Toolkit for Weighting and Analysis of Nonequivalent Groups: A tutorial for the `twang` package

Greg Ridgeway, Dan McCaffrey, Andrew Morral, Lane Burgette and Beth Ann Griffin*
RAND

April 3, 2020

1 Introduction

The Toolkit for Weighting and Analysis of Nonequivalent Groups, `twang`, contains a set of functions and procedures to support causal modeling of observational data through the estimation and evaluation of propensity scores and associated weights. This package was developed in 2004. After extensive use, it received major updates in 2012 and 2020. In the most recent version (2.0), the `twang` package has undergone extensive revisions to improve computational efficiency. Code written for prior versions of `twang` should still run without modification. However, in certain situations, results from the updated version of `twang` will no longer replicate prior implementations. This tutorial provides an introduction to `twang` and demonstrates its use through illustrative examples.

The foundation to the methods supported by `twang` is the propensity score. The propensity score is the probability that a particular case would be assigned or exposed to a treatment condition. Rosenbaum & Rubin (1983) showed that knowing the propensity score is sufficient to separate the effect of a treatment on an outcome from observed confounding factors that influence both treatment assignment and outcomes, provided the necessary conditions hold. The propensity score has the balancing property that given the propensity score the distribution of features for the treatment cases is the same as that for the control cases. While the treatment selection probabilities are generally not known, good estimates of them can be effective at diminishing or eliminating confounds between pretreatment group differences and treatment outcomes in the estimation of treatment effects.

There are now numerous propensity scoring methods in the literature. They differ in how they estimate the propensity score (e.g. logistic regression, CART), the target estimand (e.g. treatment effect on the treated, population treatment effect), and how they utilize the resulting estimated propensity scores (e.g. stratification, matching, weighting, doubly robust estimators). We originally developed the `twang` package with a particular process in mind, namely, generalized boosted regression to estimate the propensity scores and weighting of the comparison cases to estimate the average treatment effect on the treated (ATT). However, we have updated the package to also meaningfully handle the case where interest lies in using the population weights (e.g., weighting of comparison and treatment cases to estimate the population average treatment effect, ATE).

The main workhorse of `twang` is the `ps()` function which implements generalized boosted regression modeling to estimate the propensity scores. However, the framework of the package is flexible enough to allow the user to use propensity score estimates from other methods and to assess the usefulness of those estimates for ensuring equivalence (or "balance") in the pretreatment covariate distributions of treatment and

*The twang package and this tutorial were developed under NIDA grants R01 DA017507 and R01 DA015697-03

control groups using tools from the package. The same set of functions is also useful for other tasks such as non-response weighting, as discussed in Section 4.

The `twang` package aims to (i) compute from the data estimates of the propensity scores which yield accurate causal effect estimates, (ii) check the quality of the resulting propensity score weights by assessing whether or not they have the balancing properties that we expect in theory, and (iii) use them in computing treatment effect estimates. Users who are more comfortable with SAS than R are encouraged to visit `www.rand.org/statistics/twang/sas-tutorial.html` for information SAS macros that implement these methods.

2 An ATT example to start

2.1 Load the data

Start by installing the package by typing `install.packages("twang")`. `twang` relies on other R packages, especially `gbm`, `survey`, and `lattice`. You may have to run `install.packages()` for these as well if they are not already installed. You will only need to do this step once. In the future running `update.packages()` regularly will ensure that you have the latest versions of the packages, including bug fixes and new features.

To start using `twang`, first load the package. You will have to do this step once for each R session that you run. We also set the seed of R's pseudo random number generator so that the results are exactly replicable. (There is a stochastic element to the fitting of the propensity score models.)

```
> library(twang)
> set.seed(1)
```

To demonstrate the package we utilize data from Lalonde's National Supported Work Demonstration analysis (Lalonde 1986, Dehejia & Wahba 1999, `http://users.nber.org/~rdehejia/nswdata2.html`). This dataset is provided with the `twang` package.

```
> data(lalonde)
```

R can read data from many other sources. The manual "R Data Import/Export," available at `http://cran.r-project.org/doc/manuals/R-data.pdf`, describes that process in detail.

2.2 Fit the ps() function

For the `lalonde` dataset, the variable `treat` is the 0/1 treatment indicator, 1 indicates "treatment" by being part of the National Supported Work Demonstration and 0 indicates "comparison" cases drawn from the Current Population Survey. In order to estimate a treatment effect for this demonstration program that is unbiased by pretreatment group differences on other observed covariates, we include these covariates in a propensity score model of treatment assignment: age, education, black, Hispanic, having no degree, married, earnings in 1974 (pretreatment), and earnings in 1978 (pretreatment). Note that we specify no outcome variables at this time. The `ps()` function is the primary method in `twang` for estimating propensity scores. This step is computationally intensive and can take a few minutes.

```
> ps.lalonde <- ps(treat ~ age + educ + black + hispan + nodegree +
+                          married + re74 + re78,
+                  data = lalonde,
+                  n.trees=5000,
+                  interaction.depth=2,
+                  shrinkage=0.01,
+                  perm.test.iters=0,
```

```
+                   estimand = "ATT",
+                   stop.method=c("es.mean","ks.max"),
+                   bag.fraction = 1,
+                   n.minobsinnode = 10,
+                   n.keep = 1,
+                   n.grid = 25,
+                   ks.exact = NULL,
+                   verbose=FALSE)
```

The arguments to `ps()` require some discussion.

2.2.1 Description and usage of the arguments to `ps()` function

`formula` the first argument, specifies a formula indicating that `treat` is the 0/1 treatment indicator and that the propensity score model should predict `treat` from the eight covariates listed there separated by "+". As when fitting linear models via the `lm()` function, "+" does *not* mean that these variables are being summed; in contrast to the `lm()` function, neither does it mean that the model is linear. This is just R's notation for including predictor variables in the model. There is no need to specify interaction terms in the formula. There is also no need — and it can be counterproductive — to create indicator, or "dummy coded," variables to represent categorical covariates, provided the categorical variables are stored as a `factor` or as `ordered` (see `help(factor)` for more details).

`data` indicates the dataset to be used.

`n.trees` indicates the number of gbm iterations passed on to `gbm`. Default value is 10000.

`interaction.depth` is a positive integer denoting the tree depth used in gradient boosting. Default value is 3.

`shrinkage` is a numeric value between 0 and 1 denoting the learning rate. Default value is 0.01.

`n.trees`, `interaction.depth`, and `shrinkage` are the boosting options, parameters for the `gbm` model that `ps()` computes and stores. The resulting `gbm` object describes a family of candidate propensity score models indexed by the number of GBM iterations from one to `n.trees`. The argument `n.trees` is the maximum number of iterations that `gbm` will run. `ps()` will issue a warning if the estimated optimal number of iterations is too close to the bound selected in this argument because it indicates that balance may improve if more complex models (i.e., those with more trees) are considered. The user should increase `n.trees` or decrease `shrinkage` if this warning appears. The argument `interaction.depth` controls the level of interactions allowed in the GBM. We typically use the default in our analyses. The GBM estimation algorithm uses shrinkage to enhance the smoothness of resulting model. The `shrinkage` argument controls the amount of shrinkage. Small values such as 0.005 or 0.001 yield smooth fits but require greater values of `n.trees` to achieve adequate fits. Computational time increases inversely with `shrinkage` argument. Additional details on `n.trees`, `interaction.depth`, and `shrinkage` can be found McCaffrey, Ridgeway, and Morral (2004).

`perm.test.iters` specifies whether p-values for KS statistics should be calculated using Monte Carlo methods, which is slow but can be accurate, or estimated using an analytic approximation that is fast, but produces poor estimates in the presence of many ties. If `perm.test.iters=0` is called, then analytic approximations are used. If `perm.test.iters = 500` is called, then 500 Monte Carlo trials are run to establish the reference distribution of KS statistics for each covariate. Higher numbers of trials will produce more precise p-values. Specifying `perm.test.iters` greater than zero can greatly slow down the `twang` computations. We tend to rely on the approximations (`perm.test.iters`=0) when using `twang` in practice.

`estimand` is used to indicate whether the analyst is interested in estimating the average treatment effect (ATE) or the average treatment effect on the treated (ATT), as we do above. ATE addresses the question of how outcomes would differ if everyone in the sample were given the treatment versus everyone being given the control (Wooldridge, 2002). ATT, on the other hand, estimates the analogous quantity averaging only over the subjects who were actually treated. The `estimand` argument was added to the 2012 revision of the package which integrated ATE weighting into the package and the `ps` function estimate of the propensity score. Defaualt value is `"ATE"`.

`stop.method` specifies a set (or sets) of rules and measures for assessing the balance, or equivalence, established on the pretreatment covariates of the weighted treatment and control groups. The `ps` function selects the optimal number of GBM iterations to minimize the differences between the treatment and control groups as measured by the rules of the given `stop.method` object. The package includes four built-in `stop.method` objects. They are `es.mean`, `es.max`, `ks.mean`, and `ks.max`. The four stopping rules are defined by two components: a balance metric for covariates and rule for summarizing across covariates.

The balance metric summarizes the difference between two univariate distributions of a single pretreatment variable (e.g. age). The default stopping rules in `twang` use two balance metrics: absolute standardized bias (also referred to as the absolute standardized mean difference or the *E*ffect *S*ize) and the Kolmogorov-Smirnov (KS) statistic. The stopping rule use two different rules for summarizing across covariates: the mean of the covariate balance metrics ("mean") or the maximum of the balance metrics ("max"). The first piece of the stopping rule name identifies the balance metric (ES or KS) and the second piece specifies the method for summarizing across balance metrics. For instance, `es.mean` uses the effect size or the absolute standardized bias and summarizes across variables with the mean and the `ks.max` uses the KS statistics to assess balances and summarizes using the maximum across variables and the other two stopping rules use the remaining two combinations of balance metrics and summary statistics. The variable distributions used in the balance metrics depend on whether we are interested in estimating the ATT or ATE, and correct specification of these distributions is set automatically by the specification of the `estimand` in the `ps()` function.

Other additional boosting parameters that can be passed are as follows:

`bag.fraction` is the subsampling fraction. A numeric value between 0 and 1 denoting the fraction of the observations randomly selected in each iteration of the gradient boosting algorithm to propose the next tree. Thus `bag.fraction=0.5` denotes that half of the training sample will be randomly selected at each iteration. Default value is 1.0.

`n.minobsinnode` specifies an integer denoting the minimum number of observations in the terminal nodes of the trees used in the gradient boosting. Default value is 10. This is a new addition in the current version where user can pass the minimum number of observations in the terminal nodes of the trees, to the underlying gradient boosting algorithm. Previously, this parameter was not an available argument for the user and was always fixed at 10.

```
> Ex_M1 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             n.minobsinnode = 100,
+             verbose = FALSE)
> Ex_M1$gbm.obj$n.minobsinnode

[1] 100
```

`bag.fraction` and `n.minobsinnode` along with `n.trees`, `interaction.depth`, and `shrinkage` are parameters needed to fine-tune the `gbm` model.

Two of the new options included in Version 2.0 of `twang` are `n.keep` and `n.grid`. These options control how `twang` determines the iterations over which the algorithm optimizes balance. Before describing the options, we first describe how `twang` determines the optimal iteration. The algorithm first creates a grid of points, which is a 25 point grid by default, and assesses the balance at each iteration of the grid. The algorithm finds the optimal iteration based on this grid, and then does a finer search for the optimum between this grid point and its neighbors. For example, say we find that grid point 15 achieves the optimal balance, then the algorithm focuses it search for the optimal iteration between grid points 14 and 16.

`n.keep` option is a numeric variable indicating that the algorithm should only consider every `n.keep`-th iteration of the propensity score model and optimize balance over this set instead of all iterations. The default value is 1, which is to optimize over all iterations.

`n.grid` option is a numeric variable that sets the grid size for an initial search of the region most likely to minimize the `stop.method`. Default grid size is 25. When `n.keep=1`, a value of `n.grid=50` uses a 50 point grid from `1:n.trees`.

```
> Ex_M2 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             estimand = "ATT",
+             n.grid = 50,
+             verbose = FALSE)
> # how many points are in the grid?
> length(Ex_M2$iters)

[1] 50
```

The options `n.grid` and `n.keep` can be combined. The option `n.grid` corresponds to a grid of points on the kept iterations as defined by `n.keep`. For example, `n.keep=10` with `n.trees=5000` will keep the iterations `(10,20,...,5000)`. The option `n.grid=10` then splits this vector into 10 points. Thus, `n.grid*n.keep` must be less than or equal to `n.trees`.

```
> Ex_M3 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             estimand = "ATT",
+             n.trees = 5000,
+             n.keep = 10,
+             n.grid = 10,
+             verbose = FALSE)
> summary(Ex_M3)

            n.treat n.ctrl ess.treat  ess.ctrl    max.es    mean.es    max.ks
unw             185    429       185 429.00000 1.7567745 0.54253254 0.6404460
ks.mean.ATT     185    429       185  23.12362 0.1504030 0.07587719 0.1439794
es.mean.ATT     185    429       185  16.02817 0.1258404 0.07370936 0.1592989
            max.ks.p    mean.ks iter
unw               NA 0.24661532   NA
ks.mean.ATT       NA 0.05834134 1190
es.mean.ATT       NA 0.06230717 1640
```

In case `n.grid*n.keep` is greater than `n.trees` the user should recieve an error message.

```
> Ex_M4 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             estimand = "ATT",
+             n.trees = 5000,
+             n.keep = 100,
+             n.grid = 100,
+             verbose = FALSE)

Error in ps.fast(formula = formula, data = data, n.trees = nrounds, interaction.depth = max_depth,  :
  n.tress must be at least n.grid times n.keep
```

`version` option is another new addition to `twang` 2.0 which helps specifying the version of the package, `"gbm"`, `"xgboost"`, or `"legacy"`. The default value is `version="gbm"` which uses gradient boosting from the `gbm` package and it provides a faster implementation of key components of the `twang` package to improve speed when calculating and optimizing the balance statistics.

`version="xgboost"` uses gradient boosting from the `xgboost` package. This provides users with access to a cutting-edge implementation of gradient boosting, while also providing substantial speed improvements in larger datasets. The behavior of `xgboost` can be controlled using the the boosting parameters of `gbm` discussed earlier, but can also be controlled directly using the options of `xgboost`. See https://xgboost.readthedocs.io/en/latest/parameter.html for a description of the different options.

```
> ## The following model specifications are equivalent
> #  Model 1 using "gbm" options
> Ex_M5 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             estimand = "ATT",
+             n.trees = 5000,
+             shrinkage = 0.05,
+             interaction.depth = 3,
+             n.minobsinnode = 15,
+             version = "xgboost",
+             verbose = FALSE)
> #  Model 2 using "xgboost" options
> Ex_M6 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             estimand = "ATT",
+             nrounds = 5000,
+             eta = 0.05,
+             max_depth = 3,
+             min_child_weight = 15,
+             version = "xgboost",
+             verbose = FALSE)
> # verify this is true by checking balance.
> all.equal(bal.table(Ex_M5),bal.table(Ex_M6))

[1] TRUE
```

Alternatively, the list of parameters passed to `xgboost` can be specified with `params`.

```
> Ex_M7 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             estimand = "ATT",
+             nrounds = 5000,
+             params = list(eta = 0.05, max_depth = 3, min_child_weight = 15),
+             version = "xgboost",
+             verbose = FALSE)
> all.equal(bal.table(Ex_M6),bal.table(Ex_M7))

[1] TRUE
```

Details on assessing "balance" via `bal.table` is discussed in later sections of this turtorial.

`version="legacy"` uses the prior implementation of the `ps` function. Version 2.0 of `twang` defaults to using updated balance and optimization routines. Once installed, the user does not need to modify any existing code to use the update functions that improves computational efficiency. If the user wishes to use the prior implementation of twang, they can request this by specifying the `version="legacy"` option.

```
> Ex_M8 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             version = "legacy",
+             verbose = FALSE)
```

However, the new options included in Version 2.0 of `twang` are not allowed when the user requests the `legacy` version of the code. If specified together, the user should receive an error message. For example,

```
> Ex_M9 <- ps(treat ~ age + educ + black + hispan + nodegree +
+                     married + re74 + re78,
+             data = lalonde,
+             version = "legacy",
+             ks.exact = TRUE,
+             verbose = FALSE)

Error in ps(treat ~ age + educ + black + hispan + nodegree + married +  :
  Option ks.exact is not allowed with version='legacy'
```

`ks.exact` is a new option in Version 2.0 to improve the speed when calculating the p-value for the weighted two-sample Kolmogorov–Smirnov (KS) statistic. The default behavior when calculating the weighted two-sample KS p-value differs from previous implementations of `twang`. Specifically, if `ks.exact=NULL` and the product of the effective sample sizes is less than 10,000, then an approximation based on the exact distribution of the unweighted KS statistic is used. This approximation via the exact distribution can also be requested directly by `ks.exact=TRUE`. Otherwise, an approximation based on the asymptotic distribution of the unweighted KS statistic is used.

`sampw` are optional sampling weights.

`multinom` is to `TRUE` only when called from `mnps` function.

`print.level` is the amount of detail to print to the screen. Default value is 2.

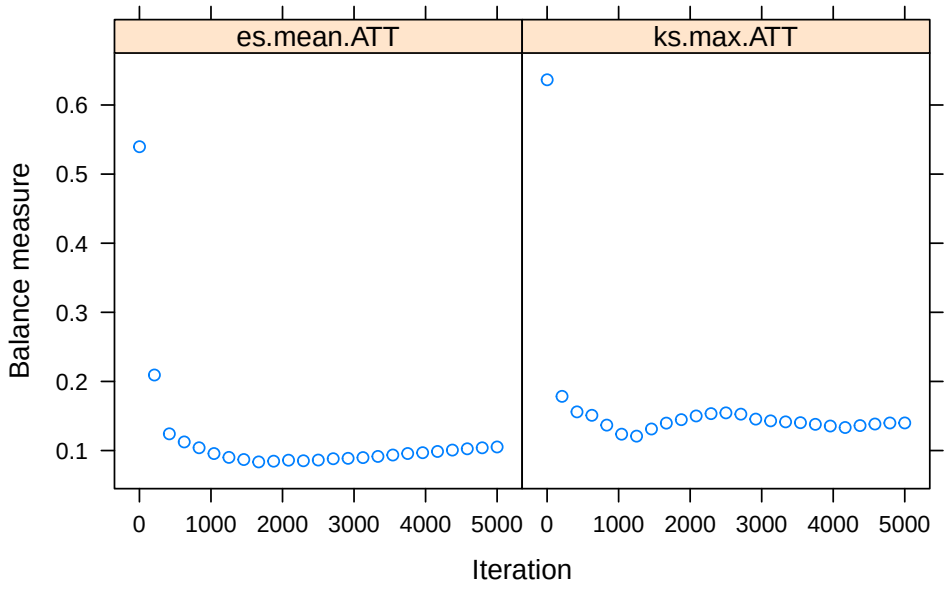
`verbose` is the argument that determines the printing of information on the console. If `TRUE`, lots of information will be printed to monitor the progress of the fitting. Default value is `TRUE`. Thus printing to the console will be carried out by default unless explicitly mentioned in the function call earlier as `verbose=FALSE`.

2.2.2 Diagnostic checks with plot() function

Having fit the `ps` object, the analyst should perform several diagnostic checks before estimating the causal effect in question. The first of these diagnostic checks makes sure that the specified value of `n.trees` allowed GBM to explore sufficiently complicated models. We can do this quickly with the `plot()` function.[1]

As a default, the `plot()` function applied to a `ps` object gives the balance measures as a function of the number of iterations in the GBM algorithm, with higher iterations corresponding to more complicated fitted models. In the example below, 1672 iterations minimized the average effect size difference and 1151 iterations minimized the largest of the eight Kolmogorov-Smirnov (KS) statistics computed for the covariates. If it appears that additional iterations would be likely to result in lower values of the balance statistic, `n.trees` should be increased. However, after a point, additional complexity typically makes the balance worse, as in the example below. This figure also gives information on how compatible two or more stopping rules are: if the minima for multiple stopping rules under consideration are near one another, the results should not be sensitive to which stopping rule one uses for the final analysis. See Section 5.3 for a discussion of these and other balance measures.

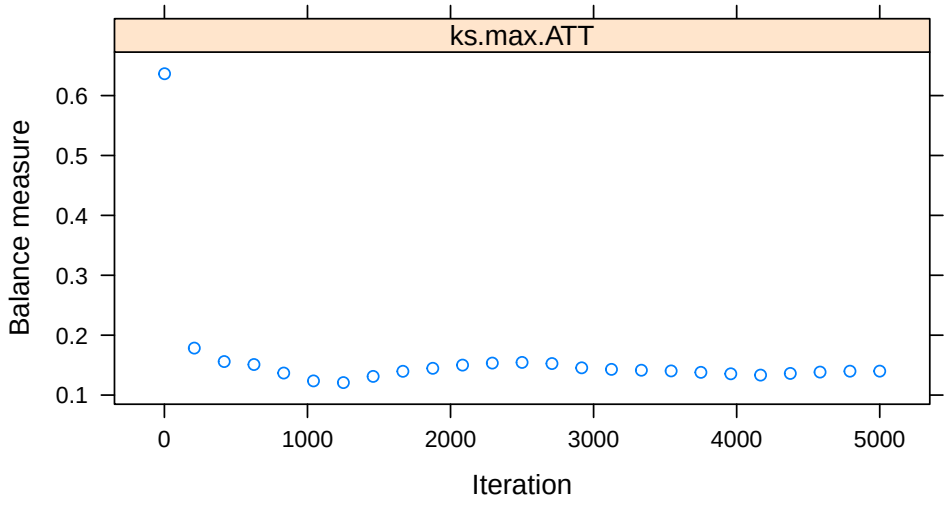
```
> plot(ps.lalonde)
```



If we wish to focus on only one stopping rule, the plotting commands also take a `subset` argument.

[1] In versions 1.0.x of the `twang` package, the `ps` function itself included some plotting functions. This is no longer the case (and the function no longer includes a `plots` argument); these functions have been moved to the generic `plot()` function.

```
>     plot(ps.lalonde, subset = 2)
```



The `gbm` package has various tools for exploring the relationship between the covariates and the treatment assignment indicator if these are of interest. `summary()` computes the relative influence of each variable for estimating the probability of treatment assignment. The gbm estimates depend on the number of iterations at which the gbm model is evaluated, which is specified by the `n.trees` argument in the `summary` method for gbm. In this example, we choose the number of iterations to be the optimal number for minimizing the largest of the KS statistics. This value can be found in the `ps.lalonde$desc$ks.max.ATT$n.trees`. Figure 1 shows the barchart of the relative influence and is produced when `plot=TRUE` in the call to `summary()`.

```
> summary(ps.lalonde$gbm.obj,
+         n.trees=ps.lalonde$desc$ks.max.ATT$n.trees,
+         plot=FALSE)
```

```
             var     rel.inf
black       black 59.63958492
age           age 17.51573062
re74         re74 14.46374950
re78         re78  3.06884428
married   married  2.73870020
educ         educ  2.19683545
nodegree nodegree  0.29462679
hispan     hispan  0.08192825
```

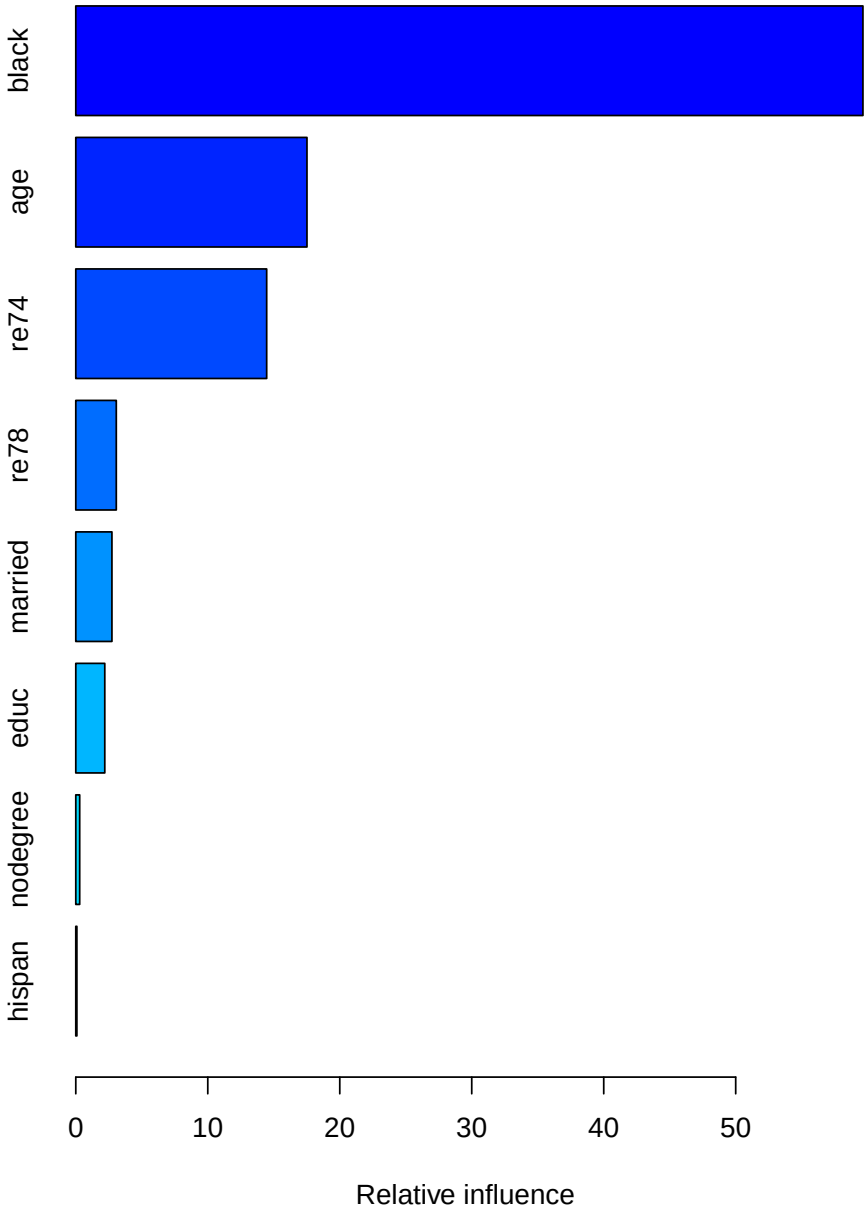


Figure 1: Relative influence of the covariates on the estimated propensity score.

2.3 Assessing "balance" using balance tables

Having estimated the propensity scores, `bal.table()` produces a table that shows how well the resulting weights succeed in manipulating the control group so that its weighted pretreatment characteristics match, or balance, those of the unweighted treatment group if `estimand = "ATT"` or the control and treatment groups so that the weighted pretreatment characteristics match, or balance, with one another if `estimand = "ATE"`. By default, the `bal.table()` function uses the value of `estimand` set with the `ps()` function call (default value of `estimand = "ATE"`. For example, in the analysis we set `estimand = "ATT"` when calling `ps()` to estimate the propensity scores and the resulting `ps.object`, `ps.lalonde`, contains an element "estimand" which takes the value `"ATT"`. The function `bal.table()` checks this value and automatically uses `ATT` weights when checking balance and comparing the distributions of pre-treatment variables for the weighted control group with those from the unweighted treatment group.

```
> lalonde.balance <- bal.table(ps.lalonde)
> lalonde.balance

$unw
            tx.mn    tx.sd    ct.mn    ct.sd std.eff.sz   stat     p    ks ks.pval
age        25.816    7.155   28.030   10.787     -0.309 -2.994 0.003 0.158   0.003
educ       10.346    2.011   10.235    2.855      0.055  0.547 0.584 0.111   0.081
black       0.843    0.365    0.203    0.403      1.757 19.371 0.000 0.640   0.000
hispan      0.059    0.237    0.142    0.350     -0.349 -3.413 0.001 0.083   0.339
nodegree    0.708    0.456    0.597    0.491      0.244  2.716 0.007 0.111   0.081
married     0.189    0.393    0.513    0.500     -0.824 -8.607 0.000 0.324   0.000
re74     2095.574 4886.620 5619.237 6788.751     -0.721 -7.254 0.000 0.447   0.000
re78     6349.144 7867.402 6984.170 7294.162     -0.081 -0.939 0.348 0.099   0.162

$es.mean.ATT
            tx.mn    tx.sd    ct.mn    ct.sd std.eff.sz   stat     p    ks ks.pval
age        25.816    7.155   25.146    7.091      0.094  0.759 0.448 0.087   0.993
educ       10.346    2.011   10.677    2.030     -0.165 -1.087 0.277 0.095   0.983
black       0.843    0.365    0.831    0.375      0.033  0.262 0.793 0.012   1.000
hispan      0.059    0.237    0.044    0.204      0.067  0.765 0.444 0.016   1.000
nodegree    0.708    0.456    0.614    0.488      0.207  0.963 0.336 0.095   0.983
married     0.189    0.393    0.189    0.392      0.000  0.001 0.999 0.000   1.000
re74     2095.574 4886.620 1785.471 4137.463      0.063  0.542 0.588 0.043   1.000
re78     6349.144 7867.402 6059.538 5435.679      0.037  0.316 0.752 0.140   0.763

$ks.max.ATT
            tx.mn    tx.sd    ct.mn    ct.sd std.eff.sz   stat     p    ks ks.pval
age        25.816    7.155   25.135    7.568      0.095  0.780 0.436 0.116   0.763
educ       10.346    2.011   10.669    2.182     -0.161 -1.095 0.274 0.108   0.832
black       0.843    0.365    0.813    0.391      0.084  0.718 0.473 0.031   1.000
hispan      0.059    0.237    0.044    0.206      0.064  0.751 0.453 0.015   1.000
nodegree    0.708    0.456    0.600    0.490      0.236  1.262 0.207 0.108   0.832
married     0.189    0.393    0.207    0.405     -0.045 -0.327 0.744 0.017   1.000
re74     2095.574 4886.620 1978.965 4453.419      0.024  0.208 0.836 0.035   1.000
re78     6349.144 7867.402 6074.258 5700.234      0.035  0.305 0.760 0.115   0.766
```

`bal.table()` returns information on the pretreatment covariates before and after weighting. The object is a list with named components, one for an unweighted analysis (named `unw`) and one for each `stop.method` specified, here `es.mean` and `ks.max`. McCaffrey et al (2004) essentially used `es.mean` for the analyses, but

our more recent work has sometimes used `ks.max`. See McCaffrey et al. (2013) for a more detailed description of these choices.

If there are missing values (represented as `NA`) in the covariates, `twang` will attempt to construct weights that also balance rates of missingness in the treatment and control arms. In this case, the `bal.table()` will have an extra row for each variable that has missing entries. User should note that missing data in `xgboost` is handled differently than in `gbm`. In `gbm`, missing values are placed in their own node, while in `xgboost` missing values are placed in the left or right node based on minimizing the objective function.

The columns of the table consist of the following items:

`tx.mn, ct.mn` The treatment means and the control means for each of the variables. The unweighted table `(unw)` shows the unweighted means. For each stopping rule the means are weighted using weights corresponding to the gbm model selected by `ps()` using the stopping rule. When `estimand = "ATT"` the weights for the treatment group always equal 1 for all cases and there is no difference between unweighted and propensity score weighted tx.mn.

`tx.sd, ct.sd` The propensity score weighted treatment and control groups' standard deviations for each of the variables. The unweighted table (unw) shows the unweighted standard deviations.

`std.eff.sz` The standardized effect size, defined as the treatment group mean minus the control group mean divided by the treatment group standard deviation if `estimand = "ATT"` or divided by the pooled sample (treatment and control) standard deviation if `estimand = "ATE"`. (In discussions of propensity scores this value is sometimes referred to as "standardized bias".) Occasionally, lack of treatment group or pooled sample variance on a covariate results in very large (or infinite) standardized effect sizes. For purposes of analyzing mean effect sizes across multiple covariates, we set all standardized effect sizes larger than 500 to `NA` (missing values).

`stat, p` Depending on whether the variable is continuous or categorical, `stat` is a t-statistic or a χ^2 statistic. `p` is the associated p-value

`ks, ks.pval` The Kolmogorov-Smirnov test statistic and its associated p-value. P-values for the KS statistics are either derived from Monte Carlo simulations or analytic approximations, depending on the specifications made in the `perm.test.iters` argument of the `ps` function. For categorical variables this is just the χ^2 test p-value.

Components of these tables are useful for demonstrating that pretreatment differences between groups on observed variables have been eliminated using the weights. The `xtable` package aids in formatting for LaTeX and Word documents. Table 1 shows the results for `ks.max` reformatted for a LaTeX document. For Word documents, paste the LaTeX description of the table into a Word document, highlight it and use Word tools to convert the text to a table using "&" as the separator.

```
> library(xtable)
> pretty.tab <- lalonde.balance$ks.max.ATT[,c("tx.mn","ct.mn","ks")]
> pretty.tab <- cbind(pretty.tab, lalonde.balance$unw[,"ct.mn"])
> names(pretty.tab) <- c("E(Y1|t=1)","E(Y0|t=1)","KS","E(Y0|t=0)")
> xtable(pretty.tab,
+        caption = "Balance of the treatment and comparison groups",
+        label = "tab:balance",
+        digits = c(0, 2, 2, 2, 2),
+        align=c("l","r","r","r","r"))
```

	E(Y1|t=1)	E(Y0|t=1)	KS	E(Y0|t=0)
age	25.82	25.14	0.12	28.03
educ	10.35	10.67	0.11	10.23
black	0.84	0.81	0.03	0.20
hispan	0.06	0.04	0.01	0.14
nodegree	0.71	0.60	0.11	0.60
married	0.19	0.21	0.02	0.51
re74	2095.57	1978.96	0.04	5619.24
re78	6349.14	6074.26	0.12	6984.17

Table 1: Balance of the treatment and comparison groups

The `summary()` method for `ps` objects offers a compact summary of the sample sizes of the groups and the balance measures. If `perm.test.iters>0` was used to create the `ps` object, then Monte Carlo simulation is used to estimate p-values for the maximum KS statistic that would be expected across the covariates, had individuals with the same covariate values been assigned to groups randomly. Thus, a p-value of 0.04 for `max.ks.p` indicates that the largest KS statistic found across the covariates is larger than would be expected in 96% of trials in which the same cases were randomly assigned to groups.

```
> summary(ps.lalonde)

            n.treat n.ctrl ess.treat  ess.ctrl    max.es    mean.es    max.ks
unw             185    429       185 429.00000 1.7567745 0.54253254 0.6404460
es.mean.ATT     185    429       185  23.82797 0.2074424 0.08325078 0.1395227
ks.max.ATT      185    429       185  36.43108 0.2362076 0.09285819 0.1160305
            max.ks.p    mean.ks iter
unw               NA 0.24661532   NA
es.mean.ATT       NA 0.06077614 1672
ks.max.ATT        NA 0.06811037 1151
```

In general, weighted means can have greater sampling variance than unweighted means from a sample of equal size. The effective sample size (ESS) of the weighted comparison group captures this increase in variance as

$$\text{ESS} = \frac{\left(\sum_{i \in C} w_i\right)^2}{\sum_{i \in C} w_i^2}. \tag{1}$$

The ESS is approximately the number of observations from a simple random sample that yields an estimate with sampling variation equal to the sampling variation obtained with the weighted comparison observations. Therefore, the ESS will give an estimate of the number of comparison participants that are comparable to the treatment group when `estimand = "ATT"`. The ESS is an accurate measure of the relative size of the variance of means when the weights are fixed or they are uncorrelated with outcomes. Otherwise the ESS underestimates the effective sample size (Little & Vartivarian, 2004). With propensity score weights, it is rare that weights are uncorrelated with outcomes. Hence the ESS typically gives a lower bound on the effective sample size, but it still serves as a useful measure for choosing among alternative models and assessing the overall quality of a model, even if it provides a possibly conservative picture of the loss in precision due to weighting.

The `ess.treat` and `ess.ctrl` columns in the summary results shows the ESS for the estimated propensity scores. Note that although the original comparison group had 429 cases, the propensity score estimates effectively utilize only 24 or 36.4 of the comparison cases, depending on the rules and measures used to estimate the propensity scores. While this may seem like a large loss of sample size, this indicates that many

of the original cases were unlike the treatment cases and, hence, were not useful for isolating the treatment effect. Moreover, similar or even greater reductions in ESS would be expected from alternative approaches to using propensity scores, such as matching or stratification. Since the estimand of interest in this example is ATT, `ess.treat = n.treat` throughout (i.e., all treatment cases have a weight of 1).

2.4 Graphical assessments of balance

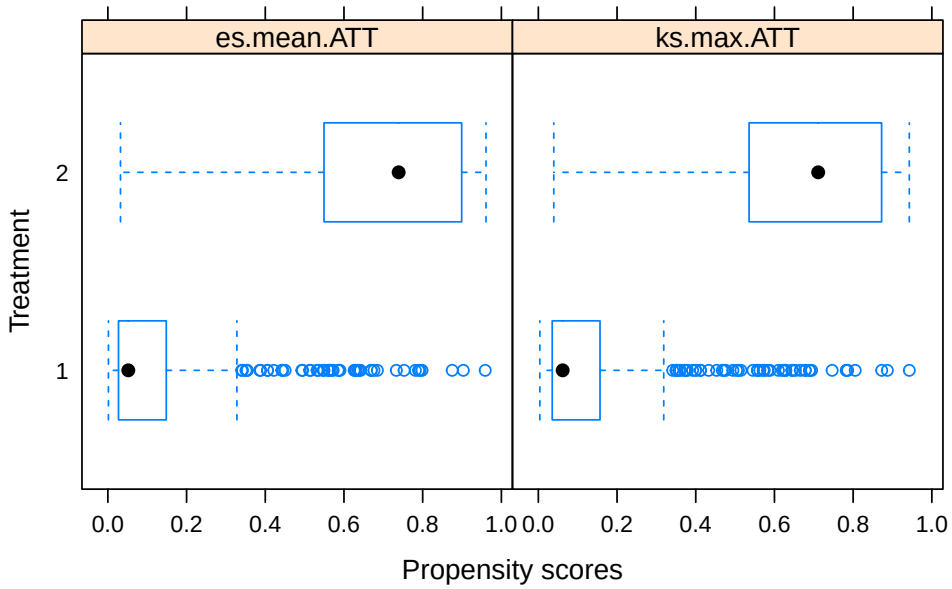
The `plot()` function can generate useful diagnostic plots from the propensity score objects. The full set of plots available in `twang` and the argument value of `plot` to produce each one are given in Table 2. The convergence plot — the default — was discussed above.

Descriptive argument	Numeric argument	Description
`"optimize"`	1	Balance measure as a function of GBM iterations
`"boxplot"`	2	Boxplot of treatment/control propensity scores
`"es"`	3	Standardized effect size of pretreatment variables
`"t"`	4	t-test p-values for weighted pretreatment variables
`"ks"`	5	Kolmogorov-Smirnov p-values for weighted pretreatment variables
`"histogram"`	6	Histogram of weights for treatment/control

Table 2: Available options for `plots` argument to `plot()` function.

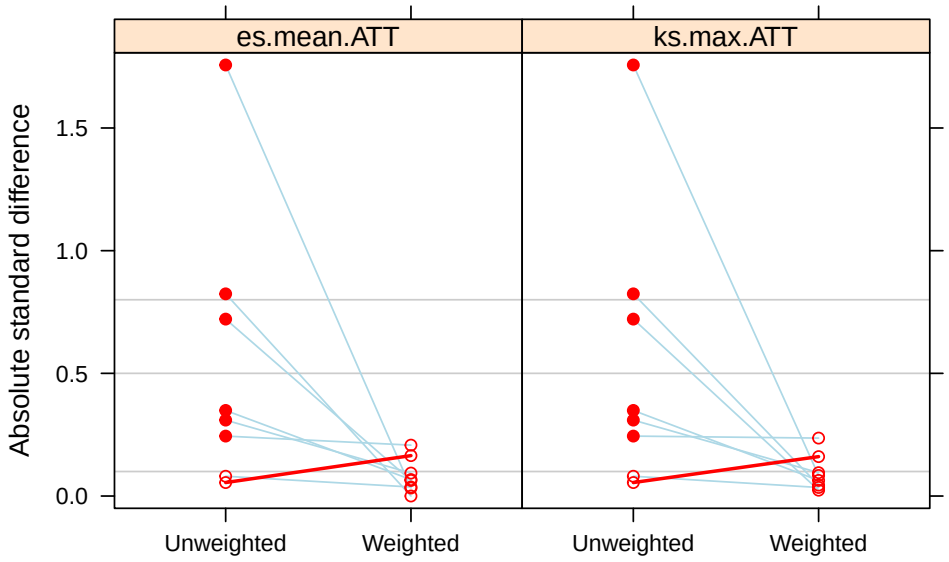
The `plot()` function takes a `plots` argument in order to produce other diagnostic plots. For example, specifying `plots = 2` or `plots = "boxplot"` produces boxplots illustrating the spread of the estimated propensity scores in the treatment and comparison groups. Whereas propensity score stratification requires considerable overlap in these spreads, excellent covariate balance can often be achieved with weights, even when the propensity scores estimated for the treatment and control groups show little overlap.

```
> plot(ps.lalonde, plots=2)
```



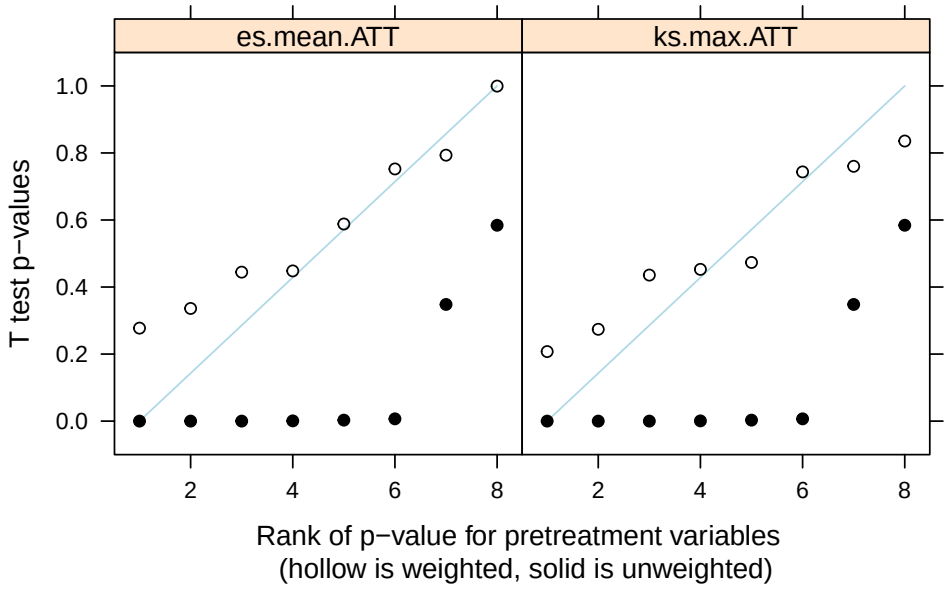
The effect size plot illustrates the effect of weights on the magnitude of differences between groups on each pretreatment covariate. These magnitudes are standardized using the standardized effect size described earlier. In these plots, substantial reductions in effect sizes are observed for most variables (blue lines), with only one variable showing an increase in effect size (red lines), but only a seemingly trivial increase. Closed red circles indicate a statistically significant difference, many of which occur before weighting, none after. In some analyses variables can have very little variance in the treatment group sample or the entire sample and group differences can be very large relative to the standard deviations. In these situations, the user is warned that some effect sizes are too large to plot.

```
> plot(ps.lalonde, plots=3)
```



When many of the p-values testing individual covariates for balance are very small, the groups are clearly imbalanced and inconsistent with what we would expect had the groups been formed by random assignment. After weighting we would expect the p-values to be larger if balance had been achieved. We use a QQ plot comparing the quantiles of the observed p-values to the quantiles of the uniform distribution (45 degree line) to conduct this check of balance. Ideally, the p-values from independent tests in which the null hypothesis is true will have a uniform distribution. Although the ideal is unlikely to hold even if we had random assignment (Bland, 2013), severe deviation of the p-values below the diagonal suggests lack of balance and p-values running at or above the diagonal suggests balance might have been achieved. The p-value plot (`plots=4` or `plots="t"`) allows users to visually to inspect the p-values of the t-tests for group differences in the covariate means.

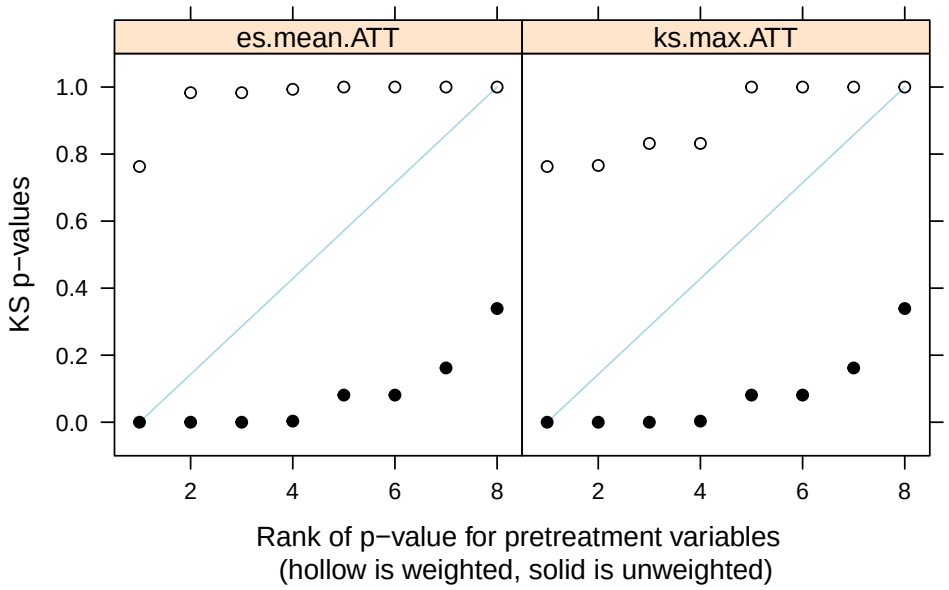
```
> plot(ps.lalonde, plots = 4)
```



Before weighting (closed circles), the groups have statistically significant differences on many variables (i.e., p-values are near zero). After weighting (open circles) the p-values are generally above the 45-degree line, which represents the cumulative distribution of a uniform variable on [0,1]. This indicates that the p-values are even larger than would be expected in a randomized study.

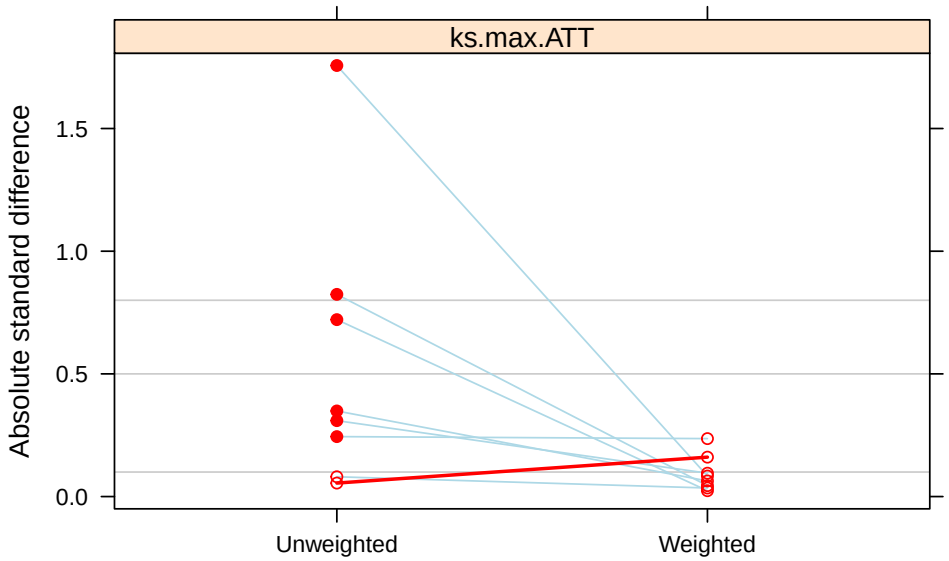
One can inspect similar plots for the KS statistic with the argument `plots = "ks"` or `plots = 5`.

```
> plot(ps.lalonde, plots = 5)
```



In all cases, the `subset` argument can be used if we wish to focus on results from one stopping rule.

```
> plot(ps.lalonde, plots = 3, subset = 2)
```



2.5 Analysis of outcomes

A separate R package, the `survey` package, is useful for performing the outcomes analyses using weights. Its statistical methods account for the weights when computing standard error estimates. It is not a part of the standard R installation but installing `twang` should automatically install `survey` as well.

```
> library(survey)
```

The `get.weights()` function extracts the propensity score weights from a `ps` object. Those weights may then be used as case weights in a `svydesign` object. By default, it returns weights corresponding to the estimand (ATE or ATT) that was specified in the original call to `ps()`. If needed, the user can override the default via the optional `estimand` argument.

```
> lalonde$w <- get.weights(ps.lalonde, stop.method="es.mean")
> design.ps <- svydesign(ids=~1, weights=~w, data=lalonde)
```

The `stop.method` argument specifies which GBM model, and consequently which weights, to utilize.

The `svydesign` function from the `survey` package creates an object that stores the dataset along with design information needed for analyses. See `help(svydesign)` for more details on setting up `svydesign` objects.

The aim of the National Supported Work Demonstration analysis is to determine whether the program was effective at increasing earnings in 1978. The propensity score adjusted test can be computed with `svyglm`.

```
> glm1 <- svyglm(re78 ~ treat, design=design.ps)
> summary(glm1)
```

```
Call:
svyglm(formula = re78 ~ treat, design = design.ps)

Survey design:
svydesign(ids = ~1, weights = ~w, data = lalonde)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6059.5      712.9   8.499   <2e-16 ***
treat          289.6      917.4   0.316    0.752
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 47808696)

Number of Fisher Scoring iterations: 2
```

The analysis estimates an increase in earnings of $290 for those that participated in the NSW compared with similarly situated people observed in the CPS. The effect, however, does not appear to be statistically significant.

Some authors have recommended utilizing both propensity score adjustment and additional covariate adjustment to minimize mean square error or to obtain "doubly robust" estimates of the treatment effect (Huppler-Hullsiek & Louis 2002, Bang & Robins 2005). These estimators are consistent if either the propensity scores are estimated correctly *or* the regression model is specified correctly. For example, note that the balance table for `ks.max.ATT` made the two groups more similar on `nodegree`, but still some differences remained, 70.8% of the treatment group had no degree while 60% of the comparison group had no degree. While linear regression is sensitive to model misspecification when the treatment and comparison groups are dissimilar, the propensity score weighting has made them more similar, perhaps enough so that additional modeling with covariates can adjust for any remaining differences. In addition to potential bias reduction, the inclusion of additional covariates can reduce the standard error of the treatment effect if some of the covariates are strongly related to the outcome.

```
> glm2 <- svyglm(re78 ~ treat + nodegree, design=design.ps)
> summary(glm2)

Call:
svyglm(formula = re78 ~ treat + nodegree, design = design.ps)

Survey design:
svydesign(ids = ~1, weights = ~w, data = lalonde)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7182.7     1192.8   6.022 2.98e-09 ***
treat          462.7      940.7   0.492   0.6230
nodegree     -1830.6     1089.8  -1.680   0.0935 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 47070924)

Number of Fisher Scoring iterations: 2
```

Adjusting for the remaining group difference in the `nodegree` variable slightly increased the estimate of the program's effect to $463, but the difference is still not statistically significant. We can further adjust for the other covariates, but that too in this case has little effect on the estimated program effect.

```
> glm3 <- svyglm(re78 ~ treat + age + educ + black + hispan + nodegree +
+                       married + re74 + re75,
+               design=design.ps)
> summary(glm3)

Call:
svyglm(formula = re78 ~ treat + age + educ + black + hispan +
    nodegree + married + re74 + re75, design = design.ps)

Survey design:
svydesign(ids = ~1, weights = ~w, data = lalonde)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.092e+03  4.264e+03  -0.491  0.62390
treat        3.959e+02  9.129e+02   0.434  0.66464
age          3.965e+01  5.375e+01   0.738  0.46100
educ         6.674e+02  2.565e+02   2.601  0.00951 **
black       -6.558e+02  9.595e+02  -0.683  0.49457
hispan       6.167e+02  1.661e+03   0.371  0.71049
nodegree     3.847e+02  1.472e+03   0.261  0.79397
married      6.804e+02  1.035e+03   0.657  0.51118
re74         6.461e-02  1.630e-01   0.397  0.69185
re75         6.283e-02  1.490e-01   0.422  0.67349
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 45688078)

Number of Fisher Scoring iterations: 2
```

2.6 Estimating the program effect using linear regression

The more traditional regression approach to estimating the program effect would fit a linear model with a treatment indicator and linear terms for each of the covariates.

```
> glm4 <- lm(re78 ~ treat + age + educ + black + hispan + nodegree +
+                   married + re74 + re75,
+            data=lalonde)
> summary(glm4)

Call:
lm(formula = re78 ~ treat + age + educ + black + hispan + nodegree +
    married + re74 + re75, data = lalonde)

Residuals:
   Min     1Q Median     3Q    Max
```

```
-13595  -4894  -1662   3929  54570

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.651e+01  2.437e+03   0.027   0.9782
treat        1.548e+03  7.813e+02   1.982   0.0480 *
age          1.298e+01  3.249e+01   0.399   0.6897
educ         4.039e+02  1.589e+02   2.542   0.0113 *
black       -1.241e+03  7.688e+02  -1.614   0.1071
hispan       4.989e+02  9.419e+02   0.530   0.5966
nodegree     2.598e+02  8.474e+02   0.307   0.7593
married      4.066e+02  6.955e+02   0.585   0.5590
re74         2.964e-01  5.827e-02   5.086 4.89e-07 ***
re75         2.315e-01  1.046e-01   2.213   0.0273 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6948 on 604 degrees of freedom
Multiple R-squared:  0.1478,        Adjusted R-squared:  0.1351
F-statistic: 11.64 on 9 and 604 DF,  p-value: < 2.2e-16
```

This model estimates a rather strong treatment effect, estimating a program effect of $1548 with a p-value=0.048. Several variations of this regression approach also estimate strong program effects. For example using square root transforms on the earnings variables yields a p-value=0.016. These estimates, however, are very sensitive to the model structure since the treatment and control subjects differ greatly as seen in the unweighted balance comparison ($unw) from `bal.table(ps.lalonde)`.

2.7 Propensity scores estimated from logistic regression

Propensity score analysis is intended to avoid problems associated with the misspecification of covariate adjusted models of outcomes, but the quality of the balance and the treatment effect estimates can be sensitive to the method used to estimate the propensity scores. Consider estimating the propensity scores using logistic regression instead of `ps()`.

```
> ps.logit <- glm(treat ~ age + educ + black + hispan + nodegree +
+                         married + re74 + re78,
+                 data = lalonde,
+                 family = binomial)
> lalonde$w.logit <- rep(1,nrow(lalonde))
> lalonde$w.logit[lalonde$treat==0] <- exp(predict(ps.logit,subset(lalonde,treat==0)))
```

`predict()` for logistic regression model produces estimates on the log-odds scale by default. Exponentiating those predictions for the comparison subjects gives the ATT weights $p/(1-p)$. `dx.wts()` from the `twang` package diagnoses the balance for an arbitrary set of weights producing a balance table. This function requires the user to specify the estimand argument in order to perform the appropriate calculations relative to the target group on which we are drawing inferences. The function `dx.wts` has not been updated in Version 2.0 and still relies on the older version of the balance calculations.

```
> bal.logit <- dx.wts(x = lalonde$w.logit,
+                     data=lalonde,
+                     vars=c("age","educ","black","hispan","nodegree",
```

```
+                       "married","re74","re78"),
+                     treat.var="treat",
+                     perm.test.iters=0, estimand = "ATT")
> bal.logit

  type n.treat n.ctrl ess.treat  ess.ctrl   max.es    mean.es    max.ks
1  unw     185    429       185 429.00000 1.756775 0.54253254 0.6404460
2          185    429       185  96.95957 0.142658 0.03794634 0.3171096
     mean.ks iter
1 0.24661532   NA
2 0.08555837   NA
```

Applying the `bal.table()` function to this object returns a variable-by-variable summary of balance, just like it did for the `ps` object.

```
> bal.logit <- dx.wts(x = lalonde$w.logit,
+                     data=lalonde,
+                     vars=c("age","educ","black","hispan","nodegree",
+                       "married","re74","re78"),
+                     treat.var="treat",
+                     perm.test.iters=0, estimand = "ATT")
> bal.logit

  type n.treat n.ctrl ess.treat  ess.ctrl   max.es    mean.es    max.ks
1  unw     185    429       185 429.00000 1.756775 0.54253254 0.6404460
2          185    429       185  96.95957 0.142658 0.03794634 0.3171096
     mean.ks iter
1 0.24661532   NA
2 0.08555837   NA
```

For weights estimated with logistic regression, the largest KS statistic was reduced from the unweighted sample's largest KS of 0.64 to 0.31, which is still quite a large KS statistic. Table 3 shows the details of the balance of the treatment and comparison groups. The means of the two groups appear to be quite similar while the KS statistic shows substantial differences in their distributions.

```
> pretty.tab <- bal.table(bal.logit)[[2]][,c("tx.mn","ct.mn","ks")]
> pretty.tab <- cbind(pretty.tab, bal.table(bal.logit)[[1]]$ct.mn)
> names(pretty.tab) <- c("E(Y1|t=1)","E(Y0|t=1)","KS","E(Y0|t=0)")
> xtable(pretty.tab,
+        caption = "Logistic regression estimates of the propensity scores",
+        label = "tab:balancelogit",
+        digits = c(0, 2, 2, 2, 2),
+        align=c("l","r","r","r","r"))
```

Table 4 compares the balancing quality of the weights directly with one another.

```
> design.logit <- svydesign(ids=~1, weights=~w.logit, data=lalonde)
> glm6 <- svyglm(re78 ~ treat, design=design.logit)
> summary(glm6)

Call:
svyglm(formula = re78 ~ treat, design = design.logit)
```

```
Survey design:
svydesign(ids = ~1, weights = ~w.logit, data = lalonde)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6096.8      686.6   8.879   <2e-16 ***
treat          252.4      897.1   0.281    0.779
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 52483292)

Number of Fisher Scoring iterations: 2
```

	E(Y1|t=1)	E(Y0|t=1)	KS	E(Y0|t=0)
age	25.82	24.80	0.32	28.03
educ	10.35	10.39	0.04	10.23
black	0.84	0.85	0.00	0.20
hispan	0.06	0.06	0.00	0.14
nodegree	0.71	0.70	0.01	0.60
married	0.19	0.17	0.02	0.51
re74	2095.57	2185.76	0.22	5619.24
re78	6349.14	6096.79	0.07	6984.17

Table 3: Logistic regression estimates of the propensity scores

	n.treat	ess.ctrl	max.es	mean.es	max.ks	mean.ks
unw	185	429.00	1.76	0.54	0.64	0.25
logit	185	96.96	0.14	0.04	0.32	0.09
es.mean.ATT	185	23.83	0.21	0.08	0.14	0.06
ks.max.ATT	185	36.43	0.24	0.09	0.12	0.07

Table 4: Summary of the balancing properties of logistic regression and gbm

The analysis estimates an increase in earnings of $252 for those that participated in the NSW compared with similarly situated people observed in the CPS. Table 5 compares all of the treatment effect estimates.

Treatment effect	PS estimate	Linear adjustment
$290	GBM, minimize ES	none
$463	GBM, minimize ES	nodegree
$396	GBM, minimize ES	all
$1548	None	all
$252	Logistic regression	none
$236	Logistic regression	all

Table 5: Treatment effect estimates by various methods

3 An ATE example

In the analysis of Section 2, we focused on estimating ATT for the `lalonde` dataset. In this situation, the ATE is not of great substantive interest because not all people who are offered entrance into the program could be expected to take advantage of the opportunity. Further, there is some evidence that the treated subjects were drawn from a subset of the covariate space. In particular, in an ATE analysis, we see that we are unable to achieve balance, especially for the "black" indicator.

We now turn to an ATE analysis that is feasible and meaningful. We focus on the `lindner` dataset, which was included in the `USPS` package (Obenchain 2011), and is now included in `twang` for convenience. A tutorial by Helmreich and Pruzek (2009; HP) for the `PSAgraphics` package also uses propensity scores to analyze a portion of these data. HP describe the data as follows on p. 3 with our minor recodings in square braces:

> The `lindner` data contain data on 996 patients treated at the Lindner Center, Christ Hospital, Cincinnati in 1997. Patients received a Percutaneous Coronary Intervention (PCI). The data consists of 10 variables. Two are outcomes: [`sixMonthSurvive`] ranges over two values... depending on whether patients surved to six months post treatment [denoted by `TRUE`] or did not survive to six months [`FALSE`]... Secondly, `cardbill` contains the costs in 1998 dollars for the first six months (or less if the patient did not survive) after treatment... The treatment variable is `abcix`, where 0 indicates PCI treatment and 1 indicates standard PCI treatment and additional treatment in some form with abciximab. Covariates include `acutemi`, 1 indicating a recent acute myocardial infarction and 0 not; `ejecfrac` for the left ventricle ejection fraction, a percentage from 0 to 90; `ves1proc` giving the number of vessels (0 to 5) involved in the initial PCI; `stent` with 1 indicating coronary stent inserted, 0 not; `diabetic` where 1 indicates that the patient has been diagnosed with diabetes, 0 not; `height` in centimeters and `female` coding the sex of the patent, 1 for female, 0 for male.

HP focus on `cardbill` — the cost for the first months after treatment — as their outcome of interest. However, since not all patients survived to six months, it is not clear whether a lower value of `cardbill` is good or not. For this reason, we choose six-month survival (`sixMonthSurvive`) as our outcome of interest.

Ignoring pre-treatment variables, we see that `abcix` is associated with lower rates of 6-month mortality:

```
> data(lindner)
> table(lindner$sixMonthSurvive, lindner$abcix)

         0   1
  FALSE  15  11
  TRUE  283 687

> chisq.test(table(lindner$sixMonthSurvive, lindner$abcix))

        Pearson's Chi-squared test with Yates' continuity correction

data:  table(lindner$sixMonthSurvive, lindner$abcix)
X-squared = 8.5077, df = 1, p-value = 0.003536
```

The question is whether this association is causal. If health care policies were to be made on the basis of these data, we would wish to elicit expert opinion as to whether there are likely to be other confounding pretreatment variables. For this tutorial, we simply follow HP in choosing the pre-treatment covariates. The `twang` model is fit as follows

```
> set.seed(1)
> ps.lindner <- ps(abcix ~ stent + height + female + diabetic +
+                  acutemi + ejecfrac + ves1proc,
+                  data = lindner,
+                  estimand = "ATE",
+                  verbose = FALSE)
```

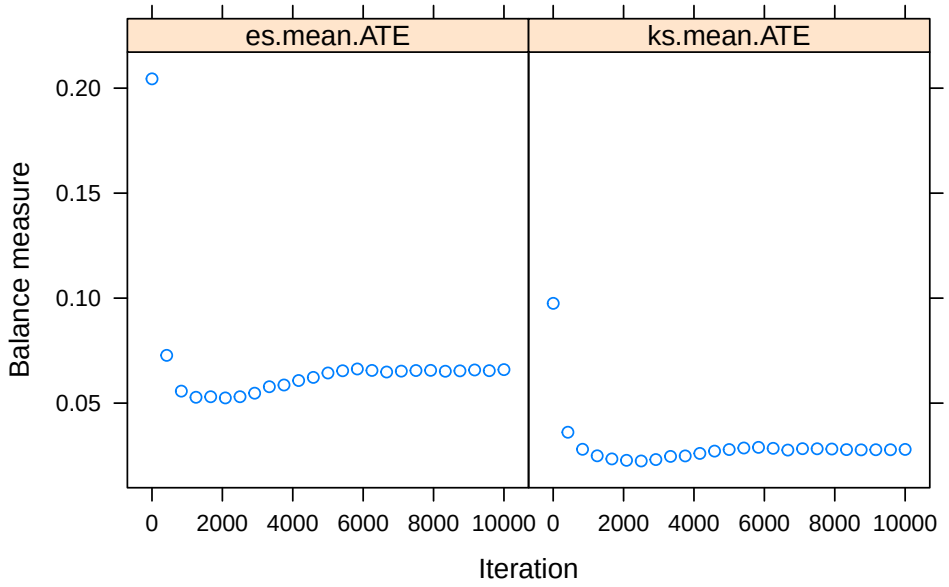
We set `estimand = "ATE"` because we are interested in the effects of abciximab on everyone in the population. We do not specify the stopping rules. Consequently `ps()` uses the defaults: `es.mean` and `ks.mean`. We then inspect pre- and post-weighting balance with the command

```
> bal.table(ps.lindner)
```

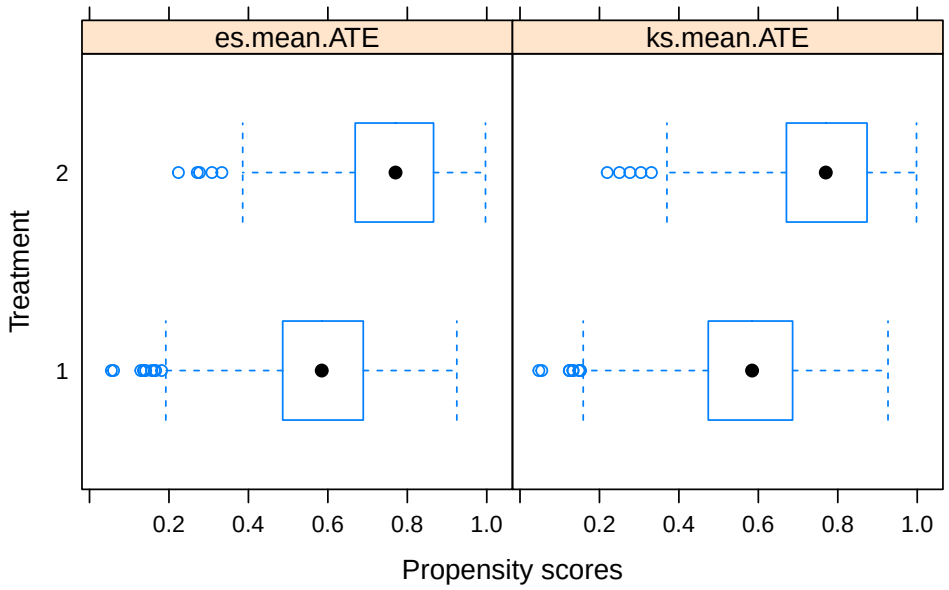
```
$unw
           tx.mn  tx.sd   ct.mn  ct.sd std.eff.sz   stat     p    ks ks.pval
stent      0.705  0.456   0.584  0.494      0.257  3.624 0.000 0.121   0.004
height   171.443 10.695 171.446 10.589      0.000 -0.005 0.996 0.025   0.999
female     0.331  0.471   0.386  0.488     -0.115 -1.647 0.100 0.055   0.554
diabetic   0.205  0.404   0.268  0.444     -0.152 -2.127 0.034 0.064   0.367
acutemi    0.179  0.384   0.060  0.239      0.338  5.923 0.000 0.119   0.006
ejecfrac  50.403 10.419  52.289 10.297     -0.181 -2.640 0.008 0.114   0.009
ves1proc   1.463  0.706   1.205  0.480      0.393  6.693 0.000 0.188   0.000

$ks.mean.ATE
           tx.mn  tx.sd   ct.mn  ct.sd std.eff.sz   stat     p    ks ks.pval
stent      0.683  0.466   0.657  0.475      0.054  0.720 0.472 0.026   1.000
height   171.470 10.549 171.589 10.594     -0.011 -0.153 0.879 0.015   1.000
female     0.338  0.473   0.345  0.476     -0.015 -0.202 0.840 0.007   1.000
diabetic   0.215  0.411   0.229  0.421     -0.033 -0.430 0.667 0.014   1.000
acutemi    0.148  0.355   0.107  0.310      0.114  1.333 0.183 0.040   0.947
ejecfrac  51.051 10.334  51.604  9.112     -0.053 -0.799 0.425 0.027   1.000
ves1proc   1.395  0.666   1.337  0.573      0.089  1.199 0.231 0.027   1.000

$es.mean.ATE
           tx.mn  tx.sd   ct.mn  ct.sd std.eff.sz   stat     p    ks ks.pval
stent      0.683  0.466   0.656  0.476      0.056  0.751 0.453 0.027    1.00
height   171.467 10.542 171.586 10.660     -0.011 -0.151 0.880 0.016    1.00
female     0.338  0.473   0.345  0.476     -0.015 -0.206 0.837 0.007    1.00
diabetic   0.215  0.411   0.231  0.422     -0.039 -0.506 0.613 0.016    1.00
acutemi    0.148  0.355   0.108  0.311      0.113  1.327 0.185 0.040    0.95
ejecfrac  51.037 10.348  51.546  9.171     -0.049 -0.733 0.464 0.027    1.00
ves1proc   1.396  0.666   1.342  0.579      0.082  1.087 0.277 0.025    1.00
```

This balance table shows that `stent`, `acutemi`, and `ves1proc` were all significantly imbalanced before weighting. After weighting (using either `stop.method` considered) we do not see problems in this regard. Examining `plot(ps.lindner, plots = x)` for `x` running from 1 to 5 does not reveal problems, either. In regard to the optimize plot, we note that the scales of the KS and ES statistics presented in the optimize plots are not necessarily comparable. The fact that the KS values are lower than the ES values in the optimize plot does not suggest that the KS stopping rule is finding superior models. Each panel of the optimize plot indicates the gbm model that minimizes each stopping rule. The panels should not be compared other than to compare the number of iterations selected by each rule.

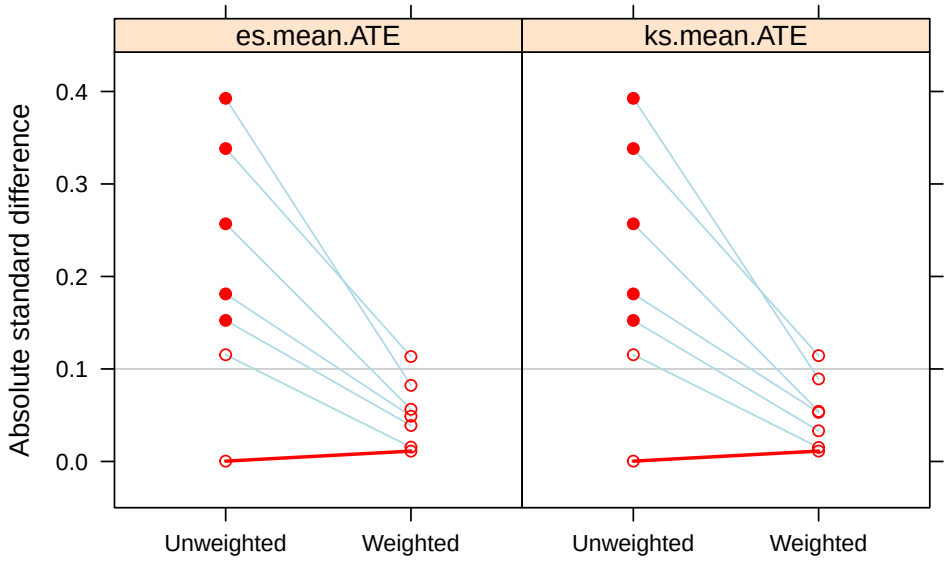
```
> plot(ps.lindner, plots = 1)
```



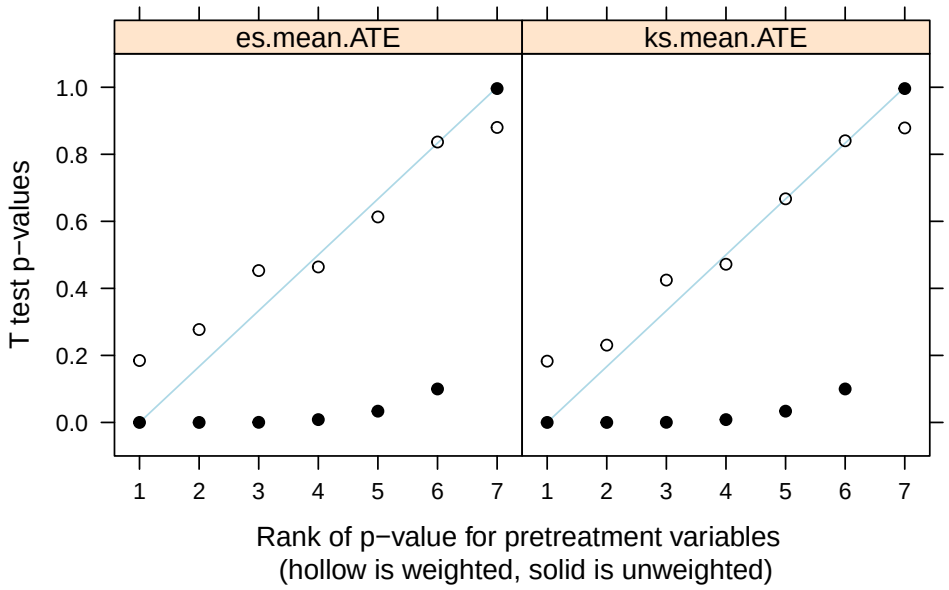

```
> plot(ps.lindner, plots = 2)
```



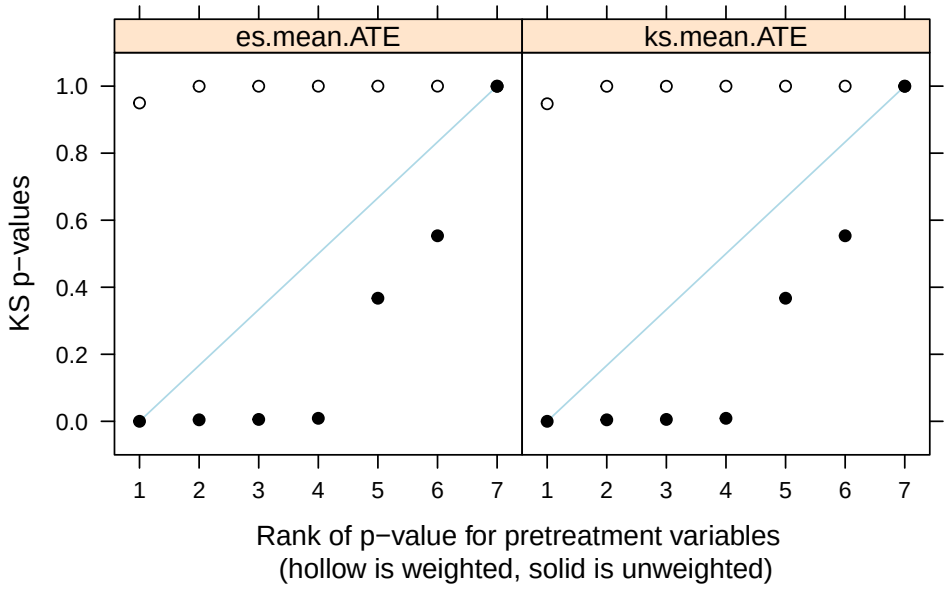
```
> plot(ps.lindner, plots = 3)
```




```
> plot(ps.lindner, plots = 4)
```



```
> plot(ps.lindner, plots = 5)
```



From a call to `summary()`, we see that the `es.mean.ATE` stopping rule results in a slightly higher ESS with comparable balance measures, so we proceed with those weights. Also, we note that `ess.treat` is no longer equal to `n.treat` since we are focusing on ATE rather than ATT.

```
> summary(ps.lindner)

            n.treat n.ctrl ess.treat ess.ctrl    max.es    mean.es     max.ks
unw             698    298  698.0000 298.0000 0.3925637 0.20528943 0.18841945
ks.mean.ATE     698    298  655.7339 229.0034 0.1144059 0.05292428 0.04013745
es.mean.ATE     698    298  658.3496 230.5940 0.1134167 0.05236774 0.03979038
            max.ks.p    mean.ks iter
unw               NA 0.09791845   NA
ks.mean.ATE       NA 0.02233864 2582
es.mean.ATE       NA 0.02262830 2116
```

As before, we use the `survey` package to reweight our sample and perform the analysis.

```
> lindner$w <- get.weights(ps.lindner, stop.method = "es.mean")
> design.ps <- svydesign(ids=~1, weights = ~w, data = lindner)
> svychisq(~sixMonthSurvive + abcix, design = design.ps)

        Pearson's X^2: Rao & Scott adjustment

data:  svychisq(~sixMonthSurvive + abcix, design = design.ps)
F = 9.3574, ndf = 1, ddf = 995, p-value = 0.00228
```

The reweighting does not diminish the association between the treatment and the outcome. Indeed, it is marginally more significant after the reweighting.

4 Non-response weights

The `twang` package was designed to estimate propensity score weights for the evaluation of treatment effects in observational or quasi-experimental studies. However, we find that the package includes functions and diagnostic tools that are highly valuable for other applications, such as for generating and diagnosing non-response weights for survey nonresponse or study attrition. We now present an example that uses the tools in `twang`. This example uses the subset of the US Sustaining Effects Study data distributed with the HLM software (Bryk, Raudenbush, Congdon, 1996) and also available in the R package `mlmRev`. The data include mathematics test scores for 1721 students in kindergarten to fourth grade. They also include student race (black, Hispanic, or other), gender, an indicator for whether or not the student had been retained in grade, the percent low income students at the school, the school size, the percent of mobile students, the students' grade-levels, student and school IDs, and grades converted to year by centering. The study analysis plans to analyze growth in math achievement from grade 1 to grade 4 using only students with complete data. However, the students with complete data differ from other students. To reduce bias that could potentially result from excluding incomplete cases, our analysis plan is to weight complete cases with nonresponse weights.

The goal of nonresponse weighting is to develop weights for the respondents that make them look like the entire sample — both the respondents and nonrespondents. Since the respondents already look like themselves, the hard part is to figure out how well each respondent represents the nonrespondents. Nonresponse weights equal the reciprocal of the probability of response and are applied only to respondents.

Note that the the probability of response is equivalent to the propensity score if we consider subjects with an observed outcome to be the "treated" group, and those with an unobserved outcome to be the "controls". We wish to reweight the sample to make it equivalent to the population from which the sample was drawn, so ATE weights are more appropriate in this case. Further, recall that the weights for the treated subjects are $1/p$ in an ATE analysis. Therefore we can reweight the sample of respondents using the `get.weights()` function.

Before we can generate nonresponse weights, we need to prepare the data using the following commands. First we load the data.

```
> data(egsingle)
```

Next we create a response indicator variable that we can merge onto the student by test score level data. We want to include only students with scores from all of grades 1 to 4. The data include scores from kindergarten (grade = 0) to grade 5 with some students having multiple scores from the same grade. First we keep the unique grades for each student:

```
> tmp <- tapply(egsingle$grade, egsingle$childid, unique)
```

Because students do not all have the same number of score, `tapply()` returns a list with one element per student. Each element contains the unique set of grades observed for each student. We now check for grades in 1 to 4:

```
> tmp <- lapply(tmp, function(x){return(x %in% 1:4)})
```

The list tmp now contains a boolean vector for each student, where "TRUE" indicates the grade took on a value in 1 to 4. The sum of this vector for each student determines how many of grades 1 to 4 we observed for him or her.

```
> tmp <- lapply(tmp, sum)
```

A student is a respondent if he or she has scores from all four of grades 1 to 4 or if the value of `tmp` is 4.

```
> tmp <- sapply(tmp, function(x){as.numeric(x == 4)})
```

We create a data frame of response indicators so we can merge them onto the test scores data:

```
> tmp <- data.frame(tmp)
> names(tmp) <- "resp"
> tmp$childid <- row.names(tmp)
```

and merge this back to create a single data frame

```
> egsingle <- merge(egsingle, tmp)
```

Because nonresponse is a student-level variable rather than a student-by-year-level variable we create one record per student.

```
> egsingle.one <-unique(egsingle[,-c(3:6)])
```

We also create a race variable

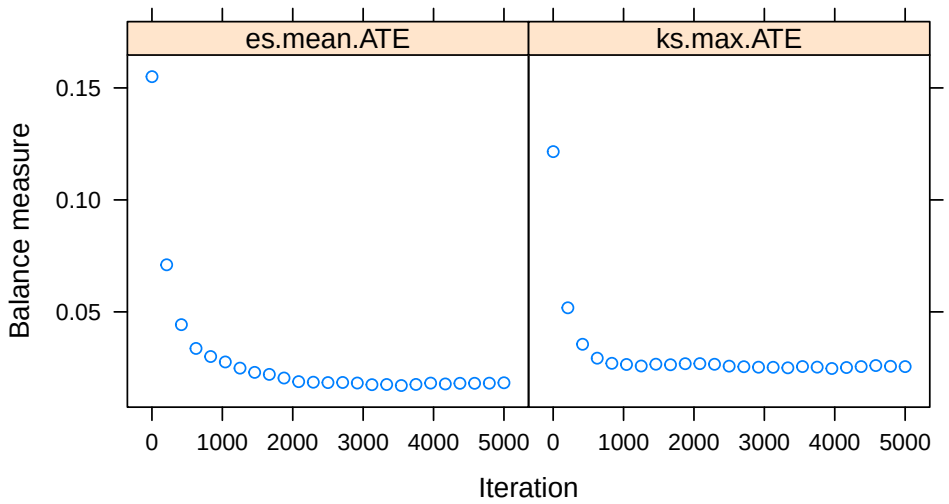
```
> egsingle.one$race <- as.factor(race <- ifelse(egsingle.one$black==1, 1,
+                                          ifelse(egsingle.one$hispanic==1, 2, 3)))
```

As discussed above, to use `ps()` to estimate nonresponse, we need to let respondents be the treatment group by modeling an indicator of response.

```
> egsingle.ps <-  ps(resp ~ race + female + size + lowinc + mobility,
+       data=egsingle.one,
+       stop.method=c("es.mean","ks.max"),
+       n.trees=5000,
+       verbose=FALSE,
+       estimand = "ATE")
```

As in standard propensity score applications, we should check that `n.trees` was set large enough, so that balance would seem not to be improved if more complex models were considered. Recall that for `es.mean.ATE` the measure is the average effect size difference between the two groups and for `ks.max.ATE` the measure is the largest of the KS statistics.

```
> plot(egsingle.ps)
```



By default the balance table generated by `ps()` compares the weighted treatment group (respondents) to the weighted comparison group (nonresponders) – both groups weighted to equal the overall population. However, the goal is to weight the respondents to match the population not to compare the weighted respondents and nonrespondents. The default balance table may be useful for evaluating the propensity scores, but it does not directly assess the quality of the weights for balancing the weighted respondents with the overall population.

We can "trick" the `dx.wts()` function in the `twang` package into making the desired comparison. We want to compare the weighted respondents to the unweighted full sample. When evaluating ATT weights we compare the weighted comparison group with the unweighted treatment group. If we apply `dx.wts()` to a data set where the "treatment" group is the entire `esingle.one` sample and the "control" group is the `esingle.one` respondents and the weights equal one for every student in the pseudo-treatment group and equal the weights from `ps()` for every student in the pseudo-control group, we can obtain the balance statistics we want.

We begin by setting up the data with the pseudo-treatment and control groups. We add ATE weights from the "ks.max" stopping rule as our nonresponse weights.

```
> egsingle.one$wgt <- get.weights(egsingle.ps, stop.method="ks.max")
```

We now stack the full sample and the respondents. The variable "nr2" is the pseudo-treatment indicator. We set it equal to one for the full sample and 0 for the respondents. Similarly, "wgt2" is the pseudo-ATT weight which is set equal to one for the full sample and equal to the nonresponse weights for the respondents.

```
> egtmp <- rbind(data.frame(egsingle.one, nr2=1, wgt2=1),
+                data.frame(egsingle.one, nr2=0, wgt2=egsingle.one$wgt)[egsingle.one$resp==1,])
```

We now run `dx.wts()` to obtain the balance statistics. Switching to ATT from ATE yields a warning that can be ignored in this case.

```
> egdxwts <- dx.wts(x=egtmp$wgt2,
+                   data=egtmp,
+                   estimand="ATT",
+                   vars=c("race", "female", "size",  "lowinc",  "mobility"),
+                   treat.var="nr2")

> pretty.tab<-bal.table(egdxwts)[[2]][,c("tx.mn","ct.mn","std.eff.sz","ks")]
> names(pretty.tab) <- c("OverallS Sample","Weighted responders","Std ES","KS")
> xtable(pretty.tab,
+         caption = "Balance of the nonrespondents and respondents",
+         label = "tab:balance2",
+         digits = c(0, 2, 2, 2, 2),
+         align=c("l","r","r","r","r"))
```

The balance is very good. We can use the weighted respondents for analysis. We select only the records with an observed outcome. This will be our analysis sample and the variable "wgt" will contains the nonreponse weights.

```
> egsinge.resp <- merge(subset(egsingle, subset=resp==1),
+                       subset(egsingle.one, subset=resp==1,
+                              select=c(childid, wgt)) )
```

	OverallS Sample	Weighted responders	Std ES	KS
race:1	0.69	0.69	0.01	0.00
race:2	0.14	0.14	0.01	0.00
race:3	0.16	0.17	-0.02	0.01
female:Female	0.49	0.49	0.01	0.01
female:Male	0.51	0.51	-0.01	0.01
size	755.89	756.53	-0.00	0.02
lowinc	78.17	78.51	-0.01	0.03
mobility	34.59	34.22	0.03	0.02

Table 6: Balance of the nonrespondents and respondents

5 The details of `twang`

5.1 Propensity scores and weighting

Propensity scores can be used to reweight comparison cases so that the distribution of their features match the distribution of features of the treatment cases, for ATT, or cases from both treatment and control groups to match each other, for ATE (Rosenbaum 1987, Wooldridge 2002, Hirano and Imbens 2001, McCaffrey *et al.* 2004) Let $f(\mathbf{x}|t=1)$ be the distribution of features for the treatment cases and $f(\mathbf{x}|t=0)$ be the distribution of features for the comparison cases. If treatments were randomized then we would expect these two distributions to be similar. When they differ for ATT we will construct a weight, $w(\mathbf{x})$, so that

$$f(\mathbf{x}|t=1) = w(\mathbf{x})f(\mathbf{x}|t=0). \tag{2}$$

For example, if $f(\text{age=}65, \text{sex=F}|t=1) = 0.10$ and $f(\text{age=}65, \text{sex=F}|t=0) = 0.05$ (i.e. 10% of the treatment cases and 5% of the comparison cases are 65 year old females) then we need to give a weight of 2.0 to every 65 year old female in the comparison group so that they have the same representation as in the treatment group. More generally, we can solve (2) for $w(\mathbf{x})$ and apply Bayes Theorem to the numerator and the denominator to give an expression for the propensity score weight for comparison cases,

$$w(\mathbf{x}) = K\frac{f(t=1|\mathbf{x})}{f(t=0|\mathbf{x})} = K\frac{P(t=1|\mathbf{x})}{1-P(t=1|\mathbf{x})}, \tag{3}$$

where K is a normalization constant that will cancel out in the outcomes analysis. Equation (3) indicates that if we assign a weight to comparison case i equal to the odds that a case with features $\mathbf{x}_i$ would be exposed to the treatment, then the distribution of their features would balance. Note that for comparison cases with features that are atypical of treatment cases, the propensity score $P(t=1|\mathbf{x})$ would be near 0 and would produce a weight near 0. On the other hand, comparison cases with features typical of the treatment cases would receive larger weights.

For ATE, each group is weighted to match the population. The weight must satisfy:

$$f(\mathbf{x}|t=1) = w(\mathbf{x})f(\mathbf{x}), \text{ and} \tag{4}$$

$$f(\mathbf{x}|t=0) = w(\mathbf{x})f(\mathbf{x}), \text{ and} \tag{5}$$

Again using Bayes Theorem we obtain $w(\mathbf{x}) \propto 1/f(t=1|\mathbf{x})$ for the treatment group and $w(\mathbf{x}) \propto 1/f(t=0|\mathbf{x})$ for the control group.

5.2 Estimating the propensity score

In randomized studies $P(t=1|\mathbf{x})$ is known and fixed in the study design. In observational studies the propensity score is unknown and must be estimated, but poor estimation of the propensity scores can cause

just as much of a problem for estimating treatment effects as poor regression modeling of the outcome. Linear logistic regression is the common method for estimating propensity scores, and can suffice for many problems. Linear logistic regression for propensity scores estimates the log-odds of a case being in the treatment given $\mathbf{x}$ as

$$\log \frac{P(t=1|\mathbf{x})}{1-P(t=1|\mathbf{x})} = \beta'\mathbf{x} \tag{6}$$

Usually, β is selected to maximize the logistic log-likelihood

$$\ell\beta = \frac{1}{n}\sum_{i=1}^{n} t_i\beta'\mathbf{x}_i - \log\left(1+\exp(\beta'\mathbf{x}_i)\right) \tag{7}$$

Maximizing (7) provides the maximum likelihood estimates of β. However, in an attempt to remove as much confounding as possible, observational studies often record data on a large number of potential confounders, many of which can be correlated with one another. Standard methods for fitting logistic regression models to such data with the iteratively reweighted least squares algorithm can be statistically and numerically unstable. To improve the propensity score estimates we might also wish to include non-linear effects and interactions in $\mathbf{x}$. The inclusion of such terms only increases the instability of the models.

One increasingly popular method for fitting models with numerous correlated variables is the lasso (least absolute subset selection and shrinkage operator) introduced in statistics in Tibshirani (1996). For logistic regression, lasso estimation replaces (7) with a version that penalizes the absolute magnitude of the coefficients

$$\ell\beta = \frac{1}{n}\sum_{i=1}^{n} t_i\beta'\mathbf{x}_i - \log\left(1+\exp(\beta'\mathbf{x}_i)\right) - \lambda\sum_{j=1}^{J}|\beta_j| \tag{8}$$

The second term on the right-hand side of the equation is the penalty term since it decreases the overall of $\ell\beta$ when there are coefficient that are large in absolute value. Setting $\lambda = 0$ returns the standard (and potentially unstable) logistic regression estimates of β. Setting λ to be very large essentially forces all of the β_j to be equal to 0 (the penalty excludes β_0). For a fixed value of λ the estimated $\hat{\beta}$ can have many coefficients exactly equal to 0, not just extremely small but precisely 0, and only the most powerful predictors of t will be non-zero. As a result the absolute penalty operates as a variable selection penalty. In practice, if we have several predictors of t that are highly correlated with each other, the lasso tends to include all of them in the model, shrink their coefficients toward 0, and produce a predictive model that utilizes all of the information in the covariates, producing a model with greater out-of-sample predictive performance than models fit using variable subset selection methods.

Our aim is to include as covariates all piecewise constant functions of the potential confounders and their interactions. That is, in $\mathbf{x}$ we will include indicator functions for continuous variables like $I(\text{age} < 15), I(\text{age} < 16), \ldots, I(\text{age} < 90)$, etc., for categorical variables like $I(\text{sex} = \text{male}), I(\text{prior MI} = \text{TRUE})$, and interactions among them like $I(\text{age} < 16)I(\text{sex} = \text{male})I(\text{prior MI} = \text{TRUE})$. This collection of basis functions spans a plausible set of propensity score functions, are computationally efficient, and are flat at the extremes of $\mathbf{x}$ reducing the likelihood of propensity score estimates near 0 and 1 that can occur with linear basis functions of $\mathbf{x}$. Theoretically with the lasso we can estimate the model in (8), selecting a λ small enough so that it will eliminate most of the irrelevant terms and yield a sparse model with only the most important main effects and interactions. Boosting (Friedman 2001, 2003, Ridgeway 1999) effectively implements this strategy using a computationally efficient method that Efron *et al.* (2004) showed is equivalent to optimizing (8). With boosting it is possible to maximize (8) for a range of values of λ with no additional computational effort than for a specific value of λ. We use boosted logistic regression as implemented in the generalized boosted modeling (`gbm`) package in R (Ridgeway 2005).

5.3 Evaluating the weights

As with regression analyses, propensity score methods cannot adjust for unmeasured covariates that are uncorrelated with the observed covariates. Nonetheless, the quality of the adjustment for the observed covariates achieved by propensity score weighting is easy to evaluate. The estimated propensity score weights should equalize the distributions of the cases' features as in (2). This implies that weighted statistics of the covariates of the comparison group should equal the same statistics for the treatment group. For example, the weighted average of the age of comparison cases should equal the average age of the treatment cases. To assess the quality of the propensity score weights one could compare a variety of statistics such as means, medians, variances, and Kolmogorov-Smirnov statistics for each covariate as well as interactions. The `twang` package provides both the standardized effect sizes and KS statistics and p-values testing for differences in the means and distributions of the covariates for analysts to use in assessing balance.

5.4 Analysis of outcomes

With propensity score analyses the final outcomes analysis is generally straightforward, while the propensity score estimation may require complex modeling. Once we have weights that equalize the distribution of features of treatment and control cases by reweighting. For ATT, we give each treatment case a weight of 1 and each comparison case a weight $w_i = p(\mathbf{x}_i)/(1 - p(\mathbf{x}_i))$. To estimate the ATE, we give control cases weight $w_i = 1/(1 - p(\mathbf{x}_i))$ and we give the treatment cases $w_i = 1/p(\mathbf{x}_i)$. We then estimate the treatment effect estimate with a weighted regression model that contains only a treatment indicator. No additional covariates are needed if the weights account for differences in $\mathbf{x}$.

A combination of propensity score weighting and covariate adjustment can be useful for several reasons. First, the propensity scores may not have been able to completely balance all of the covariates. The inclusion of these covariates in addition to the treatment indicator in a weighted regression model may correct this if the imbalance is relatively small. Second, in addition to exposure, the relationship between some of the covariates and the outcome may also be of interest. Their inclusion can provide coefficients that can estimate the direction and magnitude of the relationship. Third, as with randomized trials, stratifying on covariates that are highly correlated with the outcome can improve the precision of estimates. Lastly, the some treatment effect estimators that utilize an outcomes regression model and propensity scores are "doubly robust" in the sense that if either the propensity score model is correct or the regression model is correct then the treatment effect estimator will be unbiased (Bang & Robins 2005).

References

[1] Bang H. and J. Robins (2005). "Doubly robust estimation in missing data and causal inference models," *Biometrics* 61:692–972.

[2] Bland M. (2013). "Do baseline p-values follow a uniform distribution in randomised trials?" *PLoS ONE* 8(10): e76010: 1–5.

[3] Dehejia, R.H. and S. Wahba (1999). "Causal effects in nonexperimental studies: re-evaluating the evaluation of training programs," *Journal of the American Statistical Association* 94:1053–1062.

[4] Efron, B., T. Hastie, I. Johnstone, R. Tibshirani (2004). "Least angle regression," *Annals of Statistics* 32(2):407–499.

[5] Friedman, J.H. (2001). "Greedy function approximation: a gradient boosting machine," *Annals of Statistics* 29(5):1189–1232.

[6] Friedman, J.H. (2002). "Stochastic gradient boosting," *Computational Statistics and Data Analysis* 38(4):367–378.

[7] Friedman, J.H., T. Hastie, R. Tibshirani (2000). "Additive logistic regression: a statistical view of boosting," *Annals of Statistics* 28(2):337–374.

[8] Hastie, T., R. Tibshirani, and J. Friedman (2001). *The Elements of Statistical Learning*. Springer-Verlag, New York.

[9] Helmreich, J.E., and R.M. Pruzek (2009). "PSAgraphics: An R package to support propensity score analysis," *Journal of Statistical Software* 29(6):1–23.

[10] Hirano, K. and G. Imbens (2001). "Estimation of causal effects using propensity score weighting: An application to data on right heart catheterization," *Health Services and Outcomes Research Methodology* 2:259–278.

[11] Huppler-Hullsiek, K. and T. Louis (2002) "Propensity score modeling strategies for the causal analysis of observational data," *Biostatistics* 3:179–193.

[12] Lalonde, R. (1986). "Evaluating the econometric evaluations of training programs with experimental data," *American Economic Review* 76:604–620.

[13] Little, R. J. and S. Vartivarian (2004). "Does weighting for nonresponse increase the variance of survey means?" *ASA Proceedings of the Joint Statistical Meetings*, 3897-3904 American Statistical Association (Alexandria, VA) `http://www.bepress.com/cgi/viewcontent.cgi?article=1034&context=umichbiostat`.

[14] McCaffrey, D., G. Ridgeway, A. Morral (2004). "Propensity score estimation with boosted regression for evaluating adolescent substance abuse treatment," *Psychological Methods* 9(4):403–425.

[15] Obenchain, B. (2011). *USPS 1.2 package manual*. `http://cran.r-project.org/web/packages/USPS/USPS.pdf`

[16] Ridgeway, G. (1999). "The state of boosting," *Computing Science and Statistics* 31:172–181.

[17] Ridgeway, G. (2005). *GBM 1.5 package manual*. `http://cran.r-project.org/doc/packages/gbm.pdf`.

[18] Ridgeway, G. (2006). "Assessing the effect of race bias in post-traffic stop outcomes using propensity scores." *Journal of Quantitative Criminology* 22(1):1–29.

[19] Rosenbaum, P. and D. Rubin (1983). "The Central Role of the Propensity Score in Observational Studies for Causal Effects," *Biometrika* 70(1):41–55.

[20] Rosenbaum, P. (1987). "Model-based direct adjustment," *Journal of the American Statistical Association* 82:387–394.

[21] Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B* 58(1):267–288.

[22] Wooldridge, J. (2002). *Econometric analysis of cross section and panel data*, MIT Press, Cambridge.