# Heterogeneous Treatment Effects: Estimation and Inference

November 20, 2015

## Contents

# 1 Introduction

The following documentation describes the R code for Machine Learning for Heterogeneous Causal Effects.
See: http://arxiv.org/abs/1504.01132 and http://arxiv.org/abs/1510.04342

# 2 Installing R and RStudio

## 2.1 About R and RStudio

R and Rstudio are free and open-source.

R is a statistical computing language, and RStudio is an IDE for
RStudio is a separate open-source project that brings many powerful coding tools together into an intuitive, easy-to-learn interface.R, like other programming languages, is extended (or developed) through user-written functions. An integrated development environment (IDE), such as RStudio, is designed to facilitate such work. In addition, unlike many other statistical software packages in which a graphical user interface is employed, a typical user interacts with R primarily through the command line.

You should install R before installing RStudio.

## 2.2 Installing R

1. Go to http://cran.r-project.org/mirrors.html.
The R installation files are distributed by the Comprehensive R Archive Network (CRAN). CRAN is a collection of sites which carry identical materials and were created as mirror sites to lessen the load on any one server.

2. Click on one of the links, such as http://cran.stat.ucla.edu/.

3. In the 'Download and Install R' box, click on the link for your operating system.

## 2.3 Installing RStudio

RStudio runs in all major platforms (Windows, Mac, Linux) and through a web browser (using the server installation).

1. Go to: https://www.rstudio.com/products/rstudio/download/
The RStudio package is available for Download here.

2. Click the Download RStudio Desktop button.

There is a choice between a Desktop version and a Server version. The Desktop version is appropriate for single-user use.

3. Select the installation file for your system.
The files come in a common format for binary installation (e.g., exe, dmg, deb, or rpm).

4. Run the installation file.

*It is also possible to also install RStudio from its source code. A link for the source tarball for the current stable version appears on the appropriate download page. For the adventurous, the latest development build files are available from https://github.com/rstudio/rstudio. Installation details are in the INSTALL file accompanying the source code. *

# 3 Installing Packages

## 3.1 Installing Basic R Packages

To install basic R Packages, first open RStudio. This can either be done in your command line, or at the top of the script you are executing.

```
> install.packages("ggplot2")
```

Each time you open RStudio again, you will need to load the libraries you downloaded, using library(). Note that to install packages, you put the package name in quotations (i.e. "plyr") but to load a package after installing it, you remove the quotations.

```
> library(ggplot2)
```

The package should then install easily with this command *unless* you are using linux and don't have root access. You will be asked to select your local mirror, i.e. which CRAN server should you use to download the package.

### 3.1.1 Installing packages without root access

First, you need to designate a directory where you will store the downloaded packages. On my machine, I use the directory /data/Rpackages/ After creating a package directory, to install a package we use the command:

```
> install.packages("ggplot2", lib="/data/Rpackages/")
> library(ggplot2, lib.loc="/data/Rpackages/")
```

It's a bit of a pain having to type /data/Rpackages/ all the time. To avoid this burden, we create a file .Renviron in our home area, and add the line R_LIBS=/data/Rpackages/ to it. This means that whenever you start R, the directory /data/Rpackages/ is added to the list of places to look for R packages and so:

```
> install.packages(”ggplot2”)
> library(ggplot2)
```

## 3.2 Installing Packages from Github

The devtools package contains a command for installing packages directly from github. This command attempts to install packages from a github repository. See http://www.rdocumentation.org/packages/devtools/functions/install_github for Further Documentation on this command.

Be sure to install devtools, rpart, and rpart.plot before installing the packages from github. See Figure 1 for example code for guidance on the packages necessary for working with this code, and the order of which to download them.

### 3.2.1 Installing Public Packages from Github

Once you have followed the above steps, you can install public packages via github in the command line, as is done in this example.

```
install_github(”swager/randomForestCI””
```

### 3.2.2 Installing Private Packages from Github

For private packages, you will need to create a github account here: and follow these instructions in order to create a Personal Access Token: https://help.github.com/articles/creating-an-access-token-for-command-line-use/. This token will allow you to identify yourself without putting your github password in your code. Then, you will need to email us with your github username, to request access to the private github, before your access token will work.

If you are interested in installing a particular branch of the repository. Otherwise, install_github defaults to master.

```
install_github(”swager/causalForest”,
               auth_token = ”11961a96f08678d7fb8850efd691e3fb5dca2d700”)

install_github(”swager/causalForest”, ref=”master2”,
               auth_token = ”11961a96f08678d7fb8850efd691e3fb5dca2d700”)
```

# 4 Preparing Your Data

If there are missing values in your data, causalForest will throw the error: ”There are missing values in the input.” To adapt to this, you can replace missing values with the column mean, and add a dummy variable (a new column in your dataframe) that is 1 if the value is missing for that observation, and 0 if it is not.

Your data should have a column for W, the treatment for your experiment, a column for Y, the outcome, and columns with the covariates.
W should be a binary variable, and Y can be either binary or continuous, depending on the structure of your experiment. If your data includes factor variables, create binary variables from these factor variables, as causalForest. You could also potentially convert these to continuous variables, but the former will likely have better results.

For the causalTree estimation, split your data into three sections, A, B, and C, where the percentages are 45%, 45%, 10% respectively. A is the dataset for building the tree, B is for estimating the leaves, and C is the test dataset, upon which you do your evaluation.

For the causalForest estimation, your dataset needs only to be split into two sets, training (90% of your sample) and testing, as the causal Forest function imposes honesty by taking two subsamples for each tree, one to build the tree, and one to estimate treatment effects. So, for this function, you can combine the A and B samples using rbind, to make one training set.

For the causalForest and predict function (for a causal Forest model), the covariates must be have integer variable names. (i.e. 1, 2, 3, 4, 5...etc.). To do this, rename the columns temporarily, before running causalForest, as is done in the example code.

# 5 Basic Packages

## 5.1 rpart.plot

https://cran.r-project.org/web/packages/rpart.plot/rpart.plot.pdf

## 5.2 rattle

https://cran.r-project.org/web/packages/rattle/rattle.pdf
http://www.inside-r.org/packages/cran/rattle/docs/fancyRpartPlot
This will allow you to plot your trees.

```
fancyRpartPlot(model, main="", sub, ...)
```

**Arguments**

- model=an rpart object
- main=title for the plot
- sub= sub title for the plot. The default is a Rattle string with date, time and username . . .

### 5.3 devtools

http://www.inside-r.org/packages/cran/devtools/docs/install_github

### 5.4 rpart

https://cran.r-project.org/web/packages/rpart/rpart.pdf
Recursive partitioning and regression trees

### 5.5 randomForest

https://github.com/swager/randomForest
https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

```
randomForest(data, y, keep.inbag=TRUE, ntree=10, replace=TRUE)
```

**Arguments**

- data=an optional data frame containing the variables in the model. By default the variables are taken from the environment which randomForest is called from.
- y=a response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, randomForest will run in unsupervised mode.
- keep.inbag=Should an n by ntree matrix be returned that keeps track of which samples are in-bag in which trees (but not how many times, if sampling with replacement)
- ntree= number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
- replace=Should sampling of cases be done with or without replacement?

### 5.6 ggplot2

http://ggplot2.org/
You can also use ggplot2's plot function to plot your decision trees.

## 6 Our Packages

### 6.1 randomForestCI

https://github.com/swager/randomForestCI
This is a public package installed from github. This package estimates the confidence interval for random forest estimates. It also works for causalForest,

which is a randomForest of causal trees. You can use this function in order to get get a confidence interval on the estimated treatment effects produced by causalForest.

```
randomForestInfJack(rf, newdata, calibrate = TRUE)
returns a two column dataframe, where the first column is the estimated
treatment effect, and the second column is the corresponding variance
```

**Arguments**

- rf= random forest trained with replace = TRUE and keep.inbag = TRUE
- newdata= a set of test points at which to evaluate standard errors
- calibrate whether to apply calibration to mitigate Monte Carlo noise warning: if calibrate = FALSE, some variance estimates may be negative due to Monte Carlo effects if the number of trees in rf is too small

## 6.2 causalForest

https://github.com/swager/causalForest
This package contains causalTree, causalForest, and functions to create honest trees with different methods.

```
causalTree(formula, data, weights, treatment, subset,
   na.action = na.causalTree, method = "anova", split.option, cv.option,
   minsize = 2L, model = FALSE, x = FALSE, y = TRUE, parms,
   p, control, alpha = 0.5, cost, ...))
```

**Arguments**

- formula= is in the format outcome covariate1+covariate2+covariate3, etc.. See below for further tips on this.
- data = specifies the data frame of the training data
- weights=
- treatment=vector of treatments for each of the observations in the training dataset
- minsize= minimum number of control and treated cases in a leaf -¿the default is 2.
- split.option= CT or TOT, splitting rule
- cv.option= cross validation option, TOT or matching
- cp= complexity parameter: the main role of this parameter is to save computing time by pruning splits that are obviously not worthwhile.

- p=propensity score

There are two ways to create the formula object, to be passed into causalTree. One way is to write the formula inline as the first parameter.

```
causalTree(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=A, treatment=A$w,
          method=”anova”, parms=1, cp=0, minsize = 10,
          cv.option = ”matching”, p = 0.5)
```

Another option is to create the formula outside of the function, and then pass it into the causalTree function. For example, suppose you have the list of the covariate names in a vector names covariate.names and wanted to create a linear model using all of your covariates. You could create a string as such:

```
sumx= paste(covariates.names, collapse=”+”)
linear<-paste(”y”,paste(”w”,sumx, sep=”+”), sep=”~”)
```

```
This will result in this string ”y~w+x1+x2+x3+x4+x5+x6+x7+x8+x9+x10”
```

Then, convert the string into a formula object:

```
linear.formula<-as.formula(linear)
```

And pass the formula to the causalTree function similarly to how it is done above:

```
linear.caustree<-causalTree(linear, data=A, treatment=A$w, method = ”anova”,
control=rpart.control(cp = 0, xval = 10), parms = 1,minsize = 10,
cv.option = ”matching”, p = 0.5)
```

The causal tree partitions the data into subgroups, in the form of a hierachical tree structure, and estimates a treatment effect for each of these subgroups. These subgroups are the leaves of the trees, or the nodes at the bottom which have no children. We can view the estimates within these nodes by plotting, but we can also use the method used in the example code. This method is helpful because it allows us to view the standard errors.

After creating the tree, we can do estimates on our A and B sets. Below is an example of first the model applied to the training sample, and then to the B (leaf estimating) sample. The important results are the coefficients in which the leaf is interacting with the treatment. These rows give us the treatment effect (in column ”Estimate”) and the standard error for those estimates (in column ”Std. Error”). Note that for the training sample, the model applied to the training data, the treatment effect in for the given leaf, corresponds to the name of the leaf. This is because the estimates for the training dataset are exact. This is a good check to make sure that everything is working properly.

```
Call:
lm(formula = y ~ -1 + leavesf + leavesf * w - w, data = sampleA)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-1052.78  -116.90    -2.84   130.58  1044.28

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
leavesf-1201.8515    -303.13      55.32  -5.480 7.01e-08 ***
leavesf-959.5196     -468.96      71.42  -6.567 1.38e-10 ***
leavesf-674.875      -294.26      78.23  -3.761 0.000191 ***
leavesf-480.7832     -125.76      61.85  -2.033 0.042579 *
leavesf-440.4048     -249.40      56.76  -4.394 1.38e-05 ***
leavesf-411.8558     -132.08      68.62  -1.925 0.054850 .
leavesf-196.074        93.46      68.62   1.362 0.173814
leavesf-142.4067     -136.75      78.23  -1.748 0.081121 .
leavesf-43.5154        51.64      71.42   0.723 0.470025
leavesf42.5301        -67.93      58.31  -1.165 0.244643
leavesf137.804         51.93      71.42   0.727 0.467498
leavesf185.3885       279.61      78.23   3.574 0.000388 ***
leavesf347.7434       101.22      68.62   1.475 0.140855
leavesf389.7945       183.95      74.59   2.466 0.014019 *
leavesf557.9918       159.63      78.23   2.040 0.041875 *
leavesf666.643        382.92      71.42   5.362 1.30e-07 ***
leavesf763.4154       185.15      53.99   3.430 0.000658 ***
leavesf1027.2793      578.50      58.31   9.921  < 2e-16 ***
leavesf-1201.8515:w -1201.85      82.98 -14.484  < 2e-16 ***
leavesf-959.5196:w   -959.52     103.27  -9.291  < 2e-16 ***
leavesf-674.875:w    -674.87     102.43  -6.589 1.21e-10 ***
leavesf-480.7832:w   -480.78      99.73  -4.821 1.94e-06 ***
leavesf-440.4048:w   -440.40      74.67  -5.898 7.09e-09 ***
leavesf-411.8558:w   -411.86     101.35  -4.064 5.67e-05 ***
leavesf-196.074:w    -196.07      87.31  -2.246 0.025187 *
leavesf-142.4067:w   -142.41     110.64  -1.287 0.198690
leavesf-43.5154:w     -43.52     105.93  -0.411 0.681409
leavesf42.5301:w       42.53      78.63   0.541 0.588832
leavesf137.804:w      137.80      97.33   1.416 0.157471
leavesf185.3885:w     185.39     110.64   1.676 0.094487 .
leavesf347.7434:w     347.74      92.38   3.764 0.000188 ***
leavesf389.7945:w     389.79     105.49   3.695 0.000246 ***
leavesf557.9918:w     557.99     110.64   5.043 6.57e-07 ***
leavesf666.643:w      666.64      99.04   6.731 4.98e-11 ***
leavesf763.4154:w     763.42      83.64   9.128  < 2e-16 ***
leavesf1027.2793:w   1027.28      97.57  10.528  < 2e-16 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 247.4 on 464 degrees of freedom
```

```
Multiple R-squared:  0.8624,Adjusted R-squared:  0.8517
F-statistic: 80.75 on 36 and 464 DF,  p-value: < 2.2e-16
```

And, the B dataset

```
Call:
lm(formula = y ~ -1 + leavesf + leavesf * w - w, data = sampleB)

Residuals:
     Min       1Q   Median       3Q      Max
-1013.42  -112.25   -10.52   122.05  1143.95

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
leavesf-1201.8515    -345.52      78.92  -4.378 1.48e-05 ***
leavesf-959.5196     -558.47      91.12  -6.129 1.89e-09 ***
leavesf-674.875      -291.45      75.82  -3.844 0.000138 ***
leavesf-480.7832     -219.98      82.42  -2.669 0.007878 **
leavesf-440.4048     -246.44      49.91  -4.938 1.11e-06 ***
leavesf-411.8558     -129.47      78.92  -1.641 0.101562
leavesf-196.074        38.96      57.00   0.683 0.494635
leavesf-142.4067     -168.11      91.12  -1.845 0.065697 .
leavesf-43.5154        69.10      96.65   0.715 0.475007
leavesf42.5301        -59.73      62.72  -0.952 0.341375
leavesf137.804         25.96      75.82   0.342 0.732237
leavesf185.3885       300.70      86.45   3.478 0.000552 ***
leavesf347.7434       124.29      64.43   1.929 0.054341 .
leavesf389.7945       267.46      91.12   2.935 0.003500 **
leavesf557.9918       277.69      82.42   3.369 0.000817 ***
leavesf666.643        283.46      68.34   4.148 4.00e-05 ***
leavesf763.4154       180.26      68.34   2.638 0.008630 **
leavesf1027.2793      496.95      82.42   6.029 3.37e-09 ***
leavesf-1201.8515:w  -973.30      98.10  -9.921  < 2e-16 ***
leavesf-959.5196:w   -746.64     122.87  -6.077 2.56e-09 ***
leavesf-674.875:w    -694.66     109.44  -6.348 5.21e-10 ***
leavesf-480.7832:w   -210.82     127.02  -1.660 0.097654 .
leavesf-440.4048:w   -356.18      73.25  -4.863 1.59e-06 ***
leavesf-411.8558:w   -136.00     114.11  -1.192 0.233945
leavesf-196.074:w     144.34      77.57   1.861 0.063408 .
leavesf-142.4067:w   -244.05     125.61  -1.943 0.052619 .
leavesf-43.5154:w     -97.37     147.64  -0.660 0.509895
leavesf42.5301:w      -69.65      86.56  -0.805 0.421445
leavesf137.804:w       39.76     103.59   0.384 0.701292
leavesf185.3885:w     185.41     149.73   1.238 0.216234
leavesf347.7434:w     272.98      95.57   2.856 0.004479 **
leavesf389.7945:w     243.56     118.54   2.055 0.040470 *
```

```
leavesf557.9918:w     473.58     127.02   3.728 0.000217 ***
leavesf666.643:w      783.68     104.40   7.507 3.12e-13 ***
leavesf763.4154:w     426.38      98.25   4.340 1.75e-05 ***
leavesf1027.2793:w    819.31     111.99   7.316 1.13e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 273.4 on 464 degrees of freedom
Multiple R-squared:  0.8143,Adjusted R-squared:  0.7999
F-statistic: 56.52 on 36 and 464 DF,  p-value: < 2.2e-16
```

The following functions use three different methods to create honest trees based upon the the second sample. The causal tree function can make predictions on the B sample, but these functions will build and return a new tree. Honest TOT and Honest ST are for comparison to the CT model. The TOT and CT models accept a causal tree object, while the ST model expects an rpart object.

```
honestST(object, data, na.action = na.rpart)
```

**Arguments**

- object=an rpart model
- data=leaf estimation dataset

```
honestTOT(object, data, treatment, p = 0.5, na.action = na.causalTree)
```

**Arguments**

- object=a fitted causal tree object
- data=leaf estimation dataset
- treatment=vector of treatments that corresponds to the leaf estimation dataset
- p=propensity

```
honestCTree(object, data, treatment, na.action = na.causalTree)
```

- object=a fitted causal tree object
- data=leaf estimation dataset
- treatment=vector of treatments that corresponds to the leaf estimation dataset

```
causalForest(X, Y, W, num.trees, sample.size = floor(length(Y) / 10),
                      mtry = ceiling(ncol(X)/3), nodesize = 1)
```

This function builds a causal forest for estimating heterogeneous treatment effects that extends Breiman's widely used random forest algorithm. The random forest produced by this function is composed of honest causal trees, which are split based on the treatment effects. This function handles both building the tree, and estimating the leaves, by taking two subsamples for each tree, one to build the tree and one to estimate treatment effects.

**Arguments**

- X=data frame with the covariates
- Y=corresponding outcomes for the observations in X
- W= corresponding treatment for the observations in X
- num.trees=number of causal trees to build the forest with
- sample.size=size of sample

In yet-to-be-committed versions of the causalForest code, the code will allow the user to select the subsample fraction and the size of the tree-building and leaf-estimating samples.
Simulations showed that using 50% of the training sample for subsampling, and splitting that in half for tree-building and leaf-estimating, worked well across a range of simulations. For example, in the example.R file, we have set the sample size to 225, because there are 450 observations in training dataset.

```
predict(forest, newdata, predict.all = FALSE)
```

**Arguments**

- forest=the fitted causalForest object
- newdata=the new test points at which the causalForest predictions are to be evaluated

The Causal Forest function returns a forest model. The predict function uses this fitted model to return estimates for tau, corresponding to each row of newdata. There are no standard ways to visualize random forest output, but some ideas include plotting the average predictions by deciles or quintiles of covariate(s) or using plotmo in order to plot the response as the predictors vary.

## 7 Demo

See the attached file: example.R, to see examples of each of the functions run with a simulated dataset.