HW2 MSiA 420

Yintai Ma

Problem 1

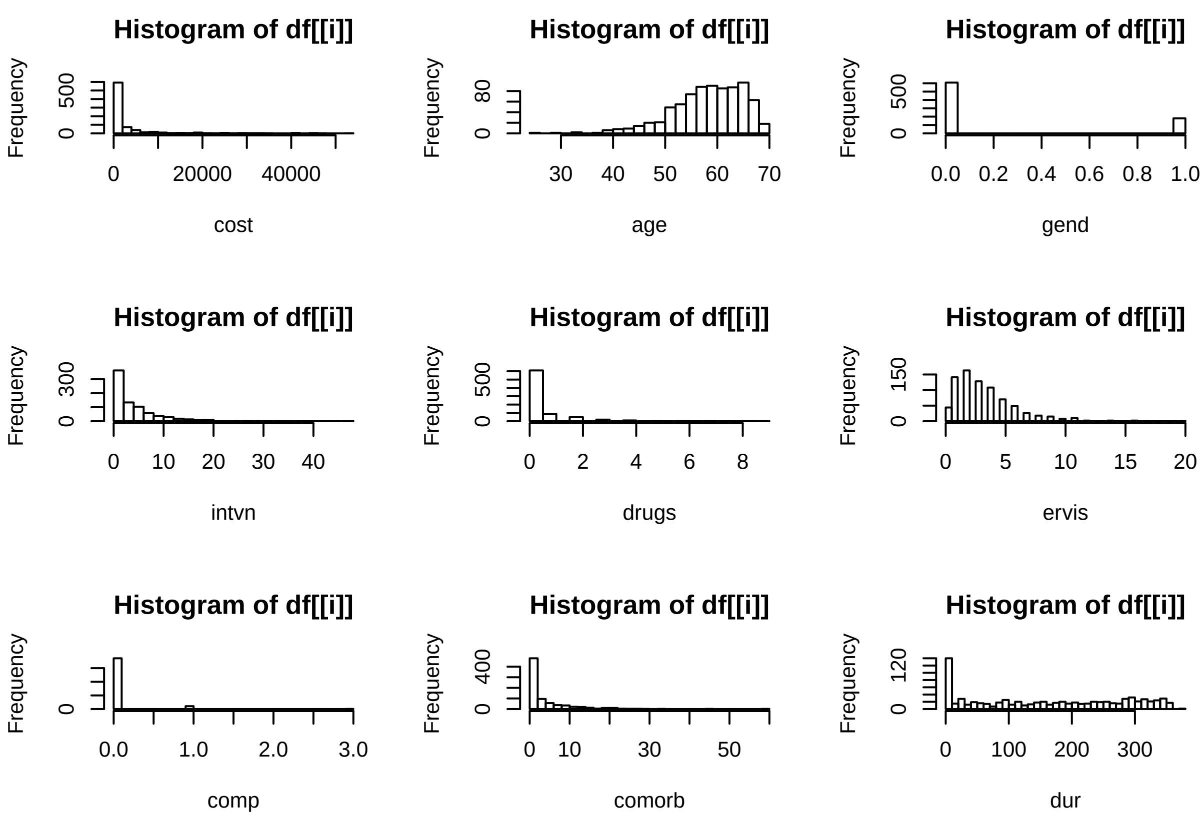
(a)Fit a linear model and discuss the predictive power.

Answer: I take log10 transform to the cost, the response variable, and then fit the model with all predictors unchanged. The R^2 is 0.5831. The model with every predicots standardized has R-square 0.5527. For those predicots that seems to be heavily tailed, I apply log transform on these predictos to see if that will help to imporve the model. Then the R^2 increases to 0.658. Generally, those predictors significant before transform are also significant afterwards.

```r
require(gdata)
df<-read.xls("./HW2_data.xls",sheet=1,header=TRUE)
```

The histogram of each columns

```r
par(mfrow=c(3,3))
for (i in seq(2,10)) hist(df[[i]],breaks=30,xlab=names(df)[i])
```



summary of model 1

```r
mod1<-lm(log10(cost)~.,data = df[-1])
summary(mod1)
```

```
##
```

```
## Call:
## lm(formula = log10(cost) ~ ., data = df[-1])
##
## 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)  2.2228290  0.1698975  13.083  < 2e-16 ***
## age         -0.0044135  0.0028817  -1.532   0.1260
## gend        -0.0669173  0.0460024  -1.455   0.1462
## intvn        0.0878065  0.0038090  23.053  < 2e-16 ***
## drugs       -0.0257198  0.0213709  -1.203   0.2291
## ervis        0.0224358  0.0090588   2.477   0.0135 *
## comp         0.3270883  0.0794497   4.117 4.25e-05 ***
## comorb       0.0228849  0.0037393   6.120 1.48e-09 ***
## dur          0.0012181  0.0001874   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
```

Also, I tried to standardize each variable to see effect.

```
df_std<-df
df_std$cost <- log10(df_std$cost)
df_std[2:10]<-sapply(df_std[2:10], function(x) (x-mean(x))/sd(x))
mod2<-lm(cost~.,data=df_std[-1])
summary(mod2)
```

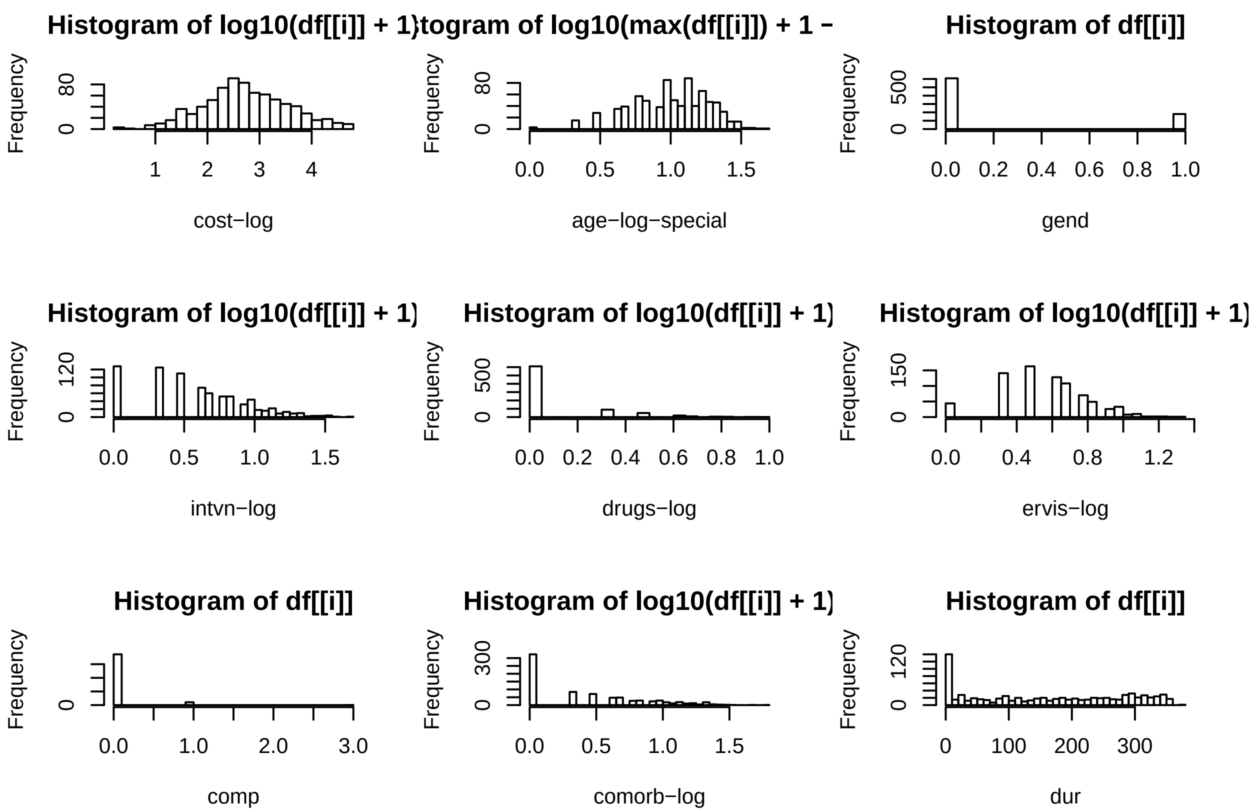
```
##
## Call:
## lm(formula = cost ~ ., data = df_std[-1])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.95741 -0.36347  0.01268  0.34153  2.08450
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.651e-17  2.312e-02   0.000   1.0000
## age         -3.600e-02  2.351e-02  -1.532   0.1260
## gend        -3.395e-02  2.334e-02  -1.455   0.1462
## intvn        5.933e-01  2.574e-02  23.053  < 2e-16 ***
## drugs       -3.305e-02  2.746e-02  -1.203   0.2291
## ervis        7.147e-02  2.886e-02   2.477   0.0135 *
## comp         9.800e-02  2.381e-02   4.117 4.25e-05 ***
## comorb       1.645e-01  2.688e-02   6.120 1.48e-09 ***
## dur          1.779e-01  2.737e-02   6.501 1.43e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.649 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
```

here is the histogram showing the relation between the log response and predictors.

```
par(mfrow=c(3,3))
for (i in seq(2,10)) {
  if (i %in% c(2,5,6,7,9)){
    hist(log10(df[[i]]+1),breaks=30,xlab=paste(names(df)[i],'log',sep='-'))
  }
  if (i==3){
    hist(log10(max(df[[i]])+1-df[[i]]),breaks=30,xlab = paste(names(df)[i],'log','special',sep='-'))
  }
  if (i %in% c(4,8,10)){
    hist(df[[i]],breaks=30,xlab=names(df)[i])
  }

}
```



Here is the model that I log some of the predictos to adjust the heavy tailed.

```
mod3<-lm(log10(cost)~1+log10(max(age)+1-age)+gend+log10(intvn+1)+log10(drugs+1)+log10(ervis+1)+comp+log
summary(mod3)
```

```
##
## Call:
## lm(formula = log10(cost) ~ 1 + log10(max(age) + 1 - age) + gend +
##     log10(intvn + 1) + log10(drugs + 1) + log10(ervis + 1) +
##     comp + log10(comorb + 1) + dur, data = df)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.00074 -0.29367  0.00423  0.26744  1.57651
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.3638884  0.0867482  15.722  < 2e-16 ***
## log10(max(age) + 1 - age)  0.1127965  0.0651800   1.731  0.08393 .
## gend                     -0.0609166  0.0416472  -1.463  0.14396
## log10(intvn + 1)          1.3925417  0.0497154  28.010  < 2e-16 ***
## log10(drugs + 1)         -0.0333703  0.0987980  -0.338  0.73563
## log10(ervis + 1)          0.2454623  0.0790414   3.105  0.00197 **
## comp                      0.3009104  0.0718485   4.188 3.13e-05 ***
## log10(comorb + 1)         0.5040925  0.0496197  10.159  < 2e-16 ***
## dur                       0.0004283  0.0001846   2.320  0.02058 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4866 on 779 degrees of freedom
## Multiple R-squared:  0.658,  Adjusted R-squared:  0.6545
## F-statistic: 187.4 on 8 and 779 DF,  p-value: < 2.2e-16
```

Prob 1(b)

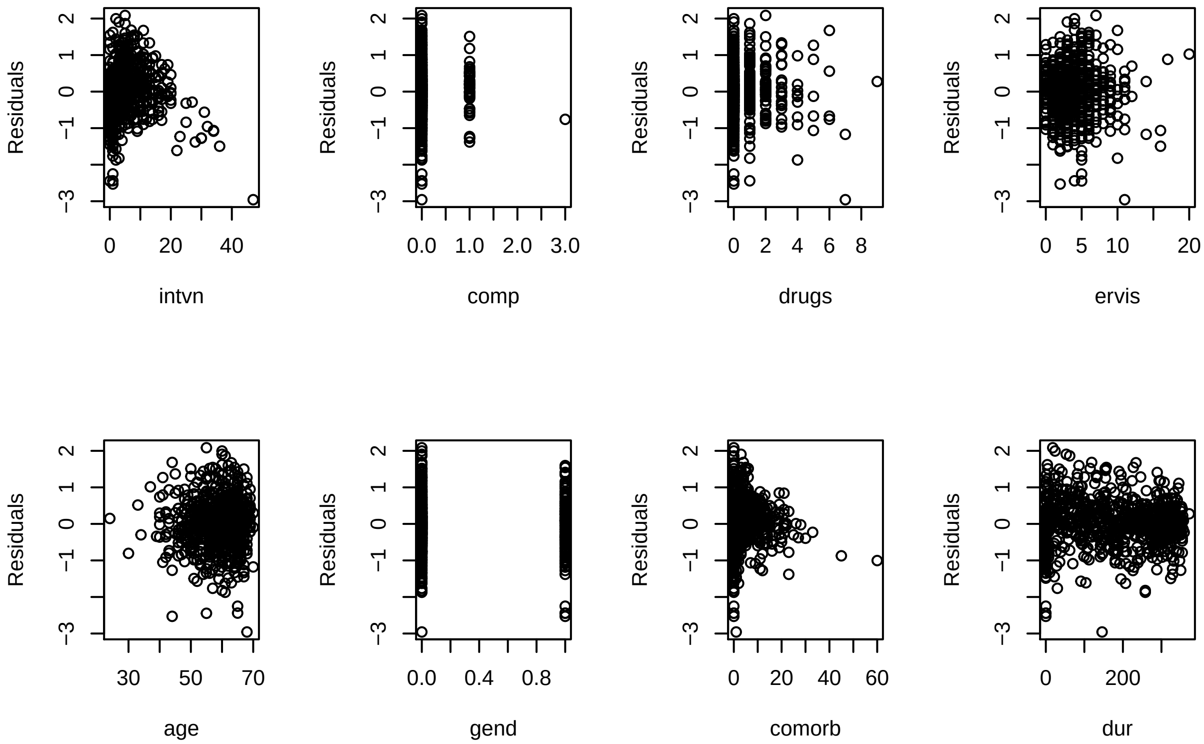
Answer: It seems that the interventions has the largest value of coefficient in each of three models I fitted above. Therfore, the interventions has the most influence on the cost.

Prob 1(c)

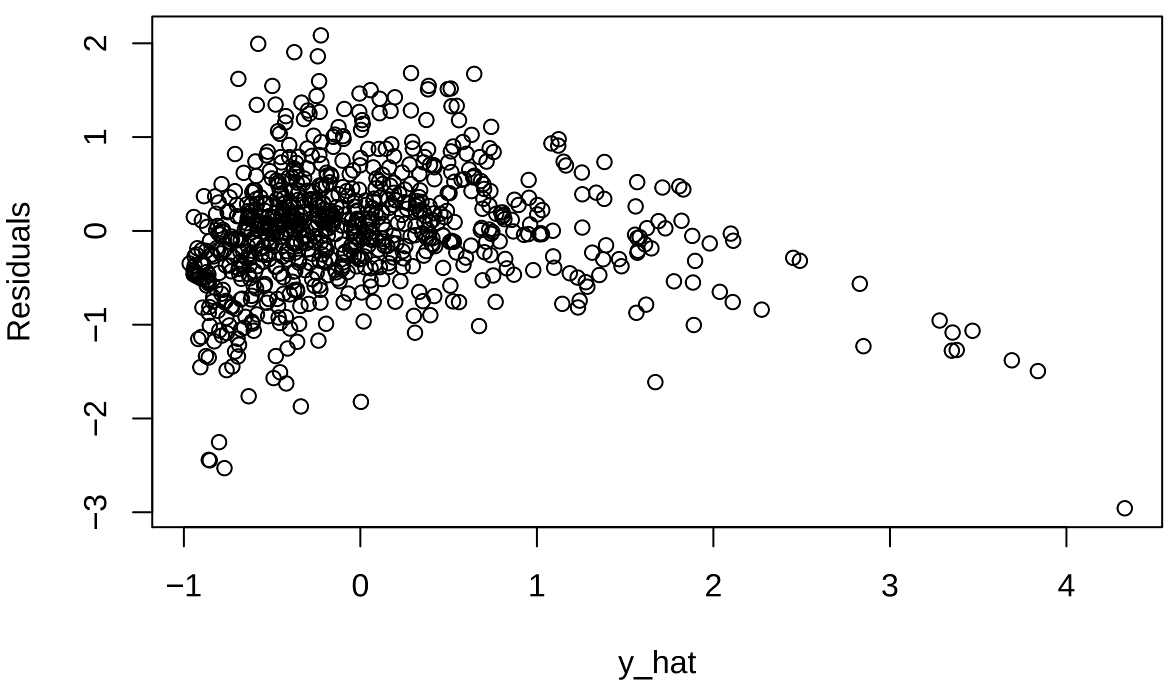
Answer: I use the second model as the observatino object to see if there is any problem. I have drawn residual plot versus y_hat and each variables to see if there is any indication for nonlinearity. For the residual plots vs. predictors, it seems that most of them has no clear signal showing nonlinearity. The intervention and age seems to have some degree of positive correlation with the residual. For the residual plot vs. y_hat, which is the fitted values. It seems that there is a linear relationship between the residual and the fitted value. For the last plot, the residual vs. log(cost), we can observe positive correlation between the residual and the log(cost) when the log(cost) is large.

```r
par(mfrow=c(2,4))
plot(df$intvn,resid(mod2),ylab="Residuals",xlab="intvn")
plot(df$comp,resid(mod2),ylab="Residuals",xlab="comp")
plot(df$drugs,resid(mod2),ylab="Residuals",xlab="drugs")
plot(df$ervis,resid(mod2),ylab="Residuals",xlab="ervis")

plot(df$age,resid(mod2),ylab="Residuals",xlab="age")
plot(df$gend,resid(mod2),ylab="Residuals",xlab="gend")
plot(df$comorb,resid(mod2),ylab="Residuals",xlab="comorb")
plot(df$dur,resid(mod2),ylab="Residuals",xlab="dur")
```

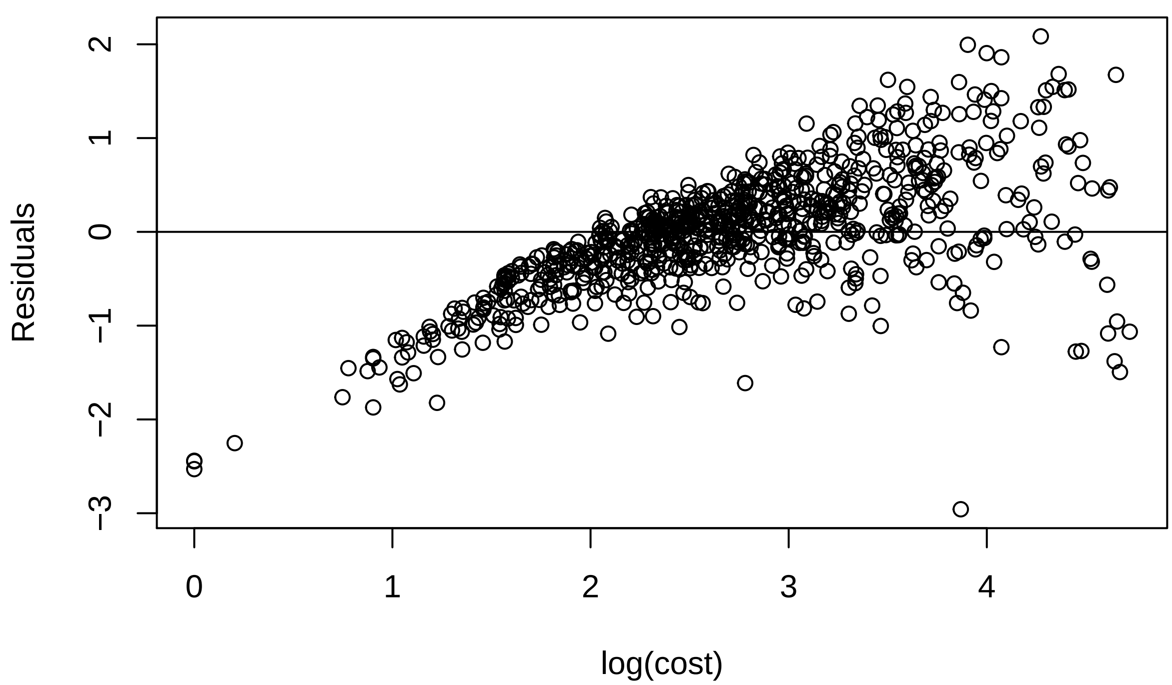



```r
par(mfrow=c(1,1))
plot(mod2$fitted.values,resid(mod2),ylab="Residuals",xlab="y_hat")
```




```r
par(mfrow=c(1,1))
plot(log10(df$cost),resid(mod2),ylab="Residuals",xlab="log(cost)",main="Ischemic heart disease-standardi
abline(0, 0)
```

Ischemic heart disease−standardized predictors with log(cost)−lm



Problem 2

Prob 2(a)

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

Answer: I conducted 10-fold CV to find the best parameters. The best choice is:

CV index random generator

```
CVInd <- function(n,K) {  #n is sample size; K is number of parts; returns K-length list of indices for
  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
}
```

Now use multiple reps of CV to compare Neural Nets and linear reg models###

```
library(nnet)
Nrep<-5 #number of replicates of CV
```

```r
K<-10  #K-fold CV on each replicate
n.lam = 10 #number of lambda
n.num_hidnode = 3 #number of different numbers of hidden nodes
n.models = n.lam*n.num_hidnode #number of different models to fit
n=nrow(df_std)
y<-df_std$cost
yhat=matrix(0,n,n.models)
lam_seq = 10^seq(-as.integer(n.lam/2),as.integer(n.lam/2)-1)
num_hidnode_seq = 5*seq(1,n.num_hidnode)
mod_par=matrix(c(rep(lam_seq,times=1,each=n.num_hidnode),rep(num_hidnode_seq,times=n.lam,each=1)),2,n.m
MSE<-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  print(c(0,0,0,j))#Print out the index of replicates of CV
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    print(k)#Print out the index of different fold of CV
    for (m in 1:n.models) {
      out<-nnet(cost~.,df_std[-Ind[[k]],],linout = T, skip=F,size=as.integer(mod_par[2,m]),decay=mod_pa
      yhat[Ind[[k]],m]<-as.numeric(predict(out,df_std[Ind[[k]],]))
    }
  } #end of k loop
  MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
} #end of j loop
```

```r
MSE
```

```
##           [,1]      [,2]      [,3]      [,4]    [,5]      [,6]      [,7]
## [1,] 0.3375826 0.6489008 0.9584477 0.3684288 1.08066 0.4932864 0.4055308
##           [,8]     [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## [1,] 0.5106135 1.058115 0.3761575 0.5035685 0.7175263 0.3358672 0.3742484
##          [,15]     [,16]     [,17]     [,18]     [,19]     [,20]     [,21]
## [1,] 0.4322105 0.3221292 0.3283505 0.3314952 0.3570771 0.3594142 0.3603294
##          [,22]     [,23]     [,24]    [,25]   [,26]    [,27]     [,28]
## [1,] 0.7013133 0.6716678 0.6614968 1.000336 1.00069 1.000824 0.9990304
##          [,29]     [,30]
## [1,] 0.9991644 0.9992822
```

```r
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error
```

```
##  [1] 0.3375826 0.6489008 0.9584477 0.3684288 1.0806602 0.4932864 0.4055308
##  [8] 0.5106135 1.0581150 0.3761575 0.5035685 0.7175263 0.3358672 0.3742484
## [15] 0.4322105 0.3221292 0.3283505 0.3314952 0.3570771 0.3594142 0.3603294
## [22] 0.7013133 0.6716678 0.6614968 1.0003362 1.0006902 1.0008243 0.9990304
## [29] 0.9991644 0.9992822
```

```r
MSEsd <- apply(MSE,2,sd); MSEsd   #SD of mean square CV error
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [24] NA NA NA NA NA NA NA
```

```r
r2<-1-MSEAve/var(y); r2  #CV r^2
```

```
##  [1]  0.6624173589  0.3510992260  0.0415523447  0.6315712018 -0.0806601879
##  [6]  0.5067135597  0.5944692198  0.4893865211 -0.0581149803  0.6238424793
## [11]  0.4964315184  0.2824736705  0.6641328206  0.6257516220  0.5677894857
## [16]  0.6778707965  0.6716494568  0.6685048361  0.6429229273  0.6405858114
```

```
## [21]  0.6396705965  0.2986866823  0.3283321803  0.3385032038 -0.0003361886
## [26] -0.0006901827 -0.0008243395  0.0009696125  0.0008355903  0.0007178001
```

```r
##The best model in terms of the minimum MSEAve or the maximum r2.
min(MSEAve)
```

```
## [1] 0.3221292
```

```r
max(r2)
```

```
## [1] 0.6778708
```

```r
##Return the index of the minimum MSEAve or the maximum r2.
which(MSEAve==min(MSEAve))
```

```
## [1] 16
```

```r
which(r2==max(r2))
```

```
## [1] 16
```

```r
##The optimal lambda and number of hidden nodes
mod_par[,which(MSEAve==min(MSEAve))]
```

```
## [1] 1 5
```