Cross Tissue DNAm Biomarker Prediction using Transfer Learning

KRISTEN M MCGREEVY, Department of Biostatistics, University of California, Los Angeles, USA
BRIAN H CHEN, San Francisco Coordinating Center, California Pacific Medical Center Research Institute, San Francisco, California, USA; Department of Epidemiology and Biostatistics, University of California San Francisco, San Francisco, California, USA; The Herbert Wertheim School of Public Health Human Longevity Science, USA
STEVE HORVATH, Altos Labs, UK
DONATELLO TELESCA, Department of Biostatistics, University of California, Los Angeles, USA

DNA methylation (DNAm) is an epigenetic mechanism vital for regulating gene expression and influencing disease states. Developing accurate DNAm biomarkers often requires data from specific tissues, which are sometimes difficult to access. This study explores the use of Transfer Learning (TL) to predict blood DNAm biomarkers using saliva DNAm data, aiming to overcome limitations posed by sample size and tissue accessibility. We developed TL-based algorithms that integrate DNAm data from multiple tissues. These algorithms were evaluated against traditional Lasso regression and direct saliva DNAm estimates. Our results show that TL significantly improves the prediction accuracy of DNAm biomarkers, outperforming traditional methods in 20 out of 26 biomarkers. We further validated our models using independent datasets, demonstrating that TL-derived predictions reflect known biological relationships, such as sex differences in telomere length and the impact of smoking on DNAm biomarkers. Our findings highlight the potential of TL in enhancing DNAm biomarker prediction across tissues, providing a valuable tool for epigenetic research. The developed algorithms and methodologies are accessible to researchers, fostering advancements in personalized medicine and aging research. This study establishes a framework for utilizing TL to bridge the gap between accessible and pertinent tissue data, paving the way for more accurate and versatile DNAm biomarker applications.

Additional Key Words and Phrases: DNA methylation, DNAm Biomarkers, cross tissue prediction, Transfer Learning

ACM Reference Format:
Kristen M McGreevy, Brian H Chen, Steve Horvath, and Donatello Telesca. 2024. Cross Tissue DNAm Biomarker Prediction using Transfer Learning. 1, 1 (June 2024), 43 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Kristen M McGreevy, Department of Biostatistics, University of California, Los Angeles, USA, kristenmae@ucla.edu; Brian H Chen, San Francisco Coordinating Center, California Pacific Medical Center Research Institute, San Francisco, California, USA; Department of Epidemiology and Biostatistics, University of California San Francisco, San Francisco, California, USA; The Herbert Wertheim School of Public Health Human Longevity Science, USA, B1chen@ucsd.edu; Steve Horvath, Altos Labs, UK, shorvath@altoslabs.com; Donatello Telesca, Department of Biostatistics, University of California, Los Angeles, USA, donatello.telesca@gmail.com.

ACM XXXX-XXXX/2024/6-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

DNA methylation (DNAm) is an epigenetic mechanism that varies across tissues and contributes to cell type, regulates gene expression, and influences disease states [1]. Beyond this, DNAm has been leveraged for developing surrogate estimates, known as DNAm biomarkers, for a variety of phenotypes including age [2, 3], mortality risk [4], fitness [5], blood cell counts [6], and others [7–9]. DNAm continues to be studied offering a window into the biological processes and aging within cells [10]. Traditionally, blood has been the tissue of choice for DNAm biomarker development, serving as a versatile medium that interacts with and carries information from an array of organs. However, this choice is not without its limitations. The performance of DNAm based biomarkers is inherently tied to the relevance of the tissue that DNAm is measured in [11, 12], where biomarkers built with tissues more related to the condition or trait of interest are likely to be more accurate and informative.

Yet, herein lies a significant challenge—the tissues most pertinent to certain diseases or conditions, such as the brain, adipose, and bone, are often the hardest to access. Their collection is typically invasive, painful, and expensive, leading to small sample sizes. This scarcity significantly limits the development of tissue-specific DNAm biomarkers. While some research has looked at estimating methylation across tissues [13], algorithms that do this without having access to multiple tissue data are still unavailable. While some research has looked at estimating methylation across tissues [13] or predicting species' average methylation across tissues [14], algorithms are not available for individual level prediction yet or without needing access to multiple tissue data. Moreover, simply measuring these tissues and applying current DNAm biomarkers may not yield meaningful insights, a point underscored by researchers who advise caution when using methylation markers from surrogate tissues [15].

This landscape delineates two explicit needs in the field of DNAm research. First, there is an urgent requirement for using more accessible tissues to accurately measure biomarkers of interest [11, 16]. Saliva emerges as a promising candidate, offering a non-invasive and easily accessible alternative to blood [17]. Its collection is patient-friendly and uncomplicated, allowing for larger sample sizes and broader applications. Second, there is a pressing need for novel methodologies that can develop accurate biomarkers in tissues traditionally challenged by inadequate sample sizes [16, 18].

Moreover, this context highlights an area for innovation: the use and understanding of Transfer Learning (TL) in the realm of DNAm research and biomarker development. TL, a powerful technique in machine learning, is adept at leveraging information from related data sources and applying knowledge from one context to enhance precision in another [19]. TL has been used in bioinformatics for tasks like imputing the methylome in low-coverage cases [20] and augmenting gene expression data [21], but its application in DNAm biomarkers, particularly for cross-tissue prediction, has yet to be explored.

Our study aims to address these needs from three angles. We demonstrate how transfer learning can improve DNAm biomarker accuracy, especially with limited data sources. We introduce cross tissue prediction algorithms for estimating common blood DNAm biomarkers using saliva DNAm. Additionally, our study offers practical tools and guidelines for researchers, empowering them to implement TL methods and develop algorithms tailored to their specific biomarker interests. Through these contributions, we aspire to provide easily usable methods for researchers to better estimate their favorite biomarkers across different tissues by leveraging shared information from other tissues, as well as methods to develop more accurate biomarkers in typically inaccessible tissues by combining data from similar sources.

Here, we adapt TL methods and develop cross tissue prediction algorithms. We focus on validating these algorithms with three key benchmarks deemed critical for any robust cross-tissue prediction algorithm. First, we assess whether TL offers an advantage in predicting blood DNAm biomarkers from saliva DNAm by comparing accuracy to direct saliva DNAm estimates. Second, we compare the TL algorithms against conventional Lasso regression to evaluate the benefits of adopting advanced computational techniques for developing cross tissue prediction algorithms. Third, we evaluate the ability of our algorithms to extend beyond mere accuracy and reflect biologically meaningful and established relationships in a variety of validation datasets.

2 METHODS

The core aim of our study was to assess the utility of Transfer Learning (TL) in the context of DNA methylation (DNAm) and DNAm-based biomarkers. Our analytical strategy is predominantly built upon the translasso framework [21]. In our research, we prioritized this class of algorithms due to their compatibility with the widely utilized Lasso regression techniques [22]. This choice ensures that our methods can be seamlessly integrated by researchers accustomed to Lasso regression, thereby promoting wider adoption within the field. Lasso, as opposed to elastic net regression, was chosen for simplicity and future methods can extend to other penalized regression frameworks. We employ a transfer learning paradigm for high-dimensional linear regression, utilizing prediction and estimation procedures outlined in the translasso study [21] to investigate whether transfer learning can be utilized in high dimensional epigenetics to improve DNAm biomarker prediction across tissues.

Our methods section has the following format. First, we describe the basic model set-up and datasets used. Then, we describe the transfer learning methodology that we employ. After that, we describe the variations to this methodology that we explore in order to optimize parameterization and TL application to our cross tissue DNAm setting. We then describe how we classify optimizations and develop the final cross tissue prediction algorithms. Finally, we describe the validation process for testing our algorithms in test datasets.

2.1 Subsetting Potential Covariates: C+S and C Method

We developed two distinct algorithms for predicting DNA methylation (DNAm) biomarkers across different tissues. The first algorithm, referred to as the C+S method, combines both the DNAm biomarkers derived from saliva and the raw methylation values from saliva samples. This approach allows for the saliva DNAm biomarkers to be 'updated' via methylation loci, enhancing their predictive accuracy. The terminology "C+S" references **C**pGs and **S**aliva DNAm Biomarkers being included as covariates. The second algorithm, known as the "C" method, exclusively uses saliva methylation values, ie CpGs, to predict blood DNAm biomarkers. This CpGs-only strategy is especially relevant when creating new biomarkers or when faced with datasets that lack a significant number of CpGs necessary for computing DNAm biomarkers, often due to variations in array platforms. By focusing solely on saliva methylation values, the C method provides a valuable tool in situations where comprehensive CpG data is unavailable or when developing novel biomarkers.

We restrict the potential covariates to predict each blood DNAm biomarker to CpG loci known to be informative to epigenetic clocks and conserved across commonly used methylation arrays. These loci are those published in public DNAm epigenetic clocks, specifically DNAmAge [2], DNAmAgeSkinBloodClock [23], DNAmHannumAge [3], DNAmPhenoAge [24], DNAmFitAge [5], Zhang [18], MethylDetectR [25], EpiTOC [26], and the PanMammalianClock [27]. Because some loci are not available across different array platforms, we included only CpG loci conserved across the 450K and EPIC array. This resulted in 6662 unique CpG loci to be included. For our C+S method, our potential covariates included 6663 variables: 6662 methylation sites and the saliva DNAm biomarker. In our paper, we refer to the saliva DNAm biomarker as the estimated DNAm biomarker if the current algorithms were applied directly to saliva methylation values, regardless if the algorithms were built for saliva or other tissues. For each cross tissue prediction algorithm, only 1 saliva DNAm biomarker will be included as a covariate. For the C method, we further reduced the methylation sites included to only those conserved across the 450K, EPIC, and Mammalian40K array. This resulted in 1307 unique CpG loci to be potential covariates in our C method. This is not only conducive to the development of human biomarkers but also holds potential for direct application to animal studies because the loci are conserved across mammalian species. For example, animal

models measuring DNAm under calorie restriction (CR) could be integrated into our research to better inform the conserved epigenetic changes in humans from CR. Overall, developing and comparing algorithms in these two cases informs us of information loss or gain when implementing TL in DNAm contexts.

2.2 Datasets

Our "target" data includes datasets that have DNAm values in both saliva and blood. These datasets include the tissues of the targeted research objective: predicting blood DNAm biomarkers from saliva DNAm. We refer to "auxiliary" data as datasets that include DNAm in two tissues that are not saliva and blood. Finally, we include validation datasets, which are datasets not used in the TL algorithm development stage, but instead are for testing our developed algorithms in. All the datasets used have been previously described elsewhere; we provide brief summaries here.

2.2.1 Target Data. Our target data consisted of 6 independent datasets that included DNAm in saliva and blood tissues for the same individuals. GSE111165 (n=33), GSE214901 (n=19), GSE159899 (n=19), GSE130153 (n=22), GSE59507 (n=4), and GSE73745 (n=12) for a total of 109 samples in the target datasets. In developing our methods, we employ a Leave-One-Data-Out (LODO) methodology that builds the TL model in 5 of the target datasets and tests in the 1 held out target dataset. This process is repeated for each target dataset, resulting in 6 LODO iterations per method, and the weighted average by target dataset size is used to calculate the final LODO estimate.

GSE111165 collected samples from blood, saliva, buccal, and brain tissue in epilepsy patients undergoing brain resection. DNAm was measured with both the 450K and EPIC arrays. GSE214901 measured brain, blood, saliva, and buccal in Japanese individuals undergoing neurosurgery, aged 13-73. DNA methylation was measured using the EPIC array. GSE159899 collected and measured methylation in whole blood, saliva, and T-cells using the EPIC array. GSE130153 measured saliva and blood sample methylation with the 450K array. GSE59507 measured multiple tissues from male crime scene samples aged 20-59 including blood, saliva, and semen. Methylation was measured using the 450K array. GSE73745 measured methylation in people with respiratory allergies and healthy controls in saliva and mononuclear blood cells. Methylation was measured using the 450K array.

(DT: Perhaps add some references to where these data sources have been published?)

2.2.2 Auxiliary Data. Our auxiliary datasets consisted of 5 independent datasets that included DNAm measured in two different tissues for the same individuals. Comprehensive Assessment of Long-term Effects of Reducing Intake of Energy (CALERIE) included muscle and adipose (n=130), GSE111165 included buccal and blood (n=27), GSE214901 included buccal and brain (n=19), TwinsUK included adipose and skin (n=136), and GSE48472 included fat and blood (n=6) for a total of 318 samples in the auxiliary datasets. For each of these datasets, they are presented where the first tissue listed is the tissue used as the potential covariates, and the second tissue is the tissue used for the outcome. For example, in CALERIE, muscle DNAm is used in the X and adipose DNAm biomarkers are used in the Y. In the TwinsUK data, original person ID's were not available for each tissue dataset, so individuals were matched across tissue based on SNPs, age, BMI, twin zygosity, and matching family IDs.

2.2.3 Validation Datasets. We use four datasets that measure phenotypic variables known to relate to DNAm biomarkers and DNA methylation. This included GSE119078, GSE148000, GSE149747, and HorvathHIV. The first has 59 saliva samples measured in males (n=25) and females (n=34) with and without Celiac's Disease on the 450K array. No differences were observed by disease group to the saliva DNAm biomarkers, and to our best knowledge, differential methylation from saliva DNAm for

Table 1. Dataset Summary

Purpose	Dataset	N	Tissues
Target	GSE111165	33	saliva, blood
	GSE214901	19	
	GSE159899	19	
	GSE130153	22	
	GSE59507	4	
	GSE73745	12	
Auxiliary	CALERIE	130	muscle, adipose
	GSE111165	27	buccal, blood
	GSE214901	19	buccal, brain
	TwinsUK	136	adipose, skin
	GSE48472	6	adipose, blood
Validation	GSE119078	59	saliva
	GSE148000	26	sputum
	GSE149747	122	saliva
	HorvathHIV	28-58	lymph, adipose, muscle, blood

Celiac's has not been observed elsewhere. This dataset is used to validate the relationship between sex and telomere length. The second dataset includes 26 asthma, COPD, and healthy patients where DNAm was measured in sputum using the 450K array, which is similar to, but not identically saliva. Age was observed to be different between the three disease classifications, so all models include age as a covariate. This dataset was used to look at differences in predicted DNAm biomarkers to disease status, relate predicted blood cell count models to measured lymphocyte percentage, and lung health DNAm biomarkers to reported cumulative smoking pack years. The third dataset is a exercise, diet, and sleep intervention study with saliva DNAm measured at baseline, 4 weeks, and 8 weeks after intervention on the EPIC v1 array. The HorvathHIV data measured methylation with a custom array (HorvathMammalMethylChip) from 661 samples across 11 human tissues (adipose, blood, bone marrow, heart, kidney, liver, lung, lymph node, muscle, spleen and pituitary gland) [12]. This sample included 133 clinically characterized, deceased individuals, including 75 infected with HIV. Based on the results of the initial study, we restricted our analysis to tissues with substantial sample sizes, relevant to saliva, and accessible- being lymph, muscle, and adipose.

When available, DNAm was processed using bioconductor minfi and SeSAMe packages in R with Noob normalization. This was not possible for GSE130153 which provides only processed DNAm data and not the original idat files. In addition, 3 datasets (GSE73745, 159899, and 130153) did not provide chronological age, which is required for some DNAm clocks. In these cases, their predicted age from DNAm was used in place of their chronological age. After pre-processing, DNAm data was uploaded to the online DNAm clock calculator to calculate DNAm biomarker values (https://dnamage.clockfoundation.org). The resulting epigenetic clocks were used as end points in our prediction models.

2.3 Transfer Learning Methodology

We are interested in estimating a set of regression coefficients β, mapping DNAm from saliva to DNAm biomarkers in blood. The model of interest is

$$y_j^{(0)} = x_j^{(0)\top}\beta + \epsilon_j^{(0)},$$

where y_j is the jth blood DNAm Biomarker, x is the vector of saliva DNAm inputs, and (0) refers to the target data. Because we develop two different algorithms for each DNAm biomarker, for the C+S method, x includes the jth saliva DNAm biomarker and saliva methylation values (1×6663 size vector), whereas for the C only method, x includes only saliva methylation beta values (1×1307 size vector).

Together with the target data, we observe additional data from a collection of auxiliary studies, denoted as K, to enhance our estimation of the vector β. Each auxiliary study provides samples that may vary in their relevance to the target data. The incorporation of these auxiliary samples into our analysis can take various forms—they can be merged into a single dataset, kept separate, or grouped into distinct combined datasets. To accommodate these different integration strategies, we define L as the ensemble of auxiliary data configurations utilized. Specifically, L represents any subset or combination of the K studies that we use for informing the estimation of β.

Mathematically, let us consider K auxiliary samples indexed by $k = 1, \ldots, K$. The set L then represents a collection of these samples in various configurations, such as $L = \{(4), (4, 1), (4, 1, 3), (4, 1, 3, 2)\}$, or as a single combined set $L = \{(1, 2, 3, 4)\}$, or as individual datasets $L = \{(1), (2), (3), (4)\}$ when $K = 4$. Here, the first instance illustrates L containing three groupings of l combining multiple auxiliary datasets (as is performed in the default settings of TransLasso), the second instance shows L as one grouping that combines all auxiliary datasets, and the third instance depicts L with each l representing an individual auxiliary dataset. For scenarios where auxiliary samples are considered individually, such as in the Oracle 1df method, we can substitute l with k in our notation. This substitution reflects the scenario where each auxiliary sample is assessed separately to inform the model.

The general methodology can be summarized into five primary steps below.

Step 1: Perform Lasso on the Target Data (saliva DNAm → blood DNAm biomarker):

$$\hat{\beta}_0 = \min_{\beta_0 \in \mathbb{R}^p} \|\mathbf{y}^{(0)} - \mathbf{X}^{(0)}\beta_0\|^2 + \lambda_0\|\beta_0\|_1$$

Step 2A*: Determine the informative set, L, from auxiliary data, K, as detailed in Section 2.3.1

Step 2B: Perform Lasso on Informative Auxiliary Data for each $l \in L$ (any tissue DNAm → any other tissue DNAm biomarker):

$$\hat{w}_l = \min_{w_l \in \mathbb{R}^p} \|\mathbf{y}^{(l)} - \mathbf{X}^{(l)}w_l\|^2 + \lambda_l\|w_l\|_1$$

Step 3: Perform Lasso on Target Data Residuals (saliva DNAm → Residual blood DNAm biomarker):

$$\hat{\delta}_l = \min_{\delta_l \in \mathbb{R}^p} \|(\mathbf{y}^{(0)} - \hat{\mathbf{y}}^{(0)}_{\hat{w}_l}) - \mathbf{X}^{(0)}\delta_l\|^2 + \lambda_{0l}\|\delta_l\|_1 \tag{1}$$

$$= \min_{\delta_l \in \mathbb{R}^p} \|(\mathbf{y}^{(0)} - \mathbf{X}^{(0)}\hat{w}_l) - \mathbf{X}^{(0)}\delta_l\|^2 + \lambda_{0l}\|\delta_l\|_1 \tag{2}$$

Step 4: Calculate coefficients estimated from the auxiliary samples with or without a threshold for combining:

$$\hat{\beta}_l = I_{cw}\hat{w}_l + I_{cd}\hat{\delta}_l$$

Step 5: Combine $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_L$ for $\forall m \in 0 : L$ using the squared prediction error on the target data.

$$\hat{\beta} = \theta_0\hat{\beta}_0 + \cdots + \theta_L\hat{\beta}_L$$

where

$$\theta_m = \frac{\frac{\text{Total squared error}}{m\text{th squared error}}}{\sum_{m=0}^{L}\frac{\text{Total squared error}}{m\text{th squared error}}} = \frac{1/\sum_i^{n_0}(y_i^{(0)} - x_i^{(0)\top}\hat{\beta}_m)^2}{\sum_{m=0}^{L}[1/\sum_i^{n_0}(y_i^{(0)} - x_i^{(0)\top}\hat{\beta}_m)^2]}.$$

In Steps 1-3, the tuning parameters $\lambda_0, \lambda_l, \lambda_{0l}$ are selected through cross validation and are described in greater detail in the next Methods section. In Step 4, I_{cw} and I_{cd} are indicator vectors that indicate which coefficients of $\hat{w}_l$ and $\hat{\delta}_l$ are of magnitude greater than or equal to the threshold value of c. This is also in the next Methods section, where we detail the different constants we use, including allowing any coefficient magnitudes to be added together. In the weighting scheme θ_l of step 5, the error is calculated in the training target data. The term, $x_i^{(0)\top}\hat{\beta}_m$ is the the prediction for individual i in the target data using model m, and the summation over i aggregates squared prediction errors over all individuals. The weights are based on the inverse of the squared error for each model's predictions on the target data, which gives more importance to models with lower errors. We note that this is *not* the error from the left out target data, because then this algorithm would not be applicable to researchers developing new algorithms without validation data. Error from LODO is used at a later step, however, to combine the coefficients from our various TL models in a Super Learner process to obtain the final coefficients.

This methodology is described with all 6 target datasets together in $\mathbf{y}^{(0)}$ and $\mathbf{X}^{(0)}$, which is the case for our final algorithms. However, in the process of testing parameterization and TL method specification, we employ a Leave-One-Data-Out (LODO) process, meaning 1 target dataset is held out and the remaining 5 are used to develop the TL coefficients in the steps described above. As such, for developing the TL algorithm for each biomarker and each tested TL method, the 5 steps were performed 6 times to capture $j = 6$ LODO folds. This results in $\hat{\beta}_{-0_1}, \ldots, \hat{\beta}_{-0_6}$ for each jth fold (total of 6 target datasets), which are the target data coefficients when the jth target data is left out ($\hat{\beta}_{-0_j}$). After each fold is run, the $\hat{\beta}_{-0_j}$ coefficients are applied in the jth dataset to calculate the LODO error and correlation for comparing the TL methods. The LODO error and correlation are a sum of the metrics weighted by the left out dataset size, n_j. For example, the LODO total squared error is calculated as:

$$\text{LODO Squared Error} = \sum_{j=1}^{6} \frac{n_j}{n_0} \sum_{i=1}^{n_j} \left(y_{ji}^{(0)} - \hat{y}_{ji}^{(0)} \right)^2 \tag{3}$$

$$= \sum_{j=1}^{6} \frac{n_j}{n_0} \sum_{i=1}^{n_j} \left(y_{ji}^{(0)} - x_{ji}^{(0)\top}\hat{\beta}_{-0_j} \right)^2 \tag{4}$$

where $y_{ji}^{(0)}$ is the blood DNAm biomarker of interest for the ith person in the jth left out target dataset and $x_{ji}^{(0)\top}$ is vector of saliva DNAm for the ith person in the jth left out target dataset.

2.3.1 Calculating Informative Auxiliary Sets (Step 2A):. In Step 2A, we use * to denote the possibility to skip this step, as is the case for the Oracle methods, where informativeness is a-priori known and therefore does not need calculated. We employ two oracle methods. One, which we refer to as Oracle A0, treats all auxiliary datasets as equally informative, and L is the combined set of all K auxiliary data. In the second oracle method, deemed Oracle 1df, each auxiliary dataset is considered informative, but not necessarily equal. In this case, L is exactly K, with each auxiliary dataset being treated individually.

When we are determining the informativeness of auxiliary datasets, we construct L, the set of informative auxiliary datasets. In summary, this algorithm computes differences in marginal correlations between auxiliary datasets and the target datasets. The most significant of these differences calculates an index $\widehat{R}^{(k)}$ that gives an indication of the informativeness of the kth auxiliary dataset. These are ranked and auxiliary datasets are taken in sequence to produce the informative set L.

Step 1: Calculate $\hat{\Delta}^{(k)}$:

$$\widehat{\Delta}^{(k)} = \frac{1}{n_k}\sum_{i=1}^{n_k} x_i^{(k)} y_i^{(k)} - \frac{1}{n_0}\sum_{i=1}^{n_0} x_i^{(0)} y_i^{(0)}.$$

$\hat{\Delta}^{(k)}$ holds the marginal statistics for the k^{th} auxiliary dataset, which is the average difference between the k^{th} auxiliary dataset CpG's and the target dataset CpG's correlation to the DNAm biomarker outcome, y. The term $X'y$ captures how each predictor variable individually correlates with the DNAm biomarker variable. Thinking of y as a vector in a space, and each column of X as another vector in that space, then each component of $X'y$ is essentially the projection of y onto each of those predictor vectors. In this framework, we are taking the average difference of the projection between the auxiliary data and target data for each p. ($\hat{\Delta}^{(k)}$ is a $p \times 1$ vector)

Step 2: Pick out the most significant components of $\widehat{\Delta}^{(k)}$ by obtaining the top t^* $\hat{\Delta}^{(k)}$ (largest differences) between the auxiliary and target data, called $\widehat{T}_k$. As such, $\widehat{T}_k$ are indices of the largest marginal statistics for each auxiliary dataset. The default value for t^* is one third the target dataset size ($n_0/3$), and we also explore variations to include more $\hat{\Delta}^{(k)}$. This parameter changes how many of the largest differences are considered. Call this subset $\hat{\Delta}^{(k)}_{\widehat{T}_k}$.

$$\widehat{T}_k = \left\{1 \le j \le p : \left|\widehat{\Delta}_j^{(k)}\right| \text{ is among the first } t_* \text{ largest of all}\right\}$$

By taking the largest values, we are capturing the most unique differences between the auxiliary and target data. Capturing the largest differences allows us to understand how different the auxiliary data are from the target data.

Step 3: Calculate Sparse Index ($\widehat{R}^{(k)}$) for the k^{th} auxiliary dataset.

$$\widehat{R}^{(k)} = \left\|\widehat{\Delta}^{(k)}_{\widehat{T}_k}\right\|_2^2$$

This estimated sparse index is the squared Euclidean norm of the vector $\hat{\Delta}^{(k)}_{\widehat{T}_k}$. Essentially, this is summing the total squared differences of the top t^* correlation differences. Auxiliary datasets with smaller $\widehat{R}^{(k)}$ are more informative because the total deviation between the auxiliary and target data is small. This metric will be larger for auxiliary datasets that have more significant differences from the target datasets. After this, we use these sparsity indices to make candidate sets / subsets of the auxiliary data. We rank each $\hat{R}^{(k)}$ and take the smallest values in sequence to make L subsets of size $1, \ldots, K$. Specifically, the l^{th} candidate set is:

$$l = 1 \le k \le K : \widehat{R}^{(k)} \text{is among the first } l \text{ smallest of all}$$

For illustration, we have 5 auxiliary datasets, and suppose the datasets' rank of $\widehat{R}^{(k)}$'s is 4,1,2,3,5. Then L would be the 5 element set $\{(4), (4,1), (4,1,2), (4,1,2,3), (4,1,2,3,5)\}$. Computation proceeds as described above with each subset of auxiliary datasets being run together in the Lasso algorithm. In total, there will be K subsets each with $1, \ldots, K$ auxiliary datasets included. Therefore, the most informative auxiliary dataset will be present in all L subsets.

2.4 Data Distinctions to Classical Transfer Learning

Our study falls within a unique category of 'multi-source, multi-target, high-dimensional, homogeneous transfer learning', however our data has novel characteristics not typically observed or studied in TL.

While conventional transfer learning might focus on domains with significant overlap or similarity, our approach blends data across various tissue types, some of which might share more

characteristics than others. This heterogeneity is distinct in that it involves not only different tissues (with tissue specific methylation patterns) but also different relationships across two different, distal tissues. By predicting across tissues and incorporating data from different cross tissue predictions, we are seeking to capture universal within-person shared tissue signals. Furthermore, our target datasets, focusing on saliva DNAm predicting blood DNAm biomarkers, are aggregated from six different studies. This multi-study integration is uncommon in transfer learning, where target data is usually drawn from a single or more uniform source [**? ?**]. While Horvath's DNAmAge clock was built with multiple tissues, most other DNAm biomarkers are not [2]. The diversity in our target datasets adds complexity due to variations in study design, data collection methods, and participant characteristics. This methodology requires sophisticated data handling and model optimization to manage the heterogeneity across sources and domains. We explore the optimization of TL techniques for complex epigenetic problems with methods simple enough that other researchers can readily adopt.

2.5 Optimizing Transfer Learning Methodology for Cross-Tissue DNAm Prediction

The core aim of our study was to assess the utility of Transfer Learning (TL) in the context of DNA methylation (DNAm) and DNAm-based biomarkers. We made methodological modifications within the TransLasso framework to tailor the TL process for DNAm based biomarkers. The variations explored include lambda penalty optimization, coefficient thresholding, and the integration of auxiliary dataset information, which are delineated below.

2.5.1 Lambda Penalty. The λ penalty in penalized regression models is typically determined through cross-validation (CV), either by identifying the λ that minimizes CV error or by selecting a more conservative λ within one standard error above the minimum. In transfer learning contexts involving multiple auxiliary datasets, optimizing λ for each dataset can be computationally intensive. We expand upon the TransLasso algorithm, which uses a constant calculated from the target dataset, c_0, to define λ_k for each auxiliary set: $\lambda_k = c_0 \times \sqrt{2(log(p))/n_k}$. Our investigation included this approach and allowed for λ selection through CV in each auxiliary set, aspiring to harness dataset-specific nuances to inform λ choice.

In our CV λ selection process, λ_0 is the optimal λ for the target dataset from CV on $\mathbf{y}^{(0)}$ and $\mathbf{X}^{(0)}$, λ_k is the optimal tuning parameter for $\hat{w}_k$ from $\mathbf{y}^{(k)}$ and $\mathbf{X}^{(k)}$, and λ_{0k} is the optimal tuning parameter λ_k scaled by the target dataset size, n_0. We scrutinized the constant λ and individualized λ approaches under the min and 1se parameterizations to determine the most suitable for our DNAm biomarker prediction. This approach was geared towards determining the optimal λ for our prediction problem, with the intent to refine the model's predictive accuracy and reliability.

2.5.2 Coefficient Thresholding. In the transfer learning process, a two-step coefficient computation is used for estimating coefficients from auxiliary data: initially within the auxiliary dataset to predict the outcome of interest (weights, w_k), followed by an adjustment for biases between the target and auxiliary datasets (δ_k). Lasso regression shrinks coefficients toward zero, which can result in some coefficients being small and near zero. To mitigate the incorporation of negligible coefficients from auxiliary sources, thresholding can be applied prior to combining the auxiliary weights and bias coefficients. While the conventional TransLasso method maintains coefficients exceeding a threshold of λ (described above), our exploration incorporated more lenient thresholds, including halving the threshold ($0.5 \times \lambda$) and considering all coefficients, regardless of coefficient magnitude. We compare these three thresholding approaches and refer to them as 'lambda', 'half lambda', and 'all'. This approach acknowledges the inherent characteristics of DNA methylation

(DNAm) where the effects can be minuscule. By lowering or removing the thresholding, we may account for the subtlety of DNAm effects and improve the application of TL to DNAm data.

2.5.3 Auxiliary Dataset Information. The relevance of auxiliary datasets to your target data and objective can often be ambiguous. An 'oracle' scenario would entail using only known informative auxiliary datasets; however, this is not always practical. Consequently, we need a mechanism to gauge the informativeness of auxiliary datasets. We examined both the oracle and estimation approaches for auxiliary dataset incorporation. In the oracle scenario, we considered all auxiliary data as either a single collective sample or as separate individual datasets, referred to as 'Oracle A0' and 'Oracle 1df', respectively. Informativeness of auxiliary datasets were computed as described above using the differences in marginal correlations. We varied the number of considered correlations to calculate auxiliary data similarity starting from the default value of one-third of the target dataset size ($n_0/3$). For the C+S method, we considered the top 100, 500, 2000, and 6663 (all) correlation differences. For the C method, limited to the 40K array-conserved sites, we considered the top 100, 500, and 1307 (all) correlation differences. These methods are referred to as Rhat and the corresponding number of correlations considered, like 'Rhat100'.

2.6 Evaluating TL Methods and Developing Final Algorithms

2.6.1 Evaluating Transfer Learning (TL) Methodologies. To assess the operating characteristics of various TL algorithms, we conducted a comprehensive evaluation of their performance using Leave-One-Dataset-Out (LODO) correlation, mean squared error (MSE), and mean absolute percent error (MAPE). The results of this analysis, delineated by algorithmic variation, are presented in Table 2. This approach provides a holistic view of how different parameters and methods of integrating auxiliary data affect the predictive accuracy for DNA methylation (DNAm) biomarkers. It should be noted, however, that while this evaluation offers a broad understanding of algorithmic performance, it does not specifically address variations across individual DNAm biomarkers.

2.6.2 Development of Optimized TL Algorithms. Our methodology for refining TL algorithms involved a detailed analysis of their performance across each DNAm biomarker. We calculated and then ranked both the correlation and prediction error for each TL method within individual DNAm biomarkers. This ranking was based on a weighted LODO MSE and the correlation between the predicted and actual blood DNAm biomarkers, with weights assigned in proportion to the size of the dataset left out. The most effective methods were identified based on their mean performance in terms of correlation and MSE across all DNAm biomarkers. The four top-ranked methods were subsequently applied to the entire target dataset, with the resulting coefficients being recorded.

In the final stage of our analysis, we employed a Super Learner approach to combine coefficients from these four top performing methods, thereby deriving the final algorithms' coefficients for each DNAm biomarker. The weights assigned to each coefficient set were inversely proportional to the squared errors from the LODO analysis of the respective method. We opted for the Super Learner framework over the single best-performing TL method because of the former's proven superiority in enhancing predictive accuracy and providing more stable estimates in regression models, particularly when multiple a priori techniques are utilized [28]. We developed the optimal TL algorithm separately for the two algorithms desired: one incorporating saliva DNAm biomarkers with 6662 CpGs as potential covariates and the other using only the 1307 CpGs conserved on the 40K array. Consequently, for each DNAm biomarker, we developed two cross-tissue prediction algorithms — one incorporating saliva DNAm biomarkers and the other based solely on saliva methylation beta values. However, as described in the section below, we do not provide all DNAm biomarker algorithms to researchers because the TL method does not always adequately predict DNAm biomarkers.

2.7 Validation of Algorithms

In our study, we embarked on a comprehensive evaluation of Transfer Learning (TL) to determine its potential contributions in the realm of cross-tissue prediction, particularly in the context of predicting blood DNA methylation (DNAm) biomarkers. Our evaluation framework was multi-faceted, focusing on three key benchmarks deemed critical for any robust cross-tissue prediction algorithm.

2.7.1 Comparison of TL with Direct Saliva DNAm Estimates. Initially, we assessed whether TL offers an advantage in predicting blood DNAm biomarkers from saliva DNAm. This involved contrasting the efficacy of our TL algorithms with the baseline approach of directly computing biomarkers from saliva DNAm. The TL algorithms can be considered advantageous if they demonstrate superior performance in approximating blood DNAm biomarkers compared to using saliva DNAm as a direct surrogate.

2.7.2 Benchmarking TL against Lasso Regression. We then compared the TL algorithms against conventional Lasso regression to evaluate the benefits of adopting advanced computational techniques. Lasso regression was applied solely to the target data, serving as a comparative baseline. In contrast, our TL methods leveraged both the target data and additional auxiliary cross-tissue DNAm data. The key distinction between the TL and Lasso methods lies in the incorporation of this auxiliary cross-tissue DNAm data in TL as both TL and Lasso used the same target and held out datasets for development and comparison. We analyzed and compared the Leave-One-Dataset-Out (LODO) correlations and errors generated by both the TL and Lasso algorithms. The TL algorithms can be considered advantageous for DNAm biomarkers if they demonstrate better prediction accuracy compared to Lasso techniques.

2.7.3 Comparing TL Algorithms with Different Domains. Lastly, we scrutinized the differential predictive power and accuracy between our two distinct TL algorithms developed for each DNAm biomarker. One algorithm incorporated saliva DNAm biomarkers and beta values across 6662 CpG loci, while the other was confined to 1307 CpG sites alone. This comparative analysis aimed to determine whether the inclusion of saliva DNAm biomarkers as covariates significantly enhances predictive accuracy or if their inclusion is largely redundant in the context of these TL models.

2.8 Scope of Algorithms: Application to Validation Sets

The utility of these algorithms extends beyond mere accuracy; for them to resonate with and be adopted by the broader research community, they must demonstrate an ability to reflect biologically meaningful relationships. We explore this by examining if our predicted DNAm biomarkers have the same relationships that have been established between blood DNAm biomarkers and phenotypic variables across three distinct validation datasets. Then, we examine the robustness of input tissue type to our algorithms using a validation dataset with multiple tissues and HIV status.

We calculate the association between our predicted blood DNAmTL biomarker and sex to evaluate if our predictions align with known biological trends, where females tend to have longer telomere length compared to men of the same chronological age [29]. Next, we compare predicted blood DNAm biomarkers among people with COPD, asthma, and healthy controls, using sputum DNAm and adjusting for age. Research has demonstrated older DNAm age estimates in people with COPD and asthma compared to controls [30]. We also investigate the congruence of predicted blood cell count DNAm biomarkers with lymphocyte percentages. The study also reports person-reported cumulative (or lifetime) smoking pack years, and we evaluate the association to predicted DNAm biomarkers surrounding lung health and fitness: DNAmPackYears, DNAmFEV1, and DNAmVO2max blood biomarkers. Finally, we explore if our predicted blood DNAm biomarkers change in the

expected direction from a longitudinal exercise, diet, and sleep intervention study. Here, we employ a main effects mixed model, controlling for age and study duration, to determine if our fitness-related blood biomarkers show expected improvements in the intervention group. This analysis also offers the unique opportunity to assess whether the newly developed blood DNAm fitness biomarkers exhibit expected improvements from an exercise intervention, an evaluation that has not yet been undertaken. These comparisons validate the predictive potential of the TL algorithms and demonstrate their robustness to application in novel data sources.

2.8.1 Assessing Algorithmic Flexibility to Tissue Type. Because our TL approach incorporates information from multiple human tissues, we are interested in understanding if the algorithm is robust to tissue type. For example, instead of saliva DNAm, can we provide lymph nodes or other tissue DNAm and still have accurate blood DNAm biomarker predictions? We applied the C algorithms to three accessible tissues (lymph nodes, adipose, and muscle DNAm) in HIV-negative and HIV-positive individuals and compared the predicted DNAm biomarkers. This helps us assess if the predictions accurately reflect the accelerated aging and immune dysregulation associated with HIV infection.

2.9 Summary of Terminology Used

- **LODO**: Leave-One-Data-Out, meaning leaving 1 target dataset out at a time.
- **TL**: Transfer Learning
- **C+S Method**: One algorithmic set up which includes 6662 saliva CpG loci and the saliva DNAm biomarker as covariates to predict the blood DNAm biomarker.
- **C Method**: The second algorithmic set up which includes 1307 CpG loci as covariates to predict the blood DNAm biomarker.
- **Target Data**: Datasets that include saliva and blood DNAm from the same individual. There are 6 of these and are used to develop the algorithms. Referred to with $^{(0)}$.
- **Auxiliary Data**: Datasets that include two tissues that are not saliva and blood from the same individual. There are $K = 5$ of these and are used to develop the algorithms. Referred to with $^{(k)}$.
- **Validation Data**: Four datasets that are not used to develop the algorithms, but are used to apply the algorithms to validate the signatures.
- **Oracle A0**: TL method where all auxiliary datasets are a-priori specified as equally informative and combined into 1 auxiliary dataset
- **Oracle 1df**: TL method where all auxiliary datasets are specified as informative and used as individual auxiliary datasets
- **Informative Auxiliary Sets**: When not using the Oracle TL methods, we estimate which auxiliary datasets are informative for our target data. Referred to with l and L.
- **Min / 1SE**: The λ penalty term in Lasso that either minimizes the CV error or is the value that is at most 1 standard error above the minimum CV error.
- **lambda, half lambda, all**: The three different coefficient thresholds used for combining weights, $\hat{w}_k$, and bias coefficients, $\hat{\delta}_k$, from the auxiliary data
- **Rhat**: The different number of correlation differences considered when determining auxiliary informative sets.

3 RESULTS

3.1 Optimal TL Algorithm

3.1.1 Optimal Parameters for C+S Method. The best median LOOCV correlation was observed using the minimum lambda in the C+S methods (0.515) (Table 2). Interestingly, despite a relatively lower correlation of 0.475 under the constant lambda, determined based on CV in target data and auxiliary dataset size, the error metrics are slightly reduced. This suggests improved prediction accuracy despite a less strong linear relationship between the predicted and true blood DNAm biomarker. Settings without thresholding or partial thresholding results in comparable correlation with no thresholding having 1.3% lower error. Both of these methods outperform complete (lambda) thresholding with a difference of 0.05 correlation and 2.3% higher error, suggesting complete thresholding underfits the model by removing small, but informative coefficients.

The integration of auxiliary datasets through the Oracle 1df approach yielded the best correlation and the lowest Mean Squared Error (MSE) on average, with a median Mean Absolute Percentage Error (MAPE) similar to other methods incorporating auxiliary information. Contrary to expectations, the Oracle 1df method outperformed the standard Oracle approach, which treats all auxiliary datasets as a single combined source, improving correlation by 0.06 and maintaining comparable MAPE. Furthermore, the estimation of the informative set indicated a beneficial trend in incorporating more Rhats, as evidenced by a gradual improvement in correlation (R = 0.494, 0.520, 0.530) with negligible variation in error metrics. To understand variability in performance for the C+S method based on parameterization, we provide Supplemental Figure 2 for each biomarker.

In summary, the C+S method's optimal generalization employs the Oracle 1df approach, does not perform coefficient thresholding, and uses the minimal lambda derived from cross-validation in each dataset.

3.1.2 Optimal Parameters for C Method. In contrast, the C method consistently demonstrates higher median R values across most parameters compared to C+S. This suggests a stronger correlation with the actual blood DNAm biomarker under the C setting. Echoing the C+S method's findings, the minimum lambda from auxiliary data CV was preferred, achieving the highest average R of 0.604 and lowest MAPE of 24.8% (Table 2).

In a departure from C+S, the C method favored more rigorous coefficient thresholding. Specifically, the median R of 0.597 with complete thresholding was 0.016 higher than settings without thresholding. This divergence between methods suggests a greater number of CpGs are needed to estimate the blood biomarker in the absence of original tissue DNAm biomarkers (C method), however, this also propagates noise in the prediction, which can be corrected by removing smaller coefficients. Finally, auxiliary methods using either Oracle 1df or estimating the informative set with more Rhats perform comparably well based on R and MAPE. While Oracle and Oracle 1df have similar correlations, Oracle 1df decreases error by approximately 2%.

Thus, the C method's best practice includes the minimal CV lambda from each auxiliary dataset combined with complete (lambda) coefficient thresholding. Using either Oracle 1df or estimating the informative sets with more Rhats are similarly efficacious.

3.1.3 General Recommendations. Both C+S and C methods favor the Oracle 1df method and the minimal lambda. They diverge on coefficient thresholding — more thresholding benefits the C method, while less thresholding benefits the C+S method. These recommendations, however, do not account for individual biomarker variability, which may influence the optimal choice for a specific biomarker. Researchers with the resources to conduct an LOO procedure are advised to employ the TL tuning function to determine the best parameters for their specific scenario. In the absence of such a procedure, our broad recommendations provide a starting point.

Table 2. Optimal Parameters for Transfer Learning with DNAm

Parameter	C+S			C		
	median R	median MSE	median MAPE	median R	median MSE	median MAPE
Lambda						
Min	0.515	240	25.2	0.604	209	24.8
1SE	0.503	237	26.2	0.572	228	25.5
Constant	0.475	219	24.8	0.535	237	27.0
Coefficient Threshold						
Complete (lambda)	0.490	256	27.1	0.597	226	25.5
Partial (.5 lambda)	0.545	248	26.1	0.592	248	25.6
None	0.540	220	24.8	0.581	200	25.1
Auxiliary Informative						
Oracle	0.493	248	26.2	0.596	248	26.7
Oracle 1df	0.554	214	25.9	0.592	243	24.8
Estimate A0	0.513	244	26.1	0.586	214	25.5
Rhat nk/3	0.494	236	25.4	0.573	226	25.9
Rhat 500	0.520	248	26.2	0.592	208	25.4
Rhat All	0.530	242	26.2	0.592	213	25.4

3.2 Final Algorithms

Our previous section evaluated individual parameters, however, it's crucial to recognize that the best individual parameterizations might not synergize when combined. Therefore, we ranked each TL method within each biomarker to determine the most efficacious models. These results generally aligned with our recommendations, particularly for the C+S method where the Oracle 1 df, min lambda, and all coefficient setup emerged as the top-ranked for lowest Mean Squared Error (MSE) and highest correlation among all C+S TL methods. Despite its superior performance, it didn't achieve a universal top rank, highlighting the presence of multiple algorithms with nearly identical performance across various methods. The top 4 C+S methods were Oracle A0 1df, min lambda, all coef; Oracle A0, 1se lambda, half coef; Estimate A0 Rhat nk/3, 1se, half coef; and Estimate A0 Rhat All, 1se lambda, half coef. For the C method, the top 4 algorithms were OracleA0 1df, 1se lambda, all coef; Estimate A0 Rhat 500, min lambda, all coef; Estimate A0 Rhat nk/3, min lambda, lamb coef; and Oracle A0, min lambda, lambda coef. The variation in top methods reaffirms that no single TL method suits all scenarios. Given the absence of one-size-fits-all solution, we used a Super Learner approach to generate the final coefficients for each biomarker. This strategy capitalizes on the strengths of each top algorithm, mitigating their individual weaknesses and catering to the specificities of each biomarker. The final algorithms for each biomarker are readily accessible on our GitHub repository at https://github.com/kristenmcgreevy/EpigenTL, and are also provided in the Appendix section 7.

3.3 Algorithmic Comparison

3.3.1 Comparison to Saliva Surrogates. In the comparative analysis of Transfer Learning (TL) algorithms for estimating blood DNA methylation (DNAm) biomarkers from saliva DNAm, we found that 20 out of 26 biomarkers were more accurately predicted by TL methods than by their saliva DNAm surrogates, based on Mean Squared Error (MSE) and correlation metrics. When separating the comparison between the C+S method and C method to saliva, 18 and 15 biomarkers were more accurately predicted with the TL methods, respectively. The amount of improvement

using either of the Transfer Learning methods varied by biomarker. Some biomarkers saw drastic improvement, such as DNAmGrip_noAge and DNAmLeptin, which had 0.364 and 0.364 stronger correlation, respectively. DNAmLeptin also saw a 7 figure improvement in MSE when using our C+S algorithm. For our provided algorithms, the average LODO correlation improvement of our algorithms compared to the saliva surrogate itself is 0.120. This improvement is not universal across all DNAm biomarkers, however. One biomarker—Gran—showed notably poor correlations across TL, Lasso, and saliva surrogate methods, prompting its exclusion from any saliva to blood biomarker recommendations. In addition, CD8pCD28nCD45RAn did not have good performance with C+S method, C method, or Lasso, however the saliva surrogate had a decent correlation (0.33) (Supplemental Table 1).

3.3.2 Comparison between TL and Lasso. When comparing TL algorithms directly against Lasso regression, TL algorithms outperformed Lasso for 23 out of 26 biomarkers. This underscores the significant potential of TL algorithms in the generation of new DNAm biomarkers. Additionally, the Lasso algorithm averaged the lowest/worst rank in MSE and Correlation across all TL and Saliva surrogate methods within each biomarker. Even when Lasso outperformed the TL methods, it did not outperform the saliva surrogate alone. With these results, we did not observe any instance where Lasso would be more beneficial than TL when saliva surrogates are available. As such, we strongly encourage researchers to adopt TL methods in place of Lasso when similar data are available, like when different tissue DNAm samples are available. Additional metrics evaluating median absolute percent error are presented in the Appendix Section 1.

3.3.3 Comparison of C+S to C method. Unexpectedly, 9 biomarkers estimated with the C method surpassed the performance of the C+S method. This suggests that incorporating DNAm biomarkers from the original tissue can sometimes introduce noise to its prediction, with pure methylation loci offering clearer signals for certain biomarkers. DNAmCystatinC was one biomarker where the C method out performed the C+S method with a 0.259 improvement in correlation and 9 figure improvement in MSE compared to C+S method and saliva surrogate.

While not all TL algorithms outshined their corresponding saliva surrogates, they did exhibit substantial predictive power for 23 out of the 26 biomarkers. For instances where saliva DNAm biomarkers cannot be calculated without extensive imputation—as with the 40K array— our C method algorithms provide a valuable alternative. This resulted in 10 biomarkers having both a C+S algorithm and C algorithm. Three biomarkers have a C algorithm excluded due to inadequate predictive power: CD8pCD28nCD45RAn, Gran, and PlasmaBlast. In addition, we want to iterate that for CD4.naive, CD8.naive, DNAmB2M, and DNAmPAI1 biomarkers, the C algorithms are only recommended when saliva DNAm biomarkers are unavailable. A summary table detailing the available algorithms by biomarker can be found in Table 3.

In summary, 11 biomarkers were best predicted using the C+S method, 9 with the C method, 5 using saliva DNAm biomarkers directly, and 1 biomarker was not well estimated by any method. We provide algorithms for predicting 23 blood DNAm biomarkers from saliva DNAm along with guidelines for their use. For researchers in the field, this study offers a comprehensive suite of algorithms tailored to a variety of biomarkers, enhancing the predictability and applicability of DNAm studies. Our research underscores the efficacy of TL in biomarker prediction and development and encourages its adoption over Lasso regression when applicable, particularly when multi-tissue DNAm data are available.

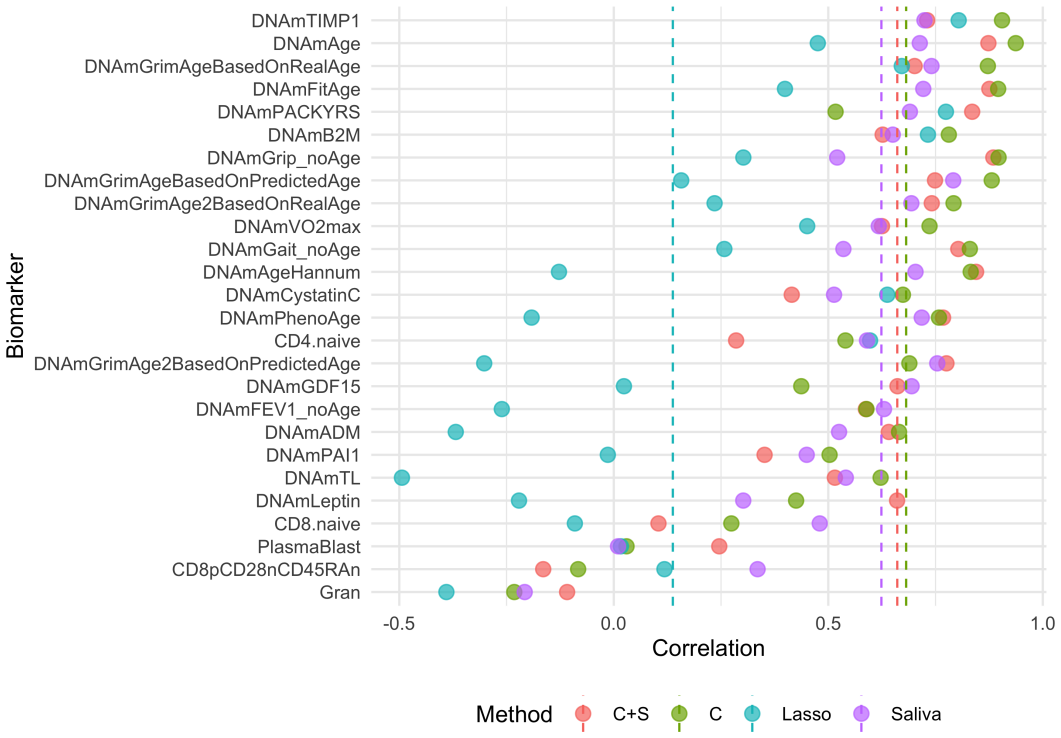


Figure 1. Correlation between True and Estimated Blood DNAm Biomarkers by top Performing TL Methods, Lasso, and Saliva Surrogates. Median correlation presented as dotted line. LODO Correlation presented for C+S, C, and Lasso methods.

3.4 Application to Validation Datasets

3.4.1 DNAmTL to Sex Relationship. We assessed the association between predicted blood DNA methylation Telomere Length (DNAmTL) biomarkers and sex, noting that females generally exhibit longer telomeres compared to males. Utilizing saliva DNA methylation (DNAm) as input, both C+S and C method blood DNAmTL predictions accurately reflected this trend. Specifically, females exhibited an average increase in telomere length of 0.29 and 0.3 for C+S and C methods, respectively, aligning closely with the observed 0.33 mean difference in the saliva surrogate. These disparities were statistically significant, as indicated by t-test and Kruskal Wallis test, with p-values ranging from 8.3E-6 to 3.5E-4 (Supplemental Table 2).

3.4.2 COPD, Asthma, and Healthy Controls. Subsequently, we compared predicted blood DNAm biomarkers among individuals with Chronic Obstructive Pulmonary Disease (COPD), asthma, and healthy controls, adjusting for age using sputum DNAm as input. Consistent with previous studies showing advanced DNAm age in COPD and asthma patients, our analysis identified elevated sputum DNAm biomarkers in COPD individuals. Notably, a previously unreported association was discovered with DNAmFitAge, indicating that COPD patients have a 5.38 older sputum DNAmFitAge relative to healthy controls after age adjustment (p=0.036). For asthma, two C+S predicted biomarkers (DNAmPhenoAge and DNAmGDF15) aligned with expectations, whereas DNAmVO2max demonstrated an inverse relationship potentially influenced by inhaler use. For

Table 3. Summary of Cross Tissue DNAm Biomarker Algorithms Provided

Biomarker	Best Model	C Algorithm	Total Algorithms available
CD4.naive	Saliva	Yes	1
CD8.naive	Saliva	Yes	1
CD8pCD28nCD45RAn	Saliva	No	0
DNAmADM	C+S	Yes	2
DNAmAge	C	Yes	1
DNAmAgeHannum	C+S	Yes	2
DNAmB2M	Saliva	Yes	1
DNAmCystatinC	C	Yes	1
DNAmFEV1_noAge	C+S	Yes	2
DNAmFitAge	C+S	Yes	2
DNAmGait_noAge	C	Yes	1
DNAmGDF15	C+S	Yes	2
DNAmGrimAge2BasedOnPredictedAge	C+S	Yes	2
DNAmGrimAge2BasedOnRealAge	C+S	Yes	2
DNAmGrimAgeBasedOnPredictedAge	C	Yes	1
DNAmGrimAgeBasedOnRealAge	C	Yes	1
DNAmGrip_noAge	C	Yes	1
DNAmLeptin	C+S	Yes	2
DNAmPACKYRS	C+S	Yes	2
DNAmPAI1	Saliva	Yes	1
DNAmPhenoAge	C+S	Yes	2
DNAmTIMP1	C	Yes	1
DNAmTL	C	Yes	1
DNAmVO2max	C	Yes	1
Gran	None	No	0
PlasmaBlast	C+S	No	1

COPD patients, three predicted biomarkers corresponded with expected outcomes, revealing higher mean Smoking Pack Years (14.3, p=0.031) and an average 4.24 older DNAmAge (p=0.040) (Table 4A).

Our study further investigated the relationship between cumulative pack years and predicted DNAm biomarkers pertinent to lung health and fitness: DNAmPackYears, DNAmFEV1, and DNAmVO2max. Strong correlations were observed between cumulative pack years with sputum and blood predicted C+S DNAmPackYears: 0.793 (p=1.4E-6) and 0.764 (p=5.6E-6), respectively. However, the C DNAmPackYears did not significantly correlate with cumulative pack years (p=0.348), likely due to it being an inferior algorithm as demonstrated in the Leave-One-Out-Dataset (LODO) samples. Both sputum and blood-predicted C+S DNAmPackYears and DNAmVO2max exhibited the anticipated directional correlation with cumulative pack years, with DNAmVO2max inversely associated, suggesting diminished DNAmVO2max in longer-term smokers. Furthermore, both sputum and blood-predicted C+S and C DNAmFitAge displayed significant correlations with self-reported cumulative pack years, moving in the expected direction, with both C+S and C methods exhibiting comparable correlations of 0.52. These findings on DNAmVO2max and DNAmFitAge are novel, illustrating that health phenotypes associated with physical fitness and lifestyle factors like smoking can be detected using predicted blood DNAm biomarkers derived from methylation profiles of alternative tissues. This insight underscores the broader systemic implications of fitness

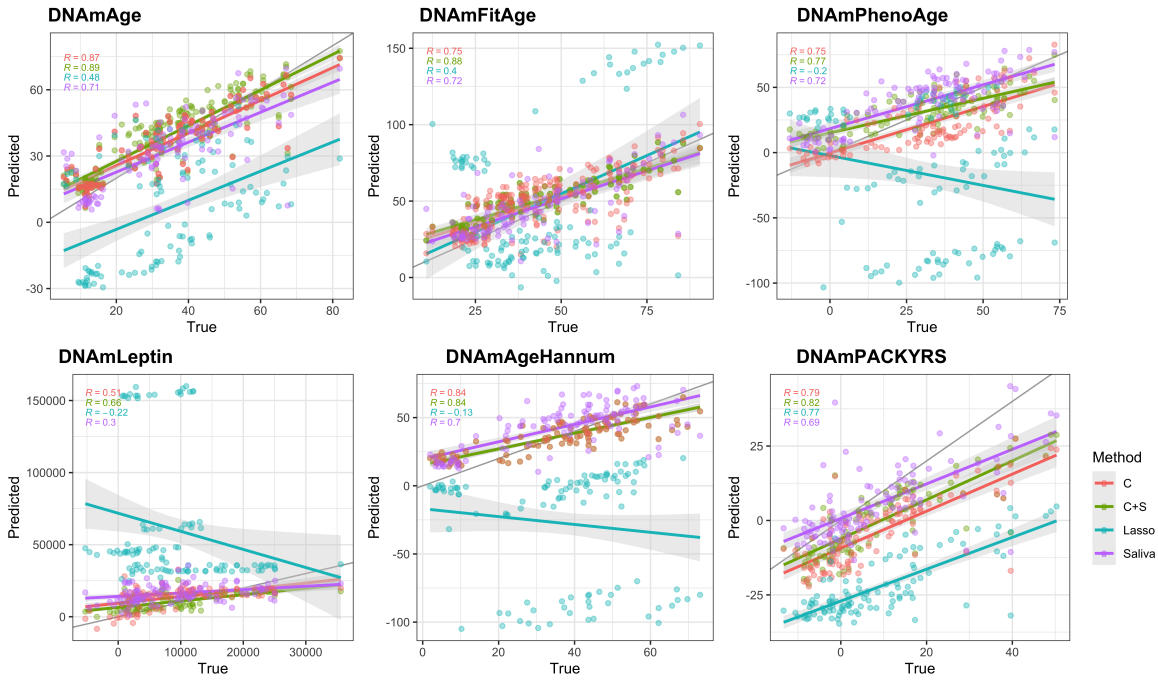


Figure 2. Scatterplots between True and Estimated Blood DNAm Biomarkers. LODO Correlation presented for C+S, C, and Lasso methods.

Table 4. COPD, Asthma, and Healthy Control Analysis

A. Comparison to Healthy Individuals After Controlling for Age

Biomarker	Asthma Effect	Asthma p-value	COPD Effect	COPD p-value
C+S DNAmAge Prediction	1.37	0.445	4.24	0.040
C+S DNAmPhenoAge Prediction	8.04	0.044	7.87	0.072
C+S CD8.naive Prediction	20.5	0.272	46.5	0.031
C+S DNAmPACKYRS Prediction	-0.80	0.831	14.3	0.0021
C+S DNAmVO2max Prediction	0.97	0.037	0.25	0.606
C+S DNAmGDF15 Prediction	181.0	0.047	58.7	0.543
Sputum DNAmFitAge	-2.37	0.288	5.38	0.036

B. Correlation with Person Reported Cumulative Pack Years

	Sputum Biomarker		C+S Predicted Biomarker		C Predicted Biomarker	
Biomarker	Pearson R	p-value	Pearson R	p-value	Pearson R	p-value
DNAmPACKYRS	0.793	1.4E-06	0.764	5.6E-06	-0.192	0.348
DNAmVO2max	-0.479	0.013	-0.377	0.058	-0.299	0.137
DNAmFEV1_noAge	-0.220	0.281	-0.199	0.330	-0.009	0.964
DNAmFitAge	0.615	8.2E-04	0.520	0.0065	0.521	0.0063

and health habits, which manifest beyond the typically studied tissues, highlighting the potential for a more holistic understanding of health and disease (Table 4B).

3.4.3 Exercise, Diet, and Sleep Intervention. Lastly, we evaluated the responsiveness of predicted blood DNAm biomarkers to a longitudinal 8-week intervention study focusing on an exercise,

diet, and sleep. Employing a main effects mixed model, we controlled for age and study duration to ascertain if fitness-related blood biomarkers demonstrated anticipated improvements in the intervention group. This analysis provided a novel opportunity to assess the effectiveness of newly developed blood DNAm fitness biomarkers. We observed a significant increase in saliva DNAmVO2max by 0.781 in the intervention group (p=0.026), marking an unprecedented finding in exercise-induced DNAm fitness biomarker improvement in just 2 months time. Additionally, blood-predicted C DNAmCystatinC (a marker of kidney function), DNAmTL, DNAmVO2max, and DNAmFEV1_noAge exhibited significant enhancements in the intervention cohort. Notably, CystatinC decreased, while telomere length, VO2max, and FEV1 increased, with the latter two showing average improvements of 0.101 liters and 0.699 mL/kg/min, respectively. These changes paralleled the saliva DNAmVO2max improvement (0.781, p=0.026), reinforcing the credibility of our C algorithm for predicting DNAmVO2max (Table 5).

Table 5. Exercise, Diet, and Sleep Intervention Effects Controlling for Age, Study Time, and Person-Specific Variation

Outcome	Treatment Effect	p-value
C DNAmCystatinC Prediction	-16580	0.030
C DNAmTL Prediction	0.112	0.031
C DNAmFEV1_noAge Prediction	0.101	0.033
C DNAmVO2max Prediction	0.699	0.033
Saliva DNAmVO2max	0.781	0.026

3.4.4 Alternative Tissues. Because our TL approach incorporates information from multiple human tissues, we were interested in understanding if the algorithm is robust to tissue type. We applied the C algorithms to three accessible tissues (lymph nodes, adipose, and muscle DNAm) in HIV-negative and HIV-positive individuals and compared the predicted DNAm biomarkers. In lymph node DNAm samples, four predicted DNAm biomarkers demonstrated significant associations in the anticipated direction with HIV status after adjusting for age. Notably, DNAmAge and DNAmPhenoAge were, on average, 6.1 and 9.4 years older, respectively, in HIV-positive individuals when accounting for chronological age (p=0.043 and p=0.012), as shown in Table 6. These findings suggest that lymph nodes might serve as viable alternative tissue for our C algorithms as they can capture some biologically known differences. Nevertheless, the limited sample size (n=28) necessitates cautious interpretation, particularly concerning the performance of the remaining 19 predicted DNAm biomarkers using lymphatic tissue.

Conversely, adipose and muscle tissues exhibited markedly poor performance with these algorithms. Specifically, nine predicted biomarkers in adipose tissue and four in muscle tissue were significantly associated with HIV status after controlling for age. However, all 13 of these biomarkers displayed effects opposing the expected direction. Consequently, these outcomes imply that adipose and muscle tissues are unsuitable for application with our current C algorithms. The distinct responses across tissues underscore the necessity for a nuanced approach to selecting appropriate tissues for DNAm biomarker analysis in the context of HIV status and potentially other conditions (Table 6).

3.5 EpigenTL Software

Our methods and cross tissue prediction algorithms are freely available for researchers to use on Github at github.com/kristenmcgreevy/EpigenTL. Researchers interested in using transfer learning for their data can use function `Epigen.TL.Lasso`. To use this function, users will need to provide

the matrix of covariates and outcome vector for the target and auxiliary datasets and specify the TL method to use. The methods are either of the three methods used to combine auxiliary datasets with the target data explored in this paper: Oracle A0, Oracle 1df, or Estimate A0. Other inputs can be specified by the user, including the coefficient threshold, number of Rhats to use, and how lambda is calculated. The function will return the transfer learning coefficients. We hope this function will be used by many to incorporate data from multiple sources, like directly incorporating animal data to improve development of new biomarkers with limited data, like those involved with brain aging.

While our methods were developed with cross tissue prediction for DNA methylation in mind, this methodology is not limited to methylation, and can be applied to any continuous outcome. We hope these methods will be used in expanding areas to develop other biomarkers not including methylation-based surrogate biomarkers.

For researchers interested in acquiring blood DNAm aging biomarkers by measuring saliva methylation instead of blood, they can use the function `Saliva.2.Blood.DNAmBiomarkers`. Users supply the saliva methylation values and specify whether to calculate biomarkers using the C or C+S algorithms. Because DNAmGrimAge is proprietary, algorithms for cross tissue prediction of DNAmGrimAge are excluded from online sources, however, researchers at a university can contact authors (KMM) directly to acquire these algorithms.

4 DISCUSSION

In this study, we explored the utility of Transfer Learning (TL) methodologies for predicting blood DNA methylation (DNAm) biomarkers from saliva DNAm, a critical advancement in the non-invasive assessment of epigenetic biomarkers. Our comprehensive approach not only sheds light on optimal TL application strategies but also provides researchers with tools to estimate 24 blood DNAm biomarkers from saliva, as well as functions to incorporate TL into their work. This research underscores the potential of TL to enhance the prediction accuracy and development of DNAm biomarkers.

Through extensive experimentation with different parameter settings and auxiliary data configurations, we identified optimal strategies for applying transfer learning in our context. This includes determining the best set of auxiliary data, the informative auxiliary set, and tuning the combination of model coefficients for improved DNAm biomarker prediction. Notably, our models demonstrated superior blood DNAm biomarker prediction accuracy from saliva compared to saliva surrogates and Lasso models, highlighting the effectiveness of TL in integrating diverse datasets and the utility of incorporating auxiliary data in high-dimensional settings.

A critical aspect of TL is the integration of auxiliary datasets, and our exploration into oracle and estimation-based methods for determining dataset informativeness addresses a broader challenge in TL: the need to leverage additional data without compromising or misdirecting the predictive focus of the model. Our comprehensive analysis, coupled with provided functions and recommendations, empowers researchers to employ our methods and final algorithms for cross-tissue DNAm biomarker prediction effectively. The Oracle 1df method, in particular, demonstrated good performance across both C+S and C methods. The divergence in optimal coefficient thresholding between the C+S and C methods suggests that the inclusion of saliva DNAm biomarkers requires a different approach to noise control compared to using CpGs alone.

The ability of TL predictions to reflect known biological trends validates the biological relevance of our algorithms. We demonstrate they reflect the longer telomeres in females and successfully differentiate between COPD, asthma, and healthy controls, with certain predicted biomarkers aligning with known disease characteristics. We show that predicted DNAm fitness biomarkers are responsive to an intervention study, demonstrating the promise of our algorithms for monitoring

lifestyle changes and health interventions. This not only validates our algorithms but also illustrates their potential for novel discoveries, as evidenced by the identification of elevated sputum DNAmFitAge in COPD patients and increased saliva DNAmVO2max following an 8 week exercise intervention. These findings demonstrate recently developed fitness DNAm biomarkers can be measured in saliva, expanding the horizons of non-invasive health monitoring.

We provide 34 algorithms to estimate 24 blood DNAm biomarkers from saliva DNAm. We provide a comprehensive outline for preferred algorithm usage in various settings, and include 10 additional C algorithms with good predictive power for cases where saliva DNAm biomarkers are unmeasurable - like due to array differences. By developing our algorithms on CpG loci measured across multiple arrays, we ensure broad compatibility and utility.

The strength of our study lies in its extensive evaluation of TL algorithms across various parameters, coupled with comparisons to conventional methods. This provides a robust framework for understanding the relative performance of different approaches and offers detailed guidelines for employing TL in practice. The versatility and broader applicability of our algorithms are demonstrated through their successful application to various validation datasets, including alternative tissues. The ability of the algorithms to perform well in other tissues, like lymph nodes, provides promise for their application and reliability in other tissues by capturing shared information across different DNA methylation profiles. However, the observed poor performance in adipose and muscle tissues highlights the necessity of verifying biomarker reliability when applied to non-saliva tissues, emphasizing the need for careful tissue selection in research and clinical applications.

Our approach, rooted in the familiar terrain of penalized linear regression, aims to offer epigenetic and aging researchers advancements without overwhelming complexity. While the simplicity of linear regression broadens accessibility and improves its potential methodological integration, it does have limitations. For example, linear regression assumes a linear relationship between the saliva CpGs and the DNAm biomarker. Other methods, like Random forest and boosted tree models, may be better able to incorporate non-linear relationships, such as the presence of SNPs into models. Future research could explore TL frameworks incorporating non-linear or advanced regression models, potentially capturing complex biological relationships. Furthermore, our methods only employed 1 method for estimating auxiliary data informativeness, and future research could explore other methods not based in marginal associations.

These insights open new avenues for non-invasive biomarker development and facilitate cross-tissue studies. The methodologies we propose are particularly advantageous for creating biomarkers in tissues that are traditionally challenging to access or available only in limited quantities, such as the brain and other internal organs. Additionally, given expansive data derived from animal models, we envision our outlined TL methods as ways for future research to integrate animal data as auxiliary datasets. The capacity of our approaches to accurately estimate informative datasets is a crucial asset in this context. It acts as a safeguard, ensuring that non-informative or distantly related data does not compromise the model's integrity. This approach has the potential to significantly accelerate biomarker development, particularly for human tissues that are typically elusive to study. We strongly encourage researchers to adopt these TL strategies, not only as a means to expand the utility of extensive animal data but also to transform it into actionable insights and human biomarkers.

The study's results provide valuable insights into the optimal parameters and comparative performance of TL algorithms in predicting blood DNAm biomarkers from saliva DNAm. The findings demonstrate the potential of TL to enhance the prediction accuracy of DNAm biomarkers, offer guidelines for selecting the right TL approach, and showcase the algorithms' applicability to various biological and clinical scenarios. These contributions significantly advance the field of

epigenetic research and open new avenues for non-invasive biomarker development and cross-tissue studies.

5 ACKNOWLEDGMENTS

REFERENCES

[1] Moore LD, Le T, and Fan G. Dna methylation and its basic function. *Neuropsychopharmacology*, 38(1):23–38, 2013.
[2] Horvath S. Dna methylation age of human tissues and cell types. *Genome Biol*, 14(10), 2013.
[3] Hannum G, et al. Genome-wide methylation profiles reveal quantitative views of human aging rates. *Mol Cell*, 49(2):359–367, Jan 2013.
[4] Lu AT, et al. Dna methylation grimage strongly predicts lifespan and healthspan. *Aging (Albany NY)*, 11(2), 2019.
[5] McGreevy KM, Radak Z, et al. Dnamfitage: biological age indicator incorporating physical fitness. *Aging (Albany NY)*, 15(10), Feb 2023.
[6] Houseman EA, Accomando WP, Koestler DC, et al. Dna methylation arrays as surrogate measures of cell mixture distribution. *BMC Bioinformatics*, 13(86), 2012.
[7] Chen BH, Marioni RE, et al. Dna methylation-based measures of biological age: meta-analysis predicting time to death. *Aging (Albany NY)*, 8(9):1844–1865, 2016.
[8] Belsky DW, et al. Dunedinpace, a dna methylation biomarker of the pace of aging. *Elife*, 11(e73420), Jan 2022.
[9] Sehgal R, et al. Systems age: A single blood methylation test to quantify aging heterogeneity across 11 physiological systems. *bioRxiv [Preprint]*, Jul 2023.
[10] Rezwan FI, Imboden M, et al. Association of adult lung function with accelerated biological aging. *Aging (Albany NY)*, 12(1):518–542, Jan 2020.
[11] Bakulski KM et al. Epigenetic research in neuropsychiatric disorders: the 'tissue issue'. *Curr Behav Neurosci Rep*, 3(3), Sep 2016.
[12] Horvath S, Lin DTS, et al. Hiv, pathology and epigenetic age acceleration in different human tissues. *Geroscience*, 44(3), June 2022.
[13] Ma B, Wilker EH, et al. Predicting dna methylation level across human tissues. *Nucleic Acids Res*, 42(6), Apr 2014.
[14] Emily Maciejewski, Steve Horvath, Jason Ernst. Cross-species and tissue imputation of species-level dna methylation samples across mammalian species. *bioRxiv*.
[15] Huang YT et al. Epigenome-wide profiling of dna methylation in paired samples of adipose tissue and blood. *Epigenetics*, 3(11), Mar 2016.
[16] Braun PR, Han S, Hing B, Nagahama Y, et al. Genome-wide dna methylation comparison between live human brain and peripheral tissues within individuals. *Transl Psychiatry*, 9(1), Jan 2019.
[17] Langie S, Moisse M, et al. Salivary dna methylation profiling: aspects to consider for biomarker identification. *Basic Clin Pharmacol Toxicol*, 121, 2017.
[18] Zhang Q, Vallerga CL, et al. Improved precision of epigenetic clock estimates across tissues and its implication for biological ageing. *Genome Med*, 11(54), 2019.
[19] Hosna A, Merry E, Gyalmo J, et al. Transfer learning: a friendly introduction. *J Big Data*, 9(102), Oct 2022.
[20] Dodlapati S, Jiang Z, and Sun J. Completing single-cell dna methylome profiles via transfer learning together with kl-divergence. *Front Genet*, 13, Jul 2022.
[21] Li S, Cai TT, and Li H. Transfer learning for high-dimensional linear regression: Prediction, estimation and minimax optimality. *J of the Royal Statistical Society Series B: Statistical Methodology*, 84(1), Feb 2022.
[22] Tibshirani R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
[23] Horvath S, Oshima J, et al. Epigenetic clock for skin and blood cells applied to hutchinson gilford progeria syndrome and ex vivo studies. *Aging (Albany NY)*, 10(7), Jul 2018.
[24] Levine ME, Lu AT, Quach A, et al. An epigenetic biomarker of aging for lifespan and healthspan. *Aging (Albany NY)*, 10(4):573–591, 2018.
[25] Hillary RF and Marioni RE. Methyldetectr: a software for methylation-based health profiling. *Wellcome Open Res*, 13(5), Apr 2021.
[26] Yang Z, Wong A, Kuh D, et al. Correlation of an epigenetic mitotic clock with cancer risk. *Genome Biol*, 17(205), Oct 2016.
[27] Lu AT, Fei Z, Haghani A, et al. Universal dna methylation age across mammalian tissues. *Nat Aging*, 3(9), Sep 2023.
[28] van der Laan MJ, Polley EC, and Hubbard AE. Super learner. *Stat Appl Genet Mol Biol*, 6(25), 2007.
[29] Gardner M, Bann D, et al. Gender and telomere length: systematic review and meta-analysis. *Exp Gerontol*, 51, Mar 2014.
[30] Chiappa S, Winn J, Viñuela A, Tipney H, Spector TD. A probabilistic model of biological ageing of the lungs for analysing the effects of smoking, asthma and copd. *Respir Res*, 14(1), May 2013.

6 SUPPLEMENTARY MATERIAL

Supplemental Table 1. Complete Comparison Among Transfer Learning, Lasso, and Saliva Surrogate Methods to Estimate Blood DNAm Biomarkers in C+S Methods

	Transfer Learning C+S Method		Transfer Learning C Method		Lasso with Saliva DNAm Biomarkers		Saliva DNAm Biomarkers		C+S to Saliva		C+S to Lasso		C to C+S		C to Saliva	
Biomarker	MSE	Corr	MSE	Corr	MSE	Corr	MSE	Corr	Corr Diff	MSE Diff	Corr Diff	MSE Diff	Corr Diff	MSE Diff	Corr Diff	MSE Diff
CD4.naive	5.4E+04	0.285	4.5E+04	0.54	1.1E+06	0.597	4.7E+04	0.590	-0.305	-7413.94	-0.312	1.1E+06	0.255	9887.86	-0.050	2473.92
CD8.naive	9.3E+03	0.104	1.0E+04	0.274	5.9E+05	-0.091	8.3E+03	0.480	-0.376	-962.02	0.195	5.8E+05	0.170	-807.92	-0.206	-1769.95
CD8pCD28nCD45RAn	51.23	-0.165	55.9	-0.0831	525.5	0.118	52.7	0.335	-0.500	1.50	-0.283	474.2	0.081	-4.67	-0.418	-3.17
DNAmADM	1198.0	0.641	2380.0	0.665	28482.6	-0.368	2.1E+03	0.525	0.116	909.65	1.009	27284.7	0.024	-1182.04	0.140	-272.39
DNAmAge	75.31	0.873	75.4	0.937	1265.2	0.476	164.9	0.713	0.160	89.64	0.398	1189.9	0.064	-0.09	0.224	89.55
DNAmAgeHannum	107.5	0.845	112.0	0.832	6391.1	-0.128	227.9	0.703	0.141	120.38	0.972	6283.5	-0.013	-4.46	0.129	115.92
DNAmB2M	1.0E+11	0.627	1.1E+11	0.781	2.3E+11	0.732	5.0E+10	0.650	-0.024	-5.2E+10	-0.106	1.3E+11	0.154	-1.19E+10	0.131	-6.4E+10
DNAmCystatinC	9.3E+09	0.415	5.6E+09	0.674	4.1E+10	0.638	8.7E+09	0.513	-0.099	-6.3E+08	-0.223	3.2E+10	0.259	3.74E+09	0.161	3.1E+09
DNAmFEV1_noAge	0.47	0.588	0.65	0.589	14.50	-0.261	0.6	0.630	-0.042	0.10	0.849	14.0	0.001	-0.18	-0.041	-0.08
DNAmFitAge	131.3	0.875	134.0	0.896	1894.0	0.399	211.5	0.721	0.154	80.13	0.476	1762.7	0.021	-2.68	0.175	77.45
DNAmGait_noAge	0.05	0.803	0.0485	0.83	4.98	0.258	0.1	0.535	0.268	0.03	0.545	4.9	0.027	0.00	0.295	0.03
DNAmGDF15	6.2E+04	0.661	1.1E+05	0.437	1.0E+06	0.024	6.8E+04	0.694	-0.033	6477.09	0.638	9.6E+05	-0.224	-4.75E+04	-0.257	-4.1E+04
DNAmGrimAge2Based OnPredictedAge	151.9	0.775	207	0.689	2175.0	-0.302	206.1	0.754	0.021	54.22	1.077	2023.1	-0.086	-55.10	-0.065	-0.88
DNAmGrimAge2Based OnRealAge	124.6	0.741	150	0.792	1292.1	0.235	196.1	0.694	0.048	71.58	0.506	1167.5	0.051	-25.43	0.098	46.15
DNAmGrimAgeBased OnPredictedAge	168.8	0.749	109	0.881	876.2	0.157	141.0	0.791	-0.043	-27.89	0.592	707.3	0.132	59.84	0.090	31.95
DNAmGrimAgeBased OnRealAge	151.3	0.701	94.4	0.872	684.2	0.671	126.9	0.741	-0.039	-24.41	0.030	532.9	0.171	56.90	0.131	32.49
DNAmGrip_noAge	27.93	0.885	21.6	0.897	800.3	0.302	92.4	0.521	0.364	64.44	0.583	772.3	0.012	6.33	0.376	70.77
DNAmLeptin	3.7E+07	0.660	9.0E+07	0.425	2.3E+09	-0.221	1.2E+08	0.302	0.359	8.5E+07	0.881	2.2E+09	-0.235	-5.32E+07	0.123	3.2E+07
DNAmPACKYRS	136.9	0.835	183	0.517	1049.7	0.774	125.3	0.690	0.145	-11.68	0.061	912.7	-0.318	-46.05	-0.173	-57.73
DNAmPAI1	2.5E+07	0.351	2.7E+07	0.503	1.1E+08	-0.014	1.1E+07	0.450	-0.098	-1.4E+07	0.365	8.6E+07	0.152	-1.81E+06	0.053	-1.6E+07
DNAmPhenoAge	188.5	0.768	228	0.758	5612.1	-0.192	306.6	0.718	0.050	118.09	0.959	5423.7	-0.010	-39.52	0.040	78.57
DNAmTIMP1	4.6E+06	0.731	3.9E+06	0.905	7.0E+07	0.804	5.0E+06	0.724	0.006	4.4E+05	-0.073	6.5E+07	0.174	7.31E+05	0.181	1.2E+06
DNAmTL	0.30	0.515	0.257	0.622	57.57	-0.494	0.5	0.541	-0.025	0.21	1.009	57.3	0.107	0.04	0.081	0.25
DNAmVO2max	7.36	0.625	7.04	0.736	2337.0	0.451	8.1	0.618	0.007	0.72	0.174	2329.6	0.111	0.32	0.118	1.04
Gran	0.13	-0.109	0.137	-0.232	1.42	-0.390	0.2	-0.208	0.099	0.05	0.281	1.3	-0.123	-0.01	-0.024	0.04
PlasmaBlast	0.20	0.246	0.236	0.0297	3.12	0.017	0.4	0.010	0.236	0.19	0.229	2.9	-0.216	-0.04	0.020	0.15

Supplemental Table 2. Relationship of DNAmTL to Sex in GSE119078

Biomarker	Mean Males (n=25)	Mean Females (n=34)	Females - Males	t-test p-value	Kruskal Wallis p-value
Saliva DNAmTL	6.54	6.88	0.33	1.4E-04	4.2E-04
C+S DNAmTL Blood Prediction	6.86	7.16	0.29	8.3E-06	2.1E-05
C DNAmTL Blood Prediction	7.20	7.50	0.30	5.3E-04	3.5E-04

Table 6. Evaluation of Lymph, Adipose, and Muscle Tissue as TL Algorithm Input After Controlling for Age in HorvathHIV dataset

Biomarker	Lymph Node (n=28)		Adipose (n=58)		Muscle (n=57)	
	HIV + Effect	p-value	HIV + Effect	p-value	HIV + Effect	p-value
CD4.naive C Prediction	-39.6	0.113	-21.5	0.213	2.06	0.801
CD8.naive C Prediction	-23.8	0.177	0.39	0.957	-9.26	0.164
DNAmADM C Prediction	10.01	0.095	-7.36	0.095	-10.6	0.014
DNAmAge C Prediction	6.11	0.043	-2.72	0.091	-3.13	0.050
DNAmAgeHannum C Prediction	5.59	0.080	-1.64	0.285	-1.69	0.201
DNAmB2M C Prediction	65721	0.064	-32378	0.131	-34684	0.114
DNAmCystatinC C Prediction	46021	0.018	-21372	0.021	-9953	0.141
DNAmFEV1_noAge C Prediction	-0.24	0.168	0.36	0.009	0.26	0.038
DNAmFitAge C Prediction	2.45	0.295	-4.66	0.014	-1.69	0.278
DNAmGait_noAge C Prediction	0.04	0.243	0.04	0.107	0.02	0.409
DNAmGDF15 C Prediction	265	0.039	-28.48	0.453	-68.5	0.177
DNAmGrimAge2BasedOnPredictedAge C Prediction	3.52	0.127	-3.00	0.047	-0.79	0.543
DNAmGrimAge2BasedOnRealAge C Prediction	3.92	0.071	-2.63	0.042	-0.45	0.702
DNAmGrimAgeBasedOnPredictedAge C Prediction	4.24	0.067	-3.90	0.030	-1.10	0.440
DNAmGrimAgeBasedOnRealAge C Prediction	3.66	0.120	-3.15	0.019	-0.94	0.442
DNAmGrip_noAge C Prediction	-3.63	0.197	4.22	0.045	4.03	0.064
DNAmLeptin C Prediction	894	0.696	-1318	0.189	414.1	0.775
DNAmPACKYRS C Prediction	6.59	0.298	-0.54	0.750	0.36	0.847
DNAmPAI1 C Prediction	519	0.182	5.07	0.983	-51.4	0.822
DNAmPhenoAge C Prediction	9.45	0.012	-2.25	0.199	-1.56	0.355
DNAmTIMP1 C Prediction	931	0.060	-561.5	0.080	-298	0.191
DNAmTL C Prediction	-0.12	0.458	0.01	0.870	-0.02	0.633
DNAmVO2max C Prediction	-0.94	0.098	0.97	0.009	0.65	0.049

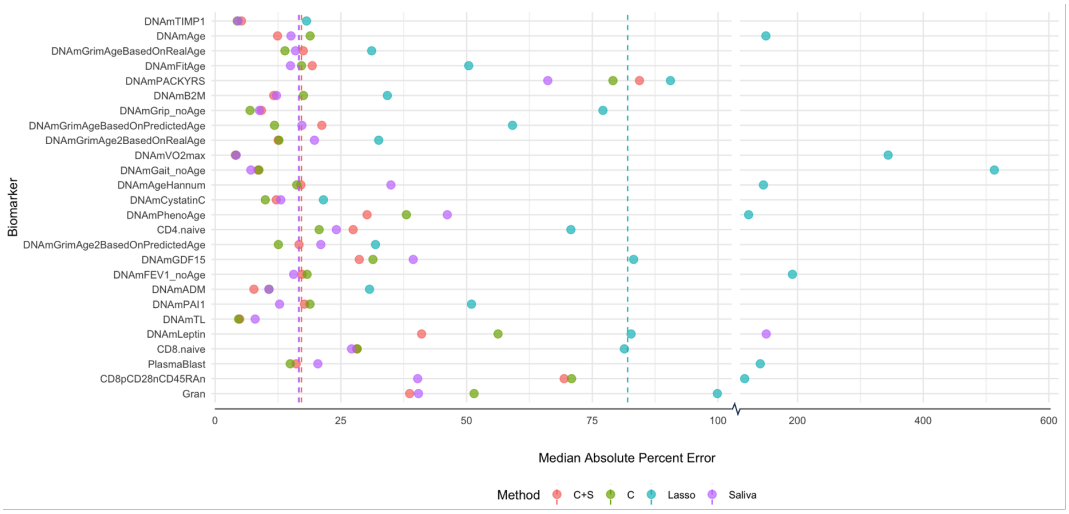
7 EPIGENTL FUNCTIONS

All of these functions can be found on Github with examples and more details on their use and function calls: https://github.com/kristenmcgreevy/EpigenTL. These functions are supplements to Section 3.2.

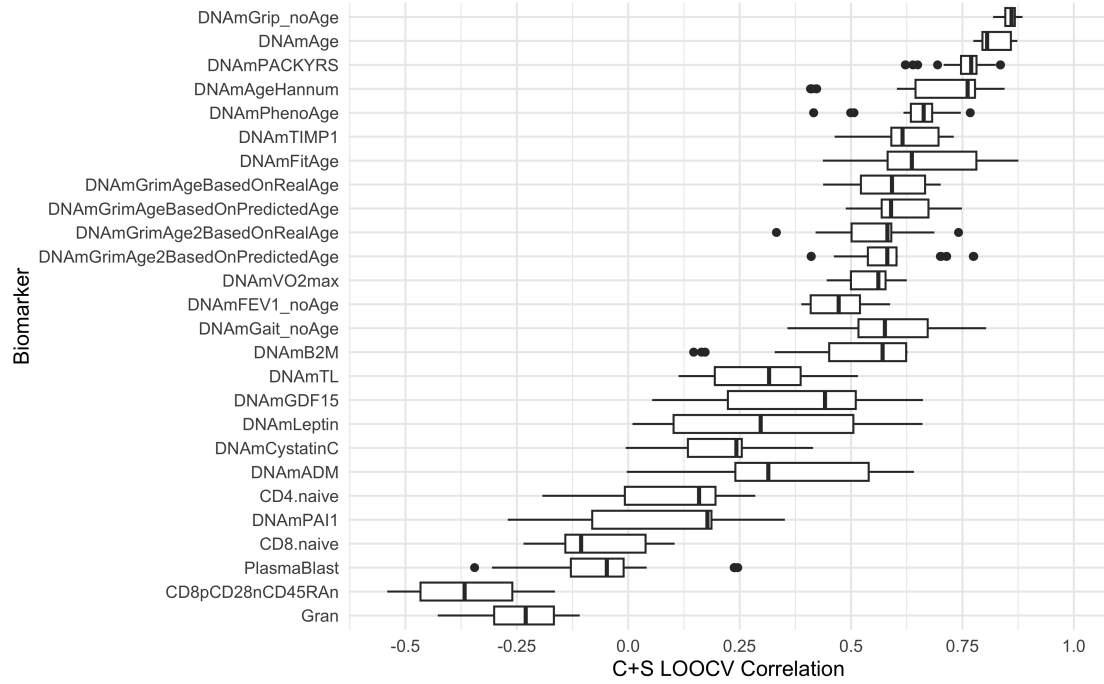
7.1 Epigen.TL.Lasso Function

`Epigen.TL.Lasso` performs Transfer Learning Lasso allowing either the Auxiliary Dataset Information to be known (Oracle or Oracle 1df) or estimated in the process (Estimate A0).

As necessary input, X_matrix is the matrix of covariates (nxp) concatenated between the Target and all Auxiliary Datasets, in that order. n should therefore be n_0 + n_1 + ... + n_k. X_matrix needs to be a matrix and not a dataframe to allow for matrix multiplication to be carried out in the function. Y_vector is the outcome vector to develop the TL Lasso model to estimate. It should align with your X_matrix with Target and Auxiliary outcomes concatenated and be a nx1 vector. N_vector is a vector of number of observations in the Target and each Auxiliary datasets in the order they are concatenated. AuxInformation is either "Estimate A0", "Oracle", or "Oracle 1df" to signify the informative auxiliary datasets need estimated (EstA0) or they should be treated as known. If known, they can either be treated as equally informative and combined into 1 dataset (Oracle), or they can be treated individually (Oracle 1df). RhatCount is either "n0/3" or a value between 1, ..., p to specify how many marginal correlations to consider when calculating information. This should be set to NULL if AuxInformation is "Oracle" or "Oracle 1df". LambdaType is either "Constant" or



Supplemental Figure 1. Median Absolute Percent Error (MeAPE) between True and Estimated Blood DNAm Biomarkers by top Performing TL Methods, Lasso, and Saliva Surrogates. Median MeAPE presented as dotted line with a change in X axis scaling. LODO MeAPE presented for C+S, C, and Lasso methods.



Supplemental Figure 2. Boxplots of C+S Method LODO Correlation by DNAm Biomarker to demonstrate variability in C+S method parameterizations.

"CV" to indicate if the lambda calculated from the target dataset should be used and adjusted based on the aux dataset size ("Constant") or whether the optimal lambda should be calculated via cross validation ("CV") for each set of auxiliary information. Default is "CV". seedstart is the seed to set in the calculation for reproducibility. If not specified, it is set to 123.

This function will return a data.frame with TL coefficients in the columns. The first column, "Variable" labels the intercept and columns from your X matrix that coefficient values correspond to. Columns 2:7 or 2:4 have the final coefficients with slight variations in their calculation. "min" and "1se" correspond to the lambda that either minimizes CV error or is at most 1se above it, respectively. "allcoef", "halfcoef", and "lambcoef" correspond to the coefficient thresholding used before combining coefficients across the auxiliary and target dataset. If "Constant" LambdaType was specified, only the minimum lambda from the target data is used and so "1se" coefficients are not presented.

```
Epigen.TL.Lasso <- function(X_matrix, Y_vector, N_vector,
                            AuxInformation, RhatCount = NULL, LambdaType = "CV",
                            seedstart = 123) {

  if(AuxInformation %!in% c("Estimate A0", "Oracle", "Oracle 1df")){
    stop("You must specify a method to capture auxiliary data informativeness.
         Options include 'Estimate A0', 'Oracle', or 'Oracle 1df'")
  }

  if(AuxInformation == "Estimate A0" & is.null(RhatCount)){
    stop("You must provide the number of Rhats to use when calculating
         auxiliary data informativeness.
        Options include any integer up to the column size of X_matrix OR 'n0/3'")
  }

  if(AuxInformation == "Estimate A0" & !is.null(RhatCount)){
    if(RhatCount > dim(X_matrix)[2]){
      stop("Rhat must be smaller than the number of columns in X.
         Please specify any integer up to the column size of X_matrix OR 'n0/3'")
    }

    if(RhatCount == 0){
     stop("Rhat must be a positive number from 1, ..., column size of X_matrix.")
    }
  }

  if(LambdaType %!in% c("CV", "Constant")){
    stop("LambdaType must be either 'CV' or 'Constant' to specify which
         lambda parameter is used for the auxiliary data.")
  }

  if(is.null(X_matrix)){
    stop("Please supply the X matrix")
  }
  if(is.null(Y_vector)){
```

```
    stop("Please supply the outcome")
  }
  if(is.null(N_vector)){
    stop("Please supply the N vector specifying the number of observations
         in target and auxiliary datasets")
  }

  if(sum(N_vector) != dim(X_matrix)[1]){
    stop("N_vector and X_matrix do not have the same number of observations")
  }

  if(sum(N_vector) != length(Y_vector)){
    stop("N_vector and Y do not have the same number of observations")
  }

  if(!is.null(RhatCount)){
    if(RhatCount == 'n0/3'){
        # set before calling the other functions.
        RhatCount <- round(N_vector[1] / 3)
    }
  }


    # TransLasso for Estimating informative set
    if(AuxInformation == "Estimate A0"){
      # for reproducibility
      set.seed(seedstart)

      Translasso_output <- TransLasso.EstA0(X = X_matrix, y = Y_vector,
                            n.vec = N_vector, AuxInformation = AuxInformation,
                            LambdaType = LambdaType, RhatCount = RhatCount)
    }
    # TransLasso for Oracle or Oracle 1df
    if(AuxInformation != "Estimate A0"){
      # for reproducibility
      set.seed(seedstart)

      Translasso_output <- TransLasso.Oracles(X = X_matrix, y = Y_vector,
                            n.vec = N_vector, AuxInformation = AuxInformation,
                            LambdaType = LambdaType)
    }

    # calculate intercepts
    intercept_beta_min <- mean(Y_vector - X_matrix %*%
                                 Translasso_output$beta.hat_min,
                               na.rm=TRUE)
    intercept_beta_1se <- mean(Y_vector - X_matrix %*%
                                 Translasso_output$beta.hat_1se,
```

```
                                na.rm=TRUE)

    intercept_beta_min_lambda <- mean(Y_vector -
                        X_matrix %*% Translasso_output$beta.hat_min_lambda,
                                     na.rm=TRUE)
    intercept_beta_1se_lambda <- mean(Y_vector -
                        X_matrix %*% Translasso_output$beta.hat_1se_lambda,
                                     na.rm=TRUE)

    intercept_beta_min_halflambda <- mean(Y_vector -
                        X_matrix %*% Translasso_output$beta.hat_min_halflambda,
                                    na.rm=TRUE)
    intercept_beta_1se_halflambda <- mean(Y_vector -
                        X_matrix %*% Translasso_output$beta.hat_1se_halflambda,
                                    na.rm=TRUE)

    # keep all coefficients.
    TL_beta_coef <- data.frame(Variable = c("Intercept", colnames(X_matrix)),
                        beta_min_allcoef = c(intercept_beta_min,
                        Translasso_output$beta.hat_min),
                        beta_1se_allcoef = c(intercept_beta_1se,
                        Translasso_output$beta.hat_1se),
                        beta_min_halfcoef = c(intercept_beta_min_halflambda,
                        Translasso_output$beta.hat_min_halflambda),
                        beta_1se_halfcoef = c(intercept_beta_1se_halflambda,
                        Translasso_output$beta.hat_1se_halflambda),
                        beta_min_lambcoef = c(intercept_beta_min_lambda,
                        Translasso_output$beta.hat_min_lambda),
                        beta_1se_lambcoef = c(intercept_beta_1se_lambda,
                        Translasso_output$beta.hat_1se_lambda))

    # if Constant, only the min lambda is used.
    if(LambdaType == "Constant"){
        TL_beta_coef <- data.frame(Variable = c("Intercept", colnames(X_matrix)),
                             beta_min_allcoef = c(intercept_beta_min,
                                Translasso_output$beta.hat_min),
                            beta_min_halfcoef = c(intercept_beta_min_halflambda,
                                Translasso_output$beta.hat_min_halflambda),
                             beta_min_lambcoef = c(intercept_beta_min_lambda,
                                Translasso_output$beta.hat_min_lambda))
    }

  # return Final TL coefficients as a dataframe
  return(TL_beta_coef)

}
```

7.2 `Saliva.2.Blood.DNAmBiomarkers` Function

`Saliva.2.Blood.DNAmBiomarkers` functions calculates the blood DNAm Biomarkers from methylation values using either the C or C+S method algorithms.

As input, X is the matrix or dataframe of saliva DNA methylation beta values with samples in rows and methylation sites in columns. To see the list of CpG loci needed for computation, please call colnames to CS_Algorithms_GitHub or C_Algorithms_GitHub (which are loaded in the global environment). method should be either "C+S" or "C" to indicate which set of algorithms you are interested in calculating. If "C+S", you must also provide SalivaDNAmBiom. SalivaDNAmBiom should be the matrix or dataframe of Saliva DNAm biomarkers (ie DNAm biomarkers directly calculated with saliva methylation values) if using the C+S algorithms. Otherwise, this should be NULL. Default is NULL.

This function outputs a dataframe with predicted DNAm Biomarkers in each column with rows in the same order as the originally supplied X.

```
Saliva.2.Blood.DNAmBiomarkers <- function(X, method, SalivaDNAmBiom = NULL) {

  # make sure ppl provide the saliva DNAm biomarkers if its C+S
  if (method == "C+S" && is.null(SalivaDNAmBiom)) {
    stop("For method C+S, saliva DNAm Biomarker matrix must be provided.")
  }


  if (method == "C") {
    TLcoeffMatrix <- C_Algorithms_GitHub
    Varstart <- 2
  } else if (method == "C+S") {
    TLcoeffMatrix <- CS_Algorithms_GitHub
    Varstart <- 3
  } else {
    stop("Invalid method specified. Please specify either 'C' or 'C+S'")
  }


 cpgs_needed <- TLcoeffMatrix$Variable[Varstart:length(TLcoeffMatrix$Variable)]
  if(sum(cpgs_needed %!in% colnames(X)) > 0){
    cpgs_missing <<- cpgs_needed[cpgs_needed %!in% colnames(X)]
    stop("Not all CpG columns are present in X for this prediction.
         Please see cpgs_missing for a list of missing but necessary CpGs.")
  }


  X_keep <- X[, cpgs_needed]
  if (any(is.na(X_keep))) {
    stop("Missing values are not allowed.
            Please impute missing values in the X matrix.")
  }

  ### Now can start the computations ###
```

```
  new_biomarker_list <- colnames(TLcoeffMatrix)[-1]

  n_biom <- dim(TLcoeffMatrix)[2]
  n_var <- dim(TLcoeffMatrix)[1]

  # make temp dataset with intercept, saliva, and columns in correct order
  if (method == "C"){
    newdata_temp <- data.frame(intercept = 1,
                               X[, TLcoeffMatrix$Variable[Varstart:n_var]])
  }else{
    newdata_temp <- data.frame(intercept = 1, SalivaDNAmBiom = NA,
                               X[, TLcoeffMatrix$Variable[Varstart:n_var]])
  }

  # Turn Coefficient matrix into a real matrix for multiplying
  TLcoeffMatrix2 <- matrix(unlist(TLcoeffMatrix[, 2:n_biom]),
                           ncol = (n_biom-1), nrow = n_var)

  # initialize predictions dataframe
  pred_df <- data.frame(rep(NA, dim(X)[1]))
  k <- 1

  for (i in new_biomarker_list) {

    if(method == "C+S"){
      # set Saliva value to the biomarker of interest
      newdata_temp$SalivaDNAmBiom <- SalivaDNAmBiom[, i]
    }

    # get the column of the TL coef matrix the biomarker is in
    TL_coef_Col <- (which(colnames(TLcoeffMatrix) == i) - 1)

    # turn into matrix for multiplication
    newdata_temp2 <- matrix(unlist(newdata_temp), ncol = dim(newdata_temp)[2],
                            nrow = dim(newdata_temp)[1])

    # data has ppl in rows, variables in columns,
    # TLcoef matrix has coefficients for each biomarker in a column
    cur_pred <- newdata_temp2 %*% TLcoeffMatrix2[, TL_coef_Col]

    # set prediction to the dataframe
    pred_df[, k] <- c(cur_pred)
    colnames(pred_df)[k] <- i

    # make k go up
    k <- k + 1
```

```
    # remove values so we don't repeat by accident
    rm(cur_pred); rm(newdata_temp2)

    if(method == "C+S"){
      # set Saliva value to NA so we don't repeat by accident
      newdata_temp$SalivaDNAmBiom <- NA
    }

  } # end of for loop in biomarker list

  # return the predicted DNAm values.
  if(method == "C+S"){
    colnames(pred_df) <- paste0(colnames(pred_df), "_CS_Pred")
  } else{
    colnames(pred_df) <- paste0(colnames(pred_df), "_C_Pred")
  }

  return(pred_df)
}
```

7.3 TL_Lasso Function

`TL_Lasso` is the actual Transfer Learning Lasso loop process called inside both `TL.Lasso.EstA0` and `TL.Lasso.Oracles`. As input, X is the matrix of covariates, y is the outcome vector, A0 is the informative Aux Set. It is either NULL or estimated. n.vec is a vector of number of observations in the target and aux datasets in that order, lam.const is whether we are calculating the optimal lambda via CV in each informative aux set or the constant value if using the methods outlined in TransLasso paper.

It outputs the estimated beta coefficients with and without thresholding (when performed in auxiliary data) and the estimated lambda values.

```
TL_Lasso <- function(X, y, A0, n.vec, lam.const=NULL, lam.const_1se = NULL, ...){

  p <- ncol(X)
  size.A0 <- length(A0) # set to NULL so its 0

  if(size.A0 > 0){ # only for Aux data, otherwise SKIP to below

    ind.kA <- ind.set(n.vec, c(1, A0+1))
    ind.1 <- 1:n.vec[1] # vector of all values to build initial model.

    y.A <- y[ind.kA]

   # if null, CV done for each Informative Set and both min and 1se lambda kept.
    if(is.null(lam.const)){
    # this gets run on first run because we have it set to NULL
      # does its own grid search for lambda
      cv.init<-cv.glmnet(X[ind.kA,], y.A, nfolds=8)
    # now, it will just take whatever the best value was and calculate the constant.
```

```
    lam.const <- cv.init$lambda.min/sqrt(2*log(p)/length(ind.kA))
    lam.const_1se <- cv.init$lambda.1se/sqrt(2*log(p)/length(ind.kA))
  }
  if(!is.null(lam.const) & is.null(lam.const_1se)){
    lam.const_1se <- lam.const
  }

  # w.kA = coefficients from Xk predicts Yk
  w.kA_min <- as.numeric(glmnet(X[ind.kA,], y.A,
                  lambda=lam.const*sqrt(2*log(p)/length(ind.kA)))$beta)
  w.kA_1se <- as.numeric(glmnet(X[ind.kA,], y.A,
                  lambda=lam.const_1se*sqrt(2*log(p)/length(ind.kA)))$beta)

  # w.k coefficient thresholding
  w.kA_min_halflambda <- w.kA_min*(abs(w.kA_min) >=
                          0.5*lam.const*sqrt(2*log(p)/length(ind.kA)))
  w.kA_min_lambda <- w.kA_min*(abs(w.kA_min) >=
                           lam.const*sqrt(2*log(p)/length(ind.kA)))

  w.kA_1se_halflambda <- w.kA_1se*(abs(w.kA_1se) >=
                          0.5*lam.const_1se*sqrt(2*log(p)/length(ind.kA)))
  w.kA_1se_lambda <- w.kA_1se*(abs(w.kA_1se) >=
                          lam.const_1se*sqrt(2*log(p)/length(ind.kA)))

  # build model in target where outcome is what is left after
  # taking the yhat from kth model coefficients.
  # delta.kA = coefficients from X0 predicts (Y0 - Y0hat from w.kA)
delta.kA_min <- as.numeric(glmnet(x=X[ind.1,],y=y[ind.1]-X[ind.1,]%*%w.kA_min,
                       lambda=lam.const*sqrt(2*log(p)/length(ind.1)))$beta)
delta.kA_1se <- as.numeric(glmnet(x=X[ind.1,],y=y[ind.1]-X[ind.1,]%*%w.kA_1se,
                     lambda=lam.const_1se*sqrt(2*log(p)/length(ind.1)))$beta)

  # delta.k coefficient thresholding
  delta.kA_min_halflambda <- delta.kA_min*(abs(delta.kA_min) >=
                              0.5*lam.const*sqrt(2*log(p)/length(ind.1)))
  delta.kA_min_lambda <- delta.kA_min*(abs(delta.kA_min) >=
                              lam.const*sqrt(2*log(p)/length(ind.1)))

  delta.kA_1se_halflambda <- delta.kA_1se*(abs(delta.kA_1se) >=
                              0.5*lam.const_1se*sqrt(2*log(p)/length(ind.1)))
  delta.kA_1se_lambda <- delta.kA_1se*(abs(delta.kA_1se) >=
                              lam.const_1se*sqrt(2*log(p)/length(ind.1)))

  # final beta coefficients (the kth lasso coefficients are the weights
  # from kth lasso model + kth lasso on target)

  # no thresholding
  beta.kA_min <- w.kA_min + delta.kA_min
```

```
    beta.kA_1se <- w.kA_1se + delta.kA_1se

    # half lambda thresholding
    beta.kA_min_halflambda <- w.kA_min_halflambda + delta.kA_min_halflambda
    beta.kA_min_lambda <- w.kA_min_lambda + delta.kA_min_lambda

    # lambda thresholding
    beta.kA_1se_halflambda <- w.kA_1se_halflambda + delta.kA_1se_halflambda
    beta.kA_1se_lambda <- w.kA_1se_lambda + delta.kA_1se_lambda

   lam.const=NULL # reset lambda because we don't want it to loop through as !null
    # first auxillary dataset because we supply the lambda constant.

    # output all coefficients
    # recall if constant lambda was specified, min and 1se will be identical.
    list(beta.kA_min = as.numeric(beta.kA_min), w.kA_min=w.kA_min,
       beta.kA_1se = as.numeric(beta.kA_1se), w.kA_1se=w.kA_1se,
       beta.kA_min_halflambda = as.numeric(beta.kA_min_halflambda),
            w.kA_min_halflambda=w.kA_min_halflambda,
       beta.kA_1se_halflambda = as.numeric(beta.kA_1se_halflambda),
            w.kA_1se_halflambda=w.kA_1se_halflambda,
       beta.kA_min_lambda = as.numeric(beta.kA_min_lambda),
            w.kA_min_lambda=w.kA_min_lambda,
       beta.kA_1se_lambda = as.numeric(beta.kA_1se_lambda),
            w.kA_1se_lambda=w.kA_1se_lambda,
       lam.const=lam.const, lam.const_1se=lam.const_1se)
  }else{ # end of if(size.A0 > 0)

    # BEGINNING CODE FOR INITIAL / TARGET DATA
    cv.init <- cv.glmnet(X[1:n.vec[1],], y[1:n.vec[1]], nfolds=8)

    # When constant lambda selected, min lambda is used.
    lam.const <- cv.init$lambda.min/sqrt(2*log(p)/n.vec[1])
    lam.const_1se <- cv.init$lambda.1se/sqrt(2*log(p)/n.vec[1])

    # extract coefficients (excluding the intercept)
    beta.kA_min <- predict(cv.init, s='lambda.min', type='coefficients')[-1]
    beta.kA_1se <- predict(cv.init, s='lambda.1se', type='coefficients')[-1]
    w.kA_min <- w.kA_1se <- NA

    list(beta.kA_min = as.numeric(beta.kA_min), w.kA_min=w.kA_min,
       beta.kA_1se = as.numeric(beta.kA_1se), w.kA_1se=w.kA_1se,
       lam.const=lam.const, lam.const_1se = lam.const_1se)
  }

}
```

7.4 TransLasso.Oracles Function

TransLasso.Oracles performs Oracle Transfer Learning Lasso meaning the set of auxiliary datasets is specified to either run all at once (Oracle) or 1 dataset at a time (Oracle 1df). As input, X is the matrix of covariates, y is the outcome vector, n.vec is a vector of number of observations in the target and aux datasets in that order, AuxInformation is either "Oracle" or "Oracle 1df", LambdaType is either "Constant" or "CV" to indicate if the lambda calculated from the target dataset should be used and adjusted based on the aux dataset size ("Constant") or whether the optimal lambda should be calculated via cross validation ("CV") for each set of auxiliary information.

It outputs the aggregated coefficients at various parameterizations and the weights used when aggregating.

```
TransLasso.Oracles <- function(X, y, n.vec, AuxInformation = "Oracle 1df",
                               LambdaType = "CV", ...) {

  M = length(n.vec)-1
  p <- ncol(X)

  # row indices of where these observations are for target
  ind.1 <- ind.set(n.vec, 1)

  Tset <- list()

  if(AuxInformation == "Oracle"){
      # make Tset actually just all the aux datasets (for oracle all at once)
      # aux datasets start at 2.
      Tset[[1]] <- c(1:M)
  }
  # take 1 aux dataset at a time, noting that we index by dataset before.
  if(AuxInformation == "Oracle 1df"){
    for(kk in 1:M){ #use Rhat as the selection rule
     Tset[[kk]] <- kk
    } # the sets of aux datasets to take for ranking of datasets.
  }

  k0 = length(Tset)
  Tset <- unique(Tset)

  beta.T_min <- beta.T_min_lambda <- beta.T_min_halflambda <- list()
  beta.T_1se <- beta.T_1se_lambda <- beta.T_1se_halflambda <- list()

  # Lasso on Target Data only
  init.re <- TL_Lasso(X=X, y=y, A0=NULL, n.vec=n.vec)

  beta.T_min[[1]] <- init.re$beta.kA_min
  beta.T_1se[[1]] <- init.re$beta.kA_1se

  # if constant lambda specified, it is here.
  c1_lambda_const <- init.re$lam.const
```

```
  c1_lambda_const_1se <- init.re$lam.const_1se

  og_lasso_coef_min <- init.re$beta.kA_min
  og_lasso_coef_1se <- init.re$beta.kA_1se


  beta.T_min_lambda <- beta.T_min_halflambda <- beta.T_min
  beta.T_1se_lambda <- beta.T_1se_halflambda <- beta.T_1se

  # go through TL Lasso for each informative set
  for(kk in 1:length(Tset)){
    T.k <- Tset[[kk]]

    # changed function call to lam.const = NULL to do Aux CV
    if(LambdaType == "CV"){
      re.k <- TL_Lasso(X=X, y=y, A0=T.k, n.vec=n.vec, lam.const = NULL)
    }else{ # constant lambda specification
     re.k <- TL_Lasso(X=X, y=y, A0=T.k, n.vec=n.vec, lam.const = c1_lambda_const,
                      lam.const_1se = c1_lambda_const_1se)
    }

    # extract coefficients for each informative auxiliary set
    beta.T_min[[kk+1]] <- re.k$beta.kA_min
    beta.pool.T_min[[kk+1]] <- re.k$w.kA_min

    beta.T_1se[[kk+1]] <- re.k$beta.kA_1se
    beta.pool.T_1se[[kk+1]] <- re.k$w.kA_1se

    beta.T_min_halflambda[[kk+1]] <- re.k$beta.kA_min_halflambda
    beta.pool.T_min_halflambda[[kk+1]] <- re.k$w.kA_min_halflambda

    beta.T_1se_lambda[[kk+1]] <- re.k$beta.kA_1se_lambda
    beta.pool.T_1se_lambda[[kk+1]] <- re.k$w.kA_1se_lambda
  }

  beta.T_min <- beta.T_min[!duplicated((beta.T_min))]
  beta.T_min <- as.matrix(as.data.frame(beta.T_min))

  beta.T_1se <- beta.T_1se[!duplicated((beta.T_1se))]
  beta.T_1se <- as.matrix(as.data.frame(beta.T_1se))

  beta.T_min_halflambda <- beta.T_min_halflambda[
                           !duplicated((beta.T_min_halflambda))]
  beta.T_min_halflambda <- as.matrix(as.data.frame(beta.T_min_halflambda))
  beta.T_min_lambda <- beta.T_min_lambda[!duplicated((beta.T_min_lambda))]
  beta.T_min_lambda <- as.matrix(as.data.frame(beta.T_min_lambda))

  beta.T_1se_halflambda <- beta.T_1se_halflambda[
```

```
                                !duplicated((beta.T_1se_halflambda))]
  beta.T_1se_halflambda <- as.matrix(as.data.frame(beta.T_1se_halflambda))
  beta.T_1se_lambda <- beta.T_1se_lambda[!duplicated((beta.T_1se_lambda))]
  beta.T_1se_lambda <- as.matrix(as.data.frame(beta.T_1se_lambda))

  ## aggregate coefficients using squared error.
  ## aggregate w.kA and delta.kA
  agg.re1_min <- coef.aggr(B= beta.T_min, X = X, y = y, N_vector = n.vec)
  agg.re1_1se <- coef.aggr(B= beta.T_1se, X = X, y = y, N_vector = n.vec)

  agg.re1_min_halflambda <- coef.aggr(B= beta.T_min_halflambda, X = X, y = y,
                                      N_vector = n.vec)
  agg.re1_1se_halflambda <- coef.aggr(B= beta.T_1se_halflambda, X = X, y = y,
                                      N_vector = n.vec)

  agg.re1_min_lambda <- coef.aggr(B= beta.T_min_lambda, X = X,
                            y = y, N_vector = n.vec)
  agg.re1_1se_lambda <- coef.aggr(B= beta.T_1se_lambda, X = X,
                            y = y, N_vector = n.vec)



  return(list(beta.hat_min = agg.re1_min$beta, theta.hat_min = agg.re1_min$theta,
              beta.hat_1se = agg.re1_1se$beta, theta.hat_1se = agg.re1_1se$theta,
               beta.hat_min_halflambda = agg.re1_min_halflambda$beta,
               theta.hat_min_halflambda = agg.re1_min_halflambda$theta,
               beta.hat_1se_halflambda = agg.re1_1se_halflambda$beta,
               theta.hat_1se_halflambda = agg.re1_1se_halflambda$theta,
               beta.hat_min_lambda = agg.re1_min_lambda$beta,
               theta.hat_min_lambda = agg.re1_min_lambda$theta,
               beta.hat_1se_lambda = agg.re1_1se_lambda$beta,
               theta.hat_1se_lambda = agg.re1_1se_lambda$theta,
               c1_lambda_const = c1_lambda_const))
}
```

7.5 `TransLasso.EstA0` Function

`TransLasso.EstA0` performs Transfer Learning Lasso while estimating the Informative Auxiliary Data. X is the matrix of covariates (nxp), y is the outcome vector (nx1), n.vec is a vector of number of observations in the target and aux datasets in that order, RhatCount is either "n0/3" or a value between 1, ..., p to specify how many marginal correlations to consider when calculating information. LambdaType is either "Constant" or "CV" to indicate if the lambda calculated from the target dataset should be used and adjusted based on the aux dataset size ("Constant") or whether the optimal lambda should be calculated via cross validation ("CV") for each set of auxiliary information.

It outputs the aggregated coefficients at various parameterizations and the weights used when aggregating.

```
TransLasso.EstA0 <- function(X, y, n.vec, RhatCount, LambdaType, ...){
```

```
# count of aux datasets
M = length(n.vec)-1

Rhat <- rep(0, M+1)
p <- ncol(X)

# make row indices of where target observations are
ind.1 <- ind.set(n.vec, 1)

# calculate informativeness for each aux study
for(k in 2: (M+1)){
  ind.k <- ind.set(n.vec, k) # row indices for kth aux sample.

 # calculate difference in marginal correlations between k aux and target data.
 Xty.k <- t(X[ind.k, ])%*%y[ind.k] / n.vec[k] - t(X[ind.1,])%*%y[ind.1]/ n.vec[1]

  # Rhat adjusts how many marginal correlations are looked at
  # take the top largest correlation differences based on RhatCount
  margin.T <- sort(abs(Xty.k), decreasing=T)[1:RhatCount]

  # estimated sparse index for kth aux sample.
  Rhat[k] <- sum(margin.T^2)
}

Tset <- list()
k0 = 0
# get ordering of smallest to largest Rhat for aux samples.
kk.list <- unique(rank(Rhat[-1]))

for(kk in 1:length(kk.list)){#use Rhat as the selection rule
  Tset[[k0+kk]] <- which(rank(Rhat[-1]) <= kk.list[kk])
} # the sets of aux datasets to take for each ranking of datasets to include.

k0 = length(Tset)
Tset <- unique(Tset)

beta.T_min <- beta.T_min_lambda <- beta.T_min_halflambda <- list()
beta.T_1se <- beta.T_1se_lambda <- beta.T_1se_halflambda <- list()

# Lasso on Target Data only
init.re <- TL_Lasso(X=X, y=y, A0=NULL, n.vec=n.vec)

beta.T_min[[1]] <- init.re$beta.kA_min
beta.T_1se[[1]] <- init.re$beta.kA_1se

# if constant lambda specified, it is here.
c1_lambda_const <- init.re$lam.const
```

```
  c1_lambda_const_1se <- init.re$lam.const_1se

  og_lasso_coef_min <- init.re$beta.kA_min
  og_lasso_coef_1se <- init.re$beta.kA_1se

  beta.T_min_lambda <- beta.T_min_halflambda <- beta.T_min
  beta.T_1se_lambda <- beta.T_1se_halflambda <- beta.T_1se

  # go through TL Lasso for each informative set
  for(kk in 1:length(Tset)){
    T.k <- Tset[[kk]]

    # which lambda type changes which section of function call it goes into
    if(LambdaType == "CV"){
      re.k <- TL_Lasso(X=X, y=y, A0=T.k, n.vec=n.vec, lam.const = NULL)
    }
    if(LambdaType == "Constant"){
      re.k <- TL_Lasso(X=X, y=y, A0=T.k, n.vec=n.vec,
                           lam.const = c1_lambda_const,
                           lam.const_1se = c1_lambda_const_1se)
    }

    # extract coefficients for each informative auxiliary set
    beta.T_min[[kk+1]] <- re.k$beta.kA_min
    beta.T_1se[[kk+1]] <- re.k$beta.kA_1se
    beta.T_min_halflambda[[kk+1]] <- re.k$beta.kA_min_halflambda
    beta.T_min_lambda[[kk+1]] <- re.k$beta.kA_min_lambda
    beta.T_1se_lambda[[kk+1]] <- re.k$beta.kA_1se_lambda
    beta.T_1se_halflambda[[kk+1]] <- re.k$beta.kA_1se_halflambda
  }

  beta.T_min <- beta.T_min[!duplicated((beta.T_min))]
  beta.T_min <- as.matrix(as.data.frame(beta.T_min))

  beta.T_1se <- beta.T_1se[!duplicated((beta.T_1se))]
  beta.T_1se <- as.matrix(as.data.frame(beta.T_1se))

  beta.T_min_halflambda <- beta.T_min_halflambda[
                             !duplicated((beta.T_min_halflambda))]
  beta.T_min_halflambda <- as.matrix(as.data.frame(beta.T_min_halflambda))
  beta.T_min_lambda <- beta.T_min_lambda[!duplicated((beta.T_min_lambda))]
  beta.T_min_lambda <- as.matrix(as.data.frame(beta.T_min_lambda))

  beta.T_1se_halflambda <- beta.T_1se_halflambda[
                          !duplicated((beta.T_1se_halflambda))]
  beta.T_1se_halflambda <- as.matrix(as.data.frame(beta.T_1se_halflambda))
  beta.T_1se_lambda <- beta.T_1se_lambda[!duplicated((beta.T_1se_lambda))]
  beta.T_1se_lambda <- as.matrix(as.data.frame(beta.T_1se_lambda))
```

```
  ## aggregate coefficients using squared error.
  # No coef thresholding
  agg.re1_min <- coef.aggr(B= beta.T_min, X = X, y = y, N_vector = n.vec)
  agg.re1_1se <- coef.aggr(B= beta.T_1se, X = X, y = y, N_vector = n.vec)

  # half lambda coef thresholding
  agg.re1_min_halflambda <- coef.aggr(B= beta.T_min_halflambda, X = X, y = y,
                                      N_vector = n.vec)
  agg.re1_1se_halflambda <- coef.aggr(B= beta.T_1se_halflambda, X = X, y = y,
                                      N_vector = n.vec)

  # lambda coef thresholding
  agg.re1_min_lambda <- coef.aggr(B= beta.T_min_lambda, X = X,
                    y = y, N_vector = n.vec)
  agg.re1_1se_lambda <- coef.aggr(B= beta.T_1se_lambda, X = X,
                    y = y, N_vector = n.vec)


  # theta are the returned weights of each dataset.
  # betas are the final, aggregated coefficients for potential covariates
  return(list(beta.hat_min = agg.re1_min$beta, theta.hat_min = agg.re1_min$theta,
              beta.hat_1se = agg.re1_1se$beta, theta.hat_1se = agg.re1_1se$theta,
               beta.hat_min_halflambda = agg.re1_min_halflambda$beta,
               theta.hat_min_halflambda = agg.re1_min_halflambda$theta,
               beta.hat_1se_halflambda = agg.re1_1se_halflambda$beta,
               theta.hat_1se_halflambda = agg.re1_1se_halflambda$theta,
               beta.hat_min_lambda = agg.re1_min_lambda$beta,
               theta.hat_min_lambda = agg.re1_min_lambda$theta,
               beta.hat_1se_lambda = agg.re1_1se_lambda$beta,
               theta.hat_1se_lambda = agg.re1_1se_lambda$theta,
               c1_lambda_const = c1_lambda_const,
               og_lasso_coef_min = og_lasso_coef_min,
               og_lasso_coef_1se = og_lasso_coef_1se))
}
```

7.6 Helper Functions

`coef.aggr` function aggregates the coefficients from target and auxiliary data using the error in the target data. This function is called in the background; it is not being called directly by users.

As input, this function requires B, the coefficient vector, X, the matrix of covariates, y, the outcome vector, and N_vector, a vector of number of observations in the target and aux datasets in that order. It outputs a two item list, with the first item, theta, being the weights used to aggregate the coefficients, and beta, the final, aggregated coefficients.

```
coef.aggr <- function(B, X, y, N_vector){

  # if all coefficients are zero, just return zero
  if(sum(B == 0) == ncol(B)*nrow(B)){
```

```
    return(rep(0,nrow(B)))
  }
  p <- nrow(B)
  K <- ncol(B)
  colnames(B) <- NULL

    # Take the difference in target y and predicted y
    # from each aux data coefficients
    y0hatk <- -log(colSums(y[1:N_vector[1]] - X[1:N_vector[1], ] %*% B)^2)
    theta.hat <- exp(y0hatk)
    theta.hat = theta.hat / sum(theta.hat)
    # weights by fraction of total squared error

    # multiply betak by weights for each kth aux
    beta <- as.numeric(B%*%theta.hat)

  list(theta = theta.hat, beta = beta)

}
```

ind.set tells the TL functions where the first and last observation is for each dataset (target, aux 1, ..., aux k) based on the sample sizes in n.vec. n.vec is the vector of number of observations in the target and aux datasets in that order. k is index of the dataset we are interested in extracting values from. It returns the indices in the X matrix and y vector to extract.

```
ind.set <- function(n.vec, k.vec){
  ind.re <- NULL
  for(k in k.vec){
    if(k==1){
      ind.re<-c(ind.re,1: n.vec[1])
    }else{
      ind.re<- c(ind.re, (sum(n.vec[1:(k-1)])+1): sum(n.vec[1:k]))
    }
  }
  ind.re
}
```

Received June 2024