STAT-S 610 Final Project: A Forward Stepwise Regression Approach for Matched Difference-in-Differences Models

Luis Navarro

December 15, 2022

Introduction

Difference-in-difference models are widely used across fields of social science to test the effectiveness of an intervention/policy. Aiming to mimic the experimental setting of a randomized control trial, the difference-in-difference model compares the outcome of interest between two arms/groups: a treatment group and a control group. However, unlike controlled experiments, in observational studies treatment is likely to not be randomly assigned, therefore posing a challenge in terms of identifying the treatment effect through a standard linear regression model of the outcome (as the dependent variable) on the treatment (as the independent variable). To overcome this challenge, we need to make a comparison of the treatment and control groups, before and after the intervention.

Let i denote units and t periods. Hence, y_{it} denotes outcome variable y for unit i on period t. Let T_i be a dummy variable equal to 1 if unit i is in the treatment group and 0 otherwise. Let t^* be the period when the intervention takes place. Denote P_t as a dummy variable equal to 1 if $t \geq t^*$ (i.e. after the intervention) and 0 otherwise. Define $d_{it} = T_i \times P_t$ as the interaction between these two variables. Let a_i be unit fixed-effects (i.e. time-invariant characteristics specific to each unit) and b_t time fixed-effects (i.e. unit-invariant characteristics specific to each period). Let X_{it} be a vector of covariates, and e_{it} a random disturbance. The generalized difference-in-difference model observes the following structure.

$$y_{it} = \beta_0 + \beta d_{it} + \gamma X_{it} + a_i + b_t + e_{it} \tag{1}$$

For β to retrieve the Average Treatment Effect (ATE) we must satisfy the parallel trends assumption (i.e. the main identification assumption of this research design). Intuitively speaking, the parallel trends assumption requires that in the absence of treatment, the outcome of the treatment group should follow a similar trend to the one observed in the control group. If this holds, hence any differences observed in the outcome of the treatment group after the intervention could be attributed to the treatment. For this assumption to be met, however, the selection of the control group is crucial. In a perfect world, we would like a control group that is identical to the treatment group where the only difference observed between the two could be attributed to the analyzed intervention.

Matched Difference-in-Differences

In the econometrics literature, a method that has become widely used by scholars to address potential bias stemming from deviations of the parallel trends assumption is to use weighted least squares to estimate Equation 1, where the weights assign more relevance to observations both in the treatment and control group that are more "similar" according to their probability of receiving the treatment. These weights are called *Inverse Probability Weights* as they stem from a propensity score model with the following structure.

Let g denote the Logit link function. Hence, the following logistic regression model predicts the probability of being assigned to the treatment group, conditional on a set of pre-determined characteristics Z.

$$T_i = g(\beta_0 + \theta Z_i + u_i) \tag{2}$$

Denote p_i as the predicted value from fitting the previous regression model. Hence, the inverse probability weights are defined as $w_i = \frac{1}{p_i}$ if $T_i = 1$ and $w_i = \frac{1}{1-p_i}$ if $T_i = 0$. In summary, the process to estimating a matched difference-in-difference model is:

1. Estimate Equation 2
2. Compute the IPWs w_i using the predicted values from Equation 2
3. Use the IPWs and estimate Equation 1 using weighted least squares.

Empirical Challenges and My Proposed Solution

One of the main challenges when implementing matched difference-in-differences is to get a model that reflects accurately the data-generating process for treatment assignment. If the model is specified wrongly, then the weights might heighten the prevailing bias on the estimate for the ATE. To address this concern, I propose a forward step wise regression algorithm that will explore a set of potential models for the treatment assignment rule and choose the one with highest accuracy (lowest prediction error) to estimate the IPWs and use them to alleviate potential bias on the difference-in-difference model.

In a nutshell, the forward stepwise regression algorithm does the following:

1. For a given vector of variables z, define a vector $b(z)$ that includes all the variables in z as well as their squared terms, inverse hyperbolic sin transformation, and all the possible interactions between the variables.
2. For each element $x \in b(z)$, run the logistic regression model of T_i on x and the chosen predictors (for the first lap of the loop, this is an empty vector). Compute the root mean squared prediction error (RMSPE). Denote $RMSPE(x)$ as the prediction error of the logistic regression model when x is the vector of independent variables used in the model. Hence, I'll have a vector R that contains all the $RMSPE(x)$ of each variable x.
3. Choose the x such that $RMSPE(x) = \underset{x}{\text{argmin}} R$, store that x in a list of chosen predictors, and remove it from the vector $b(z)$.
4. Repeat step 2 and 3 until we ran out of independent variables or until there is no improvement in the accuracy of the model.

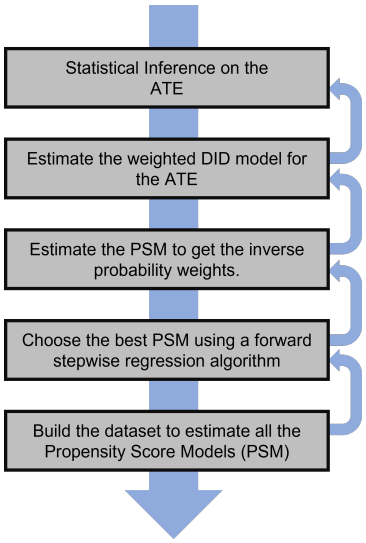
Once I chose the model that better predicts treatment assignment in the observed population, I compute IPWs with such model and use them to estimate the Average Treatment Effect. I compute standard errors and p-values using randomization inference (i.e. estimating a placebo distribution of the Average Treatment Effect using random draws from a uniform distribution as treatment variables).

Implementation

To implement this algorithm I rely on 5 main functions that will take as input a data frame with all the variables required to estimate the Propensity Score Model, or the Difference-in-Difference model, accordingly.

Also, I wrote two auxiliary functions in order to perform statistical inference for the Average Treatment Effect. The first auxiliary function creates a random treatment assignment rule according to a uniform distribution. Then, I use this randomly assigned treatment T_i to estimate an empirical distribution of the ATE, under random assignment. With this distribution, I can compute standard errors, rank-based p-values. The second function takes this empirical distribution and creates a graph that shows its kernel density and compares it with the estimated Average Treatment Effect. This provides a visual representation of the null hypothesis for statistical significance.

The following diagram depicts the top-down design of my implementation.



Functions Description

The following table summarizes the input-output relation between the written functions, as well as the type of each input. It will serve as guide for keeping track of the relation across the objects in the environment.

Function	Input (type): description	Output (type): description
did_randomization	• outcome (string): name of the outcome variable for the DID model on the data frame. • treat_var (string): name of the variable that denotes treatment status. • treat_period (double): intervention period. • time_var (string): name of the variable that identifies periods t on the data frame. • unit_var (string): name of the variable that identifies units i on the data frame. • weights (string): name of the variable that contains the IPW on the data frame. • data (data frame): name of the data frame to estimate the DID model. • samples (double): number of samples to perform statistical inference.	• output (list): list of two elements ▪ model_results (data frame): data frame showing the Average Treatment Effect, the standard error, and its p-value ▪ empirical_distribution (data frame): data frame with the results from the randomization inference. This output is used to do the hypothesis test graph.
did_model	• outcome (string): name of the outcome variable for the DID model in the data frame. • treat_var (string): name of the variable that denotes treatment status. • treat_period (double): intervention period. • time_var (string): name of the variable that identifies periods t on the supplied data frame. • unit_var (string): name of the variable that identifies units i on the supplied data frame. • weights (string): name of the variable that contains the IPW supplied data frame. • data (data frame): name of the data frame to estimate the DID model.	• model_did (list): list with all the elements from the regression model estimated using the "fixest" library from R.
psm_weights	• outcome (string): name of the variable that denotes treatment status. • predictors (characters): vector of names (strings) with all the variables that will be used to fit the PSM. • unit_var (string): name of the variable that identifies units i on the supplied data frame. • data (string): name of the data frame to estimate the PSM model.	• psm_results (list): list of 3 elements. ▪ model (list): list with two output elements from the model selection function described below. ▪ psm_weights (data frame): data frame equal to the data fed to the function to estimate the PSM model, but including the IPW as an additional variable named "ipw" ▪ rmspe (double): root mean squared prediction error from the model used to estimate the IPWs.
model_selection	• outcome (string): name of the variable that denotes treatment status. • predictors (characters): vector of names (strings) with all the variables that will be used to fit the PSM. • unit_var (string): name of the variable that identifies units i on the supplied data frame. • data (string): name of the data frame to estimate the PSM model.	• results (list): list of two elements. ▪ predictor_star (character): vector with the names of the chosen variables from the stepwise regression algorithm. ▪ rmspe_master (matrix): vector with the prediction accuracy (RMSPE) for each model, as the algorithm increases the number of independent variables in the regression.
data_build	• outcome (string): name of the variable that denotes treatment status. • predictors (characters): vector of names (strings) with all the variables that will be used to fit the PSM. • unit_var (string): name of the variable that identifies units i on the supplied data frame. • data (string): name of the data frame to estimate the PSM model.	• reg_data (data frame): data frame with all the potential variables that could be used to estimate the PSM model

1. **did randomization**: since getting a measure of the reliability of the estimates (i.e. standard errors and p-values) is the last step when estimating a regression model, this is my top function. This function takes as inputs all the variables needed to estimate the DID model, estimates it using the IPW stemming from the model with highest prediction accuracy and performs statistical inference using randomization inference.

2. **did model**: this function takes as input all the variables required to estimate the DID model, and uses a fixed-effects estimator ("fixest" library) to compute the coefficient of the ATE by running Equation 1.

3. **psm weights**: this function estimates the logit regression model with the set of predictors that yields the highest prediction accuracy from all the explored models. With the fitted values of such model, inverse probability weights are computed and stored in a new data frame identical to the data used to estimate the model, but with an additional variable for the weights. This function also reports the accuracy metrics of the model (i.e. the Root Mean Squared Prediction Error).

4. **model selection**: this function iteratively estimates propensity score models using different combinations of the potential set of predictors. Here is where the forward stepwise regression algorithm is implemented. As output it reports a vector with the names of the chosen variables, along with a matrix showing how the prediction accuracy improved as an additional variable was included in the model.

5. **data build**: this function delimits the space from which the selection algorithm will pick the model. It takes as input a list of potential predictors and creates a data frame including such predictors, along with some transformations of them (i.e. squared values and inverse hyperbolic sine transform) and the interactions they could have with each other.

Remarks on design decisions

- **Input structure:** adhering to the notion behind top-down design, I make that all my functions call upon each other from top to down. To avoid R's lazy evaluation, I specified that all the functions need to specify all the required variables to estimate the regression models. If a variable is not included, the function shows an error message. This input structure, however, obeys to the nature of the regression model estimated at each step. For example, the last three functions are related with the estimation of the IPWs through the logistic regression. Hence, I need to at least specify the outcome variable, the set of predictors (independent variables for the regression), and the unit variable. This last one is relevant as it serves as the link between the dataset used for the propensity score model and the one used for the difference-in-differences model.

  Moreover, I included lines of code to show error messages in case inputs were not defined, and warning messages in case the weights were not specified for the estimation of the DID model. Yet, I only included these lines of code on the *data build* and *did model* functions. The reason for this is the nested structure I imposed on the functions. Since each function calls the one next to it at the top down structure, the code will naturally evaluate first the last function. Hence, it suffices that only the last function on the chain includes these lines of code.

- **Stepwise Selection Algorithm:** I chose to write this algorithm using a while loop to potentially save some computational time. For a given vector of predictor variables z with cardinality n, the data build function will create a data frame with $2+3n+\frac{n!}{2(n-2)!}$ variables. The first 2, accounts for the unit and outcome variables, the $3n$ for three versions of the each variable in z: as it is on the data set, and both the squared and inverse hyperbolic sine transformations, and the last term accounts for all the possible combinations of the untransformed variables. Since this expression involves a factorial term, as the number of variable increases, the number of potential models increases more than exponentially. Hence, testing all the potential models seems to be a daunting lengthy and potentially unnecessary task. What I did, instead, was to set the algorithm to explore individually all the potential predictors and keep adding them to the vector of chosen variables if they improve the prediction accuracy of the model. The loop stops when the prediction error stops to decrease. Intuitively, if my algorithm works properly, it should retrieve the model with highest prediction accuracy.

- **Potential Predictors:** for this version of the project I chose to test only the squared and inverse hyperbolic sine transformation of the variables, as well as the linear interactions between each other. The motivation for this is to approximate a second order polynomial of the variables, where I am allowing for the variables to show either a convex (i.e. quadratic) or concave (i.e. inverse hyperbolic

sine) relation with the outcome. I chose the inverse hyperbolic sine to avoid the problems that logs have with zeros.

Simulation Parameters

To test whether my algorithm is effective addressing the underlying bias of the ATE estimate, I use data from Annual Census of Local Government Finances, the U.S. Census, and Bureau of Labor Statistics to build a dataset that contains yearly observations from a sample of 1,100 U.S. counties, following from the period 2010-2020.

My experimental setting consists in analyzing a fake intervention that affects the revenue from the property income tax perceived by county governments. To simulate the potential complexity surrounding the unobserved treatment assignment rule I use an arbitrarily complex function h of a vector of economic variables x to determine treatment assignment. If $h(X_i) > h^*$, then the unit is treated, where h^* is random scalar drawn from a normal distribution with mean 0.4 and standard deviation 0.15. I chose these parameters to induce some random unbalancedeness in the panel. In practice, this is part of the complexities faced when estimating difference-in-difference models.

```
### Non Random Treatment Assignment -- Complex Non-Linear Data Generating Process
treat_formula = unemployment_rate*female + sqrt(age4564) + sqrt(taxes_pc/exp_pc) + currexp^2 +
                log(1 + college) + log(popestimate) + rnorm(nfips, mean = 0, sd = 1),
treat_aux = percent_rank(treat_formula),
### If above the cutoff, the county is treated
treat_det = case_when(treat_aux <= cutoff_treatment_assignment ~ 0,
                      treat_aux > cutoff_treatment_assignment ~ 1),)
```

To test whether the model is retrieving the right parameter, I create a fake outcome using data from property tax revenue and applying the treatment effect to the units that were assigned to the treatment arm under assignment rule h. Recalling that $did = treat \times post$ I define this fake outcome using the linear structure of the difference-in-difference model. This is equivalent to assume the difference-in-difference model is correctly specified (i.e. is not vulnerable to endogeneity in the error term/ omitted variable bias). I made this simplifying assumption since my main interest lies in the specification of the inverse probability weights, hence I run the simulation assuming I have the right data generating process for the outcome of interest.

```
fake_outcome = logproptax + ate_beta*(did) + treat_det + post + fips_fe + year_fe + epsilon)
```

For the simulation I use 2015 as the treatment period in order to have a symmetric comparison window of 5 periods before and after the intervention. For simplicity, I assume a fixed value of 7 for the Average Treatment Effect. I perform statistical inference using a placebo distribution of a 1000 samples. Given the parameters described above, this is the composition of the treatment and control group in the simulation leads to a distribution of 70% - 30%, between the treatment and control arm of the study, respectively.

Simulation Results

The following table shows the results from estimating the DID model without any weights. As we can see, in this example the complexity of the data generating process is inducing some bias on the ATE estimate.

```
##                 unweighted_mode..
## Dependent Var.:           outcome
##
## did_var         6.956*** (0.0517)
## Fixed-Effects:  -----------------
```

```
## unit_var                       Yes
## time_var                       Yes
## ---------------- ------------------
## S.E.: Clustered      by: unit_var
## Observations              12,870
## R2                       0.95254
## Within R2                0.70255
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The following graph shows the RMSPE as the number of predictors included in the propensity score model increases. The panel shows four graphs, each for a different number of predictor variables in vector z. The first one shows the case when there are only two potential predictor variables. Hence, the algortihm explores only 3 candidate (i.e. $||b(z)|| = 3$), where the fully saturated one yields the lowest prediction error. The second panel on the top shows the case when there are 4 predictor variables in vector z. Hence, $||b(z)|| = 20$. However, as we can see the algorithm stops when it reaches 16 predictors, and does not evaluates the last four as it reached a point where accuracy was not improved by adding variables. Similar stories are observed for the two graphs at the bottom.

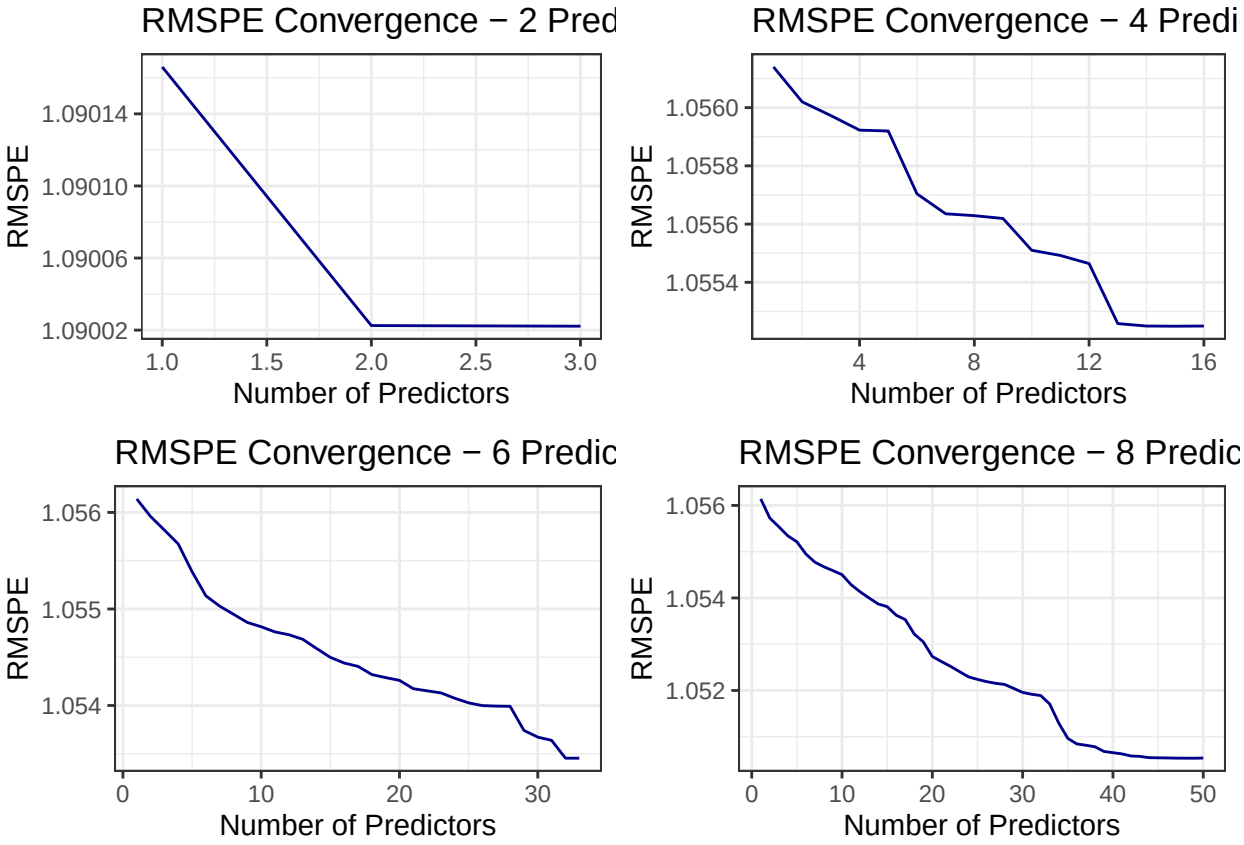


Table 1 shows the coefficient estimates for the ATE as the number of variables in the predictor vector z increases. In this baseline simulation it is clear the algorithm is effective improving the prediction accuracy of the ATE. Recall the theoretical value used for this simulation is 7. Table 2 compares the result from the model with highest prediction accuracy from Table 1 against the results from the unweighted regression model.

To show how the randomization inference algorithm works I show a graph with the empirical placebo distribution for the ATE coming from the model with highest treatment prediction accuracy. As expected, the placebo distribution is centered around zero and observing a large enough ATE coming from the DID model

Table 1: Average Treatment Effect Estimates

ATE	SE	RMSPE
6.9575	0.2517	1.0900
6.9724	0.5552	1.0560
6.9724	0.5420	1.0560
6.9613	0.6620	1.0552
6.9648	0.6782	1.0541
6.9777	0.6083	1.0535
6.9933	0.8200	1.0515
7.0140	0.7801	1.0505

Table 2: Unweighted vs Weighted Model

	Unweighted	IPW
ATE	6.9562	7.0140
SE	0.0419	0.7801

suggests the treatment effect is statistically significant. Observing the weird lumps near the mean of the distribution, however, is something that I was not expecting from the simulation.

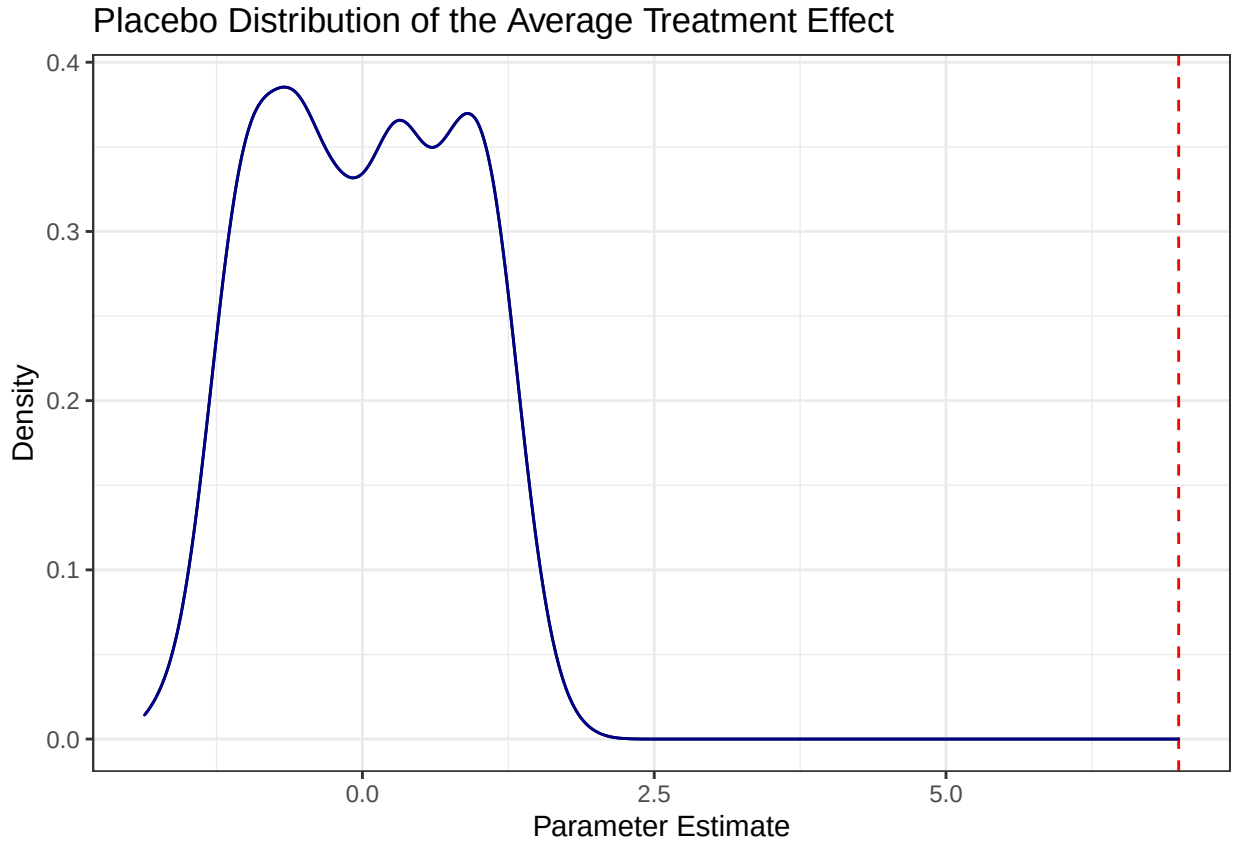


Table 3: Prediction Error

RMSPE1	RMSPE2
0.1024	0.1106

Tests

I developed a set of simple tests to verify the functions are working as they should. As mentioned above, I included some lines of code to display an error message in case some of the inputs is not defined. I formally test whether the functions show these error messages using the *testthat* library. This is a list of all the tests I implemented for my functions:

1. Verify the warning and error messages are displayed correctly. Error messages appear when there is a missing argument in the function. Warning messages appear when such missing argument has a default option. In case of the weights, is a vector of equal weight and for the number of samples my default option is 100.
2. Verify the estimation of the ATE is the same regardless forgets to specify the weights (i.e. missing argument) or the weights are set to NULL.
3. Verify the data build function is computing correctly the right number of potential variables. To test that, I compare the dimensions of the output data frame from such function and the result from calculating manually the number of potential variables. As mentioned above, this latter one is given by the following equation. Following the notation above, I am calculating the cardinality of vector $b(z)$, which I will denote with n.

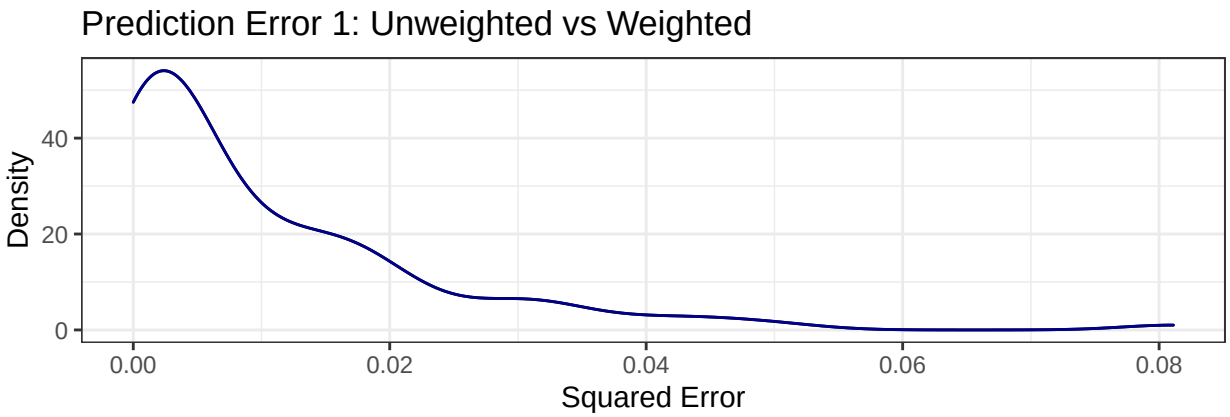
$$n = 2 + 3n + \frac{n!}{2(n-2)!} \tag{3}$$

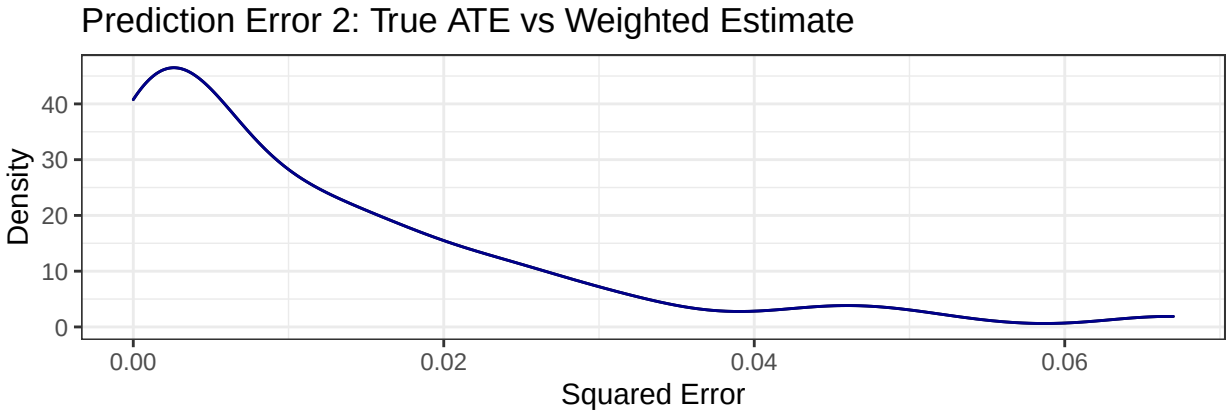
   The R script named '$ScriptTests.R$' contains all the details on the implementation of these tests.

Finally, I did an informal simulation test to assess to which extent the algorithm improves the estimation of the ATE for different potential values this parameter could take. Hence, I took a random draw of 1000 samples from a uniform distribution ranging from 1 to 10. Without losing generality, to make the code run faster I simplified the testing environment to just include a vector of 4 potential covariates. Then, I implemented the algorithm assuming the true ATE was a sample of such distribution and computed two types of prediction errors:

- Prediction Error 1: compares the coefficient estimates from the unweighted and IPW weighted regression
- Prediction Error 2: compares the coefficient estimate from the weighted regression with the true theoretical value of the ATE.

Intuitively, the first prediction error should show the overall improvement the algorithm has on reducing the bias from estimating the DID model without IP weights. The second prediction error describes the average bias of the algorithm. If this technique is indeed improving the estimation accuracy of the model, then the expected value of the second prediction error should be approximately zero. The following graph shows the results from the simulation exercise.





Concluding Remarks

For this project I implemented a forward stepwise regression algorithm to improve the accuracy on the estimation of the ATE through a matched difference-in-difference model. The key problem this algorithm tries to solve is approximating the data generating process of the treatment assignment rule, which is unknown in most observational studies. This is particularly useful when such assignment rule has complex functional forms that might induce some bias on the traditional coefficient estimate. In this sense, this is a data-driven algorithm to choose the model with highest prediction accuracy for the treatment assignment rule and use that to compute inverse probability weights that, if used to estimate the ATE, should reduce the underlying bias induced by the complex nature of the data generating process sorting units into treatment.

I test this algorithm using public finance data and in my simple toy example I show the potential benefits of nailing down the true data generating process of the treatment assignment rule. However, the current version of this algorithm is not bulletproof as it not always derives in a reduction of the bias. Therefore, there is room for improvement on the current implementation.

Github

You can find all the code in my github repository here. https://github.com/LuisNavarro07/stat610_finalproject.git