PSAboot: An R Package for Bootstrapping Propensity Score Analysis

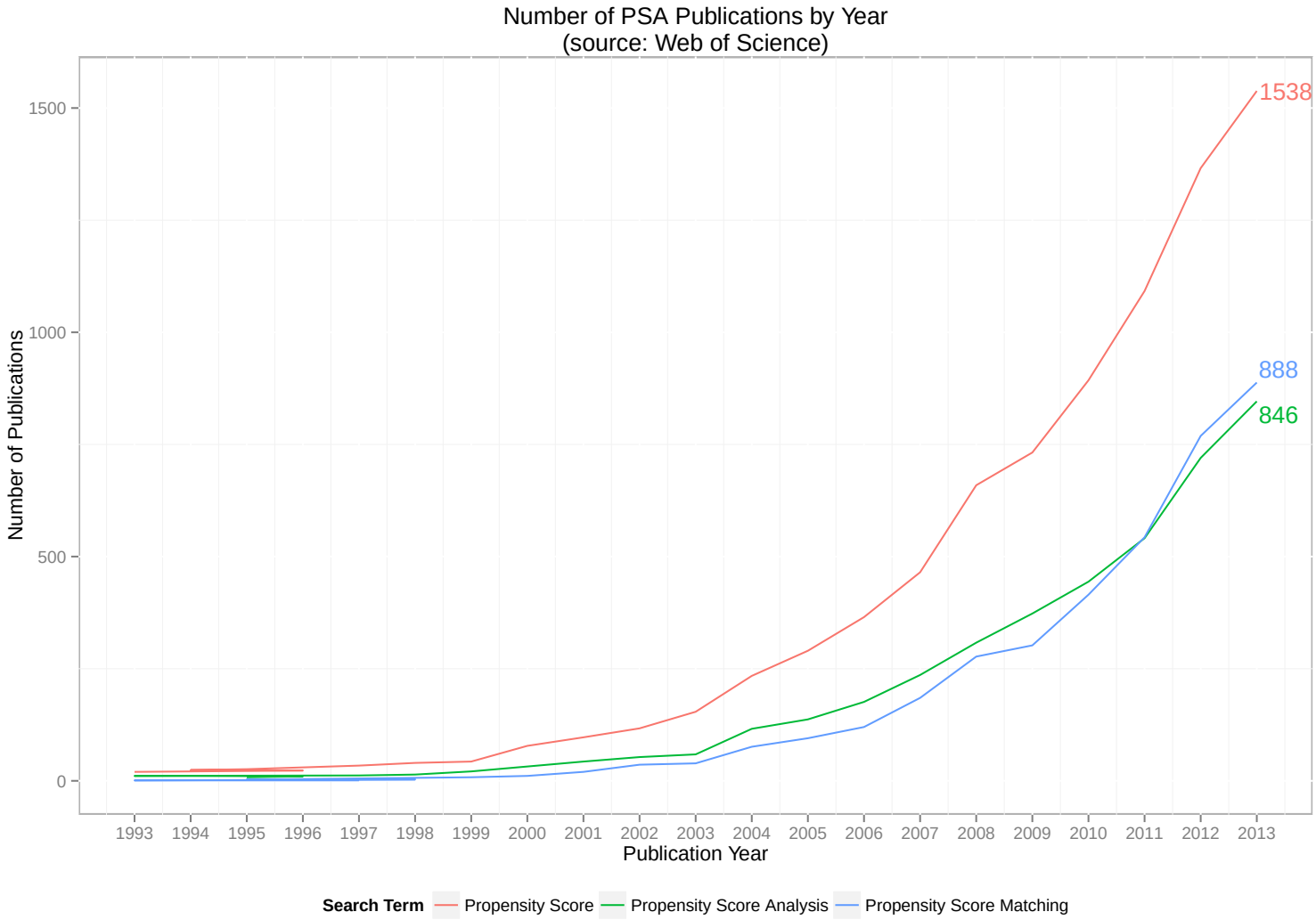
2014 useR! Conference

Jason M. Bryer

http://jason.bryer.org/PSAboot
jason@bryer.org

July 2, 2014

Popularity of Propensity Score Analysis



Propensity Score Analysis

- The use of propensity score methods for estimating causal effects has been increasing in the social sciences (Thoemmes & Kim, 2011) and in medical research (Austin, 2008) in the last decade.
- The goal of PSA is to adjust for selection bias in observational studies.
- PSA is conducted in two phases.
  - *Phase I* Calculate propensity scores using observed covariates then match or stratify rows using the propensity scores.
  - *Phase II* Compare the outcome of interest between matched pairs or within strata.

Propensity Score Analysis

- The use of propensity score methods for estimating causal effects has been increasing in the social sciences (Thoemmes & Kim, 2011) and in medical research (Austin, 2008) in the last decade.
- The goal of PSA is to adjust for selection bias in observational studies.
- PSA is conducted in two phases.
  - *Phase I* Calculate propensity scores using observed covariates then match or stratify rows using the propensity scores.
  - *Phase II* Compare the outcome of interest between matched pairs or within strata.
- The propensity score is the "conditional probability of assignment to a particular treatment given a vector of observed covariates" (Rosenbaum & Rubin, 1983, p. 41).

  $$\pi(X_i) \equiv Pr(T_i = 1|X_i)$$

  The balancing property under exogeneity:

  $$T_i \perp\!\!\!\perp X_i \mid \pi(X_i)$$

  We can then restate the ignorability assumption with the propensity score:

  $$(Y_i(1), Y_i(0)) \perp\!\!\!\perp T_i \mid \pi(X_i)$$

Materials for an introduction to PSA here: `https://github.com/jbryer/psa`

Propensity Score Methods

There are two broad approaches to conducting propensity score analysis:

- Matching - Involves finding matched pairs between the treatment and control groups. There are two R packages that implement matching:
  - `Matching` Jasjeet S. Sekhon (2011). Multivariate and Propensity Score Matching Software with Automated Balance Optimization: The Matching Package for R. *Journal of Statistical Software, 42*(7), 1-52. `http://www.jstatsoft.org/v42/i07/`
  - `MatchIt` Daniel E. Ho, Kosuke Imai, Gary King, Elizabeth A. Stuart (2011). MatchIt: Nonparametric Preprocessing for Parametric Causal Inference. *Journal of Statistical Software, 42*(8), pp. 1-28. `http://www.jstatsoft.org/v42/i08/`
- Stratification - Involves finding subgroups (i.e. strata) of similar treatment and control units. Some approaches include:
  - Stratifying on the propensity scores (e.g. quintiles)
  - Classification trees (e.g. `rpart`, `ctree`, Random Forests)

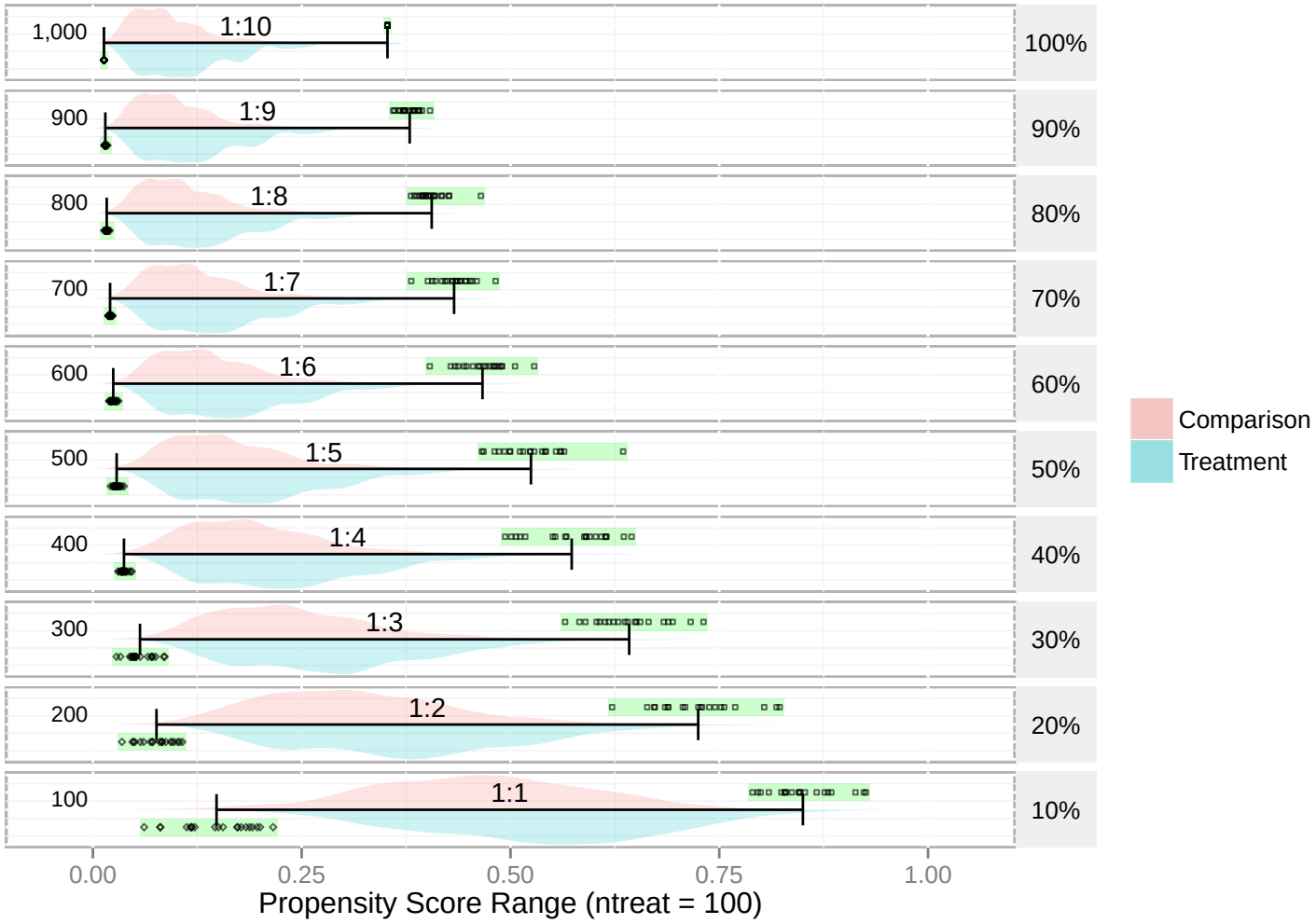
Why Bootstrap PSA?

The bootstrap is a great approach to get estimate standard errors.

It is good practice to use multiple propensity score methods. See Rosenbaum's (2012) paper, *Testing one hypothesis twice in observational studies*. The `PSAboot` will test the hypothesis Mxm times, where M is the number of bootstrap samples and m is the number of different PSA methods.

Many observational studies often have many more control units then treatment units. As the ratio of treatment-to-control increases, the range of propensity scores (i.e. fitted values from a logistic regression) tends to shrink. It may be appropriate to randomly select control units to decrease this ratio.

Propensity Score Ranges



More detailed description here:
`http://jason.bryer.org/multilevelPSA/psranges.html`

Bootstrapping Propensity Score Analysis

The `PSAboot` package/function will:

- Estimate the effects using the full dataset (i.e. the non-bootstrapped analysis).
- Draw M stratified bootstrap samples. Stratified on the treatment variable so that each bootstrap sample has the same number of treatment and control units.
- For each bootstrap sample, estimate the effect for each method (default is five methods).
- Evaluate the balance for each method and bootstrap sample combination.
- Provide an overall pooled estimate across all bootstrap samples.

Example: Tutoring

Students can opt to utilize tutoring services to supplement math courses. Of those who used tutoring services, approximately 58% of students used the tutoring service once, whereas the remaining 42% used it more than once. Outcome of interest is course grade.

Military	Active military status.
Income	Income level.
Employment	Employment level.
NativeEnglish	Is English their native language
EdLevelMother	Education level of their mother.
EdLevelFather	Education level of their father.
Ethnicity	American Indian or Alaska Native, Asian, Black or African American, Hispanic, Native Hawaiian or Other Pacific Islander, Two or more races, Unknown, White
Gender	Male, Female
Age	Age at course start.
GPA	Student GPA at the beginning of the course.

Installation

The PSAboot is available on CRAN.

```
> install.packages(c("PSAboot"), repos="http://cran.r-project.org")
```

Or the latest version can be installed from Github.

```
> devtools::install_github("jbryer/PSAboot")
```

Then load the package.

```
> require(PSAboot)
```

Data Preparation

```
> data(tutoring, package="TriMatch")

> tutoring$treatbool <- tutoring$treat != "Control"

> covs <- tutoring[,c("Gender", "Ethnicity", "Military", "ESL",
    "EdMother", "EdFather", "Age", "Employment",
    "Income", "Transfer", "GPA")]

> table(tutoring$treatbool)

FALSE  TRUE
  918   224
```

Bootstrapping PSA

```
> tutoring.boot <- PSAboot(Tr=tutoring$treatbool,
                           Y=tutoring$Grade,
                           X=covs, seed=2112)
100 bootstrap samples using 5 methods.
Bootstrap sample sizes:
   Treated=224 (100%) with replacement.
   Control=918 (100%) with replacement.
```

Bootstrapping PSA

```
> tutoring.boot <- PSAboot(Tr=tutoring$treatbool,
                           Y=tutoring$Grade,
                           X=covs, seed=2112)
100 bootstrap samples using 5 methods.
Bootstrap sample sizes:
   Treated=224 (100%) with replacement.
   Control=918 (100%) with replacement.


> class(tutoring.boot)
[1] "PSAboot"
> ls(tutoring.boot)
 [1] "M"                    "Tr"
 [3] "X"                    "Y"
 [5] "complete.details"     "complete.summary"
 [7] "control.replace"      "control.sample.size"
 [9] "pooled.details"       "pooled.summary"
[11] "seed"                 "treated.replace"
[13] "treated.sample.size"
```

PSAboot Parameters

Tr	numeric (0 or 1) or logical vector of treatment indicators.
Y	vector of outcome varaible.
X	matrix or data frame of covariates used to estimate the propensity scores.
M	number of bootstrap samples to generate (default is 100).
formu	formula used for estimating propensity scores. The default is to use all covariates in X.
control.ratio	the ratio of control units to sample relative to the treatment units.
control.sample.size	the size of each bootstrap sample of control units.
control.replace	whether to use replacement when sampling from control units.
treated.sample.size	the size of each bootstrap sample of treatment units. The default uses all treatment units for each boostrap sample.
treated.replace	whether to use replacement when sampling from treated units.
methods	a named vector of functions for each PSA method to use.
seed	random seed. Each iteration, i, will use a seed of seed + i.
parallel	whether to run the bootstrap samples in parallel.
...	other parameters passed to the PSA methods.

Summary

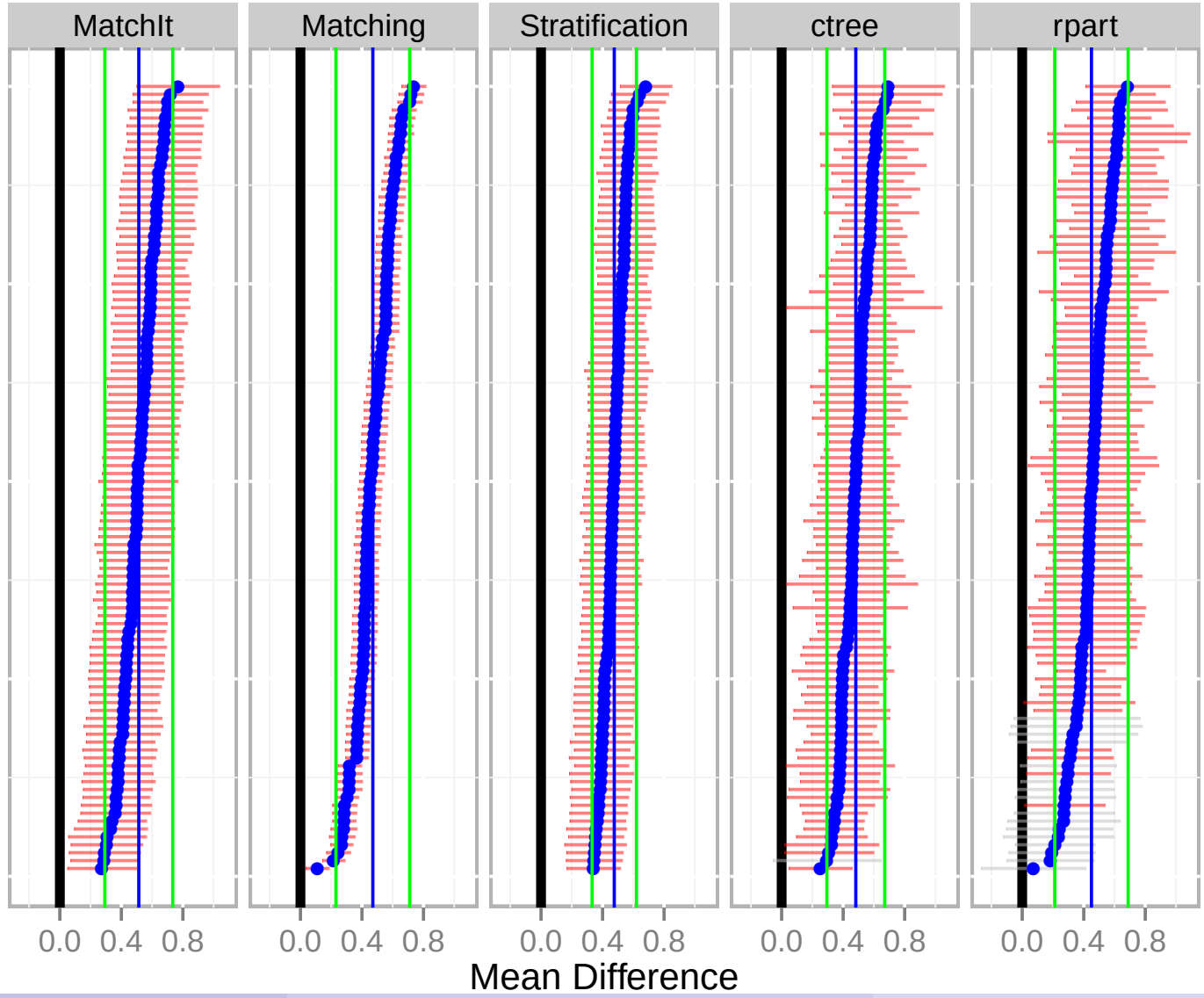
```
> summary(tutoring.boot)
Stratification Results:
   Complete estimate = 0.482
   Complete CI = [0.3, 0.665]
   Bootstrap pooled estimate = 0.476
   Bootstrap weighted pooled estimate = 0.475
   Bootstrap pooled CI = [0.332, 0.62]
   100% of bootstrap samples have confidence intervals that do not s
      100% positive.
      0% negative.
ctree Results:
   Complete estimate = 0.458
   Complete CI = [0.177, 0.739]
   Bootstrap pooled estimate = 0.482
   Bootstrap weighted pooled estimate = 0.477
   Bootstrap pooled CI = [0.294, 0.67]
   99% of bootstrap samples have confidence intervals that do not sp
      99% positive.
      0% negative.
```

Summary Data Frame

```
> as.data.frame(summary(tutoring.boot))
          method bootstrap.estimate bootstrap.ci.min
1 Stratification                0.48              0.33
2          ctree                0.48              0.29
3          rpart                0.45              0.21
4       Matching                0.47              0.23
5        MatchIt                0.51              0.29
  bootstrap.ci.max complete.estimate complete.ci.min
1             0.62                0.48              0.30
2             0.67                0.46              0.18
3             0.69                0.47              0.17
4             0.71                0.48              0.39
5             0.73                0.50              0.25
  complete.ci.max
1            0.67
2            0.74
3            0.78
4            0.57
5            0.75
```

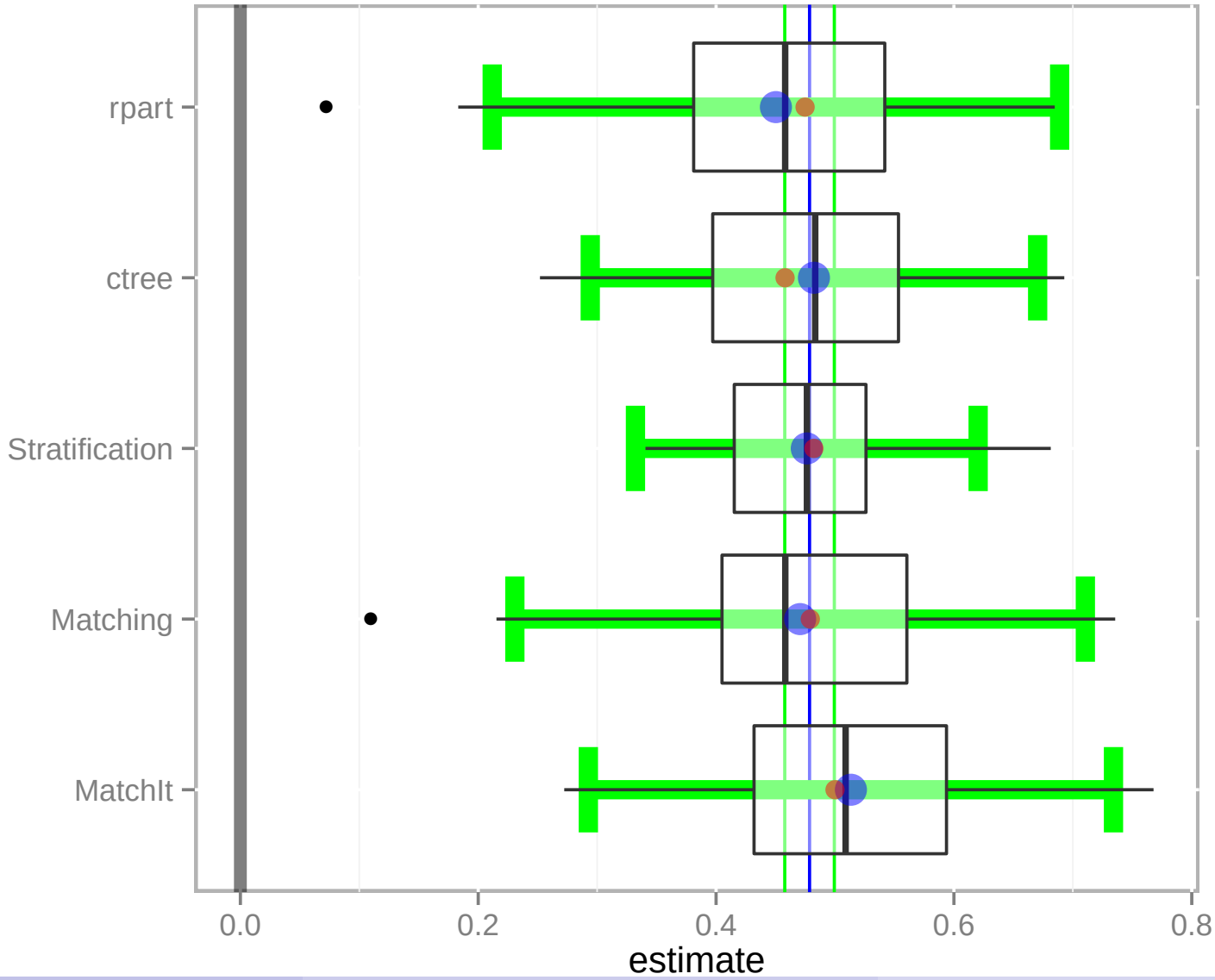
Plotting

```
> plot(tutoring.boot)
```



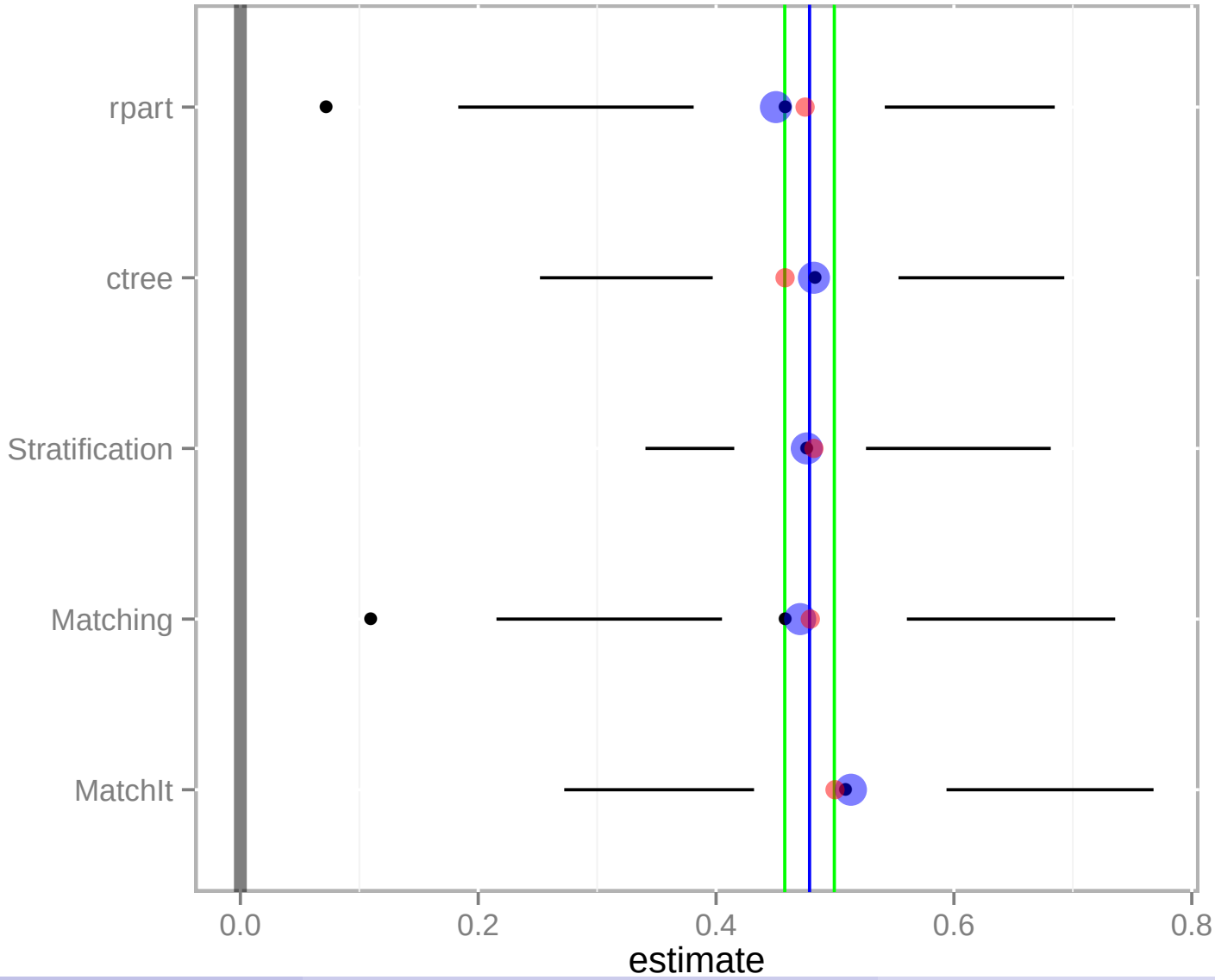
Boxplot

```
> boxplot(tutoring.boot)
```



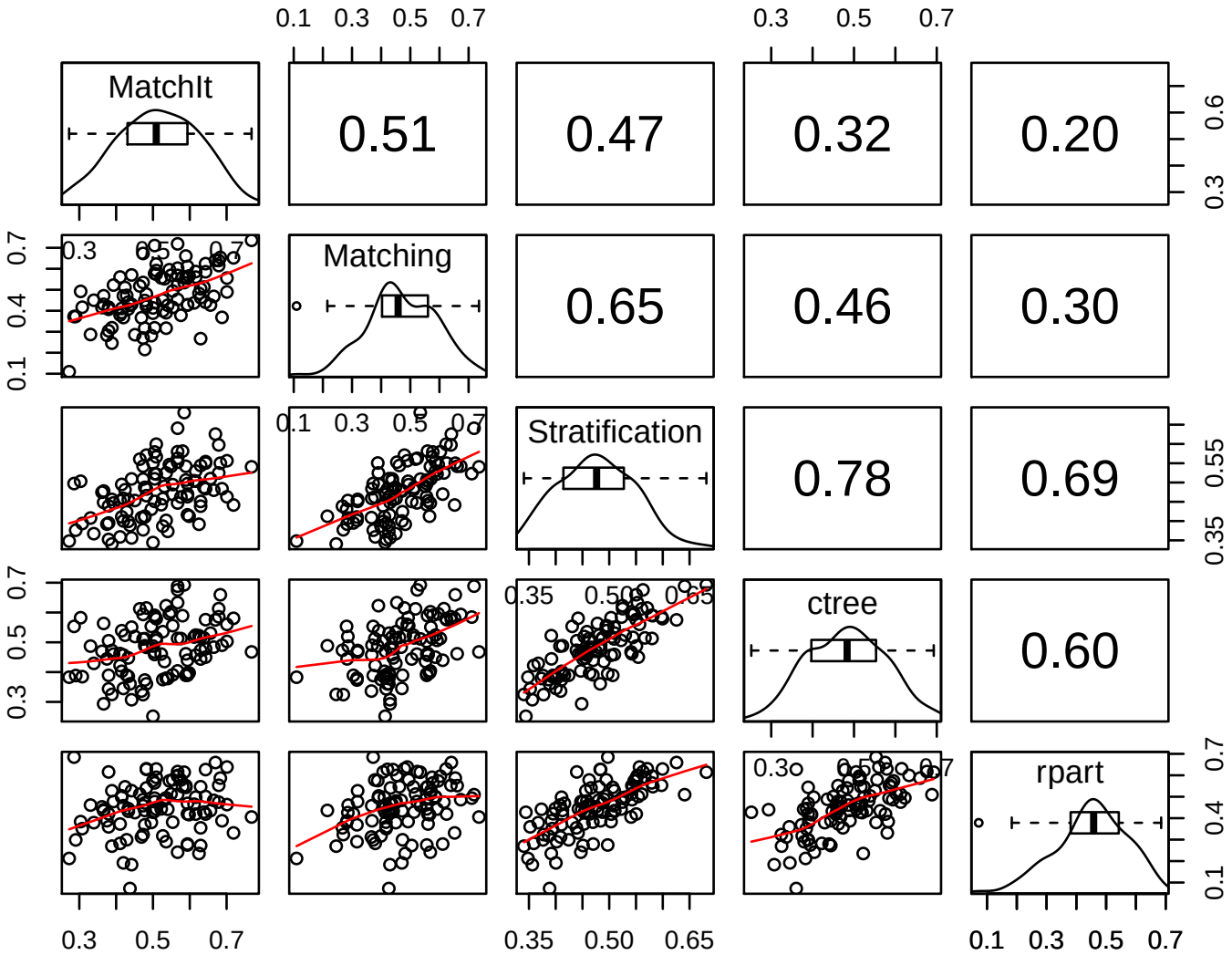
Boxplot (Tufte style)

```
> boxplot(tutoring.boot, tufte=TRUE, bootstrap.ci.size=NA)
```



Matrix Plot

```
> matrixplot(tutoring.boot)
```



Checking Balance

The estimates are only as good as the balance achieved!

```
> tutoring.bal <- balance(tutoring.boot)
> tutoring.bal

Unadjusted balance: 0.117875835338968
                Complete Bootstrap
Stratification     0.029     0.038
ctree              0.044     0.069
rpart              0.078     0.087
Matching           0.045     0.067
MatchIt            0.051     0.058
```

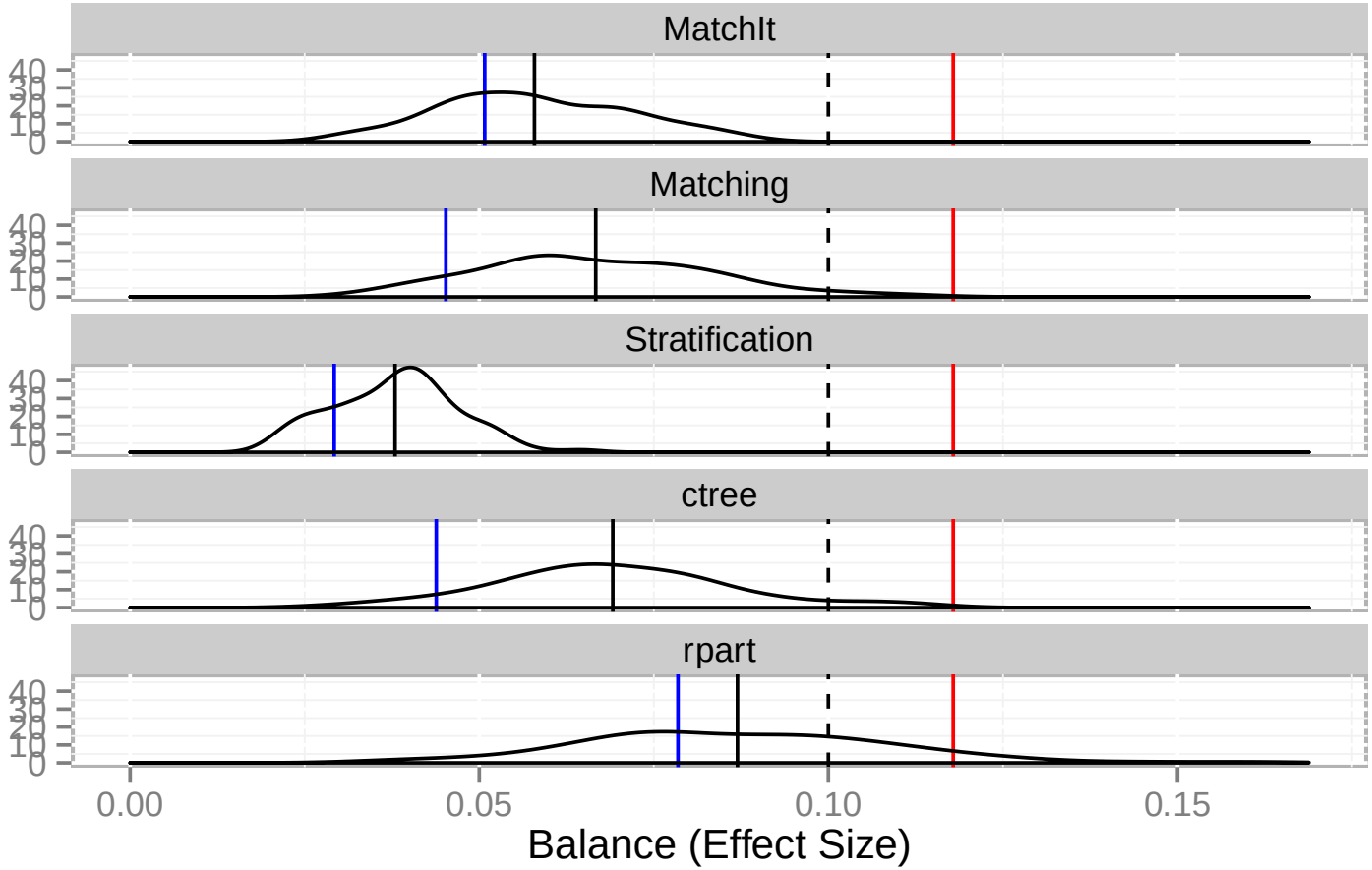
The `balance` function will calculate the standardized effect sizes for each covariate after adjustment. The `pool.fun` allows you to define how the balance statistics are combined. It defaults to `mean`, but other options include q25, q75, `median` or `max`.

```
> ls(tutoring.bal)

[1] "balances"   "complete"   "pool.fun"   "pooled"
[5] "unadjusted"
```

Checking Balance: Density Plots

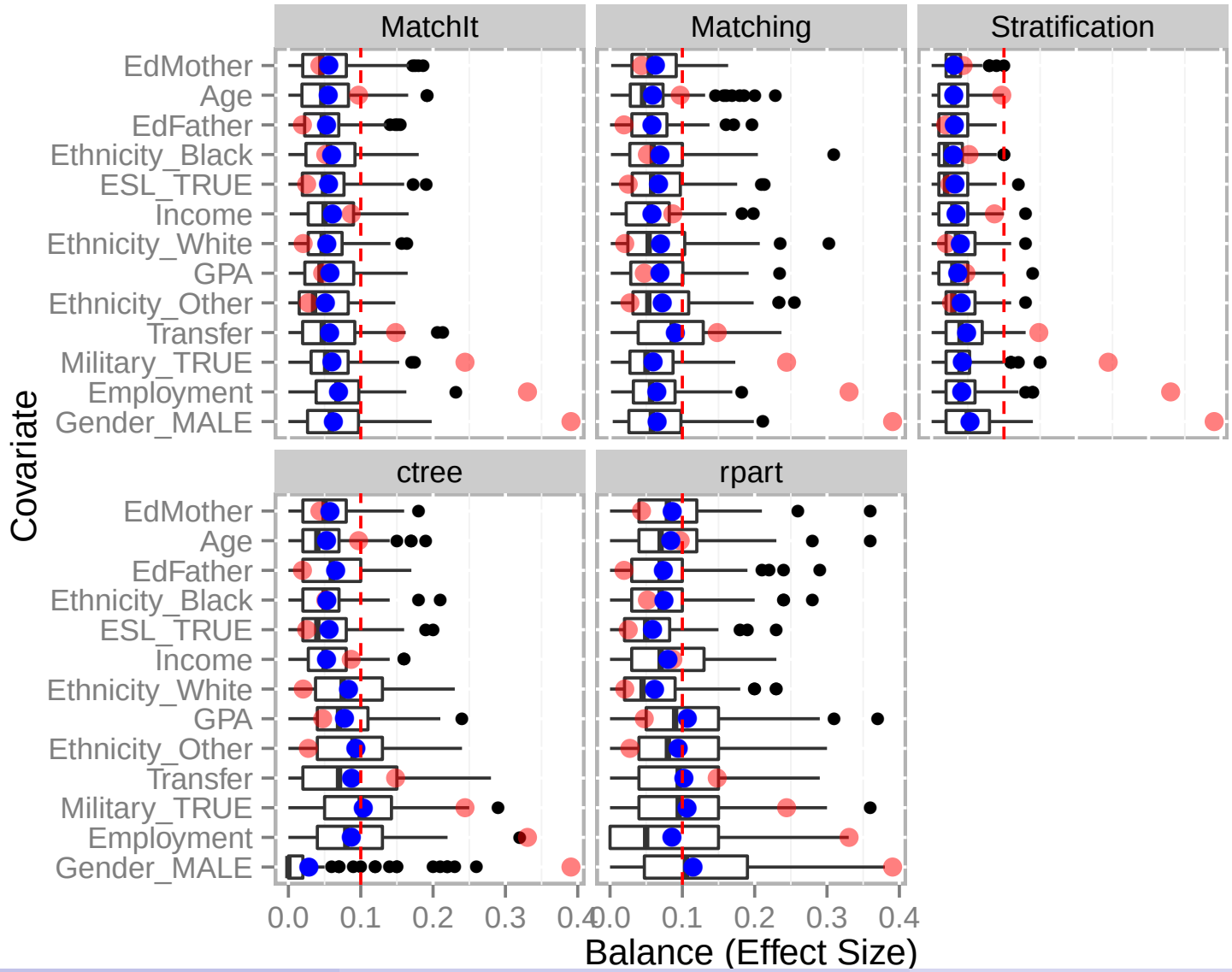
```
> plot(tutoring.bal) + geom_vline(xintercept=.1, linetype=2)
```



Red line is unadjusted balance; Blue line is the non-bootstrap balance; Black line is the pooled bootstrap balance.

Checking Balance: Boxplots

```
> boxplot(tutoring.bal) + geom_hline(yintercept=.1, color='red', li
```



Extending PSAboot for Other Methods

Define a function with the following parameters: `Tr` (vector of treatment indicators), `Y` (vector outcome measure), `X` (data frame of covariates), `X.trans` (numeric matrix with non-numeric variables dummy coded), `formu` (the formula used for estimating propensity scores, `...` (other parameters passed from the user).

```
> boot.matching.1to3 <- function(Tr, Y, X, X.trans, formu, ...) {
    return(boot.matching(Tr=Tr, Y=Y, X=X,
           X.trans=X.trans,
           formu=formu, M=3, ...))
  }
```

Extending PSAboot for Other Methods

Define a function with the following parameters: `Tr` (vector of treatment indicators), `Y` (vector outcome measure), `X` (data frame of covariates), `X.trans` (numeric matrix with non-numeric variables dummy coded), `formu` (the formula used for estimating propensity scores, `...` (other parameters passed from the user).

```
> boot.matching.1to3 <- function(Tr, Y, X, X.trans, formu, ...) {
    return(boot.matching(Tr=Tr, Y=Y, X=X,
           X.trans=X.trans,
           formu=formu, M=3, ...))
  }
```

```
> tutoring.boot <- PSAboot(Tr=tutoring$treatbool,
    Y=tutoring$Grade,
    X=covs,
    methods=c("Matcing-1-to-3"=boot.matching.1to3,
       getPSAbootMethods()))
```

The `getPSAbootMethods()` function returns a vector of the five default functions. Note that the name of each element in `methods` will be the name used in the figures.

Getting More Information

- Package Vignette
  `vignette("PSAboot")`
- Lalonde Demo
  `demo("PSAbootLalonde")`
- Tutoring Demo
  `demo("PSAbootTutoring")`
- Programme of International Student Assessment Demo
  `demo("PSAbootPISA")`

Functions Available

```
> ls("package:PSAboot")
 [1] "PSAboot"            "balance"            "balance.matching"
 [4] "balance.matchit"    "boot.ctree"         "boot.matching"
 [7] "boot.matchit"       "boot.rpart"         "boot.strata"
[10] "getPSAbootMethods" "matrixplot"         "psa.strata"
[13] "q25"                "q75"
```

Thank You

Jason Bryer (jason@bryer.org)
http://jason.bryer.org/PSAboot
http://github.com/jbryer/PSAboot