Gradient boosted weighting for linkage failure in US Census Bureau datasets

Matthew Cefalu, John Sullivan, Narayan Sastry, and Elizabeth Fussell

2021-03-01

Introduction

The `twangRDC` R package is a streamline version of the `twang` R package created specifically for use in the RDCs. It utilizes gradient boosted models to estimate weights to handle linkage failures and comparison group construction. Results using `twangRDC` will not necessarily be reproduced using `twang` due to important differences in implementation. The `twangRDC` package allows for much larger datasets and many more covariates, but users should note that with smaller datasets, the `twang` package is computationally more efficient. The `twangRDC` package uses `xgboost` for gradient boosting, which provides user with an optimized gradient boosting library that automatically utilizes parallel computation within the RDCs.

Methodology

The `twangRDC` package utilizes gradient boosted models to derive weights for nonequivalent groups. The algorithm alternates between adding iterations to the gradient boosted model (increasing its complexity) and evaluating balance. The algorithm automatically stops when additional iterations no longer improve balance. The package allows the user to generate weights for two different scenarios: linkage failures and the construction of a comparison group.

Linkage failures

Protected Identification Keys (PIKs) are linking keys used by the Census Bureau to match person records across datasets[1]. For example, using a PIK, a respondent in the 2000 Decennial Census can be linked to their response to the 2010 Decennial Census. PIKs are assigned through a probabilistic matching process that links survey or census records containing Personally Identifiable Information (PII) like name, address, date of birth and place of birth, to a Social Security reference file. Not all survey and census records can be linked to the reference file and as such not all records will be assigned a PIK. Linkage failures may occur if the PII used to match to the reference file is of insufficient quality or if an individual has no matching record in the Social Security reference file. PIKs can be assigned for the large majority of 2000 Decennial Census records. A Decennial Census record must have a PIK in order to link it to administrative records that contain other information, such as morality or annual residence.

The `twangRDC` pacakge can generate weights to account for linkage failures such that observations with PIKs are representative of the population. It does this by using standard nonresponse weights, where nonresponse is defined by linkage failure. This is achieved by weighting observations with PIKs by the inverse probability of having a PIK. As a note, although designed for linkage failure, the `twangRDC` can be used for general nonresponse or missing data weights.

The steps of the algorithm are described here assuming the user specifies strata in the model, but the steps are the same without strata by considering the data coming from a single stratum.

[1] Wagner, Deborah and Mary Layne. The Person Identification Validation System (PVS): Applying the Center Administrative Records Research and Applications' (CARRA) Record Linkage Software. US Census Bureau, Center Administrative Records Research and Applications Working Paper #2014-01. 2014.

1. Calculate the population-level means of the covariates within strata. If weights are specified by the user, the population-level means are calculated accounting for the weights.
2. Initialize a gradient boosted model that includes all covariates and the strata as a factor.
3. Add `iters.per.step` iterations to the gradient boosted model.
4. Calculate the means of the covariates within strata among observations with PIK using weights based on the current model.
5. Calculate the standardized differences of the covariates between weighted observations with PIK versus the population within strata.
6. Summarize the model fit by taking the maximum of the absolute standardized differences within strata.
7. Check if the average of the maximum absolute standardized differences across strata has been minimized. If a minimum has been achieved, stop. Otherwise, return to step #3.

Comparison group construction

If the goal is the construction of a comparison group, `twangRDC` can applied gradient boosted propensity score weights for the average treatment effect on the treated. Treatment observations receive a weight of 1, while comparison observations receive a weight of the odds of treatment based on the propensity score model. The steps of the algorithm are described here assuming the user specifies strata in the model, but the steps are the same without strata by considering the data coming from a single stratum.

1. Calculate the treatment group means of the covariates within strata. If weights are specified by the user, the treatment group means are calculated accounting for the weights.
2. Initialize a gradient boosted model that includes all covariates and the strata as a factor.
3. Add `iters.per.step` iterations to the gradient boosted model.
4. Calculate the means of the covariates within strata among comparison cases using propensity score weights based on the current model.
5. Calculate the standardized differences of the covariates between the treatment and propensity score weighted comparison group means.
6. Summarize the model fit by taking the maximum of the absolute standardized differences within strata.
7. Check if the average of maximum absolute standardized differences across strata has been minimized. If a minimum has been achieved, stop. Otherwise, return to step #3.

Description of arguments

Argument	Description
formula	A symbolic description of the model to be fit with an observed data indicator or a treatment indicator on the left side of the formula and the variables to be balanced on the right side. If `strata` is specified, the model automatically includes the strata as a factor variable on the right hand side of the equation.
linkage	An indicator of whether the weighting should be for linkage failure or comparison group construction. `linkage=TRUE` requests weighting to account for linkage failure, while `linkage=FALSE` requests weighting for comparison group construction.
strata	An optional factor variable identifying the strata. If specified, balance is optimized within strata.
data	The data.
params	`xgboost` parameters. Details below.
file	An optional filename to save intermediate results. The file is overwritten at each step of the algorithm.

Argument	Description
iters.per.step	An integer specifying the number of iterations to add to the model at each step of algorithm.
max_steps	An integer specifying the maximum number of steps to take while optimizing the gradent boosted model. The maximum number of iterations considered during optimization is given by `max_steps*iters.per.step`.
id.var	A variable that uniquely identifies observations.
min.width	An integer specifying the minimum number of iterations between the current number of model iterations and the optimal value.
save.model	A logical value indicating if the xgboost model be saved as part of the output object.
weights	An optional variable that identifies user defined weights to be incorporated into the optimization. E.g., sampling design weights.

Motivation

We will highlight two uses of the `twangRDC` R package. First, we will generate weights that ensure individuals with PIKs are representative of their geographic region. Second, we will highlight using the `twangRDC` R package to generate propensity score weights such that a group of southern metropolitan areas can be used as a comparison group for residents of Orleans Parish.

The Data

A simulated data file was created for use in this vignette. It contains simulated records for residents of Orleans Parish, Louisiana, and other metropolitan areas in the South census region. We built the file exclusively from public use data, but it mirrors the structure of restricted versions of the 2000 Decennial Census available through the FSRDC network[2]. Each simulated record includes basic individual demographic characteristics, basic household characteristics, and a set of neighborhood-level characteristics. Each record also includes two important indicators, one to simulate whether the individual record received a PIK and another to denote whether the individual lived in Orleans Parish.

The data was created by extracting all "short form" variables for households and individuals from the 5% Integrated Public Use Microdata Sample (IPUMS[3]) of the 2000 Decennial Census for the city of New Orleans (Orleans Parish) and for a selection of other southern metropolitan areas. We extracted contextual information from public use tract level tabulations of the 2000 Decennial Census (distributed by NHGIS[4]) and created select factor-based measures. We simulated assignment of households to census tracts and attach tract identifiers and characteristics.

We also simulated an indicator of PIK assignment (PIK=yes/no) to person records. Public use data do not include PIK assignment, so we estimated a predicted probability of receiving a PIK by using estimated regression parameters from Bond et al. 2013[5]. We added a random error to the predicted probability so that the PIK assignment status **is not deterministic** and convert the predicted probability into a dichotomous PIK=yes/no variable.

[2]https://www.census.gov/fsrdc

[3]Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas and Matthew Sobek. IPUMS USA: Version 10.0 [dataset]. Minneapolis, MN: IPUMS, 2020. https://doi.org/10.18128/D010.V10.0

[4]Steven Manson, Jonathan Schroeder, David Van Riper, and Steven Ruggles. IPUMS National Historical Geographic Information System: Version 14.0 [Database]. Minneapolis, MN: IPUMS. 2019. http://doi.org/10.18128/D050.V14.0

[5]Bond, Brittany, J. David Brown, Adela Luque and Amy O'Hara. The nature of the bias when studying only linkable person records: Evidence from the ACS. US Census Bureau, Center Administrative Records Research and Applications Working Paper #2014-08, 2014.

Lastly, we pooled Orleans Parish records with those from other southern metropolitan areas, create an indicator for Orleans Parish residence and, for the purposes of this vignette, sampled the data to shrink the size of the dataset. The simulated file contains individual records for 4,606 residents of Orleans Parish, LA and 9,519 individual records for residents from select southern metropolitan areas.

Table 1 provides a description of the data element included in the simulated data file.

Table 2: Description of data elements

Data element	Description	Labels and Codings
metarea	A categorical variable for metropolitan area.	Atlanta, GA (52) Memphis, TN/AR/MS (492) New Orleans, LA (556) Washington, DC/MD/VA (884)
c00_age12	A categorical variable for age in years at the 2000 Decennial Census.	0 to 2 years old (1)3 to 5 years old (2)6 to 9 years old (3)10 to 14 years old (4) 15 to 18 years old (5) 19 to 24 years old (6) 25 to 34 years old (7) 35 to 44 years old (8) 45 to 54 years old (9) 55 to 64 years old (10) 65 to 74 years old (11) 75 and older (12)
c00_sex	A binary indicator of sex as reported on the 2000 Decennial Census.	Male (0) Female (1)
c00_race	A categorical variable for race as reported on the 2000 Decennial Census.	White (1)African American (2)American Indian or Alaskan Native (3)Asian (4)Other Asian or Pacific Islander (5)Some other race (6)
c00_nphu	The number of persons in housing unit as reported on the 2000 Decennial Census.	1 to 16
hhid	Household identifier.	
tract_id_str	Census tract identifier.	
concdis	Tract level factor measure of concentrated disadvantage/	
res_stab	Tract level factor measure of residential stability.	
imm_conc	Tract level factor measure of immigrant concentration.	
sim_pik	Simulated binary indicator of PIK assignment.	No PIK assigned (0)PIK assigned (1)
nola_rec	Binary indicator for record from Orleans Parish.	Not Orleans Parish Record (0)Orleans Parish Record (1)
id	An ID variable that uniquely identifies individuals in the dataset.	

Weighting to account for linkage failure

As previously mentioned, we will generate weights that ensure individuals with PIKs are representative of their geographic region. To keep the computational time of this vignette down, we focus only on a subset of

Orleans parish. First, we load the `twangRDC` package and our simulated dataset.

```
  library(twangRDC)
#> Loading required package: xgboost
#> Loading required package: data.table
#> Loading required package: ggplot2
  data(nola_south)
```

Next, it is important that the variables of the dataset are coded as intended. In this case, we convert several of variables to factors.

```
  # factors need to be coded as such
  nola_south$metarea = as.factor(nola_south$metarea)
  nola_south$c00_age12 = as.factor(nola_south$c00_age12)
  nola_south$c00_race = as.factor(nola_south$c00_race)
```

In a final data preparation step, we limit the dataset to Orleans parish and select only 10 of the Census tracts.

```
  # only consider Orleans parish
  nola_only = subset(nola_south , metarea==556)

  # keep only 10 tracts for computational speed
  to.keep = unique(nola_only$tract_id_str)[1:10]
  nola_only = nola_only[ nola_only$tract_id_str %in% to.keep, ]
```

In this case, we wish to generate weights to ensure that individuals with PIKs are representative of their entire Census tract. That is, for each Census tract, we want to generate weights such that the observations with PIKs within the Census tract are representative of the tract's population. This is achieved by specifying `linkage=TRUE`, which tells the function that we wish to generate nonresponse weights based on linkage failure, and `strata="tract_id_str"`, which tells the function that we wish to generate weights that are representative within strata defined by Census tracts. The formula `sim_pik ~ c00_age12 + c00_race + c00_sex` identifies the observations with PIK on the left hand side and the characteristics that we want to consider when estimating the weights the right hand side.

```
  # set the model parameters
  params = list(eta = .01 , max_depth = 10 , subsample = 1 ,
                max_delta_step=0 , gamma=0 , lambda=0 , alpha=0,
                min_child_weight=5 , objective = "binary:logistic")

  # fit the xgboost model
  res.pik = ps.xgb(sim_pik ~ c00_age12 + c00_race + c00_sex ,
                strata="tract_id_str",
                data=nola_only,
                params=params,
                max.steps=10,
                iters.per.step=500,
                min.iter=500,
                id.var="id",
                linkage = TRUE)
```

The parameters of the underlying `xgboost` model are specified in `params`. These were described in detail in a previous section. The `id.var="id"` provides a unique identifier for observations such that the generated weights can easily be merged back in with the original data. The other options specified in the `ps.xgb` function control how frequently the algorithm checks for convergence and how many iterations should be considered before stopping. First, `iters.per.step=50` tells the algorithm to only consider every 50-th iteration when evaluating convergence. Larger values improve computational time by reducing the number of balance

evaluations, while smaller values may achieve slightly better balance. Next, `min.iter=50` tells the algorithm that at least 50 iterations must be used before stopping for convergence. Larger values ensure that more complex models are evaluated before determining the optimal set of weights. Finally, `max.steps=2` indicates that the algorithm should only evaluate the balance of the weights twice before stopping. The maximum number of iterations of the `xgboost` model is given by `max.steps*iters.per.setp`, which in this case is 100. In general, this value should be large to ensure that the optimum set of weights is achieved. The default value is `max.steps=Inf`, which will continue adding iterations to the model until the convergence criteria is met. Due to computational concerns, we recommend testing your code with values of `iters.per.step` and `max.steps` such that the total number of iterations is small (1000 to 10000). Once you have determined the model is working as intended, set `max.steps` to `Inf`.

Now that the weights have been estimated, we can evaluate their quality. First, we need to ensure that a sufficient number of iterations have been used such that the balance criteria is minimized.

```
plot(res.pik)
```

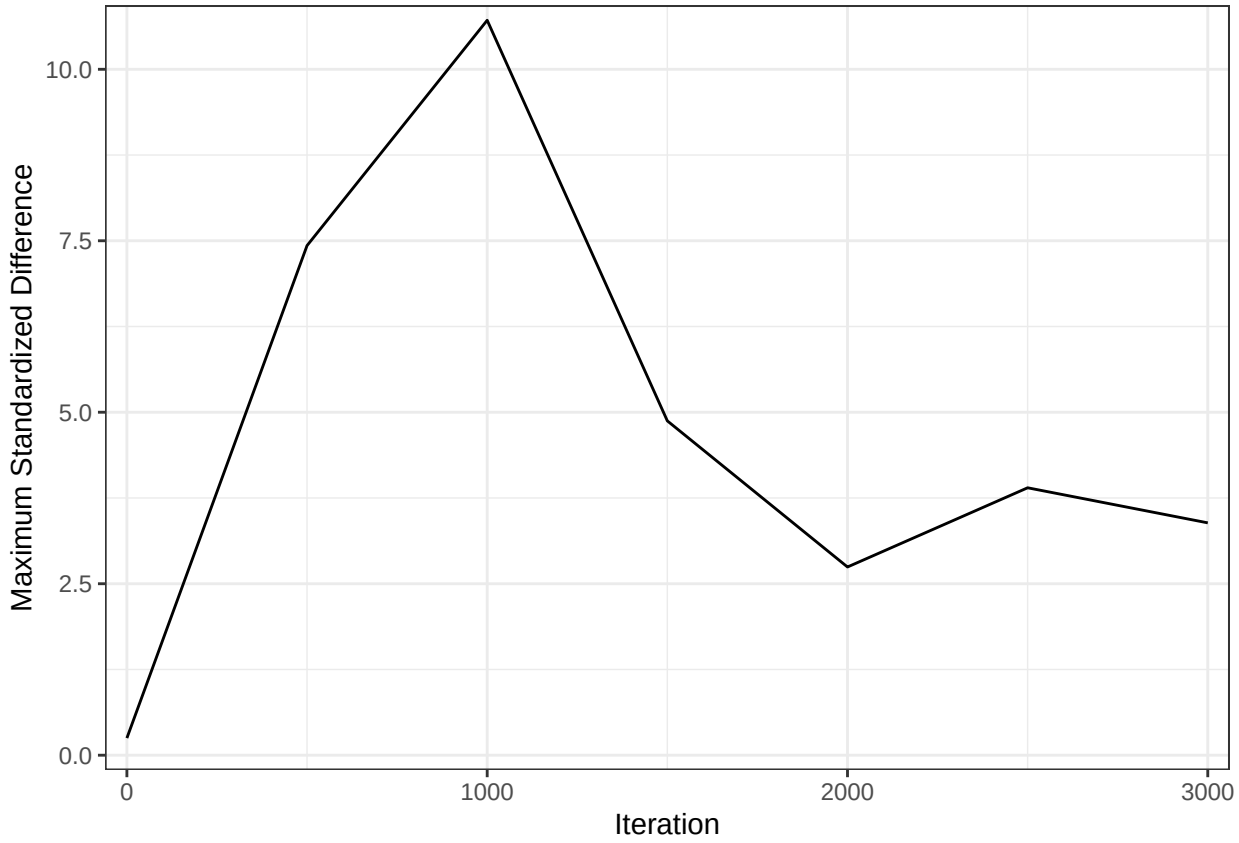

Figure 1: Convergence of gradient boosted model for linkage failure.

The `plot` function provides a plot of the balance criteria, which in this case is the average of the strata-specific maximum standardized difference of the covariates, versus the number iterations. As shown in this figure, we are verifying that the algorithm has run for a sufficient number of iterations such that a clear minimum has been achieved.

Balance tables

Now that it has been determined that we have achieved convergence, we can access the quality of the weights using balance tables. First, the `bal.table` function will produce the population mean for each covariate, as well as the unweighted and the weighted mean among those with PIK. It also provides the standardized

difference for each covariate before and after weighting. The goal is for the standardized differences after weighting to all be close to zero.

```
bal.table(res.pik)
```

	Population Mean	Unadjusted Mean	Adjusted Mean	Unadjusted Standardized Difference	Adjusted Standardized Difference
c00_age12:1	0.0356	0.0667	0.2751	-0.1673	-1.2916
c00_age12:2	0.0955	0.0667	-0.1734	0.0980	0.9151
c00_age12:3	0.4091	0.0667	0.2112	0.6964	0.4025
c00_age12:4	0.0456	0.0000	0.0000	0.2187	0.2187
c00_age12:5	0.0365	0.0667	0.0553	-0.1607	-0.1000
c00_age12:6	0.0508	0.0000	0.0000	0.2314	0.2314
c00_age12:7	0.0797	0.2000	0.2048	-0.4444	-0.4622
c00_age12:8	0.1029	0.0000	0.0000	0.3386	0.3386
c00_age12:9	0.0566	0.2667	0.2168	-0.9088	-0.6933
c00_age12:10	0.0348	0.1333	0.1015	-0.5381	-0.3642
c00_age12:11	0.0367	0.1333	0.1087	-0.5139	-0.3831
c00_age12:12	0.0162	0.0000	0.0000	0.1285	0.1285
c00_race:1	0.0813	0.4000	0.4810	-1.1656	-1.4621
c00_race:2	0.8229	0.6000	0.5190	0.5839	0.7962
c00_race:3	0.0060	0.0000	0.0000	0.0777	0.0777
c00_race:5	0.0166	0.0000	0.0000	0.1300	0.1300
c00_race:6	0.0731	0.0000	0.0000	0.2809	0.2809
c00_sex	0.2899	0.6667	0.8101	-0.8295	-1.1452

Next, balance can be assessed within Census tract since our goal for this model was to generate weights such that observations with PIKs are representative of their Census tract. The `type='strata'` option tells `bal.table` to provide the maximum absolute standardized difference by the strata variable, in this case the Census tract. We additionally specify`include.var=TRUE`, which identifies the covariate that the maximum absolute standardized difference corresponds to, `decreasing = T`, which orders the strata in decreasing order of their weighted standardized differences, and `n=3`, which only prints the top three strata.

```
bal.table(res.pik , type='strata' , include.var=TRUE , n=3 , decreasing = T)
```

	Stratum	Unadjusted Maximum Standardized Difference	Adjusted Maximum Standardized Difference	Variable
9	22071010100	0.1731	6.4337	c00_age12:3
1	22071001722	0.2900	4.5189	c00_age12:3
8	22071004900	0.3273	3.8982	c00_age12:3

This table allows us to assess which strata had the worst balance after weighting, and among those strata, which covariates were problematic.

Extracting weights

The `get.weights` function extracts the weights at the optimal iteration. The resulting data contains the weights and the ID variable specified in `id.var`. The weights can then be merged back in with the original data using the `id.var`. Note that base R merge function is slow compared to modern alternatives. If your data is large, consider `data.table` or `dplyr`.

```
# extract weights
w = get.weights(res.pik)

# merge weights into data -- use data.table because its merge is faster than base R
dta=merge(dta , w , by='id' , all=TRUE)
```

Comparison group construction

Our second example use of `twangRDC` will generate a comparison group for Orleans Parish consisting of residents of other southern metropolitan areas. The steps of the process remain the same as the PIK weighting example, but with minor adjustments to the interpretation and specification of the model. First, `linkage = FALSE` specifies that we are no longer interested in weights to account for linkage failure, but instead, we wish to generate a comparison group using propensity score weights. Note that `ps.xgb` only estimates the average treatment effect on the treated, with the treatment group identified by records with a value of 1 in the left hand side of the formula. Our formula now includes three tract-level summaries that we wish to balance in addition to the three person-level characteristics. Finally, we no longer specify a stratification variable, as we are not attempting to create a comparison group for each tract of Orleans parish, but instead, a single comparison group that is representative of Orleans parish. If we were interested in using the linkage failure weights from our previous example, they could be specified in the `weights` option, but we would need to derive such weights for all observations in the dataset.

```
# set the model parameters
params = list(eta = .1 , max_depth = 10 , subsample = 1 ,
              max_delta_step=0 , gamma=0 , lambda=0 , alpha=0,
              min_child_weight=5 , objective = "binary:logistic")

# fit the xgboost model
res.ps = ps.xgb(nola_rec ~ c00_age12 + c00_race + c00_sex + concdis + res_stab + imm_conc ,
             data=nola_south,
             params=params,
             max.steps=10,
             iters.per.step=500,
             min.iter=500,
             id.var="id",
             linkage = FALSE)
```

Following the model fit, the same steps covered in the linkage failure example should be used here as well. First, evaluate the convergence of the model using the `plot` function. We are verifying that the algorithm has run for a sufficient number of iterations such that a clear minimum has been achieved.

```
plot(res.ps)
```

Next, we evaluate the performance of the algorithm by looking at balance tables. The `bal.table` function will produce the treatment group mean for each covariate, as well as the mean before and after propensity score weighting among comparison cases. It also provides the standardized differences for each covariate. The goal is for the standardized differences after weighting to all be close to zero.

```
bal.table(res.ps)
```

	Treatment Group Mean	Unadjusted Comparison Group Mean	Adjusted Comparison Group Mean	Unadjusted Standardized Difference	Adjusted Standardized Difference
c00_age12:1	0.0393	0.0457	0.0457	-0.0329	-0.0332
c00_age12:2	0.0413	0.0481	0.0480	-0.0345	-0.0341
c00_age12:3	0.0599	0.0635	0.0634	-0.0149	-0.0147

	Treatment Group Mean	Unadjusted Comparison Group Mean	Adjusted Comparison Group Mean	Unadjusted Standardized Difference	Adjusted Standardized Difference
c00_age12:4	0.0808	0.0766	0.0765	0.0153	0.0157
c00_age12:5	0.0660	0.0552	0.0551	0.0437	0.0439
c00_age12:6	0.0942	0.0708	0.0706	0.0802	0.0808
c00_age12:7	0.1366	0.1622	0.1624	-0.0747	-0.0752
c00_age12:8	0.1468	0.1743	0.1749	-0.0778	-0.0795
c00_age12:9	0.1431	0.1446	0.1446	-0.0042	-0.0045
c00_age12:10	0.0797	0.0801	0.0801	-0.0014	-0.0016
c00_age12:11	0.0601	0.0430	0.0427	0.0722	0.0732
c00_age12:12	0.0523	0.0361	0.0358	0.0727	0.0741
c00_race:1	0.2794	0.5902	0.5929	-0.6926	-0.6987
c00_race:2	0.6709	0.2977	0.2947	0.7941	0.8006
c00_race:3	0.0022	0.0034	0.0034	-0.0256	-0.0257
c00_race:4	0.0026	0.0109	0.0110	-0.1632	-0.1645
c00_race:5	0.0200	0.0392	0.0393	-0.1373	-0.1382
c00_race:6	0.0250	0.0586	0.0587	-0.2157	-0.2164
c00_sex	0.5371	0.5219	0.5216	0.0305	0.0311
concdis	0.8648	-0.4007	-0.4162	0.8451	0.8553
res_stab	0.8726	0.4852	0.4813	0.4261	0.4304
imm_conc	-0.3736	0.4585	0.4724	-3.0697	-3.1209

Since no strata were specified in the model, the `type='strata'` option used in the linkage failure example is no longer useful.

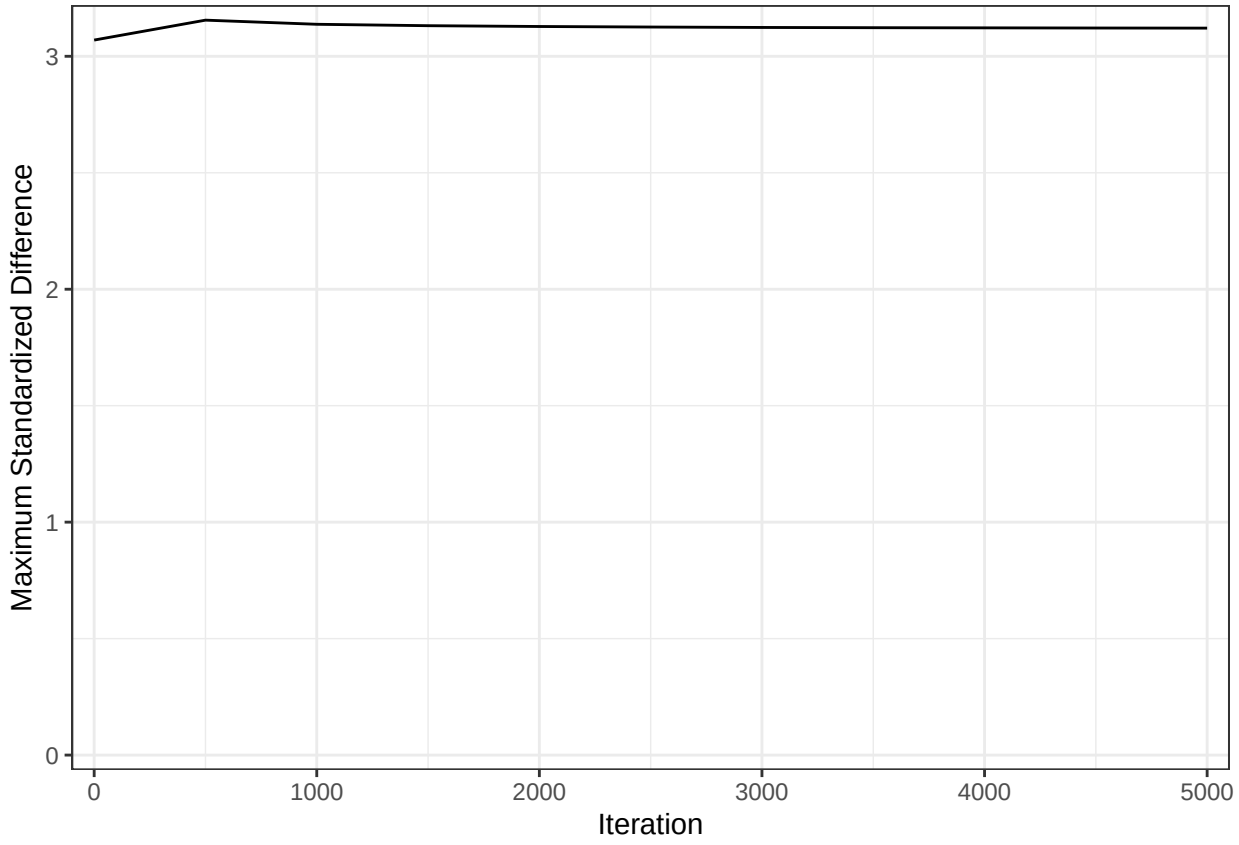

Figure 2: Convergence of gradient boosted model for comparison group construction.