RF Estimator Bias Method 1: Default in randomForest

Let b_0 and b_1 denote the estimated intercept and slope, respectively, obtained by fitting a simple linear regression of the true training case response y_i on its out-of-bag prediction $\hat{y}_i, i = 1, \cdots, n$.

The bias-corrected prediction is $\hat{y}_i^0 = b_0 + b_1\hat{y}_i, i = 1, \cdots, n$.

This method assumes a linear relationship between corrected prediciton $\hat{y}^0$ and the original prediciton $\hat{y}$.

Example 1: true model:

$f(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon$

where $x_i's$ are independent, $\sim Uniform(0, 1)$, $\epsilon \sim Normal(0, 1)$

```
n=200
sigma=1
x1=runif(n)
x2=runif(n)
x3=runif(n)
x4=runif(n)
x5=runif(n)
meany=10*sin(pi*x1*x2)+20*(x3-0.5)^2+10*x4+5*x5 # meany is just f(x)
y=rnorm(n,meany,sigma)
traindata=cbind(y,x1,x2,x3,x4,x5)
#set.seed(1000)
#output=randomForest(y~.,data=traindata)
#yhat=output$predicted
#set.seed(1000)
output2=randomForest(y~.,data=traindata,corr.bias=TRUE) #bias corrected prediction
lmout=lm(y~yhat)  # regress y on yhat
yhatcorr=lmout$fitted.values #fitted y values from linear regression
output2$predicted[1:20]
yhatcorr[1:20]
plot(yhat,yhatcorr)
plot(y,yhatcorr)
abline(0,1)

output$mse[500] # check mse from bias uncorrected rf
output2$mse[500] # mse from bias corrected rf

mse1=mean((y-yhat)^2)
mse1
mse2=mean((y-yhatcorr)^2)
mse2

curveout=lm(y~yhat+I(yhat^2)) #fit quadratic curve
yhat3=curveout$fitted.values
```

```
mse3=mean((y-yhat3)^2)
mse3

# plot(meany,output$predicted,xlim=c(0,20),ylim=c(0,20))
# abline(0,1)

# add test data
n=dim(boston)[1]
sam=sample(1:n,size=floor(n/3)) #randomly choose 1/3 test data
testdataB=boston[sam,]
traindataB=boston[-sam,]
ytrain=traindataB[,14]
xtestB=testdataB[,1:13]
ytestB=testdataB[,14]
set.seed(2000)
rf1=randomForest(medv~.,data=traindataB,xtest=xtestB,ytest=ytestB)
set.seed(2000)
rf2=randomForest(medv~.,data=traindataB,xtest=xtestB,ytest=ytestB,corr.bias=T)
rf1$mse[500]
rf2$mse[500]

yhat=rf1$predicted
lmout=lm(traindataB[,14]~yhat)
lmout$coefficients


b0=lmout$coefficients[1]
b1=lmout$coefficients[2]
ytestcorr=b0+b1*rf1$test$predicted
newtestdata=data.frame(yhat=rf1$test$predicted)
ytestcorr2=predict(lmout,newdata=newtestdata) # can also use
                                              # predict to get prediction on test data
ytestcorr[1:20]
ytestcorr2[1:20]
rf2$test$predicted[1:20]


# use smooth spline to do prediction
library(splines)
ytrain=traindataB[,14]
splineout=smooth.spline(yhat,ytrain,cv=FALSE)
newdata=rf1$test$predicted
sp.predict=predict(splineout,newdata)
mean((ytestB-sp.predict$y)^2)
```

Example 2: $x_1 \sim U(0.1, 0.9), x_2 \sim U(0,1), n = 20, f(x) = x_1 + \epsilon, \epsilon \sim N(0, 0.2)$

```
x1=runif(21,0.1,0.9)
x2=runif(21,0,1)
meany=x1
y=rnorm(21,meany,0.1)
traindata=cbind(y,x1,x2)
output=randomForest(y~.,data=traindata)
plot(meany,output$predicted,xlim=c(0,1),ylim=c(-1,2))
abline(0,1)


# E(Y|x)=2*x_1

x1=runif(20,0.1,0.9)
x2=runif(20,0,1)
meany=2*x1
y=rnorm(20,meany,0.2)
traindata=cbind(y,x1,x2)
output=randomForest(y~.,data=traindata)
plot(meany,output$predicted,xlim=c(0,2),ylim=c(0,2))
abline(0,1)
```

Use the loop function to do replications. Assume $f(x) = x_1$

```
 stepsize=0.04
 x1=seq(0.1,0.9,by=stepsize) # fix x1 values
 n=length(x1)
 meany=x1
 k=50
 sigma=0.2
 myresult=matrix(0,n,k) # use this n by k matrix to store the 50 predicted vectors,
                        # inialize the values with 0
for (i in 1:k)
{
 x2=runif(n,0,1)
 y=rnorm(n,x1,sigma)
 traindata=cbind(y,x1,x2)
 rfout=randomForest(y~.,data=traindata)
 myresult[,i]=rfout$predicted #replace each column with predicted values
}
finalpred=apply(myresult,1,mean)  # use apply function to average the columns
```

```
                                    # (i.e, average the 50 vectors of predicted values)

plot(meany,finalpred,xlim=c(0,1),ylim=c(0,1))
abline(0,1)
```

add squared bias metric in addition to mse use Friedman model

```
 ntest=2000
 x1=runif(ntest)
 x2=runif(ntest)
 x3=runif(ntest)
 x4=runif(ntest)
 x5=runif(ntest)
 meanytest=10*sin(pi*x1*x2)+20*(x3-0.5)^2+10*x4+5*x5 # f(x) for test data
 ytest=rnorm(ntest,meanytest,1)
 xtestdata=cbind(x1,x2,x3,x4,x5)
 xtestdata=data.frame(xtestdata)  # hold test data predictors fixed



 n=200
 sigma=1
 k=50
 pred1=matrix(-1000,ntest,k)
 pred2=matrix(-1000,ntest,k)
 mspe=rep(-1000,k)
 for (i in 1:k) {
 x1=runif(n)
 x2=runif(n)
 x3=runif(n)
 x4=runif(n)
 x5=runif(n)
 meany=10*sin(pi*x1*x2)+20*(x3-0.5)^2+10*x4+5*x5 # meany is f(x)
 y=rnorm(n,meany,sigma)
 traindata=cbind(y,x1,x2,x3,x4,x5)
 rf1.out=randomForest(y~.,data=traindata,xtest=xtestdata)
 rf2.out=randomForest(y~.,data=traindata,xtest=xtestdata,corr.bias=T)
 pred1[,i]=rf1.out$test$predicted
 pred2[,i]=rf2.out$test$predicted
}

finalpred1=apply(pred1,1,mean)
finalpred2=apply(pred2,1,mean)
biassq1=mean((finalpred1-meanytest)^2)
biassq2=mean((finalpred2-meanytest)^2)
```

```
biassq1;biassq2



# How to subset a data set by variables
n=dim(boston)[1]
sam=sample(1:n,size=floor(n/3))
testdata=boston[sam,]
traindata=boston[-sam,]
ytest=subset(testdata,select=medv) # take  y
ytest=ytest[,1] # turn into a vector
xtest=subset(testdata,select=-medv)
rf1=randomForest(medv~.,data=traindata,xtest=xtest,ytest=ytest)
rf2=randomForest(medv~.,data=traindata,xtest=xtest,ytest=ytest,corr.bias=TRUE)
mse1=rf1$test$mse[500]
mse2=rf2$test$mse[500]
mse1;mse2
```