MSiA 420, HW #2

For all problems for which you use R, include your R script in an appendix to your homework (clearly label which parts of the script correspond to which homework problems).

1) Consider the Ischemic heart disease data described in Appendix C.9 of KNN. The data are in HW2_data.xls. For this and subsequent problems, let the response variable be the log (base 10) of the total cost. There were originally a handful of zeros in the "cost" column, which have been replaced by "1" in the Excel file (otherwise, you can't take the log). Notice that some of the predictors are very heavy tailed, so it may be better to take the log of these predictors up front (although the zeros in the discrete predictors would create problems if you took the log). But for the sake of consistency, do NOT take the log of the predictors for these problems.

   (a) Fit a linear regression model to the ischemic heart disease data. Using any and all arguments that are relevant, discuss how well the model fits the data in terms of its predictive power.

```
> IHD<-read.table('heart_dis1.txt',sep="")
## standardize the predictors
> IHD1<-IHD;IHD1[2:9]<-sapply(IHD1[2:9],function(x) (x-mean(x))/sd(x))
> out<-lm(log10(cost)~.,IHD1)
> summary(out)

Call:
lm(formula = log10(cost) ~ ., data = IHD1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.44852 -0.30093  0.01049  0.28276  1.72581

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.73172    0.01914 142.723  < 2e-16 ***
age         -0.02981    0.01946  -1.532   0.1260
gend        -0.02811    0.01932  -1.455   0.1462
intvn        0.49125    0.02131  23.053  < 2e-16 ***
drugs       -0.02736    0.02274  -1.203   0.2291
ervis        0.05917    0.02389   2.477   0.0135 *
comp         0.08114    0.01971   4.117 4.25e-05 ***
comorb       0.13619    0.02225   6.120 1.48e-09 ***
dur          0.14729    0.02266   6.501 1.43e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 0.5373 on 779 degrees of freedom
Multiple R-squared: 0.5831, Adjusted R-squared: 0.5789
F-statistic: 136.2 on 8 and 779 DF, p-value: < 2.2e-16

Adjusted r^2 = 0.579, i.e., about 58% of the variability in cost is due to its (linear) dependence on the predictors. So the response is somewhat predictable with a linear regression model. But see the discussion for part (c).

(b) Which variables appear to have the most influence on the cost?

Look at VIFs before interpreting t-test P-values:

```
> library(HH);vif(out);detach(package:HH)
    age   gend   intvn   drugs   ervis    comp  comorb     dur
1.032749 1.018121 1.238005 1.409274 1.556264 1.058985 1.349956 1.399433
```

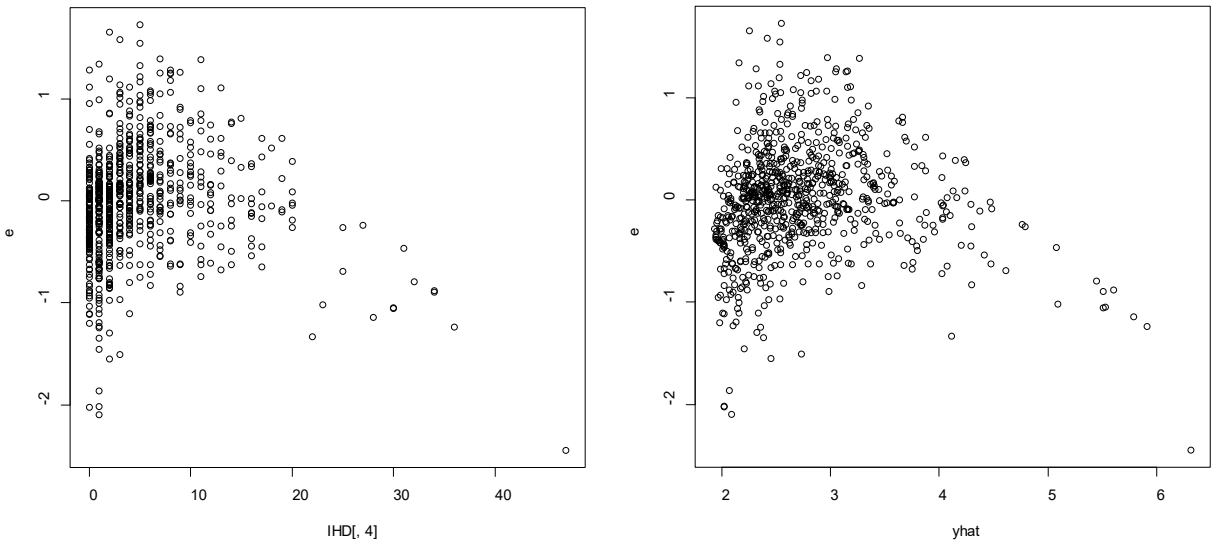
None of the VIFs are large (indicating that multicollinearity is not a problem), so large P-values for the t-tests can be interpreted as the predictor not having a significant effect on the response. Hence, intvn, comp, comorb, and dur all have very statistically significant effects on the response. ervis also has a significant effect, although much less so. Since the predictors have been standardized, we can compare their estimated coefficients directly. The order of importance (as measured by the magnitude of the coefficients) is intvn, dur, comorb, and comp.

(c) Construct appropriate diagnostics and residual plots to assess whether you think there are any problems with the data set that require remedial action or any nonlinearity in the relationship between the response and the predictors.

Plot residuals versus each predictor and versus the predicted response:

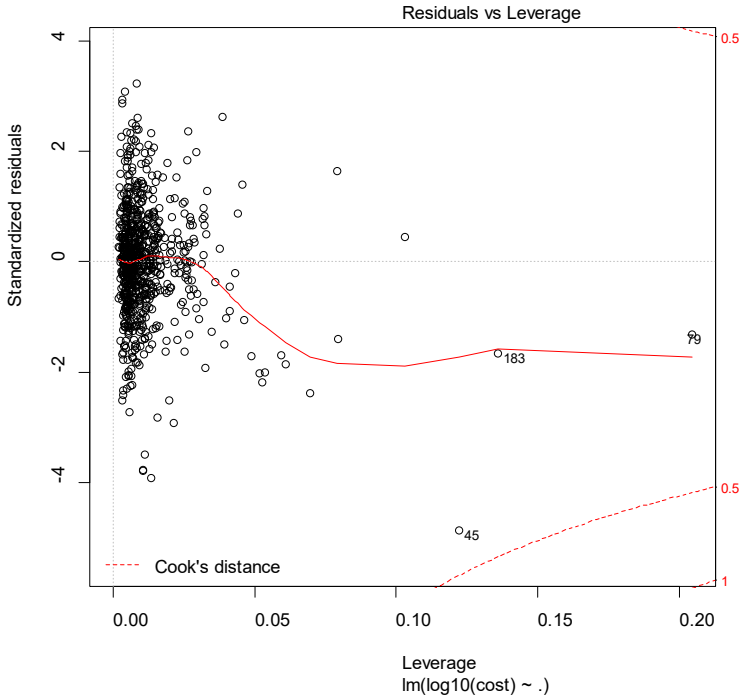
```
> yhat<-predict(out)
> y<-log10(IHD$cost)
> e<-y-yhat
> plot(IHD[,4],e)  #no interesting patterns in e vs. the other predictors (not shown below)
> plot(yhat,e)
```

The plots below show that there is clearly a nonlinearity.



Other diagnostics are produced by:

plot(out)



The last plot produced by plot(out) is shown above. This indicates that a few observations may be high influence, so we should re-do the regression fit without them to see if the conclusions change:

```
> out<-lm(log10(cost)~.,IHD1[-c(45,79,183),]);summary(out)

Call:
lm(formula = log10(cost) ~ ., data = IHD1[-c(45, 79, 183), ])

Residuals:
    Min      1Q  Median      3Q     Max
-2.06859 -0.29250  0.01001  0.29746  1.69176

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.737576   0.018867 145.101  < 2e-16 ***
age         -0.022977   0.019192  -1.197   0.2316
gend        -0.028345   0.019045  -1.488   0.1371
intvn        0.518404   0.021768  23.815  < 2e-16 ***
drugs        0.001948   0.023121   0.084   0.9329
ervis        0.046195   0.023640   1.954   0.0511 .
comp         0.085952   0.021307   4.034 6.03e-05 ***
comorb       0.148417   0.023470   6.324 4.31e-10 ***
dur          0.135649   0.022527   6.022 2.66e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5284 on 776 degrees of freedom
Multiple R-squared:  0.596,     Adjusted R-squared:  0.5919
F-statistic: 143.1 on 8 and 776 DF,  p-value: < 2.2e-16
```

The preceding shows that the conclusions are essentially the same without the three influential observations, so they do not appear to have corrupted the results. But nonlinearity is clearly an issue that needs to be addressed by fitting a nonlinear model (as in the next problems).

2) For this problem, your objective is to find the best neural network model for the ischemic heart disease data. For consistency, use a linear output activation function, and do NOT rescale the response (log of cost) to the [0,1] interval.

   (a) Use 10-fold cross-validation to find the best combination of shrinkage parameter λ and number of hidden nodes.

Standardize all predictors.

```
> IHD1<-IHD;IHD1[2:9]<-sapply(IHD1[2:9],function(x) (x-mean(x))/sd(x))
```

The following is the function defined in lecture for creating a CV index partition

```
> CVInd <- function(n,K) {  #n is sample size; K is number of parts; returns K-length list
of indices for each part
+   m<-floor(n/K)  #approximate size of each part
+   r<-n-m*K
+   I<-sample(n,n)  #random reordering of the indices
+   Ind<-list()  #will be list of indices for all K parts
+   length(Ind)<-K
+   for (k in 1:K) {
+     if (k <= r) kpart <- ((m+1)*(k-1)+1):((m+1)*k)
+       else kpart<-((m+1)*r+m*(k-r-1)+1):((m+1)*r+m*(k-r))
+     Ind[[k]] <- I[kpart]  #indices for kth part of data
+   }
+   Ind
+ }
```

You should do 10-fold CV a number of times (e.g., 20) and average the results. I got the following:

Skip	size	decay	CV SSE
T	10	.1	233
T	10	1	172
T	10	10	206
F	10	1	172
F	10	2	174
F	20	1	172
F	20	5	184
F	5	1	172
F	5	.5	173

decay = 1 appears about the best. Size (in the range 5—20) and Skip do not seem to make much difference.

(b) Fit the final best model and discuss how good you think the model is, in terms of its predictive power.

```
> out<-nnet(log10(cost)~.,IHD1,linout=T,skip=F,size=10,decay=1,maxit=1000,trace=F)
> yhat<-as.numeric(predict(out))
> e<-y-yhat
> c(sd(y),sd(e))
[1] 0.8279264 0.4495331
> 1-var(e)/var(y)
[1] 0.7051915
```

Thus, r^2 is about 71% for the neural network model, which is much higher (better) than linear regression. But this could be due to overfitting, we need to assess the CV SSE for the model. 20 replicates of 10-fold CV give a CV SSE of

171.2 for the neural network, vs
233.6 for the linear regression model

Since SST is:

```
> SST=sum((y-mean(y))^2);SST
[1] 539.4587
```

The CV r^2 for the neural network and linear regression models are:

```
> 1-171.25/SST
[1] 0.6825521
> 1-233.56/SST
[1] 0.5670474
```

Notice that the CV r^2 for the neural network is not much worse than its regular r^2 (68.2% vs 71%), so the shrinkage parameter of decay = 1 did a good job of preventing overfitting.

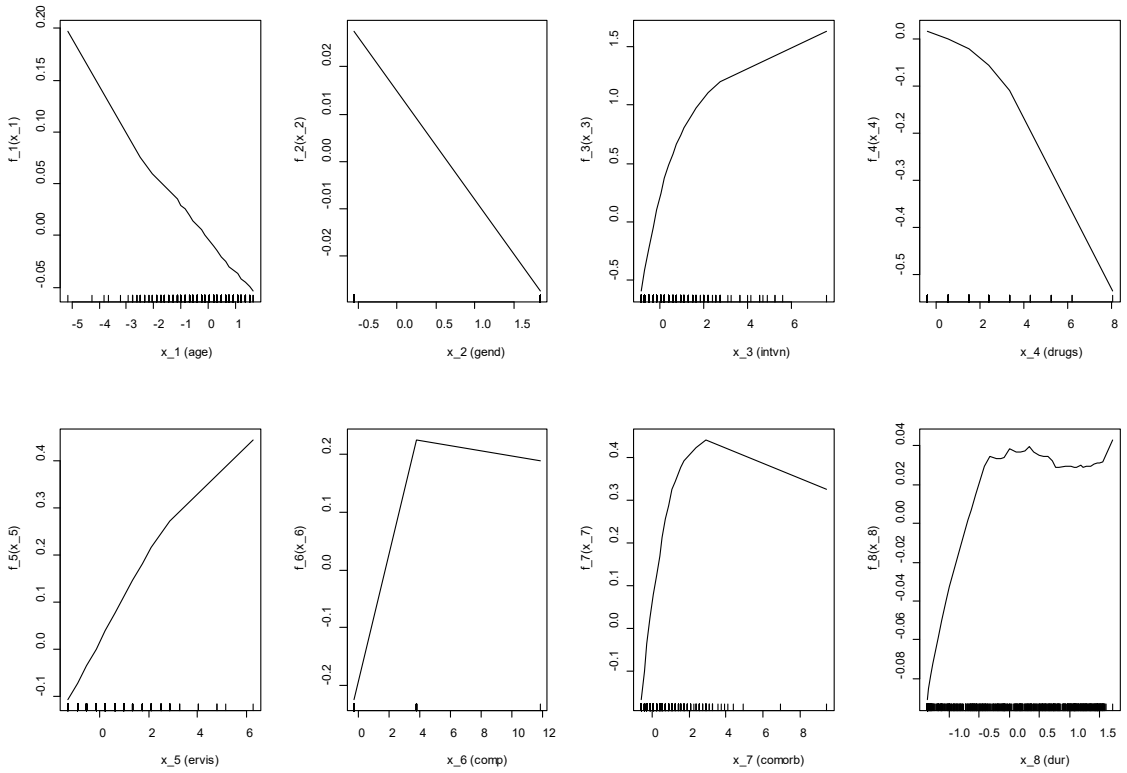
(c) Which variables appear to have the most influence on the cost?

```
> summary(out)
a 8-10-1 network with 101 weights
options were - linear output units  decay=1
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1
 -0.59   0.32   0.23  -0.06  -0.45  -0.28  -0.14  -0.89  -0.43
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2
  0.01   0.03  -0.01   0.34  -0.05   0.20  -0.03  -0.07   0.11
 b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3
  0.01   0.03  -0.01   0.34  -0.05   0.20  -0.03  -0.07   0.11
 b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4
  0.01   0.03  -0.01   0.34  -0.05   0.20  -0.03  -0.07   0.12
 b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5
  0.01   0.02  -0.01   0.34  -0.05   0.20  -0.03  -0.06   0.11
 b->h6 i1->h6 i2->h6 i3->h6 i4->h6 i5->h6 i6->h6 i7->h6 i8->h6
  0.00   0.04  -0.02  -0.09   0.12  -0.14  -0.02  -0.02  -0.14
 b->h7 i1->h7 i2->h7 i3->h7 i4->h7 i5->h7 i6->h7 i7->h7 i8->h7
  0.23  -0.15   0.42  -0.63  -0.16  -0.09  -0.27  -0.27  -1.22
 b->h8 i1->h8 i2->h8 i3->h8 i4->h8 i5->h8 i6->h8 i7->h8 i8->h8
 -0.01   0.26  -0.26   0.87   0.13  -0.15   0.50   0.05  -0.20
 b->h9 i1->h9 i2->h9 i3->h9 i4->h9 i5->h9 i6->h9 i7->h9 i8->h9
  1.54  -0.04   0.07   1.71  -0.24  -0.07   0.14   0.66   0.41
 b->h10 i1->h10 i2->h10 i3->h10 i4->h10 i5->h10 i6->h10 i7->h10 i8->h10
   0.00    0.04   -0.02   -0.09    0.12   -0.14   -0.02   -0.02   -0.14
  b->o  h1->o  h2->o  h3->o  h4->o  h5->o  h6->o  h7->o  h8->o  h9->o h10->o
```

0.06 -1.01 0.47 0.46 0.46 0.46 -0.25 0.93 0.99 2.03 -0.25

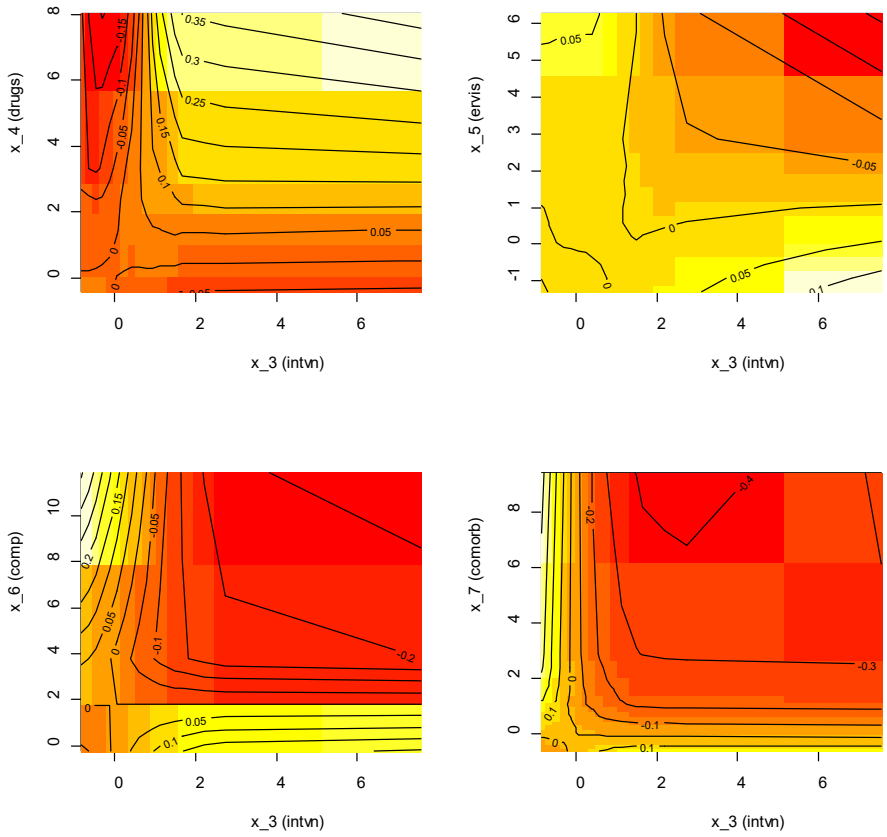
The summary output is impossible to interpret, but ALE plots can be used to understand the effects of the predictors, as follows.

```
> library(ALEPlot)
> yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata))
> par(mfrow=c(2,4))
> for (j in 1:8)  {ALEPlot(IHD1[,2:9], out, pred.fun=yhat, J=j, K=50, NA.plot = TRUE)
+   rug(IHD1[,j+1]) }  ## This creates main effect ALE plots for all 8 predictors
> par(mfrow=c(1,1))
```



Based on the main effects, intvn is by far the most important, and drugs, ervis, comorb, and comp are the next most important with all having about the same importance. Let's create ALE plots between intvn and the other four predictors with sizeable main effects:

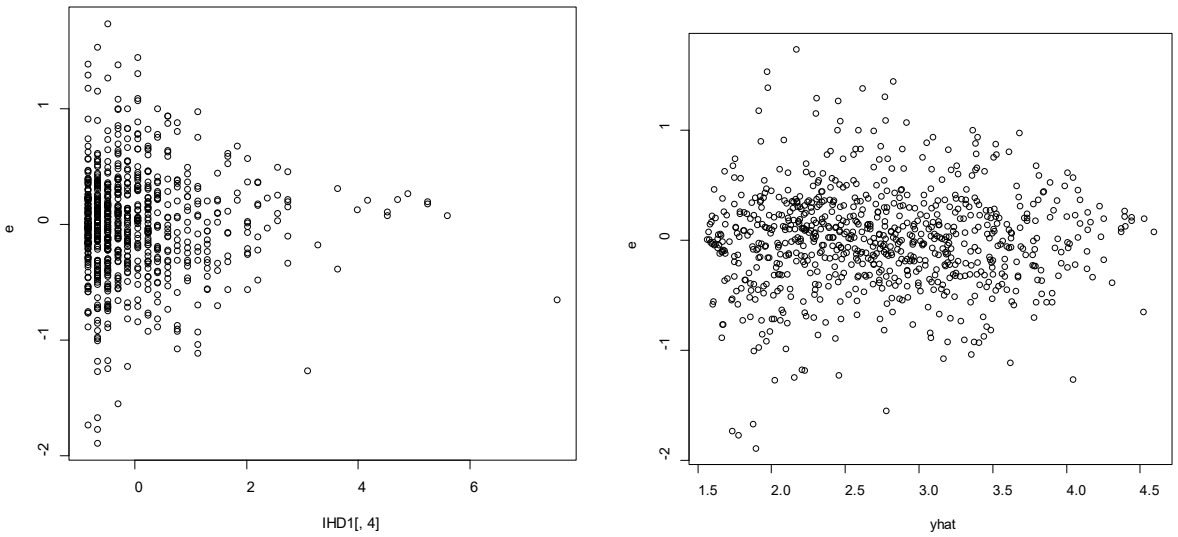
```
> par(mfrow=c(2,2))
> ALEPlot(IHD1[,2:9], out, pred.fun=yhat, J=c(3,4), K=50, NA.plot = TRUE)
> ALEPlot(IHD1[,2:9], out, pred.fun=yhat, J=c(3,5), K=50, NA.plot = TRUE)
> ALEPlot(IHD1[,2:9], out, pred.fun=yhat, J=c(3,6), K=50, NA.plot = TRUE)
> ALEPlot(IHD1[,2:9], out, pred.fun=yhat, J=c(3,7), K=50, NA.plot = TRUE)
> par(mfrow=c(1,1))
```



The magnitude of the interactions are fairly sizeable, relative to the main effects, and so should be considered.

(d) Construct appropriate residual plots to assess whether there remains any linearity not captured by the neural network model.

```
> plot(IHD1[,4],e)
> plot(yhat,e)
```



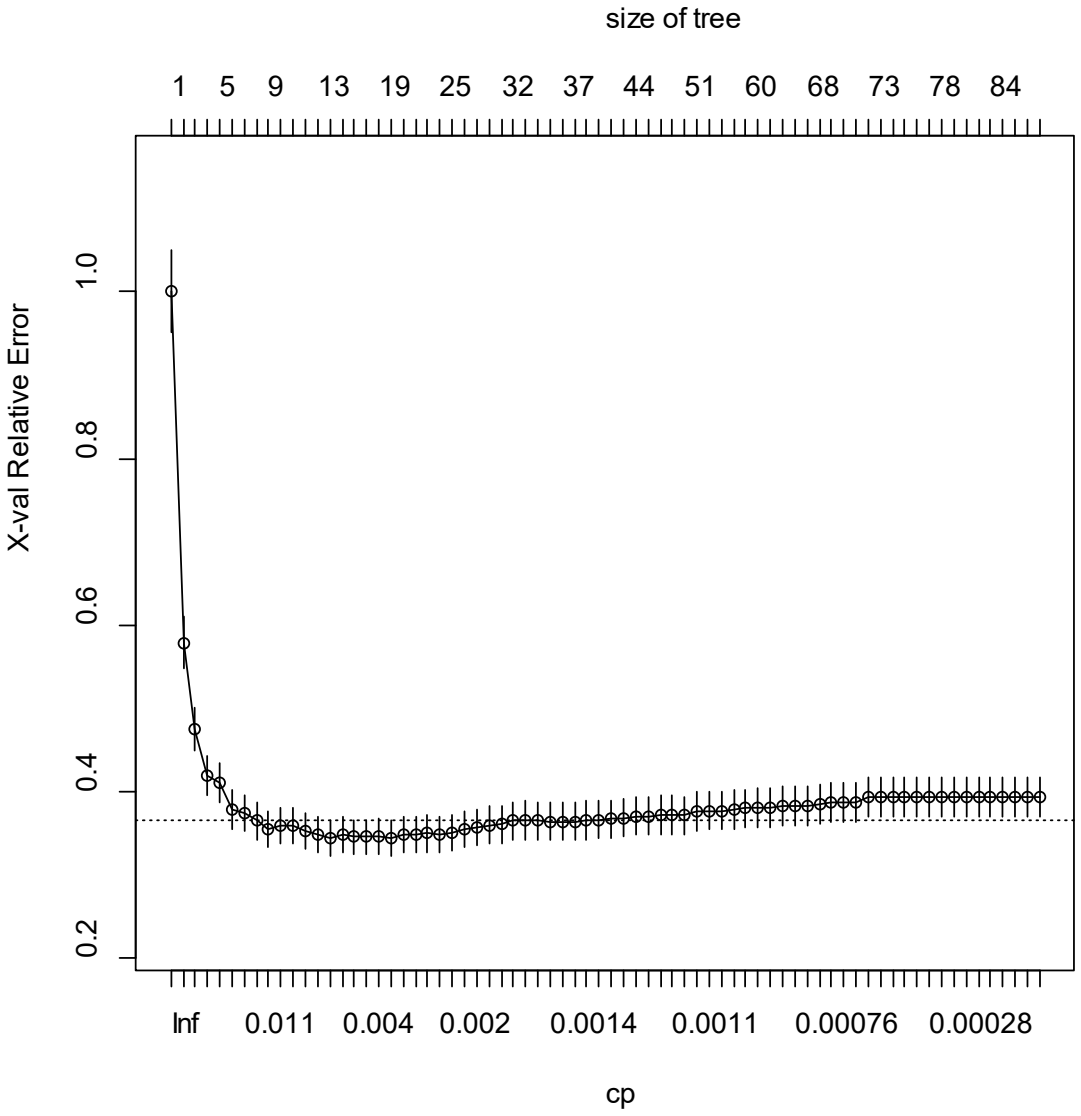
There are no evident nonlinear patterns in the residuals versus the predicted response and the third predictor (show above) or versus any of the other predictors (not shown).

3) Repeat Problem 2, but for a regression tree. Specifically:

(a) Use 10-fold cross-validation to find the best tree size or complexity parameter value.

```
> library(rpart)
> control  <- rpart.control(minbucket = 5, cp = 0.00001, xval = 10, maxsurrogate = 0, usesurrogate = 0) # choose cp as small as you can. This cp corresponds to the largest tree to grow.
> out  <- rpart(log10(cost) ~ ., data = IHD1, method = 'anova', control = control)
>
> plotcp(out)
```

The following is a plot of the CV deviance versus the tree size/complexity parameter, from which we could see that the tree was grown large enough (CV error goes up at the end).



The best cp value could be chosen from the cptable output as follows:

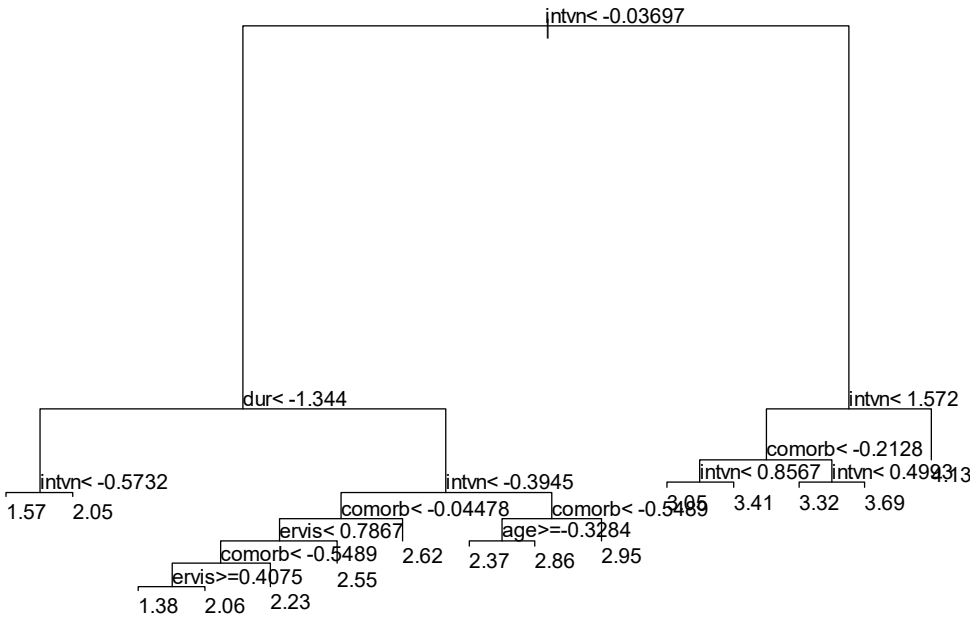
```
> bestcp <- out$cptable[which.min(out$cptable[,"xerror"]),"CP"] # cp parameter with minimal
> bestcp
[1] 0.004090893
```

The minimal CV relative error corresponds to an optimal complexity parameter of roughly 0.0041 (or, equivalently, tree size = 15 terminal nodes).

(b) Fit the final best model and discuss how good you think the model is, in terms of its predictive power.

Use the optimal complexity parameter = 0.0041:

```
> out1 <- prune(out, cp= 0.0041)
> plot(out1); text(out1)
```




```
> y<-log10(IHD1$cost)
> yhat<-as.numeric(predict(out1))
> e<-y-yhat
> c(sd(y),sd(e))
[1] 0.8279264 0.4458116
> 1-var(e)/var(y)
[1] 0.7100526
```

Thus, training r^2 is about 71% for the tree, compared to about 71% for the neural network model. This could also be due to overfitting, so we need to assess the CV SSE for the model. Looking at the output of:

```
> printcp(out1)
```

```
Regression tree:
rpart(formula = log10(cost) ~ ., data = IHD1, method = "anova",
    control = control)

Variables actually used in tree construction:
[1] age    comorb dur    ervis  intvn

Root node error: 539.46/788 = 0.68459
```

n= 788

	CP	nsplit	rel error	xerror	xstd
1	0.4393807	0	1.00000	1.00068	0.049669
2	0.0958190	1	0.56062	0.57851	0.031253
3	0.0582852	2	0.46480	0.47536	0.025675
4	0.0307018	3	0.40652	0.41868	0.024087
5	0.0247526	4	0.37581	0.40972	0.023898
6	0.0116665	5	0.35106	0.37817	0.022900
7	0.0099124	6	0.33939	0.37406	0.022402
8	0.0066471	7	0.32948	0.36448	0.022332
9	0.0062292	8	0.32283	0.35509	0.021741
10	0.0060774	9	0.31661	0.35890	0.021857
11	0.0057761	10	0.31053	0.35903	0.021827
12	0.0054328	11	0.30475	0.35308	0.021266
13	0.0046859	12	0.29932	0.34861	0.020983
14	0.0041000	14	0.28995	0.34364	0.020706

The 10-fold CV $1 - r^2$ of the pruned tree (highlighted above) is approximately 0.34, i.e. 10-fold CV $r^2 \approx 65.6\%$. However, this result is obtained using only a replicate of 10-fold-CV. Doing 20 replicates of 10-fold CV manually, we obtain CV $r^2 \approx 64.4\%$, compared to about 68.2% for the neural network model.

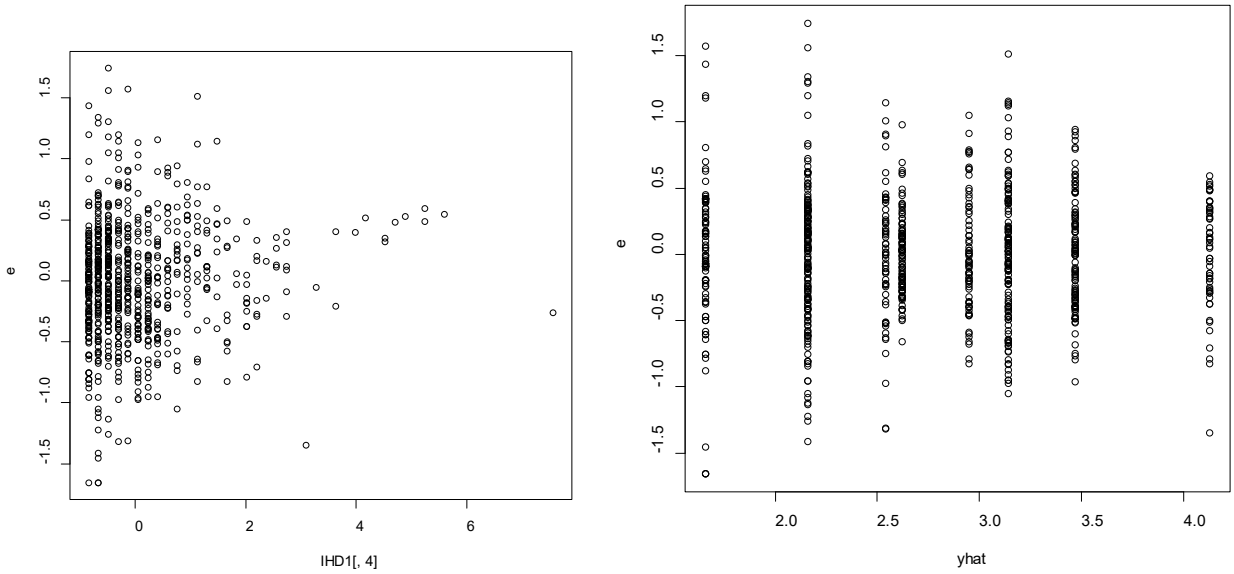
(c) Which variables appear to have the most influence on the cost?

```
> out1$variable.importance
    intvn       dur    comorb      ervis       age
295.094756  51.690372  27.152595   5.827806   3.278480
```

Thus, intv is the most important, followed by dur and comorb. Ervis and age also have minor effects. This can also be gauged by inspection of the plot of the tree, for which intv is the first split (and with long branch length), and dur and comorb also appear relatively high up in the tree structure.

(d) Construct appropriate residual plots to assess whether there remains any linearity not captured by the regression tree model.

```
> plot(IHD1[,4],e)
> plot(yhat,e)
```



There are no evident nonlinear patterns in the residuals versus the predicted response and the third predictor (show above) or versus any of the other predictors (not shown).

Something to think about: Why are the values for yhat in the above discrete, whereas they were not for the neural network or linear regression model?

(e) Which model (the linear regression, neural network, or tree) would you recommend for this data set, and why?

In terms of pure predictive power (as measured by the r^2 from CV), the neural network with size 10 and $\lambda = 1$ is the best (CV $r^2 = 68.2\%$), followed by the tree with size 15 (CV $r^2 = 65.6\%$), followed by for the linear regression model (we expect CV $r^2 <$ training $r^2 = 57.9\%$). In terms of explanatory purposes (interpretability and giving an indication of which variables are most important), the ordering is the opposite.

4) Reconsider the forensic glass data used in lecture. These data are in HW2_data.xls and also posted on Blackboard as a text file (fgl.txt). These are data on fragments of glass in a forensic study. There are n = 214 rows with 10 variables. "type" is a factor with six levels that represents the type of glass, and the other 9 variables are predictor variables that represent the chemical composition of the glass and the refractive index (RI). The levels of type are window float glass (WinF: 70), window non-float glass (WinNF: 76), vehicle window glass (Veh: 17), containers (Con: 13), tableware (Tabl: 9) and vehicle headlamps (Head: 29).This is a classification problem, in which the objective is to classify the categorical response "type" into one of six different glass types. Type library(MASS) and ?fgl to see additional description of the data (which

are in the MASS package). In lecture, we converted the 6-category response into a binary response (window glass or other). In this problem, you will retain the 6-category response and attempt to classify the glass type into one of the six categories.

(a) Use 10-fold cross-validation to find the best neural network model for classifying the class type.

You should do 10-fold CV a number of times (e.g., 20) and average the results. I got the following:

Skip	size	decay	CV Missclassification rate
T	10	.02	29.4%
T	10	.05	28.7%
T	10	.1	28.9%
F	10	.2	31.9%
F	10	1	37%
F	20	.02	28.8%
F	20	.05	27.7%
F	5	.2	30.6%
F	5	.4	32.9%

decay = 0.05 appears about the best, and size = 20 may be slightly better than size = 10. Using these parameters, the final fitted model is:

```
> out<-nnet(type~.,FGL1,linout=F,skip=F,size=20,decay=.05,maxit=1000,trace=F)
> phat<-predict(out,FGL1)
> ind<-apply(phat,1,which.max)
> yhat<-as.factor(levels(y)[ind])
> MISCLASS=sum(y != yhat)/length(y)
> sum(y != yhat)/length(y)
[1] 0.03738318
> summary(out)
a 9-20-6 network with 326 weights
options were - softmax modelling  decay=0.05
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1
 -1.81   0.96   0.42  -0.56  -1.72   0.61  -0.47  -0.05   0.27   2.95
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2
  0.01  -0.48  -1.68   0.27  -0.40  -1.38   0.15   1.62   0.88   0.89
 b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3 i9->h3
  0.80  -1.92  -0.19   2.10  -1.11  -0.83   0.25  -0.91   0.01   1.14
 b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4 i9->h4
  0.94  -2.61   0.42   0.34   1.23  -0.15  -0.28  -0.63  -0.38  -1.67
 b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5 i9->h5
  0.75   0.66   0.20  -0.22   1.06  -2.08   2.43  -1.29   2.60  -0.08
 b->h6 i1->h6 i2->h6 i3->h6 i4->h6 i5->h6 i6->h6 i7->h6 i8->h6 i9->h6
 -1.86  -0.35   0.05  -0.58   0.79   1.04   0.28  -0.57   0.85   0.76
```

b->h7 i1->h7 i2->h7 i3->h7 i4->h7 i5->h7 i6->h7 i7->h7 i8->h7 i9->h7
0.27 0.49 -1.50 1.47 -1.51 0.74 0.54 -1.10 0.55 -0.58
b->h8 i1->h8 i2->h8 i3->h8 i4->h8 i5->h8 i6->h8 i7->h8 i8->h8 i9->h8
0.82 0.84 1.52 1.09 -2.97 0.88 -0.78 -0.33 -0.37 0.64
b->h9 i1->h9 i2->h9 i3->h9 i4->h9 i5->h9 i6->h9 i7->h9 i8->h9 i9->h9
0.10 -1.02 -0.37 -3.02 2.26 -0.09 0.60 0.64 0.27 -1.19
b->h10 i1->h10 i2->h10 i3->h10 i4->h10 i5->h10 i6->h10 i7->h10 i8->h10 i9->h10
-0.44 1.72 -0.42 -0.47 1.38 0.53 1.02 -0.67 1.31 0.58
b->h11 i1->h11 i2->h11 i3->h11 i4->h11 i5->h11 i6->h11 i7->h11 i8->h11 i9->h11
-2.11 -6.41 0.34 0.85 -2.74 -2.02 -1.48 1.90 -0.40 0.96
b->h12 i1->h12 i2->h12 i3->h12 i4->h12 i5->h12 i6->h12 i7->h12 i8->h12 i9->h12
-0.86 1.55 -3.57 0.04 -1.29 0.60 1.45 -0.04 -0.36 -2.75
b->h13 i1->h13 i2->h13 i3->h13 i4->h13 i5->h13 i6->h13 i7->h13 i8->h13 i9->h13
-1.14 1.42 1.57 1.07 1.52 1.11 0.22 -4.99 2.36 -0.59
b->h14 i1->h14 i2->h14 i3->h14 i4->h14 i5->h14 i6->h14 i7->h14 i8->h14 i9->h14
-0.44 1.17 2.11 -0.77 -1.47 1.69 -0.21 -0.72 1.16 -0.03
b->h15 i1->h15 i2->h15 i3->h15 i4->h15 i5->h15 i6->h15 i7->h15 i8->h15 i9->h15
0.58 -0.49 -0.16 -3.55 0.64 1.37 -1.55 2.81 0.62 0.23
b->h16 i1->h16 i2->h16 i3->h16 i4->h16 i5->h16 i6->h16 i7->h16 i8->h16 i9->h16
-0.68 -1.42 -1.98 1.98 0.35 1.37 -0.31 -2.60 -0.65 0.01
b->h17 i1->h17 i2->h17 i3->h17 i4->h17 i5->h17 i6->h17 i7->h17 i8->h17 i9->h17
-1.33 0.08 -1.62 -0.54 -0.31 0.58 1.10 1.28 -0.46 0.61
b->h18 i1->h18 i2->h18 i3->h18 i4->h18 i5->h18 i6->h18 i7->h18 i8->h18 i9->h18
0.57 1.40 1.80 -0.48 -1.35 0.08 -1.13 1.23 -1.06 -1.62
b->h19 i1->h19 i2->h19 i3->h19 i4->h19 i5->h19 i6->h19 i7->h19 i8->h19 i9->h19
1.25 -3.93 -0.92 -2.69 1.15 2.71 2.15 -0.22 -0.29 -0.18
b->h20 i1->h20 i2->h20 i3->h20 i4->h20 i5->h20 i6->h20 i7->h20 i8->h20 i9->h20
-1.20 0.94 1.81 -0.60 -0.01 0.41 1.00 -1.70 1.15 -1.34
b->o1 h1->o1 h2->o1 h3->o1 h4->o1 h5->o1 h6->o1 h7->o1 h8->o1 h9->o1 h10->o1 h11->o1
-0.28 -0.88 1.89 0.19 -1.08 -0.70 1.27 -1.67 -1.76 0.86 1.61 -0.68
h12->o1 h13->o1 h14->o1 h15->o1 h16->o1 h17->o1 h18->o1 h19->o1 h20->o1
0.76 0.19 -0.78 -0.09 -1.64 1.87 -0.28 1.07 -0.10
b->o2 h1->o2 h2->o2 h3->o2 h4->o2 h5->o2 h6->o2 h7->o2 h8->o2 h9->o2 h10->o2 h11->o2
-0.27 -0.64 -2.08 -2.72 -0.94 2.01 1.12 0.96 -1.17 0.08 2.34 -0.45
h12->o2 h13->o2 h14->o2 h15->o2 h16->o2 h17->o2 h18->o2 h19->o2 h20->o2
0.32 1.20 1.68 0.03 -0.46 -0.76 -0.01 -0.58 2.01
b->o3 h1->o3 h2->o3 h3->o3 h4->o3 h5->o3 h6->o3 h7->o3 h8->o3 h9->o3 h10->o3 h11->o3
-0.09 -1.02 -1.35 -0.69 1.36 -1.93 -0.02 -1.21 0.52 0.33 -1.25 0.61
h12->o3 h13->o3 h14->o3 h15->o3 h16->o3 h17->o3 h18->o3 h19->o3 h20->o3
-1.60 -0.69 0.65 1.88 -0.45 -1.40 1.05 0.49 0.66
b->o4 h1->o4 h2->o4 h3->o4 h4->o4 h5->o4 h6->o4 h7->o4 h8->o4 h9->o4 h10->o4 h11->o4
0.14 -1.80 1.99 1.80 0.76 0.57 -1.32 -0.56 -1.96 0.27 -0.82 4.38

```
h12->o4 h13->o4 h14->o4 h15->o4 h16->o4 h17->o4 h18->o4 h19->o4 h20->o4
  -0.60   -2.97    1.72   -1.42    0.67   -1.02    0.72   -0.72   -2.26
  b->o5  h1->o5  h2->o5  h3->o5  h4->o5  h5->o5  h6->o5  h7->o5  h8->o5  h9->o5 h10->o5 h11->o5
   0.09    0.73    0.41    2.45    1.67   -2.35    0.61    2.94    2.56   -3.68   -0.22   -5.10
h12->o5 h13->o5 h14->o5 h15->o5 h16->o5 h17->o5 h18->o5 h19->o5 h20->o5
  -2.37   -2.46   -1.30   -3.20   -1.79    1.65    1.85    3.08    1.67
  b->o6  h1->o6  h2->o6  h3->o6  h4->o6  h5->o6  h6->o6  h7->o6  h8->o6  h9->o6 h10->o6 h11->o6
   0.41    3.61   -0.87   -1.02   -1.76    2.40   -1.66   -0.46    1.81    2.13   -1.66    1.24
h12->o6 h13->o6 h14->o6 h15->o6 h16->o6 h17->o6 h18->o6 h19->o6 h20->o6
   3.49    4.74   -1.97    2.79    3.67   -0.34   -3.34   -3.35   -1.98
> out
a 9-20-6 network with 326 weights
inputs: RI Na Mg Al Si K Ca Ba Fe
output(s): type
options were - softmax modelling  decay=0.05
```

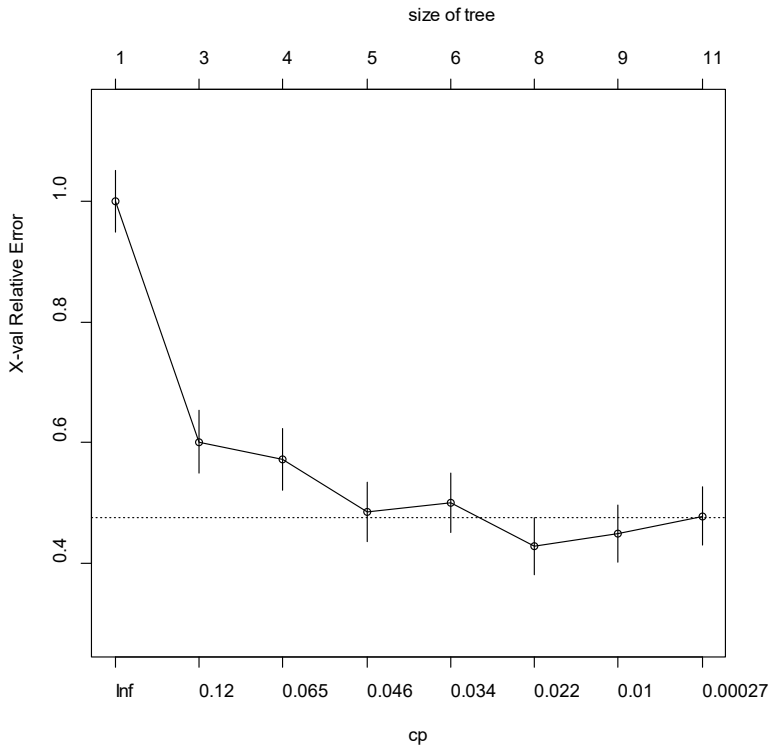
Note that the training misclassification rate was only 3.7% (versus 27.7% for the CV misclassification rate). The 27.7% CV misclassification rate is not too bad, considering that there are 6 response categories. With more response categories, classification is always more challenging.

(b) Use 10-fold cross-validation to find the best classification tree model for classifying the class type.

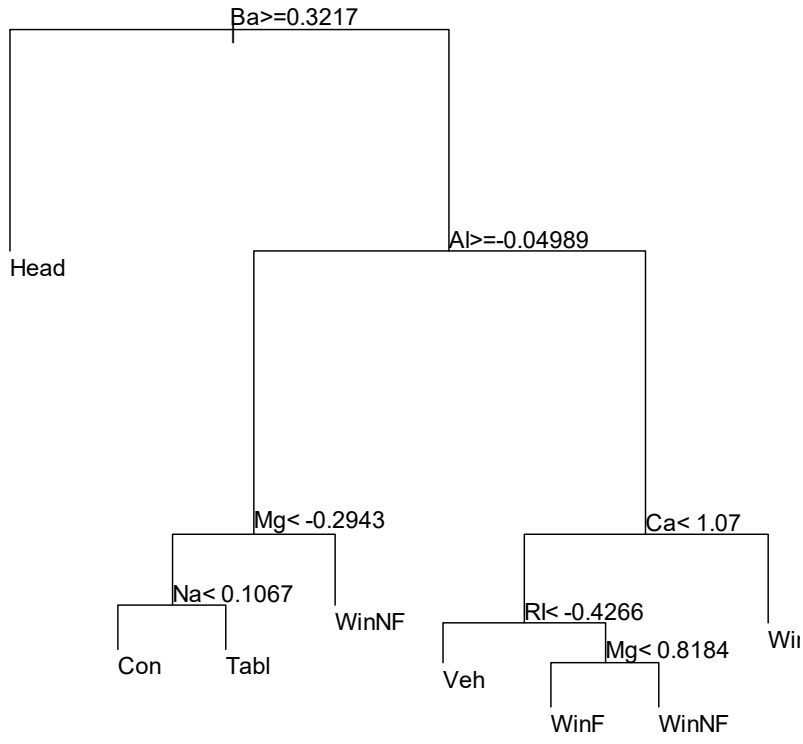
```
> control  <- rpart.control(minbucket = 5, cp = 0.00001, xval = 10, maxsurrogate = 0, usesurrogate = 0)
> out <- rpart(type ~ .,FGL1,control=control)
> plotcp(out)
```

The following is a plot of the CV relative misclassification rate versus the tree size/complexity parameter. The tree was not grown to the minimal cp value we set above (0.00001), probably due to the minbucket parameter. CV error goes up at the end; hence the tree was grown large enough. The best (minimal CV deviance) cp parameter can be obtained by:

```
> bestcp <- out$cptable[which.min(out$cptable[,"xerror"]),"CP"] # cp with minimal deviance
> bestcp
[1] 0.01449275
```




```
> out1<-prune(out, cp=0.0145)
> plot(out1); text(out1,digits=3)
```



```
> printcp(out1)

Classification tree:
rpart(formula = type ~ ., data = FGL1, control = control)

Variables actually used in tree construction:
[1] Al Ba Ca Mg Na RI

Root node error: 138/214 = 0.64486

n= 214

        CP nsplit rel error  xerror     xstd
1 0.206522      0   1.00000 1.00000 0.050729
2 0.072464      2   0.58696 0.60145 0.051652
3 0.057971      3   0.51449 0.57246 0.051156
4 0.036232      4   0.45652 0.48551 0.049160
5 0.032609      5   0.42029 0.50000 0.049548
6 0.014500      7   0.35507 0.42754 0.047370

> y<-FGL1$type
> phat<-predict(out1)
> ind<-apply(phat,1,which.max)
> yhat<-as.factor(levels(y)[ind])
> sum(y != yhat)/length(y)
[1] 0.228972
```

The training misclassification rate is 22.9%. Note that the CV error reported by the printcp() function is the relative misclassification rate (to the root node), and hence, the CV misclassification rate is 0.42754 × 0.64486 = 27.6%. However, this result was obtained by only a replicate of CV. Doing manually 10-fold CV with 20 replicates, the CV misclassification rate of the tree is about 30%.

(c) An alternative to a neural network or classification tree is to use nominal logistic regression, which is like the binary logistic regression with which you are familiar from IEMS 304, except that it applies when you have more than two response categories. Sometimes this is also referred to as "multinomial" regression or "polytomous logistic regression". See Section 14.11 of KNN and/or Section 4.4 of HTF for descriptions of the approach. In R, you can use the multinom() function (which is part of the nnet package) to fit a multinomial model. Fit a multinomial model and discuss the results.

```
> y<-FGL1$type
> out<-multinom(type~.,FGL1,trace=F)
```

```
> summary(out)
Call:
multinom(formula = type ~ ., data = FGL1, trace = F)

Coefficients:
     (Intercept)       RI       Na       Mg       Al       Si        K       Ca
Head   -71.440830 62.713749 100.513048  52.282376  38.68711  81.606612  43.433806
17.722863
Tabl  -127.475167 44.743678  84.443201  41.691749  39.92880  59.731812 -77.755795
-2.592969
Veh      2.705392 -5.497787  -8.322113  -5.538671 -13.40721 -13.038247  -7.405909 -
9.431868
WinF     4.918660 -1.010151  -5.756843  -1.630685 -12.10653  -8.141071  -3.092023 -
8.230390
WinNF    6.588443 -0.291301  -9.333402 -11.516493 -12.42484 -12.135504  -6.237797 -
16.029211
           Ba          Fe
Head   31.414945 -39.55026899
Tabl  -45.974856 -55.58666065
Veh    -4.535127   0.03794772
WinF   -2.141032   0.14213212
WinNF  -5.605689   0.34595753

Std. Errors:
     (Intercept)        RI        Na        Mg        Al        Si         K        Ca
Head         109.702616   252.484114   329.25009   123.67811   154.473955   278.68909
115.494751  66.05251
Tabl   207.136827 103.658893 131.03074   40.93399   79.752400   67.46635 128.457533
130.86862
Veh      2.548394    3.217843   11.19879   18.92422    8.209148   11.67762    9.112509
19.25846
WinF     2.342871    3.003850   11.16484   18.75152    8.163209   11.62350    9.042013
19.11765
WinNF    2.264480    2.903164   11.03730   18.50888    8.086915   11.52415    8.855361
18.88592
           Ba         Fe
Head   75.960564 178.829213
Tabl  271.018164 492.477434
Veh     7.424995   1.074113
WinF    7.177962   1.038150
WinNF   7.013184   1.017553

Residual Deviance: 242.6955
AIC: 342.6955
> phat<-predict(out, type="probs")
> ind<-apply(phat,1,which.max)
```

```
> yhat<-as.factor(levels(y)[ind])
> sum(y != yhat)/length(y)
[1] 0.2616822
>
```

From the preceding, the training misclassification rate is 0.26, whereas the CV misclassification rate is (from 20 replicates of 10-fold CV) 38.0%

(d) Compare the three models from parts (a)—(c) in terms of their predictive ability and interpretability. Which model do you think is the most appropriate for predicting glass type?

In terms of predictability, the models are, in order of best to worst, the neural network model with $\lambda = 0.05$ and size = 20 (27.7%), the classification tree with 8 terminal nodes (CV misclassification rate = 30%), and the multinomial model (CV misclassification rate = 38.0%).

In terms of interpretability, the tree is probably the most interpretable (easy to see that Mg is the most important predictor variable). The multinomial model is still rather difficult to interpret, because, unlike in binary logistic regression, there is more than one set of coefficients.