Variable Selection and Regression Analysis of Smoking Cessation Outcomes Using Regularization: A Reanalysis of the BASC-Varenicline Trial Data

Thomas B. Arnold

Abstract

Background: Individuals with past or present major depressive disorder (MDD) experience unique barriers to smoking cessation. While varenicline has been shown to be more effective than traditional nicotine replacement therapy in achieving abstinence for individuals with and without MDD, its lower effectiveness for those with MDD suggests that MDD-responsive treatment strategies are necessary. A recent determined that varenicline improved abstinence, but BASC did not, either with or without varenicline. In light of this surprising result, we use data from this study to examine baseline variables as potential predictors of end-of-treatment abstinence and moderators of behavioral treatment's effect thereon.

Methods: We examine data from a randomized, placebo-controlled 2x2 factorial design study of 300 smokers with past or present MDD, who received either placebo or varenicline and standard treatment (ST) or behavioral activation for smoking cessation (BA). Missing data was imputed through multiple imputation. A cross-validated Lasso was applied at the optimally chosen value of λ for coefficient estimates. Subsequently a bootstrap procedure is carried out to estimate confidence intervals and assess significance of the coefficients.

Results: The participant's Fagerstrom Test for Cigarette Dependence score, complementary reinforcers (i.e., pleasurable events associated with smoking), and the participant's Nicotine Metabolite Ratio were all baseline predictors of abstinence controlling for treatment type. The models demonstrated fair to good discriminative ability. Our moderator analysis demonstrated that no variables could be identified as potential moderators of the therapy based treatment (BA or ST).

Conclusion: Lasso regression with bootstrap confidence interval estimates identified meaningful predictors of abstinence noted above. We were unable to identify treatment effect moderators. Limitations include the relatively small sample size, which led to convergence issues and challenges handling class imbalance in the outcome which suggests these findings should be validated in larger studies. Future research might benefit from more sophisticated regularization approaches and larger, more diverse samples to better understand treatment effect heterogeneity in this population.

Introduction

More than 30% of individuals with major depressive disorder (MDD) are daily smokers (Han et al. 2022; Smith et al. 2020; Weinberger et al. 2020). These individuals experience certain characteristic barriers to smoking cessation. Compared to smokers without MDD, they tend to be more likely to smoke heavily, to experience greater dependence on smoking, and to consider smoking to be more rewarding than other activities; it may thus be unsurprising that smokers with MDD experience more severe withdrawal symptoms (Breslau, Kilbey, and Andreski 1992; Spring, Pingitore, and McChargue 2003; Weinberger, Desai, and McKee 2010; Lyons et al. 2008). Smoking relapse is also associated with lower experience of reward and cognition, both of which tend to be impaired among those diagnosed with MDD (Leventhal et al. 2009; Cook et al. 2010; Patterson et al. 2009). Even past MDD adversely affects smoking cessation treatment outcomes (Hitsman, Papandonatos, et al. 2013).

Perhaps because of these challenges, individuals with MDD have typically been excluded from smoking treatment clinical trials (Hitsman et al. 2003; Hitsman, Papandonatos, et al. 2013; Talukder et al. 2023). Only six trials have explored smoking cessation in individuals with MDD (Evins et al. 2008; Anthenelli et al. 2013; Thorsteinsson et al. 2001; Hall et al. 2006; Minami et al. 2022; Hitsman et al. 2023), two of which had sample sizes of fewer than 50 participants (Thorsteinsson et al. 2001; Minami et al. 2022).

Additional research evaluating approaches to smoking cessation in this population is necessary. One such approach is treatment with varenicline, a pharmaceutical intervention shown to lessen cravings, withdrawal, withdrawal-related cognitive impairment, and reward from smoking (Patterson et al. 2009; Hitsman, Hogarth, et al. 2013; Perkins et al. 2010; Sofuoglu et al. 2009; West et al. 2008; McClure et al. 2012; Cinciripini et al. 2013). While cessation with varenicline was higher in smokers without any mental health disorders (Anthenelli et al. 2016), one trial of varenicline for individuals with MDD found greater abstinence versus placebo at 52 weeks (28.5% vs. 17.5%) (Anthenelli et al. 2013). However, individuals with mental health disorders, including MDD, are more likely to be prescribed nicotine replacement therapy, a comparatively less effective therapy, than varenicline (Anthenelli et al. 2016; Taylor et al. 2020). This hesitance to prescribing varenicline may stem from a prior boxed warning that, as of 2016, has been removed based on studies evincing the safety of varenicline for individuals with and without mental health disorders (Anthenelli et al. 2016).

The data in our study is derived from a recent randomized, placebo-controlled trial which examined whether the effectiveness of varenicline for smoking cessation in adults with current or past MDD is enhanced by behavioral activation for smoking cessation (BASC) (Hitsman et al. 2023). Behavioral activation (BA) increases reward experience and decreases avoidance-based coping (35-37 Cuijpers, Van Straten, and Warmerdam 2007; Dimidjian et al. 2011; Hopko et al. 2003; MacPherson et al. 2010). A prior pilot study indicated that individuals with elevated depression symptoms, but not MDD, had better cessation results at 26 weeks with BA and nicotine replacement therapy compared to those with standard behavioral treatment (ST) and nicotine replacement therapy (14.3% vs. 0%) (MacPherson et al. 2010).

The Hitsman study from which we draw our data uses a 2x2 factorial design, with 300 participants with past or present MDD treated, over 12 weeks, with BASC or ST alongside placebo or varenciline. Although the Hitsman study found that BASC did not outperform ST with respect to abstinence at 27 weeks, with or without varenicline, it did find that varenicline improved short- and longer-term abstinence compared to placebo (Hitsman et al. 2023). The dual goals of our analysis of the Hitsman et al. (2023) data will be to assess whether and how baseline variables (1) moderate behavioral treatment's effect on end-of-treatment (EOT) abstinence and (2) predict EOT abstinence, controlling for pharmaceutical and behavioral treatments.

In order to accomplish these dual goals, we use regularized linear models. Regularization, also known as shrinkage, reduces variance by fitting a model with all p predictors, with the estimated coefficients shrunken towards zero relative to the least square estimates (Hastie, Tibshirani, and Friedman (2009); pp. 204). We use this technique here because our goal can be understood as a variable selection problem. The Lasso method of regularization, which comepls some of the coefficients to be exactly equal to zero, can also be used for variable selection [Hastie, Tibshirani, and Friedman (2009); pp. 204, 219]. Best subset regression seeks the optimal subset of predictors by evaluating all possible combinations, thereby ensuring selection of the model with the lowest prediction error, although at a computational cost that increases exponentially with the number of predictors (Hazimeh and Mazumder 2020; Hastie, Tibshirani, and Wainwright 2015).

In contrast, L0+L2 regularization combines the sparsity-promoting L0 penalty, which counts the number of non-zero coefficients, with the L2 penalty that shrinks coefficients towards zero without zeroing them,

effectively balancing variable selection with the need to manage multicollinearity and enhance model accuracy (Hazimeh and Mazumder 2020; Hastie, Tibshirani, and Wainwright 2015; Paul 2024). The relaxed lasso, a modification of the Lasso method, initially applies the Lasso to determine a subset of variables by shrinking some coefficients to zero, then refits these selected variables with a reduced or absent Lasso penalty, potentially reducing bias while retaining the variable selection advantage of the original Lasso approach (Meinshausen 2007). L0+L2 and relaxed Lasso are considered the best trade-offs between variable selection and goodness of fit [Paul (2024); hazimeh2020fast]. Best subset, L0+L2, and relaxed Lasso are more computationally difficult than standard Lasso.

Another important extension of the Lasso regularization technique are those techniques which allow for specification of groups, order, or structure within the covariates such that penalties are applied with respect to the structure rather than individual covariates independently. These techniques include hierarchical Lasso (Bien, Taylor, and Tibshirani 2013), group Lasso [Meier, Van De Geer, and Bühlmann (2008); simon2012standardization], and sparse-group Lasso (Simon et al. 2013). This structural approach is particularly valuable when dealing with features that have natural groupings or hierarchical relationships, such as we examining complex interaction terms. As with he extensions of Lasso mentioned above, these techniques are less computationally efficient than standard Lasso. For this reason we opt to focus on application of the standard Lasso and leave extensions to modified Lasso regressions to future work.

Methods

Variables

First, we note that all covariates, other than abstinence (`abst`), are baseline variables. The following variables are binary: pharmacotherapy (`Var`); psychotherapy (`BA`); sex (`sex_ps`); indicators of whether the participant identifies as non-Hispanic white, Black, and/or Hispanic (`NHW`, `Black`, and `Hisp`); whether the individual smokes within five minutes of waking up (`ftcd.5.mins`); other lifetime DSM-5 diagnosis (`otherdiag`); whether the participant takes antidepressant medication (`antidepmed`); current or past MDD (`mde_curr`); and exclusive use of methylated cigarettes (`Only.Menthol`). The numeric variables are age (`age_ps`); cigarettes per day (`cpd_ps`); and Nicotine Metabolism Ratio (`NMR`). All remaining variables are ordinal. However, we treat income (`inc`) and education (`edu`) as ordinal, but (as we discuss below) treat the following variables as numeric: FTCD score (`ftcd_score`); `bdi_score_pq1`; cigarette reward value (`crv_total_pg1`); substitute reinforcer score (`hedonsum_n_pq1`); complementary reinforcer score (`hedonsum_y_pq1`); anhedonia (`shaps_score_pg1`); and readiness to quit smoking (`readiness`).

The `ftcd_score` is the participant's score on the Fagerstrom Test for Cigarette Dependence, a six-item survey intended to evaluate the quantity of cigarette consumption, dependence, and compulsion to smoke; answers are scored and summed to yield a total score of 0-10, with higher scores indicating a greater physical dependence on nicotine (Fagerström 2011). The `bdi_score_pq1` is the participant's score on the Beck Depression Inventory, or BDI-II, a 21-item survey scored from 0 to 63 measuring the severity of depression, with higher scores indicating greater severity (Beck, Steer, and Brown 1996).

The `crv_total_pq1` score measures the participant's score on a questionnaire measuring preference for smoking over other traditionally rewarding activities, measured on a scale of 0 to 15, where 1 point is added for each time the participant chooses smoking (Spring, Pingitore, and McChargue 2003). Substitute and complimentary reinforcer scores (`hedonsum_n_pq1` and `hedonsum_y_pg1`) come from a participant's score of the cross-product of frequency (measured from 0 to 2) and level of enjoyableness (0 to 2) for a 45-item version of the Pleasant Events Schedule (MacPhillamy and Lewinsohn 1982); based

on the participant's report of whether an event is or is not associated with smoking, it is designated a complementary or substitute reinforcer, the cross products of which are summed.

Anhedonia (`shaps_score_pq1`) is assessed with the 14-item Snaith-Hamilton Pleasure Scale (SHAPS), where 14 statements are ranked on a 4-point Likert scale, with higher scores indicating greater enjoyment of typically rewarding experiences (Snaith et al. 1995). Nicotine metabolism ratio, otherwise known as nicotine metabolite ratio (`NMR`), is calculated as the ratio of 3'hydroxycotinine (3HC) to cotinine; NMR is a biomarker where higher NMR is associated with faster metabolism of nicotine (Siegel et al. 2020).

Exploratory Data Analysis

The original paper uses a 2x2 factorial design. However, we choose to keep the treatment variables separate because the sample size is very small and further reducing the sample size to the four groups at issue would significantly limit the Lasso's ability to fit the data. That the 2x2 factorial design is not necessary to answer the research question validates our choice.

In addition to summary statistics available in Table 1 and the correlation matrix, we also inspect univariate histograms (omitted for space) for all variables. This reveals one non-normal variable, NMR, for which we consider transformations as noted below. Bivariate relationships are explored in Figure 1. Additionally we examine the distribution and association of all variables between levels of `abst`, `BA`, and `Var` (tables omitted for space).

When dividing the sample for each variable by `abst` we see that there is a significant difference between those who abstain and those who do no in the `Var`. This is because of the noted effectiveness of the treatment `Var`. We also observe marginal significance in the variable `ftcd_score` with abstainers having lower baseline scores. This could be a confounder, but makes logical sense as higher `ftcd_score` indicates greater dependency which would make it more difficult to abstain. There are no other significant differences in variables between groups of `abst`. Finally, when dividing the sample for each variable by `BA` we see that there are no significant differences in variables between groups of `BA`. Lastly, when dividing the sample for each variable by `Var` we see one significant difference in `abst`, essentially a mirror image of the difference discussed prior.

We note that certain survey variables, specifically `ftcd_score`, `crv_total_pq1`, `readiness`, `shaps_score_pq1`, `hedonsum_y_pq1`, and `hedonsum_n_pq1` are ordinal but will be treated here as either numeric integers or continuous for the purpose of the modeling. This presents a limitation in that we are treating ordinal variables as evenly spaced. Doing so is more or less justifiable for different variables. For example, we are more justified in treating `hedonsum_y_pq1` and `hedonsuum_n_pq1` as continuous, given that each score is made up of the product of two ordinal scales; by using the cross-product, the study authors have made some assumptions about relative scale that obfuscates the ordinal nature of the original.

Nevertheless, treating these variables as ordinal is untenable here, as it would make individual cell sizes incredibly small and lead to parameter proliferation in modeling. Another potential solution would be grouping ordinal variables into discernible levels, as with `inc`, discussed below, but this would take an in-depth understanding of the meaning and spacing for each score that we do not have at present. Said another way, grouping the variables without fully understanding how relative distance between ordinal levels is expressed would lead to our making similar assumptions about the spacing between levels as we make in treating the variables as continuous. We note this is consistent with the handling of survey score variables in Table 1 of the original paper.

To ameliorate the potential loss of accurate fit from treating ordinal variables as numeric, we use polynomial fit for each variable, allowing non-linearities between the log odds ratio for these variables and

the outcome. The default contrast handling for ordinal covariates in R is a polynomial contrast. Applying the polynomial score transformation approximates this relationship, without incurring the cost of a parameter for all but one level of the variable. We note that this polynomial relationship is important to the predictive power of the model. Further, these polynomial terms allow us to model and interpret predictions for the outcome across the full scale of ordinal covariates even when that relationship is non-linear. Future work could examine fitting a model without these polynomial terms, which would trade interpretability for parsimony.

Table 1: Summary of Variables

Characteristic	Overall n = 300	BASC + placebo n = 68	BASC + varenicline n = 83	ST + placebo n = 68	ST + varenicline n = 81
age_ps	50.0 (12.6)	50.7 (13.5)	50.3 (13.2)	50.3 (10.8)	48.7 (12.7)
sex_ps					
Male	135.0 (45.0%)	30.0 (44.1%)	39.0 (47.0%)	29.0 (42.6%)	37.0 (45.7%)
Female	165.0 (55.0%)	38.0 (55.9%)	44.0 (53.0%)	39.0 (57.4%)	44.0 (54.3%)
NHW	105.0 (35.0%)	24.0 (35.3%)	34.0 (41.0%)	22.0 (32.4%)	25.0 (30.9%)
Black	157.0 (52.3%)	37.0 (54.4%)	37.0 (44.6%)	40.0 (58.8%)	43.0 (53.1%)
Hisp	18.0 (6.0%)	5.0 (7.4%)	4.0 (4.8%)	4.0 (5.9%)	5.0 (6.2%)
inc					
Less than $20,000	110.0 (37.0%)	25.0 (37.3%)	30.0 (36.6%)	26.0 (38.2%)	29.0 (36.3%)
$20,000–35,000	68.0 (22.9%)	16.0 (23.9%)	17.0 (20.7%)	14.0 (20.6%)	21.0 (26.3%)
$35,001–50,000	46.0 (15.5%)	8.0 (11.9%)	13.0 (15.9%)	14.0 (20.6%)	11.0 (13.8%)
$50,001–75,000	38.0 (12.8%)	12.0 (17.9%)	12.0 (14.6%)	8.0 (11.8%)	6.0 (7.5%)
More than $75,000	35.0 (11.8%)	6.0 (9.0%)	10.0 (12.2%)	6.0 (8.8%)	13.0 (16.3%)
edu					
Grade School	1.0 (0.3%)	1.0 (1.5%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)
Some High School	16.0 (5.3%)	3.0 (4.4%)	7.0 (8.4%)	2.0 (2.9%)	4.0 (4.9%)
High School Graduate/GED	76.0 (25.3%)	23.0 (33.8%)	15.0 (18.1%)	11.0 (16.2%)	27.0 (33.3%)
Some College/Technical School	116.0 (38.7%)	22.0 (32.4%)	32.0 (38.6%)	38.0 (55.9%)	24.0 (29.6%)
College Graduate	91.0 (30.3%)	19.0 (27.9%)	29.0 (34.9%)	17.0 (25.0%)	26.0 (32.1%)
ftcd_score	5.2 (2.1)	5.3 (2.0)	5.1 (2.3)	5.4 (2.1)	5.2 (2.1)
ftcd.5.mins	138.0 (46.0%)	32.0 (47.1%)	33.0 (39.8%)	35.0 (51.5%)	38.0 (46.9%)
readiness	6.8 (1.2)	6.8 (1.4)	6.7 (1.2)	7.0 (1.3)	6.7 (1.1)
cpd_ps	15.1 (7.9)	15.6 (9.1)	15.5 (8.5)	15.0 (7.2)	14.4 (6.6)
crv_total_pq1	7.2 (3.7)	7.4 (3.8)	7.2 (3.9)	7.0 (3.7)	7.1 (3.5)
hedonsum_n_pq1	22.6 (19.6)	23.2 (20.3)	22.9 (19.0)	20.8 (20.1)	23.4 (19.5)
hedonsum_y_pq1	25.4 (19.4)	27.7 (21.5)	22.4 (17.0)	27.4 (19.9)	25.0 (19.4)
otherdiag	133.0 (44.3%)	35.0 (51.5%)	30.0 (36.1%)	28.0 (41.2%)	40.0 (49.4%)
antidepmed	82.0 (27.3%)	28.0 (41.2%)	24.0 (28.9%)	15.0 (22.1%)	15.0 (18.5%)
mde_curr					
Past	153.0 (51.0%)	36.0 (52.9%)	43.0 (51.8%)	37.0 (54.4%)	37.0 (45.7%)
Current	147.0 (49.0%)	32.0 (47.1%)	40.0 (48.2%)	31.0 (45.6%)	44.0 (54.3%)
NMR	0.4 (0.2)	0.3 (0.2)	0.4 (0.2)	0.4 (0.3)	0.4 (0.2)
Only.Menthol	178.0 (59.7%)	40.0 (58.8%)	48.0 (58.5%)	43.0 (64.2%)	47.0 (58.0%)

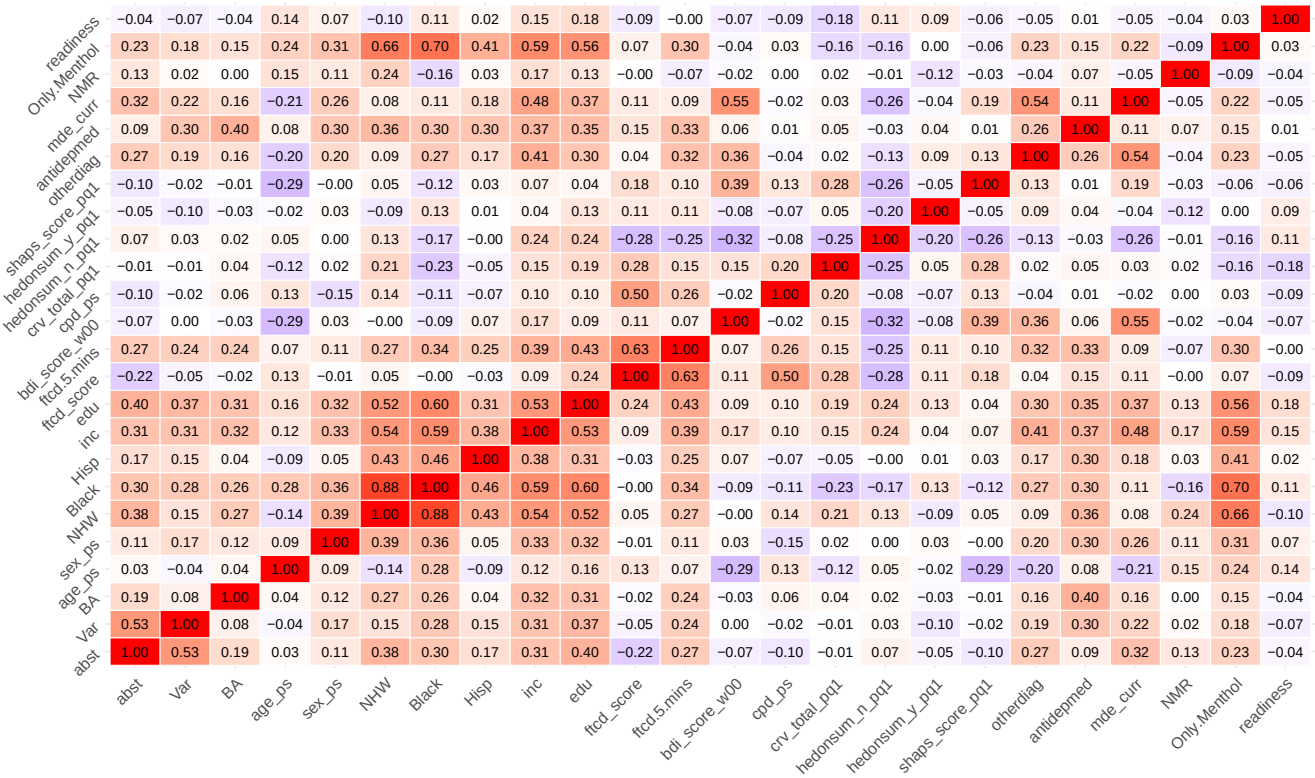
Note:
Mean (SD); n (%)

Other notable decisions include omission of `Hisp` due to insufficient sample size (18 individuals), which is a limitation of our analysis. We considered taking the three race and ethnicity variables `NHW`, `Black`, and `Hisp` and producing a single race plus ethnicity variable with mutually exclusive groups for non-Hispanic white, non-Hispanic Black, Hispanic, and Other. However because there are only 18 individuals noted as `Hisp` (and only two of those individuals are `Black`), this hypothetical combined race plus ethnicity variable would've suffered from the same cell size issue as other factors with many levels.

We further note that based on our correlation matrix there is a very high correlation (well over 85%) between `NHW` and `Black`. This issue combined with the issue noted previously lead us to also drop the `NHW` variable. We also note that use of only menthol cigarettes is highly correlated with both `NHW`, `Black`, `inc`, and `edu`. This is a concern, but as menthol cigarette use may be a unique trait and we are conducting regularized regression, we elect to leave this variable in the model and allow the penalty to aid our choice.

We conclude by grouping levels of `edu` and `inc` to limit parameter proliferation and allow for estimation and model convergence, given that Table 1 shows that there is insufficient sample within each level when not grouped. We also note that the continuous covariate NMR is right-skewed and could accommodate a log transformation to encourage normality. Log transforming skewed variables could help by encouraging convergence and simplifying the loss landscape. However, empirical results for model performance in models fit with and without the log transformation on NMR indicate that the model preforms similarly or better without the log transformation. Because interpreting untransformed coefficients is more straightforward than their transformed counterparts, and taking into account the lack of penalty in term of model fit for not transforming, we opt to leave NMR in the raw form. Finally, we note that there is a minor class imbalance issue (~2.1:10). The outcome is somewhat rare, and this could lead to model fitting issues. Correcting for this class imbalance is a potential area for improvement and a limitation.

Figure 1: Correlation Matrix



Missingness

We examined the amount of missing data by variable and by sample. The most missing data in any single record is two missing variables. There are seven variables missing data. The most missing data exists in `NMR` with 21 missing values (7%). There are 241 complete cases, which would reduce our sample by 59 records. This strongly suggests the need for imputation. After inspecting the data, we ran Little's

test for MCAR. The null hypothesis for this test is that the data are not MCAR. The p-value was over 0.05, so we cannot reject the null hypothesis. We conclude that there is evidence that these data are missing completely at random. Missing data in covariates is imputed in two distinct points in our estimation process, once during the coefficient estimation process and again within the bootstrap loop used for confidence interval estimation (additional details are in the section below).

Modeling

We use a Lasso model to answer both questions at issue. To evaluate baseline variables that predict `abst`, we construct a Lasso model that includes all main effects and polynomial terms for score variables. To evaluate baseline moderators of the effect of `BA` on `abst`, we also use a Lasso model, but the model that we use this time includes all main effects and polynomial terms of scores, as well as interactions of those terms with `BA`. We note that there is still the assumption that there is a linear relationship between the log odds of predictors and the outcome, but not an assumption of linearity between the predictors and the outcome itself.

We choose a Lasso model due to its constraints on potential values of the coefficients, introducing desirable bias to the model when performed correctly, which prevent overfitting. We choose not to use L0, L0+L2, or relaxed Lasso: while all three are computationally more difficult and would have more convergence issues than Lasso, the former two also have a harsher penalty (Hazimeh and Mazumder 2020; Hastie, Tibshirani, and Wainwright 2015; Meinshausen 2007).

First, we perform an 80/20 validation training split, where each split has the same proportion of positive outcomes (where `abst = 1`). From this training data we generate five imputed datasets. These five datasets are stacked and used first to find the optimal λ for our primary Lasso model via a process of cross-validation to minimize deviance (log-likelihood). Once this optimal λ is determined, the primary Lasso model is fit on the five stacked imputed datasets. The primary Lasso model is so called because this model is responsible for estimating our coefficients (as to distinguish it from the Lasso models fit within the bootstrap procedure).

Having obtained out coefficient estimates, we proceed to the bootstrap procedure to estimate confidence intervals. First, we take a bootstrap re-sample of the raw, unimputed training data. We then produce five new imputed datasets and stack them. These stacked imputed datasets are used to fit a Lasso model from which coefficient values are extracted and stored to be used in the estimation of confidence intervals. This process is repeated 6,000 times to ensure reliable confidence interval estimates.

After the estimates are derived from the procedure described above, we evaluate the model's performance using the validation data. We tested the model's performance using the pooled estimates on validation data in order to assess calibration and discrimination. For the former, we use calibration plots comparing the model's abstinence predictions to the study's actual results. For the latter, we use AUC and ROC to test whether the model could correctly identify participants who were or were not abstinent.

This procedure is identical for the main effects (baseline) and interactions (moderator) models except that the interactions model does not allow for penalized main effects. This is done to ensure interpretability of the interactions, but also creates a difficult optimization landscape as well as hampering the Lasso's ability to improve model fit by enforcing sparsity. For both the main effects (baseline) and interactions (moderator) model, enforcement of additional sparsity beyond the Lasso's inherent variable selection was unnecessary. This result is expected for the main effects model as the number of parameters is reasonable given the size of the data, but somewhat unexpected for the interactions model where the number of parameters is large relative the the data. Though empirical results, we note that further

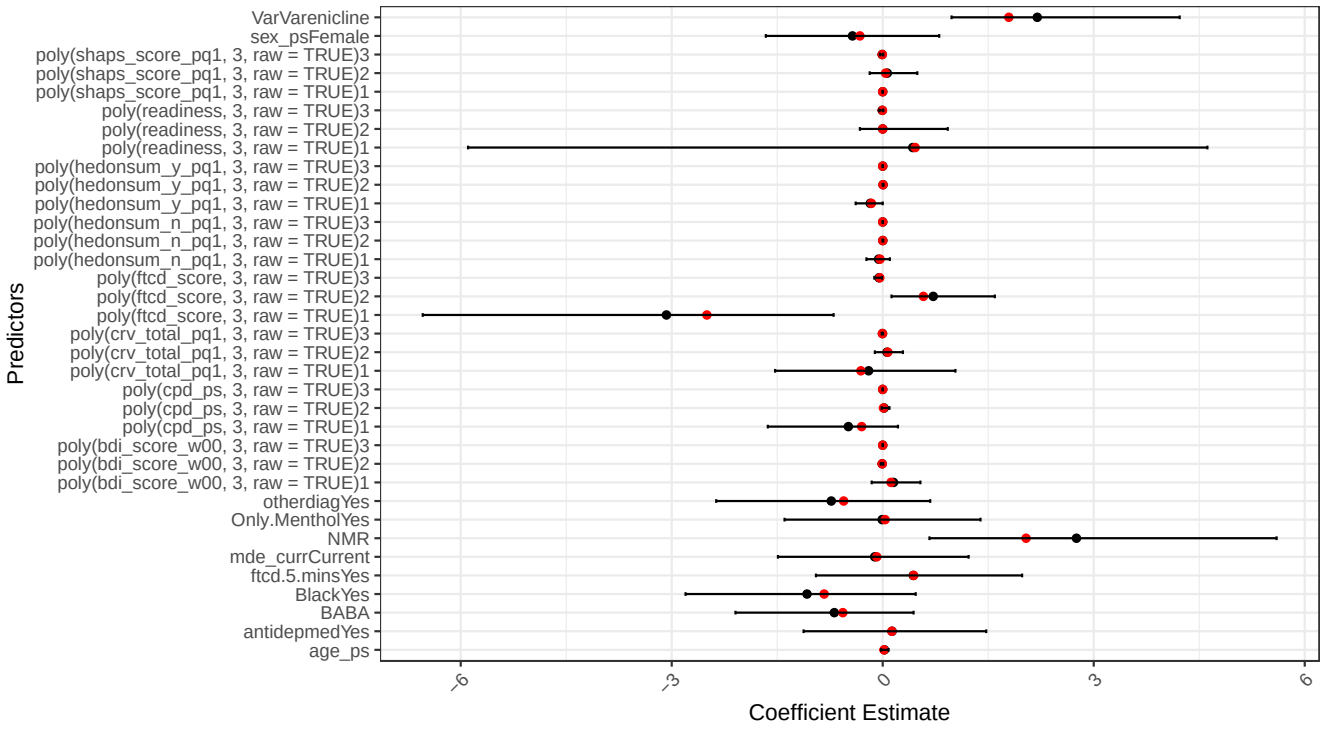
enforcing sparsity on the very complicated moderator model does not improve model fit as evaluated by our diagnostic criteria.

Results

Baseline Variables Predict Abstinence

Results are presented for bootstrapped Lasso regression on outcome abstinence, including main effects of baseline covariates and controlling for other treatments. Only coefficients with confidence intervals not containing zero are presented in the table of results (noted as significant).

Figure 2: Bootstrap Confidence Intervals for Lasso Coefficients



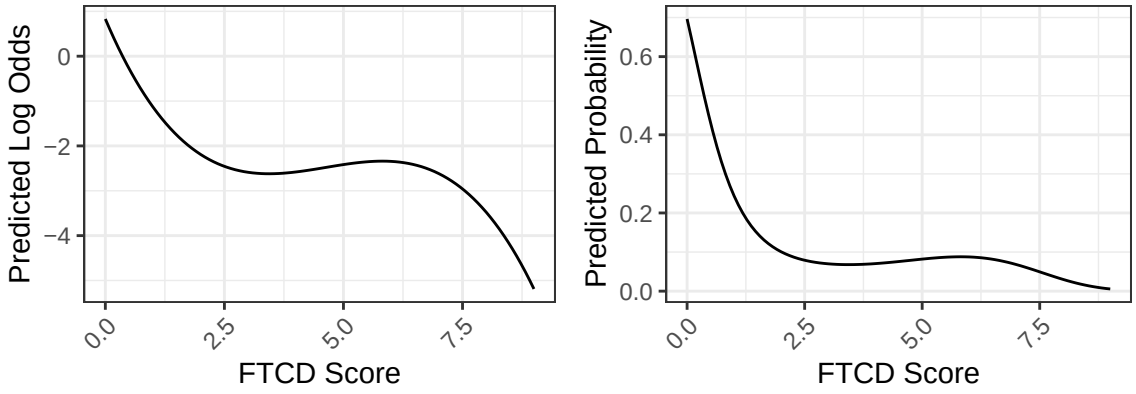
There are four significant baseline predictors of abstinence: `ftcd_score`, `Var`, `hedonsum_y_pq1`, and `NMR`. Varenicline treatment showed a strong positive association with abstinence (OR = 6.01, 95% CI (OR): 2.66-68.16), with participants receiving varenicline having about 6 times higher odds of achieving abstinence compared to those receiving placebo, consistent with prior literature on varenicline's efficacy. Additionally, nicotine metabolism rate demonstrated a positive relationship with abstinence outcomes, with each unit increase in NMR associated with 8 times higher odds of abstinence (95% CI (OR): 1.94-270.48). That NMR predicts abstinence is also understandable, but the direction of the association is unexpected. As noted above, higher NMR indicates faster metabolism of nicotine. Among other potential explanations for this variable's predictive value, individuals with faster nicotine metabolism tend to experience more intense withdrawal symptoms (Liakoni et al. 2019), as well as greater physical dependence and reward (Sofuoglu et al. 2012), presumably making abstinence more difficult. This finding regarding NMR is particularly noteworthy as it appears to diverge from some previous research suggesting faster metabolizers typically experience greater difficulty achieving cessation, though the relationship between metabolism and cessation outcomes is known to be complex and may be treatment-dependent.

Table 2: Lasso Regression Coefficient Estimates

Predictor	Estimate (Primary)	Estimate (Mean)	Estimate (Median)	Odds Ratio	Lower 2.5%	Upper 97.5%	Proportion Non-Zero	Sig.
VarVarenicline	1.793	2.313	2.198	6.007	0.979	4.222	1.000	Yes
poly(ftcd_score, 3, raw = TRUE)1	-2.500	-3.220	-3.075	0.082	-6.540	-0.698	0.999	Yes
poly(ftcd_score, 3, raw = TRUE)2	0.578	0.761	0.718	1.783	0.125	1.594	0.996	Yes
poly(ftcd_score, 3, raw = TRUE)3	-0.042	-0.056	-0.052	0.959	-0.119	-0.009	0.998	Yes
NMR	2.038	2.848	2.755	7.675	0.664	5.600	1.000	Yes

The polynomial variables are difficult to interpret in the same efficient manner as the continuous `NMR` or binary `Var`, therefore we produce plots to enhance our understanding. The FTCD score (`ftcd_score`) demonstrated significant non-linear relationships with abstinence. As visualized in Figure 3, the relationship between FTCD score and abstinence probability follows a complex cubic pattern, with the highest probability of abstinence occurring at low levels of dependence and decreasing non-linearly as the score increases. This relationships between FTCD score (`ftcd_score`) and the outcome are produced while holding all other variables at zero (for numeric variables) or their reference level (for categorical variables).

Figure 3: Relationship between FTCD Score and Estimates



Diagnostics

Figure 4: Lasso ROC / AUC Plot

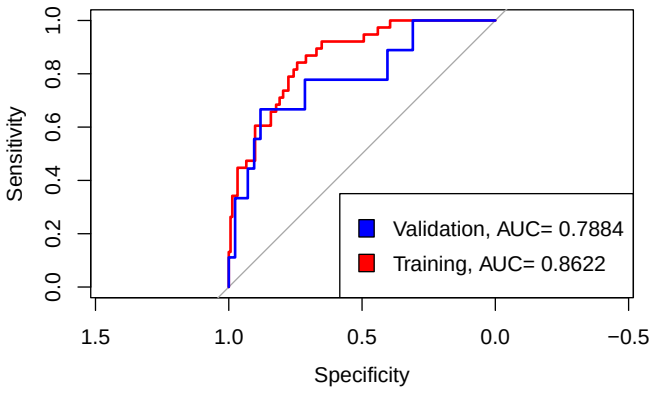
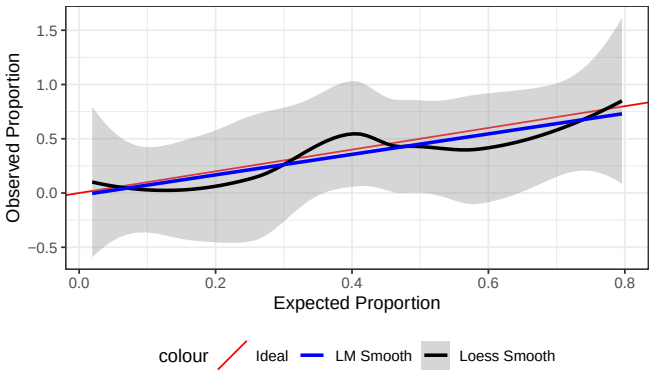


Figure 5: Lasso Calibration Plot: Validation Data



The model's discriminative ability was assessed using ROC curves and calibration plots (Figures 4 and 5). The Lasso model demonstrated good discriminative performance with an AUC of 0.7884 on the validation set, though some overfitting is suggested by the higher training AUC of 0.8622. The calibration plot reveals generally good agreement between predicted and observed probabilities at all ranges but shows increasing uncertainty at very high and very low probability ranges, as evidenced by the widening confidence bands around the Loess curve. The overall calibration suggests the model provides reasonably reliable probability estimates, especially in the more commonly observed lower probability ranges.

Baseline Variables as Moderators of BA

Results are presented for bootstrapped Lasso regression on the outcome abstinence, with main effects included and unpenalized. Interactions of baseline covariates with `BA` are included and penalized, allowing the Lasso to select the interaction terms of consequence. The coefficients derived from the primary non-bootstrap Lasso fit are noted as such. Only coefficients with confidence intervals not containing zero are presented in the table of results (noted as significant).

Table 3: Lasso Regression Coefficient Estimates

Predictor	Estimate (Primary)	Estimate (Mean)	Estimate (Median)	Odds Ratio	Lower 2.5%	Upper 97.5%	Proportion Non-Zero	Sig.
VarVarenicline	2.466	10.637	7.905	11.781	2.642	32.636	1	Yes

Our moderator analysis revealed that there were no significant interactions terms and therefore no baseline covariates which conclusively moderate the effect of BA on abstinence. The model also confirmed the robust effect of varenicline (OR = 11.78, 95% CI (OR): 14.05-1.4913627 $\times 10^{14}$), though the wide confidence interval suggests considerable uncertainty in the effect size estimate.

Diagnostics

Figure 6: Lasso ROC / AUC Plot

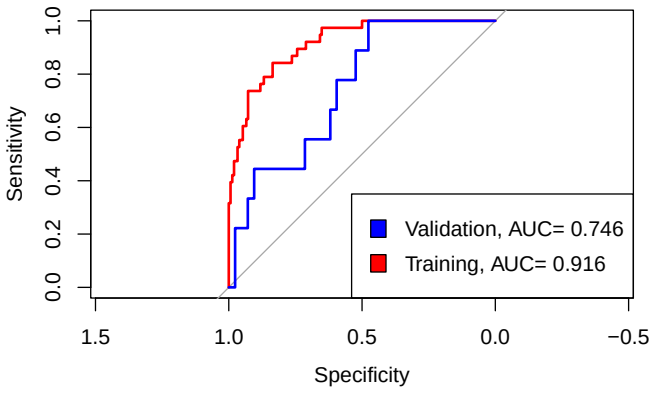
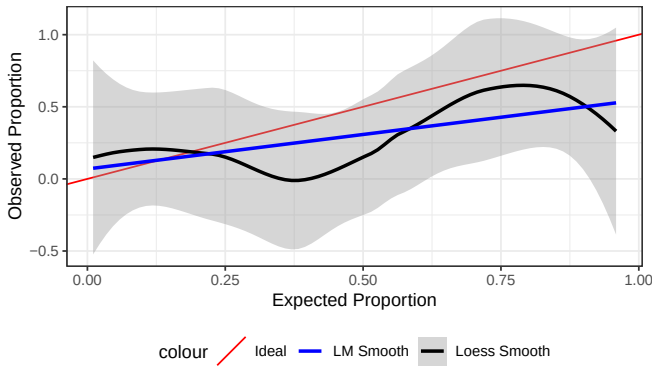


Figure 7: Lasso Calibration Plot: Validation Data



The diagnostic plots for the moderator analysis (Figures 6 and 7) suggest more modest model performance compared to our main effects model. The ROC curves show fair discriminative ability with an AUC of 0.746 on the validation set and 0.916 on the training set. This decreased performance relative to our previous model likely reflects the increased complexity introduced by interaction terms and the

inherent difficulty in predicting treatment effect heterogeneity especially in small samples. This pattern is clearly indicative of overfitting which is unsurprising given the large number of parameters and small amount of data.. We note this overfitting is stubborn insofar as manually enforcing sparsity in coefficients by thresholding narrows the gap between test and train AUC, but at the expense of overall performance. The calibration plot reveals deviations from the ideal diagonal, with the model showing a tendency toward under-prediction at higher probabilities. The wide confidence bands around the meandering Loess curve indicate considerable uncertainty in these predictions. Despite these limitations, the diagnostic plots suggest the model provides fairly reliable probability estimates, but does tend to overfit the data.

Conclusion

In this reanalysis of the BASC-Varenicline trial data, we identified several important predictors of smoking cessation success among individuals with current or past MDD. Our primary analysis confirmed varenicline's strong positive effect on abstinence and revealed complex non-linear relationships between cessation success and baseline characteristics such as nicotine dependence (FTCD score). Notably, higher nicotine metabolism rates were associated with improved cessation outcomes, a finding that warrants further investigation given its divergence from some previous research.

Several limitations of our analysis warrant discussion. First, our analytical approach was constrained by sample size limitations, which led to the exclusion of Hispanic ethnicity data due to insufficient representation (n=18) and necessitated the grouping of educational and income categories to ensure model convergence. Second, the class imbalance in our outcome (21.33% abstinent) posed challenges for model fitting and likely contributed to the decreased accuracy of predictions at higher probability ranges, particularly in our moderator analysis. Third, convergence issues stemming from our relatively small sample size and numerous dichotomous variables limited our ability to employ more sophisticated regularization approaches. Finally, despite doing 6,000 bootstrap iterations, if given more time larger bootstrap runs would be welcome. Future analyses with larger samples might benefit from exploring alternative methods such as L0 best subset selection, L0+L2 regularization, relaxed Lasso, group Lasso, or hierarchical Lasso, which could potentially offer improved variable selection and prediction accuracy.

References

Anthenelli, Robert M, Neal L Benowitz, Robert West, Lisa St Aubin, Thomas McRae, David Lawrence, John Ascher, Cristina Russ, Alok Krishen, and A Eden Evins. 2016. "Neuropsychiatric Safety and Efficacy of Varenicline, Bupropion, and Nicotine Patch in Smokers with and Without Psychiatric Disorders (EAGLES): A Double-Blind, Randomised, Placebo-Controlled Clinical Trial." *The Lancet* 387 (10037): 2507–20.

Anthenelli, Robert M, Chad Morris, Tanya S Ramey, Sarah J Dubrava, Kostas Tsilkos, Cristina Russ, and Carla Yunis. 2013. "Effects of Varenicline on Smoking Cessation in Adults with Stably Treated Current or Past Major Depression: A Randomized Trial." *Annals of Internal Medicine* 159 (6): 390–400.

Beck, Aaron T, Robert A Steer, and Gregory Brown. 1996. "Beck Depression Inventory–II." *Psychological Assessment*.

Bien, Jacob, Jonathan Taylor, and Robert Tibshirani. 2013. "A Lasso for Hierarchical Interactions." *Annals of Statistics* 41 (3): 1111.

Breslau, Naomi, M Marlyne Kilbey, and Patricia Andreski. 1992. "Nicotine Withdrawal Symptoms and Psychiatric Disorders: Findings from an Epidemiologic Study of Young Adults." *The American Journal of Psychiatry* 149 (4): 464–69.

Cinciripini, Paul M, Jason D Robinson, Maher Karam-Hage, Jennifer A Minnix, Cho Lam, Francesco Versace, Victoria L Brown, Jeffrey M Engelmann, and David W Wetter. 2013. "Effects of Varenicline and Bupropion Sustained-Release Use Plus Intensive Smoking Cessation Counseling on Prolonged Abstinence from Smoking and on Depression, Negative Affect, and Other Symptoms of Nicotine Withdrawal." *JAMA Psychiatry* 70 (5): 522–33.

Cook, Jessica, Bonnie Spring, Dennis McChargue, and Neal Doran. 2010. "Effects of Anhedonia on Days to Relapse Among Smokers with a History of Depression: A Brief Report." *Nicotine & Tobacco Research* 12 (9): 978–82.

Cuijpers, Pim, Annemieke Van Straten, and Lisanne Warmerdam. 2007. "Behavioral Activation Treatments of Depression: A Meta-Analysis." *Clinical Psychology Review* 27 (3): 318–26.

Dimidjian, Sona, Manuel Barrera Jr, Christopher Martell, Ricardo F Muñoz, and Peter M Lewinsohn. 2011. "The Origins and Current Status of Behavioral Activation Treatments for Depression." *Annual Review of Clinical Psychology* 7 (1): 1–38.

Evins, A Eden, Melissa A Culhane, Jonathan E Alpert, Joel Pava, Bruce S Liese, Amy Farabaugh, and Maurizio Fava. 2008. "A Controlled Trial of Bupropion Added to Nicotine Patch and Behavioral Therapy for Smoking Cessation in Adults with Unipolar Depressive Disorders." *Journal of Clinical Psychopharmacology* 28 (6): 660–66.

Fagerström, Karl. 2011. "Determinants of Tobacco Use and Renaming the FTND to the Fagerström Test for Cigarette Dependence." *Nicotine & Tobacco Research* 14 (1): 75–78.

Hall, Sharon M, Janice Y Tsoh, Judith J Prochaska, Stuart Eisendrath, Joseph S Rossi, Colleen A Redding, Amy B Rosen, Marc Meisner, Gary L Humfleet, and Julie A Gorecki. 2006. "Treatment for Cigarette Smoking Among Depressed Mental Health Outpatients: A Randomized Clinical Trial." *American Journal of Public Health* 96 (10): 1808–14.

Han, Beth, Nora D Volkow, Carlos Blanco, Douglas Tipperman, Emily B Einstein, and Wilson M Compton. 2022. "Trends in Prevalence of Cigarette Smoking Among US Adults with Major Depression or Substance Use Disorders, 2006-2019." *Jama* 327 (16): 1566–76.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. "An Introduction to Statistical Learning."

Hastie, Trevor, Robert Tibshirani, and Martin Wainwright. 2015. "Statistical Learning with Sparsity." *Monographs on Statistics and Applied Probability* 143 (143): 8.

Hazimeh, Hussein, and Rahul Mazumder. 2020. "Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms." *Operations Research* 68 (5): 1517–37.

Hitsman, Brian, Belinda Borrelli, Dennis E McChargue, Bonnie Spring, and Raymond Niaura. 2003. "History of Depression and Smoking Cessation Outcome: A Meta-Analysis." *Journal of Consulting and Clinical Psychology* 71 (4): 657.

Hitsman, Brian, Lee Hogarth, Li-Jung Tseng, Jordan C Teige, William G Shadel, Dana Britt DiBenedetti, Spencer Danto, Theodore C Lee, Lawrence H Price, and Raymond Niaura. 2013. "Dissociable Effect of Acute Varenicline on Tonic Versus Cue-Provoked Craving in Non-Treatment-Motivated Heavy Smokers." *Drug and Alcohol Dependence* 130 (1-3): 135–41.

Hitsman, Brian, George D Papandonatos, Jacqueline K Gollan, Mark D Huffman, Raymond Niaura, David C Mohr, Anna K Veluz-Wilkins, et al. 2023. "Efficacy and Safety of Combination Behavioral Activation for Smoking Cessation and Varenicline for Treating Tobacco Dependence Among Individuals with Current or Past Major Depressive Disorder: A 2× 2 Factorial, Randomized, Placebo-Controlled Trial." *Addiction* 118 (9): 1710–25.

Hitsman, Brian, George D Papandonatos, Dennis E McChargue, Andrew DeMott, María José Herrera, Bonnie Spring, Belinda Borrelli, and Raymond Niaura. 2013. "Past Major Depression and Smoking Cessation Outcome: A Systematic Review and Meta-Analysis Update." *Addiction* 108 (2): 294–306.

Hopko, Derek R, CW Lejuez, Kenneth J Ruggiero, and Georg H Eifert. 2003. "Contemporary Behavioral Activation Treatments for Depression: Procedures, Principles, and Progress." *Clinical Psychology Review* 23 (5): 699–717.

Leventhal, Adam M, Andrew J Waters, Christopher W Kahler, Lara A Ray, and Steve Sussman. 2009. "Relations Between Anhedonia and Smoking Motivation." *Nicotine & Tobacco Research* 11 (9): 1047–54.

Liakoni, Evangelia, Kathryn C Edwards, Gideon St. Helen, Natalie Nardone, Delia A Dempsey, Rachel F Tyndale, and Neal L Benowitz. 2019. "Effects of Nicotine Metabolic Rate on Withdrawal Symptoms and Response to Cigarette Smoking After Abstinence." *Clinical Pharmacology & Therapeutics* 105 (3): 641–51.

Lyons, Michael, Brian Hitsman, Hong Xian, Matthew S Panizzon, Beth A Jerskey, Susan Santangelo, Michael D Grant, et al. 2008. "A Twin Study of Smoking, Nicotine Dependence, and Major Depression in Men." *Nicotine & Tobacco Research* 10 (1): 97–108.

MacPherson, Laura, Matthew T Tull, Alexis K Matusiewicz, Samantha Rodman, David R Strong, Christopher W Kahler, Derek R Hopko, Michael J Zvolensky, Richard A Brown, and CW3108050 Lejuez. 2010. "Randomized Controlled Trial of Behavioral Activation Smoking Cessation Treatment for Smokers with Elevated Depressive Symptoms." *Journal of Consulting and Clinical Psychology* 78 (1): 55.

MacPhillamy, Douglas J, and Peter M Lewinsohn. 1982. "The Pleasant Events Schedule: Studies on Reliability, Validity, and Scale Intercorrelation." *Journal of Consulting and Clinical Psychology* 50 (3): 363.

McClure, Erin A, Ryan G Vandrey, Matthew W Johnson, and Maxine L Stitzer. 2012. "Effects of Varenicline on Abstinence and Smoking Reward Following a Programmed Lapse." *Nicotine & Tobacco Research* 15 (1): 139–48.

Meier, Lukas, Sara Van De Geer, and Peter Bühlmann. 2008. "The Group Lasso for Logistic Regression." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 70 (1): 53–71.

Meinshausen, Nicolai. 2007. "Relaxed Lasso." *Computational Statistics & Data Analysis* 52 (1): 374–93.

Minami, Haruka, Shadi Nahvi, Julia H Arnsten, Hannah R Brinkman, Monica Rivera-Mindt, David W Wetter, Erika Litvin Bloom, et al. 2022. "A Pilot Randomized Controlled Trial of Smartphone-Assisted Mindfulness-Based Intervention with Contingency Management for Smokers with Mood Disorders." *Experimental and Clinical Psychopharmacology* 30 (5): 653.

Patterson, Freda, Christopher Jepson, Andrew A Strasser, James Loughead, Kenneth A Perkins, Ruben

C Gur, Joseph M Frey, Steven Siegel, and Caryn Lerman. 2009. "Varenicline Improves Mood and Cognition During Smoking Abstinence." *Biological Psychiatry* 65 (2): 144–49.

Paul, Alice. 2024. "PHP 2550 - Variable Selection."

Perkins, Kenneth A, Melissa Mercincavage, Carolyn A Fonte, and Caryn Lerman. 2010. "Varenicline's Effects on Acute Smoking Behavior and Reward and Their Association with Subsequent Abstinence." *Psychopharmacology* 210: 45–51.

Siegel, Scott D, Caryn Lerman, Alex Flitter, and Robert A Schnoll. 2020. "The Use of the Nicotine Metabolite Ratio as a Biomarker to Personalize Smoking Cessation Treatment: Current Evidence and Future Directions." *Cancer Prevention Research* 13 (3): 261–72.

Simon, Noah, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2013. "A Sparse-Group Lasso." *Journal of Computational and Graphical Statistics* 22 (2): 231–45.

Smith, Philip H, Mohammad Chhipa, Josef Bystrik, Jordan Roy, Renee D Goodwin, and Sherry A McKee. 2020. "Cigarette Smoking Among Those with Mental Disorders in the US Population: 2012–2013 Update." *Tobacco Control* 29 (1): 29–35.

Snaith, R Philip, Max Hamilton, S Morley, A Humayan, D Hargreaves, and P Trigwell. 1995. "A Scale for the Assessment of Hedonic Tone the Snaith–Hamilton Pleasure Scale." *The British Journal of Psychiatry* 167 (1): 99–103.

Sofuoglu, Mehmet, Aryeh I Herman, Marc Mooney, and Andrew J Waters. 2009. "Varenicline Attenuates Some of the Subjective and Physiological Effects of Intravenous Nicotine in Humans." *Psychopharmacology* 207: 153–62.

Sofuoglu, Mehmet, Aryeh I Herman, Haleh Nadim, and Peter Jatlow. 2012. "Rapid Nicotine Clearance Is Associated with Greater Reward and Heart Rate Increases from Intravenous Nicotine." *Neuropsychopharmacology* 37 (6): 1509–16.

Spring, Bonnie, Regina Pingitore, and Dennis E McChargue. 2003. "Reward Value of Cigarette Smoking for Comparably Heavy Smoking Schizophrenic, Depressed, and Nonpatient Smokers." *American Journal of Psychiatry* 160 (2): 316–22.

Talukder, Saki Rubaiya, Julia M Lappin, Veronica Boland, Hayden McRobbie, and Ryan James Courtney. 2023. "Inequity in Smoking Cessation Clinical Trials Testing Pharmacotherapies: Exclusion of Smokers with Mental Health Disorders." *Tobacco Control* 32 (4): 489–96.

Taylor, Gemma MJ, Taha Itani, Kyla H Thomas, Dheeraj Rai, Tim Jones, Frank Windmeijer, Richard M Martin, Marcus R Munafò, Neil M Davies, and Amy E Taylor. 2020. "Prescribing Prevalence, Effectiveness, and Mental Health Safety of Smoking Cessation Medicines in Patients with Mental Disorders." *Nicotine and Tobacco Research* 22 (1): 48–57.

Thorsteinsson, Haraldur S, J Christian Gillin, Christi A Patten, Shahrokh Golshan, Laura D Sutton, Sean Drummond, Camellia P Clark, John Kelsoe, and Mark Rapaport. 2001. "The Effects of Transdermal Nicotine Therapy for Smoking Cessation on Depressive Symptoms in Patients with Major Depression." *Neuropsychopharmacology* 24 (4): 350–58.

Weinberger, Andrea H, Michael O Chaiton, Jiaqi Zhu, Melanie M Wall, Deborah S Hasin, and Renee D Goodwin. 2020. "Trends in the Prevalence of Current, Daily, and Nondaily Cigarette Smoking and Quit Ratios by Depression Status in the US: 2005–2017." *American Journal of Preventive Medicine* 58 (5): 691–98.

Weinberger, Andrea H, Rani A Desai, and Sherry A McKee. 2010. "Nicotine Withdrawal in US Smokers with Current Mood, Anxiety, Alcohol Use, and Substance Use Disorders." *Drug and Alcohol Dependence* 108 (1-2): 7–12.

West, Robert, Christine L Baker, Joseph C Cappelleri, and Andrew G Bushmakin. 2008. "Effect of Varenicline and Bupropion SR on Craving, Nicotine Withdrawal Symptoms, and Rewarding Effects of Smoking During a Quit Attempt." *Psychopharmacology* 197: 371–77.

Appendix I: Script Code

```
# --- Preamble ---
# Date of last update: Nov. 12, 2024
# R Version: 4.3.1
# Package Versions:
#   tidyverse: 2.0.0
#   knitr: 1.45
#   kableExtra: 1.3.4
#   ggplot2: 3.4.3
#   naniar 1.0.0
#   visdat 0.6.0
#   car 3.1-2
#   lme4 1.1-34
#   ggpubr 0.6.0
#   glmnet 4.1-8
#   mice 3.16.0
#   caret 6.0-94
#   pROC 1.18.4
#   gt 0.9.0
#   gtsummary 1.7.2


setwd("~/GitHub/smoking_cessation_proj")

# Knitr Engine Setup
knitr::opts_chunk$set(message=F,
                      warning=F,
                      error=F,
                      echo=F,
                      fig.pos = "H" ,
                      fig.align = 'center')

# Packages
options(kableExtra.latex.load_packages = FALSE) # Required to avoid floatrow error
library(knitr)
library(kableExtra)
library(ggplot2)
library(naniar) # For mcar_test()
library(visdat) # For vis_dat()
library(tidyverse)
library(car) # For qqPlot(), vif()
library(lme4)
library(lmerTest) # Satterthwaite approximation for computing p-values on lme4
library(ggpubr)
library(glmnet)
library(mice)
library(caret)
library(pROC)
```

```r
library(gtsummary)
library(gt)

source("_helpers.R")

data <- read.csv("data/project2.csv", comment.char="#")




# Convert variables
data <- data %>%
  mutate(
    abst = factor(abst, levels = c(0, 1), labels = c("No", "Yes")),
    Var = factor(Var, levels = c(0, 1),
                          labels = c("Placebo", "Varenicline")),
    BA = factor(BA, levels = c(0, 1), labels = c("Standard", "BA")),
    age_ps = as.numeric(age_ps),
    sex_ps = factor(sex_ps, levels = c(1, 2), labels = c("Male", "Female")),
    NHW = factor(NHW, levels = c(0, 1), labels = c("No", "Yes")),
    Black = factor(Black, levels = c(0, 1), labels = c("No", "Yes")),
    Hisp = factor(Hisp, levels = c(0, 1), labels = c("No", "Yes")),
    inc = factor(inc, levels = 1:5,
                 labels = c("Less than $20,000",
                            "$20,000-35,000",
                            "$35,001-50,000",
                            "$50,001-75,000",
                            "More than $75,000"), ordered = TRUE),
    edu = factor(edu, levels = 1:5,
                 labels = c("Grade School",
                            "Some High School",
                            "High School Graduate/GED",
                            "Some College/Technical School",
                            "College Graduate"), ordered = TRUE),
    ftcd_score = as.integer(ftcd_score),
    ftcd.5.mins = factor(ftcd.5.mins, levels = c(0, 1), labels = c("No", "Yes")),
    bdi_score_w00 = as.integer(bdi_score_w00),
    cpd_ps = as.integer(cpd_ps),
    crv_total_pq1 = as.integer(crv_total_pq1),
    hedonsum_n_pq1 = as.integer(hedonsum_n_pq1),
    hedonsum_y_pq1 = as.integer(hedonsum_y_pq1),
    shaps_score_pq1 = as.integer(shaps_score_pq1),
    otherdiag = factor(otherdiag, levels = c(0, 1), labels = c("No", "Yes")),
    antidepmed = factor(antidepmed, levels = c(0, 1), labels = c("No", "Yes")),
    mde_curr = factor(mde_curr, levels = c(0, 1), labels = c("Past", "Current")),
    NMR = as.numeric(NMR),
    Only.Menthol = factor(Only.Menthol, levels = c(0, 1), labels = c("No", "Yes")),
    readiness = as.integer(readiness)
```

```
  )


df_tbl <- data %>%
  mutate(
    treatment_arm = case_when(
      Var == "Placebo" & BA == "Standard" ~ "ST + placebo",
      Var == "Placebo" & BA == "BA" ~ "BASC + placebo",
      Var == "Varenicline" & BA == "Standard" ~ "ST + varenicline",
      Var == "Varenicline" & BA == "BA" ~ "BASC + varenicline"
    )
  )

# gtsummary table
tbl <- df_tbl %>%
  select(
    treatment_arm,
    age_ps,
    sex_ps,
    NHW,
    Black,
    Hisp,
    inc,
    edu,
    ftcd_score,
    ftcd.5.mins,
    readiness,
    cpd_ps,
    crv_total_pq1,
    hedonsum_n_pq1,
    hedonsum_y_pq1,
    otherdiag,
    antidepmed,
    mde_curr,
    NMR,
    Only.Menthol
  ) %>%
  tbl_summary(
    by = treatment_arm,
    missing = "no",
    type = list(readiness ~ "continuous"),
    statistic = list(
      all_continuous() ~ "{mean} ({sd})",
      all_categorical() ~ "{n} ({p}%)"
    ),
    digits = list(
      all_continuous() ~ 1,
      all_categorical() ~ 1
    )
```

```
  ) %>%
  add_overall() %>%
  # Note: the nice header work here doesn't work because we are forced to use
  # as_kable for rendering so we can control the width
  #modify_header(label = "**Characteristic**") %>%
  modify_header(label = "Characteristic") %>%
  #modify_spanning_header(all_stat_cols() ~ "**Treatment Arm**")  %>%
  modify_header( # To match paper columns
    all_stat_cols() ~ paste0(
      #"**{level}**\n",
      "{level}      ",
      "n = {n}"
    )
  )

tbl <- tbl %>% as_kable(
  booktabs = T,
  longtable = T,
  escape = T,
  align = c('l','c','c','c','c','c')) %>%
  column_spec(1, width="4cm", latex_valign = "m") %>%
  column_spec(2, width="2cm", latex_valign = "m") %>%
  column_spec(3, width="2cm", latex_valign = "m") %>%
  column_spec(4, width="2cm", latex_valign = "m") %>%
  column_spec(5, width="2cm", latex_valign = "m") %>%
  column_spec(6, width="2cm", latex_valign = "m") %>%
  kable_styling(
    font_size = 8,
    latex_options = c(
    "repeat_header",
    'striped'),
    full_width = F,
    position = 'center'
  ) %>%
  footnote(general = "Mean (SD); n (%)", escape = T)



# tbl <- tbl %>%
#   as_gt() %>%
#   # cols_width( # Note: neither cols_width nore table.width work currently. cols_width
#   #             # works in gt tables but not gtsummary tables. table.width doesn't work
#   #             # for any LaTex/PDF Quarto docs. (12/13/24)
#   #   variable ~ "50pt",
#   #   var_type ~ "50pt",
#   #   var_label ~ "50pt",
#   #   row_type ~ "50pt",
#   #   label ~ "50pt",
#   #   stat_0 ~ "50pt",
```

```
#   #   stat_1 ~ "50pt",
#   #   stat_2 ~ "50pt",
#   #   stat_3 ~ "50pt",
#   #   stat_4 ~ "50pt"
#   #   #`**Characteristic**` ~ "50pt" # Not an actual column, just a label
#   #   #starts_with("var") ~ "100pt"
#   #   #variable ~ px(100) # LaTeX doesn't like px()
#   # ) %>%
# #  tab_options(
# #    #table.font.size = "small" # Note: Doesn't currently work in PDF/LaTeX
# #    #table.width = "500pt" # Note: Doesn't currently work in PDF/LaTeX
# #  ) %>%
#   as_latex()
#


tbl

# Note: using tbl-cap rather than caption= in kbl() breaks repeated title is span page


# Table 1
summary_table(data[-1]) %>%
  kbl(#caption = "Summary of Variables", # Conflicts with Quarto referencing
      booktabs = T,
      longtable = T, # LONGTABLE
      escape = T,
      align = "c") %>%
  column_spec(1, width="2.2cm", latex_valign = "m") %>%
  column_spec(2, width="1.3cm", latex_valign = "m") %>%
  column_spec(3, width="6cm", latex_valign = "m") %>%
  column_spec(4, width="1.0cm", latex_valign = "m") %>%
  column_spec(5, width="1.0cm", latex_valign = "m") %>%
  column_spec(6, width="1.25cm", latex_valign = "m") %>%
  column_spec(7, width="1.25cm", latex_valign = "m") %>%
  kable_styling(
    font_size = 8, # Added for LONGTABLE
    latex_options = c(#'HOLD_position', # Removed for LONGTABLE
    #'scale_down', # Removed for LONGTABLE
    "repeat_header",  # Added for LONGTABLE
    'striped'),
    full_width = F, # Note: TRUE does not work with LONGTABLE
    position = 'center'# Added for LONGTABLE
  ) %>%
  footnote(general = "Shapiro-Wilk test for normality;
             Grubb's test for outliers.", escape = F)


# Table 2
```

```r
summary_table(data[-1], stratify_var = "abst") %>%
  kbl(
      booktabs = T,
      longtable = T, # LONGTABLE
      escape = T,
      align = "c") %>%
  column_spec(1, width="2.21cm", latex_valign = "m") %>%
  column_spec(2, width="4cm", latex_valign = "m") %>%
  column_spec(3, width="4cm", latex_valign = "m") %>%
  column_spec(4, width=".35cm", latex_valign = "m") %>%
  kable_styling(
    font_size = 7.6, # Added for LONGTABLE
    latex_options = c(#'HOLD_position', # Removed for LONGTABLE
    #'scale_down', # Removed for LONGTABLE
    "repeat_header",  # Added for LONGTABLE
    'striped'),
    full_width = F, # Note: TRUE does not work with LONGTABLE
    position = 'center'# Added for LONGTABLE
  ) %>%
  footnote(general = "ns = P > 0.05,
           * = P $\\\\leq$ 0.05,
           ** = P $\\\\leq$ 0.01,
           *** = P $\\\\leq$ 0.001,
           **** = P $\\\\leq$ 0.0001
           ", escape = F) %>%
  footnote(general = "Kruskal-Wallis test for continuous variables,
           Chi-Square test for categorical variables.
           Bonferroni correction applied.", escape = F)

# Table 2
summary_table(data[-1], stratify_var = "BA") %>%
  kbl(
      booktabs = T,
      longtable = T, # LONGTABLE
      escape = T,
      align = "c") %>%
  column_spec(1, width="2.21cm", latex_valign = "m") %>%
  column_spec(2, width="4cm", latex_valign = "m") %>%
  column_spec(3, width="4cm", latex_valign = "m") %>%
  column_spec(4, width=".35cm", latex_valign = "m") %>%
  kable_styling(
    font_size = 7.6, # Added for LONGTABLE
    latex_options = c(#'HOLD_position', # Removed for LONGTABLE
    #'scale_down', # Removed for LONGTABLE
    "repeat_header",  # Added for LONGTABLE
    'striped'),
    full_width = F, # Note: TRUE does not work with LONGTABLE
    position = 'center'# Added for LONGTABLE
  ) %>%
```

```
  footnote(general = "ns = P > 0.05,
           * = P $\\\\leq$ 0.05,
           ** = P $\\\\leq$ 0.01,
           *** = P $\\\\leq$ 0.001,
           **** = P $\\\\leq$ 0.0001
           ", escape = F) %>%
  footnote(general = "Kruskal-Wallis test for continuous variables,
           Chi-Square test for categorical variables.
           Bonferroni correction applied.", escape = F)

# Table 4
summary_table(data[-1], stratify_var = "Var") %>%
  kbl(
      booktabs = T,
      longtable = T, # LONGTABLE
      escape = T,
      align = "c") %>%
  column_spec(1, width="2.21cm", latex_valign = "m") %>%
  column_spec(2, width="4cm", latex_valign = "m") %>%
  column_spec(3, width="4cm", latex_valign = "m") %>%
  column_spec(4, width=".35cm", latex_valign = "m") %>%
  kable_styling(
    font_size = 7.6, # Added for LONGTABLE
    latex_options = c(#'HOLD_position', # Removed for LONGTABLE
    #'scale_down', # Removed for LONGTABLE
    "repeat_header",  # Added for LONGTABLE
    'striped'),
    full_width = F, # Note: TRUE does not work with LONGTABLE
    position = 'center'# Added for LONGTABLE
  ) %>%
  footnote(general = "ns = P > 0.05,
           * = P $\\\\leq$ 0.05,
           ** = P $\\\\leq$ 0.01,
           *** = P $\\\\leq$ 0.001,
           **** = P $\\\\leq$ 0.0001
           ", escape = F) %>%
  footnote(general = "Kruskal-Wallis test for continuous variables,
           Chi-Square test for categorical variables.
           Bonferroni correction applied.", escape = F)

# Histograms of variable distributions

# Identify numeric columns
numeric_cols <- sapply(data, is.numeric)

# Create histograms for each numeric column
for (col in names(data)[numeric_cols]) {
  hist(data[[col]], main = paste("Histogram of", col), xlab = col)
}
```

```r
# Generate psuedo correlation matrix
psuedo_cor_matrix <- psuedo_cor_mat(data[-1])

# Plot the heatmap (with variable names and values)
ggplot(melt(psuedo_cor_matrix), aes(x = Var2, y = Var1, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = sprintf("%.2f", value)), size = 2.8, color = "black") +
  scale_fill_gradient2(low = "blue",
                       high = "red",
                       mid = "white",
                       midpoint = 0,
                       limits = c(-1, 1),
                       space = "Lab",
                       name = "Correlation") +
  theme_minimal() +
  labs(x = "", y = "") +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 9),
        axis.ticks.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.y = element_text(angle = 45, hjust = 1, size = 9),
        legend.position="none") +
  coord_fixed(ratio = .55)


missing_by_variable <- data %>%
  summarise(across(everything(), ~ sum(is.na(.)))) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Missing_Count") %>%
  mutate(Missing_Percentage = round((Missing_Count / nrow(data)) * 100, 2))

missing_by_record <- data %>%
  mutate(Row = row_number()) %>%
  mutate(Missing_Count = rowSums(is.na(across(everything())))) %>%
  select(Row, Missing_Count)

total_complete_cases <- data %>%
  summarise(Complete_Cases = sum(complete.cases(.))) %>%
  pull(Complete_Cases)

# Test for MCAR
naniar::mcar_test(data)


# prep data for analysis
data <- data %>%
  mutate(abst = as.numeric(abst) - 1,
    edu = factor(recode_factor(edu,
                               "Grade School" = "High School or Less",
```

```
                                "Some High School" = "High School or Less",
                                "High School Graduate/GED" = "High School or Less"),
                    levels = c("High School or Less",
                               "Some College/Technical School",
                               "College Graduate"),
                    ordered = TRUE),
    inc = factor(recode_factor(inc,
                                "$50,001-75,000" = "More than $50,000",
                                "More than $75,000" = "More than $50,000"),
                    levels = c("Less than $20,000",
                               "$20,000-35,000",
                               "$35,001-50,000",
                               "More than $50,000"),
                    ordered = TRUE)
         )

data <- data %>%
  #mutate(NMR = log(NMR)) %>% # Toggle off to test model fit w/o transform
  select(-c(Hisp, NHW))

# --- Validation / Train split ---

set.seed(123)
split_indices <- createDataPartition(y = data$abst, p = 0.8, list = FALSE)

# split data training / validation sets
train_data <- data[split_indices, ]
validation_data <- data[-split_indices, ]
data <- train_data
#GOOD; Q1; baseline
# Def All Params
B <- 1000      # num of bootstraps
m <- 5       # num of multiple imputations per bootstrap
k <- 5       # num of cv folds for cv.glmnet
poly_degree <- 3  # degree for polynomial terms, set to 1 to test model fit w/o poly

# remove id
data_complete <- data %>% select(-id)

# Flag predictor vars
# exclude 'abst' from predictors
predictor_vars <- setdiff(names(data_complete), c("abst"))

# Flag trt vars
treatment_vars <- c("BA", "Var")
baseline_vars <- setdiff(predictor_vars, treatment_vars)

# Flag numeric and int baseline vars
num_baseline_vars <- baseline_vars[sapply(data_complete[baseline_vars],
```

```
                                        function(x) is.numeric(x) & !is.integer(x))]

non_binary_factor_vars <- baseline_vars[sapply(data_complete[baseline_vars],
                                               function(x) is.integer(x))]
# Flag Binary baseline vars
binary_vars <- baseline_vars[sapply(data_complete[baseline_vars],
                                    function(x) length(unique(na.omit(x))) == 2)]

# Flag factor vars and store levels
factor_vars <- names(data_complete)[sapply(data_complete, is.factor)]
levels_list <- lapply(data_complete[, factor_vars, drop = FALSE], levels)

# --- Build Formula ---
# Main Effects
main_effects <- paste("BA + Var +",
                      paste(c(
                        paste0("poly(", non_binary_factor_vars,
                               ", ", poly_degree, ", raw=TRUE)"),
                        num_baseline_vars,
                        binary_vars
                      ), collapse = " + "))



# Full Model Formula
full_formula <- as.formula(paste("abst ~", main_effects))

# fix factor levels in the entire dataset (ensure consistency) BUGFIX
if (length(factor_vars) > 0) {
  for (var in factor_vars) {
    data_complete[[var]] <- factor(data_complete[[var]], levels = levels_list[[var]])
  }
}

# Gen master model matrix
master_model_matrix <- model.matrix(full_formula, data = data_complete)
master_coef_names <- colnames(master_model_matrix)
total_coeffs <- length(master_coef_names)

# init storage for coefs
coef_matrix <- matrix(NA, nrow = B, ncol = total_coeffs)
colnames(coef_matrix) <- master_coef_names

# init object for nonzero coef indicator
nonzero_matrix <- matrix(0, nrow = B, ncol = total_coeffs)
colnames(nonzero_matrix) <- master_coef_names
```

```r
# --- Main estimation and bootstrap loop ---

# --- Primary model fit on original data with imputation ---
# Initial imputation for primary fit
predictor_matrix <- make.predictorMatrix(data_complete)
predictor_matrix[, "abst"] <- 0
predictor_matrix["abst", ] <- 0
imp_primary <- mice(data_complete, m = m, printFlag = FALSE, seed = 123,
                    predictorMatrix = predictor_matrix, method = 'pmm')

# Stack imputations for primary fit
stacked_data_primary <- complete(imp_primary, action = "long", include = FALSE)
stacked_data_primary <- stacked_data_primary %>% dplyr::select(-.id, -.imp)

# Fix factor levels
if (length(factor_vars) > 0) {
  for (var in factor_vars) {
    stacked_data_primary[[var]] <- factor(stacked_data_primary[[var]],
                                          levels = levels_list[[var]])
  }
}

# Create model matrix for primary fit
model_data_full <- model.matrix(full_formula, data = stacked_data_primary)
x_primary <- model_data_full[, -1]
y_primary <- stacked_data_primary$abst

# Remove zero variance predictors and align columns
zero_var_cols <- apply(x_primary, 2, function(x) var(x) == 0)
if (any(zero_var_cols)) {
  x_primary <- x_primary[, !zero_var_cols]
  current_coef_names <- colnames(x_primary)
} else {
  current_coef_names <- colnames(x_primary)
}

# Align columns with master names
master_coef_names_no_intercept <- master_coef_names[-1]
missing_cols <- setdiff(master_coef_names_no_intercept, current_coef_names)
if (length(missing_cols) > 0) {
  zeros_matrix <- matrix(0, nrow = nrow(x_primary), ncol = length(missing_cols))
  colnames(zeros_matrix) <- missing_cols
  x_primary <- cbind(x_primary, zeros_matrix)
}

extra_cols <- setdiff(current_coef_names, master_coef_names_no_intercept)
if (length(extra_cols) > 0) {
  x_primary <- x_primary[, !(colnames(x_primary) %in% extra_cols)]
}
```

```r
x_primary <- x_primary[, master_coef_names_no_intercept]

# Fit primary Lasso model
cv_lasso_primary <- cv.glmnet(
  x = x_primary,
  y = y_primary,
  family = "binomial",
  alpha = 1,
  standardize = TRUE,
  relax = FALSE
)

primary_fit <- glmnet(
  x = x_primary,
  y = y_primary,
  family = "binomial",
  alpha = 1,
  lambda = cv_lasso_primary$lambda.min,
  standardize = TRUE,
  relax = FALSE
)

# Get aligned primary coefficients
primary_coef_values <- as.vector(coef(primary_fit))
primary_coef_names <- rownames(coef(primary_fit))
names(primary_coef_values) <- primary_coef_names
aligned_primary_coef <- setNames(rep(0, length(master_coef_names)), master_coef_names)
matched_primary <- names(primary_coef_values) %in% master_coef_names
aligned_primary_coef[names(primary_coef_values)[matched_primary]] <-
  primary_coef_values[matched_primary]




primary_nonzero <- abs(aligned_primary_coef) > 1e-8  # New







# --- Bootstrap loop for confidence intervals ---
coef_matrix <- matrix(NA, nrow = B, ncol = length(master_coef_names))
colnames(coef_matrix) <- master_coef_names

nonzero_matrix <- matrix(0, nrow = B, ncol = length(master_coef_names))
colnames(nonzero_matrix) <- master_coef_names
```

```r
set.seed(1)
for (b in 1:B) {
  cat("Bootstrap iteration:", b, "of", B, "\n")

  bootstrap_sample <- data_complete %>%
    group_by(abst) %>%
    sample_frac(size = 1, replace = TRUE) %>%
    ungroup()

  predictor_matrix <- make.predictorMatrix(bootstrap_sample)
  predictor_matrix[, "abst"] <- 0
  predictor_matrix["abst", ] <- 0
  imp <- mice(bootstrap_sample, m = m, printFlag = FALSE, seed = b,
              predictorMatrix = predictor_matrix, method = 'pmm')

  stacked_data <- complete(imp, action = "long", include = FALSE)
  stacked_data <- stacked_data %>% dplyr::select(-.id, -.imp)

  if (length(factor_vars) > 0) {
    for (var in factor_vars) {
      stacked_data[[var]] <- factor(stacked_data[[var]], levels = levels_list[[var]])
    }
  }

  model_data_full <- model.matrix(full_formula, data = stacked_data)
  x <- model_data_full[, -1]
  y_impute <- stacked_data$abst

  zero_var_cols <- apply(x, 2, function(x) var(x) == 0)
  if (any(zero_var_cols)) {
    x <- x[, !zero_var_cols]
    current_coef_names <- colnames(x)
  } else {
    current_coef_names <- colnames(x)
  }

  missing_cols <- setdiff(master_coef_names_no_intercept, current_coef_names)
  if (length(missing_cols) > 0) {
    zeros_matrix <- matrix(0, nrow = nrow(x), ncol = length(missing_cols))
    colnames(zeros_matrix) <- missing_cols
    x <- cbind(x, zeros_matrix)
  }

  extra_cols <- setdiff(current_coef_names, master_coef_names_no_intercept)
  if (length(extra_cols) > 0) {
    x <- x[, !(colnames(x) %in% extra_cols)]
  }

  x <- x[, master_coef_names_no_intercept]
```

```r
  # Fit bootstrap model using same lambda as primary fit
  boot_fit <- tryCatch({
    glmnet(
      x = x,
      y = y_impute,
      family = "binomial",
      alpha = 1,
      lambda = cv_lasso_primary$lambda.min,
      standardize = TRUE,
      relax = FALSE
    )
  }, error = function(e) {
    cat("Error in glmnet for bootstrap", b, ":", e$message, "\n")
    return(NULL)
  })

  if (is.null(boot_fit)) {
    next
  }

  coef_values <- as.vector(coef(boot_fit))
  coef_names <- rownames(coef(boot_fit))
  names(coef_values) <- coef_names

  aligned_coef_values <- setNames(rep(0, length(master_coef_names)), master_coef_names)
  matched <- names(coef_values) %in% master_coef_names
  aligned_coef_values[names(coef_values)[matched]] <- coef_values[matched]

  coef_matrix[b, ] <- aligned_coef_values
  nonzero_matrix[b, ] <- ifelse(aligned_coef_values != 0, 1, 0)
}


saveRDS(coef_matrix, file = "baseline_coef_CI-fix_6k-NMR.rds")
saveRDS(nonzero_matrix, file = "baseline_nonzero_CI-fix_6k-NMR.rds")

coef_matrix <- readRDS(file = "baseline_coef_CI-fix_6k-NMR.rds")
nonzero_matrix <- readRDS(file = "baseline_nonzero_CI-fix_6k-NMR.rds")

coef_matrix[, !primary_nonzero] <- 0

# --- Pool coefs ---

# identify bad runs
bad_runs <- apply(coef_matrix, 1, function(x) all(is.na(x)) | all(x == 0)) &
            apply(nonzero_matrix, 1, function(x) all(is.na(x)) | all(x == 0))

# rm bad runs from objects
```

```r
coef_matrix <- coef_matrix[!bad_runs, ]
nonzero_matrix <- nonzero_matrix[!bad_runs, ]

# make coef data frame
coef_mean <- colMeans(coef_matrix, na.rm = TRUE)
coef_median <- apply(coef_matrix, 2, median, na.rm = TRUE)
coef_lower <- apply(coef_matrix, 2, quantile, probs = 0.025, na.rm = TRUE)
coef_upper <- apply(coef_matrix, 2, quantile, probs = 0.975, na.rm = TRUE)
proportion_nonzero <- colSums(nonzero_matrix, na.rm = TRUE) / sum(nonzero_matrix[,1])

coef_ci <- data.frame(
  Predictor = colnames(coef_matrix),
  Mean = coef_mean,
  Median = coef_median,
  Lower_2.5 = coef_lower,
  Upper_97.5 = coef_upper,
  Proportion_Nonzero = proportion_nonzero
)

coef_ci$Primary_Estimate <- aligned_primary_coef



# plot the CIs
ggplot(coef_ci[-1,], aes(x = Predictor, y = Median)) +
  geom_point() +
  geom_point(aes(y=Primary_Estimate), color = "red") +
  geom_errorbar(aes(ymin = Lower_2.5, ymax = Upper_97.5), width = 0.2) +
  coord_flip() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(
    y = "Coefficient Estimate",
    x = "Predictors"
  )

# plot nonzero prop
ggplot(coef_ci, aes(x = Predictor, y = Proportion_Nonzero)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  theme_bw() +
  labs(
    title = "Proportion of Bootstrap Samples with Non-Zero Coefficients",
    y = "Proportion",
    x = "Predictors"
  )
```

```r
# --- Placeholder for: Enforce Sparsity (not done for main effects model) ---

coef_matrix_sparse <- coef_matrix

coef_mean <- colMeans(coef_matrix_sparse, na.rm = TRUE)
coef_median <- apply(coef_matrix_sparse, 2, median, na.rm = TRUE)
coef_lower <- apply(coef_matrix_sparse, 2, quantile, probs = 0.025, na.rm = TRUE)
coef_upper <- apply(coef_matrix_sparse, 2, quantile, probs = 0.975, na.rm = TRUE)
proportion_nonzero <- colSums(nonzero_matrix, na.rm = TRUE) / sum(nonzero_matrix[,1])

coef_ci_sparse <- data.frame(
  Predictor = colnames(coef_matrix_sparse),
  Primary_Estimate = aligned_primary_coef,
  Mean = coef_mean,
  Median = coef_median,
  Lower_2.5 = coef_lower,
  Upper_97.5 = coef_upper,
  Proportion_Nonzero = proportion_nonzero
)

# Results Table

coef_ci_sparse <- coef_ci_sparse %>%
  mutate(
    #Odds_Ratio = exp(Median)  # Odds Ratio
    Odds_Ratio = exp(Primary_Estimate)  # Odds Ratio (now using main lasso est.)
  )

coef_ci_sparse <- coef_ci_sparse %>%
  mutate(
    Rounded_Lower_2.5 = round(Lower_2.5, 4),
    Rounded_Upper_97.5 = round(Upper_97.5, 4),

    # Determine significance
    Significant = ifelse(
      Rounded_Lower_2.5 > 0 | Rounded_Upper_97.5 < 0,
      "Yes",
      "No"
    )
  ) %>%
  select(-Rounded_Lower_2.5, -Rounded_Upper_97.5)

table_data <- coef_ci_sparse %>%
  select(
    Predictor,
    Primary_Estimate,
    Mean,
    Median,
    Odds_Ratio,
```

```
    Lower_2.5,
    Upper_97.5,
    Proportion_Nonzero,
    Significant
  ) %>%
  rename(
    "Estimate (Primary)" = Primary_Estimate,
    "Estimate (Mean)" = Mean,
    "Estimate (Median)" = Median,
    "Odds Ratio" = Odds_Ratio,
    "Lower 2.5%" = Lower_2.5,
    "Upper 97.5%" = Upper_97.5,
    "Proportion Non-Zero" = Proportion_Nonzero,
        "Sig." = Significant

  )

table_data_significant <- table_data %>%
  filter(`Sig.` == "Yes")

table_data_significant %>%
  kbl(row.names = F,
    booktabs = TRUE,
    longtable = TRUE,
    escape = TRUE,
    align = "c",
    digits = 3,
    caption = "Lasso Regression Coefficient Estimates"
  ) %>%
  column_spec(1, width = "3.2cm", latex_valign = "m") %>%
  column_spec(2, width = "1.5cm", latex_valign = "m") %>%
  column_spec(3, width = "1.5cm", latex_valign = "m") %>%
  column_spec(4, width = "1.5cm", latex_valign = "m") %>%
  column_spec(5, width = "1.5cm", latex_valign = "m") %>%
  column_spec(6, width = "1.5cm", latex_valign = "m") %>%
  column_spec(7, width = "1.5cm", latex_valign = "m") %>%
  column_spec(8, width = "1.5cm", latex_valign = "m") %>%
  column_spec(9, width = ".5cm", latex_valign = "m") %>%
  kable_styling(
    font_size = 7.6,
    latex_options = c("repeat_header", "striped"),
    full_width = FALSE,
    position = "center"
  )


# FTCD

ftcd_coefs <- coef_ci[grep("poly\\(ftcd_score, 3, raw = TRUE\\)", coef_ci$Predictor), ]
```

```r
ftcd_range <- seq(min(na.omit(data$ftcd_score)),
                  max(na.omit(data$ftcd_score)), length.out = 100)

# calc predicted log odds
predicted_log_odds <- coef_ci$Primary_Estimate[coef_ci$Predictor == "(Intercept)"] +
                      ftcd_coefs$Primary_Estimate[1] * ftcd_range +
                      ftcd_coefs$Primary_Estimate[2] * ftcd_range^2 +
                      ftcd_coefs$Primary_Estimate[3] * ftcd_range^3

# convert to probabilities
predicted_probs <- 1/(1 + exp(-predicted_log_odds))

plot_data <- data.frame(
  ftcd_score = ftcd_range,
  log_odds = predicted_log_odds,
  probability = predicted_probs
)



# plot log odds
p1 <- ggplot(plot_data, aes(x = ftcd_score, y = log_odds)) +
  geom_line() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "FTCD Score",
       y = "Predicted Log Odds",
       title = "")

# plot probabilities
p2 <- ggplot(plot_data, aes(x = ftcd_score, y = probability)) +
  geom_line() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "FTCD Score",
       y = "Predicted Probability",
       title = "")


ftcd_plot <- ggarrange(
  p1,
  p2,
  ncol = 2,
  nrow = 1
)

print(ftcd_plot)
```

```r
### Note: interesting dynamic where when B <- 1000 this is no longer significant,
### but when B <- 6000 it is. Volatile results.
### Note: Now when B <- 6000 but NMR is raw not logged, it is again not sig.


# Hedonsum
hedon_coefs <- coef_ci[grep("poly\\(hedonsum_y_pq1, 3, raw = TRUE\\)", coef_ci$Predictor), ]
hedon_range <- seq(min(na.omit(data$hedonsum_y_pq1)),
                   max(na.omit(data$hedonsum_y_pq1)), length.out = 100)

# calc predicted log odds
predicted_log_odds <- coef_ci$Primary_Estimate[coef_ci$Predictor == "(Intercept)"] +
                      hedon_coefs$Primary_Estimate[1] * hedon_range +
                      hedon_coefs$Primary_Estimate[2] * hedon_range^2 +
                      hedon_coefs$Primary_Estimate[3] * hedon_range^3

# convert to probabilities
predicted_probs <- 1/(1 + exp(-predicted_log_odds))

plot_data <- data.frame(
  hedonsum_score = hedon_range,
  log_odds = predicted_log_odds,
  probability = predicted_probs
)

# plot log odds
p1 <- ggplot(plot_data, aes(x = hedonsum_score, y = log_odds)) +
  geom_line() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "Hedonsum Score",
       y = "Predicted Log Odds",
       title = "")

# plot probabilities
p2 <- ggplot(plot_data, aes(x = hedonsum_score, y = probability)) +
  geom_line() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "Hedonsum Score",
       y = "Predicted Probability",
       title = "")

hedon_plot <- ggarrange(
  p1,
  p2,
  ncol = 2,
  nrow = 1
```

```
)
print(hedon_plot)

# --- Generate Predictions ---

# fix factor levels in the entire dataset (ensure consistency) BUGFIX
if (length(factor_vars) > 0) {
  for (var in factor_vars) {
    validation_data[[var]] <- factor(validation_data[[var]], levels = levels_list[[var]])
  }
}

# compute ROC for valid data

# prepare valid data
model_data_valid_full <- model.matrix(full_formula, data = validation_data)
current_coef_names_valid <- colnames(model_data_valid_full) # align cols
missing_cols <- setdiff(master_coef_names, current_coef_names_valid)
if (length(missing_cols) > 0) {
  zeros_matrix <- matrix(0,
                         nrow = nrow(model_data_valid_full),
                         ncol = length(missing_cols))
  colnames(zeros_matrix) <- missing_cols
  model_data_valid_full <- cbind(model_data_valid_full, zeros_matrix)
}
extra_cols <- setdiff(current_coef_names_valid, master_coef_names)
if (length(extra_cols) > 0) {
  model_data_valid_full <-
    model_data_valid_full[, !(colnames(model_data_valid_full) %in% extra_cols)]
}

model_data_valid_full <- model_data_valid_full[, master_coef_names] # reorder
#predictor_coefs <- coef_ci_sparse$Median
predictor_coefs <- coef_ci_sparse$Primary_Estimate # NOTE: changed from using $Median
names(predictor_coefs) <- coef_ci_sparse$Predictor
predictor_coefs <- predictor_coefs[master_coef_names] # ensure match

# predictions for valid data
linear_predictor <- as.vector(model_data_valid_full %*% predictor_coefs)
pred_prob <- 1 / (1 + exp(-linear_predictor))

# BUGFIX
if (length(validation_data) < length(pred_prob)) {
  validation_data <- validation_data[complete.cases(validation_data),]
}

validation_data$pred_prob <- pred_prob
```

```r
# compute ROC for training data

# prepare training data
data_complete_cc <- data_complete[complete.cases(data_complete),]
model_data_train <- model.matrix(full_formula, data = data_complete_cc)
current_coef_names_train <- colnames(model_data_train)
missing_cols_train <- setdiff(master_coef_names, current_coef_names_train)
if (length(missing_cols_train) > 0) {
  zeros_matrix_train <- matrix(0, nrow = nrow(model_data_train),
                               ncol = length(missing_cols_train))
  colnames(zeros_matrix_train) <- missing_cols_train
  model_data_train <- cbind(model_data_train, zeros_matrix_train)
}
extra_cols_train <- setdiff(current_coef_names_train, master_coef_names)
if (length(extra_cols_train) > 0) {
  model_data_train <-
    model_data_train[, !(colnames(model_data_train) %in% extra_cols_train)]
}
model_data_train <- model_data_train[, master_coef_names]

# predictions for training data
linear_predictor_train <- as.vector(model_data_train %*% predictor_coefs)
pred_prob_train <- 1 / (1 + exp(-linear_predictor_train))
data_complete_cc$pred_prob <- pred_prob_train


# ROCs
roc_valid <- roc(validation_data$abst, validation_data$pred_prob)
roc_train <- roc(data_complete_cc$abst, data_complete_cc$pred_prob)

# plot ROC curves
plot(roc_train, col = "red", main = "")
plot(roc_valid, add = TRUE, col = "blue")
legend("bottomright",
       legend = c(paste("Validation, AUC=", round(roc_valid$auc, 4)),
                  paste("Training, AUC=", round(roc_train$auc, 4))),
       fill = c("blue", "red"))

# calc residuals
data_complete_cc$residuals <- data_complete_cc$abst - data_complete_cc$pred_prob

# plot residuals v. fitted
ggplot(data_complete_cc, aes(x = pred_prob, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(x = "Predicted Probability", y = "Residuals") +
  ggtitle("Residuals vs. Predicted Probabilities")
```

```r
# Histogram of Residuals
hist(data_complete_cc$residuals, main = "Histogram of Residuals")

# Q-Q Plot of Residuals
qqnorm(data_complete_cc$residuals)
qqline(data_complete_cc$residuals)

# calibration plot
num_cuts <- 24

# calibration for validation data
test_calib <- data.frame(
  prob = validation_data$pred_prob,
  bin = cut(validation_data$pred_prob, breaks = num_cuts),
  class = as.numeric(as.character(validation_data$abst))
)

test_calib <- test_calib %>%
  group_by(bin) %>%
  summarize(observed = sum(class)/n(),
            expected = sum(prob)/n(),
            se = sqrt(observed * (1-observed) / n()))

cols <- c("Ideal"="red","Loess Smooth"="black","LM Smooth"="blue")

ggplot(test_calib) +
  geom_abline(aes(intercept = 0, slope = 1, color="Ideal")) +
  geom_smooth(aes(x = expected,
                  y = observed,
                  color="Loess Smooth"), se=TRUE) +
  geom_smooth(aes(x = expected,
                  y = observed,
                  color="LM Smooth"), se=FALSE, method="lm") +
  scale_color_manual(values=cols) +
  labs(x = "Expected Proportion",
       y = "Observed Proportion",
       title="") +
  theme_bw() + theme(legend.position = "bottom")
#GOOD; Q2; Moderator
# Def All Params
B <- 6000      # num of bootstraps
m <- 5       # num of multiple imputations per bootstrap
k <- 5       # num of cv folds for cv.glmnet
poly_degree <- 3  # degree for polynomial terms, set to 1 to test model fit w/o poly

# remove id
data_complete <- data %>% select(-id)

# Flag predictor vars
```

```r
# exclude 'abst' from predictors
predictor_vars <- setdiff(names(data_complete), c("abst"))

# Flag trt vars
treatment_vars <- c("BA", "Var")
baseline_vars <- setdiff(predictor_vars, treatment_vars)

# Flag numeric and int baseline vars
num_baseline_vars <- baseline_vars[sapply(data_complete[baseline_vars],
                                          function(x) is.numeric(x) & !is.integer(x))]

non_binary_factor_vars <- baseline_vars[sapply(data_complete[baseline_vars],
                                               function(x) is.integer(x))]
# Flag Binary baseline vars
binary_vars <- baseline_vars[sapply(data_complete[baseline_vars],
                                    function(x) length(unique(na.omit(x))) == 2)]

# Flag factor vars and store levels
factor_vars <- names(data_complete)[sapply(data_complete, is.factor)]
levels_list <- lapply(data_complete[, factor_vars, drop = FALSE], levels)

# --- Build Formula ---
# Main Effects
main_effects <- paste("BA + Var +",
                      paste(c(
                        paste0("poly(", non_binary_factor_vars,
                               ", ", poly_degree, ", raw=TRUE)"),
                        num_baseline_vars,
                        binary_vars
                      ), collapse = " + "))

# Interaction Terms
interaction_terms <- paste(c(
  paste0("BA:", c(paste0("poly(", non_binary_factor_vars,
                         ", ", poly_degree, ", raw=TRUE)"), num_baseline_vars, binary_vars))
), collapse = " + ")

# Full Model Formula
full_formula <- as.formula(paste("abst ~", main_effects, "+", interaction_terms))

# fix factor levels in the entire dataset (ensure consistency) BUGFIX
if (length(factor_vars) > 0) {
  for (var in factor_vars) {
    data_complete[[var]] <- factor(data_complete[[var]], levels = levels_list[[var]])
  }
}

# Gen master model matrix
master_model_matrix <- model.matrix(full_formula, data = data_complete)
```

```r
master_coef_names <- colnames(master_model_matrix)
total_coeffs <- length(master_coef_names)


# --- Main estimation and bootstrap loop ---


# --- Primary model fit with imputation ---
# Initial imputation for primary fit
predictor_matrix <- make.predictorMatrix(data_complete)
predictor_matrix[, "abst"] <- 0
predictor_matrix["abst", ] <- 0
imp_primary <- mice(data_complete, m = m, printFlag = FALSE, seed = 123,
                    predictorMatrix = predictor_matrix, method = 'pmm')

# Stack imputations for primary fit
stacked_data_primary <- complete(imp_primary, action = "long", include = FALSE)
stacked_data_primary <- stacked_data_primary %>% dplyr::select(-.id, -.imp)

# Fix factor levels
if (length(factor_vars) > 0) {
  for (var in factor_vars) {
    stacked_data_primary[[var]] <- factor(stacked_data_primary[[var]],
                                          levels = levels_list[[var]])
  }
}

# Create model matrix for primary fit
model_data_full <- model.matrix(full_formula, data = stacked_data_primary)
x_primary <- model_data_full[, -1]
y_primary <- stacked_data_primary$abst

# Remove zero variance predictors and align columns
zero_var_cols <- apply(x_primary, 2, function(x) var(x) == 0)
if (any(zero_var_cols)) {
  x_primary <- x_primary[, !zero_var_cols]
  current_coef_names <- colnames(x_primary)
} else {
  current_coef_names <- colnames(x_primary)
}

# Align columns with master names
master_coef_names_no_intercept <- master_coef_names[-1]
missing_cols <- setdiff(master_coef_names_no_intercept, current_coef_names)
if (length(missing_cols) > 0) {
  zeros_matrix <- matrix(0, nrow = nrow(x_primary), ncol = length(missing_cols))
  colnames(zeros_matrix) <- missing_cols
  x_primary <- cbind(x_primary, zeros_matrix)
}
```

```r
extra_cols <- setdiff(current_coef_names, master_coef_names_no_intercept)
if (length(extra_cols) > 0) {
  x_primary <- x_primary[, !(colnames(x_primary) %in% extra_cols)]
}

x_primary <- x_primary[, master_coef_names_no_intercept]

# Set up penalty factors for primary fit
penalty_factors_primary <- ifelse(grepl(":", colnames(x_primary)), 1, 0)

# Verify penalty factor dimensions
if (ncol(x_primary) != length(penalty_factors_primary)) {
  stop("Mismatch between x columns and penalty_factors length in primary fit")
}

# Fit primary Lasso model with penalty factors
cv_lasso_primary <- cv.glmnet(
  x = x_primary,
  y = y_primary,
  family = "binomial",
  alpha = 1,
  penalty.factor = penalty_factors_primary,
  standardize = TRUE,
  relax = FALSE
)

primary_fit <- glmnet(
  x = x_primary,
  y = y_primary,
  family = "binomial",
  alpha = 1,
  lambda = cv_lasso_primary$lambda.min,
  penalty.factor = penalty_factors_primary,
  standardize = TRUE,
  relax = FALSE
)

# Get aligned primary coefficients
primary_coef_values <- as.vector(coef(primary_fit))
primary_coef_names <- rownames(coef(primary_fit))
names(primary_coef_values) <- primary_coef_names
aligned_primary_coef <- setNames(rep(0, length(master_coef_names)), master_coef_names)
matched_primary <- names(primary_coef_values) %in% master_coef_names
aligned_primary_coef[names(primary_coef_values)[matched_primary]] <-
  primary_coef_values[matched_primary]
```

```r
primary_nonzero <- abs(aligned_primary_coef) > 1e-8  # New


# --- Bootstrap loop for confidence intervals ---
coef_matrix <- matrix(NA, nrow = B, ncol = length(master_coef_names))
colnames(coef_matrix) <- master_coef_names

nonzero_matrix <- matrix(0, nrow = B, ncol = length(master_coef_names))
colnames(nonzero_matrix) <- master_coef_names


set.seed(1)
for (b in 1:B) {
  cat("Bootstrap iteration:", b, "of", B, "\n")

  bootstrap_sample <- data_complete %>%
    group_by(abst) %>%
    sample_frac(size = 1, replace = TRUE) %>%
    ungroup()

  predictor_matrix <- make.predictorMatrix(bootstrap_sample)
  predictor_matrix[, "abst"] <- 0
  predictor_matrix["abst", ] <- 0
  imp <- mice(bootstrap_sample, m = m, printFlag = FALSE, seed = b,
              predictorMatrix = predictor_matrix, method = 'pmm')

  stacked_data <- complete(imp, action = "long", include = FALSE)
  stacked_data <- stacked_data %>% dplyr::select(-.id, -.imp)

  if (length(factor_vars) > 0) {
    for (var in factor_vars) {
      stacked_data[[var]] <- factor(stacked_data[[var]], levels = levels_list[[var]])
    }
  }

  model_data_full <- model.matrix(full_formula, data = stacked_data)
  x <- model_data_full[, -1]
  y_impute <- stacked_data$abst

  zero_var_cols <- apply(x, 2, function(x) var(x) == 0)
  if (any(zero_var_cols)) {
    x <- x[, !zero_var_cols]
    current_coef_names <- colnames(x)
  } else {
    current_coef_names <- colnames(x)
  }

  missing_cols <- setdiff(master_coef_names_no_intercept, current_coef_names)
  if (length(missing_cols) > 0) {
```

```
  zeros_matrix <- matrix(0, nrow = nrow(x), ncol = length(missing_cols))
  colnames(zeros_matrix) <- missing_cols
  x <- cbind(x, zeros_matrix)
}

extra_cols <- setdiff(current_coef_names, master_coef_names_no_intercept)
if (length(extra_cols) > 0) {
  x <- x[, !(colnames(x) %in% extra_cols)]
}

x <- x[, master_coef_names_no_intercept]

# Set penalty factors for bootstrap iterations
penalty_factors <- ifelse(grepl(":", colnames(x)), 1, 0)

# Verify penalty factor dimensions
if (ncol(x) != length(penalty_factors)) {
  stop("Mismatch between x columns and penalty_factors length in bootstrap")
}

# Fit bootstrap model using same lambda as primary fit and penalty factors
boot_fit <- tryCatch({
  glmnet(
    x = x,
    y = y_impute,
    family = "binomial",
    alpha = 1,
    lambda = cv_lasso_primary$lambda.min,
    penalty.factor = penalty_factors,
    standardize = TRUE,
    relax = FALSE
  )
}, error = function(e) {
  cat("Error in glmnet for bootstrap", b, ":", e$message, "\n")
  return(NULL)
})

if (is.null(boot_fit)) {
  next
}

coef_values <- as.vector(coef(boot_fit))
coef_names <- rownames(coef(boot_fit))
names(coef_values) <- coef_names

aligned_coef_values <- setNames(rep(0, length(master_coef_names)), master_coef_names)
matched <- names(coef_values) %in% master_coef_names
aligned_coef_values[names(coef_values)[matched]] <- coef_values[matched]
```

```
  coef_matrix[b, ] <- aligned_coef_values
  nonzero_matrix[b, ] <- ifelse(aligned_coef_values != 0, 1, 0)
}



saveRDS(coef_matrix, file = "moderators_coef_CI-fix_6k-NMR.rds")
saveRDS(nonzero_matrix, file = "moderators_nonzero_CI-fix_6k-NMR.rds")

coef_matrix <- readRDS(file = "moderators_coef_CI-fix_6k-NMR.rds")
nonzero_matrix <- readRDS(file = "moderators_nonzero_CI-fix_6k-NMR.rds")

coef_matrix[, !primary_nonzero] <- 0

# --- Pool coefs ---

# identify bad runs
bad_runs <- apply(coef_matrix, 1, function(x) all(is.na(x)) | all(x == 0)) &
            apply(nonzero_matrix, 1, function(x) all(is.na(x)) | all(x == 0))

# rm bad runs from objects
coef_matrix <- coef_matrix[!bad_runs, ]
nonzero_matrix <- nonzero_matrix[!bad_runs, ]

# make coef data frame
coef_mean <- colMeans(coef_matrix, na.rm = TRUE)
coef_median <- apply(coef_matrix, 2, median, na.rm = TRUE)
coef_lower <- apply(coef_matrix, 2, quantile, probs = 0.025, na.rm = TRUE)
coef_upper <- apply(coef_matrix, 2, quantile, probs = 0.975, na.rm = TRUE)
proportion_nonzero <- colSums(nonzero_matrix, na.rm = TRUE) / sum(nonzero_matrix[,1])

coef_ci <- data.frame(
  Predictor = colnames(coef_matrix),
  Mean = coef_mean,
  Median = coef_median,
  Lower_2.5 = coef_lower,
  Upper_97.5 = coef_upper,
  Proportion_Nonzero = proportion_nonzero
)

coef_ci$Primary_Estimate <- aligned_primary_coef



# plot the CIs
ggplot(coef_ci[-1,], aes(x = Predictor, y = Median)) +
  geom_point() +
  geom_point(aes(y=Primary_Estimate), color = "red") +
  geom_errorbar(aes(ymin = Lower_2.5, ymax = Upper_97.5), width = 0.2) +
```

```
  coord_flip() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(
    y = "Coefficient Estimate",
    x = "Predictors"
  )

# plot nonzero prop
ggplot(coef_ci, aes(x = Predictor, y = Proportion_Nonzero)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  theme_bw() +
  labs(
    title = "Proportion of Bootstrap Samples with Non-Zero Coefficients",
    y = "Proportion",
    x = "Predictors"
  )




# --- Enforce Sparsity ---

# first criteria
# note: from experimentation we see (for the B <- 6000 case using log(NMR)) when
# threshold is set to .60, no additional variables are zeroed (no additional sparsity).
# The model over-fits in this case. However, when threshold is moved to even .605,
# just one variable is zeroed out which causes the calibration to improve slightly
# but destroys the AUC / discriminative ability. very interesting behavior to study
# later.
threshold <- 0
predictors_to_keep <- names(proportion_nonzero)[proportion_nonzero >= threshold]
coef_matrix_sparse <- coef_matrix

# columns (predictors) to zero out
predictors_to_zero <- setdiff(colnames(coef_matrix_sparse), predictors_to_keep)

# second criteria
# note: from experimentation we know this second criteria doesn't cause any coefs
# to go to zero in the tested setting (B = 6000).
zero_ci_predictors <- coef_ci$Predictor[
  round(coef_ci$Median, 4) == 0 &
  round(coef_ci$Lower_2.5, 4) == 0 &
  round(coef_ci$Upper_97.5, 4) == 0]

predictors_to_zero <- union(predictors_to_zero, zero_ci_predictors)

# zero out the coefs
if(length(predictors_to_zero) > 0){
```

```r
  coef_matrix_sparse[, predictors_to_zero] <- 0
  aligned_primary_coef[predictors_to_zero] <- 0
}

# coef_matrix_sparse <- coef_matrix # Uncomment to test not enforcing sparsity

coef_mean <- colMeans(coef_matrix_sparse, na.rm = TRUE)
coef_median <- apply(coef_matrix_sparse, 2, median, na.rm = TRUE)
coef_lower <- apply(coef_matrix_sparse, 2, quantile, probs = 0.025, na.rm = TRUE)
coef_upper <- apply(coef_matrix_sparse, 2, quantile, probs = 0.975, na.rm = TRUE)
proportion_nonzero <- colSums(nonzero_matrix, na.rm = TRUE) / sum(nonzero_matrix[,1])

coef_ci_sparse <- data.frame(
  Predictor = colnames(coef_matrix_sparse),
  Primary_Estimate = aligned_primary_coef,
  Mean = coef_mean,
  Median = coef_median,
  Lower_2.5 = coef_lower,
  Upper_97.5 = coef_upper,
  Proportion_Nonzero = proportion_nonzero
)

# Results Table

coef_ci_sparse <- coef_ci_sparse %>%
  mutate(
    #Odds_Ratio = exp(Median)  # Odds Ratio
    Odds_Ratio = exp(Primary_Estimate)  # Odds Ratio (now using main lasso est.)
  )

coef_ci_sparse <- coef_ci_sparse %>%
  mutate(
    Rounded_Lower_2.5 = round(Lower_2.5, 4),
    Rounded_Upper_97.5 = round(Upper_97.5, 4),

    # Determine significance
    Significant = ifelse(
      Rounded_Lower_2.5 > 0 | Rounded_Upper_97.5 < 0,
      "Yes",
      "No"
    )
  ) %>%
  select(-Rounded_Lower_2.5, -Rounded_Upper_97.5)

table_data <- coef_ci_sparse %>%
  select(
    Predictor,
    Primary_Estimate,
    Mean,
```

```
    Median,
    Odds_Ratio,
    Lower_2.5,
    Upper_97.5,
    Proportion_Nonzero,
    Significant
  ) %>%
  rename(
    "Estimate (Primary)" = Primary_Estimate,
    "Estimate (Mean)" = Mean,
    "Estimate (Median)" = Median,
    "Odds Ratio" = Odds_Ratio,
    "Lower 2.5%" = Lower_2.5,
    "Upper 97.5%" = Upper_97.5,
    "Proportion Non-Zero" = Proportion_Nonzero,
        "Sig." = Significant

  )

table_data_significant <- table_data %>%
  filter(`Sig.` == "Yes")

table_data_significant %>%
  kbl(row.names = F,
    booktabs = TRUE,
    longtable = TRUE,
    escape = TRUE,
    align = "c",
    digits = 3,
    caption = "Lasso Regression Coefficient Estimates"
  ) %>%
  column_spec(1, width = "3.2cm", latex_valign = "m") %>%
  column_spec(2, width = "1.5cm", latex_valign = "m") %>%
  column_spec(3, width = "1.5cm", latex_valign = "m") %>%
  column_spec(4, width = "1.5cm", latex_valign = "m") %>%
  column_spec(5, width = "1.5cm", latex_valign = "m") %>%
  column_spec(6, width = "1.5cm", latex_valign = "m") %>%
  column_spec(7, width = "1.5cm", latex_valign = "m") %>%
  column_spec(8, width = "1.5cm", latex_valign = "m") %>%
  column_spec(9, width = ".5cm", latex_valign = "m") %>%
  kable_styling(
    font_size = 7.6,
    latex_options = c("repeat_header", "striped"),
    full_width = FALSE,
    position = "center"
  )

### Plot omitted in revision as this is no longer significant
```

```r
# polynomial coefficients
shaps_coefs <- coef_ci[grep("poly\\(shaps_score_pq1, 3, raw = TRUE\\)", coef_ci$Predictor), ]
# interaction coefficients
shaps_baba_coefs <- coef_ci[grep("BABA:poly\\(shaps_score_pq1, 3, raw = TRUE\\)", coef_ci$Predict
#  BABA main effect
baba_coef <- coef_ci$Median[coef_ci$Predictor == "BABA"]

shaps_range <- seq(min(na.omit(data$shaps_score_pq1)),
                   max(na.omit(data$shaps_score_pq1)), length.out = 100)

# Function to calculate predicted values for a given BABA value
calculate_predictions <- function(baba_value) {
    predicted_log_odds <- coef_ci$Median[coef_ci$Predictor == "(Intercept)"] +
                          baba_coef * baba_value +
                          shaps_coefs$Median[1] * shaps_range +
                          shaps_coefs$Median[2] * shaps_range^2 +
                          shaps_coefs$Median[3] * shaps_range^3 +
                          baba_value * shaps_baba_coefs$Median[1] * shaps_range +
                          baba_value * shaps_baba_coefs$Median[2] * shaps_range^2 +
                          baba_value * shaps_baba_coefs$Median[3] * shaps_range^3

    predicted_probs <- 1/(1 + exp(-predicted_log_odds))

    return(data.frame(
        shaps_score = shaps_range,
        log_odds = predicted_log_odds,
        probability = predicted_probs,
        BABA = as.factor(baba_value)
    ))
}

plot_data <- rbind(
    calculate_predictions(0),
    calculate_predictions(1)
)

# plot log odds
p1 <- ggplot(plot_data, aes(x = shaps_score, y = log_odds, color = BABA)) +
    geom_line() +
    theme_minimal() +
    scale_color_discrete(labels = c("Standard", "BA")) +
    labs(x = "SHAPS Score",
         y = "Predicted Log Odds",
         title = "")

# plot probabilities
p2 <- ggplot(plot_data, aes(x = shaps_score, y = probability, color = BABA)) +
    geom_line() +
    theme_minimal() +
```

```
    scale_color_discrete(labels = c("Standard", "BA")) +
    labs(x = "SHAPS Score",
         y = "Predicted Probability",
         title = "")

shaps_plot <- ggarrange(
    p1,
    p2,
    ncol = 2,
    nrow = 1,
    common.legend = TRUE,
    legend = "bottom"
)
print(shaps_plot)


# --- Generate Predictions ---

# fix factor levels in the entire dataset (ensure consistency) BUGFIX
if (length(factor_vars) > 0) {
  for (var in factor_vars) {
    validation_data[[var]] <- factor(validation_data[[var]], levels = levels_list[[var]])
  }
}

# compute ROC for valid data

# prepare valid data
model_data_valid_full <- model.matrix(full_formula, data = validation_data)
current_coef_names_valid <- colnames(model_data_valid_full) # align cols
missing_cols <- setdiff(master_coef_names, current_coef_names_valid)
if (length(missing_cols) > 0) {
  zeros_matrix <- matrix(0,
                         nrow = nrow(model_data_valid_full),
                         ncol = length(missing_cols))
  colnames(zeros_matrix) <- missing_cols
  model_data_valid_full <- cbind(model_data_valid_full, zeros_matrix)
}
extra_cols <- setdiff(current_coef_names_valid, master_coef_names)
if (length(extra_cols) > 0) {
  model_data_valid_full <-
    model_data_valid_full[, !(colnames(model_data_valid_full) %in% extra_cols)]
}

model_data_valid_full <- model_data_valid_full[, master_coef_names] # reorder
#predictor_coefs <- coef_ci_sparse$Median
predictor_coefs <- coef_ci_sparse$Primary_Estimate # NOTE: changed from using $Median
names(predictor_coefs) <- coef_ci_sparse$Predictor
predictor_coefs <- predictor_coefs[master_coef_names] # ensure match
```

```r
# predictions for valid data
linear_predictor <- as.vector(model_data_valid_full %*% predictor_coefs)
pred_prob <- 1 / (1 + exp(-linear_predictor))

# BUGFIX
if (length(validation_data) < length(pred_prob)) {
  validation_data <- validation_data[complete.cases(validation_data),]
}

validation_data$pred_prob <- pred_prob



# compute ROC for training data

# prepare training data
data_complete_cc <- data_complete[complete.cases(data_complete),]
model_data_train <- model.matrix(full_formula, data = data_complete_cc)
current_coef_names_train <- colnames(model_data_train)
missing_cols_train <- setdiff(master_coef_names, current_coef_names_train)
if (length(missing_cols_train) > 0) {
  zeros_matrix_train <- matrix(0, nrow = nrow(model_data_train),
                               ncol = length(missing_cols_train))
  colnames(zeros_matrix_train) <- missing_cols_train
  model_data_train <- cbind(model_data_train, zeros_matrix_train)
}
extra_cols_train <- setdiff(current_coef_names_train, master_coef_names)
if (length(extra_cols_train) > 0) {
  model_data_train <-
    model_data_train[, !(colnames(model_data_train) %in% extra_cols_train)]
}
model_data_train <- model_data_train[, master_coef_names]

# predictions for training data
linear_predictor_train <- as.vector(model_data_train %*% predictor_coefs)
pred_prob_train <- 1 / (1 + exp(-linear_predictor_train))
data_complete_cc$pred_prob <- pred_prob_train


# ROCs
roc_valid <- roc(validation_data$abst, validation_data$pred_prob)
roc_train <- roc(data_complete_cc$abst, data_complete_cc$pred_prob)

# plot ROC curves
plot(roc_train, col = "red", main = "")
plot(roc_valid, add = TRUE, col = "blue")
legend("bottomright",
       legend = c(paste("Validation, AUC=", round(roc_valid$auc, 4)),
```

```
                  paste("Training, AUC=", round(roc_train$auc, 4))),
       fill = c("blue", "red"))

# calc residuals
data_complete_cc$residuals <- data_complete_cc$abst - data_complete_cc$pred_prob

# plot residuals v. fitted
ggplot(data_complete_cc, aes(x = pred_prob, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(x = "Predicted Probability", y = "Residuals") +
  ggtitle("Residuals vs. Predicted Probabilities")

# Histogram of Residuals
hist(data_complete_cc$residuals, main = "Histogram of Residuals")

# Q-Q Plot of Residuals
qqnorm(data_complete_cc$residuals)
qqline(data_complete_cc$residuals)

# calibration plot
num_cuts <- 24

# calibration for validation data
test_calib <- data.frame(
  prob = validation_data$pred_prob,
  bin = cut(validation_data$pred_prob, breaks = num_cuts),
  class = as.numeric(as.character(validation_data$abst))
)

test_calib <- test_calib %>%
  group_by(bin) %>%
  summarize(observed = sum(class)/n(),
            expected = sum(prob)/n(),
            se = sqrt(observed * (1-observed) / n()))

cols <- c("Ideal"="red","Loess Smooth"="black","LM Smooth"="blue")

ggplot(test_calib) +
  geom_abline(aes(intercept = 0, slope = 1, color="Ideal")) +
  geom_smooth(aes(x = expected,
                  y = observed,
                  color="Loess Smooth"), se=TRUE) +
  geom_smooth(aes(x = expected,
                  y = observed,
                  color="LM Smooth"), se=FALSE, method="lm") +
  scale_color_manual(values=cols) +
  labs(x = "Expected Proportion",
       y = "Observed Proportion",
```

```
     title="") +
theme_bw() + theme(legend.position = "bottom")
```

Appendix II: Functions Code

```
# Helper Functions for Smoking Cessation Project

# --- Preamble ---
# Date of last update: Oct. 17, 2024
# R Version: 4.3.1
# Package Versions:
#   outliers: 0.15
#   tidyverse: 2.0.0
#   reshape2: 1.4.4
#   moments: 0.14.1
#   vcd: 1.4-12


# Libraries for functions
suppressMessages(library(tidyverse))
library(outliers) # for grubbs.test()
library(reshape2) # for melt()
library(moments) # for skewness() and kurtosis()
library(vcd) # for assocstats()









# -- BEGIN Psuedo-correlation Matrix Section --

# Helper function to calculate Eta^2
eta_squared <- function(aov_model) {
  #' This function calculates the Eta^2 stat from an anova model. Eta^2 is a measure
  #' of effect size. Eta^2 describes the proportion of the total variance attributable
  #' to a given factor
  #'
  #' @param aov_model An aov object
  #' @return A numeric value of Eta^2 (proportion of the total variance explained)
  #'
  sum_of_squares_model <- summary(aov_model)[[1]]$"Sum Sq"[1]
  sum_of_squares_total <- sum(summary(aov_model)[[1]]$"Sum Sq")
  eta_sq <- sum_of_squares_model / sum_of_squares_total
  return(eta_sq)
}


# Main function to compute correlations or psuedo-correlations
# NOTE: Below still does not work with logical variables, but now fixed for NAs
```

```r
psuedo_cor_mat <- function(data) {
  #' Builds a pseudo correlation matrix for a given dataset, flexible for numerical
  #' and categorical variables. Uses Pearson correlation for numerical-numerical pairs,
  #' Cramer's V for categorical-categorical pairs, point biserial correlation for
  #' numerical-binary categorical pairs, and the square root of Eta^2 for
  #' numerical-multi-category pairs.
  #'
  #' @param data A df with any mix of numerical and categorical variables.
  #' @return A symmetric matrix with correlation coefficients
  #' (or equivalent measures) for all variable pairs.
  #'
  variables <- names(data)
  n <- length(variables)
  cor_matrix <- matrix(NA, n, n, dimnames = list(variables, variables))

  for (i in 1:n) {
    for (j in i:n) {
      if (i == j) {
        cor_matrix[i, j] <- 1
      } else {
        var_i <- data[[i]]
        var_j <- data[[j]]

        if (is.numeric(var_i) && is.numeric(var_j)) {
          # Pearson correlation for continuous-continuous
          cor_matrix[i, j] <- cor_matrix[j, i] <- cor(var_i, var_j,
                                                      use = "pairwise.complete.obs")

        } else if (is.factor(var_i) && is.factor(var_j)) {
          # Remove NAs for both variables
          valid_idx <- !is.na(var_i) & !is.na(var_j)
          var_i_clean <- var_i[valid_idx]
          var_j_clean <- var_j[valid_idx]

          # Drop unused levels
          var_i_clean <- droplevels(var_i_clean)
          var_j_clean <- droplevels(var_j_clean)

          # Cramer's V for categorical-categorical
          if (length(var_i_clean) > 0 && length(var_j_clean) > 0) {
            cramers_v <- sqrt(assocstats(table(var_i_clean, var_j_clean))$cramer)
            cor_matrix[i, j] <- cor_matrix[j, i] <- cramers_v
          } else {
            cor_matrix[i, j] <- cor_matrix[j, i] <- NA
          }

        } else {
          # Continuous-categorical pairs
          if (is.numeric(var_i) && (is.factor(var_j) || is.character(var_j))) {
```

```
            continuous <- var_i
            factor_var <- as.factor(var_j)
          } else if ((is.factor(var_i) || is.character(var_i)) && is.numeric(var_j)) {
            continuous <- var_j
            factor_var <- as.factor(var_i)
          } else {
            next  # Skip if variables are not appropriate types
          }

          # Remove NAs in both variables
          valid_idx <- !is.na(continuous) & !is.na(factor_var)
          continuous <- continuous[valid_idx]
          factor_var <- factor_var[valid_idx]

          # Drop unused levels
          factor_var <- droplevels(factor_var)

          if (length(unique(factor_var)) == 2) {
            # Point biserial correlation for binary factors
            factor_numeric <- as.numeric(factor_var) - 1
            cor_matrix[i, j] <- cor_matrix[j, i] <- cor(continuous, factor_numeric)
          } else if (length(unique(factor_var)) > 2) {
            # Eta-squared for multi-category factors
            if (length(factor_var) > 0) {
              aov_result <- aov(continuous ~ factor_var)
              eta_sq <- eta_squared(aov_result)
              cor_matrix[i, j] <- cor_matrix[j, i] <- sqrt(eta_sq)
            } else {
              cor_matrix[i, j] <- cor_matrix[j, i] <- NA
            }
          } else {
            # Cannot compute correlation if factor_var has insufficient levels
            cor_matrix[i, j] <- cor_matrix[j, i] <- NA
          }
        }
      }
    }
  }
  return(cor_matrix)
}

# -- END Psuedo-correlation Matrix Section --
```

```r
# Function to generate summary table aka "Table 1"
# NOTE: Does not like the following variable types: Date, `hms` num / difftime
# NOTE: Will likely need custom subroutines to handle dates / hms / difftime
summary_table <- function(df, stratify_var = NULL) {
  #' Create summary table for a data frame with option for stratification. It
  #' summarizes numeric variables by mean, standard deviation, and IQR.
  #' Categorical variables by count and percentage. Optionally stratifies by a
  #' specified variable.
  #'
  #' @param df A data frame
  #' @param stratify_var (Optional) A variable on which to stratify the summaries.
  #'
  #' @return A data frame (or tibble) containing the summarized statistics.

  # Internal function to handle the actual summarization
  summarize_internal <- function(df, stratify_var = NULL, is_recursive_call = FALSE) {

    # --- SUMMARIZER SECTION ---
    # Function to summarize continuous variable
    summarize_continuous <- function(data, var) {
      var_data <- data[[var]]
      mean_sd <- paste0(round(mean(var_data, na.rm = TRUE), 2),
                        " (",
                        round(sd(var_data, na.rm = TRUE), 2),
                        ")")
      quantiles <- quantile(var_data, probs = c(0.25, 0.75), na.rm = TRUE)
      quantile_range <- paste0("(",
                               round(quantiles[1], 2),
                               " - ",
                               round(quantiles[2], 2),
                               ")")
      paste(mean_sd, quantile_range)
    }

    # Func to summarize categorical variable
    summarize_categorical <- function(data, var) {
      var_data <- table(data[[var]])
      percentages <- prop.table(var_data) * 100
```

```
  paste(names(var_data), ": ",
        var_data, " (",
        round(percentages, 2),
        "%)",
        sep = "",
        collapse = ", ")
}

# Handler for strata
variable_names <- names(df)
if (!is.null(stratify_var) && stratify_var %in% variable_names) {
  variable_names <- variable_names[variable_names != stratify_var]
}

# Summarize all into a list for table
summary_list <- map(variable_names, ~ {
  var_type <- ifelse(is.numeric(df[[.x]]) | is.integer(df[[.x]]),
                     "Numeric", "Categorical")
  summarizer <- ifelse(var_type == "Numeric",
                       summarize_continuous,
                       summarize_categorical)
  summary <- summarizer(df, .x)
  variable_name <- ifelse(var_type == "Numeric",
                          paste(.x, "[Mean (SD) (Quantile)]"),
                          paste(.x, "[n (%)]"))
  tibble(Variable = variable_name, Type = var_type, Summary = summary)
})

# Bind to table
summary_table <- bind_rows(summary_list)

# --- NORMALITY AND OUTLIERS SECTION ---
# Perform Shapiro-Wilk and Grubbs tests only if it's not a recursive call
# and no stratify_var is given
if (!is_recursive_call && is.null(stratify_var)) {
  # Get numeric variables
  numeric_vars <- variable_names[sapply(df[variable_names],
                                        function(x) is.numeric(x) | is.integer(x))]

  # Init object for S-W / Grubbs results
  test_results <- map(numeric_vars, ~ {
    x <- df[[.x]]
    x <- na.omit(x)
    n <- length(x)
    normality <- NA
    outlier <- NA
    skewness_label <- NA
    kurtosis_label <- NA
```

```r
# Shapiro-Wilk test (Note: now accommodates n > 5000)
# NOTE: Consider in future -- KS test, AD test, Jarque-Bera test
if (n >= 3) {
  if (n > 5000) {
    shapiro_test <- shapiro.test(sample(x, 4999))
  } else {
    shapiro_test <- shapiro.test(x)
  }
  normality <- ifelse(shapiro_test$p.value < 0.05, "No", "Yes")
} else {
  normality <- "Insufficient data"
}

# Grubbs test
if (n >= 3) {
  grubbs_test <- grubbs.test(x)
  outlier <- ifelse(grubbs_test$p.value < 0.05, "Yes", "No")
} else {
  outlier <- "Insufficient data"
}

# Skewness and Kurtosis
if (n >= 2) {
  skewness <- skewness(x)
  kurtosis <- kurtosis(x)

  # Skewness label
  if (abs(skewness) < 0.5) {
    skewness_label <- "Centered"
  } else if (skewness > 0.5) {
    skewness_label <- "Right-skewed"
  } else {
    skewness_label <- "Left-skewed"
  }

  # Kurtosis label
  if (kurtosis < 2.5) {
    kurtosis_label <- "Platykurtic"
  } else if (kurtosis > 3.5) {
    kurtosis_label <- "Leptokurtic"
  } else {
    kurtosis_label <- "Mesokurtic"
  }
} else {
  skewness_label <- "Insufficient data"
  kurtosis_label <- "Insufficient data"
}

tibble(Variable = .x,
```

```
            `Normal Distribution` = normality,
            `Outlier(s) Present` = outlier,
            `Skewness` = skewness_label,
            `Kurtosis` = kurtosis_label)
  })

  # Bind test results into df
  test_results_df <- bind_rows(test_results)

  # Get actual variable names from summary_table for merge
  # Note: this could break in some datasets if the variable names include "["
  # therefore a more general solution later is preferred
  summary_table$ActualVariable <- gsub(" \\[.*\\]", "", summary_table$Variable)

  # Merge summary_table with test_results_df
  summary_table <- left_join(summary_table,
                             test_results_df,
                             by = c("ActualVariable" = "Variable"))

  # Remove temp matching ActualVariable column
  summary_table <- summary_table %>% dplyr::select(-ActualVariable)
} else if (!is.null(stratify_var)) {
  # --- STRATA SECTION ---

  # Stratification handling
  stratified_summaries <- df %>%
    group_by(!!sym(stratify_var)) %>%
    do(summarize_internal(., stratify_var = NULL, is_recursive_call = TRUE))

  summary_table <- stratified_summaries %>%
    ungroup() %>%
    dplyr::select(-group_cols()) %>% # This doesn't seem necessary, why'd I do it?
    pivot_wider(names_from = !!sym(stratify_var), values_from = Summary)

  # Function for significance testing (numeric variables)
  test_continuous <- function(data, var, group_var) {
    formula <- as.formula(paste0("`", var, "` ~ `", group_var, "`")) #BUGFIX: spaces
    num_groups <- length(unique(data[[group_var]]))
    if (num_groups == 2) {
      test_result <- t.test(formula, data = data)
      p_value <- test_result$p.value
    } else {
      test_result <- kruskal.test(formula, data = data)
      p_value <- test_result$p.value
    }
    p_value
  }

  # Function for significance testing (categorical variables)
```

```r
test_categorical <- function(data, var, group_var) {
  table_data <- table(data[[var]], data[[group_var]])
  table_data <- table_data[rowSums(table_data) > 0, # This prevents NaN error
                           colSums(table_data) > 0, # in ChiSq test.
                           drop = FALSE]
  test_result <- chisq.test(table_data)
  p_value <- test_result$p.value
  p_value
}

# Perform significance testing (if stratify_var is specified)
significance_tests <- map(variable_names, ~ {
  var_type <- ifelse(is.numeric(df[[.x]]) | is.integer(df[[.x]]),
                     "Numeric", "Categorical")
  tester <- ifelse(var_type == "Numeric",
                   test_continuous,
                   test_categorical)
  p_value <- tester(df, .x, stratify_var)
  tibble(Variable = .x, `P-value` = p_value)
})

significance_table <- bind_rows(significance_tests)

# Adjust p-values using Bonferroni correction
significance_table$`Adjusted P-value` <- p.adjust(significance_table$`P-value`,
                                                  method = "bonferroni")

# Get actual variable names from summary_table for merge
# Note: this could break in some datasets if the variable names include "["
# therefore a more general solution later is preferred
summary_table$ActualVariable <- gsub(" \\[.*\\]", "", summary_table$Variable)

# Merge summary_table with significance_table on actual variable names
summary_table <- left_join(summary_table,
                           significance_table,
                           by = c("ActualVariable" = "Variable"))

# Remove rows corresponding to stratify_var
summary_table <- summary_table %>% filter(ActualVariable != stratify_var)

# Create 'Sig.' column based on Adjusted P-value
summary_table$`Sig.` <- case_when(
  summary_table$`Adjusted P-value` <= 0.0001 ~ "****",
  summary_table$`Adjusted P-value` <= 0.001 ~ "***",
  summary_table$`Adjusted P-value` <= 0.01  ~ "**",
  summary_table$`Adjusted P-value` <= 0.05  ~ "*",
  TRUE                                      ~ "ns"
)
```

```
      # Remove temp matching ActualVariable, adjusted/unadjusted P-Values
      summary_table <- dplyr::select(summary_table,
                                     c(-ActualVariable,
                                       -`P-value`,
                                       -`Adjusted P-value`))
    }

    # Remove "Type" column for stratified tables
    if (!is.null(stratify_var) && !is_recursive_call) {
      summary_table <- summary_table %>% dplyr::select(-Type)
    }

    return(summary_table)
  }

  # Call internal function to initiate procedure
  summarize_internal(df, stratify_var)
}
```

```
# Function to count number of variables where missing data occours
count_missing_vars <- function(df) {
  #' Calc the number of variables in a data frame that have at least one missing
  #' value.
  #'
  #' @param df A data frame
  #'
  #' @return A data frame with a single row and column indicating the count
  #' of variables with any missing values.
  #'
  missing_count <- df %>%
    summarise_all(~ sum(is.na(.))) %>%
    pivot_longer(everything()) %>%
    filter(value > 0) %>%
    nrow()
```

```r
  return(data.frame(Num_Missing_Vars = missing_count))
}
```

```r
# Count by variable type
summarize_variables_types <- function(data) {
  #' Summarizes the types of variables in a dataset by type. Possible types:
  #' Numeric, or Categorical (factors or logical).
  #'
  #' @param data A data frame
  #' @return A table with the counts of variables categorized by type
  #'
  var_types <- sapply(data, function(x) {
    if(is.numeric(x) | is.integer(x)) {
      return("Numeric")
    }
    else {
      return("Categorical")
    }
  })

  var_counts <- table(var_types)
  return(var_counts)
}
```

```r
# Function to find groups of correlated variables
findCorrelatedGroups <- function(corr_matrix, threshold) {
  #' Find Groups of Correlated Variables
  #'
  #' This function identifies groups of variables in a correlation matrix that
  #' exceed a specified correlation threshold.
  #'
  #' @param corr_matrix A square symmetric matrix representing variable correlations.
  #' @param threshold A numeric threshold for defining significant correlation.
  #' @return A list of numeric vectors, each containing indices of a group of
  #' correlated variables.
  correlation <- abs(corr_matrix)
  diag(correlation) <- 0 # blank out diagonal to ignore self-correlation
  groups <- list()
  visited <- rep(FALSE, ncol(correlation))

  for (i in 1:ncol(correlation)) {
    if (!visited[i]) {
      # Find index of variables correlated w/ the current var
      high_cor_vars <- which(correlation[, i] > threshold)
      if (length(high_cor_vars) > 0) {
        group <- unique(c(i, high_cor_vars)) # include current var in the group
        groups[[length(groups) + 1]] <- group
        visited[group] <- TRUE
      }
      else {
        visited[i] <- TRUE
      }
    }
  }
  return(groups)
}




# Function to drop select highly correlated vars
reduceDataset <- function(df, corr_matrix, threshold) {
  #' Reduce Dataset by Dropping Highly Correlated Variables
  #'
  #' This function reduces a dataset by identifying and dropping highly correlated
  #' variables based on the provided correlation matrix and threshold.
  #'
  #' @param df A dataframe containing the dataset to be reduced.
  #' @param corr_matrix A square symmetric matrix of correlations between
  #' variables in `df`.
  #' @param threshold A numeric threshold to identify high correlations.
  #' @return A dataframe with reduced variables.
```

```r
  groups <- findCorrelatedGroups(corr_matrix, threshold)
  to_drop <- numeric()

  for (group in groups) {
    # Subroutine to select which to drop (inverse of variance explained)
    variances <- sapply(group, function(index) {
      # Below is to fix bug with fread/data.table (depending on how data is read in)
      if (is.data.table(df)) {
        var(as.vector(df[[index]]))
      } else {
        var(df[, index, drop = FALSE])
      }
    })

    # Drop variable with minimum variance
    min_variance_index <- which.min(variances)

    to_drop_var <- group[min_variance_index]

    to_drop <- c(to_drop, to_drop_var)
  }

  to_drop <- unique(to_drop)

  # only drop columns that exist in the dataset (bug fix)
  to_drop <- to_drop[to_drop <= ncol(df)]

  if (length(to_drop) > 0) {
    reduced_df <- dplyr::select(df, -to_drop)
  }
  else {
    reduced_df <- df # If nothing to drop, return the original dataset (bug fix)
  }

  return(reduced_df)
}
```

```r
check_mar <- function(data) {
  #' Check Missing At Random (MAR) in Data
  #'
```

```r
#' This function tests each variable in the dataset for missingness being at
#' random, conditional on all other variables, using logistic regression models.
#'
#' @param data A dataframe with variables to test for MAR.
#' @return A list where each entry corresponds to a variable in `data` and
#' contains results of MAR tests.


all_vars <- names(data)

# Init object for results
results <- list()

for (var in all_vars) {
  # Generate missing indicator
  missing_var_name <- paste0("missing_", make.names(var))
  data[[missing_var_name]] <- as.integer(is.na(data[[var]]))

  # formula for GLM
  formula_vars <- sapply(all_vars[all_vars != var], function(v) paste0("`", v, "`"))
  formula_str <- paste0(missing_var_name, " ~ ", paste(formula_vars, collapse = " + "))
  formula <- as.formula(formula_str)

  model <- tryCatch(
    {
      glm(formula, data = data, family = binomial())
    },
    error = function(e) {
      return(NULL)
    }
  )

  if (!is.null(model)) {
    # check for sig. variables
    summary_model <- summary(model)
    pvalues <- summary_model$coefficients[, "Pr(>|z|)"]
    significant_vars <- names(pvalues[pvalues < 0.05 & !is.na(pvalues)])

    # removve intercept from significant vars (if present)
    significant_vars <- significant_vars[significant_vars != "(Intercept)"]

    # rm backticks from significant vars for cleaner output
    significant_vars <- gsub("`", "", significant_vars)

    results[[var]] <- list(
      is_mar = length(significant_vars) == 0,
      significant_vars = significant_vars
    )
  } else {
```

```
      results[[var]] <- list(
        is_mar = NA,
        significant_vars = NA,
        error = "Error in fitting GLM"
      )
    }

    # remove temporary missing indicator col
    data[[missing_var_name]] <- NULL
  }

  return(results)
}




calculate_mse <- function(actual, predicted) {
  #' Calculate Mean Squared Error
  #'
  #' This function computes the mean squared error between actual and predicted
  #' numerical values.
  #'
  #' @param actual A numeric vector of actual values.
  #' @param predicted A numeric vector of predicted values.
  #' @return Numeric value representing the mean squared error.

  # inputs must be numeric
  actual <- as.numeric(actual)
  predicted <- as.numeric(predicted)

  # calc squared differences
  squared_diff <- (actual - predicted)^2

  # Remove NA / infinite values
  valid_diff <- squared_diff[is.finite(squared_diff) & !is.na(squared_diff)]

  # calc MSE
  mse <- sum(valid_diff) / length(valid_diff)

  return(mse)
}



variable_importance_reg <- function(model, data, target_column, n_iterations = 20) {
  #' Calculate Variable Importance for Linear Regression Models
  #'
  #' This function estimates the importance of each predictor variable in a
```

```r
  #' regression model by measuring the increase in prediction error after
  #' permuting each predictor variable.
  #'
  #' @param model A regression model object (e.g., lm, lmer).
  #' @param data A dataframe containing the data used in the model.
  #' @param target_column The name of the target (dependent) variable in `data`.
  #' @param n_iterations The number of iterations to perform for estimating
  #' importance (default is 20).
  #' @return A dataframe with variables and their relative importance scores.

  predictor_vars <- all.vars(formula(model))[-1]
  original_preds <- as.numeric(predict(model, data))
  original_mse <- calculate_mse(data[[target_column]], original_preds)

  importance_scores <- matrix(0, nrow = n_iterations, ncol = length(predictor_vars))
  colnames(importance_scores) <- predictor_vars

  for (i in 1:n_iterations) {
    for (var in predictor_vars) {
      data_shuffled <- data
      data_shuffled[[var]] <- sample(data_shuffled[[var]])

      shuffled_preds <- as.numeric(predict(model, data_shuffled))
      shuffled_mse <- calculate_mse(data_shuffled[[target_column]], shuffled_preds)

      importance_scores[i, var] <- shuffled_mse - original_mse
    }
  }

  avg_importance <- abs(colMeans(importance_scores))
  se_importance <- apply(importance_scores, 2, sd) / sqrt(n_iterations)

  result <- data.frame(
    Variable = predictor_vars,
    Importance = avg_importance,
    SE = se_importance
  )

  result <- result[order(-result$Importance), ]
  result$`Relative Importance` <- result$Importance / max(result$Importance) * 100

  return(result)
}
```

```r
select_best_model <- function(model_backward, model_forward) {
  #TODO: INSERT ROXYGEN
  # Compare the formulas of the two models
  if ((formula(model_backward) == formula(model_forward))) {
    # Formulas are the same, models are the same
    return(model_backward)
  } else {

    # --- Formulas are different ---

    # Extract terms from models
    terms_backward <- attr(terms(model_backward), "term.labels")
    terms_forward <- attr(terms(model_forward), "term.labels")

    # Check if models are nested -- BUG FIX 10/17/24
    if (all(terms_backward %in% terms_forward)) {
      # Backward model is nested within forward model
      smaller_model <- model_backward
      larger_model <- model_forward
    } else if (all(terms_forward %in% terms_backward)) {
      # Forward model is nested within backward model
      smaller_model <- model_forward
      larger_model <- model_backward
    } else {
      # Models are not nested; cannot perform LRT therefore we compare BIC to
      # select the best model
      aic_backward <- BIC(model_backward)
      aic_forward <- AIC(model_forward)

      if (aic_backward < aic_forward) {
        return(model_backward)
      } else {
        return(model_forward)
      }
    }

    # Perform LRT
    lrt_result <- tryCatch({
```

```
      anova(smaller_model, larger_model, test = "LRT")
    }, error = function(e) {
      stop("Error in performing LRT: ", e$message)
    })

    p_value <- lrt_result$`Pr(>Chi)`[2]

    if (is.na(p_value)) {
      stop("LRT failed: p-value is NA")
    }

    # Decide which model to choose based on LRT
    if (p_value < 0.05) {
      return(larger_model) # Larger model is significantly better
    } else {
      return(smaller_model) # Smaller model is sufficient

    }
  }}
```

```
create_cross_table <- function(data) {
  vars <- names(data)
  n_vars <- length(vars)

  results <- list()

  for (i in 1:(n_vars-1)) {
    for (j in (i+1):n_vars) {
      var1 <- vars[i]
      var2 <- vars[j]

      # variable types
      type1 <- class(data[[var1]])[1]
      type2 <- class(data[[var2]])[1]

      # result based on variable types
```

```
      if (type1 %in% c("factor", "character") && type2 %in% c("factor", "character")) {
        result <- table(data[[var1]], data[[var2]])

      } else if (type1 %in% c("numeric", "integer") && type2 %in% c("numeric", "integer")) {
        result <- cor(data[[var1]], data[[var2]], use = "complete.obs")

      } else {
        if (type1 %in% c("factor", "character")) {
          result <- tapply(data[[var2]], data[[var1]], mean, na.rm = TRUE)
        } else {
          result <- tapply(data[[var1]], data[[var2]], mean, na.rm = TRUE)
        }
      }

      #  result
      results[[paste(var1, "x", var2)]] <- result
    }
  }

  return(results)
}
```