MSiA 420, HW #3

1) Reconsider the ischemic heart disease data that you analyzed in HW2. As in HW2, for this and subsequent problems, let the response variable be the log (base 10) of the total cost, and do NOT take the log of the predictors. This problem involves using nearest neighbors to predict cost.

   (a) Use *n*-fold CV to find the best *K* for predicting cost using K-NN. What are the pros and cons of using *n*-fold CV, versus say 10-fold CV, for nearest neighbors.

```
IHD<-read.table('heart_dis1.txt',sep="")
IHD1<-IHD;IHD1[2:9]<-sapply(IHD1[2:9],function(x) (x-mean(x))/sd(x))
library(yaImpute)
```

n-fold CV gives using the ann() function gives:

K	CV SSE
1	391
5	233
8	223
10	224
12	229
20	234

The best K appears to be about 8

n-fold CV always gives a more accurate estimate of the test error variance than K-fold for K < n. With most methods, and with large n, n-fold CV takes far longer than 10-fold CV, because you have to train n separate models. With nearest neighbors, however, since there is no training phase, n-fold CV is no more computationally expensive than K-fold CV.

   (b) For the optimal *K* from part (a), what is the CV estimate of the prediction error standard deviation?

```
> sqrt(223/788)
[1] 0.5319727
```

   (c) What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
> test<-c(59, 0, 10, 0, 3, 0, 4, 300)
> xbar<-apply(as.matrix(IHD[,2:9]),2,mean)
> std<-apply(as.matrix(IHD[,2:9]),2,sd)
> test<-(test-xbar)/std  #must standardize
```

```
> test<-matrix(test,1,8)
> K<-8
> out<-ann(as.matrix(IHD1[,2:9]),test,K,verbose=F)
> ind<-matrix(out$knnIndexDist[,1:K],nrow(test),K1)
> yhat<-mean(log10(IHD1$cost[ind]))
> ind  #indices of 8 nearest neighbors
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]  278  717  778  517   37  659  114  456
> yhat
[1] 3.379489
```

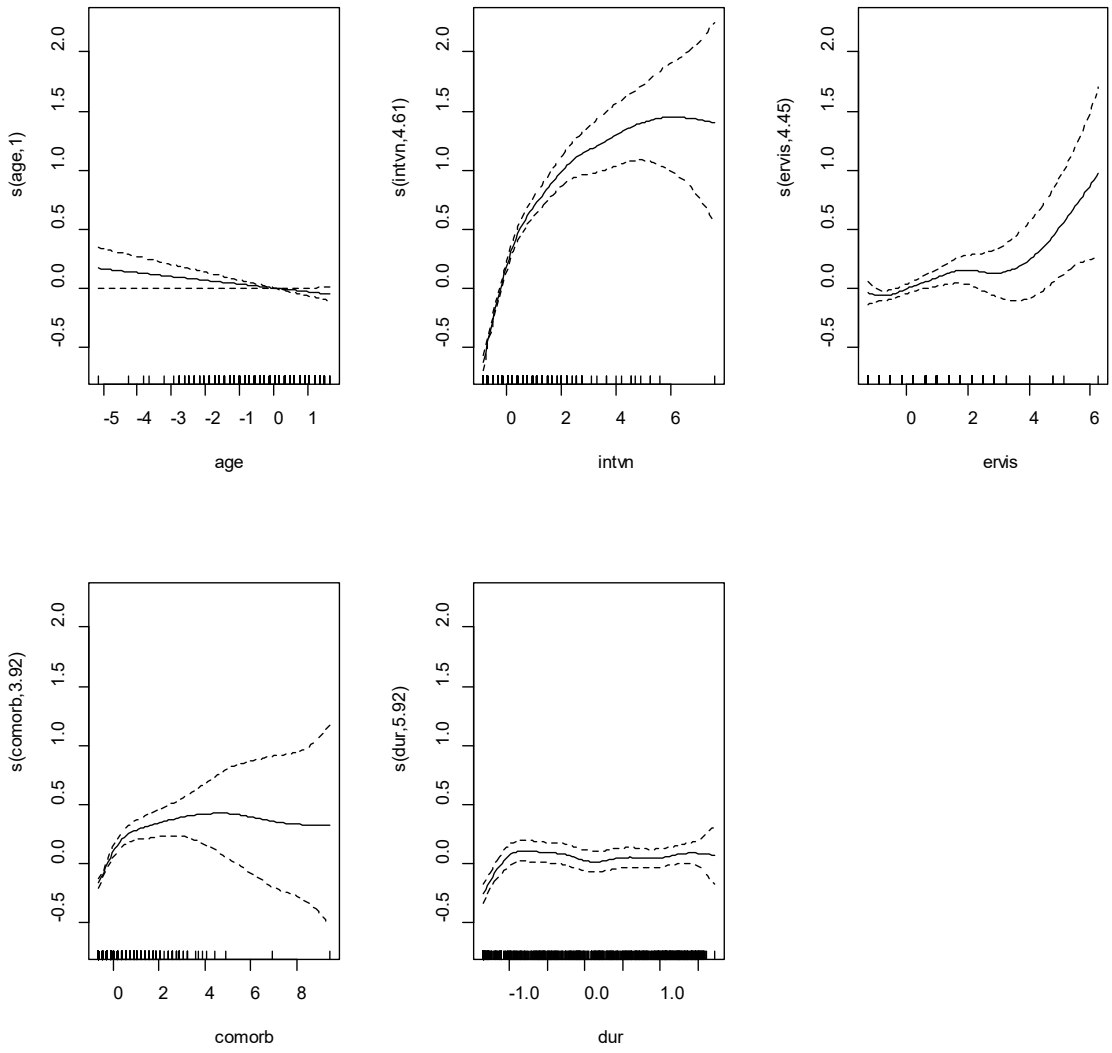
2) This problem involves fitting a generalized additive model to predict cost for the ischemic heart disease data, for which you can use the gam() function of package:gmcv.

   (a) Fit a GAM model without interactions, and construct plots of the component functions. Which predictors appear to be the most relevant for predicting cost?

Note that since gend, drugs, and comp are discrete with only a very few categories, it is not possible to fit a nonparametric smoother to them, and so the gam() function will give an error. To include them in the model as linear terms, you use the formula below.

```
> library(mgcv)
> out<-gam(log10(cost)~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp + s(comorb)
+ s(dur), data=IHD1, family=gaussian())
> summary(out)

Family: gaussian
Link function: identity

Formula:
log10(cost) ~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp +
    s(comorb) + s(dur)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.73172    0.01656 164.919  < 2e-16 ***
gend        -0.02975    0.01685  -1.766   0.0777 .
drugs       -0.02904    0.02074  -1.400   0.1619
comp         0.07184    0.01729   4.154 3.63e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
          edf Ref.df     F  p-value
s(age)  1.000  1.000 3.617  0.05757 .
```

```
s(intvn)  4.610  5.563 136.168  < 2e-16 ***
s(ervis)  4.450  5.435   3.101  0.00729 **
s(comorb) 3.924  4.809  17.023 2.90e-15 ***
s(dur)    5.917  7.043   5.514 3.17e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.685   Deviance explained = 69.4%
GCV score = 0.22296  Scale est. = 0.2162    n = 788
> par(mfrow=c(2,3))
> plot(out)  #plot component functions
```



The four most significant predictors in terms of the P-values are intvn, comorb, dur, and comp, which agrees with the linear regression results from HW2. The plots of the

function components is consistent with intvn, comorb, and dur being the most important predictors.

(b) For the model from part (a), what is the CV estimate of the prediction error standard deviation? What are the pros and cons of using *n*-fold CV, versus say 10-fold CV, for GAMs?

```
> par(mfrow=c(2,3))
> plot(out)  #plot component functions
> Nrep<-1 #number of replicates of CV
> K<-nrow(IHD1)  #K-fold CV on each replicate
> n=nrow(IHD1)
> y<-log10(IHD1$cost)
> SSE<-matrix(0,Nrep,1)
> for (j in 1:Nrep) {
+   Ind<-CVInd(n,K)
+   yhat1<-y;
+   yhat2<-y;
+   for (k in 1:K) {
+         out<- gam(log10(cost)~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp +
s(comorb) + s(dur), data=IHD1[-Ind[[k]],], family=gaussian())
+      yhat1[Ind[[k]]]<-as.numeric(predict(out,IHD1[Ind[[k]],]))
+   } #end of k loop
+   SSE[j,]=sum((y-yhat1)^2)
+ } #end of j loop
> SSE
          [,1]
[1] 177.5628
> sqrt(177.56/n)
[1] 0.4746893
>
```

As with most methods (except nearest neighbors and local kernel weighting methods), n-fold CV takes much longer than K-fold for K < n. In this case, 20 replicates of 10-fold CV is much faster. n-fold CV always gives a more accurate estimate of the test error variance than K-fold for K < n, but the slightly better accuracy with n-fold may not outweigh the much larger computational expense.

(c) What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
> out<-gam(log10(cost)~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp + s(comorb)
+ s(dur), data=IHD1, family=gaussian())
> test<-c(59, 0, 10, 0, 3, 0, 4, 300)
> xbar<-apply(as.matrix(IHD[,2:9]),2,mean)
> std<-apply(as.matrix(IHD[,2:9]),2,sd)
```

```
> test<-(test-xbar)/std  #must standardize
> X<-data.frame(matrix(test,1,8))
> names(X)<-names(IHD1[2:9])
> yhat<-predict(out,newdata=X)
> yhat
       1
3.556478
```

3) This problem involves using kernel methods to predict cost for the ischemic heart disease data. Since the loess() function in R allows at most four predictors, and the ischemic heart disease data involves eight predictors, you will have to choose a good subset of four predictors to work with. From HW2, the best regression tree had the three predictors intvn, comorb, and dur, so these three should probably be in the model. These three also had the smallest P-values from the linear regression fit from HW2 and from the GAM fit in problem 2 above, and the VIFs were all small, so the other larger P-values are not distorted by multicollinearity. The predictor comp had the next smallest P-value in HW2 and in the GAM fit above, but it is discrete, which will cause an error in loess(). So as the fourth predictor for this problem, use ervis, which appears to be the next most relevant predictor from the GAM fit above.

   (a) Use CV to find the best combination of span and degree (0 for local average, 1 for local linear, and 2 for local quadratic regression) for a kernel method.

Using 20 replicates of 10-fold CV (n-fold CV takes quite a while):

degree	span	CV SSE
1	.1	180
1	.2	175
1	.3	174
1	.4	174
1	.5	175
1	.6	176
1	.8	178
0	.01	281
0	.03	181
0	.05	177
0	.07	178
0	.1	179
0	.3	205
0	.6	250
2	.4	181
2	.7	177
2	.8	175
2	.95	176

So the best {span,degree} = {0.4,1}

(b) Use C_p to find the best combination of span and degree (0 for local average, 1 for local linear, and 2 for local quadratic regression) for a kernel method. Is this in agreement with what CV said was the best span and degree?

Use the following to calculate Cp:

```
y<-log10(IHD1$cost)
out<-loess(log10(cost)~intvn+comorb+dur+ervis,IHD1,degree=1,span=0.1)
var_hat<-(out$s)^2  #estimate of residual error std for low-bias model (with smaller span)
out<-loess(log10(cost)~intvn+comorb+dur+ervis,IHD1,degree=1,span=0.3)
yhat<-predict(out)
SSE<-sum((y-yhat)^2)
Cp<-SSE/n+2*out$trace.hat*var_hat/n
Cp
```

degree	span	Cp
1	.1	
1	.2	.228
1	.3	.225
1	.4	.224
1	.5	.224
1	.6	.226
1	.8	
0	.01	
0	.03	
0	.05	.243
0	.07	.248
0	.1	.255
0	.3	
0	.6	
2	.4	
2	.7	
2	.8	.242
2	.95	.245

This says the best {span,degree} = {0.4 or 0.5,1}, which is in agreement with CV from part (a)

(c) For the optimal model from part (a), what is the CV estimate of the prediction error standard deviation?

From above, SSE = 174, so

```
> sqrt(174/788)
[1] 0.4699066
```

(d) What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
> out<-loess(log10(cost)~intvn+comorb+dur+ervis,IHD1,degree=1,span=0.4)
> test<-c(59, 0, 10, 0, 3, 0, 4, 300)
> xbar<-apply(as.matrix(IHD[,2:9]),2,mean)
> std<-apply(as.matrix(IHD[,2:9]),2,sd)
> test<-(test-xbar)/std  #must standardize
> X<-data.frame(matrix(test,1,8))
> names(X)<-names(IHD1[2:9])
> yhat<-predict(out,newdata=X)
> yhat
       1
3.643884
```

4) This problem involves using projection pursuit regression to predict cost for the ischemic heart disease data, for which you can use the ppr() function.

(a) Use CV to find the best number of terms for the PPR model.

20 replicates of 10-fold CV gives

nterms	CV SSE
1	172
2	174
3	178
40	181

The best nterms appears to be 1.

(b) For the optimal model from part (a), what is the PPR model and the CV estimate of the prediction error standard deviation? Interpret the fitted model.

```
> out<-ppr(log10(cost)~., data=IHD1, nterms=1)
> summary(out)
Call:
ppr(formula = log10(cost) ~ ., data = IHD1, nterms = 1)

Goodness of fit:
 1 terms
165.5263
```

Projection direction vectors:

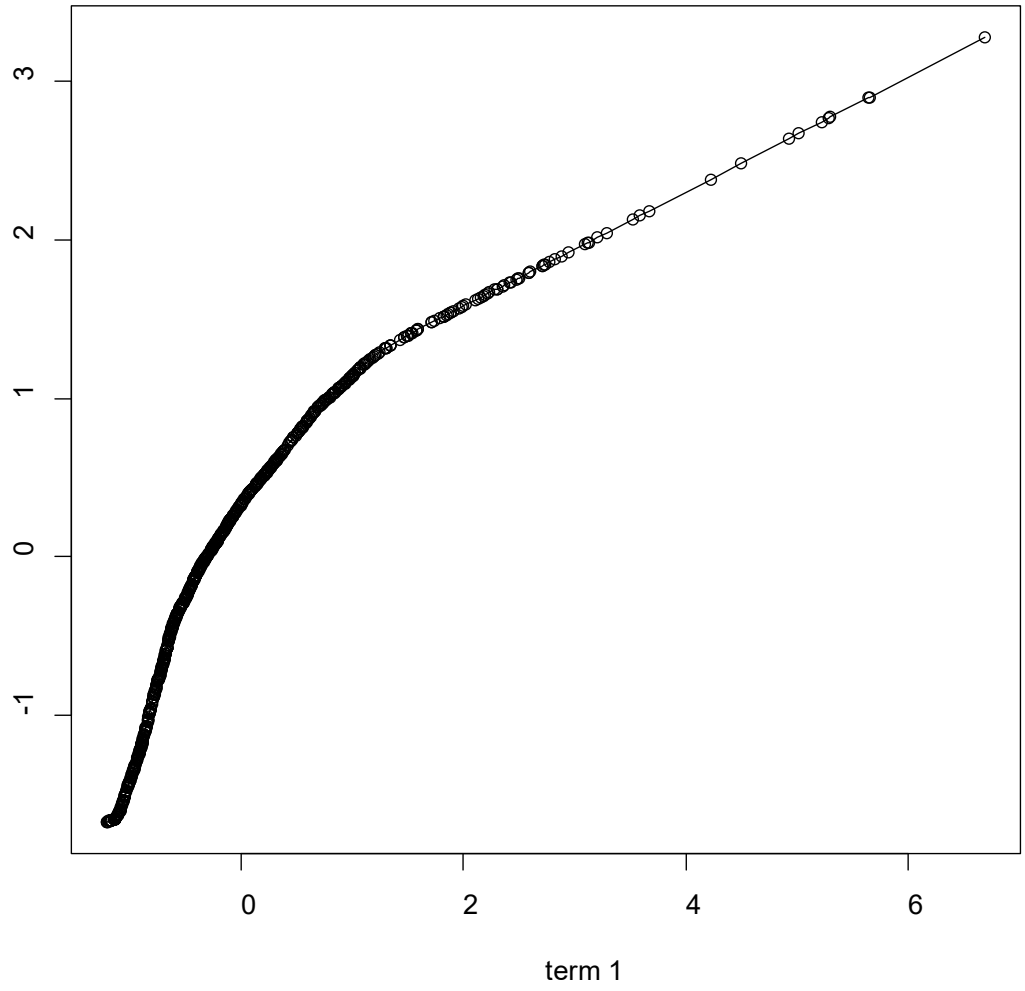
age	gend	intvn	drugs	ervis	comp	comorb	dur
-0.02759078	-0.02686806	0.93275393	-0.07100287	0.10038411	0.15520531	0.28667863	0.08423339

Coefficients of ridge terms:

term 1
0.6888639

```
> par(mfrow=c(1,1))
> plot(out)
```

The above shows the projection direction coefficients (the β's from class), and below shows the shape of the nonparametric component function. The largest direction coefficients are for intvn, comorb, ervis, and dur, which agrees with the previous results.



```
> sqrt(172/788)
[1] 0.4671982
```

(c) What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
> test<-c(59, 0, 10, 0, 3, 0, 4, 300)
> xbar<-apply(as.matrix(IHD[,2:9]),2,mean)
> std<-apply(as.matrix(IHD[,2:9]),2,sd)
> test<-(test-xbar)/std  #must standardize
> X<-data.frame(matrix(test,1,8))
> names(X)<-names(IHD1[2:9])
> yhat<-predict(out,newdata=X)
> yhat
       1
3.510637
```

5) Reconsider the forensic glass data that you analyzed in HW2. In HW2 you retained the 6-category response type. But for this problem, you will convert the 6-category response into a binary response type (window glass or other) and perform a binary classification. For consistency, use the misclassification error rate as the CV measure of prediction accuracy.

(a) Use CV to find the best nearest neighbor model for classifying the type.

n-fold CV gives:

K	misclass rate (%) for n-fold CV	misclass rate (%) for 30 reps of 10-fold CV
1	17.8	17.8
3	14.5	15.2
4	16.8	16.8
5	15.9	16.1
6	17.3	16.9
7	15.0	15.5
10	16.8	17.3
15	17.8	17.9
20	17.8	18.4
30	17.8	18.4
50	19.2	19.8
100	31.8	31.2

The best K is somewhere between 3 and roughly 7, which seem to perform similarly.

(b) Use CV to find the best GAM for classifying the type. You can do this using the binomial family in the gam() function.

In R, the GAM function automatically chooses the nonparametric smoother parameters, so we do not need CV to select anything (unless we want to select the best subset of predictors).

Four replicates of 10-fold CV using the following code give a CV misclassification rate of 20.5%, which is worse than the best nearest neighbors model.

```
Nrep<-3 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n=nrow(FGL1)
y<-FGL1$type_bin
MISCLASS<-matrix(0,Nrep,1)
for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  yhat1<-y;
  yhat2<-y;
  for (k in 1:K) {
    out<-gam(type_bin~s(RI)+s(Na)+s(Mg)+s(Al)+s(Si)+s(K)+s(Ca)+s(Ba)+s(Fe),
       data=FGL1[-Ind[[k]],c(1:9,11)], family=binomial())
    phat<-predict(out,FGL1[Ind[[k]],c(1:9,11)],type="response")
    yhat1[Ind[[k]]]<-ifelse(phat>=.5,"Win","Other")
  } #end of k loop
  MISCLASS[j,]=sum(y != yhat1)/length(y)
} #end of j loop
MISCLASS
mean(MISCLASS)
```

(c) Compare the models in parts (a) and (b) with the best neural network model.

Using 20 reps of 10-fold CV:

# nodes	lambda	misclass rate (%)
10	0	19.0
10	.1	15.3
10	.3	15.8
10	.6	14.8
10	2	15.7
10	5	18.5

10	10	31.8
20	.3	15.2
20	.6	15.9
20	2	15.8
20	5	18.6
5	.3	15.5
5	.6	15.0
5	2	15.6
5	5	19.4

The best NN model is roughly 10 nodes and l between 0.1—2.0, but the number of hidden nodes does not have much effect here.

The misclassification rate for the best GAM was about the same as for the best neural network (~ 15%). The best nearest neighbor model had substantially worse misclassification rate (~20%).

6) For this problem, you will fit a boosted tree for predicting (the log of) cost for the same ischemic heart disease data that you analyzed in the previous problems.

   (a) Find the best boosted tree for predicting cost.

You can use gbm() and the built-in CV to find SSE_{CV} and the best tree size:

```
> gbm1 <- gbm(log10(cost)~., data=IHD1, var.monotone=rep(0,8),
    distribution="gaussian", n.trees=1000, shrinkage=0.02, interaction.depth=3,
    bag.fraction = .5, train.fraction = 1, n.minobsinnode = 10, cv.folds = 10,
    keep.data=TRUE, verbose=FALSE)
> best.iter <- gbm.perf(gbm1,method="cv");best.iter
> SSECV<-gbm1$cv.error[best.iter]*nrow(IHD1);SSECV
```

Repeating the preceding for a few different combinations of shrinkage, interaction.depth, and n.trees, and for each combination repeating multiple times to average across multiple CV replicates, give the following:

n.trees	shrinkage	int.depth	CV SSE
1,000	.1	3	171
1,000	.05	3	169
1,000	.02	3	167
1,000	.01	3	168
2,000	.005	3	168
1,000	.02	2	169
1,000	.02	4	168

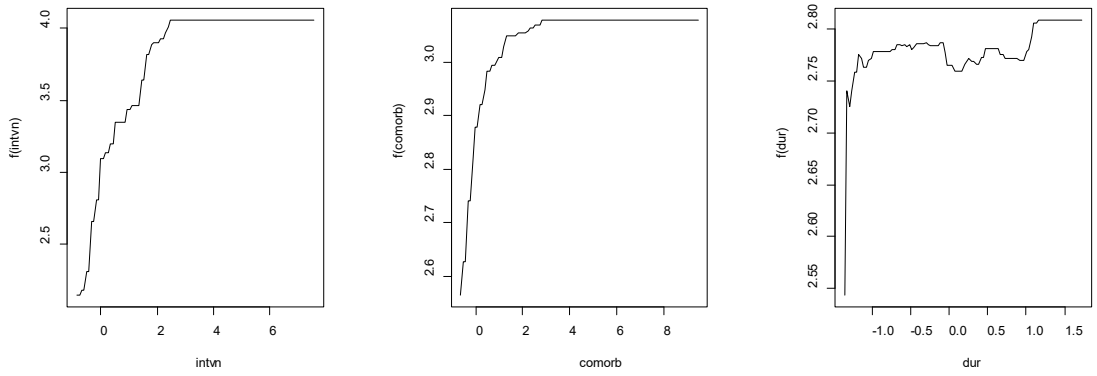
The combination n.trees=1,000, shrinkage=.02, interaction.depth=3 was about the best, although a number of other combinations were close.

(b) Provide an interpretation of which predictors appear to be the most important and what are their effects. Does this agree with the results from the other methods.

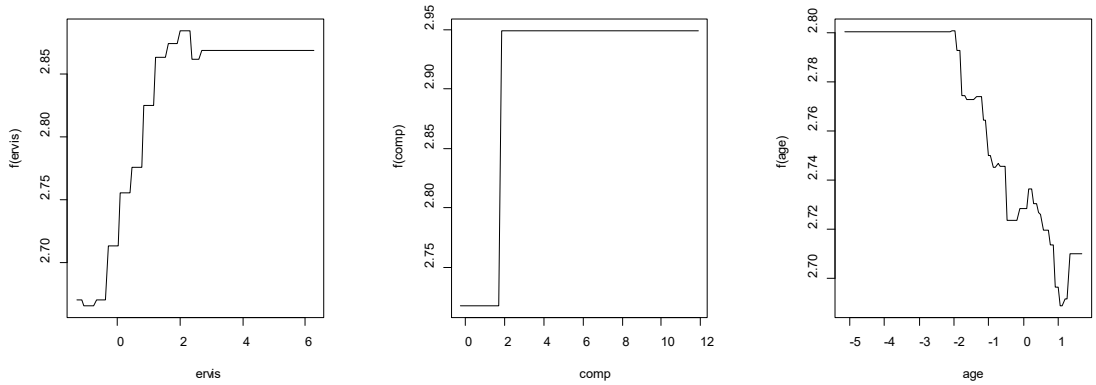
```
> summary(gbm1,n.trees=best.iter)  # based on the optimal number of trees
         var    rel.inf
intvn   intvn 72.2248334
comorb comorb 11.0340824
dur       dur  9.2164965
ervis   ervis  3.7041996
comp     comp  1.5104245
age       age  1.4726935
drugs   drugs  0.4522809
gend     gend  0.3849892
```

The most important variable by far is the number of interventions, followed by the comorbidity (the number of other diseases the subscriber had), and the duration of treatment. Less important are the number of emergency room visits, the number of complications that arose during treatment, and the patient age. The number of drugs prescribed and the gender had little effect on cost. This agrees with the results from the other methods.

```
> par(mfrow=c(1,3))
> for (i in c(3,7,8)) plot(gbm1, i.var = i, n.trees = best.iter)
```




```
> par(mfrow=c(1,3))
> for (i in c(3,7,8)) plot(gbm1, i.var = i, n.trees = best.iter)
```



The variable importance measures and marginal plots largely agree with the results of the other methods (see the plots for the GAM method and the coefficients for the linear regression model and the PPR model).

(c) What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
> test<-c(59, 0, 10, 0, 3, 0, 4, 300)
> xbar<-apply(as.matrix(IHD[,2:9]),2,mean)
> std<-apply(as.matrix(IHD[,2:9]),2,sd)
> test<-(test-xbar)/std  #must standardize
> X<-data.frame(matrix(test,1,8))
> names(X)<-names(IHD1[2:9])
> yhat<-predict(gbm1,newdata=X,n.trees=best.iter);yhat
[1] 3.491332
```

7) For this problem, use the `randomForest` package and function in R to fit a random forest model for predicting (the log of) cost for the same ischemic heart disease data that you analyzed in the previous problems.

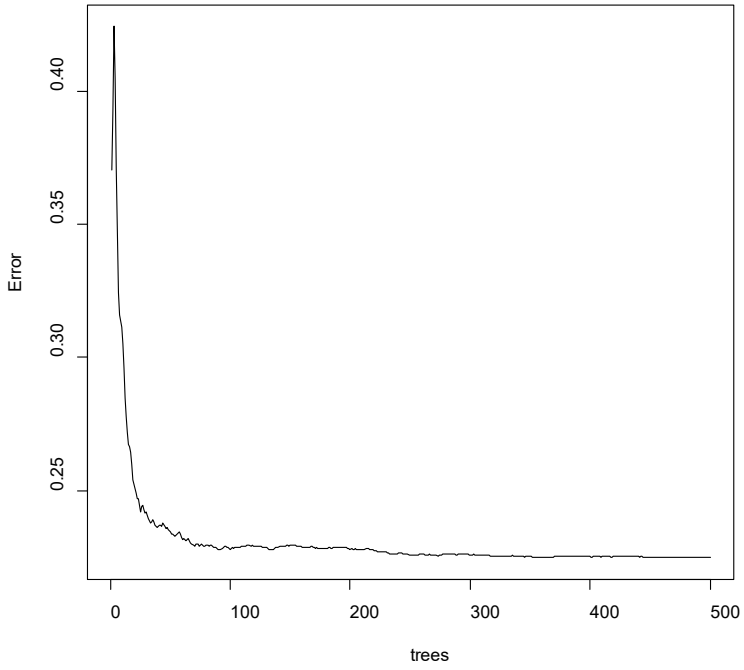
a) Fit a random forest model with `mtry=1` and `ntree = 500`. What is the out-of-bag (OOB) r^2? Note that the OOB r^2 is calculated and interpreted similarly to the CV r^2 and is produced automatically by the `print.randomForest` function. Repeat this a few times to see how much the results change from replicate to replicate, and discuss what you see.

```
library(randomForest)
rf1 <- randomForest(log10(cost)~., data=IHD1, mtry=3, ntree =
   500, importance = TRUE)
plot(rf1)  #plots OOB mse vs # trees
rf1 #check the OOB mse and r^2
```

Across replicates of the above code, typical values for the OOB r^2 are 0.671, 0.668, 0.671, 0.671, 0.673 . . ., and the average OOB r^2 was about 0.671. The average OOB MSE was about 0.226.

b) In part (a), do you think `ntree` was chosen large enough? Explain.

Yes, because the OOB MSE converges to roughly a constant value well before the number of trees is 500. This is seen in the typical results produced, below, that are produced by the command `plot(rf1)`:



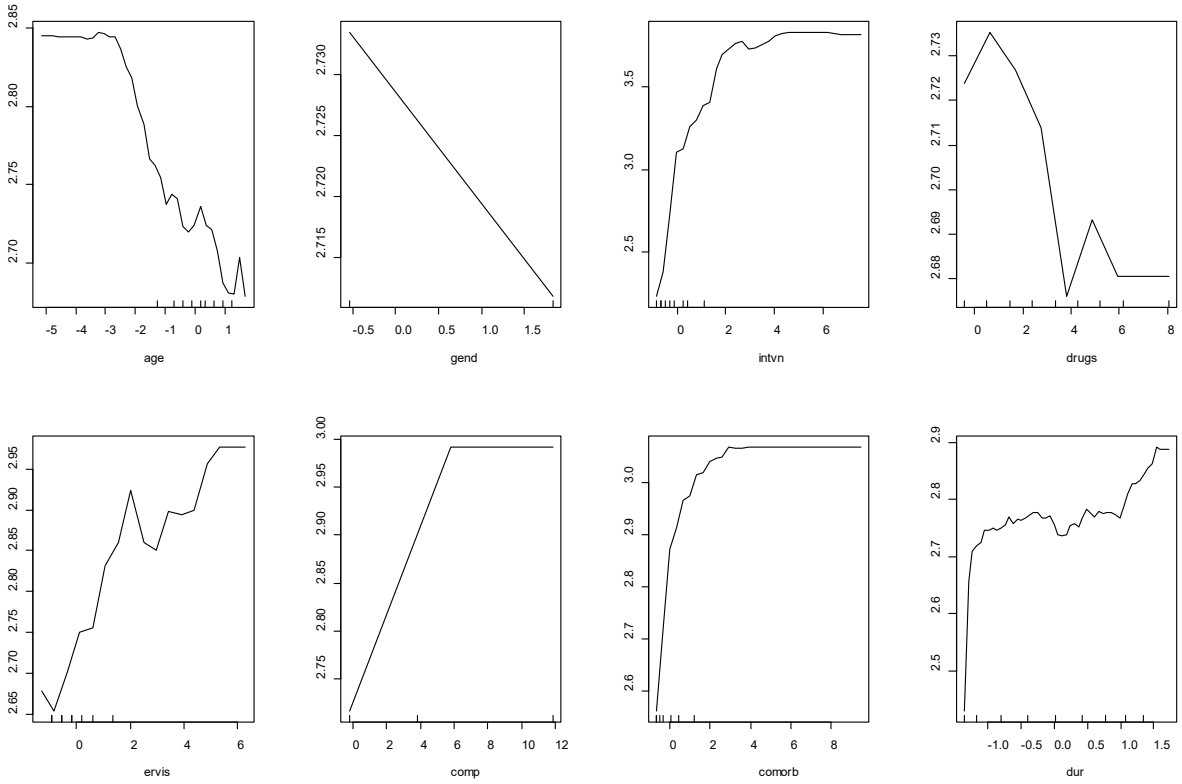
c) Based on the numerical variable importance measure produced by the `importance.randomForest` function, what are the most important variables?

```
importance(rf1)
varImpPlot(rf1)
           %IncMSE IncNodePurity
age     -1.5126491     32.977701
gend    -2.3062364      5.717323
intvn  113.4370392    241.517071
drugs    0.4731302     11.918685
ervis   17.4973644     45.897863
comp    16.2120004      9.472200
comorb  37.2113559     57.477551
dur     23.9306538    101.312750
```

Both variable importance measures says that intvn is by far the most important variable, followed by comorb and dur, and then comp and ervis). The boosted tree also had intvn by far the most important, followed by comorb and dur.

d) Use the `partialPlot.randomForest` function to produce marginal plots (aka partial dependence plots) for each of the eight predictors. Interpret the plots, in terms of whether the predictors have positive, negative, nonlinear, linear, etc. effects on the response. Also discuss whether what you see in the plots agrees with the variable importance measures from part (c).

```
par(mfrow=c(2,4))
for (i in 1:8) partialPlot(rf1, pred.data=IHD1, x.var =
   names(IHD1)[i+1], xlab = names(IHD1)[i+1], main=NULL)
   #creates "partial dependence" plots
par(mfcol=c(1,1))
```



The effects of the three most important predictors (intvn, comorb, adn dur) are very nonlinear. They have monotonically positive effects, increasing sharply as the predictors increase and then plateauing. Overall, the effects of the predictors are very similar to the effects for the boosted tree model.

e) What is the predicted response for $x_1 = 3.0$, $x_2 = 28$, $x_3 = 1.0$?

```
test<-c(59, 0, 10, 0, 3, 0, 4, 300)
xbar<-apply(as.matrix(IHD[,2:9]),2,mean)
std<-apply(as.matrix(IHD[,2:9]),2,sd)
test<-(test-xbar)/std  #must standardize
X<-data.frame(matrix(test,1,8))
```

```
names(X)<-names(IHD1[2:9])
yhat<-predict(rf1,newdata=X);yhat
       1
3.575521
```

The predicted response is 3.57.

f) Repeat part (a) but for `mtry=1` and `mtry=2`. Which of the three random forests (for the three different `mtry` values) is the best model in terms of predictive performance? You can use the OOB r^2 as the measure of predictive performance. Explain what `mtry` controls and why, for this example, lower `mtry` is better.

```
rf1 <- randomForest(log10(cost)~., data=IHD1, mtry=1, ntree =
   500, importance = TRUE)
plot(rf1)  #plots OOB mse vs # trees
rf1 #check the OOB mse and r^2
```

For `mtry=1` and `mtry=2`, the average OOB r^2 is around 0.565 and 0.667, respectively, compared to 0.671 for `mtry=3`. Thus, `mtry=3` is probably the best, followed closely by `mtry=2`. `mtry=1` is much worse.

`mtry` is the number of predictor variables that are selected to consider when making each split of each tree. An essential aspect of random forests is that they select only a subset of predictors on each split.

g) Out of all the models that you fit in Problems 1—4 and 6—7, and also the tree and neural network model that you fit in HW2, which model appears to be the best for predicting cost? Explain.

Use the CV SSE (or equivalently, CV r^2) to rank the models:

model	CV SSE for best in class
tree	198
neural net	172
K-NN	223
GAM	178
local lin. reg	174
PPR	172
Lin. Reg	233
boosted tree	167
random forest	178

For this problem, the boosted tree was the best, followed by the neural network and PPR, followed closely by local linear regression (loess) and then GAM and random forest. The regression tree and K-NN did quite poorly.