Lecture8 Random Forest Implementation Handout

Elizabeth Colantuoni

4/12/2021

I. Objectives

The goal of this document is to walk you through implementing a random forest for a continous/linear outcome.

We will do this within the NMES data; goal is to predict $log(totalexp + 1)$.

II. Prior analyses

In Lecture 6, we stratified the NMES dataset into a training and testing/validation sample and build a regression tree. The results from this procedure are below:

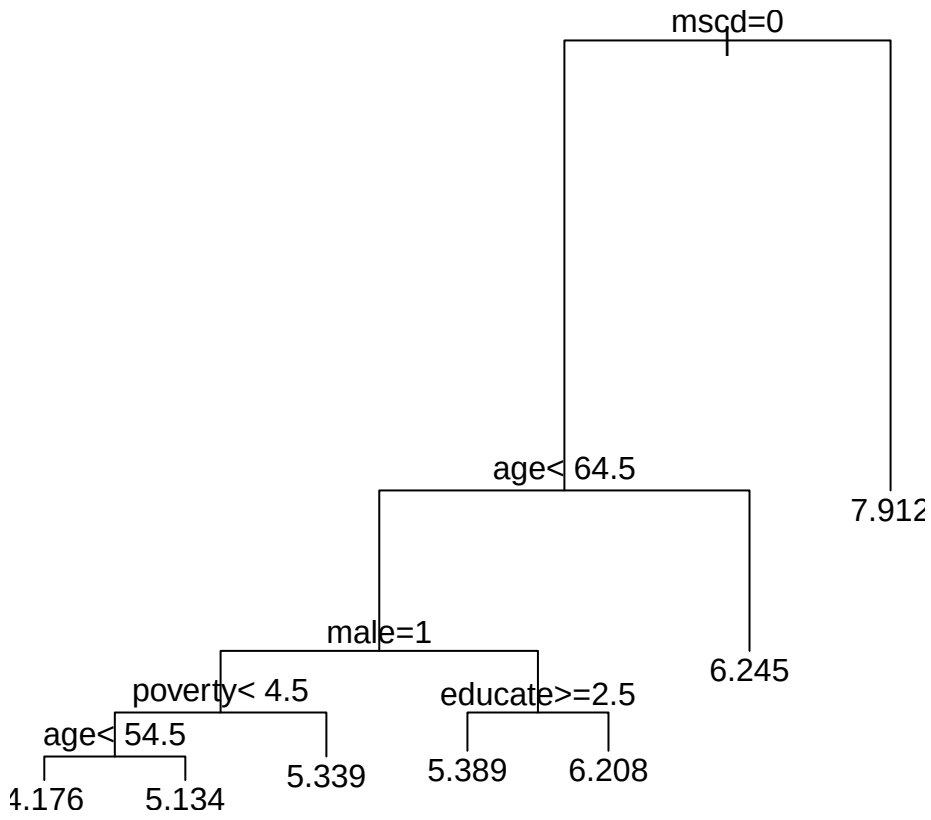
```
load('./nmes.rdata')
d1 = nmes
d1[d1=='.'] = NA

## Create the necessary variables:
d1$bigexp=ifelse(d1$totalexp>1000,1,0)
d1$mscd=ifelse(d1$lc5+d1$chd5>0,1,0)


dat=data.frame(e=log(d1$totalexp+1),
  age=d1$lastage,
  mscd=factor(d1$mscd),
  beltuse=as.numeric(d1$beltuse),
  educate=as.numeric(d1$educate),
  married=factor(d1$marital),
  poverty=as.numeric(d1$povstalb),
  male=factor(d1$male))

# Keep only the complete cases
dat = dat[complete.cases(dat),]

# Create the training and testing/validation samples
set.seed(123454321)
dat.train=dat[train<-sample(1:nrow(dat),floor(nrow(dat)/2)),]
dat.test=dat[-train,]
```

```
## Fit a first tree settting the "cp" parameter to 0.001
tree0=rpart(e~.,data=dat.train,method="anova",control=rpart.control(minsize=20,cp=.0010))
tree = prune(tree0,
```

```
cp = tree0$cptable[which.min(tree0$cptable[,"xerror"]), "CP"]
## Print the variable importance
tree$variable.importance
```

```
##       mscd        age    educate       male    married    poverty    beltuse
## 3109.1510 1282.8392  461.4138  425.8878  289.6406  260.7690   22.2820
```

```
## Plot the tree and add labels
plot(tree)
text(tree,pretty=3)
```



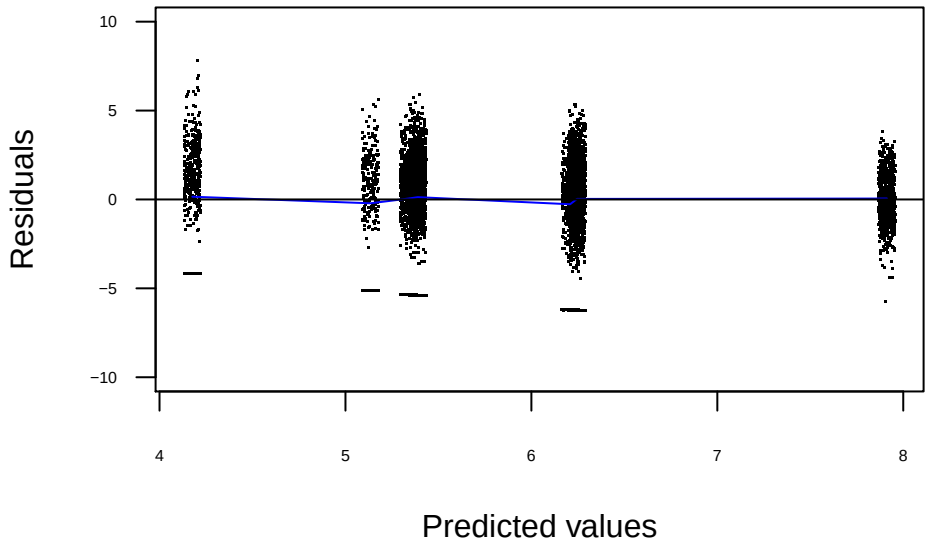
Now let's predict $log(totalexp + 1)$ for the test data set that was left out when the tree was trained.

```
# Generate predicted values
tree.yhat=predict(tree,newdata=dat.test,na.action=na.pass)
# Compute the residuals
res.tree.test=dat.test$e-tree.yhat
# Compute the MSE = sums of squared residuals / n for the
```

```
# test/validation dataset
mse.tree.test=sum(res.tree.test^2)/length(res.tree.test)
mse.tree.test
```

```
## [1] 5.806413
```

```
# Plot the residuals vs. predicted values
o=order(tree.yhat)
par(mar=c(4,4,1,1))
plot(jitter(tree.yhat,factor=6),jitter(res.tree.test,factor=6),
     pch=".",xlab="Predicted values",
     ylab="Residuals",ylim=c(-10,10),las=1,cex.axis=0.5)
lines(tree.yhat[o],predict(lm(res.tree.test~ns(tree.yhat,5)))[o],
      type="l",col="blue");abline(h=0,col="black")
```



III. Random forest implementation

A. Creating the forest

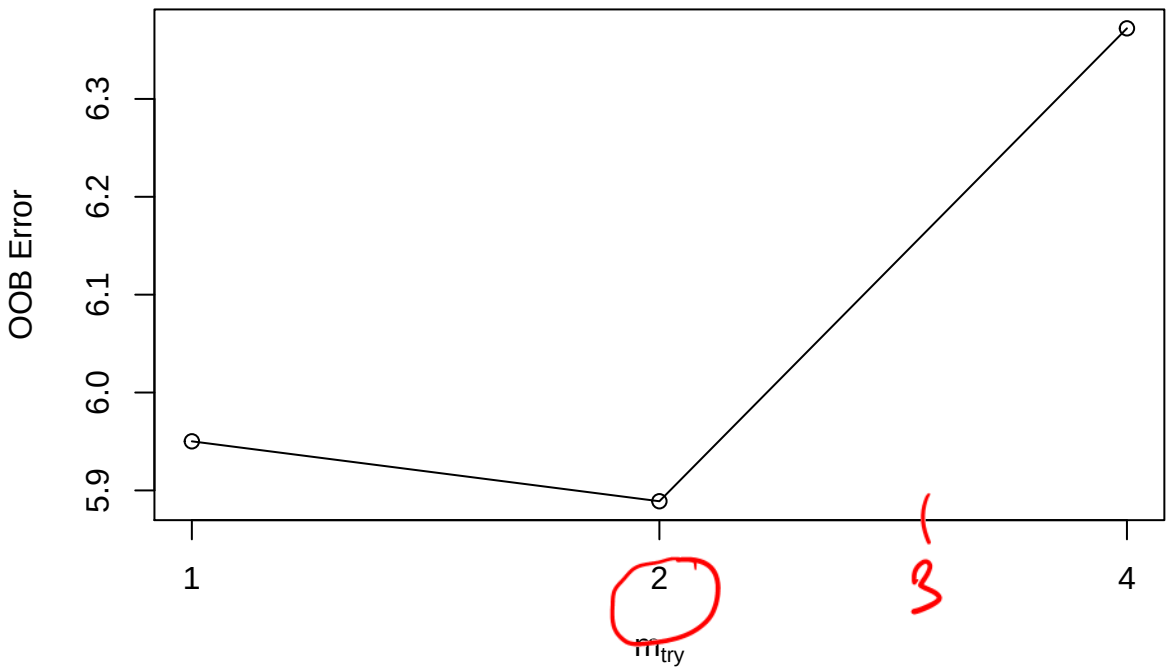
There are two main tuning parameters for the random forest construction:

- m: the number of randomly selected variables to try at each split
- *ntree*: the number of trees for the forest

We will use the "tuneRF" function to select m where we specify a large value of *ntree*.

```
tune_rf = tuneRF(x = dat.train[,2:8],y = dat.train[,1], ntreeTry = 500)
```

```
## mtry = 2  OOB error = 5.888937
## Searching left ...
## mtry = 1     OOB error = 5.950159
## -0.01039609 0.05
## Searching right ...
## mtry = 4     OOB error = 6.372113
## -0.08204803 0.05
```

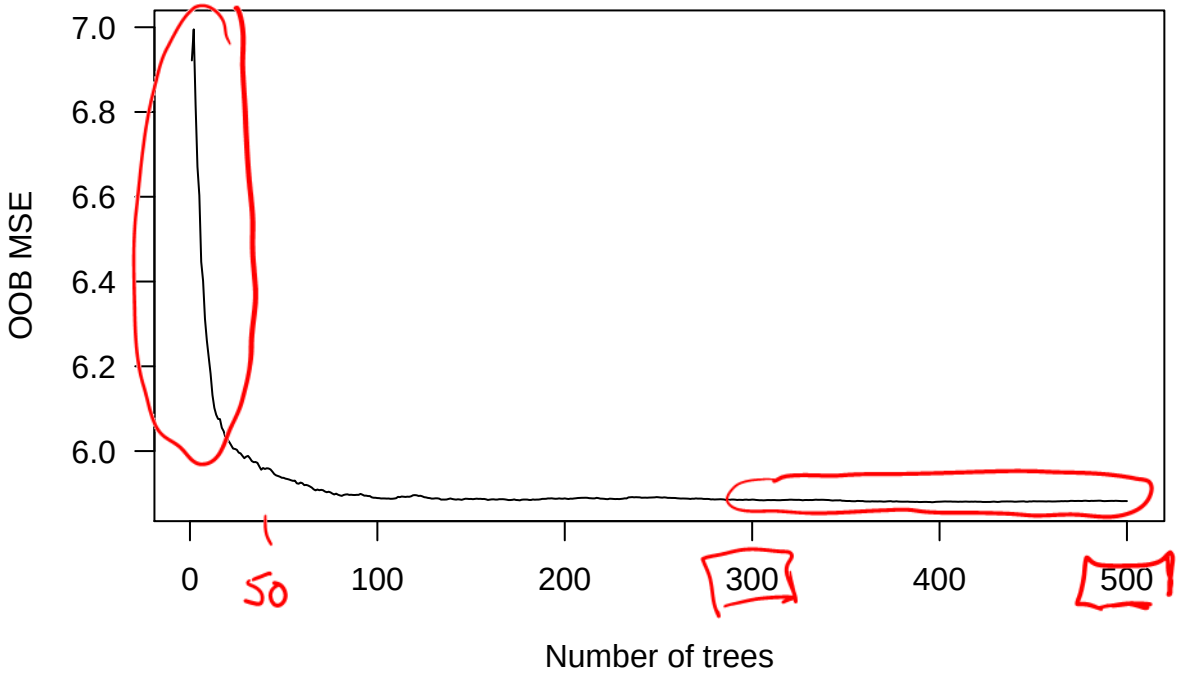



```
## Print the tuneRF summary
tune_rf
```

```
##   mtry OOBError
## 1    1 5.950159
## 2    2 5.888937
## 4    4 6.372113
```

```
## Now fix m, and plot the out-of-bag MSE vs. ntree
m = tune_rf[which.min(tune_rf[,2]),1]
rf_m = randomForest(x = dat.train[,2:8],y = dat.train[,1],mtry=m,ntree=500,keep.forest = TRUE)

## Make the plot evaluating ntree
plot(1:500,rf_m$mse,type="l",xlab="Number of trees",ylab="OOB MSE",las=1)
```

↳ out-of-bag MSE of prediction
length 500
# of trees 1 OOB MSE
2 OOB MSE
⋮
ntree = 500 OOB MSE



Based on the figure, the mean squared error of out-of-bag prediction plateaus at roughly 400 to 500 trees. Creating the the forest of 500 trees is sufficiently large.

B. Summarizing the forest

Once you have the forest, you can:

- view the variable importance
- view the out-of-bag predictions
- make predictions for new observations
- compute MSE for comparison with other models
- create a summary average tree

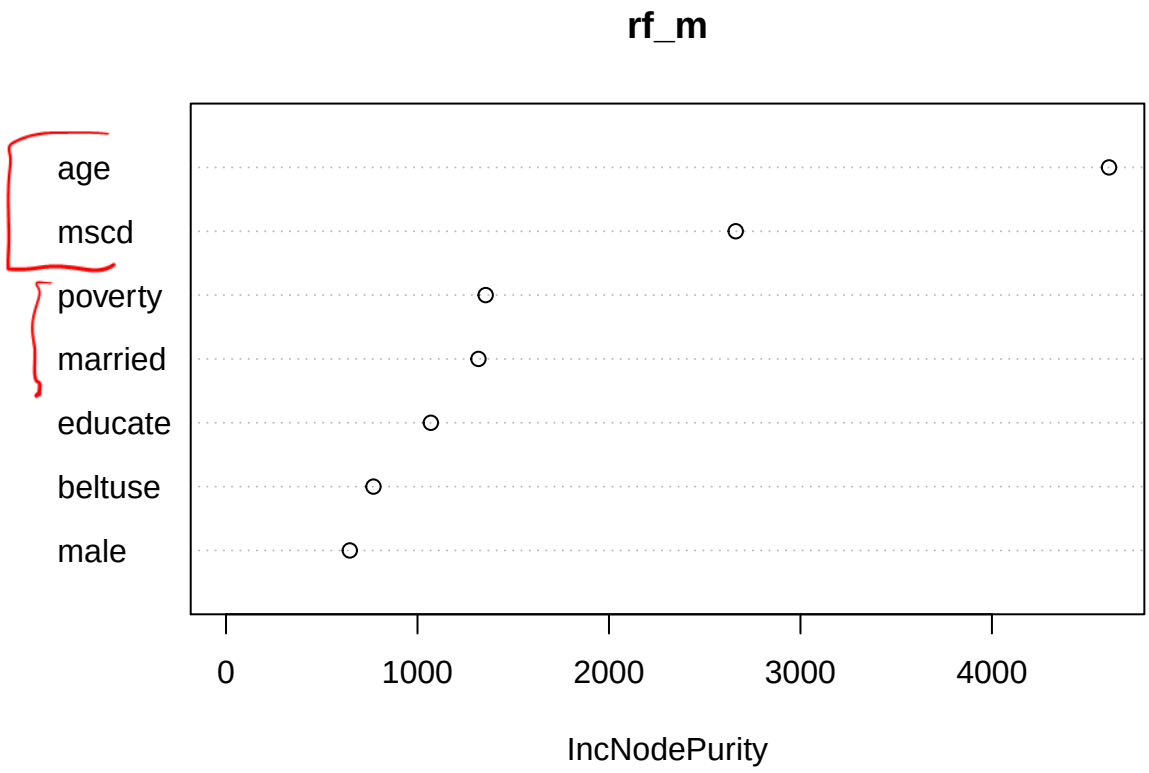
1. Variable importance

You can view or plot the variable importance:

```
## Tabular output
rf_m$importance
```

```
##          IncNodePurity
## age         4611.5887
## mscd        2661.9501
## beltuse      770.0297
## educate     1069.6653
## married     1318.3083
## poverty     1356.0329
## male         646.5250
```

```
## Make a figure
varImpPlot(rf_m)
```



2. Out-of-bag predictions

When creating the random forest for a linear/continuous outcome, you can summarize the out-of-bag samples by looking at the frequency that observations were out-of-bag and the predicted values (average predicted value from out-of-bag trees)

```
head(cbind(rf_m$oob.times,rf_m$predicted))
```

```
##       [,1]     [,2]
## 1944   189 6.701802
## 12601  174 4.100149
## 9335   178 5.533686
## 13328  193 4.813766
## 4910   169 6.019639
## 998    191 5.028437
```

You can also get out-of-bag predictions using the predict command:

```
predict_oob = predict(rf_m,type="response")
head(cbind(rf_m$predicted,predict_oob))
```

```
##                predict_oob
## 1944  6.701802    6.701802
## 12601 4.100149    4.100149
## 9335  5.533686    5.533686
## 13328 4.813766    4.813766
## 4910  6.019639    6.019639
## 998   5.028437    5.028437
```

→ different than predict(rf_m, newdata = dat.train, type = "response")

3. Make predictions for new observations

You can use the predict command to make predictions for observations outside of your training data.

```
predict_test = predict(rf_m,newdata=dat.test[,2:8],type="response")

### From here you can compute the MSE of prediction
mean((dat.test[,1]-predict_test)^2)
```

```
## [1] 5.686272
```

```
### You can compare with other prediction models
mse.tree.test
```

```
## [1] 5.806413
```

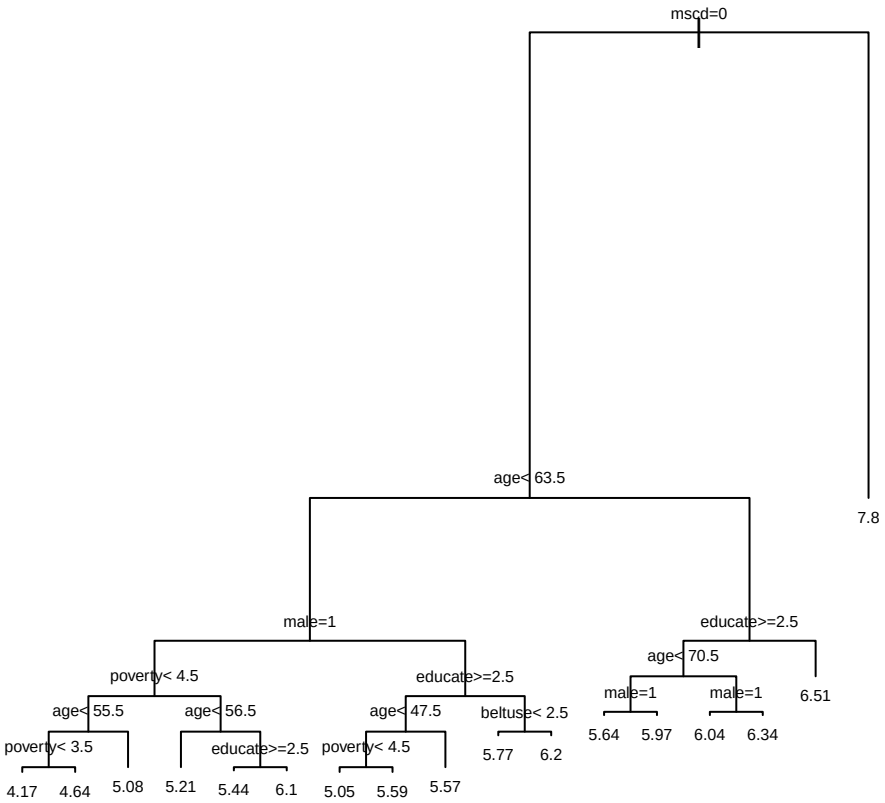
In this case, we only get a slight improvement in the prediction using the random forest.

4. Summary average prediction tree

One feature of random forests that is unsatisfying is that you don't know what any of the trees look like. You could look at the individual trees if you like OR you could compute a summary average prediction tree.

You can read more about this here: https://medinform.jmir.org/2020/6/e15791/

```
dat.train$predicted = rf_m$predicted
sumtree=rpart(predicted~.,data=dat.train[,2:9],method="anova",control=rpart.control(minsize=20,cp=.003))
## Plot the tree and add labels
plot(sumtree)
text(sumtree,pretty=3,cex=0.5,digits=3)
```



5. Last bit....

You could fit the random forest using all the data because there is an internal cross-validation during the construction of the random forest.

```
rf_all = randomForest(x = dat[,2:8],y = dat[,1],mtry=m,ntree=500,keep.forest = TRUE)
## Out-of-bag MSE of prediction
mean((dat$e - rf_all$predicted)^2)
```

```
## [1] 5.74644
```

```
rf_all$mse[500]
```

```
## [1] 5.74644
```