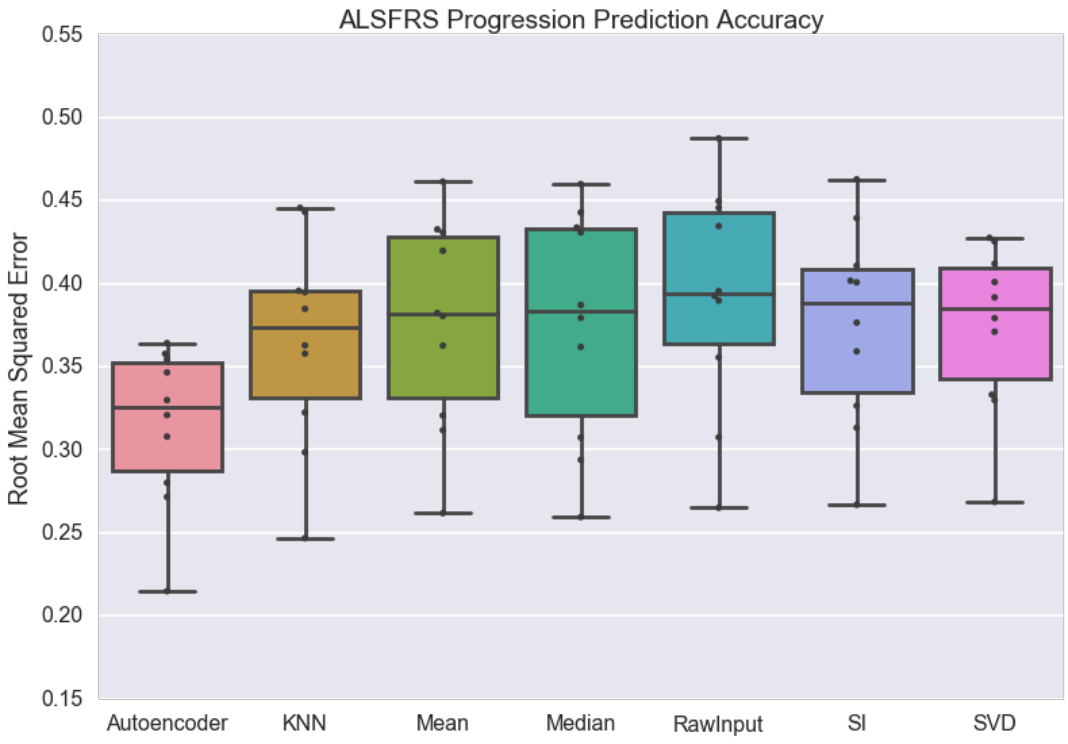


Figure 3.12: ALS Functional Rating Scale prediction accuracy.

Prediction accuracy shown for an autoencoder, k-nearest neighbors, mean averaging, median averaging, the raw input including missing values, soft impute and singular value decomposition. The box indicates inner quartiles with the line representing the median; the whiskers indicate outer quartiles excluding outliers.

3.3.9. *ALS progression predictive indicators*

Nine out of the top ten most important features in the autoencoder-imputed random forest regressor were among the top fifteen identified in the DREAM Prediction challenge (Figure 3.13 A). The amount of time using Riluzole was not among the top fifteen previously identified. Riluzole is the only FDA approved medication for ALS

treatment but it is believed to have a limited effect on survival*(101–103)*. The finding that Riluzole is protective of ALS slope indicates some level of efficacy.

Of the top ten most important features, five are missing in more than 50% of patients in the data set. This is a possible explanation for the improvement shown by Autoencoders, SVDimpute and KNNimpute over mean imputation.

By far the most important feature for prediction is the time from onset and several of the most important features are highly correlated with time from onset. ALSFRS slopes resemble a normal distribution (Figure 3.13 B). When including the entire PRO-ACT dataset, the Kolmogorov-Smirnov test score is 0.05 for patients with negative slopes. This indicates the progression of the disease is similar to a truncated normal distribution. We exclude positive slopes because ALS patients do not typically get better, and signs of doing so are likely the result of measurement error. Despite presenting in clinically heterogeneous manners, ALS progression as defined by the ALSFRS appears to be largely homogenous. Patients fall within a relatively normal distribution and have increasingly negative slopes the longer they ALS.

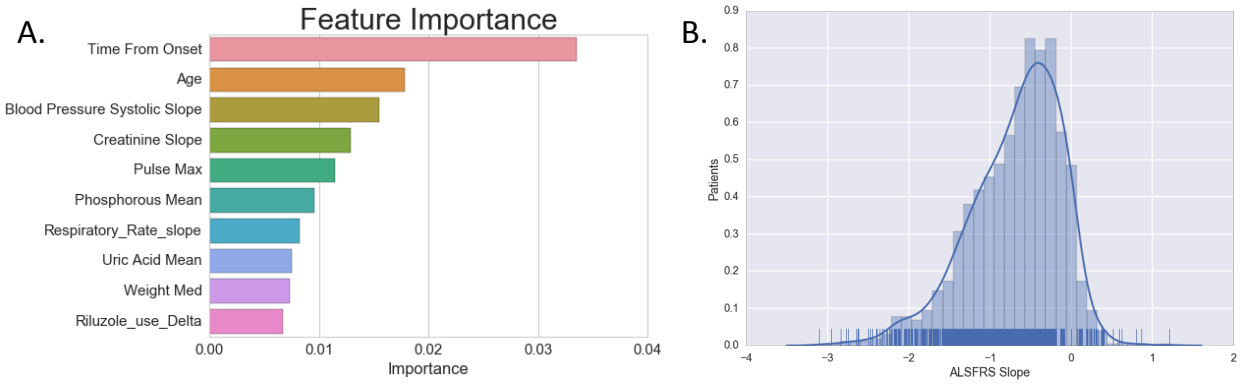


Figure 3.13: Prediction feature importance.

A.) Importance levels of the top 10 features to the random forest regressor with autoencoder imputed data. **B.)** Histogram distribution of patient ALSFRS slope levels.

3.3.10. *Discussion and Conclusions*

In this study, we compared the performance of an autoencoder approach with popular imputation techniques in ALS EHR data. A multi-layer autoencoder with dropout showed robust imputation performance across a variety of spiked-in missing data experiments designed to be both completely at random and not at random. Furthermore, we found that imputation accuracy may not strictly correlate with predictive performance but the most accurate imputer provided the most accurate predictor. The importance of imputation is demonstrated by five of the top ten most important features for prediction being missing in more than 50% of patients.

Increased deterioration of imputation performance for KNNimpute and SVDimpute with increased missing data is at odds with previous research of imputation in microarrays*(76)*. Possible explanations include either reaching a threshold of missing data where the burden is too high for these methods to accurately impute or that a confounding systematic bias is introduced from the different clinical trials.

This work is a promising first step in utilizing deep learning techniques for missing data imputation in the EHR but challenges remain. Autoencoders are computationally intensive, but less so than imputation techniques like MICE, Matrix Factorization and Nuclear Norm Minimization. With GPU resources, autoencoders train in similar amounts of time to both KNN and SVD methods for these clinical trials. As data increases, autoencoder training time increases linearly in line with the number of samples. Methods like KNN require computing a distance matrix, which increases in exponential time. In addition, further examination is necessary to determine whether the strong performance shown by autoencoders is a result of the structure of this pooled clinical trial dataset. The

subset of 1,800 patients is relatively small and methods may differ in performance increases with more patients.

This work offers promising results but has several limitations especially because it specifically analyzes pre-processed pooled clinical trial data. Clinical trials have more complete and cleaner data than raw EHR. Follow up work should be performed with other diseases and in the general patient population. These methods have also only been evaluated for quantitative values; in raw EHR data there will be an additional extraction step for raw text and qualitative observations that was not necessary due to PRO-ACT's preprocessing.

Additional future work will be concentrated on developing tools to better understand and interpret the structure of the trained autoencoder networks. We anticipate being able to recognize patterns in the trained weights to see correlation between input features. Understanding correlation will empower new clustering and visualization opportunities. Spike-in evaluations can provide a supervised context to otherwise unsupervised learning problems; further analysis should be performed on the higher-level learned features in the hidden layers of the autoencoders. We suspect these features may be useful in patient outcome classification and regression problems.

3.3.11. *Acknowledgments*

This work is supported by the Commonwealth Universal Research Enhancement (CURE) Program grant from the Pennsylvania Department of Health to JHM. The authors would like to thank Dr. Casey S Greene for helpful discussions. The authors acknowledge the support of the NVIDIA Corporation for the donation of a TitanX GPU used for this research.

Chapter 4. Enabling data sharing and reproducible research with private data.

This chapter includes adaptations from:

Beaulieu-Jones, Brett K., and Casey S. Greene. "Reproducibility of computational workflows is automated using continuous analysis." Nature Biotechnology 35.4 (2017): 342-346. doi: 10.1038/nbt.3780

B.K.B.-J. and C.S.G. conceived the study and designed the solution. B.K.B.-J. implemented continuous analysis. B.K.B.-J. and C.S.G. wrote, revised and approved the manuscript.

Supplemental Materials and Online Methods are available at: https://doi.org/10.1038/nbt.3780#supplementary-information

And

Beaulieu-Jones, B. K., Wu, Z. S., Williams, C., & Greene, C. S. (2017). Privacy-preserving generative deep neural networks support clinical data sharing. In review (2017). Preprint doi: 10.1101/159756.

B.K.B.-J. and C.S.G. conceived the study. B.K.B.-J. And C.W. performed initial analyses. B.K.B.-J. and Z.S.W. designed and validated the privacy approach. B.K.B.-J., C.S.G. and Z.S.W. wrote the manuscript and all authors revised and approved the final manuscript.

Supplemental Figures are available at: https://doi.org/10.1101/159756

4.1. *Motivation*

Advancement in science depends on the ability to trust previous work as correct and accurate. The ability to reproduce the work in question is one factor in building this trust. When Nature conducted a survey, the vast majority of researchers answered that there 'is a reproducibility crisis' *(104)*. For many experiments, providing a written description of a protocol is insufficient. Computational experiments can be exceedingly complicated and software changes over time. Written protocols often fail to explicitly state the version of

the software used. Even when the version is provided, it may no longer be available. These issues of computational reproducibility are greatly exacerbated when working with sensitive data that cannot be shared. Electronic health records are an example of this "private data". The rightful necessity to protect patient privacy prohibits sharing any identifiable data. This presents several challenges. It is difficult to determine whether a failure to reproduce a previous work is due to a difference in protocol or data. If the error is in replicating the protocol, readers have no way of knowing which step they diverged from the original work. We present technical solutions to both issues, 1.) a method that allows readers to perform analyses in the exact computing environment as the original authors and to view logs with intermediate results from the original experiment, and 2.) a method to generate data that closely resembles unshareable private data while preserving the privacy of study participants.

4.2. *Reproducibility of computational workflows is automated using continuous analysis.*

4.2.1. *Abstract*

Replication, validation and extension of experiments are crucial for scientific progress. Computational experiments are scriptable and should be easy to reproduce. However, computational analyses are designed and run in a specific computing environment, which may be difficult or impossible to match using written instructions. We report the development of continuous analysis, a workflow that enables reproducible computational analyses. Continuous analysis combines Docker, a container technology akin to virtual machines, with continuous integration, a software development technique,

to automatically rerun a computational analysis whenever updates or improvements are made to source code or data. This enables researchers to reproduce results without contacting the study authors. Continuous analysis allows reviewers, editors or readers to verify reproducibility without manually downloading and rerunning code and can provide an audit trail for analyses of data that cannot be shared.

4.2.2. *Introduction*

Leading scientific journals have highlighted a need for improved reproducibility in order to increase confidence in results and reduce the number of retractions*(105–109)*. In a recent survey, 90% of researchers acknowledged that there 'is a reproducibility crisis'*(104)*. Computational reproducibility is the ability to exactly reproduce results given the same data, as opposed to replication, which requires a new independent experiment. Computational protocols used for research should be readily reproducible because all of the steps are scripted into a machine-readable format. However, results can often only be reproduced with help from the original authors, and reproducing results requires a substantial time investment. Garijo et al.*(110)* estimated that it would take 280 hours for a non-expert to reproduce the computational construction of a drug-target network for *Mycobacterium tuberculosis(111)*. Written descriptions of computational approaches can be difficult to understand and may lack sufficient details, including data preprocessing, model parameter selection and software versions, which are crucial for reproducibility. Indeed, Ioannidis et al.*(112)* indicated that the outputs of 56% of microarray gene expression experiments could not be reproduced and another 33% could only be reproduced with discrepancies. Additionally, Hothorn and Leisch found that more than 80% of manuscripts did not report software versions*(113)*.

It has been proposed that open science could aid reproducibility*(107, 114)*. In open science the data and source code are shared. Intermediate results and project planning are sometimes also shared*(115)*. Sharing data and source code are necessary, but not sufficient, to make research reproducible. Even when code and data are shared, it remains difficult to reproduce results due to variability in computing environments, operating systems, and the versions of software used during the original analysis. It is common to use one or more software libraries during a project. Using these libraries creates a dependency to a particular version of the library; research code often only works with old versions of these libraries*(116)*. Developers of newer versions may have renamed functions, resulting in broken code, or changed the way a function works to yield a slightly different result without returning an error. For example, Python 2 would perform integer division by default, so 5/2 would return 2. Python 3 performs floating-point division by default, so the same 5/2 command now returns 2.5. In addition, old or broken dependencies can mean that it is not possible for readers or reviewers to recreate the computational environment used by the authors of a study. In this case it becomes impossible to validate or extend results.

We first illustrate, using a practical example, the problem of reproducibility of computational studies. Then we describe the development and validation of a method named continuous analysis that can address this problem.

4.2.3. *Results*

One example illustrating how data-sharing does not automatically make science reproducible can be found in routine analyses of differential gene expression. Differential expression analyses are performed first by quantifying RNA levels in two or

more conditions and then identifying the transcripts with expression levels that are altered by the experiment. When a DNA microarray is used to measure transcript expression levels, positions on the array correspond to oligonucleotides of certain sequences, termed probes. A certain set of probes is used to estimate the expression level of each gene or transcript. As our understanding of the genome changes, the optimal mapping of probes to genes or transcripts can change as well.

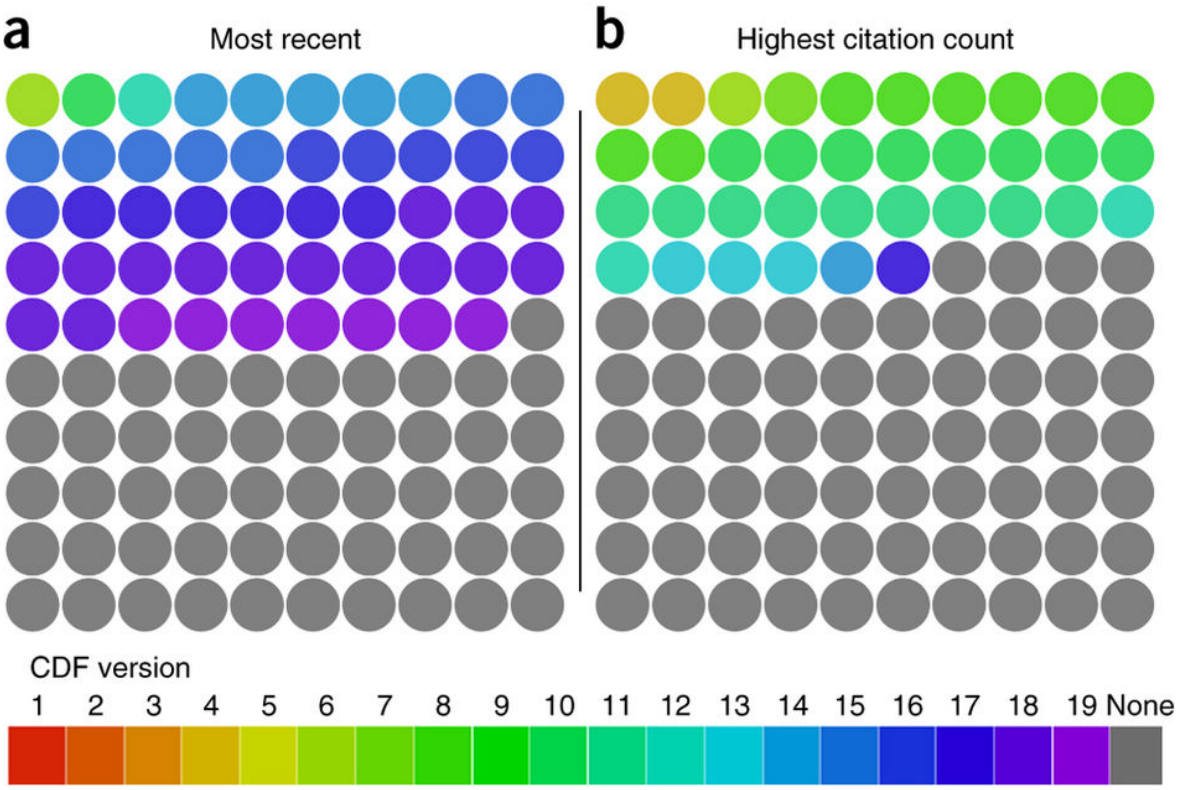


Figure 4.1: Custom CDF version reporting in recent and popular publications.

A.) The 100 most recent publications and the **B.)** 100 most cited papers citing Dai et al.*(117)* that use Custom CDF. Each circle represents one manuscript; color coding indicates the Custom CDF version used.

Dai et al.*(117)* publish and maintain a popular source of probe set description files that are routinely updated (BrainArray Custom CDF). Analyses that fail to report the probe set version, or that were performed with probe set definitions that are now missing,

can never be reproduced. We set out to ascertain the extent of this problem through a literature search. We analyzed the one hundred most recently published papers citing Dai et al. that were accessible at our institution (Supplementary Data 1). We identified these manuscripts using Web of Science on November 14, 2016. We recorded the number of papers that cited a version of Custom CDF, including which version was cited. These articles adhered to expectations of citing methods appropriately because they cited the source of their probe set definitions. Of these 100 papers, 49 (49%) specified which version was used (Figure 4.1 A). These manuscripts reported the use of versions 6, 10, 12, 14, 15, 16, 17, 18, and 19 of the BrainArray Custom CDF. As of November 14, 2016 versions 6 and 12 were no longer available for download on the BrainArray web site. To determine the extent to which a lack of version reporting of the probe set affects high impact papers, we analyzed at the one hundred most cited papers that cite Dai et al. (Supplementary Data 2). We determined the one hundred most cited papers using Web of Science on November 14, 2016. Of these 100 papers, 36 (36%) specified which version of the Custom CDF was used (Figure 4.1 B). These manuscripts used versions 4, 6, 7, 8, 10, 11, 12, 13, 14 and 17. Versions 4, 6, 7, 11 and 12 were not available for download as of November 14, 2016.

In order to evaluate how different BrainArray Custom CDF versions affect the outcomes of standard analyses, we downloaded a recently published gene expression dataset (GEO accession number GSE47664). The reported experiment with this dataset measured gene expression in normal HeLa cells and HeLa cells with TIA1 and TIAR knocked down*(118)*. We ran the same source code using the same dataset altering only the version of the BrainArray Custom CDF library (versions 18, 19 and 20). Each version

identified a different number of significantly altered genes (Figure 4.2 A). There were 15 genes identified as significant in version 19 that were not identified in version 18, and 10 genes identified as significant in version 18 that were not identified version 19. There were 18 genes identified as significant in version 20 that were not found in version 18 and 14 genes identified as significant in version 18 that were not identified in version 20. These results indicate that study outcomes are not reproducible without an accurate version number.

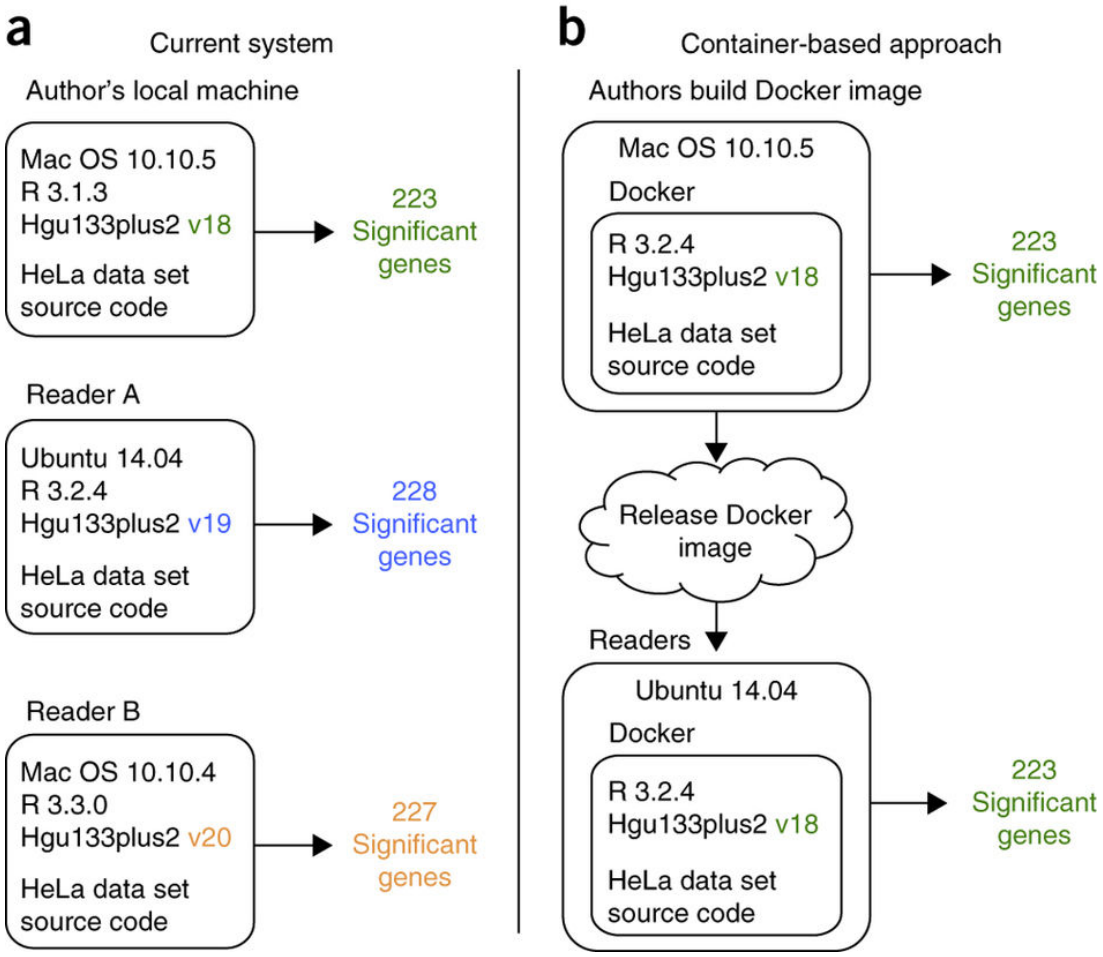


Figure 4.2: Comparison of traditional vs. container based approaches.

Research computing versus container-based approaches for differential gene expression analysis of HeLa cells. Numbers of significantly differentially expressed genes identified using different versions of software packages. (a) and a container-based approach with a defined computing environment (b). n = 3 biological replicates per group (wild-type or double-knockdown HeLa cells)

4.2.3.1. Using Docker containers improves reproducibility

To improve reproducibility, researchers can maintain dependencies using the free open-source software tool Docker*(116)*, *(42)*. Docker can be used to create an "image" that allows users to download and run a container, which is a minimalist virtual machine with a predefined computing environment. Docker images can be several gigabytes in size, but once downloaded can be started in a matter of seconds and have minimal overhead*(116)*. This technology has been widely adopted and is now supported by many popular cloud providers including Amazon, Google and Microsoft.

Docker wraps software into a container that includes everything the software needs to run (operating system, system tools, installed software libraries etc.). This allows the software to run the exact same way in any environment. Boettiger introduced Docker containers as a path to reproducible research by eliminating dependency management, remediating issues caused by imprecise documentation, limiting the effects of code rot (dependencies to specific software library versions) and eliminating barriers to software reuse*(116)*. In addition, Docker images can be tagged to coincide with software releases and paper revisions. This means that even as software is updated, the exact computing environment of a specific older version can be available through the tag of the container's revision history.

In order to assess whether using Docker containers could improve reproducibility of the same experiment we also carried out an analysis of differential gene expression using Docker containers on mismatched machines*(119)*. This process allows versions to

be matched, and produces the same number and set of differentially expressed genes (Figure 4.2 B).

Docker is a useful starting point for reproducible workflows. While it helps to match the computing environment, users must manually rebuild the environment, rerun the analysis pipeline, and update the container after each relevant change. In addition, it does not produce logs of exactly what was run. It also does not automatically track results alongside the specific versions of the code and data that generated them. In summary, Docker can provide manual reproducibility when it is used appropriately.

4.2.3.2. Continuous Analysis

Our goal when developing continuous analysis was to produce an automatic and verifiable end-to-end run for computational analyses with minimal start-up costs. The status quo process requires researchers to perform an analysis and then diligently describe each step and communicate exact versions of software library dependencies used, which can be hundreds or thousands of packages for modern operating systems. To sidestep requiring readers and reviewers to download and install multiple software packages and datasets, continuous analysis preserves the exact computing environment used for the original analysis. A Docker container is built at the time of original analysis and thus includes the exact versions used by the original authors without the risk of packages later becoming inaccessible. The continuous aspect refers to an analysis being rerun, the results saved in version control, and the container being automatically updated after any relevant changes to the software script or data.

Continuous analysis is an extension of continuous integration*(120)*. Continuous integration is widely used in software development. In this workflow, whenever

developers update code in a source control repository, an automated build process is triggered. This automated build process first runs any existing test scripts in an attempt to catch bugs introduced into software. If there are no tests or the software passes the tests, the software is automatically sent to remote servers so that users worldwide can access it.

Our continuous analysis workflows (Figure 4.3) use continuous integration services to run computational analyses, update figures, and publish changes to online repositories whenever changes are made to the source code used in an experiment. We provide continuous analysis workflows for popular continuous integration systems that can be used with multiple types of computing environments including local computing and cloud computing.

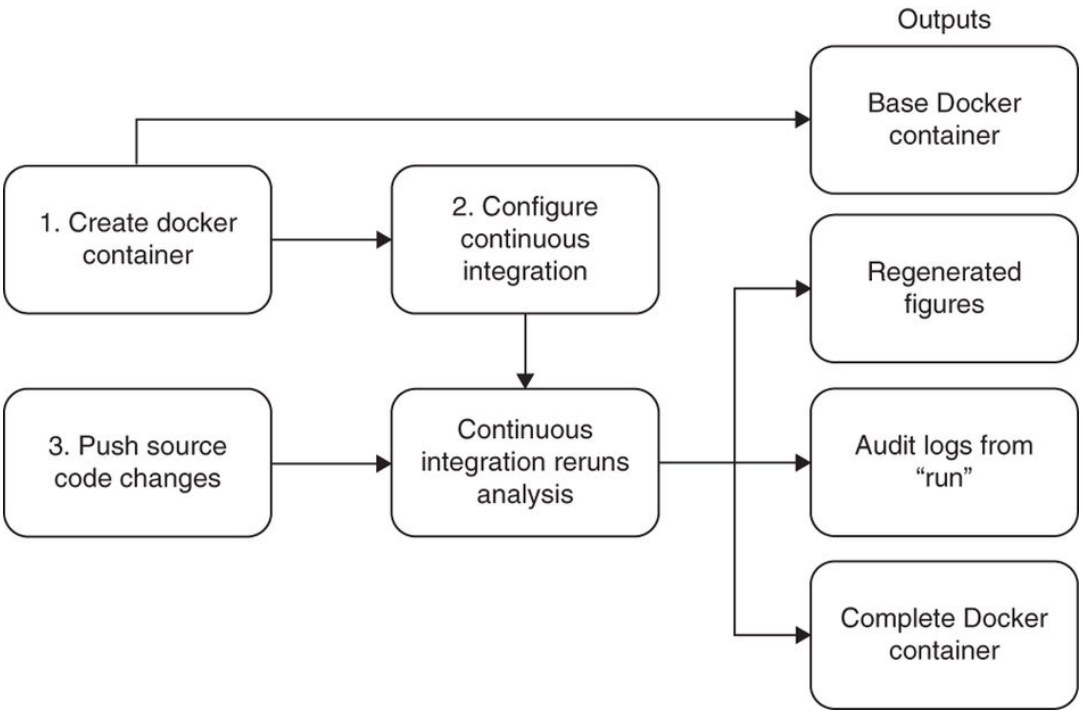


Figure 4.3: Setting up continuous analysis.

Continuous analysis can be set up in three steps. First, the researcher creates a Docker container with the required software (1). The researcher then configures a continuous integration service to use this Docker image (2) then pushes code that includes a script capable of running the analyses from start to finish (3). The continuous integration provider runs the latest version of code in the

specified Docker environment without manual intervention. This generates a Docker container with intermediate results that allows anyone to rerun analysis in the same environment, produces updated figures and stores logs describing what occurred. Example configurations are available in Online Methods and at https://github.com/greenelab/continuous_analysis.

In the continuous analysis workflows that we developed, a continuous integration service is used to monitor the source code repository. Whenever a change is made to a user-specified branch of the repository, the service re-runs the scientific analysis. Workflows are defined in files written in YAML that specify the configuration parameters and commands that should be run. The YAML language is widely used by continuous integration services to specify a human-readable set of instructions. The continuous analysis YAML files that we have developed for multiple services to enable users to employ local computing, cloud-based computing, or commercial service providers.

Each workflow begins by specifying a base Docker image to replicate the researcher's computing environment. The YAML files that we developed provide a place for researchers to choose a base Docker image. Using Docker allows other researchers to re-run code in a computing environment that matches what the original authors used, even if they do not duplicate the original authors' continuous integration configuration. Next, the continuous analysis workflow YAML files specify one or more shell commands required to perform the analysis. Researchers can replace the commands in our examples with their own analytical code. Executing these steps generates the relevant figures from the analysis. Our YAML implementation of continuous analysis then updates the remote source code repository by adding figures and results generated during the run. Finally, a Docker container with the final computing environment is

automatically updated. This continuous analysis process allows changes to be automatically tracked as a project proceeds and pairs each result with the source code, data, and Docker container used to generate it.

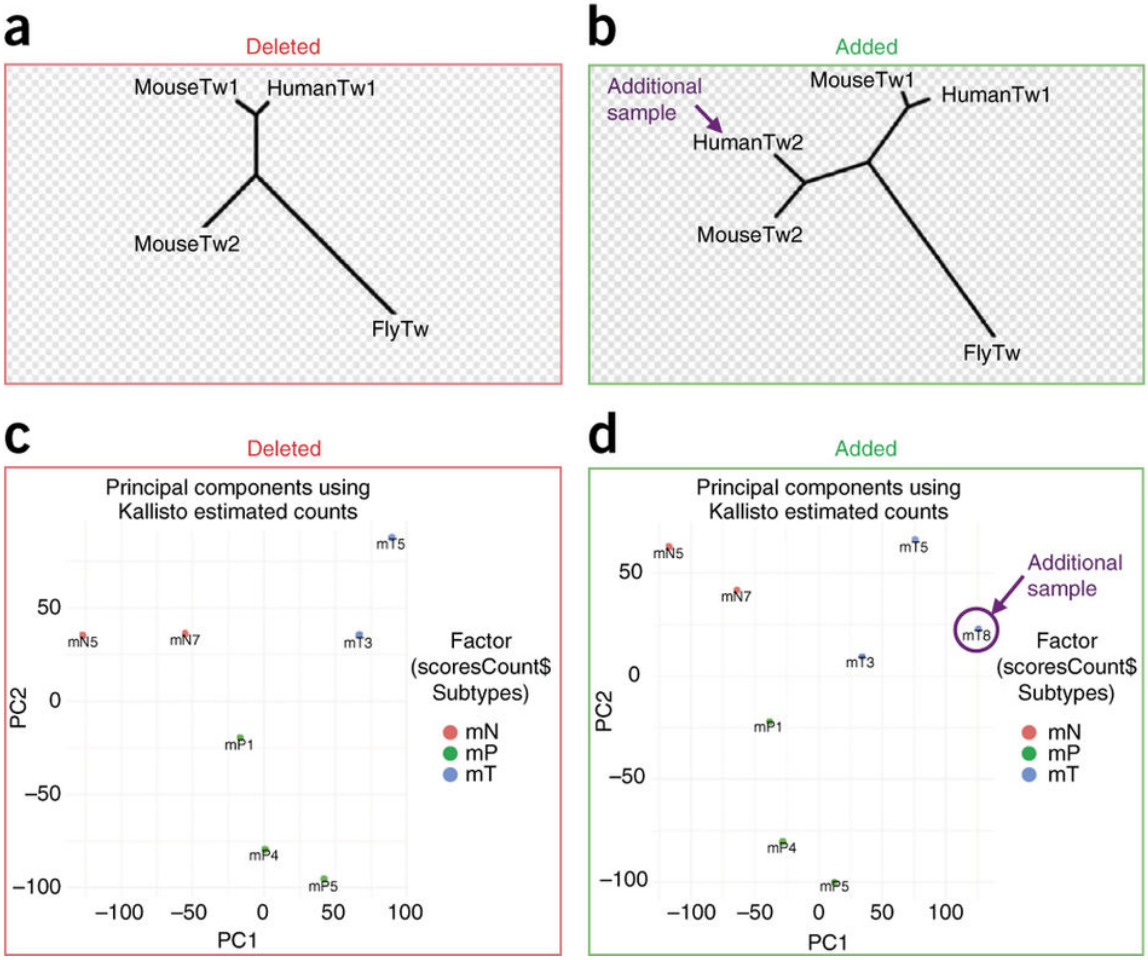


Figure 4.4: Reproducible workflows with continuous analysis.

A.,B.) Phylogenetic tree building with four mRNA samples (MouseTw1, HumanTw1, MouseTw2 and FlyTw) and an additional gene (HumanTw2). **C.,D.)** RNA-seq differential expression experiment principal component (PC) analysis before **C.)** and after **D.)** addition of a sample (mT8).

Using continuous integration in this fashion automatically generates a log of exactly what code was run that is synchronized to the code, data, and computing environment (Supplemental Figure 1). Version control systems allow for images to be easily compared, which provides continuous analysis users with the ability to observe

results before and after changes (Figure 4.4). Interactive development tools, such as Jupyter*(121, 122)*, RMarkdown*(123, 124)* and Sweave*(125)* can be incorporated to present the code and analysis in a logical graphical manner. For example, we recently used Jupyter with continuous analysis in our own publication*(88)* and corresponding repository*(41)*. Reviewers can follow what was done in an audit fashion without having to install and run software while having confidence that analyses are reproducible.

When an author is ready to publish their work, they should archive their repository, which contains the automatically generated results alongside the analytical source code and scripted commands for data retrieval. With our continuous analysis workflows, the authors can use the `docker save` command to export the latest static container, which should also be archived. There are an increasing number of services that allow digital artifacts to be archived and distributed, including Figshare or Zenodo. Journals may also allow authors to upload these files as supplementary elements. If the archiving service used by the authors provides a digital object identifier, future users can easily cite the computing environment and source code. For example, our continuous analysis environment*(126)* and results*(127)* are available in this fashion with results and our source code is provided as Supplementary Source Code 1.

This system imposes minimal cost in terms of time and money on the researcher. Continuous analysis is set up once per project, and will then run automatically for the life of the project. We provide example YAML workflows for commonly used services. Researchers can replace the steps in our example analyses with their own commands to enable automatic reproducibility for their own projects.

4.2.3.3. Setting up continuous analysis

To set up continuous analysis, a researcher needs to do three things. First they must create a Dockerfile, which specifies the software required for their analysis . Second, they need to connect a continuous integration service to their version control system and add a continuous analysis command script to run their analysis. Finally, they need to save their changes to the version control system. Many researchers already perform the first and third tasks in the course of standard procedures for computational research.

The continuous integration system (Figure 4.3) will automatically rerun the specified analysis with each change, precisely matching the source code and results. It can also be set to listen and run only when changes are marked in a specific way, e.g. by committing to a specific 'staging' branch. For the first project, continuous analysis can be set up in less than a day. For subsequent projects, the continuous analysis protocol can be amended in less than an hour.

We have set up continuous analysis using the free and open source Drone software on a PC and connected it to the GitHub version control service (detailed instructions are available in Online Methods). This method is free to users. Our GitHub repository and Supplementary Source Code include continuous analysis YAML scripts for local, cloud-based, and full-service paid configurations*(127)***.** However, it is important to note that while full service providers can be set up in minutes, they may impose computational resource limits or monthly fees. Private installations require configuration but can scale to a local cluster or cloud service to match the computational complexity of

all types of research. With free, open-source continuous integration software*(128)*, computing resources are the only associated costs.

We suggest a development workflow where continuous analysis runs only on a selected branch (Supplemental Figure 2). Our example setup configures a "staging" branch for this purpose. Researchers can push to this branch whenever they would like to generate results files and figures. If the updates to this branch succeed, the changes – along with the results of analyses – are then automatically carried over to the master or production branch and released.

4.2.3.4. Reproducible workflows

Following initial setup, continuous analysis can be adopted into existing workflows that use source control systems. We used continuous analysis in our work using neural networks to stratify patients based on their electronic health records*(41)*. In addition, we provide two example analyses using continuous analysis: a phylogenetic tree building and RNA-seq differential expression analysis.

The phylogenetic tree-building example (detailed in Online Methods) aligned 4 mRNA sequences (MouseTw1, HumanTw1, MouseTw2 and FlyTw) using MAFFT*(129)* and built a phylogenetic tree with these alignments using PHYLIP*(130)* (Figure 4.4 A). After adding an additional sample (HumanTw2), continuous analysis rebuilt the tree (Figure 4.4 B).

The RNA-seq example (detailed in Online Methods) demonstrated differential expression analysis between three different organoid models of pancreatic cancer in mice based on Boi et al.*(131)* (GEO accession number GSE63348) while reusing source code from Balli*(132)*. This analysis used kallisto*(133)*, limma*(134, 135)*, and sleuth*(136)* to

quantify the transcript counts, performed principal components analysis and ran a differential expression analysis. The analysis was initially performed with 7 samples (Figure 4.4 C). An 8th sample was added to show how continuous analysis tracked results (Figure 4.4 D). This example also demonstrated the ability of continuous analysis to scale to the analysis of large datasets – this GEO accession includes 150GB of data (approximately 480 million reads).

4.2.4. *Discussion*

Continuous analysis provides a verifiable end-to-end run of scientific software in a fully specified environment, thereby enabling true reproduction of computational analyses. Because continuous analysis runs automatically, it can be set up at the start of any project to provide an audit trail that allows reviewers, editors and readers to assess reproducibility without a large time commitment. If readers or reviewers need to re-run the code on their own (e.g. to change a parameter and evaluate the impact on results), they can easily do so with a Docker container containing the final computing environment and results that has been automatically kept up to date. Version control systems enable automatic notification of code updates and new runs to those who "star" or "watch" a repository on services such as Github, Gitlab, and Bitbucket. Wide adoption of continuous analysis could conceivably be linked with the peer review and publication process allowing interested parties to be notified of updates.

Continuous analysis can also be applied to closed data that cannot be released e.g. patient data. Without continuous analysis, reproducing or replicating computational analyses based on closed data is dependent on the original authors completely describing each step, which often becomes relegated to supplementary information. Readers and

reviewers must then diligently follow complex written instructions without any confirmation they are on the right track. The pairing of automatically updated containers, source code, and results with the audit log provides readers with confidence that results would be replicable if the data were available. This allows independent researchers to attempt to replicate findings in their own non-public datasets without worrying that a failure to replicate could be caused by source code or environment differences.

Continuous analysis currently has limitations. It may be impractical to use continuous analysis at every commit for generic preprocessing steps involving very large data or analyses requiring particularly high computational costs. In particular, steps that take days to run or incur substantial costs in computational resources (e.g. genotype imputation) may be too expensive for existing providers*(137)*. We demonstrate continuous analysis on a user-defined "staging" branch. This enables researchers to control costs by choosing when to trigger analyses that are automatically reproducible. We strongly recommend the use of continuous analysis whenever a single machine can be used. For workflows that require cluster computing, employing continuous analysis is technically feasible but requires significant systems administration expertise because the cluster must be provisioned automatically. For work involving cluster computing, researchers may elect to employ continuous analysis for steps that do not require a cluster. In this case, researchers should carefully report which steps in their workflows the process covers. It is conceivable that continuous analysis systems could be specifically designed for scientific workflows to facilitate reproducible cluster-based analyses with the same ease as single-machine analyses.

For small datasets and less intensive computational workflows it is easiest to use a full service continuous integration service. These services have the shortest setup times – often only requiring a user to enable the service and add a single file to their source code. With private data or for analyses that include large datasets or require significant computing, cloud-based or locally hosted continuous integration server can be employed.

Reproducibility could have wide-reaching benefits for the advancement of science. For authors, reproducible work is credible. Stodden et al.*(138)* highlight the importance of capturing and sharing data, software, and the computational environments. Continuous analysis addresses reproducibility in this narrow sense by automatically capturing the computational reagents needed to generate the same results from the same inputs. It does not solve reproducibility in the broader sense: how robust results are to parameter settings, starting conditions and partitions in the data. By automating narrow-sense reproducibility, continuous analysis lays the groundwork needed to address questions related to the reproducibility and robustness of findings in the broad sense.

4.2.5. *Acknowledgements*

This work was supported by the Gordon and Betty Moore Foundation under a Data Driven Discovery Investigator Award to CSG (GBMF 4552). A Commonwealth Universal Research Enhancement (CURE) Program grant from the Pennsylvania Department of Health supported BKB. We would like to thank D. Balli for providing the RNA-seq analysis design, K. Siewert for providing the phylogenetic analysis design, and A. Whan for contributing a Travis-CI implementation. We also thank M. Paul, Y. Park, G. Way, A. Campbell, J. Taroni, and L. Zhou for serving as usability testers during the implementation of continuous analysis.

4.3. *Privacy-preserving generative deep neural networks support clinical data sharing*

4.3.1. *Abstract*

Though it is widely recognized that data sharing enables faster scientific progress, the sensible need to protect participant privacy hampers this practice in medicine. We train deep neural networks that generate synthetic subjects closely resembling study participants. Using the SPRINT trial as an example, we show that machine-learning models built from simulated participants generalize to the original dataset. We incorporate differential privacy, which offers strong guarantees on the likelihood that a subject could be identified as a member of the trial. Investigators who have compiled a dataset can use our method to provide a freely accessible public version that enables other scientists to perform discovery-oriented analyses. Generated data can be released alongside analytical code to enable fully reproducible workflows, even when privacy is a concern. By addressing data sharing challenges, deep neural networks can facilitate the rigorous and reproducible investigation of clinical datasets.

4.3.2. *Introduction*

Sharing individual-level data from clinical studies remains challenging. The status quo often requires scientists to establish a formal collaboration and execute extensive data usage agreements before sharing data. These requirements slow or even prevent data sharing between researchers in all but the closest collaborations.

Recent initiatives have begun to address cultural challenges around data sharing. The New England Journal of Medicine recently held the Systolic Blood Pressure Trial (SPRINT) Data Analysis Challenge to examine possible benefits of clinical trial data sharing *(139, 140)*. The SPRINT clinical trial examined the efficacy of intensive management of systolic blood pressure (<120 mmHg) compared with standard management (<140 mmHg). Intensive management resulted in fewer cardiovascular events and the trial was stopped early. Reanalysis of the challenge data led to the development of personalized treatment scores *(141)* and decision support systems *(142)*, in addition to a more specific analysis of blood pressure management in participants with chronic kidney disease *(143)*. Such efforts have begun to address cultural norms. Even for this effort which focused on data sharing, investigators were required to execute data use agreements that included clauses to maintain security and prohibit re-identification or sharing.

We sought to remove technical barriers that hamper data sharing. Computer scientists have tackled problems considered to be particularly challenging using deep neural networks *(144)*. This class of machine learning methods is now becoming more widely used in biology and medicine *(145)*. In this work, we trained two deep neural networks against each other to generate realistic simulated participant blood pressure trajectories from the SPRINT trial dataset. One neural network, called the generator, is trained to generate a participant from a set of random numbers. The other neural network, called a discriminator, is trained to classify data as real or generated. As the networks are trained, the generator learns to build samples that fool the discriminator. Networks trained in this way are called Generative Adversarial Networks (GANs) *(146)* and can

also be used for labeled samples *(147)*. A pair of recent preprints have reported participant generation via neural networks *(148, 149)*. However, it is not enough to simply build new examples. Numerous linkage and membership inference attacks have demonstrated the ability to re-identify participants or reveal participation in a study on both biomedical datasets *(18, 150–155)* and from machine learning models *(19, 156, 157)*.

To provide a formal privacy guarantee, we build GANs under the constraint of differential privacy (16). Informally, differential privacy requires that no subject in the study has a significant influence on the information released by the algorithm (see Materials and Methods for a formal definition). Despite being a stringent notion, differential privacy allows us to generate new plausible individuals while revealing almost nothing about any single study participant. This is especially important in the biomedical domain where, for example, Homer et al. showed the ability to identify whether an individual was a part of a study even with complex genomic mixtures *(158)*. Simmons and Berger later developed a method to enable differential privacy for genome-wide association studies *(159)*. Recently, methods have been developed to train deep neural networks under differential privacy with formal assurances about privacy risks *(160, 161)*. In the context of a GAN, the discriminator is the only component that accesses the real, private, data. By training the discriminator under differential privacy, we can produce a differentially private GAN framework.

We evaluated whether or not this approach could generate biomedical data that could be shared for reanalysis while reducing participant privacy risks. We evaluated usefulness by: (1) comparing variable distributions between the real and simulated data, (2)

comparing the correlation structure between variables in the real and simulated data, (3) comparing machine learning predictors constructed on real vs. simulated data. We find that the model learns realistic distributions and that models constructed from the simulated data successfully classify participants in a held-out portion of the underlying real dataset.

4.3.3. *Results*

We used an Auxiliary Classifier Generative Adversarial Network (AC-GAN) *(147)* to simulate participants based on the population of the SPRINT clinical trial. We included all participants with measurements for the first twelve time periods (n=6,502); dividing them into a training set (n=6,000) and a test set (n=502). We trained two AC-GANs using the training set: a traditional, standard, AC-GAN (labeled non-private) and an AC-GAN trained under differentially privacy (labeled private). We used both to simulate data that we compared to the real data. We visualized participant blood pressure trajectories, analyzed variable correlation structure and evaluated transfer learning performance for a machine learning classification task.

4.3.3.1. Auxiliary Classifier GAN for SPRINT Clinical Trial Data.

An AC-GAN (Fig. 4.5 A) is made up of two neural networks competing with each other. We found convolutional layers effectively modeled the sequential measurements and used deep convolutional neural networks for both the generator and discriminator (Fig. 4.5 B-C). We trained the Generator (G) to take in a specified treatment arm (standard/intensive) and random noise and generate new participants that can fool the Discriminator (D). We trained the discriminator to differentiate real and simulated data

from a dataset containing both groups. We repeated this process until the generator created synthetic participants that were difficult to discriminate from real ones.

We trained under differential privacy by limiting the effect any single subject has on the training process and by adding random noise based on the maximum effect of a single subject. From the technical perspective, we limited the effect of participants by clipping the norm of the gradient and added proportionate Gaussian noise. This combination offered plausible deniability, training could have been guided by a different subject within or outside the real training data. The maximum effect of an outlier is limited and bounded. Comparing the loss functions of the private and non-private training process demonstrates the effects of these constraints. Under normal training the losses of the generator and discriminator converged to an equilibrium before eventually increasing steadily (Fig. 4.5 D). Under differentially private training the losses converged to and remained in a noisy equilibrium (Fig. 4.5 F). At the beginning of training the neural networks changed rapidly. As training continued and the model achieved a better fit these steps, the gradient, decreased. When the gradient became very small, the noise outweighed the signal and limited further training.

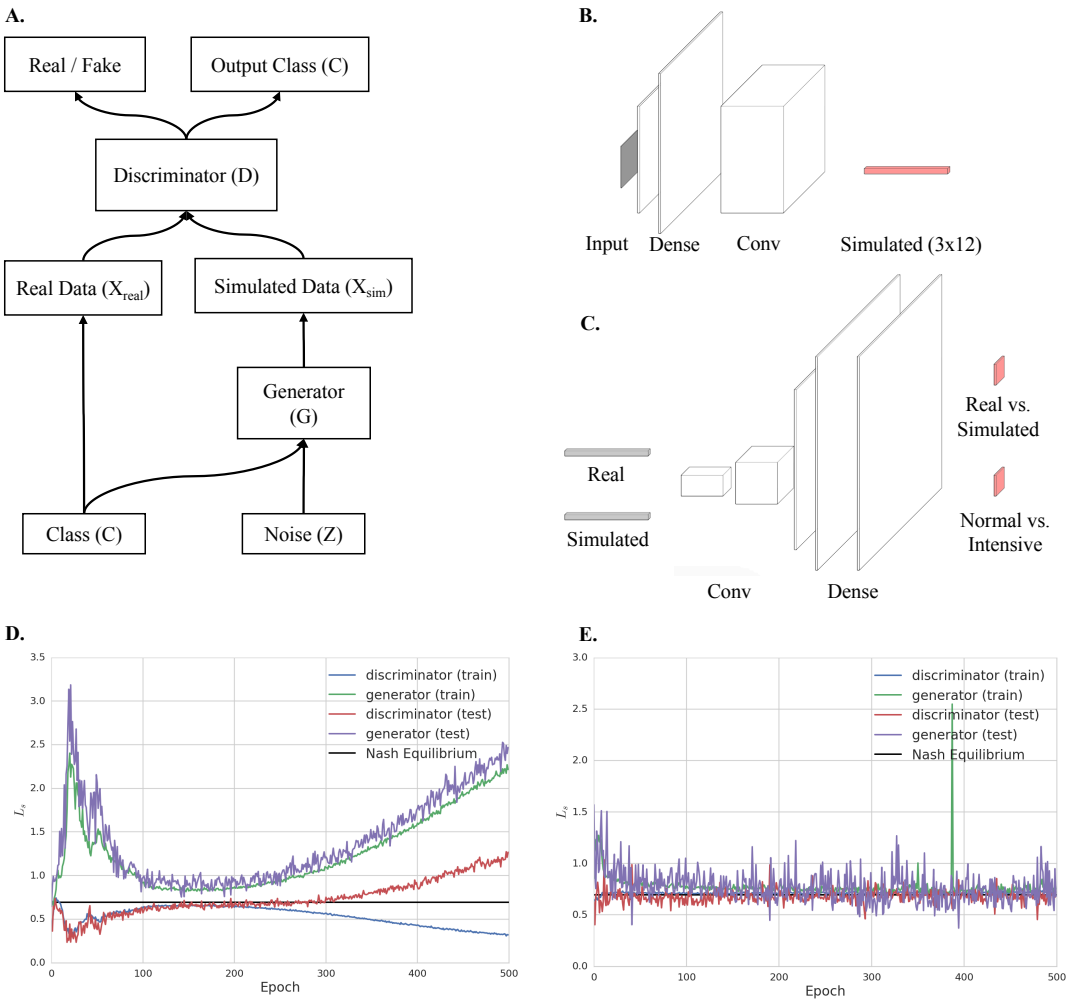


Figure 4.5: AC-GAN architecture and training.

A.) Structure of an AC-GAN. **B.)** The generator model takes a class label and random noise as input and outputs a 3x12 vector for each participant (SBP, DBP and medication counts at each time point). **C.)** The discriminator model takes both real and simulated samples as input and learns to predict the source and a class label (i.e. normal or intensive treatment group). **D.)** Training loss for a non-private AC-GAN. **E.)** Training loss for a private AC-GAN.

4.3.3.2. Evaluation of Simulated Participants

After training the AC-GAN we compared the simulated synthetic participants to the real participants (Figure 4.6). Figure 4.6 shows the median systolic blood pressures for: (1) real participants, (2) simulated participants via a non-private AC-GAN and (3) simulated participants via the differentially private AC-GAN. The non-private

participants generated at the end of training appear similar to the real participants. The private participants have wider variability because of the noise added during training (Fig. 4.6 A). As the models achieve better fit, the gradient shrinks, causing the gradient to noise ratio to decrease. This can occasionally lead to the private generator and discriminator falling out of sync (Supp. Fig. 1) or more commonly the private model generating less realistic samples. To best choose when to stop training, we developed an approach that incorporates any machine learning analysis chosen to resemble expected use cases. Here we tested each epoch's data by training an additional classifier that must distinguish whether a generated participant was a part of the normal or intensive treatment groups. We applied two common machine learning classification algorithms and selected the top epochs in a differentially private manner (Fig. 4.6 B and 4.6 C).

However, selecting only a single epoch does not account for the AC-GAN training process. Because the discriminator and generator compete from epoch to epoch, their results can cycle around the underlying distribution. The non-private models consistently improved throughout training (Supp. Fig. 2A, Supp. Fig. 3A), but this could be due to the generator eventually learning characteristics specific to individual participants. We observed that epoch selection was important for the generation of realistic populations from models that incorporated differential privacy (Supp. Fig. 2B, Supp. Fig. 3B). To address this, we simulated 1,000 participants from each of the top five epochs by both the logistic regression and random forest evaluation and combined them to form a multi-epoch training set. This process maintained differential privacy and resulted in a generated population that, throughout the trial, was consistent with the real population (Fig. 4.6 D).

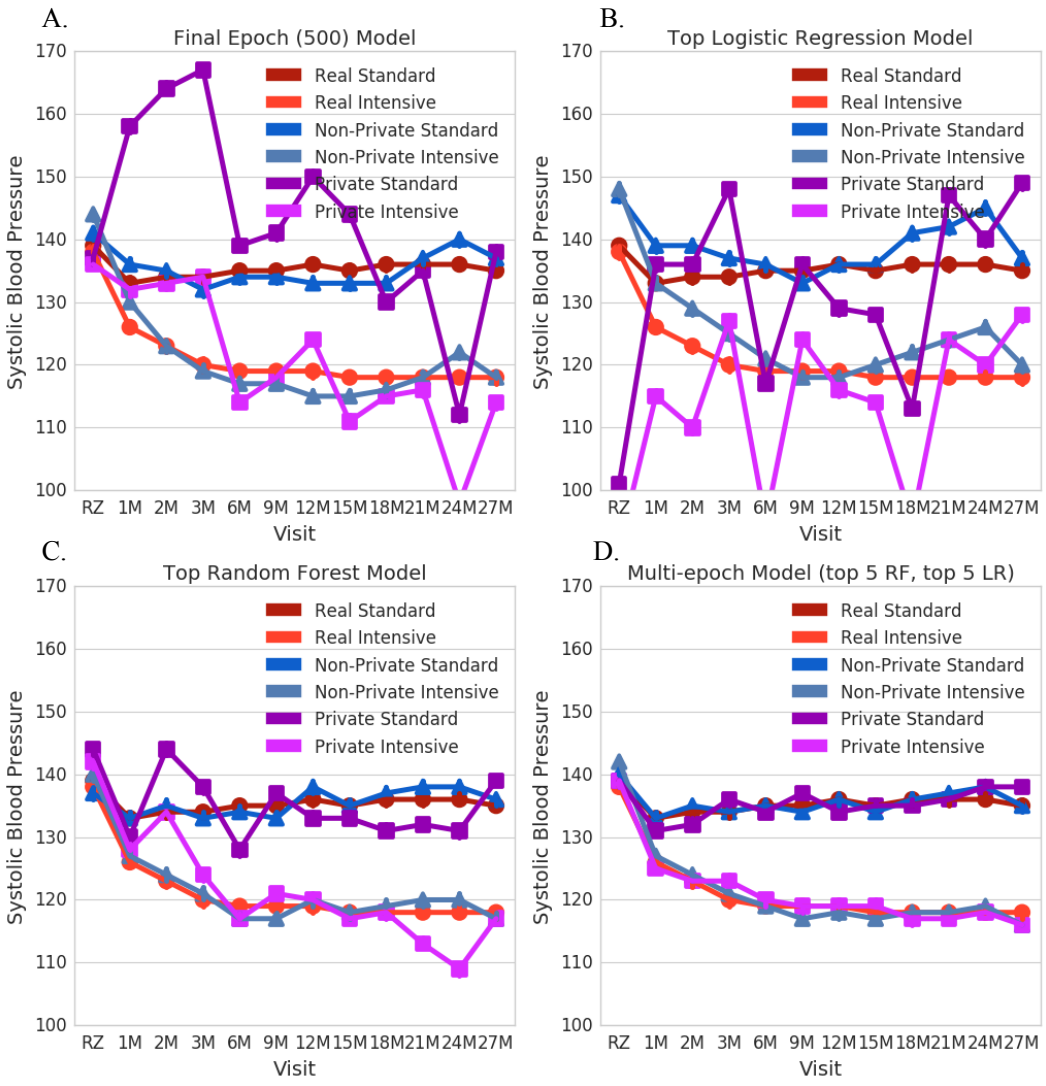


Figure 4.6: Median Systolic Blood Pressure Trajectories from initial visit to 27 months.

A.) Simulated samples (private and non-private) generated from the final (500th) epoch of training. **B.)** Simulated samples generated from the epoch with the best performing logistic regression classifier. **C.)** Simulated samples from the epoch with the best performing random forest classifier. **D.)** Simulated samples from the top five random forest classifier epochs and top five logistic regression classifier epochs.

The Pearson correlation structure of the real data (Fig. 4.7 A) was closely reflected by the correlation structure of the non-private generated data (Fig. 4.7 B). Of note was initial positive correlation between the number of medications a participant was taking

and the early systolic blood pressures, but this correlation decreased as time goes on. The private generated data generally reflects these trends, but has an increased level of noise (Fig. 4.7 C). The noisy training process of the private discriminator places an upper bound on its ability to fit the distribution of data. Increased sample sizes would help to clarify this distribution and because larger sample sizes cause less privacy loss, less noise would need to be added to achieve an acceptable privacy budget.

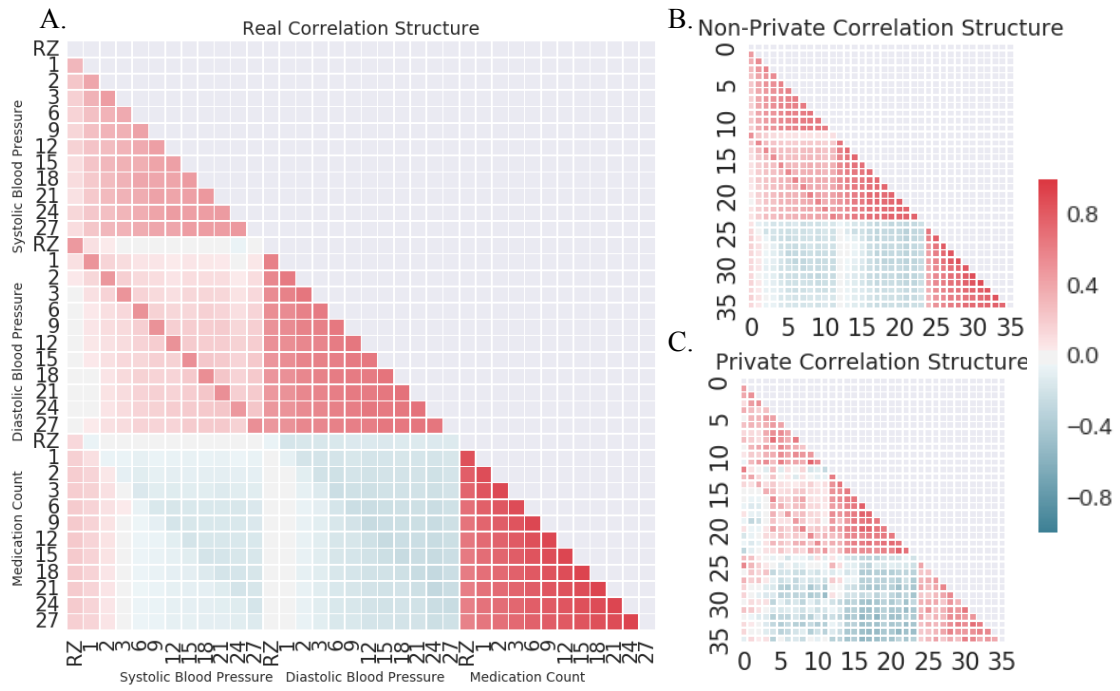


Figure 4.7: Correlation structure between variables in the data.

Pairwise Pearson correlation between columns for the **A.)** Original, real data, **B.)** Non-private, AC-GAN simulated data **C.)** Differentially private, AC-GAN simulated data.

Feasibility of Simulated Participants for Transfer Learning Task

Visualizations of patient distributions and variable correlations showed that synthetic participants appeared similar to real participants. We sought to determine whether or not synthetic participants could be used for subsequent data mining. We trained machine learning classifiers using four methods (logistic regression, random forests, support

vector machines, and nearest neighbors) to distinguish treatment groups on three different sources of data: real participants, synthetic participants generated by the non-private model, and synthetic participants generated by the private model. We compared performance of these classifiers on a holdout test set of 502 real participants (Fig. 4.8 A-D). This analysis revealed two main trends: classifiers trained on the set constructed from combined top epochs exhibited more stable performance on the test data in line with observations from the population distributions, and classifiers trained on data from the non-private model slightly outperformed those trained on data from the private model. A drop in performance was expected because adding noise to maintain privacy reduces signal. If desired, training a non-private model could provide an upper bound for expected performance.

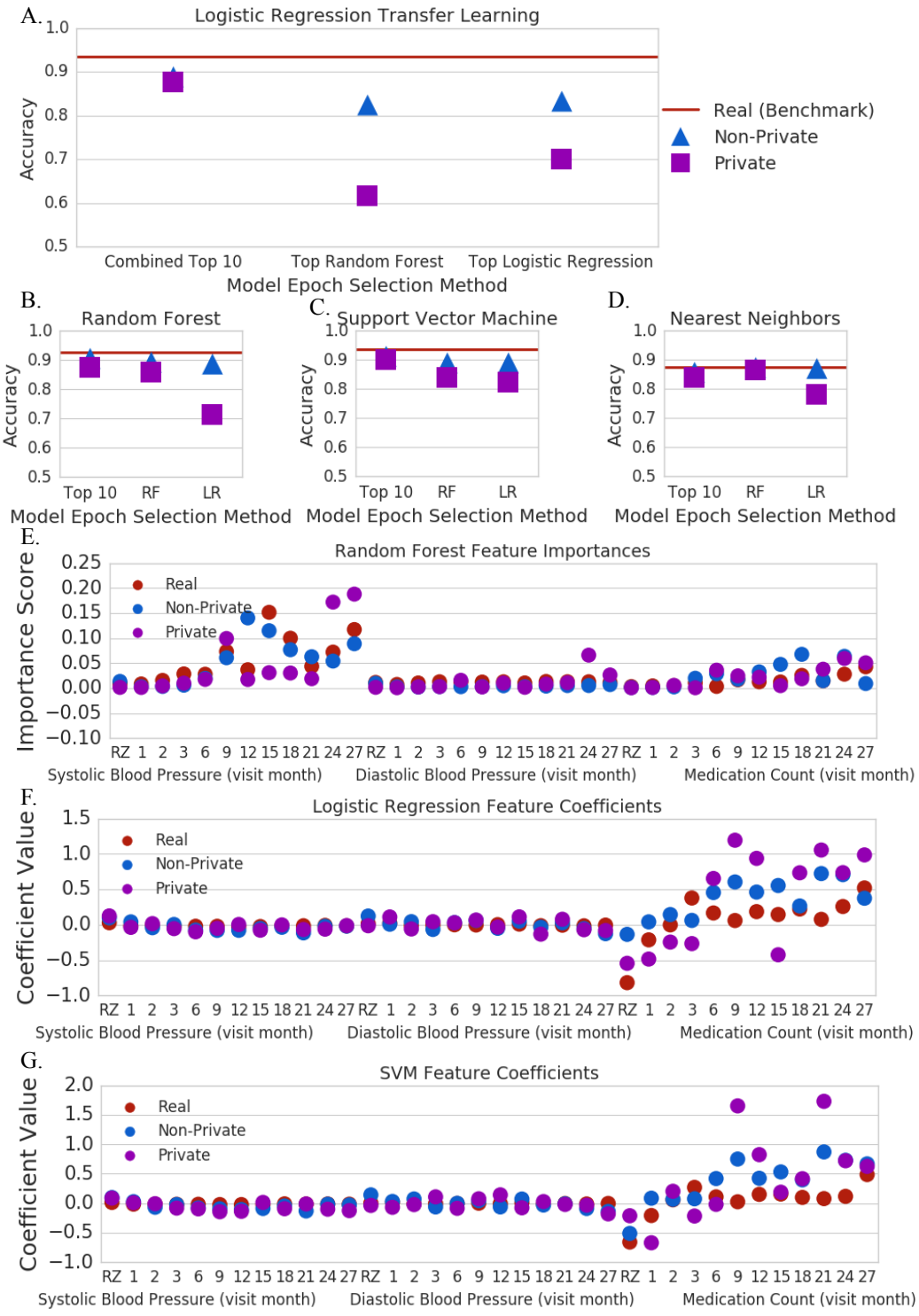


Figure 4.8: Performance and Variable importance in a transfer learning task.

A.) Performance on transfer learning task by source of training data for each machine learning model. **B.)** Random forest variable importance scores by training data. **C.)** Logistic Regression variable coefficients by training data. **D.)** Support Vector Machine variable coefficients by training data.

We also sought to determine the extent to which the classifiers were using similar predictive features. We evaluated the random forest feature importance scores (Fig. 4.8 E) as well as the logistic regression and support vector machine feature coefficients (Fig. 4.8 G, F). All showed similar trends of useful features between real and generated data, and a spearman correlation test was performed between the importance scores (random forest) and coefficients (SVM and logistic regression) of the models trained on real data and each synthetic set revealed significant associations in all cases (Table 4.1). Though all three classification methods achieved similar accuracy, the random forest classifier found the medication features to be important while these features had near zero coefficients in the SVM and logistic regression classifiers.

	Random Forest		**Support Vector Machine**		**Logistic Regression**	
	Correlation	P-Value	Correlation	P-Value	Correlation	P-Value
Real - Non-Private	0.6761	5.998e-06	0.7266	5.2284e-07	0.6687	8.2723e-06
Real - Private	0.6787	5.357e-06	0.7462	1.7493e-07	0.5425	0.00062

Table 4.1: Spearman Correlation between variable importance scores (Random Forests) and model coefficients (Support Vector Machine and Logistic Regression).

4.3.3.3. Privacy Analysis

The formal definition of differential privacy has two parameters. The key parameter ε measures the "privacy loss" incurred by the computation. The second parameter δ bounds the probability that the privacy loss exceeds ε. The values of (ε, δ) accumulate as the algorithm repeatedly accesses the private data. In our experiment, our private AC-GAN algorithm is able to generate useful synthetic data with ε = 2 and δ < 10^{-5} (Figure 4.9). The upper bound of the epoch selection task, (see Materials Methods) used

(0.05, 0) per each model included for a total of (0.5, 0) differential privacy. This established a modest, single digit epsilon privacy budget of (2.5, 10-5).

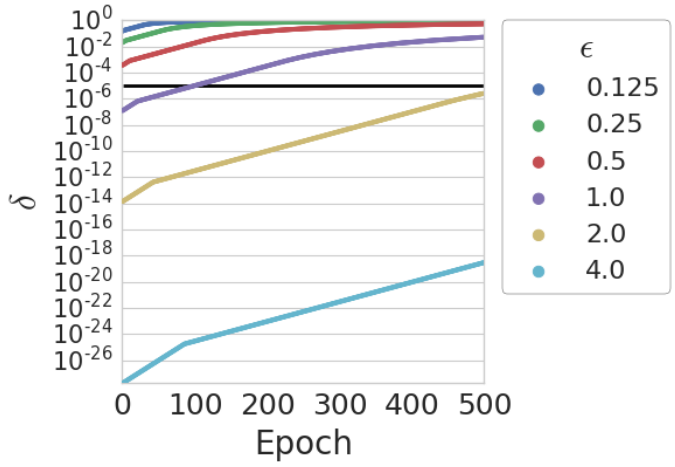


Figure 4.9: Privacy Parameter values during training.

The value of delta as a function of epoch for different epsilon values. An ε value of 2 allows for 500 epochs of training and $\delta < 10^{-5}$.

4.3.4. *Discussion*

Deep generative adversarial networks and differential privacy offer a technical solution to the challenge of sharing biomedical data to facilitate exploratory analyses. Our approach, which uses deep neural networks for data simulation, can generate synthetic data to be distributed and used for secondary analysis. We perform training with a differential privacy framework that limits the study subjects' privacy risk. We apply this approach to data from the SPRINT clinical trial due to its recent use for a data reanalysis challenge

We introduce an approach that samples from multiple epochs to improve performance while maintaining privacy. However, several challenges remain. Deep learning models have many training parameters and require substantial sample sizes,

which can hamper this method's use for small clinical trials or targeted studies. Another fruitful area of use may be large electronic health records systems, where the ability to share synthetic data may aid methods development and the initial discovery of predictive models. Similarly, financial institutions or other organizations that use outside contractors or consultants to develop risk models might choose to share generated data instead of actual client data. In very large datasets, there is evidence that differential privacy may even prevent overfitting to reduce the error of subsequent predictions *(162)*.

Though our approach provides a general framing, the precise neural network architecture may need to be tuned for specific use cases. Data with multiple types presents a challenge. EHRs contain binary, categorical, ordinal and continuous data. Neural networks require these types to be encoded and normalized, a process that can reduce signal and increase the dimensionality of data. New neural network have been designed to deal more effectively with discrete data *(163, 164)*. Researchers will need to incorporate these techniques and develop new methods for mixed types if their use case requires it. The practice of generating data under differential privacy with deep neural networks offers a technical solution for those who wish to share data to the challenge of patient privacy. This technical work complements ongoing efforts to change the data sharing culture of clinical research.

4.3.5. *Materials and Methods*

We developed an approach to train auxiliary classifier generative adversarial networks (AC-GANs) in a differentially private manner to enable privacy preserving data sharing. Generative adversarial networks offer the ability to simulate realistic-looking data that closely matches the distribution of the source data.

AC-GANs add the ability to generate labeled samples. By training AC-GANs under the differential privacy framework we generated realistic samples that can be used for initial analysis while guaranteeing a specified level of participant privacy.

The source code for all analyses is available under a permissive open source license in our repository (https://github.com/greenelab/SPRINT_gan). In addition, continuous analysis *(22)* was used to re-run all analyses, to generate docker images matching the environment of the original analysis, and to track intermediate results and logs. These artifacts are freely available (https://hub.docker.com/r/brettbj/sprint-gan/ and archival version: https://doi.org/10.6084/m9.figshare.5165731.v1).

4.3.5.1. SPRINT Clinical Trial Data

The SPRINT was a randomized, single blind treatment trial where participants were randomized into two groups, an intensive treatment group with a systolic blood-pressure target of less than 120 mmHg and a standard treatment group with a systolic blood-pressure target of less than 140 mm Hg. The trial included a total of 9,361 participants. We included 6,502 participants from the trial by filtering for all participants that had blood pressure measurements for each of the first 12 measurements (RZ, 1M, 2M, 3M, 6M, 9M, 12M, 15M, 18M, 21M, 24M, 27M). We included measurements for systolic blood pressure, diastolic blood pressure and the count of medications prescribed to each participant. This provided an input vector of shape (3, 12).

4.3.5.2. Auxiliary Classifier Generative Adversarial Network

We implemented the AC-GAN as described in Odena et al. *(147)* using Keras *(89)*. Results shown use a latent vector of dimension 100, a learning rate of 0.0002, and a batch size of 100. To handle edge cases and mimic the sensitivity of the real data

measurements, we take the floor of zero or the simulated value and convert all values to integers.

4.3.5.3. Transfer Learning Task

Each of the 6,502 in the SPRINT dataset is labeled by their treatment group. We evaluate machine learning methods (logistic regression, support vector machines, and random forests from the scikit-learn *(56)* package) by their ability to predict which group a participant belongs to. This was done by splitting the 6,502 into a training set of 6,000 participants (labeled real) and a test set of 502 participants. A vanilla AC-GAN was trained using the 6,000 participant training set providing a simulated training set (labeled non-private). A differentially private AC-GAN was trained using the 6,000 training set providing a differentially private training simulated training set (labeled private). Each classifier was then trained on the real, non-private and private training sets and evaluated on the same, real test set of participants. This allows for a comparison of classification performance between models trained on the real data, synthetic data and private synthetic data. We evaluated both accuracy as well as the correlation between important features (random forest) and model coefficients (logistic regression and support vector machine).

4.3.5.4. Differential Privacy

Differential privacy is a stability property for algorithms, specifically for randomized algorithms *(165)*. Informally, it requires that the change of any single data point in the data set has little influence on the output distribution by the algorithm. To formally define differential privacy, let us consider X as the set of all possible data records in our domain. A dataset is a collection of n data records from X. A pair of datasets D and D' are neighboring if they differ by at most one data record. In the following, we will write R to

denote the output range of the algorithm, which in our case correspond to the set of generative models.

Definition 1 [Differential Privacy *(166)*]: Let $\varepsilon, \delta > 0$. An algorithm $A: X^n \rightarrow R$ satisfies (ε, δ)-differential privacy if for any pair of neighboring datasets D, D', and any event $S \subseteq R$, the following holds

$$\Pr[A(D) \in S] \leq \Pr[A(D') \in S] \exp(\varepsilon) + \delta,$$

where the probability is taken over the randomness of the algorithm.

A crucial property of differential privacy is its resilience to post-processing --- any data independent post-processing procedure on the output by a private algorithm remains private. More formally:

Lemma [Resilience to Post-Processing]: Let algorithm $A: X^n \rightarrow R$ be an (ε, δ)-differentially private algorithm. Let $A' : R \rightarrow R'$ be a "post-processing" procedure. Then their composition of running A over the dataset D, and then running A' over the output A(D) also satisfies (ε, δ)-differential privacy.

4.3.5.5. Training AC-GANs in a Differentially Private Manner

During the training of AC-GAN, the only part that requires direct access to the private (real) data is the training of the discriminator. To achieve differential privacy, we only need to "privatize" the training of the discriminators. The differential privacy guarantee of the entire AC-GAN directly follows because the output generative models are simply post-processing from the discriminator.

To train the discriminator under differential privacy we add noise to the stochastic gradient descent process as outlined in Abadi et al. *(160)*. First, we provide an upper bound onto the norm of the gradient at any individual step. This is done by clipping the

ℓ^2-norm of the gradient. Next, we perturb each coordinate of the gradient by adding noise drawn from a Gaussian distribution with a variance proportional to the gradient clipping. The more noise we added (relative to the clipped norm of the gradient) the better privacy guarantee. To achieve a modest privacy budget, we found we could clip the ℓ^2-norm of the gradient at 0.0001 and add noise from a normal distribution with a σ^2 of 1 ($N(\mu, 1 * (0.0001^2))$). This is substantially higher than previously shown, likely due to either the dynamic nature of GAN training where the target is inexact and changes over time or averaging over many mini-batches. We used the moments accountant described in Abadi et al. *(160)* to compute the privacy parameters (ε, δ).

4.3.5.6. Differentially Private Model Selection

We found that sampling from multiple different epochs throughout training provided a more diverse training set. This provided summary statistics closer to the real data and higher accuracy in the transfer learning task. During the GAN training, we saved all the generative models across all epochs. We then generated a batch of synthetic data from each generative model, and used a machine learning algorithm (logistic regression or random forest) to train a prediction model based on each synthetic batch of data. We then tested each prediction model on the real dataset and calculate the resulting accuracy. To select epochs that generate training data for the most accurate models under differential privacy, we used the standard "Report Noisy Min" subroutine: first add independent Laplace noise to the accuracy of each model (drawn from Lab($1/(n*\varepsilon)$) to achieve (ε, 0) differential privacy where n is the size of the private dataset we perform prediction on and output the model with the best noisy accuracy.

In practice, we choose the top five models in the transfer learning task using both logistic regression classification and random forest classification (for a total of 10 models). We performed this task under (0.5, 0)-differential privacy. In each of the ten rounds of selection epsilon was set to 0.05. This achieves a good balance of accuracy while maintaining a reasonable privacy budget.

4.3.6. *Acknowledgments*

We thank Jason H. Moore (University of Pennsylvania), Aaron Roth (University of Pennsylvania), Gregory Way (University of Pennsylvania), Yoseph Barash (University of Pennsylvania) and Anupama Jha (University of Pennsylvania) for their helpful discussions. We also thank the participants of the SPRINT trial and the entire SPRINT Research Group for providing the data used in this study. This work was supported by the Gordon and Betty Moore Foundation under a Data Driven Discovery Investigator Award to C.S.G. (GBMF 4552). B.K.B.-J. Was supported by a Commonwealth Universal Research Enhancement (CURE) Program grant from the Pennsylvania Department of Health and by US National Institutes of Health grants AI116794 and LM010098. Z.S.W. is funded in part by a subcontract on the DARPA Brandeis project and a grant from the Sloan Foundation.

Chapter 5. Taking advantage of the longitudinal nature of Electronic Health Record data.

This chapter was originally published as: Beaulieu-Jones, Brett K., Patryk Orzechowski, and Jason H. Moore. " Mapping Patient Trajectories using Longitudinal Extraction and Deep Learning in the MIMIC-III Critical Care Database." *In Review* (2017). Preprint doi: https://doi.org/10.1101/177428

B.K.B.-J. and J.H.M. conceived of the study. B.K.B.-J. and P.O. performed initial data processing. B.K.B.-J. performed analyses and wrote the manuscript. All authors revised and approved the final manuscript.

5.1. *Abstract*

Electronic Health Records (EHRs) contain a wealth of patient data useful to biomedical researchers. At present, both the extraction of data and methods for analyses are frequently designed to work with a single snapshot of a patient's record. Health care providers often perform and record actions in small batches over time. By extracting these care events, a sequence can be formed providing a trajectory for a patient's interactions with the health care system. These care events also offer a basic heuristic for the level of attention a patient receives from health care providers. We show that is possible to learn meaningful embeddings from these care events using two deep learning techniques, unsupervised autoencoders and long short-term memory networks. We

compare these methods to traditional machine learning methods which require a point in time snapshot to be extracted from an EHR.

5.2.*Introduction*

After the U.S. government mandated meaningful use of electronic health records (EHRs) by 2014, they have been widely adopted with 96% of health care providers implementing an EHR *(167)*. Patient interactions with the health care system are recorded in the EHR. Many research analyses treat the EHR as a static document by taking a snapshot of a patient`s EHR and using this for downstream analyses. This fails to account for the way a patient changes over time, their trajectory.

Jensen et al. *(24)* proposed the idea of temporal disease trajectories to model expected progression for a patient over time. This study uses billing codes as disease labels, which may introduce biases inherent to the billing process. Patients may be assigned a billing code before being diagnosed for a disease in order to receive a diagnostic test. Billing codes place also binary rules on the presence of disease. Perhaps most importantly for this work billing codes are frequently assigned after a visit and are thus not helpful for tracking patient trajectories over the course of an inpatient admission or rapid series of visits.

Interactions between patients and the health care system tend to occur in bursts, related to a specific visit or a series of visits. We label these periods of activity as care events and group these actions together. These care events represent changes over time and can capture longitudinal changes of a patient`s state.

Denny et al. *(168)* first showed the ability to use autoencoders to model clinical measures in an unsupervised manner. More recently, several groups have used

autoencoders to learn high level features useful for classification *(91, 169)* and imputation *(170)*. Tan et al. also showed the ability to extract meaningful features from gene expression data using autoencoders *(26)*. We use autoencoders to represent patient care events in a low dimensional vector space that is useful for visualization. Positions in this vector space represent the patient`s condition at a point in time. By connecting these positions, or care events, in order, it is possible to see how a patient`s condition changes over time and how they move through the health system. It is also possible to cluster patients in this low dimensional space and examine when patient outcomes diverge, one group having high survival and the other having high mortality.

This care event representation also provides a natural sequence of events. Recurrent neural networks have shown an impressive ability to model sequences to solve problems in many domains including object recognition in computer vision *(171)*, image *(172)* and text generation *(173)*. Long short-term memory networks (LSTMs) *(174)* are a type of recurrent neural network that have recently been applied to clinical data to learn low dimension representations of medical concepts *(175)* and to make classifications using time series of specific clinical measures *(176, 177)*.

Trajectories have been used to model multistage dynamic decision processes (DMP) in discrete optimization problems *(178)*. In Algebraic Logical Meta-Model (ALMM) the state of the system in a certain time depends on the previous state, undertaken decision and transition function. This concept allows to easily describe the state of the patient at a particular time, with specific actions taken (e.g. application of medication) to manage the response to previous events within the progression of a disease.

In this work, we first demonstrate that deep learning approaches can (1) learn patient embeddings useful for both interpretable expert analysis via visualization and (2) do this we use the Medical Information Mart for Intensive Care III (MIMIC) database and apply both unsupervised deep autoencoders and LSTMs.

5.3. *Methods*

5.3.1. *Source Code and Analysis Availability*

Source code to reproduce the analyses in this work are provided in our repository (https://github.com/EpistasisLab/MIMIC_trajectories) under a permissive open source license. In addition, Continuous Analysis *(22)* was used to generate docker images matching the environment of the original analysis.

5.4. *Care Event Extraction*

5.4.1. *Medical Information Mart for Intensive Care III (MIMIC) Critical Care Database*

MIMIC *(179)* is a publicly available database composed of 46,297 critical care de-identified electronic health records for patients at Beth Israel Deaconess Medical Center. It includes all charted data (demographics, vital signs, medications, procedures, diagnoses, patient outputs, laboratory tests, physician notes, and treatment details) for patients from 2001 to 2012.

5.4.2. *Extracting Care Events from MIMIC*

We divided the MIMIC database into 4 groups:

1. Static data that does not change over the course of an admission (i.e. demographic data).

2. Actions performed by health care providers that have a specific time associated with them (i.e. laboratory events).
3. Actions performed by health care providers that only have a date associated with them (i.e. oral medications).
4. Streaming data measured on a per-minute basis (i.e. heart rate).

Category (MIMIC Database Table)	Example
DATETIMEEVENT	Changing equipment or standard repeated treatments (i.e. dialysis).
ICUSTAYS	Transfer to or from the Intensive Care Unit.
INPUTEVENTS_CV & INPUTEVENTS_MV	Any fluids given to the patient (i.e. an IV solution, CV and MV stand for the two systems used to track these events Philips Carevue and iMDSoft Metavision).
LABEVENTS	All lab measurements for a patient (i.e. Creatinine level).
PROCEDUREEVENTS	All procedures performed on a patient (i.e. Extubation).
SERVICES	Changes in which service a patient is under (i.e. Cardiac Surgery)

Table 5.1: Categories and examples of Care Event Actions.

To define care events, we included all actions initiated, or charted, by health care providers that have a specific time associated with them (Table 5.1). These actions were placed in sequential order and grouped together until there was a gap greater than the margin time (Figure 5.1). Because this is critical care data, the timeline between events is much smaller than typical EHR data. We found a 59 minute margin time yielded care events that had a good balance of inclusiveness while not including extended time periods. This yielded 1,566,026 total care events and an average of 26.80 care events per admission. In non-critical care datasets, we expect a margin time of several days may better capture the concept of a care event.

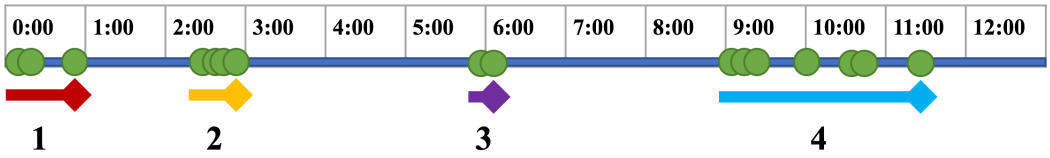


Figure 5.1: Example of care event extraction.

Example of care event extraction. Green circles indicate actions taken by health care providers. Lines and numbers below indicate care events.

5.4.3. *Stratification of Patient Attention based on type of Insurance Provider*

Care events can provide a useful heuristic to the level of interaction between the health care provider and a patient. To evaluate attention, we compared the time spent in the hospital per admission with the number of care events per admission and the average number of care events per day. We then performed Welch's t-test between patients with private insurance and each of the other types of insurance (Medicare, Medicaid, Government, Self-Payment) to see if there were significant differences between patients with differing insurance types.

5.5. *Unsupervised learning to learn embeddings of extracted Care Events*

5.5.1. *Applying Autoencoders to Extracted Care Events to cluster in a low dimensional space.*

We used the Keras library *(89)* to construct autoencoders with 7 hidden layers in (1196, 512, 256, 128, 64, 128, 256, and 512 nodes per layer). We used dropout to mask 20% of the connections between the input layer and the first hidden layer. The model was trained using binary cross entropy loss with Adam *(180)*. The middle, hidden layer (64

nodes) was used as an output for visualization using t-Stochastic Neighbor Embedding *(58)*. The resulting visualizations were labeled for enrichment of 1-year patient survival.

5.5.2. *Predicting Survival Using Care Events*

We evaluated how effectively different machine learning methods could predict patient survival throughout the course of the critical care visit, a 6-month period and a 1-year period (as measured from the original admission date). For this analysis, we performed 5-fold cross validation providing a training set of 46,751 admissions and a test set of 11,687 admissions chosen via stratified cross validation *(56)*. Survival was predicted using a standard feed forward or multi-layer perceptron deep neural network *(89)*, a random forest, logistic regression and support vector machine *(56)* after various numbers (N) of care events: 1, 3, 5, 10, 20, 30 and 50. Area under the curve of the receiver operating characteristic was used for evaluation and comparison between methods.

5.5.3. *Traditional machine learning methods to predict survival from an EHR Snapshot.*

To build a snapshot vector useable for traditional machine learning methods. We took the mean of each value from a set of care events, up to the N^{th} care event. If the patient had less than N care events, we took the mean for all of their care events. This aggregate vector was provided as input to each of the machine learning classifiers.

5.5.4. *Long Short Term Memory Networks (LSTMs) to predict survival with Care Events Sequences.*

To build the sequence vector from a set of care events we first truncated sequences longer than *N*. Sequences shorter than N were post-padded with zeros. The model was comprised of 3 types of layers, an initial embedding layer, three LSTM layers (with 100, 50 and 50 nodes respectively) and a fully connected (Dense) output layer. We trained the model using rmsprop *(181)* with a binary cross entropy loss function.

5.6. *Results*

The MIMIC dataset includes 58,438 admissions from 46,297 unique patients. This was extracted to form 1,566,026 care events (Table 5.2). Medicare patients were double the age of other patients on average. Patients using private or government insurance and Medicaid had relatively equal mortalities during the initial admission and the next 6 months. Patients using Medicare had significantly higher mortality in the 6 months after admission as their time under critical care and self-payment patients had high mortality during the admission but lower admission after leaving critical care.

5.6.1. *Treatment and Outcome Comparison*

	Total	Male	Female	Private	Medicare	Medicaid	Government	Self
Patients	46,297	26,121	20,399	19,663	21,002	4,570	1,614	600
Admissions	58,438	32,950	26,026	22,250	28,103	5,713	1,767	605
Admissions per Patient	1.26	1.26	1.28	1.13	1.34	1.25	1.09	1.01
Average Age at Admission	56.01	54.95	57.34	37.82	75.95	37.91	35.15	39.11
Care Events	1,566,026	867,941	698,085	637,968	693,254	179,182	46,722	8,900
Care Events per Admission	26.80	26.34	26.82	28.67	24.67	31.36	26.44	14.71
Visit Survival	90.84%	90.38%	89.56%	95.24%	86.40%	94.60%	95.87%	85.2%
6-Month Survival	79.81%	79.63%	78.39%	89.84%	69.47%	87.61%	91.62%	83.31%
12-Month Survival	76.28%	76.19%	74.82%	87.90%	64.33%	84.75%	90.32%	82.81%

Table 5.2: Summary statistics for MIMIC Critical Care database.

We examined the length of stay per admission by insurance type (Figure 5.2 A) and found that patients using Medicare had the longest stays but that all groups differed significantly via a Welch's t-test from patients using private insurance. It is not surprising that patients using self-payment had the shortest stays and the least number of care events per stay (Figure 5.2 A-C). Interestingly, patients with private insurance had significantly lower care events per day than the most similar other groups, government-based insurance and Medicaid (Figure 5.2 C).

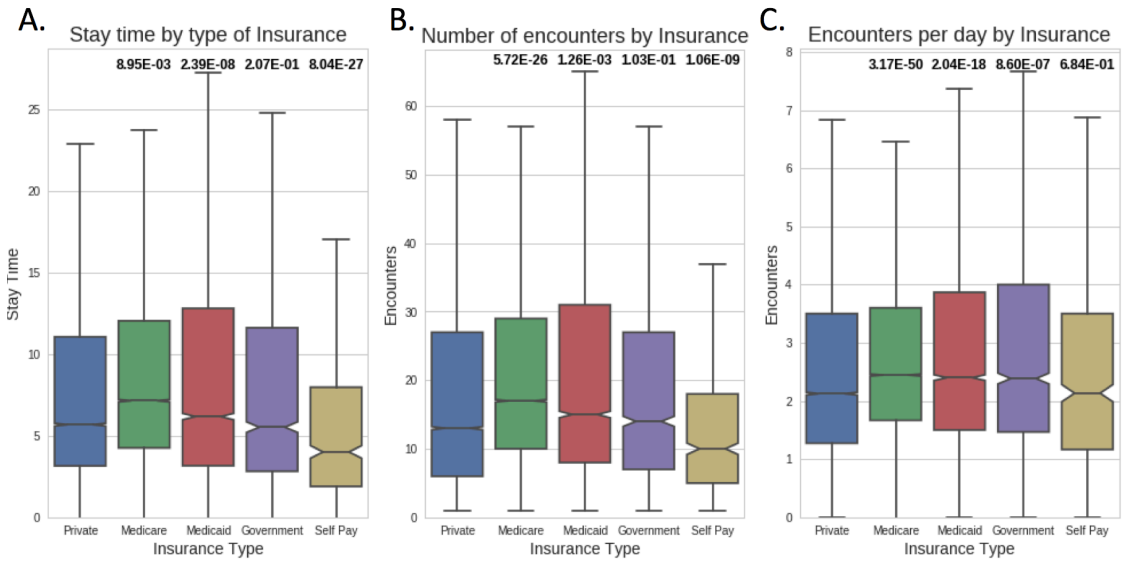


Figure 5.2: Association testing between different insurance types.

A.) Length of admission. **B.)** Number of care events in an admission. **C.)** Number of care events per day of each admission. Labels at the top indicate p-values via Welch's t-test to private group.

5.6.2. *Unsupervised modeling of patient Care Events*

To test whether unsupervised autoencoders could learn meaningful embeddings from individual care events, we plotted the innermost hidden layer using t-Stochastic Neighbor Embedding (t-SNE) and overlaid 1-year survival labels (Figure 5.3). This process yielded several clusters with high enrichment for either mortality or survival indicating the ability to learn meaningful embeddings. t-SNE does not maintain global similarity structure and as such this process is useful for visualizing single care events but not for understanding patient trajectories. In order to examine patient trajectories, it is necessary to look at the value of the innermost hidden layer before t-SNE was applied or to use a method designed to model sequential data. Recurrent neural networks, and specifically LSTMs are well suited at this task.

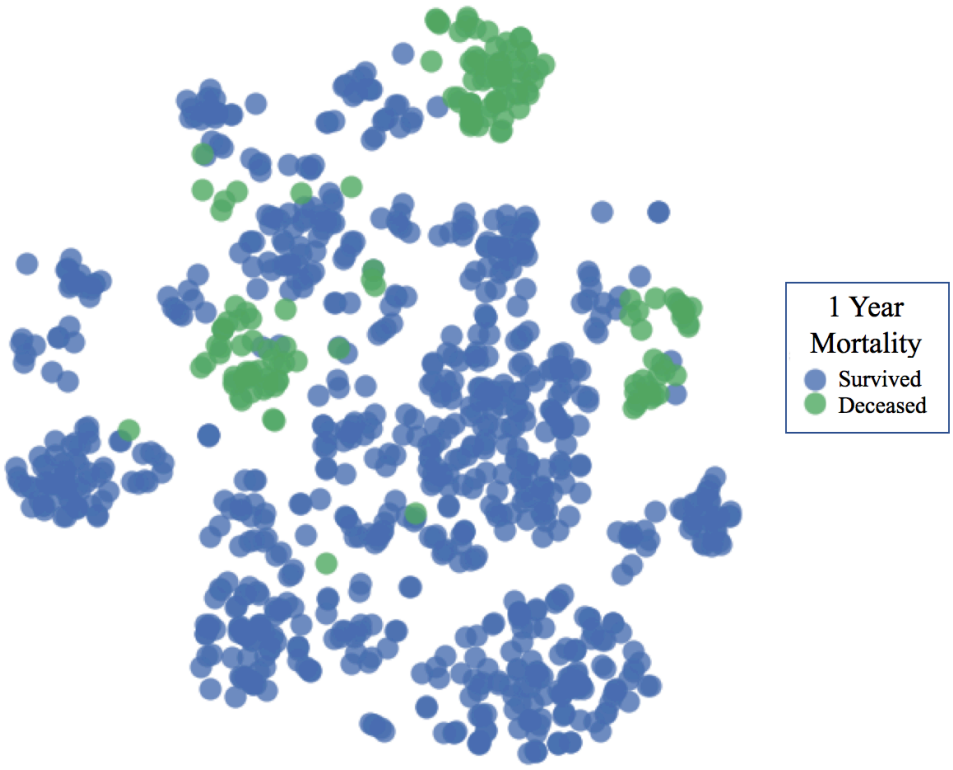


Figure 5.3: Unsupervised Care Events Embedding Visualization.

Performed by applying t-SNE to the innermost layer of autoencoder (1000 care events shown to prevent overplotting).

5.6.3. *Supervised prediction of patient survival*

Next, we preformed the supervised classification task of predicting whether a patient survived one full year from the date of their admission. We measured classification accuracy with differing numbers of care events to evaluate whether the care event based approach had advantages over traditional single point in time measurements (Figure 5.4). Of the methods predicting based on a snapshot, the random forest was by far the most effective. Despite this, it did not increase in performance as more information about an admission was added. This indicates that much of it's predictive power comes from the initial presentation. Both, linear methods and a traditional feed-forward neural network barely outperformed random chance. This may have been due to the high dimensionality

of the dataset. The care event-based LSTM increases in performance as more care events are provided. This is particularly evident when more than the median number of care events (26.8) are provided as input to the LSTM. Including more than 50 care events yielded weaker results for the LSTM. This is likely because most patients have fewer than 50 care events so most of the signal is captured in the first 50 care events. Going beyond 50 leads to a high level of padding to signal.

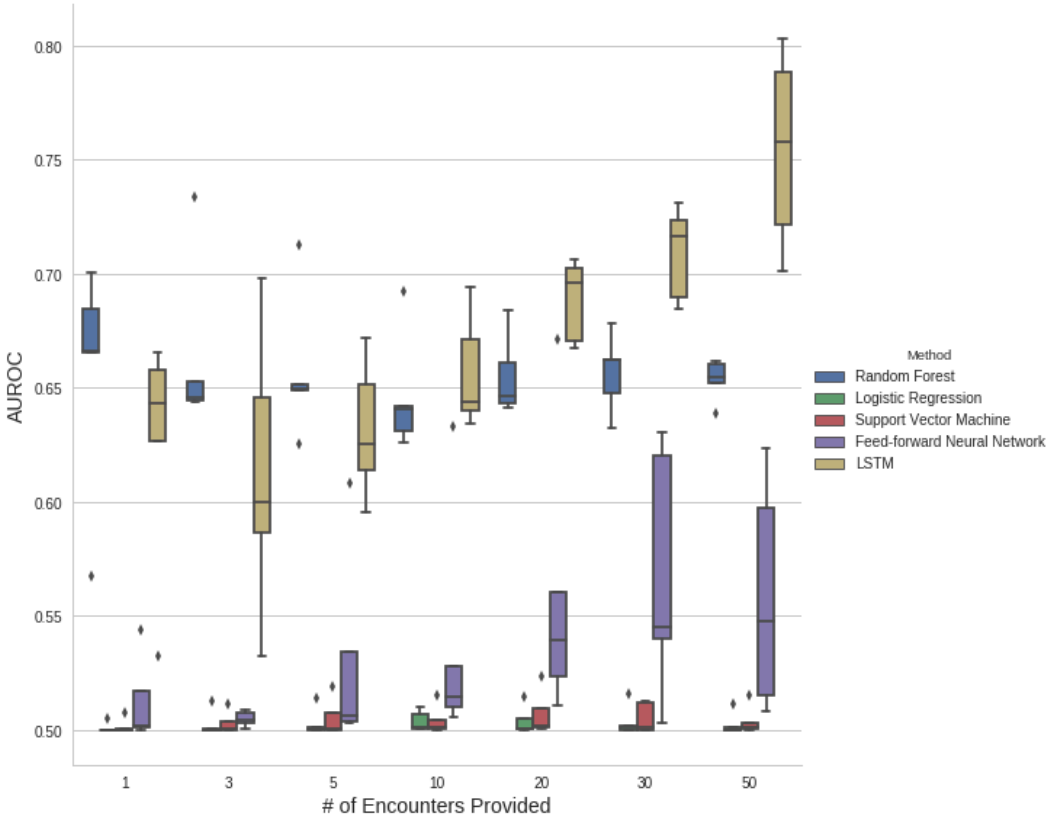


Figure 5.4: Predicting Survival with Care Event Approach.

Comparison of machine learning methods and the number of care events provided for 1-year survival prediction (AUROC).

5.7. *Discussion and Conclusions*

By limiting the usage of summary statistics to small time periods, we offer a granular method for modeling longitudinal clinical data. The care event extraction method

provides a simple data driven approach to extracting temporal data for use in time series analyses. It allows summary statistics to be computed over short time windows as opposed to an entire patient history or arbitrary timestamps. Care events also offer a heuristic to allow comparison of the level of attention different patients receive from health care providers. We demonstrated the ability to learn embeddings enriched for different endpoints using unsupervised deep learning and were able to more accurately predict patient survival using supervised long short-term memory networks.

Though our approach showed strong performance for several tasks in this dataset, this method currently has limitations in terms of generalization. Long-short term memory networks, like many deep learning approaches, require many patients to outperform other methods. This can present a challenge when studying a single phenotype instead of a wide variety of critical care patients. The greatest benefits are likely to be seen when patients have many care events, making this approach particularly well suited for chronic diseases like type 2 diabetes and Crohn's disease or for diseases that are hard to subtype such as multiple sclerosis. An additional challenge is if a patient with a disease like type 2 diabetes suffers an unrelated acute injury (i.e. broken rib in a vehicle accident) this acute injury may introduce too much noise to capture the type 2 diabetes trajectory. In future work, we hope to introduce filtering techniques to exclude or deemphasize unrelated diagnoses.

We also plan to increase the dimensionality of the encoders and applying additional techniques of visual clustering *(182)*. This includes using Shared-Nearest Neighbors (SNN) clustering to find groups of patients with similar stage of the disease in noisy data

and Mukres algorithm to map groups of patients resembling a state of the disease to clusters found in the data.

Another challenge we would like to take is including streaming data in the simulation. Some measurements, e.g. heart rate or blood pressure, are performed every minute for each patient. The information about sudden changes of patient's condition is especially relevant for intensive-care patients. While our method aggregates patient data over shorter time periods than are commonly used, we plan to adapt our model by adding more detailed relevant information extracted from streaming sources.

5.8. *Acknowledgments*

We thank Casey S. Greene (University of Pennsylvania) for his helpful discussions. Funding: This work was supported by the Commonwealth Universal Research Enhancement (CURE) Program grant from the Pennsylvania Department of Health. B.K.B.-J., P.O. and J.H.M. were also supported by US National Institutes of Health grants AI116794 and LM010098 to J.H.M..

Chapter 6. Summary and Future Directions

The ability to more precisely define diseases, extract phenotypes and advise data driven treatment strategies are just a few of the wide array of the possibilities enabled by the wealth of data contained in EHRs. We aim to provide machine learning methods to address the barriers slowing or preventing progress at these tasks. Semi-supervised learning can be used to compensate for the lack of research quality labels available in EHRs. Smart imputation strategies can infer missing values while minimizing impact on downstream analyses. Continuous analysis can synchronize code to figures and enable reproducible research even when working with private data. Adversarial training of deep neural networks can be used to generate synthetic patient populations while preserving privacy and discretization of care combined with recurrent neural networks can model patient trajectories over time. This work takes steps towards solving these barriers which we hope enables new possibilities and raises several new questions around the usage of machine learning for both academic research and predictive analytics in patient care. Despite our progress, there are significant limitations to our approaches and further work will need to be done to completely solve these challenges. This section identifies these limitations and challenges as well as speculates at promising opportunities using machine learning in the EHR over the next several years.

Machine learning in the EHR requires pre-processing and imputation (Chapter 2 & Chapter 3), but best practices in these field currently require an ad hoc implementation followed by analyses to measure how robust results are to differences in pre-processing and imputation. We demonstrated the ability to create a synthetic patient population using GANs (Chapter 4) in a well-controlled structured clinical trial. Generalizing this task

presents several difficult challenges. GANs currently are not effective at generating mixed data types (continuous, ordinal, categorical, etc.). In addition, they require large sample sizes to effectively model a distribution. Training GANs under differential privacy has not yet been shown possible for high dimensional data that is common in biomedical data (i.e. genotypes). Overcoming these challenges will require significant technical adjustments and potentially a reformulation of the problem. Challenges in generalizing the longitudinal analysis (Chapter 5) largely based on two things, 1.) variable times between care events that become more significant outside of the critical care environment, and 2.) the potential of a recurrent neural network to learn from padding, that is to learn form the number of care events rather than the contents of those events. These issues were reduced in the critical care setting because there was a lower variance in the time between care events than there would be in a traditional setting and there was not a correlation between the number of care events and survival.

Traditionally, genetic association studies relied on binary outcomes as target phenotypes for the association. Quantitative trait loci studies provide the ability to measure correlation between DNA variation and a phenotype. Quantitative traits occur on a continuum and are driven by multiple genes in conjunction with the environment. By using clustering and other machine learning techniques, researchers can represent disease as quantitative rather than binary values. This has several advantages. Patients with a common disease that present with different symptoms, different levels of effect or different paths of progression can be clustered into homogenous subgroups with similar patients. These clusters are likely enriched etiologically, meaning the reasons the disease is causing each cluster are different. Within each cluster or across the entire spectrum of

diseases, a phenotype can be constructed to better represent how severe of a case a patient has. Diseases where using a binary case control status has been effective are likely etiological homogenous, or so disruptive to a particular system that the severity is irrelevant. These represent the low hanging fruit, but many diseases present in heterogeneous manners (Cancers, Amyotrophic Lateral Sclerosis, Multiple Sclerosis, Alzheimers etc.). Fine-tuned quantitative phenotyping could have the ability to resolve homogenous subgroups, greatly increasing statistical power and creating a better target for association.

Deep Learning has already led to state of the art results in a variety of fields including image processing, speech recognition, and gameplay. Many of the early "wins" using deep learning and more generally machine learning in the EHR involve applications of algorithms proven successful in other domains. This has been particularly true in unstructured EHR data such as images (cancer tumor detection etc.) and natural language processing for free text. This is sufficient when EHR learning tasks resemble tasks that are already popular among general machine learning researchers. Within biomedical informatics, this is the low hanging fruit. Long term advances will require specialized solutions designed specifically for the unique challenges presented by EHR-based research. Algorithmic development is only one part of the equation. The proper phrasing of problems and preprocessing of data will likely have as much if not more importance than algorithmic development.

Three of the pioneers of deep learning, Yann Lecun, Yoshua Bengio and Geoffrey Hinton wrote, "Unsupervised learning had a catalytic effect in reviving interest in deep learning, but has since been overshadowed by the successes of purely supervised

learning… we expect unsupervised learning to become far more important in the longer term" *(183)*. The challenge of collecting labeled data for supervised learning in the EHR may be an ideal environment for a reemergence of unsupervised and semi-supervised learning approaches. Early examples show that deep autoencoders are adept at this task *(91, 169)*.

For some clinical decisions, a black box algorithm with high accuracy is sufficient to improve medical care. An algorithm that can more accurately identify a tumor in imaging than a human has obvious benefits. For other problems, a black box is insufficient, it is unlikely to help researchers understand the physiology or etiology of a disease. In this setting, outside of a clinical decision, a less accurate but interpretable algorithm may be preferred. In addition, clinicians are likely to be skeptical and slower to adopt algorithms whose decisions cannot be rationally explained.

Furthermore, in April 2016, the European Union passed a data protection law entitled the "General Data Protection Regulation (GDPR) which will begin in 2018. The GDPR provides stricter conditions for sensitive data collection and storage, including, for example genetic and biometric data. It also sets regulation on privacy policies and further formalizes the "right to be forgotten." Of particular interest to the machine learning community is the language prohibiting decisions "based solely on automated processing and which produces adverse legal effects concerning, or significantly affects, him or her" and provides the right "to obtain an explanation of the decision reached after such assessment or to challenge the decision." It remains to be seen how this would be applied and if it will have any effect on the usage of artificial intelligence in the clinic but it demonstrates the fact that people want to understand how and why a decision affecting

their wellbeing is made. Work to create high performing algorithms that provide interpretable, explainable decisions is increasingly important as clinicians begin to rely on the aid of artificial intelligence.

Bibliography

1. C. Bishop, Pattern recognition, *Mach. Learn.* (2006) (available at http://www.academia.edu/download/30428242/bg0137.pdf).

2. Kreybe L., Histological lung cancer types. A morphological and biological correlation., *Acta Pathol Microbiol Scand Suppl.* , 157:1-92. (1962).

3. C. F. Mountain, Revisions in the International System for Staging Lung Cancer., *Chest* **111**, 1710–7 (1997).

4. L. West, S. J. Vidwans, N. P. Campbell, J. Shrager, G. R. Simon, R. Bueno, P. a. Dennis, G. a. Otterson, R. Salgia, A novel classification of lung cancer into molecular subtypes, *PLoS One* **7**, 1–11 (2012).

5. J.-A. Shin, J.-H. Lee, S.-Y. Lim, H.-S. Ha, H.-S. Kwon, Y.-M. Park, W.-C. Lee, M.-I. Kang, H.-W. Yim, K.-H. Yoon, H.-Y. Son, Metabolic syndrome as a predictor of type 2 diabetes, and its clinical interpretations and usefulness., *J. Diabetes Investig.* **4**, 334–43 (2013).

6. L. Li, W. Cheng, B. S. Glicksberg, O. Gottesman, R. Tamler, R. Chen, E. P. Bottinger, J. T. Dudley, Identification of type 2 diabetes subgroups through topological analysis of patient similarity, **7**, 1–16 (2015).

7. F. D. Lublin, S. C. Reingold, J. A. Cohen, G. R. Cutter, P. S. Sorensen, A. J. Thompson, J. S. Wolinsky, L. J. Balcer, B. Banwell, F. Barkhof, B. Bebo, P. A. Calabresi, M. Clanet, G. Comi, R. J. Fox, M. S. Freedman, A. D. Goodman, M. Inglese, L. Kappos, B. C. Kieseier, J. A. Lincoln, C. Lubetzki, A. E. Miller, X. Montalban, P. W.

O'Connor, J. Petkau, C. Pozzilli, R. A. Rudick, M. P. Sormani, O. Stuve, E. Waubant, C. H. Polman, Defining the clinical course of multiple sclerosis: The 2013 revisions, *Neurology* **83**, 278–286 (2014).

8. J. C. Denny, M. D. Ritchie, M. A. Basford, J. M. Pulley, L. Bastarache, K. Brown-Gentry, D. Wang, D. R. Masys, D. M. Roden, D. C. Crawford, PheWAS: Demonstrating the feasibility of a phenome-wide scan to discover gene-disease associations, *Bioinformatics* **26**, 1205–1210 (2010).

9. M. Manchia, J. Cullis, G. Turecki, G. A. Rouleau, R. Uher, M. Alda, The Impact of Phenotypic and Genetic Heterogeneity on Results of Genome Wide Association Studies of Complex Diseases, *PLoS One* **8**, 1–7 (2013).

10. D. Gordon, Y. Yang, C. Haynes, S. J. Finch, N. R. Mendell, A. M. Brown, V. Haroutunian, Increasing power for tests of genetic association in the presence of phenotype and/or genotype error by use of double-sampling., *Stat. Appl. Genet. Mol. Biol.* **3**, Article26 (2004).

11. S. Buyske, G. Yang, T. C. Matise, D. Gordon, When a Case Is Not a Case: Effects of Phenotype Misclassification on Power and Sample Size Requirements for the Transmission Disequilibrium Test with Affected Child Trios, *Hum. Hered.* **67**, 287–292 (2009).

12. A. Labbe, A. Bureau, I. Moreau, M.-A. Roy, Y. Chagnon, M. Maziade, C. Merette, Symptom dimensions as alternative phenotypes to address genetic heterogeneity in schizophrenia and bipolar disorder, *Eur. J. Hum. Genet.* **20**, 1182–1188 (2012).

13. P. Chaste, L. Klei, S. J. Sanders, V. Hus, M. T. Murtha, J. K. Lowe, A. J. Willsey, D. Moreno-De-Luca, T. W. Yu, E. Fombonne, D. Geschwind, D. E. Grice, D. H.

Ledbetter, S. M. Mane, D. M. Martin, E. M. Morrow, C. A. Walsh, J. S. Sutcliffe, C. Lese Martin, A. L. Beaudet, C. Lord, M. W. State, E. H. Cook, B. Devlin, A genome-wide association study of autism using the Simons Simplex Collection: Does reducing phenotypic heterogeneity in autism increase genetic homogeneity?, *Biol. Psychiatry* **77**, 775–84 (2015).

14. L. K. Wiley, J. P. Vanhouten, D. C. Samuels, M. C. Aldrich, D. M. Roden, J. F. Peterson, J. C. Denny, STRATEGIES FOR EQUITABLE PHARMACOGENOMIC-GUIDED WARFARIN DOSING AMONG EUROPEAN AND AFRICAN AMERICAN INDIVIDUALS IN A CLINICAL POPULATION., *Pac. Symp. Biocomput.* **22**, 545–556 (2016).

15. T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, W. Xie, G. L. Rosen, B. J. Lengerich, J. Israeli, Opportunities and obstacles for deep learning in biology and medicine, *bioRXiv* , 102 (2017).

16. B. K. Beaulieu-Jones, D. R. Lavage, J. W. Snyder, J. H. Moore, S. A. Pendergrass, C. R. Bauer, Characterizing and Managing Missing Structured Data in Electronic Health Records, *bioRxiv* (2017) (available at http://www.biorxiv.org/content/early/2017/07/24/167858).

17. J. Shaw, The erosion of privacy in the internet era, *Harv. Mag.* (2009) (available at https://www.cs.duke.edu/courses/common/compsci092/papers/anon/sweeney.pdf).

18. A. Narayanan, V. Shmatikov, Robust de-anonymization of large sparse datasets, *Privacy, 2008. SP 2008. IEEE ...* (2008) (available at http://ieeexplore.ieee.org/abstract/document/4531148/).

19. R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership Inference Attacks against Machine Learning Models, (2016) (available at http://arxiv.org/abs/1610.05820).

20. F. McSherry, K. Talwar, in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, (IEEE, 2007), pp. 94–103.

21. C. Dwork, A. Roth, The Algorithmic Foundations of Differential Privacy, *Found. Trends® Theor. Comput. Sci.* **9**, 211–407 (2013).

22. B. K. B. Beaulieu-Jones, C. C. S. Greene, Reproducibility of computational workflows is automated using continuous analysis, *Nat Biotech* **35**, 342–346 (2017).

23. S. R. Group, T. S. R. Group, A Randomized Trial of Intensive versus Standard Blood-Pressure Control, *N. Engl. J. Med.* **373**, 2103–2116 (2015).

24. A. B. Jensen, P. L. Moseley, T. I. Oprea, S. G. Ellesøe, R. Eriksson, H. Schmock, P. B. Jensen, L. J. Jensen, S. Brunak, O. Camilo, L. B. Goldstein, J. Finkelstein, E. Cha, S. M. Scharf, J. M. Teno, S. Weitzen, M. L. Fenell, V. Mor, F. E. M. Murtagh, E. Murphy, N. S. Sheerin, F. E. M. Murtagh, N. S. Sheerin, J. Addington-Hall, I. J. Higginson, S. A. Murray, M. Kendall, K. Boyd, A. Sheikh, H. Petri, D. Maldonato, N. J. Robinson, D. R. Blair, C. A. Hidalgo, N. Blumm, A.-L. Barabási, N. A. Christakis, L. L. Chen, N. Blumm, N. A. Christakis, A.-L. Barabási, T. S. Deisboeck, H. Tanushi, H. Dalianis, G. H. Nilsson, S. M. Curkendall, S. Sidney, A. C. Salisbury, K. J. Reid, J. A. Spertus, S. Suissa, S. Dell'Aniello, P. Ernst, S. E. Moss, R. Klein, B. E. K. Klein, T. Y. Wong, E. M. Kohner, D. S. Freedman, D. F. Williamson, E. W. Gunter, T. Byers, A. Kelkar, A. Kuo, W. H. Frishman, Q. Yang, B. M. Farr, C. L. Bartlett, J. Wadsworth, D. L. Miller, T. S. Ingebrigtsen, Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients, *Nat. Commun.* **5**, 1769–1775 (2014).

25. J. C. Denny, L. Bastarache, M. D. Ritchie, R. J. Carroll, R. Zink, J. D. Mosley, J. R. Field, J. M. Pulley, A. H. Ramirez, E. Bowton, M. a. Basford, D. S. Carrell, P. L. Peissig, A. N. Kho, J. a. Pacheco, L. V. Rasmussen, D. R. Crosslin, P. K. Crane, J. Pathak, S. J. Bielinski, S. a. Pendergrass, H. Xu, L. a. Hindorff, R. Li, T. a. Manolio, C. G. Chute, R. L. Chisholm, E. B. Larson, G. P. Jarvik, M. H. Brilliant, C. a. McCarty, I. J. Kullo, J. L. Haines, D. C. Crawford, D. R. Masys, D. M. Roden, Systematic comparison of phenome-wide association study of electronic medical record data and genome-wide association study data, *Nat. Biotechnol.* **31**, 1102–1110 (2013).

26. J. Tan, M. Ung, C. Cheng, C. S. Greene, Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders., *Pacific Symp. Biocomput.* **20**, 132–43 (2015).

27. 111th Congress (2009-2010), *H.R.1 - American Recovery and Reinvestment Act of 2009* (2009).

28. R. Steinbrook, Health Care and the American Recovery and Reinvestment Act, *N. Engl. J. Med.* **360**, 1057–1060 (2009).

29. S. J. Hebbring, S. J. Schrodi, Z. Ye, Z. Zhou, D. Page, M. H. Brilliant, A PheWAS approach in studying HLA-DRB1*1501., *Genes Immun.* **14**, 187–91 (2013).

30. J. C. Denny, D. C. Crawford, M. D. Ritchie, S. J. Bielinski, M. a. Basford, Y. Bradford, H. S. Chai, L. Bastarache, R. Zuvich, P. Peissig, D. Carrell, A. H. Ramirez, J. Pathak, R. a. Wilke, L. Rasmussen, X. Wang, J. a. Pacheco, A. N. Kho, M. G. Hayes, N. Weston, M. Matsumoto, P. a. Kopp, K. M. Newton, G. P. Jarvik, R. Li, T. a. Manolio, I. J. Kullo, C. G. Chute, R. L. Chisholm, E. B. Larson, C. a. McCarty, D. R. Masys, D. M. Roden, M. De Andrade, Variants near FOXE1 are associated with hypothyroidism and

other thyroid conditions: Using electronic medical records for genome- and phenome-wide studies, *Am. J. Hum. Genet.* **89**, 529–542 (2011).

31. J. S. Elkins, C. Friedman, B. Boden-Albala, R. L. Sacco, G. Hripcsak, Coding neuroradiology reports for the Northern Manhattan Stroke Study: a comparison of natural language processing and manual review., *Comput. Biomed. Res.* **33**, 1–10 (2000).

32. T. Zheng, W. Xie, L. Xu, X. He, Y. Zhang, M. You, G. Yang, Y. Chen, A Machine Learning-based Framework to Identify Type 2 Diabetes through Electronic Health Records, *bioRxiv* (2016).

33. A. L. Tyler, D. C. Crawford, S. a Pendergrass, The detection and characterization of pleiotropy: discovery, progress, and promise., *Brief. Bioinform.* , bbv050- (2015).

34. W. D. Travis, E. Brambilla, M. Noguchi, A. G. Nicholson, K. R. Geisinger, Y. Yatabe, D. G. Beer, C. A. Powell, G. J. Riely, P. E. Van Schil, K. Garg, J. H. Austin, H. Asamura, V. W. Rusch, F. R. Hirsch, G. Scagliotti, T. Mitsudomi, R. M. Huber, Y. Ishikawa, J. Jett, M. Sanchez-Cespedes, J. P. Sculier, T. Takahashi, M. Tsuboi, J. Vansteenkiste, I. Wistuba, P. C. Yang, D. Aberle, C. Brambilla, D. Flieder, W. Franklin, A. Gazdar, M. Gould, P. Hasleton, D. Henderson, B. Johnson, D. Johnson, K. Kerr, K. Kuriyama, J. S. Lee, V. A. Miller, I. Petersen, V. Roggli, R. Rosell, N. Saijo, E. Thunnissen, M. Tsao, D. Yankelewitz, International association for the study of lung cancer/american thoracic society/european respiratory society international multidisciplinary classification of lung adenocarcinoma, *J Thorac Oncol* **6**, 244–285 (2011).

35. A. S. Crystal, A. T. Shaw, L. V Sequist, L. Friboulet, M. J. Niederst, E. L. Lockerman, R. L. Frias, J. F. Gainor, A. Amzallag, P. Greninger, D. Lee, A. Kalsy, M.

Gomez-Caraballo, L. Elamine, E. Howe, W. Hur, E. Lifshits, H. E. Robinson, R. Katayama, A. C. Faber, M. M. Awad, S. Ramaswamy, M. Mino-Kenudson, A. J. Iafrate, C. H. Benes, J. A. Engelman, Patient-derived models of acquired resistance can identify effective drug combinations for cancer., *Science* **346**, 1480–6 (2014).

36. F. Ariani, G. Hayek, D. Rondinella, R. Artuso, M. A. Mencarelli, A. Spanhol-Rosseto, M. Pollazzon, S. Buoni, O. Spiga, S. Ricciardi, I. Meloni, I. Longo, F. Mari, V. Broccoli, M. Zappella, A. Renieri, FOXG1 Is Responsible for the Congenital Variant of Rett Syndrome, *Am. J. Hum. Genet.* **83**, 89–93 (2008).

37. A. Bureau, A. Labbe, J. Croteau,, C. Mérette, Using disease symptoms to improve detection of linkage under genetic heterogeneity, *Genet. Epidemiol.* **32**, 476–486 (2008).

38. M. Maziade, M. A. Roy, M. Martinez, D. Cliche, J. P. Fournier, Y. Garneau, L. Nicole, N. Montgrain, C. Dion, A. M. Ponton, et al., Negative, psychoticism, and disorganized dimensions in patients with familial schizophrenia or bipolar disorder: continuity and discontinuity between the major psychoses, *Am J Psychiatry* **152**, 1458–1463 (1995).

39. Z. Wang, A. D. Shah, A. R. Tate, S. Denaxas, J. Shawe-Taylor, H. Hemingway, Extracting Diagnoses and Investigation Results from Unstructured Text in Electronic Health Records by Semi-Supervised Machine Learning, *PLoS One* **7**, 1–9 (2012).

40. D. Dligach, T. Miller, G. K. Savova, Semi-supervised Learning for Phenotyping Tasks*AMIA Annu. Symp. Proc.* **2015**, 502–511 (2015).

41. B. K. Beaulieu-Jones, Denoising Autoencoders for Phenotype Stratification (DAPS): Preprint Release. Zenodo, (2016), doi:10.5281/zenodo.46165.

42. Docker, Docker (available at https://www.docker.com).

43. Shippable, Shippable (available at https://shippable.com).

44. P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, *Proc. 25th Int. Conf. Mach. Learn. - ICML '08* , 1096–1103 (2008).

45. Y. Bengio, Learning Deep Architectures for AI, *Found. Trends® Mach. Learn.* **2**, 1–127 (2009).

46. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper with Convolutions, *arXiv Prepr. arXiv1409.4842* , 1–12 (2014).

47. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the Inception Architecture for Computer Vision, (2015) (available at http://arxiv.org/abs/1512.00567).

48. G. Hinton, L. Deng, D. Yu, G. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal Process. Mag.* **29**, 82–97 (2012).

49. Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, A Neural Probabilistic Language Model, *J. Mach. Learn. Res.* **3**, 1137–1155 (2003).

50. A. Graves, A. Mohamed, G. Hinton, Speech Recognition With Deep Recurrent Neural Networks, *Icassp* , 6645–6649 (2013).

51. J. Zhou, O. G. Troyanskaya, Predicting effects of noncoding variants with deep learning–based sequence model, *Nat. Methods* **12**, 931–934 (2015).

52. Y. Park, M. Kellis, news and views Deep learning for regulatory genomics, *Nat. Publ. Gr.* **33**, 825–826 (2015).

53. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, *J. Mach. Learn. Res.* **11**, 3371–3408 (2010).

54. F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, Y. Bengio, Theano: new features and speed improvements, *arXiv Prepr. arXiv …* , 1–10 (2012).

55. J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, in *9th Python in Science Conference*, (2010), pp. 1–7.

56. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, Scikit-learn: Machine Learning in Python, *... Mach. Learn. ...* **12**, 2825–2830 (2012).

57. F. Civeira, Guidelines for the diagnosis and management of heterozygous familial hypercholesterolemia., *Atherosclerosis* **173**, 55–68 (2004).

58. L. Van Der Maaten, G. Hinton, Visualizing Data using t-SNE, *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).

59. G. E. Hinton, S. T. Roweis, Stochastic neighbor embedding, *Adv. Neural Inf. Process. Syst.* , 833–840 (2002).

60. F. V. Nelwamondo, S. Mohamed, T. Marwala, Missing Data: A Comparison of Neural Network and Expectation Maximisation Techniques, (2007) (available at

http://arxiv.org/abs/0704.3474).

61. J. Xie, L. Xu, E. Chen, Image Denoising and Inpainting with Deep Neural Networks, .

62. L. Flintoft, Disease genetics: phenome-wide association studies go large, *Nat. Rev. Genet.* (2014) (available at https://www.nature.com/nrg/journal/v15/n1/full/nrg3637.html).

63. B. B. J. Wells, K. K. M. Chagin, A. S. A. Nowacki, M. M. W. Kattan, Strategies for handling missing data in electronic health record derived data., *EGEMS (Washington, DC)* **1**, 1035 (2013).

64. M. Bounthavong, J. J. H. Watanabe, K. M. Sullivan, Approach to addressing missing data for electronic medical records and pharmacy claims data research, *Pharmacotherapy* **35**, 380–387 (2015).

65. K. Bhaskaran, L. Smeeth, What is the difference between missing completely at random and missing at random?, *Int. J. Epidemiol.* **43**, 1336–1339 (2014).

66. D. B. Rubin, Inference and Missing Data, *Biometrika* **63**, 581–592 (1976).

67. R. Jörnsten, M. Ouyang, A meta-data based method for DNA microarray imputation, *BMC* (2007) (available at https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-109).

68. B. K. Beaulieu-Jones, J. H. Moore, MISSING DATA IMPUTATION IN THE ELECTRONIC HEALTH RECORD USING DEEPLY LEARNED AUTOENCODERS., *Pac. Symp. Biocomput.* **22**, 207–218 (2016).

69. P. Allison, Missing Data: Sage University Papers Series on Quantitative Applications in the Social Sciences (07–136), *Thousand Oaks, CA* (2001).

70. B. Beaulieu-Jones, C. Greene, Reproducibility of computational workflows is automated using continuous analysis, *Nat. Biotechnol.* (2017) (available at https://www.nature.com/nbt/journal/v35/n4/abs/nbt.3780.html).

71. F. Pedregosa, G. Varoquaux, A. Gramfort, Scikit-learn: Machine learning in Python, *J. Mach.* (2011) (available at http://www.jmlr.org/papers/v12/pedregosa11a.html).

72. A. Rubinsteyn, S. Feldman, fancyimpute: Version 0.0.9, (2016), doi:10.5281/zenodo.47151.

73. S. Buuren, K. Groothuis-Oudshoorn, mice: Multivariate imputation by chained equations in R, *J. Stat. Softw.* (2011) (available at http://doc.utwente.nl/78938/).

74. Helfand, Screening for Lipid Disorders in Adults: Selective Update of 2001 US Preventive Services Task Force Review, *Evid. Synth.* **49** (2008).

75. L. Beretta, A. Santaniello, Nearest neighbor imputation algorithms: a critical evaluation, *BMC Med.* (2016) (available at https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-016-0318-z).

76. O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, R. B. Altman, Missing value estimation methods for DNA microarrays, *Bioinformatics* **17**, 520–525 (2001).

77. V. Pestov, Is the k-NN classifier in high dimensions affected by the curse of dimensionality?, (2011) (available at http://arxiv.org/abs/1110.4347).

78. E. A. Stuart, M. Azur, C. Frangakis, P. Leaf, Multiple imputation with large data sets: A case study of the children's mental health initiative, *Am. J. Epidemiol.* **169**, 1133–

1139 (2009).

79. I. R. White, P. Royston, A. M. Wood, Multiple imputation using chained equations: Issues and guidance for practice, *Stat. Med.* **30**, 377–399 (2011).

80. T. E. Bodner, What Improves with Increased Missing Data Imputations?, *Struct. Equ. Model. A Multidiscip. J.* **15**, 651–675 (2008).

81. R. Küffner, N. Zach, R. Norel, J. Hawe, D. Schoenfeld, L. Wang, G. Li, L. Fang, L. Mackey, O. Hardiman, M. Cudkowicz, A. Sherman, G. Ertaylan, M. Grosse-Wentrup, T. Hothorn, J. van Ligtenberg, J. H. Macke, T. Meyer, B. Schölkopf, L. Tran, R. Vaughan, G. Stolovitzky, M. L. Leitner, Crowdsourced analysis of clinical trial data to predict amyotrophic lateral sclerosis progression., *Nat. Biotechnol.* **33**, 51–7 (2015).

82. K. Kollewe, U. Mauss, K. Krampfl, S. Petri, R. Dengler, B. Mohammadi, ALSFRS-R score and its ratio: A useful predictor for ALS-progression, *J. Neurol. Sci.* **275**, 69–73 (2008).

83. E. Beghi, T. Mennini, C. Bendotti, P. Bigini, G. Logroscino, A. Chiò, O. Hardiman, D. Mitchell, R. Swingler, B. J. Traynor, A. Al-Chalabi, The heterogeneity of amyotrophic lateral sclerosis: a possible explanation of treatment failure., *Curr. Med. Chem.* **14**, 3185–200 (2007).

84. M. Sabatelli, A. Conte, M. Zollino, Clinical and genetic heterogeneity of amyotrophic lateral sclerosis., *Clin. Genet.* **83**, 408–16 (2013).

85. J. M. Ravits, A. R. La Spada, ALS motor phenotype heterogeneity, focality, and spread: deconstructing motor neuron degeneration., *Neurology* **73**, 805–11 (2009).

86. J. M. Cedarbaum, N. Stambler, in *Journal of the Neurological Sciences*, (1997), vol. 152.

87. J. M. Cedarbaum, N. Stambler, E. Malta, C. Fuller, D. Hilt, B. Thurmond, A. Nakanishi, The ALSFRS-R: A revised ALS functional rating scale that incorporates assessments of respiratory function, *J. Neurol. Sci.* **169**, 13–21 (1999).

88. B. K. Beaulieu-Jones, C. S. Greene, Semi-Supervised Learning of the Electronic Health Record with Denoising Autoencoders for Phenotype Stratification, *bioRxiv* , 39800 (2016).

89. F. Chollet, *Keras* (GitHub, 2015; http://203.195.193.174/nat123CacheFolder/646F63732E626470742E6E6574/35c3a8a4cb1d4160bfd7b6e93d74ab67CD30CE37D036D032DF31CE3ACC30C533C9_e22880a46b0f1f3f3eb1e14dd5452984/media/pdf/kerascn/latest/kerascn.pdf#page=59).

90. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout : A Simple Way to Prevent Neural Networks from Overfitting, *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).

91. R. Miotto, L. Li, B. A. Kidd, J. T. Dudley, Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records, *Sci. Rep.* **6**, 26094 (2016).

92. R. Socher, J. Pennington, E. Huang, A. Ng, Semi-supervised recursive autoencoders for predicting sentiment distributions, *Proc.* (2011) (available at http://dl.acm.org/citation.cfm?id=2145450).

93. G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science (80-.).* (2006) (available at http://science.sciencemag.org/content/313/5786/504.short).

94. R. Mazumder, T. Hastie, H. Edu, R. Tibshirani, T. Edu, Spectral Regularization

Algorithms for Learning Large Incomplete Matrices, *J. Mach. Learn. Res.* **11**, 2287–2322 (2010).

95. S. van Buuren, *Flexible imputation of missing data* (2012).

96. P. Royston, Multiple imputation of missing values: update of ice, *Stata J.* (2005) (available at https://www.researchgate.net/profile/James_Cui2/publication/23780230_Buckley-James_method_for_analyzing_censored_data_with_an_application_to_a_cardiovascular_disease_and_an_HIVAIDS_study/links/53d5866d0cf228d363ea0b7a.pdf#page=59).

97. J. Kim, H. Park, Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons, .

98. C.-J. Lin, Projected Gradient Methods for Non-negative Matrix Factorization, .

99. C.-J. Hsieh, P. A. Olsen, Nuclear Norm Minimization via Active Subspace Selection, .

100. L. Breiman, A. Cutler, Random Forests. 2004, *URL http//stat-www. berkeley. edu/users/breiman/RandomForests/cc home. htm* (2014).

101. S. Zoccolella, E. Beghi, G. Palagano, A. Fraddosio, V. Guerra, V. Samarelli, V. Lepore, I. L. Simone, P. Lamberti, L. Serlenga, G. Logroscino, SLAP registry, Riluzole and amyotrophic lateral sclerosis survival: a population-based study in southern Italy., *Eur. J. Neurol.* **14**, 262–8 (2007).

102. B. J. Traynor, M. Alexander, B. Corr, E. Frost, O. Hardiman, An outcome study of riluzole in amyotrophic lateral sclerosis, *J. Neurol.* **250**, 473–479 (2003).

103. A. Czaplinski, A. A. Yen, S. H. Appel, Forced vital capacity (FVC) as an indicator of survival and disease progression in an ALS clinic population., *J. Neurol.*

Neurosurg. Psychiatry **77**, 390–2 (2006).

104. M. Baker, 1,500 scientists lift the lid on reproducibility, *Nature* **533**, 452–454 (2016).

105. Rebooting review, *Nat Biotech* **33**, 319 (2015).

106. Software with impact, *Nat Meth* **11**, 211 (2014).

107. R. D. Peng, Reproducible Research in Computational Science, *Science (80-.).* **334**, 1226–1227 (2011).

108. M. McNutt, Reproducibility, *Science (80-.).* **343**, 229 (2014).

109. Illuminating the black box, *Nature* **442**, 1 (2006).

110. D. Garijo, S. Kinnings, L. L. Xie, L. L. Xie, Y. Zhang, P. E. Bourne, Y. Gil, Quantifying reproducibility in computational biology: The case of the tuberculosis drugome, *PLoS One* **8** (2013), doi:10.1371/journal.pone.0080278.

111. S. L. Kinnings, L. L. Xie, K. H. Fung, R. M. Jackson, L. L. Xie, P. E. Bourne, The Mycobacterium tuberculosis drugome and its polypharmacological implications, *PLoS Comput. Biol.* **6** (2010), doi:10.1371/journal.pcbi.1000976.

112. J. P. A. Ioannidis, D. B. Allison, C. A. Ball, I. Coulibaly, X. Cui, A. C. Culhane, M. Falchi, C. Furlanello, L. Game, G. Jurman, J. Mangion, T. Mehta, M. Nitzberg, G. P. Page, E. Petretto, V. Van Noort, Repeatability of published microarray gene expression analyses, *Nat. Genet.* **41**, 149–155 (2009).

113. T. Hothorn, F. Leisch, Case studies in reproducibility, *Brief. Bioinform.* **12**, 288–300 (2011).

114. T. Groves, F. Godlee, Open science and reproducible research, *BMJ* **344** (2012), doi:10.1136/bmj.e4383.

115. ThinkLab (available at https://thinklab.com/).

116. C. Boettiger, An introduction to Docker for reproducible research, with examples from the R environment, *ACM SIGOPS Oper. Syst. Rev. Spec. Issue Repeatability Shar. Exp. Artifacts* **49**, 71–79 (2015).

117. M. Dai, P. Wang, A. D. Boyd, G. Kostov, B. Athey, E. G. Jones, W. E. Bunney, R. M. Myers, T. P. Speed, H. Akil, S. J. Watson, F. Meng, Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data., *Nucleic Acids Res.* **33**, e175 (2005).

118. M. Nunez, C. Sanchez-Jimenez, J. Alcalde, J. M. Izquierdo, Long-term reduction of T-cell intracellular antigens reveals a transcriptome associated with extracellular matrix and cell adhesion components, *PLoS One* **9** (2014), doi:10.1371/journal.pone.0113141.

119. B. Beaulieu-Jones, C. Greene, Continuous Analysis BrainArray: Submission Release Continuous Analysis BrainArray: Submission Release (2016), doi:10.5281/zenodo.59892.

120. P. Duvall, S. Matyas, A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk* (2007; http://portal.acm.org/citation.cfm?id=1406212).

121. F. Pérez, B. E. Granger, {IP}ython: a System for Interactive Scientific Computing, *Comput. Sci. Eng.* **9**, 21–29 (2007).

122. Jupyter (2016) (available at http://jupyter.org/).

123. RStudio, RStudio: Integrated development environment for R (Version 0.97.311)*J. Wildl. Manage.* **75**, 1753–1766 (2011).

124. B. Baumer, M. Cetinkaya-Rundel, A. Bray, L. Loi, N. J. Horton, R Markdown:

Integrating A Reproducible Analysis Tool into Introductory Statistics, *Technol. Innov. Stat. Educ.* **8**, 20 (2014).

125. Friedrich Leisch, Sweave: Dynamic generation of statistical reports using literate data analysis, *Compstat 2002 - Proc. Comput. Stat.* , 575–580 (2002).

126. B. K. Beaulieu-Jones, C. S. Greene, Continuous Analysis Example Docker Images, (2016) (available at 10.6084/m9.figshare.3545156.v1).

127. B. K. Beaulieu-Jones, A. Whan, C. S. Greene, greenelab/continuous_analysis: Continuous Analysis v1.0 [Data set]*Zenodo* (2016) (available at http://doi.org/10.5281/zenodo.178613).

128. Drone.io (available at https://drone.io/).

129. K. Katoh, K. Misawa, K. Kuma, T. Miyata, MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform, *Nucleic Acids Res.* **30**, 3059–3066 (2002).

130. D. Plotree, D. Plotgram, PHYLIP-phylogeny inference package (version 3.2), *cladistics* (1989) (available at http://onlinelibrary.wiley.com/doi/10.1111/j.1096-0031.1989.tb00562.x/abstract).

131. S. F. Boj, C.-I. Hwang, L. A. Baker, I. I. C. Chio, D. D. Engle, V. Corbo, M. Jager, M. Ponz-Sarvise, H. Tiriac, M. S. Spector, A. Gracanin, T. Oni, K. H. Yu, R. van Boxtel, M. Huch, K. D. Rivera, J. P. Wilson, M. E. Feigin, D. Öhlund, A. Handly-Santana, C. M. Ardito-Abraham, M. Ludwig, E. Elyada, B. Alagesan, G. Biffi, G. N. Yordanov, B. Delcuze, B. Creighton, K. Wright, Y. Park, F. H. M. Morsink, I. Q. Molenaar, I. H. Borel Rinkes, E. Cuppen, Y. Hao, Y. Jin, I. J. Nijman, C. Iacobuzio-Donahue, S. D. Leach, D. J. Pappin, M. Hammell, D. S. Klimstra, O. Basturk, R. H.

Hruban, G. J. Offerhaus, R. G. J. Vries, H. Clevers, D. A. Tuveson, Organoid Models of Human and Mouse Ductal Pancreatic Cancer, *Cell* **160**, 324–338 (2015).

132. D. Balli, Using Kallisto for expression analysis of published RNAseq data (2015) (available at https://benchtobioinformatics.wordpress.com/2015/07/10/using-kallisto-for-gene-expression-analysis-of-published-rnaseq-data/).

133. N. L. Bray, H. Pimentel, P. Melsted, L. Pachter, Near-optimal probabilistic RNA-seq quantification, *Nat. Biotechnol.* **34**, 525–527 (2016).

134. M. E. Ritchie, B. Phipson, D. Wu, Y. Hu, C. W. Law, W. Shi, G. K. Smyth, limma powers differential expression analyses for RNA-sequencing and microarray studies., *Nucleic Acids Res.* **43**, e47 (2015).

135. G. K. Smyth, Linear models and empirical bayes methods for assessing differential expression in microarray experiments., *Stat. Appl. Genet. Mol. Biol.* **3**, Article3 (2004).

136. H. J. Pimentel, N. Bray, S. Puente, P. Melsted, L. Pachter, Differential analysis of RNA-Seq incorporating quantification uncertainty, *bioRxiv* (2016), doi:10.1101/058164.

137. Y. Souilmi, A. K. A. Lancaster, J.-Y. J. Jung, E. Rizzo, J. B. J. Hawkins, R. Powles, S. Amzazi, H. Ghazal, P. J. P. Tonellato, D. D. P. Wall, M. Kircher, J. Kelso, M. Schatz, B. Langmead, A. Desai, A. Jere, A. Sboner, X. Mu, D. Greenbaum, R. Auerbach, M. Gerstein, F. Collins, M. Hamburg, E. Gafni, L. Luquette, A. K. A. Lancaster, J. B. J. Hawkins, J.-Y. J. Jung, Y. Souilmi, M. Abouelhoda, S. Issa, M. Ghanem, K. Karczewski, G. Fernald, A. Martin, M. Snyder, N. Tatonetti, J. Dudley, J. Goecks, A. Nekrutenko, J. Taylor, T. Galaxy, A. Nekrutenko, J. Taylor, V. Fusaro, P. Patil, E. Gafni, D. D. P. Wall,

P. J. P. Tonellato, G. Auwera, M. Carneiro, C. Hartl, R. Poplin, G. Angel, A. Levy-Moonshine, A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, M. DePristo, E. Banks, R. Poplin, K. Garimella, J. Maguire, C. Hartl, J. Dean, S. Ghemawat, H. Li, R. Durbin, K. Wang, M. Li, H. Hakonarson, H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, T. Yu, M. Chahrour, M. Coulter, S. Jiralerspong, K. Okamura-Ikeda, B. Ataman, J. Zook, B. Chapman, J. Wang, D. Mittelman, O. Hofmann, W. Hide, G. Abecasis, D. Altshuler, A. Auton, L. Brooks, R. Durbin, M. Fischer, R. Snajder, S. Pabinger, A. Dander, A. Schossig, J. Zschocke, J. Reid, A. Carroll, N. Veeraraghavan, M. Dahdouli, A. Sundquist, A. English, S. Zhao, K. Prenger, L. Smith, T. Messina, H. Fan, E. Jaeger, B. Kelly, J. Fitch, Y. Hu, D. Corsmeier, H. Zhong, A. Wetzel, Scalable and cost-effective NGS genotyping in the cloud, *BMC Med. Genomics* **8**, 64 (2015).

138. V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. A. Ioannidis, M. Taufer, Enhancing reproducibility for computational methods, *Science (80-.).* **354** (2016).

139. The SPRINT Data Analysis Challenge 2017 (available at https://challenge.nejm.org/pages/home).

140. S. R. Group, A randomized trial of intensive versus standard blood-pressure control, *N Engl J Med* (2015) (available at https://www.nejm.org/doi/full/10.1056/NEJMoa1511939).

141. S. Basu, J. B. Sussman, J. Rigdon, L. Steimle, B. Denton, R. Hayward, Development and Validation of a Clinical Decision Score to Maximize Benefit and Minimize Harm from Intensive Blood Pressure Treatment (2017) (available at

https://challenge.nejm.org/posts/5815).

142. N. Dagan, M. A. Tsadok, M. Hoshen, A. Arkiv, T. Karpati, I. Gofer, M. Leibowitz, H. Gilutz, E. Podjarny, E. Bachmat, R. Balicer, To Treat Intensively or Not – Individualized Decision Making Support Tool (2017) (available at https://challenge.nejm.org/posts/5826).

143. R. Aggarwal, J. Steinkamp, N. Chiu, M. H. Sang, J. Park, H. Mirzan, B. Petrie, Assessing the Impact of Intensive Blood Pressure Management in Chronic Kidney Disease Patients (2017) (available at https://challenge.nejm.org/posts/5837).

144. Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature* (2015) (available at http://www.nature.com/nature/journal/v521/n7553/abs/nature14539.html).

145. T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, W. Xie, G. L. Rosen, B. J. Lengerich, J. Israeli, J. Lanchantin, S. Woloszynek, A. E. Carpenter, A. Shrikumar, J. Xu, E. M. Cofer, D. J. Harris, D. DeCaprio, Y. Qi, A. Kundaje, Y. Peng, L. K. Wiley, M. H. S. Segler, A. Gitter, C. S. Greene, Opportunities And Obstacles For Deep Learning In Biology And Medicine, *bioRxiv* (2017) (available at https://doi.org/10.1101/142760).

146. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Nets, 2672–2680 (2014).

147. A. Odena, C. Olah, J. Shlens, Conditional Image Synthesis With Auxiliary Classifier GANs, (2016) (available at http://arxiv.org/abs/1610.09585).

148. E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, J. Sun, Generating Multi-label Discrete Electronic Health Records using Generative Adversarial Networks, (2017) (available at http://arxiv.org/abs/1703.06490).

149. C. Esteban, S. L. Hyland, G. Rätsch, Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs, (2017) (available at http://arxiv.org/abs/1706.02633).

150. S. L. Garfinkel, De-Identification of Personal Information, , doi:10.6028/NIST.IR.8053.

151. K. El Emam, E. Jonker, L. Arbuckle, B. Malin, J. Riedl, R. W. Scherer, Ed. A Systematic Review of Re-Identification Attacks on Health Data, *PLoS One* **6**, e28071 (2011).

152. B. Malin, L. Sweeney, How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems, *J. Biomed. Inform.* **37**, 179–192 (2004).

153. I. S. Kohane, R. B. Altman, Health-Information Altruists ? A Potentially Critical Resource, *N. Engl. J. Med.* **353**, 2074–2077 (2005).

154. L. Sweeney, k-anonymity: A model for protecting privacy, *Int. J. Uncertainty, Fuzziness* (2002) (available at http://www.worldscientific.com/doi/abs/10.1142/S0218488502001648).

155. L. Sweeney, A. Abu, J. Winn, Identifying participants in the personal genome project by name, (2013) (available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2257732).

156. M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, *Proc. 22nd ACM* (2015) (available at http://dl.acm.org/citation.cfm?id=2813677).

157. M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, Privacy in Pharmacogenetics:

An End-to-End Case Study of Personalized Warfarin Dosing., *USENIX* (2014) (available at https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-fredrikson-privacy.pdf).

158. N. Homer, S. Szelinger, M. Redman, D. Duggan, Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays, *PLoS* (2008) (available at http://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1000167).

159. S. Simmons, B. Berger, Realizing privacy preserving genome-wide association studies, *Bioinformatics* (2016) (available at http://bioinformatics.oxfordjournals.org/content/32/9/1293.short).

160. M. Abadi, A. Chu, I. Goodfellow, H. McMahan, Deep learning with differential privacy, *Proc.* (2016) (available at http://dl.acm.org/citation.cfm?id=2978318).

161. R. Shokri, V. Shmatikov, Privacy-preserving deep learning, *Proc. 22nd ACM SIGSAC* (2015) (available at http://dl.acm.org/citation.cfm?id=2813687).

162. C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, A. Roth, The reusable holdout: Preserving validity in adaptive data analysis, *Science (80-.).* **349** (2015) (available at http://science.sciencemag.org/content/349/6248/636).

163. E. Jang, S. Gu, B. Poole, Categorical Reparameterization with Gumbel-Softmax, (2016) (available at http://arxiv.org/abs/1611.01144).

164. M. J. Kusner, J. M. Hernández-Lobato, GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution, (2016) (available at http://arxiv.org/abs/1611.04051).

165. D. Cynthia, Differential privacy, *Autom. Lang. Program.* (2006).

166. C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, *Theory Cryptogr.* (2006) (available at http://link.springer.com/chapter/10.1007/11681878_14).

167. J. Henry, Y. Pylypchuk, T. Searcy, V. Patel, Adoption of Electronic Health Record Systems among US Non-Federal Acute Care Hospitals: 2008-2015, *Coord. Heal. ...* (2016) (available at https://www.healthit.gov/sites/default/files/briefs/2015_hospital_adoption_db_v17.pdf).

168. T. A. Lasko, J. C. Denny, M. A. Levy, J. Devaney, Ed. Computational Phenotype Discovery Using Unsupervised Feature Learning over Noisy, Sparse, and Irregular Clinical Data, *PLoS One* **8**, e66341 (2013).

169. B. K. B. K. Beaulieu-Jones, C. S. Greene, Semi-supervised learning of the electronic health record for phenotype stratification, *J. Biomed. Inform.* **64**, 168–178 (2016).

170. B. K. Beaulieu-Jones, J. H. Moore, MISSING DATA IMPUTATION IN THE ELECTRONIC HEALTH RECORD USING DEEPLY LEARNED AUTOENCODERS, *Pac. Symp. Biocomput.* **22** (2016).

171. J. Ba, V. Mnih, K. Kavukcuoglu, Multiple Object Recognition with Visual Attention, (2014) (available at http://arxiv.org/abs/1412.7755).

172. K. Gregor, I. Danihelka, A. Graves, D. Jimenez Rezende, D. Wierstra, DRAW: A Recurrent Neural Network For Image Generation, (available at https://arxiv.org/pdf/1502.04623.pdf).

173. I. Sutskever, J. Martens, G. Hinton, Generating Text with Recurrent Neural Networks, (available at http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf).

174. S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, *Neural Comput.* **9**, 1735–1780 (1997).

175. Y. Choi, C. Y.-I. Chiu, D. Sontag, Learning Low-Dimensional Representations of Medical Concepts., *AMIA Jt. Summits Transl. Sci. proceedings. AMIA Jt. Summits Transl. Sci.* **2016**, 41–50 (2016).

176. Z. C. Lipton, D. C. Kale, R. C. Wetzel, Phenotyping of Clinical Time Series with LSTM Recurrent Neural Networks, (available at https://arxiv.org/pdf/1510.07641.pdf).

177. Z. C. Lipton, D. C. Kale, C. Elkan, R. Wetzel, Learning to Diagnose with LSTM Recurrent Neural Networks, (2015) (available at http://arxiv.org/abs/1511.03677).

178. E. Dudek-Dyduch, Algebraic logical meta-model of decision processes-new metaheuristics, *Int. Conf. Artif. Intell.* (2015) (available at http://link.springer.com/chapter/10.1007/978-3-319-19324-3_48).

179. A. Johnson, T. Pollard, L. Shen, L. Lehman, MIMIC-III, a freely accessible critical care database, *Scientific* (2016) (available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4878278/).

180. D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, (2014) (available at http://arxiv.org/abs/1412.6980).

181. T. Tieleman, G. E. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, 26–31 (2012).

182. P. Orzechowski, K. Boryczko, Parallel approach for visual clustering of protein databases, *Comput. Informatics* (2012) (available at http://www.cai.sk/ojs/index.php/cai/article/viewArticle/140).

183. Y. LeCun, Y. Bengio, G. Hinton, L. Y., B. Y., H. G., Deep learning, *Nature* **521**, 436–444 (2015).