TUTORIAL
Contrast Trees and Distribution Boosting in R

Jerome H. Friedman

May 19, 2020

1 Introduction

This tutorial describes the R package conTree that implements the contrast tree and distribution boosting procedures described in Friedman (2020). A brief summary of the methodology is provided. This is followed by demonstrations of the application of some of the package features on data examples. A more detailed description of all procedures is included with the full package documentation. Some familiarity with Friedman (2020) and the statistical package R is assumed. R can be downloaded from https://www.r-project.org.

1.1 Contrast trees

Contrast trees are used to assess the accuracy of many types of machine learning estimates that are not amenable to standard validation techniques. These include properties of the conditional distribution $p_y(y \,|\, \mathbf{x})$ (means, quantiles, complete distribution) as functions of $\mathbf{x}$. Given a set of predictor variables $\mathbf{x} = (x_1, x_2, \cdots, x_p)$ and two outcome variables y and z associated with each $\mathbf{x}$, a contrast tree attempts to partition the space of $\mathbf{x}$ values into local regions within which the respective distributions of $y \,|\, \mathbf{x}$ and $z \,|\, \mathbf{x}$, or selected properties of those distributions such as means or quantiles, are most different.

The outcomes y and z can be different functions of $\mathbf{x}$, $y = f(\mathbf{x})$ and $z = g(\mathbf{x})$, such as predictions produced by two different learning algorithms. The goal of the contrast tree is then to identify regions in $\mathbf{x}$ - space where the two predictions most differ. In other cases the outcome y may be observations of a random variable assumed to be drawn from some distribution at $\mathbf{x}$, $y \sim p_y(y \,|\, \mathbf{x})$. The quantity z might be an estimate for some property of that distribution such as its estimated mean $\hat{E}(y \,|\, \mathbf{x})$ or p-th quantile $\hat{Q}_p(y \,|\, \mathbf{x})$ as a function of $\mathbf{x}$. One would like to identify $\mathbf{x}$ - regions where the estimates z appear to be the least compatible with the actual empirical distribution of y within the region. Alternatively z itself could also be a random variable independent of y (given $\mathbf{x}$) with distribution $p_z(z \,|\, \mathbf{x})$ and interest is in identifying regions of $\mathbf{x}$ - space where the two distributions $p_y(y \,|\, \mathbf{x})$ and $p_z(z \,|\, \mathbf{x})$ most differ.

Contrast trees can serve as a lack-of-fit measure. If the procedure is successful in finding local $\mathbf{x}$ - regions where the empirical distribution of y is inconsistent with the model predictions z, lack-of-fit of the model to the data in those regions is indicated. A measure of such lack-of-fit is the size of the discrepancy.

1.2 Contrast boosting

If the regions $\{R_m^{(1)}\}_1^M$ produced by a contrast tree uncover lack-of-fit, boosted contrast trees can often improve prediction accuracy. The predictions z in each separate region $R_m^{(1)}$ of the tree can be modified $z \to z^{(1)} = z + \delta_m^{(1)}$ $(\mathbf{x} \in R_m^{(1)})$ so as to obtain reduced discrepancy between y and $z^{(1)}$ in the region, thereby improving average discrepancy over all regions. The modified $z^{(1)}$ - values can then be contrasted with y using another contrast tree $\{R_m^{(2)}\}_1^M$ with updates

$z^{(2)} = z^{(1)} + \delta_m^{(2)}$ $(\mathbf{x} \in R_m^{(2)})$. These in turn can be contrasted with y to produce new regions $\{R_m^{(3)}\}_1^M$ and corresponding updates $\{\delta_m^{(3)}\}_1^M$. Such iterations can be continued K times until the updates become small. Each tree k in the boosted sequence $1 \leq k \leq K$ partitions the $\mathbf{x}$ - space into a set of regions $\{R_m^{(k)}\}$. Any point $\mathbf{x}$ lies within one region $m_k(\mathbf{x})$ of each tree with corresponding update $\delta_{m_k(\mathbf{x})}^{(k)}$. Starting with a specified initial value $z(\mathbf{x})$ the estimate $\hat{z}(\mathbf{x})$ at $\mathbf{x}$ is then

$$\hat{z}(\mathbf{x}) = z(\mathbf{x}) + \sum_{k=1}^{K} \delta_{m_k(\mathbf{x})}^{(k)}. \tag{1}$$

1.3 Distribution boosting

Here y and z are both considered to be random variables independently generated from respective distributions $p_y(y \,|\, \mathbf{x})$ and $p_z(z \,|\, \mathbf{x})$. The purpose of a contrast tree is to identify regions of $\mathbf{x}$ - space where the two distributions most differ. The goal of distribution boosting is to estimate a (different) transformation of z at each $\mathbf{x}$, $g_{\mathbf{x}}(z\,)$, such that the distribution of the transformed variable is the same as that of y at $\mathbf{x}$. That is,

$$p_{g_{\mathbf{x}}}(g_{\mathbf{x}}(z\,) \,|\, \mathbf{x}) = p_y(y \,|\, \mathbf{x}). \tag{2}$$

Thus, starting with z values sampled from a known distribution $p_z(z \,|\, \mathbf{x})$ at each $\mathbf{x}$, one can use the estimated transformation $\hat{g}_{\mathbf{x}}(z\,)$ to obtain an estimate $\hat{p}_y(y \,|\, \mathbf{x})$ of the y - distribution at that $\mathbf{x}$. The transformation $g_{\mathbf{x}}(z\,)$ is usually a different function of z at each different $\mathbf{x}$.

The distribution boosting procedure produces an ordered sequence of K contrast trees $\{R_m^{(k)}\}_{m=1}^{M}{}_{k=1}^{K}$. Associated with each region $R_m^{(k)}$ of each tree k is a transformation function $g_{m_k}^{(k)}(\cdot)$. Any prediction point $\mathbf{x}$ lies within one of the regions $m_k(\mathbf{x})$ of each contrast tree k with corresponding transformation function $g_{m_k(\mathbf{x})}^{(k)}(\cdot)$. A value of z can be transformed to a estimated value for y, $\hat{y} = \hat{g}_{\mathbf{x}}(z\,)$, where

$$\hat{g}_{\mathbf{x}}(z\,) = g_{m_K(\mathbf{x})}^{(K)}(g_{m_{K-1}(\mathbf{x})}^{(K-1)}(g_{m_{K-2}(\mathbf{x})}^{(K-2)} \cdots g_{m_1(\mathbf{x})}^{(1)}(z))). \tag{3}$$

That is, the transformed output of each successive tree is further transformed by the next tree in the boosted sequence.

1.4 Two-sample trees

Contrast trees are applied to a single data set where each observation has two outcomes y and z, and a single set of predictor variables $\mathbf{x}$. A similar methodology can be applied to two–sample problems where there are separate predictor variable measurements for y and z. Specifically the data consists of two samples $\{\mathbf{x}_i^{(1)}, y_i\}_1^{N_1}$ and $\{\mathbf{x}_i^{(2)}, z_i\}_1^{N_2}$. The predictor values $\mathbf{x}_i^{(1)}$ correspond to outcomes y_i, and the values $\mathbf{x}_i^{(2)}$ correspond to z_i. The goal is to identify regions in $\mathbf{x}$ - space where the two conditional distributions $p_y(y \,|\, \mathbf{x})$ and $p_z(z \,|\, \mathbf{x})$, or selected properties of those distributions, most differ.

2 ConTree

ConTree is an R package consisting of a collection of procedures implementing aspects of the contrast tree and boosting methodology. This tutorial describes the main procedures as well as examples illustrating their application on several data sets. The procedure descriptions presented here only cover the subset of their respective control parameters most commonly used. See the package documentation for a complete description of each procedure. The principal procedures for building and interpreting contrast trees are *contrast*, *nodesum, nodeplots, treesum, getnodes,* and *lofcurve*. Those for contrast and distribution boosting are *modtrast, xval, predtrast* and *ydist*.

2.1 contrast

This is the central procedure that builds contrast trees:

```
tree = contrast(x,y,z,mode="onesamp",type="dist",tree.size=10,min.node=500)
```

The important arguments are: `x` a $n \times p$ training input predictor variable data matrix or data frame. Rows are the n - observations and columns are p variables. Must be a numeric matrix or a data frame. `y` is a numeric vector of length n containing training data outcome values. `z` is a length n numeric vector containing values of the second contrasting quantity for each observation. `mode` is the contrasting mode. `mode = "onesamp"` $\Rightarrow$ ordinary one-sample tree contrasting y and z. `mode = "twosamp"` $\Rightarrow$ two - sample tree contrasting outcomes y for samples of size n_1 and n_2 respectively. In this mode $n = n_1 + n_2$. `x` contains the $n \times p$ training predictor variable data matrix or data frame of the pooled sample, `y` contains the corresponding pooled outcome values and `z` is a vector of length n specifying sample identity. `z[i]` $<$ `0` $\Rightarrow (\mathbf{x}_i, y_i)$ in the first sample, `z[i]` $>$ `0` $\Rightarrow (\mathbf{x}_i, y_i)$ in second sample.

The parameter `type` controls the nature of the contrast by specifying the discrepancy measure between y and z to be used in each region R_m to build the tree. For `mode = "onesamp"`, there are currently six possibilities for `type:`

`type = "diff"`. Contrast joint paired values of y and z using discrepancy

$$d_m = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} |\, y_i - z_i \,|.$$

N_m is the corresponding observation count in region R_m.

`type = "diffmean"` => contrast absolute mean difference between y and z. Discrepancy measure is

$$d_m = \frac{1}{N_m} \left| \sum_{\mathbf{x}_i \in R_m} (y_i - z_i) \right|. \tag{4}$$

`type = "maxmean"` => contrast signed mean difference between y and z. Discrepancy measure is

$$d_m = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} (y_i - z_i).$$

`type = "prob"` => contrast predicted with empirical probabilities. Here $y_i \in \{0, 1\}$ is the outcome, and z_i is the predicted probability $\Pr(y_i = 1)$ for ith observation. Discrepancy is given by (4).

`type = "quant"` => contrast predicted with empirical quantiles. y_i is the outcome value and z_i is the predicted pth quantile value for ith observation. Discrepancy is lack-of-coverage

$$d_m = \left| p - \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I(y_i < z_i) \right|$$

in the region. For this type an additional parameter `quant` specifies the quantile probability p with default value `quant = 0.5`.

`type = "dist"` => contrast the distribution of y with that of z (default). Let $\{t_i\} = \{y_i\} \cup \{z_i\}$ represent the pooled (y, z) sample in a region R_m. Then discrepancy between the distributions of y and z is taken to be

$$d_m = \frac{1}{2N_m - 1} \sum_{i=1}^{2N_m - 1} \frac{\left| \hat{F}_y(t_{(i)}) - \hat{F}_z(t_{(i)}) \right|}{\sqrt{i \cdot (2N_m - i)}}$$

where $t_{(i)}$ is the ith value of t in sorted order, and $\hat{F}_y$ and $\hat{F}_z$ are the respective empirical cumulative distributions of y and z in the region.

`type = "class"` => misclassification risk. Here y_i and z_i are class labels (`in 1:nclass`) for each ith observation. Region discrepancy is prediction risk

$$d_m = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} C(y_i, z_i)$$

where $C(y, z)$ is the cost of predicting class z when the truth is class y. In this case there are two additional arguments that need to be specified: `nclass` the number of classes (default `nclass = 2`), and `C` an `nclass` by `nclass` misclassification cost matrix (default `C`[i,j] $= I(i \neq j)$)

For mode = "twosamp" there are currently three possibilities for `type`:
`type= "dist"` => contrast y distributions of two samples.
`type = "diffmean"` => contrast absolute difference between y - means of two samples.
`type = "maxmean"` => contrast signed difference between y - means of two samples.

`tree.size` is the specified maximum number of regions (terminal notes) of the tree and `min.node` is the minimum number of observations allowed in each region.

The output of contrast is a contrast tree object `tree` to be used as input to interpretational procedures described below.

2.2 nodesum

This procedure produces a summary of the regions produced by a contrast tree:

```
u = nodesum(x,y,z,tree)
```

The important arguments are: `tree`, a contrast tree object produced by *contrast*. `x` is a $n \times p$ predictor variable data matrix or data frame. Rows are n - observations and columns are p variables. Must be a numeric matrix or a data frame with the same number of columns as that input to contrast. `y` is a numeric vector of length n containing data outcome values. `z` is a length n numeric vector containing values of a contrasting quantity for each observation. This data can, but need not, be the same as the input to *contrast* that produced the tree.

The output of nodesum consists of a list `u` with four components: `u$nodes` is a vector of tree terminal node identifiers. `u$cri` contains the corresponding terminal node discrepancy values (depends on contrast type - see above), `wt` is a vector containing sum of weights (counts) in each terminal node and `avecri` is observation weighted discrepancy averaged over all terminal nodes.

2.3 nodeplots

This procedure produces a graphical summary of the regions comprising a contrast tree:

```
nodeplots(x,y,z,tree,nodes=NULL,pts=FALSE)
```

The parameters `x, y, z` and `tree` are the same as in *nodesum* above. `nodes` is a vector of tree terminal node identifiers specifying the regions to be displayed . The default is all terminal nodes except for `type = "dist"` or `type = "diff"` for which the default is the nine highest discrepancy regions. `pts =TRUE/FALSE` => show points as circles/points (`type = 'dist'` only).

The output graphical representations of terminal node discrepancies depends on tree type. `type = "dist"` produces QQ–plots of y vs. z in each terminal node. Only the nine highest discrepancy nodes are shown. `type = "diff"` shows scatter plots of y versus z in each terminal node. Only nine highest discrepancy nodes are shown. `type = "class"` produces a barplot of misclassification risk (upper) and total weight/counts (lower) in each terminal node. `type = "prob"` shows upper barplot contrasting empirical (blue) and predicted (red) $\Pr(y = 1)$ in each terminal node. Lower barplot shows total weight/counts in each terminal node. `type = "quant"` produces upper barplot of fraction of y - values less than or equal to corresponding z - values (quantile prediction) in each terminal node. Horizontal line reflects specified target quantile. Lower barplot shows total weight/counts in each terminal node. For `type = "diffmean"` or `"maxmean"` upper barplot contrasts y - mean (blue) and z - mean (red) in each terminal node. Lower barplot shows total weight/counts in each terminal node.

2.4 treesum

This procedure prints the $\mathbf{x}$-region boundaries for selected terminal nodes of a contrast tree

```
treesum(tree,nodes=NULL)
```

`tree` is a contrast tree object produced by *contrast.* nodes is a vector of terminal node identifiers for the tree specifying the desired regions. The default is all terminal nodes.

The output of *treesum* is printed at the command line. It summarizes the sequence of splits producing each selected terminal node, one line per split. For a split on a numeric variable the line shows three quantities: the variable number, sign and split point value. If the sign is negative/positive the split point represents an upper/lower boundary. For splits on a categorical variable (factor) there is a variable number, sign and a subset of values (R internal representation). If the sign is positive the listed values are in the node whereas for a negative sign the complement of the listed values are in the node.

2.5 getnodes

This procedure returns the terminal node identifier of the region containing selected observations

```
nx = getnodes(x,tree)
```

`x` is an input predictor data matrix or data frame with same variables and structure input to *contrast.* Rows are observations and columns are variables. Must be a numeric matrix or a data frame. `tree` is a tree model object output from *contrast.* The output of getnodes `nx` is a vector of tree terminal node identifiers (numbers) corresponding to each observation (row of `x`).

2.6 lofcurve

This procedure computes a lack-of-fit curve for a contrast tree.

```
out = lofcurve(x,y,z,tree,doplot=TRUE)
```

The parameters `x, y, z` and `tree` are the same as in *nodesum* above. `doplot = TRUE/FALSE` => do/don't produce graphical plot. The output provides the plot points: `out$x` = horizontal values; `out$y` = vertical values.

2.7 modtrast

This is the basic procedure that builds contrast and distribution *boosting* models.

```
model = modtrast(x,y,z,tree.size=10,min.node=500,type='dist',niter=100)
```

The inputs `x, y, z, type, tree.size,` and `min.node` are the same as the corresponding input to *contrast* above. `type` $\in$ {`"diffmean", "maxmean","prob","quant"`} produces contrast boosting models for estimating the corresponding quantity ($E(y\,|\,\mathbf{x})$, $\Pr(y\ = 1|\,\mathbf{x})$, or $Q_p(y\,|\,\mathbf{x})$). For `type = "quant"` the input parameter `quant` (see above) must be specified. For contrast boosting `x, y` are the input data and `z` represents initial values for the quantity being estimated.

`type = "dist"` produces a distribution boosting model for estimating the full distribution $p_y(y\,|\,\mathbf{x})$ at each $\mathbf{x}$. For this case the input z_i for each observation i is a random number drawn from a prespecified distribution $p_z(z\,|\,\mathbf{x}_i)$ at each $\mathbf{x}_i$. `niter` specifies the number of boosted trees produced.

2.8 xval

This is a diagnostic for accessing the accuracy of models produced by *modtrast* as a function of iteration number

```
out = xval(x,y,z,model,doplot='first',col='red')
```

`model` is a contrast/distribution boosted model produced by *modtrast*. `x, y` and `z` represent data of the type used to construct the model usually based on test observations not used to build it. `doplot = 'first'` $\Rightarrow$ display plot. `doplot = 'next'` $\Rightarrow$ super impose graph on previously displayed plot. `doplot = 'none'` $\Rightarrow$ do not display plot. Outputs `out$x` and `out$y` return the plot points.

2.9 predtrast

Produce predictions from *modtrast* model (`type = "diffmean", "maxmean","prob"` or `"quant"`) for new data.

```
ypred = predtrast(x,z,model,num=model$niter)
```

`x` and `z` are the x and z-values for new data of the same type input to *modtrast*. `model` is a model object output from *modtrast*. `num` is the number of trees used to compute model values. Default is the number contained in model as produced by *modtrast*. The output `ypred` is a vector of predicted values for new data by the model.

2.10 ydist

This procedure computes distribution boosting estimates of the transformation $\hat{y} = \hat{g}_{\mathbf{x}}(z\,)$, such that $p_{\hat{y}}(\hat{y}\,|\,\mathbf{x}) \simeq p_y(y\,|\,\mathbf{x})$ (`type = "dist"` only).

```
yhat = ydist(x,z,model,num=model$niter)
```

The input `x` represents the components of a *single* point $\mathbf{x}$ in predictor variable $\mathbf{x}$-space. `z` is a vector of values to be transformed. These are usually the quantiles of the prespecified distribution $p_z(z\,|\,\mathbf{x}_i)$ at $\mathbf{x}$. `model` is a model object output from *modtrast*. `num` is number of trees used to compute model values. Default is the number contained in model as produced by *modtrast*. The output `yhat` contains the corresponding transformed z - values.

3 Examples

Here we provide examples of using different aspects of the conTree package as applied to several data sets. These examples are intended to illustrate some of the basic features, operation and control of the package procedures. See the full documentation for more advanced options. In this note lines beginning with ">" are to be interpreted as R commands.

3.1 Conditional probability estimation

Here we present R code and resulting output using conTree for the analysis presented in Section 6.2 of Friedman (2020) using the census income data. This sample, taken from 1994 US census data, consists of observations from 48842 people divided into a training set of 32561 and an independent test set of 16281. The binary outcome variable $y \in \{0, 1\}$ indicates whether or not a person's income is greater than \$50000 per year. There are 14 predictor variables $\mathbf{x}$ consisting of various demographic and financial properties associated with each person. The goal is to contrast y with estimates of $\Pr(y = 1\,|\,\mathbf{x})$ obtained by several machine learning methods: gradient boosting on logistic scale using maximum likelihood (GBL), random forest (RF), and gradient boosting on the probability scale (GBP) using least–squares.

First load conTree package

```
> library(contree)
```

The next step is to load the data with the R command

```
> load('census_income_data.Rdata')
```

This loads the following numeric vectors and data frames :

`x`	training data predictor variables (data frame)
`y`	training data outcome (high salary indicator)
`xt`	test data predictor variables (data frame)
`yt`	test data outcome
`gbl`	training data GBL estimates
`gblt`	test data GBL estimates
`rf`	training data RF estimates
`rft`	test data RF estimates
`gbp`	training data GBP estimates
`gbpt`	test data GBP estimates

Note that all of the $\Pr(y = 1 \mid \mathbf{x})$ estimates were obtained using the training data (`x`,`y`) only.

Partition *test* data into two parts

```
> dx = 1:10000; dxt = 10001:16281
```

Contrast y and GBL estimates using first part

```
> tree=contrast(xt[dx,],yt[dx],gblt[dx],type='prob')
```

Validate on second part

```
> nodesum(xt[dxt,],yt[dxt],gblt[dxt],tree)
```

produces the command line output:

```
$nodes
[1] 7 6 29 11 28 16 9 23 13 22
$cri
[1] 0.11499156 0.10586120 0.09498529 0.08688654 0.06500579 0.04966835
[7] 0.04906626 0.04751317 0.03595857 0.02631912
$wt
[1] 561 796 441 411 423 361 1612 431 795 450
$avecri
[1] 0.06556377
```

The first component `$nodes` lists the tree node identifiers for each terminal node in order of region discrepancy. The second `$cri` gives the actual discrepancy of each corresponding region. The third component `$wt` shows the corresponding number of observations in each region. The quantity `$avecri` is the observation weighted average discrepancy over all regions.

The command

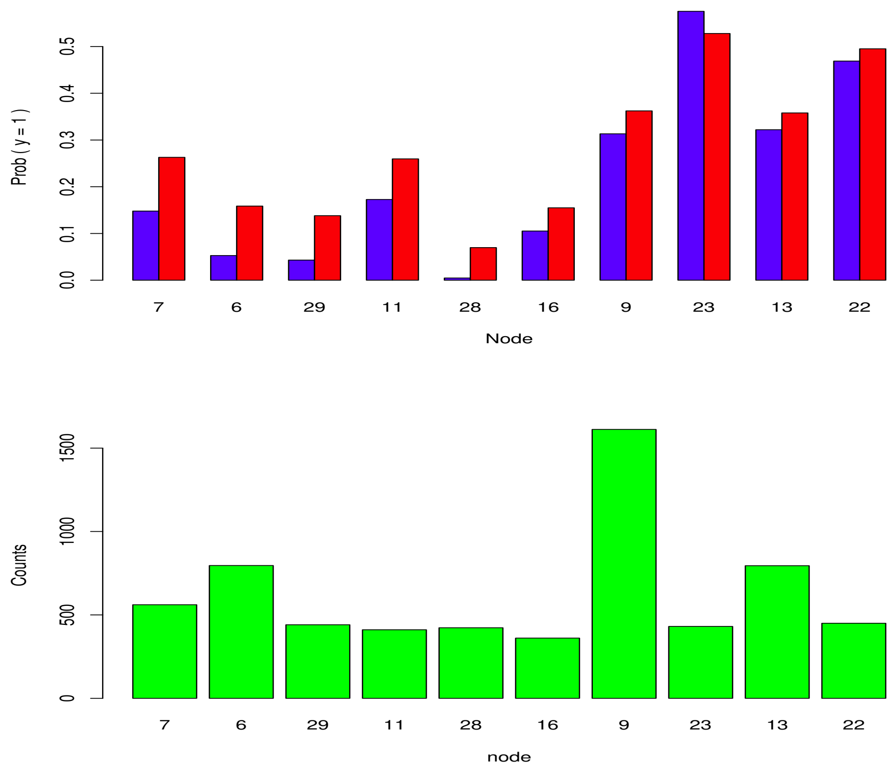

Figure 1: `nodeplots(xt[dxt,],yt[dxt],pot[dxt],tree)`

> `nodeplots(xt[dxt,],yt[dxt],gblt[dxt],tree)`

produces the graphical tree summary shown in Fig. 1. The blue bars in the upper barplot represent the empirical $\Pr(y = 1)$ in each region whereas the red ones show the mean of the GBL predictions in the corresponding regions. One sees dramatic over estimation of the small probabilities.

Region boundaries for selected nodes 7 and 29 can be obtained by

> `treesum(tree,c(7,29))`

with command line output

```
node 7    var      dir      split
          cat 6     -      2  3  5
          cat 7     -      1  2 7  8  9  10  11
node 29   var      dir      split
          cat 6     +      2 3 5
          cat 8     -      1 4  6
              5     -      12
              1     +      28
```

Interpretation of this output is described in Section 2.4.

The command

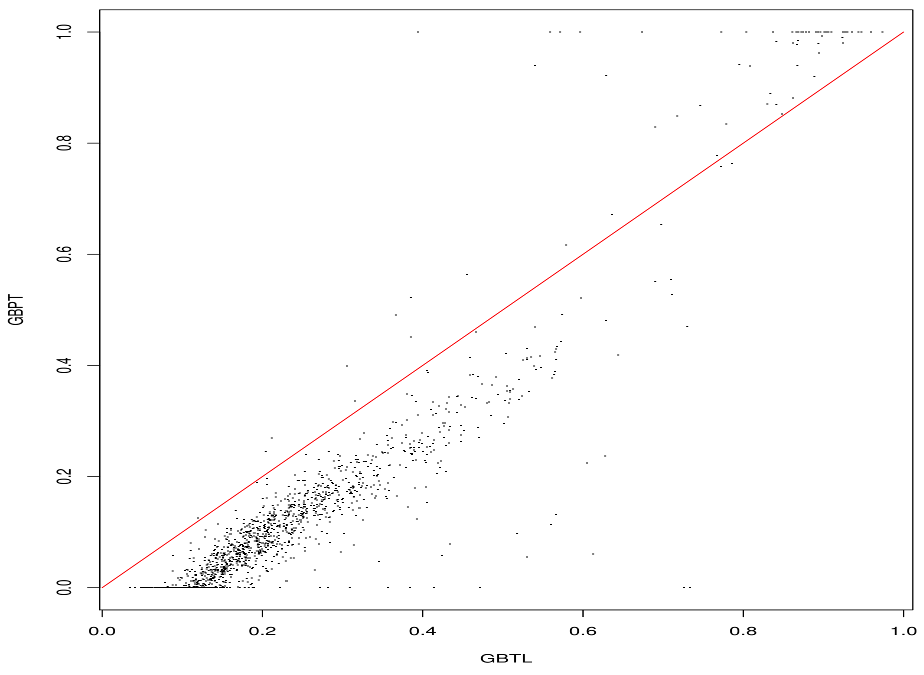

Figure 2: GBL versus GBP probability estimates for test data observations in region 7.

```
> nx=getnodes(xt,tree)
```

obtains the terminal node identity for each test observation for the tree represented in Fig. 1. The commands

```
> plot(gblt[nx==7],gbpt[nx==7],pch='.',xlab='GBTL',ylab='GBPT')
> lines(c(0,1),c(0,1),col='red')
```

plot the test data GBP predictions against those of GBL for the highest discrepancy region 7. The red line represents equality. The result shown in Fig. 2 indicates that the gradient boosting probabilities based on log-odds estimates in this region are considerably larger but proportional to those obtained by direct probability estimation using least-squares. One also sees the effect of truncating the out of range estimates produced by the latter. In spite of this the GBP probability estimates are seen in Fig. 3 (below) to be from three to four times more accurate than those of GBL

We next plot the lack-of-fit contrast curve for GBL using a 50 terminal node contrast tree

```
> tree=contrast(xt[dx,],yt[dx],gblt,type='prob',tree.size=50)
> lofcurve(xt[dxt,],yt[dxt],gblt[dxt],tree)
```

Next we add the corresponding curve for RF predicted probabilities

```
> tree=contrast(xt[dx,],yt[dx],rft[dx,2],type='prob',tree.size=50)
> u=lofcurve(xt[dxt,],yt[dxt],rft[dxt,2],tree,doplot=F)
> lines(u$x,u$y,col='blue')
```

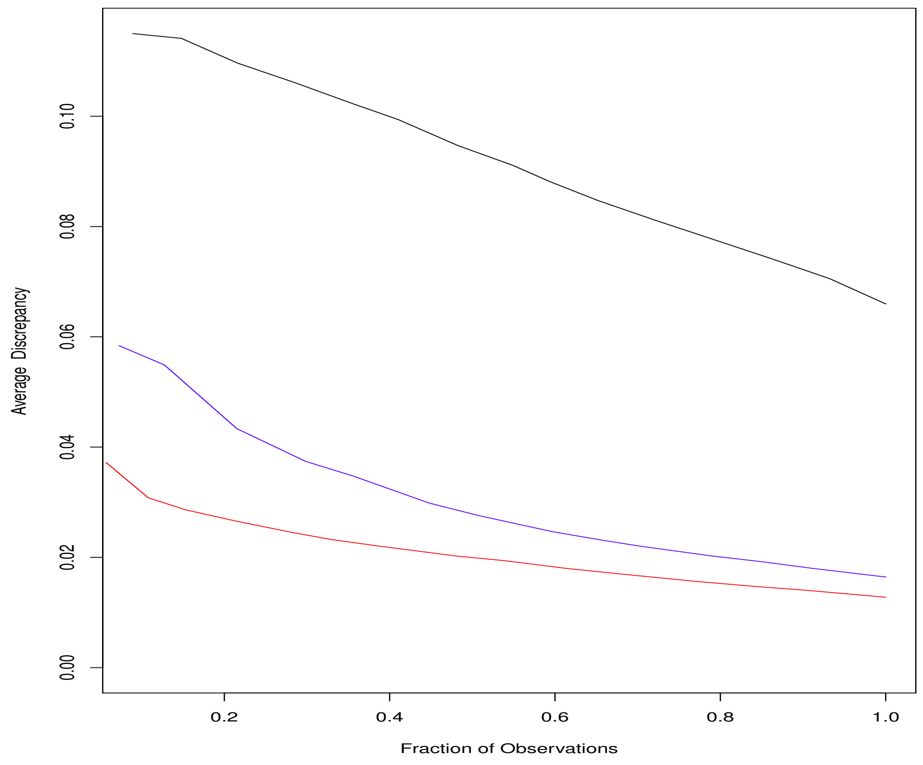

Figure 3: Lack-of-fit contrast curves for GBL (black), RF (blue) and GBP (red) for census income data.

And finally that for the GBP estimates

```
> tree=contrast(xt[dx,],yt[dx],gbpt[dx],type='prob',tree.size=50)
> u=lofcurve(xt[dxt,],yt[dxt],gbpt[dxt],tree,doplot=F)
> lines(u$x,u$y,col='red')
```

The result is shown in Fig. 3.

3.2 Contrast boosting

Here we illustrate the use of contrast boosting to improve prediction performance of GBL. The model is built using the *training* data with the command

```
> modgbl=modtrast(x,y,gbl,type='prob',niter=200)
```

We can plot tree average discrepancy versus iteration number on the training data with the command

```
> xval(x,y,gbl,modgbl,col='red')
```

and super impose a corresponding plot based on the test data set with

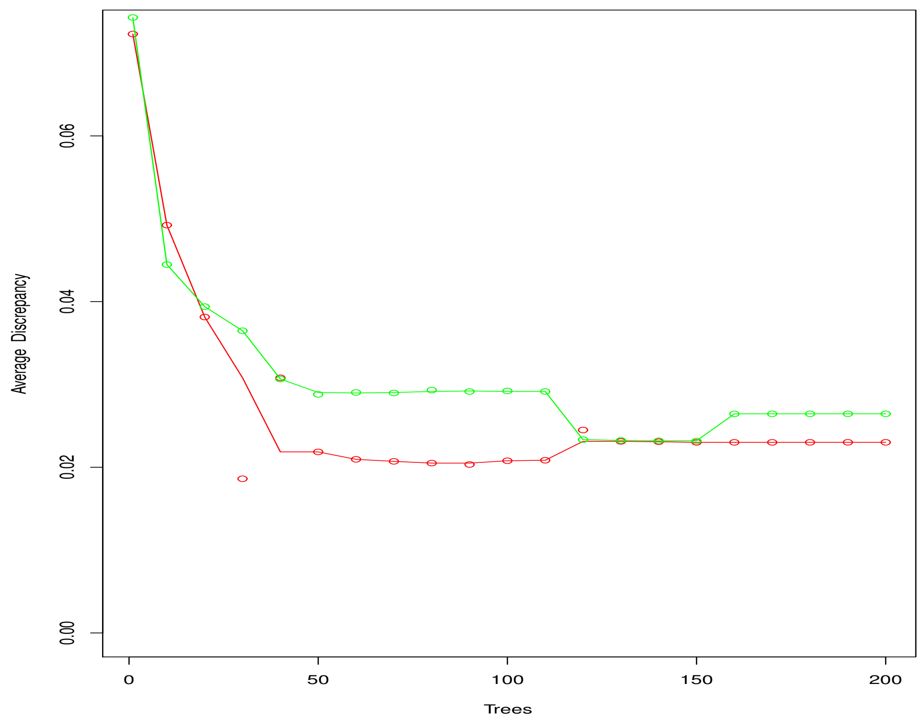

Figure 4: `xval(x,y,gbl,modgbl,col='red')`

```
> xval(xt,yt,gblt,modgbl,col='green',doplot='next')
```

The result is shown in Fig.4. Note that results at every tenth iteration are shown.

Model predictions on the test data set can be obtained with the command

```
> hblt=predtrast(xt,gblt,modgbl)
```

These can be contrasted with the y - values using the first test set sample

```
> tree=contrast(xt[dx,],yt[dx],hblt[dx],type='prob')
```

and summarized on the other test sample with the command

```
> nodeplots(xt[dxt,],yt[dxt],hblt[dxt],tree)
```

producing Fig. 5.

Comparing with Fig. 1 one sees that contrast boosting the GBL model appears to have substantially reduced the shrinking of the of the extreme probabilities estimates thereby improving its conditional probability estimation accuracy.

One can boost the RF and GBP models in the same way using the commands analogous with those used here for the GBL model. Additionally, lack-of-fit contrast curves can be produced and plotted for all three boosted models using modifications of the corresponding commands above. The results are shown in Fig. 4 of Friedman (2020).

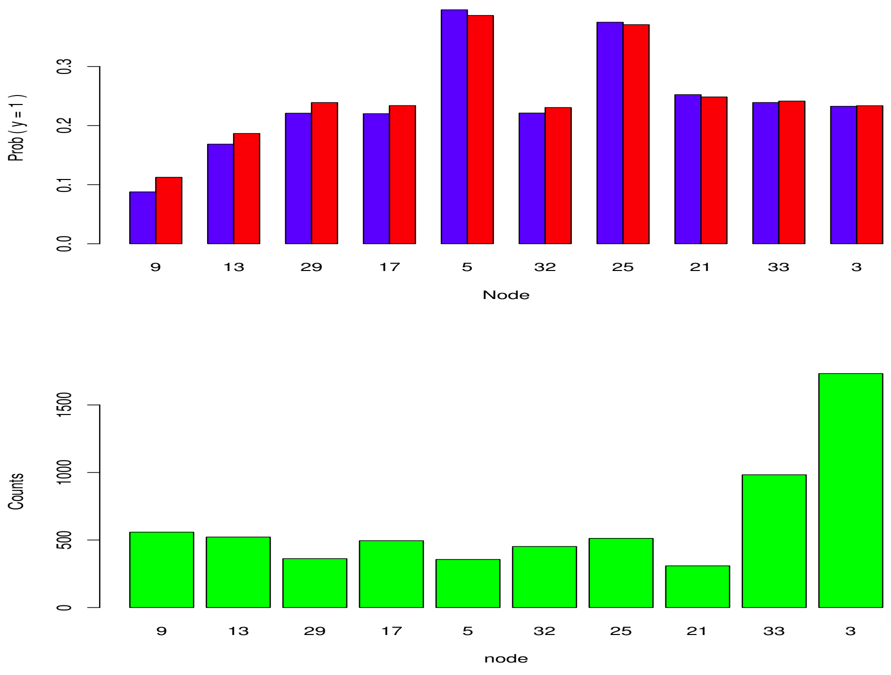

Figure 5: `nodeplots(xt[dxt,],yt[dxt],hbpt[dxt],tree)`

3.3 Conditional distributions

We illustrate contrasting and estimating conditional distributions $p_y(y \,|\, \mathbf{x})$ using the demographics data set described in Table 14.1 of Hastie, Tibshirani and Friedman (2008). Here we attempt to estimate a persons age as a function of the other 13 demographic variables. For this data set age value is reported as being in one of seven intervals

$$age \in \{13\text{-}17,\ 18\text{-}24,\ 25\text{-}34,\ 35\text{-}44,\ 45\text{-}54,\ 55\text{-}64,\ \geq\ 65\}.$$

As input to the algorithm each persons age is randomly generated uniformly within its corresponding reported interval. For the last interval an exponential distribution was used with mean corresponding to life expectancy after reaching age 65.

We first load the data

```
> load('age_data.Rdata')
```

This loads the following numeric vectors and data frames :

`yage`	outcome variable (age)
`xage`	predictor variables (other demographics - data frame)
`gbage`	gradient boosting model for median (yage | xage)

The command

```
> hist(yage)
```

produces the marginal age distribution as seen in Fig 6.

This data is divided into training and test data subsets

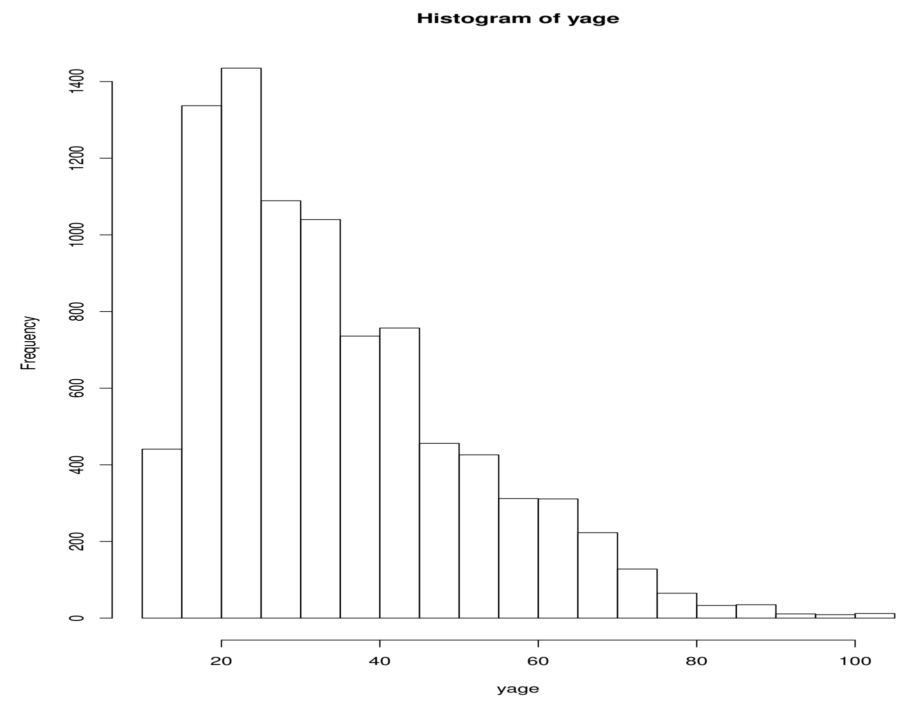

Figure 6: `hist(yage)`

```
> dl=1:5000 # training data
> dt=5001:8856 # test data
```

We next contrast the distribution of $y \mid \mathbf{x}$ with that of the no information hypothesis $p_y(y \mid \mathbf{x}) = \hat{p}_y(y)$ where $\hat{p}_y(y)$ is the empirical marginal y - distribution independent of $\mathbf{x}$ as shown in Fig. 6. First create independent marginal distribution of y as the contrasting distribution

```
> set.seed(5)
> zage=yage[sample.int(length(yage))]
```

Then contrast with y - distribution

```
> treezage=contrast(xage[dt,],yage[dt],zage[dt],tree.size=9,min.node=200)
```

with tree test set command line summary produced by

```
> nodesum(xage[dt,],yage[dt],zage[dt],treezage)

$nodes
[1] 7 6 17 26 11 27 4 20
$cri
[1] 0.9600578 0.7737634 0.5972841 0.5755558 0.5276239 0.4381393 0.1946532
[8] 0.1542807
$wt
[1] 320 597 268 367 304 294 1477 229
$avecri
[1] 0.4544866
```

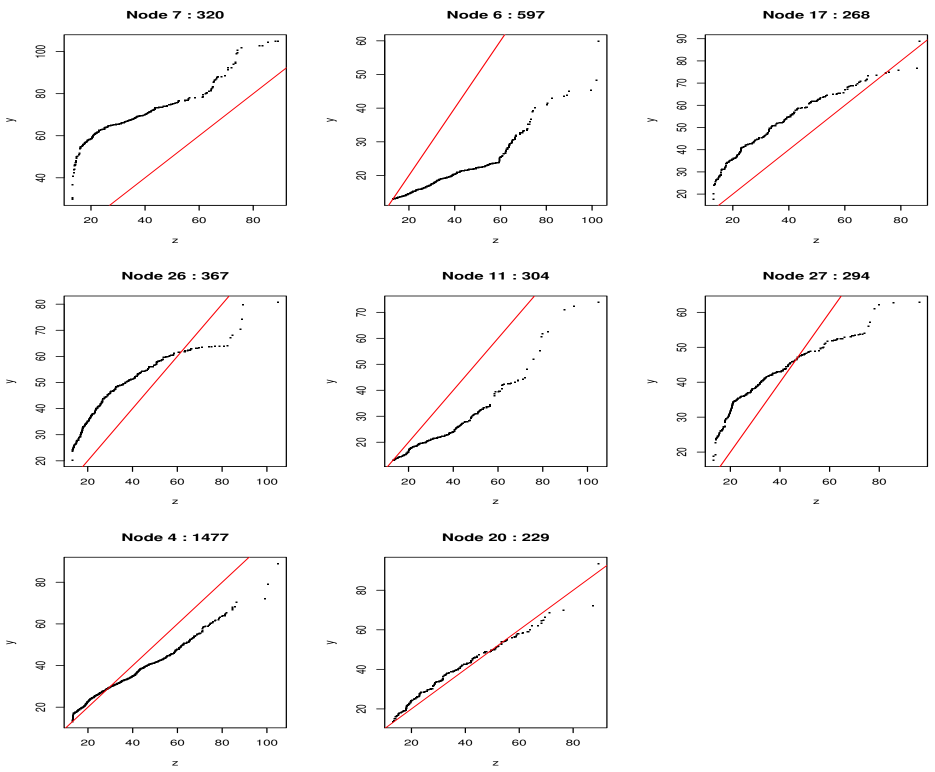

Figure 7: `nodeplots(xage[dt,],yage[dt],zage[dt],treezage)`

and graphical summary produced by

```
> nodeplots(xage[dt,],yage[dt],zage[dt],treezage)
```

shown in Fig. 7.

Each frame in Fig. 7 shows a QQ-plot of the distribution of age y versus that of its $\mathbf{x}$ independent marginal counterpart (Fig. 6) in each of eight regions returned by contrast. To the extent the two distributions are similar the points would lie close to the diagonal (red) line. Here they are see to be very different indicating that $p_y(y \,|\, \mathbf{x})$ is highly dependent on $\mathbf{x}$.

We next construct $y_B \sim p_B(y \,|\, \mathbf{x})$, the residual bootstrap approximation to $p_y(y \,|\, \mathbf{x})$ using gradient boosting median estimates for its location. This assumes that $p_y(y \,|\, \mathbf{x})$ is homoskedastic with varying location

```
> res=yage - gbage
> rbage=gbage + res[sample.int(length(res))]
```

and contrast y with y_B on the test data

```
> treerbage=contrast(xage[dt,],yage[dt],rbage[dt],tree.size=9,min.node=200)
> nodeplots(xage[dt,],yage[dt],rbage[dt],treerbage)
```

The result is shown in Fig. 8.

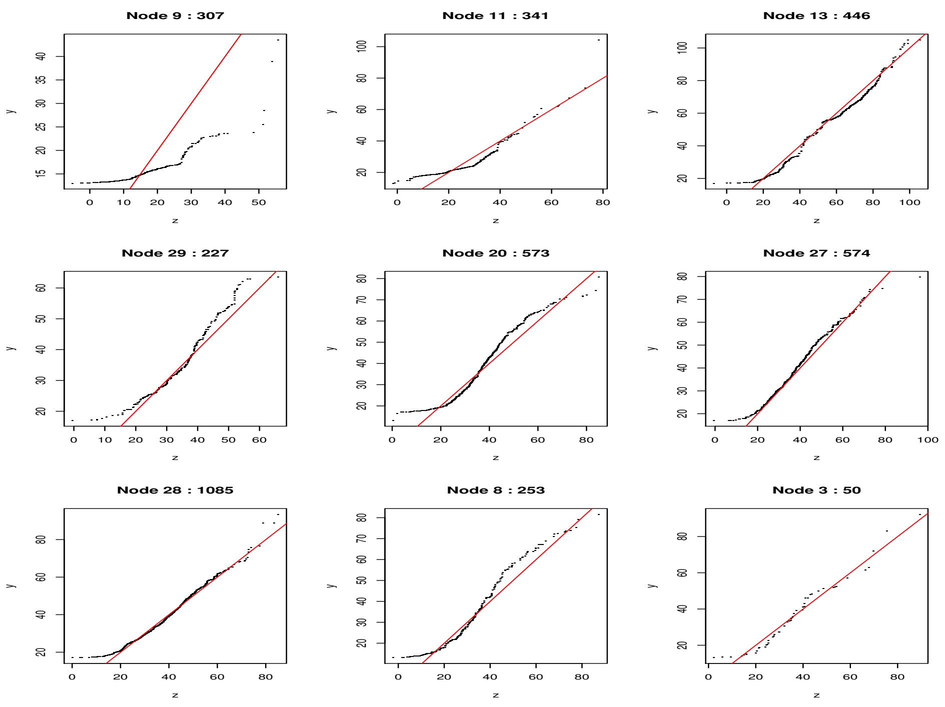

Figure 8: `nodeplots(xage[dt,],yage[dt],rbage[dt],treerbage)`

Although not perfect, the residual bootstrap $p_B(y\,|\,\mathbf{x})$ is seen to provide a much closer approximation to $p_y(y\,|\,\mathbf{x})$ than the global marginal $p_y(y)$. This indicates that its location has a strong dependence on $\mathbf{x}$ as captured by the gradient boosting conditional median estimate.

Next we attempt to further improve the estimate of $p_y(y\,|\,\mathbf{x})$ by applying distribution boosting to the training data starting with the residual boostrap approximation.

```
> mdlrb=modtrast(xage[dl,],yage[dl],rbage[dl],min.node=200)
```

The commands

```
> xval(xage[dl,],yage[dl],rbage[dl],mdlrb,col='red')
> xval(xage[dt,],yage[dt],rbage[dt],mdlrb,col='green',doplot='next')
```

produce a plot of training (red) and test (green) data average tree discrepancy as a function of iteration number (every 10th iteration) as shown is Fig. 9.

The command

```
> hrbage=predtrast(xage[dt,],rbage[dt],mdlrb)
```

transforms the test data residual bootstrap distribution $y_B \sim p_B(y\,|\,\mathbf{x})$ to the y distribution estimate $\hat{y} \sim \hat{p}_{\hat{y}}(y\,|\,\mathbf{x})$.

The commands

```
> treehrbage=contrast(xage[dt,],yage[dt],hrbage,tree.size=9,min.node=200)
> nodeplots(xage[dt,],yage[dt],hrbage,treerbage)
```

contrast the transformed distribution $\hat{y}$ with that of y on the test data. Results are shown in

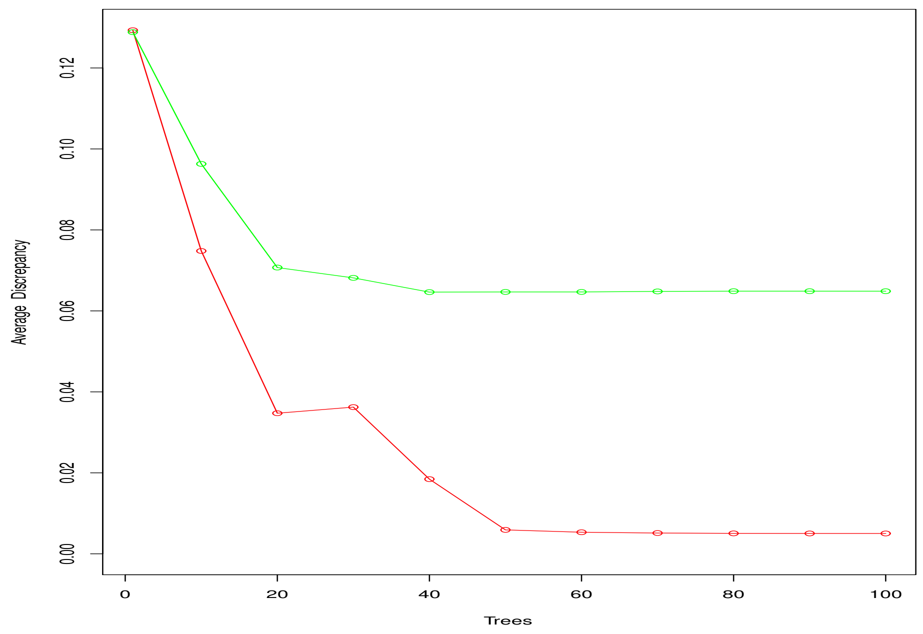

Figure 9: `xval(xage[dl,],yage[dl],rbage[dl],mdlrb,col='red')`

Fig.10.

As seen the results are not perfect but somewhat better than that for the residual bootstrap distribution shown in Fig. 8. This is verified by the corresponding lack-of-fit contrast curves shown in Fig. 11 as produced by the commands

```
> lofcurve(xage[dt,],yage[dt],zage[dt],treezage)
> u=lofcurve(xage[dt,],yage[dt],rbage[dt],treerbage,doplot=F)
> lines(u$x,u$y,col='blue')
> u=lofcurve(xage[dt,],yage[dt],hrbage,treehrbage,doplot=F)
> lines(u$x,u$y,col='red')
```

The lack-of-fit contrast curves are plotted for the three estimates of $p_y(y\,|\,\mathbf{x})$: global marginal $p_y(y)$ (black), residual bootstrap $p_B(y\,|\,\mathbf{x})$ (blue) and distribution boosting estimate $\hat{p}_y(y\,|\,\mathbf{x})$ (red). Distribution boosting is seen to improve the accuracy of the conditional distribution estimate by roughly a factor of two.

Finally we estimate $p_y(y\,|\,\mathbf{x})$ at $\mathbf{x}$ - values for nine selected observations

```
> obs=c(8843,5716,7831,6505,4949,7555,3202,6048,7134)
```

at 500 quantile values

```
> p=((1:500)-0.5)/500; qres=as.numeric(quantile(res,p))
```

and plot their CDFs with

```
> par(mfrow=c(3,3))
> for (k in 1:9) {
+ plot(ydist(xage[obs[k],],gbage[obs[k]]+qres,mdlrb),p,type='l',xlim=c(13,100),
```

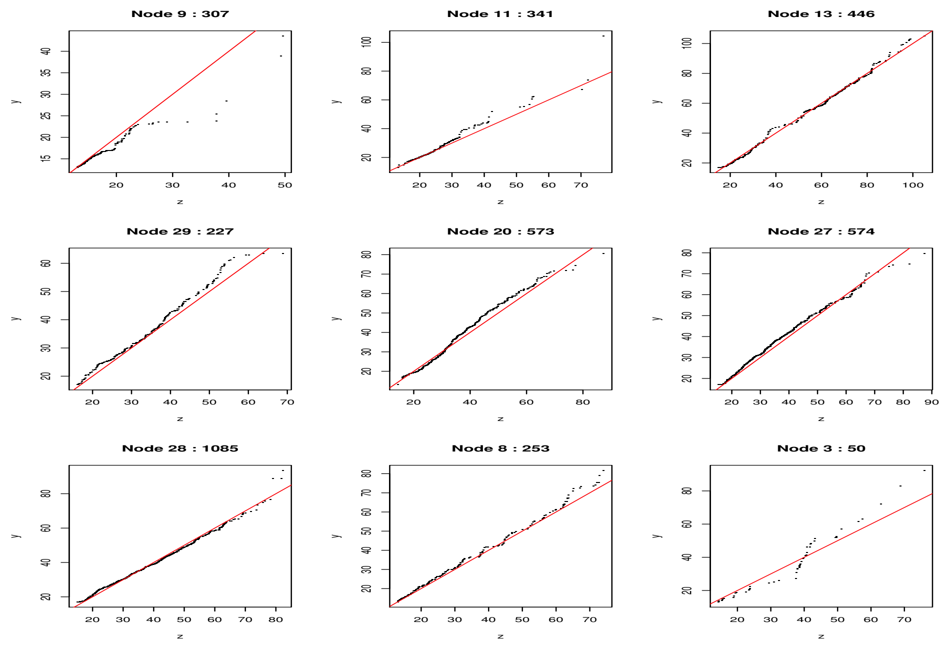

Figure 10: `nodeplots(xage[dt,],yage[dt],hrbage,treerbage)`

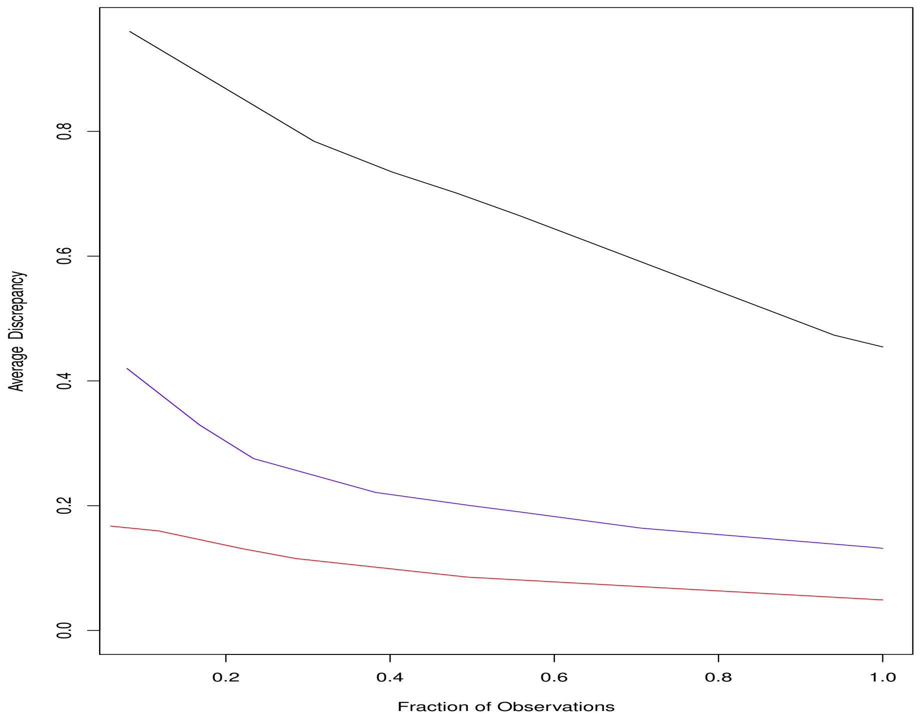

Figure 11: Lack-of-fit contrast curves.

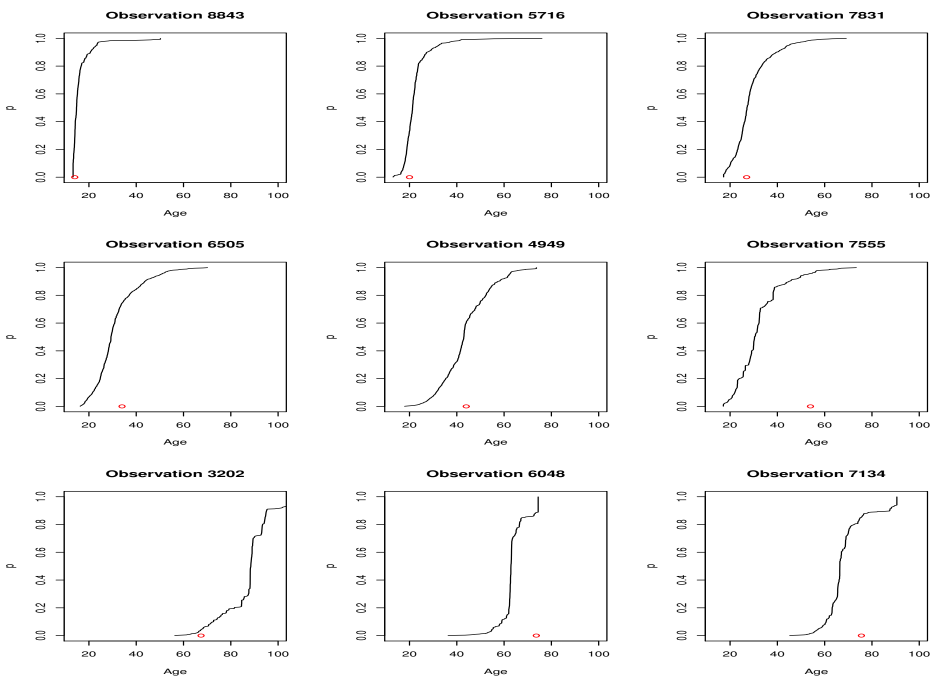

Figure 12: CDF estimates for nine **x**-values.

```
+ xlab='Age',main=paste('Observation',as.character(obs[k])))
+ points(yage[obs[k]],0,col='red')
+ title(paste('Observation',obs[k]))
+ }
```

as shown in Fig. 12. The red points shown at the bottom of each plot display the actual realized y - value (age) for that observation. Prediction intervals for each observation can be read directly from its corresponding CDF display.

Probability densities for these observations can be visualized with the commands

```
> par(mfrow=c(3,3))
> for (k in 1:9) {
+ hist(ydist(xage[obs[k],],gbage[obs[k]]+qres,mdlrb),xlim=c(13,100),nclass=10,
+ xlab='Age',main=paste('Observation',as.character(obs[k])))
+ points(yage[obs[k]],0,col='red')
+ title(paste('Observation',obs[k]))
+ }
> par(mfrow=c(1,1))
```
as seen in Fig. 13.

Considerable heteroskedasticity and skewness in the estimated conditional distributions are evident.

3.4 Two sample contrast trees

The use of two sample contrast trees is illustrated on the air quality data set also from the Irvine Machine Learning Data Repository. The data set consists of hourly averaged measurements from an array of 5 metal oxide chemical sensors embedded in an air quality chemical multisensor device. The outcome variable y is the corresponding true hourly averaged concentration CO

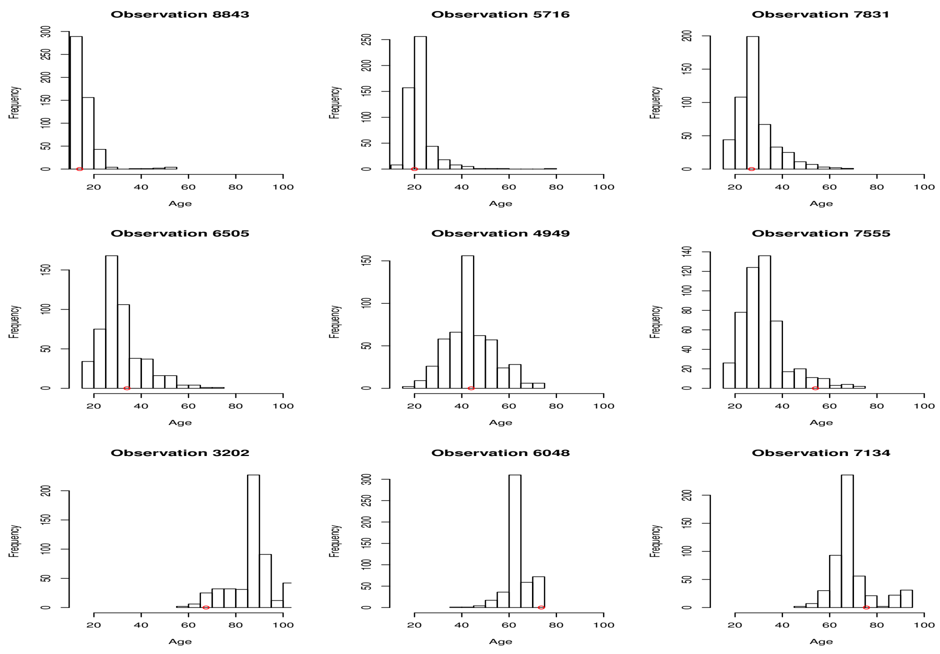

Figure 13: Corresponding probability densities for the nine observations

taken from a reference analyzer. The input variables **x** are taken to be the corresponding hourly averaged measurements of the other 13 quantities as described at the download web site. The goal here is to contrast the conditional distribution of $y \,|\, \mathbf{x}$ for data taken in the first six months (January – June) to that of the last six months (July – December).

The first step is to load the data

```
> load('air_quality_data.Rdata')
```

This loads in three vectors and a data frame

`yco`	outcome variable (CO concentration)
`xco`	predictor variables (data frame)
`zco`	sample membership indicator
`pr2`	probability propensity score.

The first quantity `yco` is the outcome variable y, `xco` is a data frame containing the 13 predictor variables **x** for each observation and `zco` indicates sample membership: `zco` $<$ `0` $\Rightarrow$ first six months, `zco` $>$ `0` $\Rightarrow$ last six months. The vector `pr2` contains gradient boosting model predictions for $\Pr(z > 0 \,|\, \mathbf{x})$; that is the predicted probability of each observation belonging to the second sample as estimated from the predictor variables.

We first contrast the means $E(y \,|\, \mathbf{x})$ of the different samples as a function of **x**. The command

```
> c(mean(yco[zco<0]),mean(yco[zco>0]))
```

displays the global means of the two samples

```
[1] 23.56415  22.98108
```

with mean difference 0.058.

We first create learning and test data sets

```
> dl=1:6000   # learning
> dt=6001:9357 # test
```

and contrast the means as a function of $\mathbf{x}$ with

```
> tree=contrast(xco[dl,],yco[dl],zco[dl],mode='twosamp',type='diffmean')
> nodesum(xco[dt,],yco[dt],zco[dt],tree)
```

producing output on the command line

```
$nodes
[1] 11 21 9 17 2 24 31 33 10 32
$cri
[1] 11.567835 6.843162 5.834586 4.559732 3.305389 3.269056 2.079791
[8] 2.008699 1.224605 1.036612
$wt
[1] 307 292 250 287 334 309 275 314 349 640
$avecri
[1] 3.790423
```

Here cri represents the mean difference in each of the nine regions uncovered by the contrast tree. One sees that there are local regions of the $\mathbf{x}$ - space where the mean difference between the two samples is much larger than that of the global means.

Since the contrast tree regions are of finite extent mean differences within each can originate from two sources. One is due to actual differences in the conditional distributions $p_y(y \mid \mathbf{x}, z < 0)$ and $p_y(y \,|\, \mathbf{x}, z > 0)$ in the region. The other is differences in the marginal $\mathbf{x}$ - distributions $p_{\mathbf{x}}(\mathbf{x} \,|\, z < 0)$ and $p_{\mathbf{x}}(\mathbf{x} \,|\, z > 0)$ over each individual region. Since interest is usually in the former one can mitigate the influence of the latter by propensity weighting based on the propensity probability scores. The command

```
> hist(pr2,nclass=100)
```

displays the distribution of the input propensity probability scores in Fig. 14. One sees that there are moderate differences between the $\mathbf{x}$ - distributions of the two samples at least globally.

The propensity weights are calculated with the commands

```
> wp=rep(0,length(zco))
> wp[zco>0]=1/pr2[zco>0]
> wp[zco<0]=1/(1-pr2[zco<0])
> wp=length(wp)*wp/sum(wp)
```

The corresponding (weighted) contrast tree is obtained with

```
> tree=contrast(xco[dl,],yco[dl],zco[dl],w=wp[dl],mode='twosamp',type='diffmean')
> nodesum(xco[dt,],yco[dt],zco[dt],tree,w=wp[dt])
```

yielding the output

```
$nodes
[1] 11 25 17 9 21 28 10 34 2 35
$cri
```

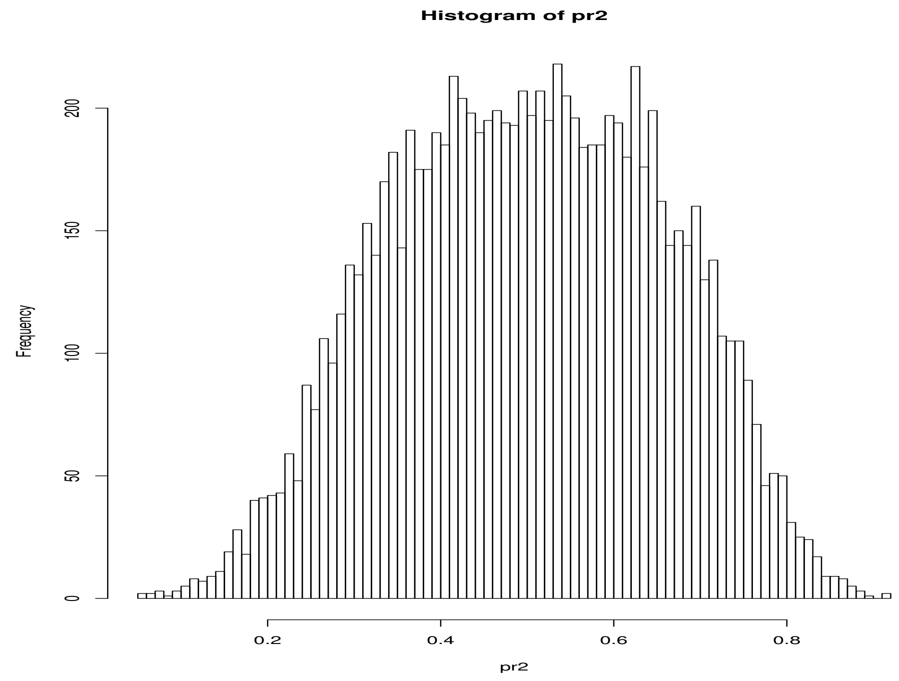

Figure 14: `hist(pr2,nclass=100)`

```
[1] 8.43006534 6.82843552 4.06862342 3.77038131 2.98880825 2.15116385
[7] 1.18141501 1.15381494 0.70247866 0.04491301
$wt
[1] 322.7211 331.9136 266.6065 293.6267 404.4632 347.7717 381.3395 582.1445
[9] 353.6817 296.8669
$avecri
[1] 2.937557
```

The component $wt contains the sum of the weights in each region. Although discrepancies remain, they are somewhat reduced when accounting for the differences of the **x** - distributions within each region. The command

```
> nodeplots(xco[dt,],yco[dt],zco[dt],tree,w=wp[dt])
```

produces the graphical summary shown in Fig. 15. The red/blue bars on the lower plot represent the fraction of the total weights of the first/second sample in each region.

A reference (null) distribution for two sample tree statistics under the hypothesis of no difference between the samples can be obtained by permutation testing. The analysis is repeatedly performed with randomly permuted z - values, thereby randomly assigning observations to each sample. The commands

```
> avedisc=rep(0,1000)
> set.seed(13)
> for (k in 1:1000) {
+ zcot=zco[sample.int(length(zco))]
+ tre=contrast(xco[dl,],yco[dl],zcot[dl],mode='twosamp',type='diffmean')
+avedisc[k]=nodesum(xco[dt,],yco[dt],zcot[dt],tre)$avecri
}
```

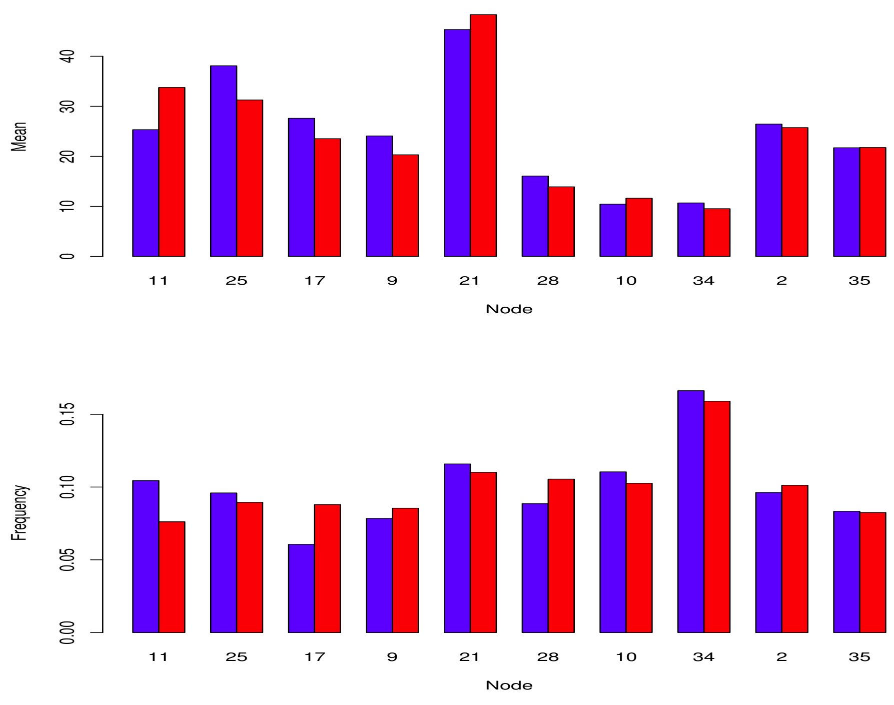

Figure 15: `nodeplots(xco[dt,],yco[dt],z[dt],tree,w=wp[dt])`

```
> hist(avedisc,xlim=c(0,4))
> points(c(2.937557,3.790423),c(0,0),col=c('blue','red'))
```

compute and display average tree discrepancy for 1000 replications of this procedure. Fig. 16 shows the histogram of these null discrepancies along with the corresponding propensity weighted/unweighted alternatives shown as the blue/red points. One sees that the results based on the original sample assignments are highly significant.

The command

```
> treesum(tree,c(11,25))
```

produces the command line output

```
node 11     var      dir     split
            12        +        106
            13        +       5232
             2        +       1112
node 25     var      dir     split
            12        +       106
            13        -       5232
            13        -       4600
            12        -       596
             6        -       3
             5        +       1036
```

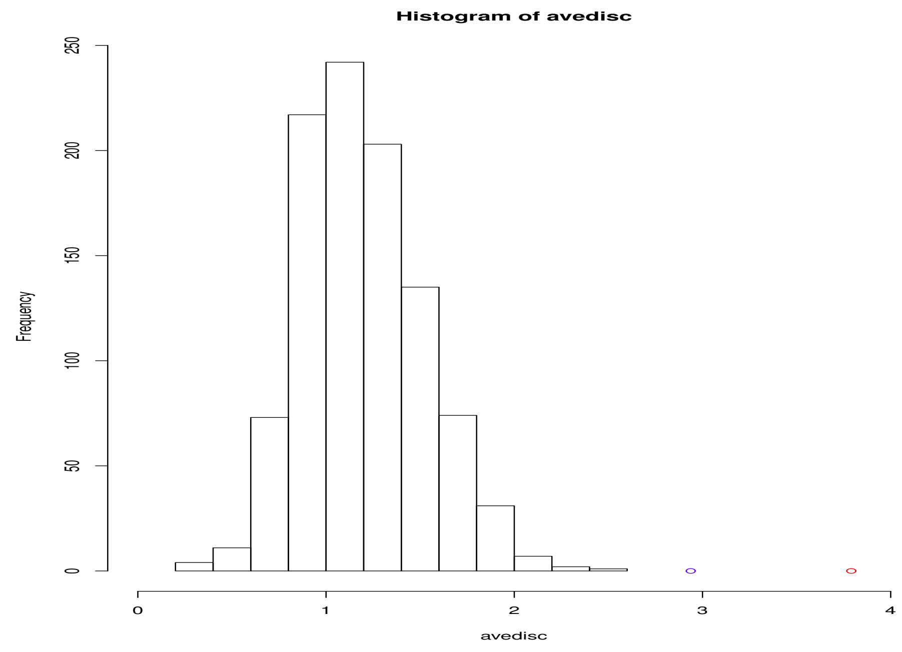

Figure 16: Two sample null distribution.

As described in Section 2.4 the output for each terminal node shows the sequence of splits that produced its corresponding region. The names corresponding to each predictor variable can be obtained with the command

```
> names(xco)
[1] "Time" "PT08.S1.CO." "NMHC.GT." "C6H6.GT."
[5] "PT08.S2.NMHC." "NOx.GT." "PT08.S3.NOx." "NO2.GT."
[9] "PT08.S4.NO2." "PT08.S5.O3." "T" "RH"
[13] "AH"
```

so that the boundaries of nodes 11 and 25 are defined by

Node	Rule
11	$RH > 106 \;\;\&\;\; AH > 5232 \;\;\&\;\; NMHC.GT > 1112$
25	$106 \leq RH < 596 \;\;\&\;\; AH \leq 4600 \;\;\&\;\; NOx.GT < 334 \;\;\&\;\; PT08.S2.NMHC > 1036$.

We next contrast the conditional distributions $y \,|\, \mathbf{x}$ of the two samples. Figure 17 shows a QQ-plot between the respective y global distributions on the test data obtained by the commands

```
> ycodt=yco[dt]
> qqplot(ycodt[zco[dt]<0],ycodt[zco[dt]>0],pch='.')
> lines(c(0,200),c(0,200),col='red')
```

The global y - distributions of the two samples are seen to be nearly identical. The corresponding (propensity weighted) contrast tree results are obtained with the commands

```
> tree=contrast(xco[dl,],yco[dl],zco[dl],w=wp[dl],mode='twosamp',tree.size=9)
> nodeplots(xco[dt,],yco[dt],zco[dt],w=wp[dt],tree)
```

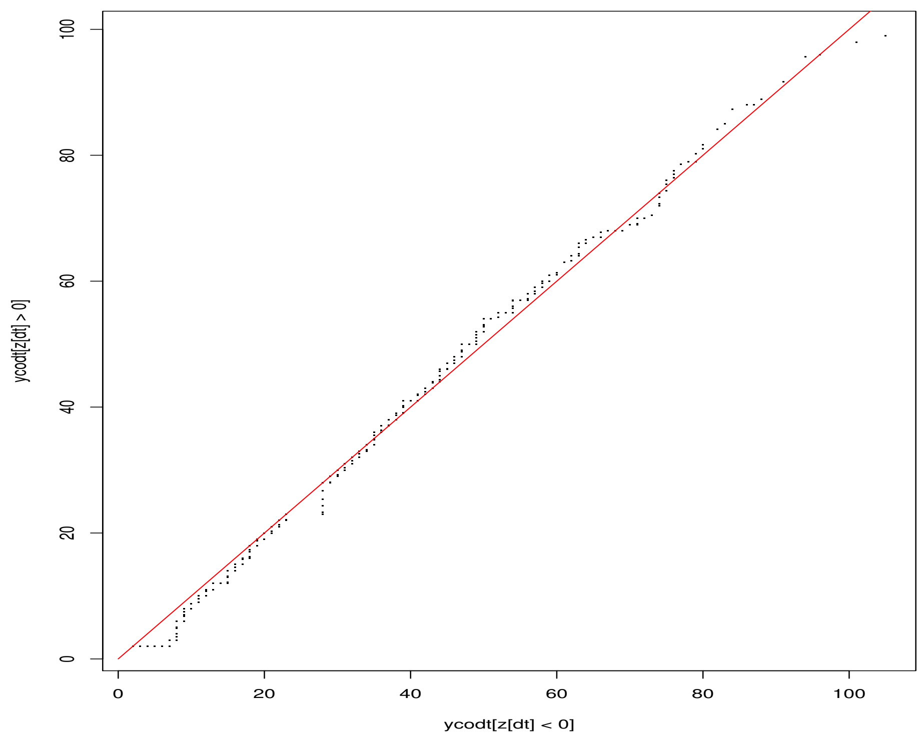

Figure 17: `qqplot(ycodt[z[dt]<0],ycodt[z[dt]>0],pch='.')`

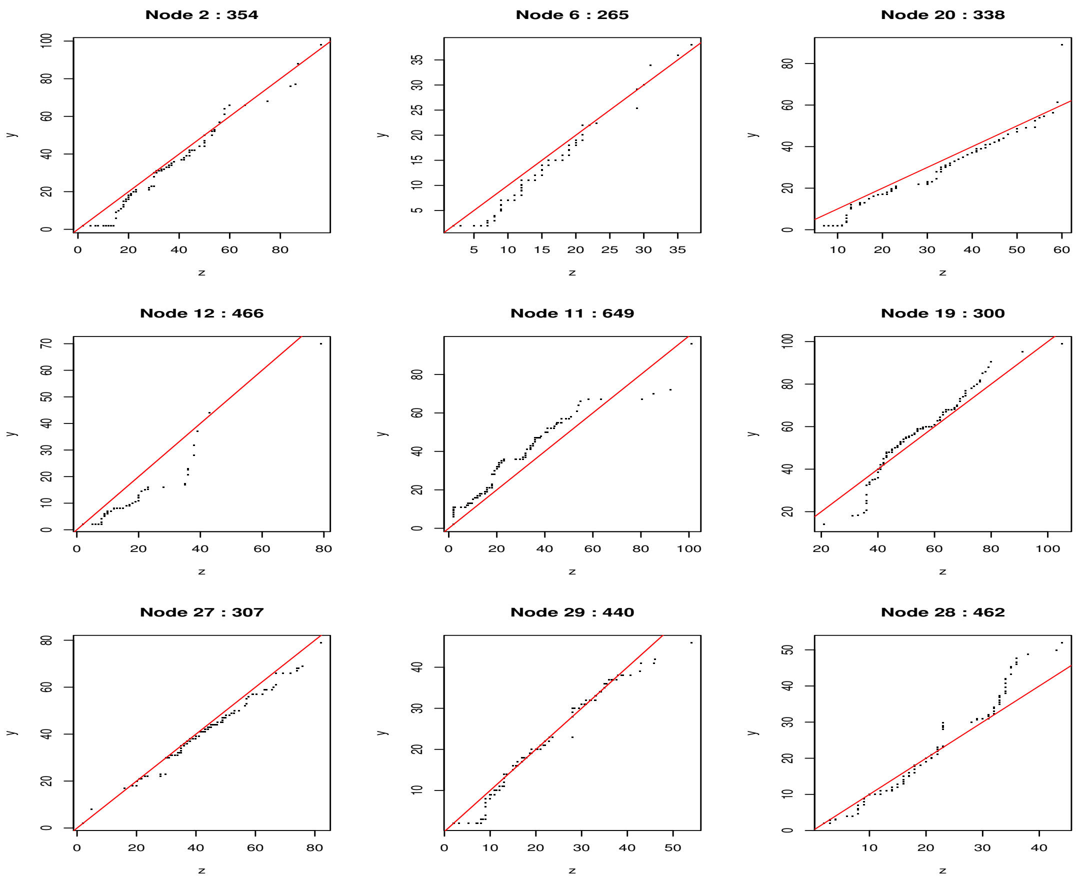

Figure 18: `nodeplots(xco[dt,],yco[dt],z[dt],w=wp[dt],tree)`

as shown in Fig. 18. The tree has uncovered a few regions where there are moderate differences between the two distributions.

4 Acknowledgment

The crucial contribution of Balasubramanian Narasimhan in constructing the contree R package is gratefully acknowledged.

References

[1] Friedman, J. H. (2020). Contrast trees and distribution boosting. *Proc. Nat. Acad. Sci.* (to appear).