R Notebook

Code

Hide

```
#Loading the data file
attach(parkinsons_updrs)
```

Hide

```
summary(parkinsons_updrs)
```

```
    subject.          age            sex           test_time        motor_UPDRS
 Min.   : 1.00   Min.   :36.0   Min.   :0.0000   Min.   : -4.263   Min.   : 5.038
 1st Qu.:10.00   1st Qu.:58.0   1st Qu.:0.0000   1st Qu.: 46.847   1st Qu.:15.000
 Median :22.00   Median :65.0   Median :0.0000   Median : 91.523   Median :20.871
 Mean   :21.49   Mean   :64.8   Mean   :0.3178   Mean   : 92.864   Mean   :21.296
 3rd Qu.:33.00   3rd Qu.:72.0   3rd Qu.:1.0000   3rd Qu.:138.445   3rd Qu.:27.596
 Max.   :42.00   Max.   :85.0   Max.   :1.0000   Max.   :215.490   Max.   :39.511
  total_UPDRS      Jitter...          Jitter.Abs.          Jitter.RAP
 Min.   : 7.00   Min.   :0.000830   Min.   :2.250e-06   Min.   :0.000330
 1st Qu.:21.37   1st Qu.:0.003580   1st Qu.:2.244e-05   1st Qu.:0.001580
 Median :27.58   Median :0.004900   Median :3.453e-05   Median :0.002250
 Mean   :29.02   Mean   :0.006154   Mean   :4.403e-05   Mean   :0.002987
 3rd Qu.:36.40   3rd Qu.:0.006800   3rd Qu.:5.333e-05   3rd Qu.:0.003290
 Max.   :54.99   Max.   :0.099990   Max.   :4.456e-04   Max.   :0.057540
  Jitter.PPQ5         Jitter.DDP           Shimmer          Shimmer.dB.
 Min.   :0.000430   Min.   :0.000980   Min.   :0.00306   Min.   :0.026
 1st Qu.:0.001820   1st Qu.:0.004730   1st Qu.:0.01912   1st Qu.:0.175
 Median :0.002490   Median :0.006750   Median :0.02751   Median :0.253
 Mean   :0.003277   Mean   :0.008962   Mean   :0.03404   Mean   :0.311
 3rd Qu.:0.003460   3rd Qu.:0.009870   3rd Qu.:0.03975   3rd Qu.:0.365
 Max.   :0.069560   Max.   :0.172630   Max.   :0.26863   Max.   :2.107
  Shimmer.APQ3      Shimmer.APQ5      Shimmer.APQ11      Shimmer.DDA
 Min.   :0.00161   Min.   :0.00194   Min.   :0.00249   Min.   :0.00484
 1st Qu.:0.00928   1st Qu.:0.01079   1st Qu.:0.01566   1st Qu.:0.02783
 Median :0.01370   Median :0.01594   Median :0.02271   Median :0.04111
 Mean   :0.01716   Mean   :0.02014   Mean   :0.02748   Mean   :0.05147
 3rd Qu.:0.02057   3rd Qu.:0.02375   3rd Qu.:0.03272   3rd Qu.:0.06173
 Max.   :0.16267   Max.   :0.16702   Max.   :0.27546   Max.   :0.48802
      NHR                HNR               RPDE              DFA
 Min.   :0.000286   Min.   : 1.659   Min.   :0.1510   Min.   :0.5140
 1st Qu.:0.010955   1st Qu.:19.406   1st Qu.:0.4698   1st Qu.:0.5962
 Median :0.018448   Median :21.920   Median :0.5423   Median :0.6436
 Mean   :0.032120   Mean   :21.680   Mean   :0.5415   Mean   :0.6532
 3rd Qu.:0.031463   3rd Qu.:24.444   3rd Qu.:0.6140   3rd Qu.:0.7113
 Max.   :0.748260   Max.   :37.875   Max.   :0.9661   Max.   :0.8656
      PPE
 Min.   :0.02198
 1st Qu.:0.15634
 Median :0.20550
 Mean   :0.21959
 3rd Qu.:0.26449
 Max.   :0.73173
```

Hide

```
#Divide the data into training and testing

library(caret)
set.seed(123) # for reproducibility

# Create a vector of row indices
rows <- 1:nrow(parkinsons_updrs)


# Randomly sample 80% of the row indices for the training set
training_rows <- sample(rows, floor(0.8 * length(rows)))

# The remaining rows are for the testing set
testing_rows <- setdiff(rows, training_rows)

# Write the training and testing sets to separate files
write.table(parkinsons_updrs[training_rows, ], file = "Park_training_data.txt", row.names = F
ALSE, col.names = FALSE)
write.table(parkinsons_updrs[testing_rows, ], file = "Park_testing_data.txt", row.names = FAL
SE, col.names = FALSE)

training_data_old <- parkinsons_updrs[training_rows, ]
testing_data_old <- parkinsons_updrs[-training_rows, ]

# Remove the variable 'motor_UPDRS' (Training and Testing)
training_data <- subset(training_data_old, select = -motor_UPDRS)
testing_data <- subset(testing_data_old, select = -motor_UPDRS)


# Create X.train and X.test data frames that exclude the total_UPDRS
X.train <- training_data[, -which(names(training_data) == "total_UPDRS")]
X.test <- testing_data[, -which(names(testing_data) == "total_UPDRS")]
Y.train <- training_data$total_UPDRS
Y.test <- testing_data$total_UPDRS

#b) Division Verification in number of Examples
cat("Number of examples in training data:", nrow(training_data), "\n")
```

```
Number of examples in training data: 4700
```

Hide

```
cat("Number of examples in testing data:", nrow(testing_data), "\n")
```

```
Number of examples in testing data: 1175
```

Hide

```
#Multiple Regression Model Generation
parkinsons_updrs_model=lm(total_UPDRS~., data=training_data)

# Use the model to make predictions on the testing data
predictions <- predict(parkinsons_updrs_model, newdata = testing_data)

#Model Summary and predictions
summary(parkinsons_updrs_model)
```

```
Call:
lm(formula = total_UPDRS ~ ., data = training_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-27.478  -6.760  -1.208   7.082  23.747 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    3.470e+01  3.420e+00  10.147  < 2e-16 ***
subject.       2.649e-01  1.214e-02  21.815  < 2e-16 ***
age            3.145e-01  1.618e-02  19.441  < 2e-16 ***
sex           -5.091e+00  3.514e-01 -14.490  < 2e-16 ***
test_time      1.762e-02  2.560e-03   6.883 6.66e-12 ***
Jitter...     -2.741e+02  2.321e+02  -1.181 0.237728    
Jitter.Abs.   -5.180e+04  1.043e+04  -4.966 7.07e-07 ***
Jitter.RAP    -3.157e+04  5.008e+04  -0.631 0.528387    
Jitter.PPQ5   -1.543e+02  2.110e+02  -0.732 0.464444    
Jitter.DDP     1.093e+04  1.669e+04   0.655 0.512698    
Shimmer       -2.371e+01  6.875e+01  -0.345 0.730199    
Shimmer.dB.    1.994e+00  5.310e+00   0.376 0.707245    
Shimmer.APQ3  -2.594e+04  5.005e+04  -0.518 0.604227    
Shimmer.APQ5   8.380e+01  6.201e+01   1.351 0.176644    
Shimmer.APQ11  8.355e+00  2.946e+01   0.284 0.776700    
Shimmer.DDA    8.592e+03  1.668e+04   0.515 0.606550    
NHR           -2.472e+01  6.726e+00  -3.675 0.000241 ***
HNR           -5.002e-01  7.379e-02  -6.778 1.36e-11 ***
RPDE           2.515e+00  1.956e+00   1.285 0.198696    
DFA           -3.552e+01  2.474e+00 -14.357  < 2e-16 ***
PPE            1.691e+01  3.122e+00   5.417 6.37e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.31 on 4679 degrees of freedom
Multiple R-squared:  0.2539,    Adjusted R-squared:  0.2507 
F-statistic: 79.62 on 20 and 4679 DF,  p-value: < 2.2e-16
```

Hide

```
summary(predictions)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  17.02   25.21   28.88   29.21   33.41   56.33 
```

Hide

```
#1- Forward Subset Selection

library(leaps)

#a) There is no predict() method for regsubsets()
# Define a function for the same purpose
predict.regsubsets=function (object,newdata,id=NULL,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}

## 10-fold CV

k=10
set.seed(1)
# Create a vector that allocates each observation to one of k = 10 folds
folds=sample(1:k,nrow(training_data),replace=TRUE)
# Create a matrix in which we will store the results
cv.errors=matrix(NA,k,20,dimnames=list(NULL,paste(1:20)))

# Loop to perform the 10-fold CV
for (j in 1:k) {
  best.fit=regsubsets(total_UPDRS~.,data=training_data[folds!=j,],nvmax=20,method="forward")
  for(i in 1:20){
    pred=predict.regsubsets(best.fit, training_data[folds==j,],id=i)
    cv.errors[j,i]=mean((training_data$total_UPDRS[folds==j]-pred)^2)
  }
}

# apply() to average over the columns (2) of the matrix in order to obtain a
# vector for which the jth element is the CV error for the j-variable model

mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
```

```
        1         2         3         4         5         6         7         8 
105.05187  97.07620  93.52291  91.19492  89.54373  88.86100  88.70812  88.19571 
        9        10        11        12        13        14        15        16 
 87.68055  87.58059  86.91920  87.02809  87.09240  87.11207  87.12421  87.07298 
       17        18        19        20 
 87.04786  87.03982  87.03327  87.03206 
```

Hide

```
# select the optimal number of variables based on the minimum mean cross-validation error
optimal.vars <- which.min(mean.cv.errors)
cat("Optimal number of variables:", optimal.vars, "\n")
```

```
Optimal number of variables: 11
```

Hide

```
# fit the best subset selection model using the optimal number of variables
reg.best <- regsubsets(total_UPDRS ~ ., data = training_data, nvmax = optimal.vars, method =
"forward")
coef(reg.best,11)
```

```
  (Intercept)      subject.           age           sex     test_time    Jitter.Abs.
 3.641776e+01  2.681204e-01  3.168525e-01 -5.109555e+00  1.696098e-02 -5.243353e+04
   Jitter.DDP       Shimmer           NHR           HNR           DFA            PPE
 1.932401e+02 -4.295563e+01 -2.178188e+01 -5.373866e-01 -3.513367e+01  1.696056e+01
```

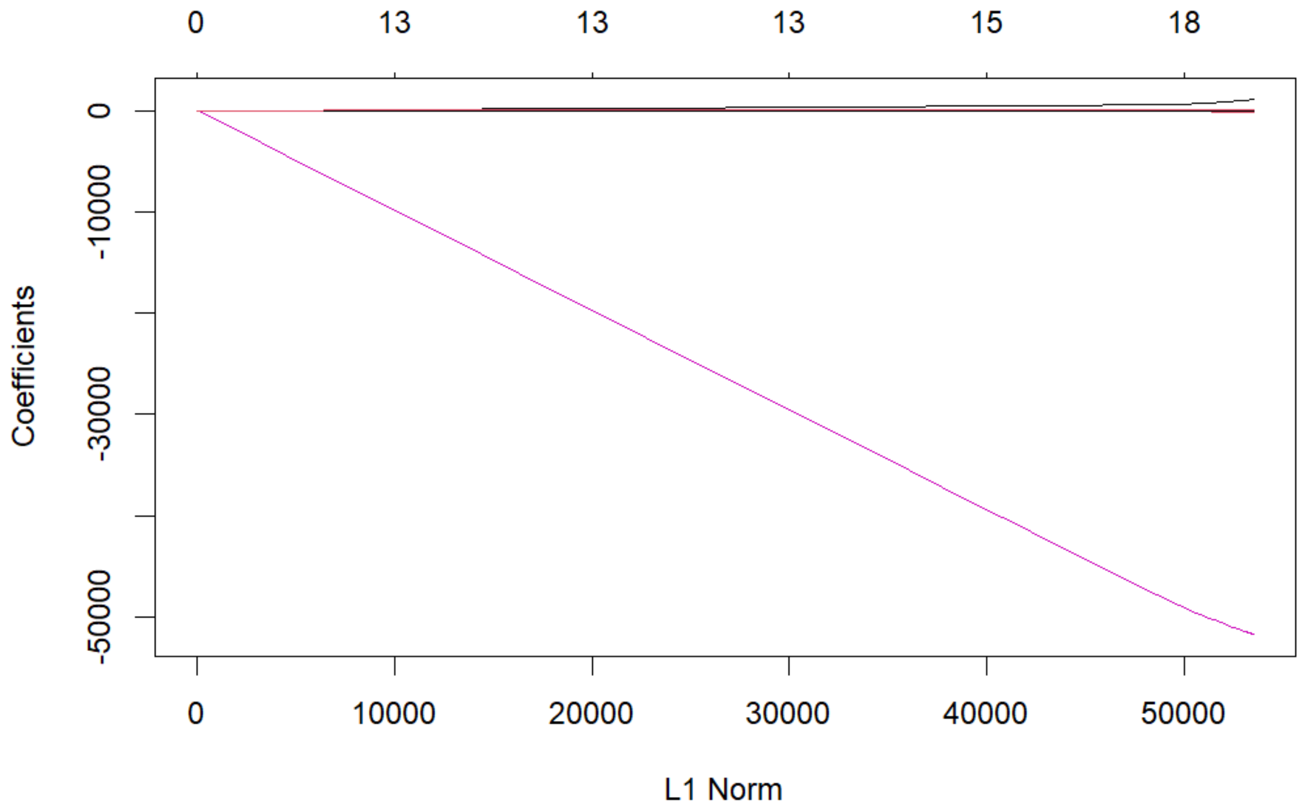
Hide

```
# predict the response for the test set using the best subset selection model
test.pred <- predict.regsubsets(reg.best,newdata=testing_data, id = optimal.vars)

#b) calculate the mean squared error on the test set
test.mse <- mean((testing_data$total_UPDRS - test.pred)^2)
cat("Test set MSE:", test.mse, "\n")
```

```
Test set MSE: 82.63166
```

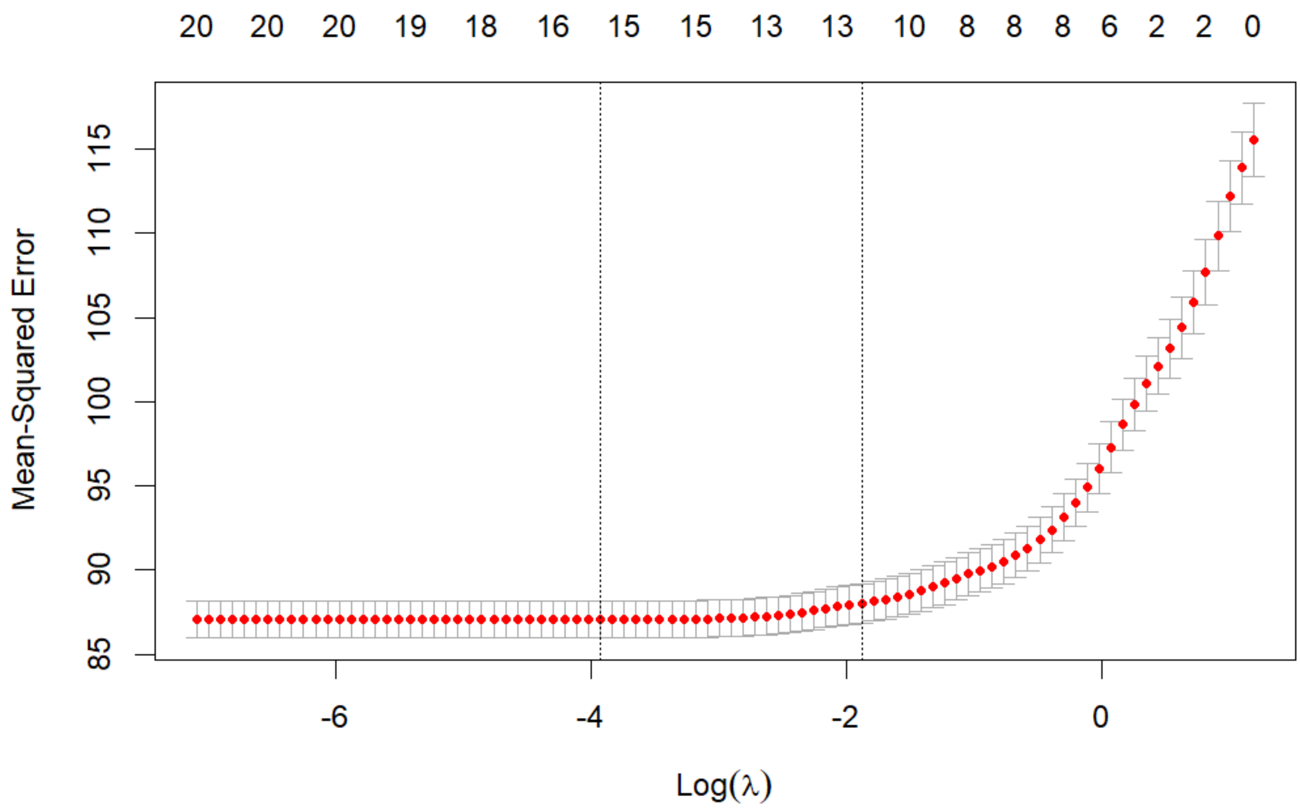
Hide

```
library(glmnet)

parkinsons_updrs_new<-subset(parkinsons_updrs, select = -motor_UPDRS)

## Lasso
x=parkinsons_updrs_new[, -which(names(parkinsons_updrs_new) == "total_UPDRS")]
y=parkinsons_updrs_new$total_UPDRS

#Matrix Generation
X.train_M<-data.matrix(X.train)
Y.train_M<-data.matrix(Y.train)
X.test_M<-data.matrix(X.test)
Y.test_M<-data.matrix(Y.test)
# alpha=1 for Lasso
lasso.mod=glmnet(x=X.train_M,y=Y.train_M,alpha=1)
plot(lasso.mod)
```



Hide

```
#a) Use CV to calculate test error
set.seed(1)
cv.out=cv.glmnet(x=X.train_M,y=Y.train_M,alpha=1)
bestlambda <- cv.out$lambda.min
plot(cv.out)
```



Hide

```
lasso.pred=predict(lasso.mod,s=bestlambda ,newx=X.test_M)
```

Hide

```
# Several coefficients are exactly zero
out=glmnet(x,y,alpha=1)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:20,]
lasso.coef
```

```
  (Intercept)      subject.           age           sex     test_time     Jitter...
 19.859936964   0.233930125   0.309506754  -3.470470280   0.008924172   0.000000000
  Jitter.Abs.    Jitter.RAP   Jitter.PPQ5    Jitter.DDP       Shimmer   Shimmer.dB.
  0.000000000   0.000000000   0.000000000   0.000000000   0.000000000   0.000000000
 Shimmer.APQ3  Shimmer.APQ5 Shimmer.APQ11   Shimmer.DDA           NHR           HNR
  0.000000000   0.000000000   0.000000000   0.000000000  -0.737587459  -0.121605942
         RPDE           DFA
  0.811694614 -22.890899617
```

Hide

```
#b) Mean Squared Error
mse <- mean((Y.test_M - lasso.pred)^2)
mse
```

```
[1] 82.53182
```

Hide

```
#Best lambda
bestlambda
```

```
[1] 0.01962448
```

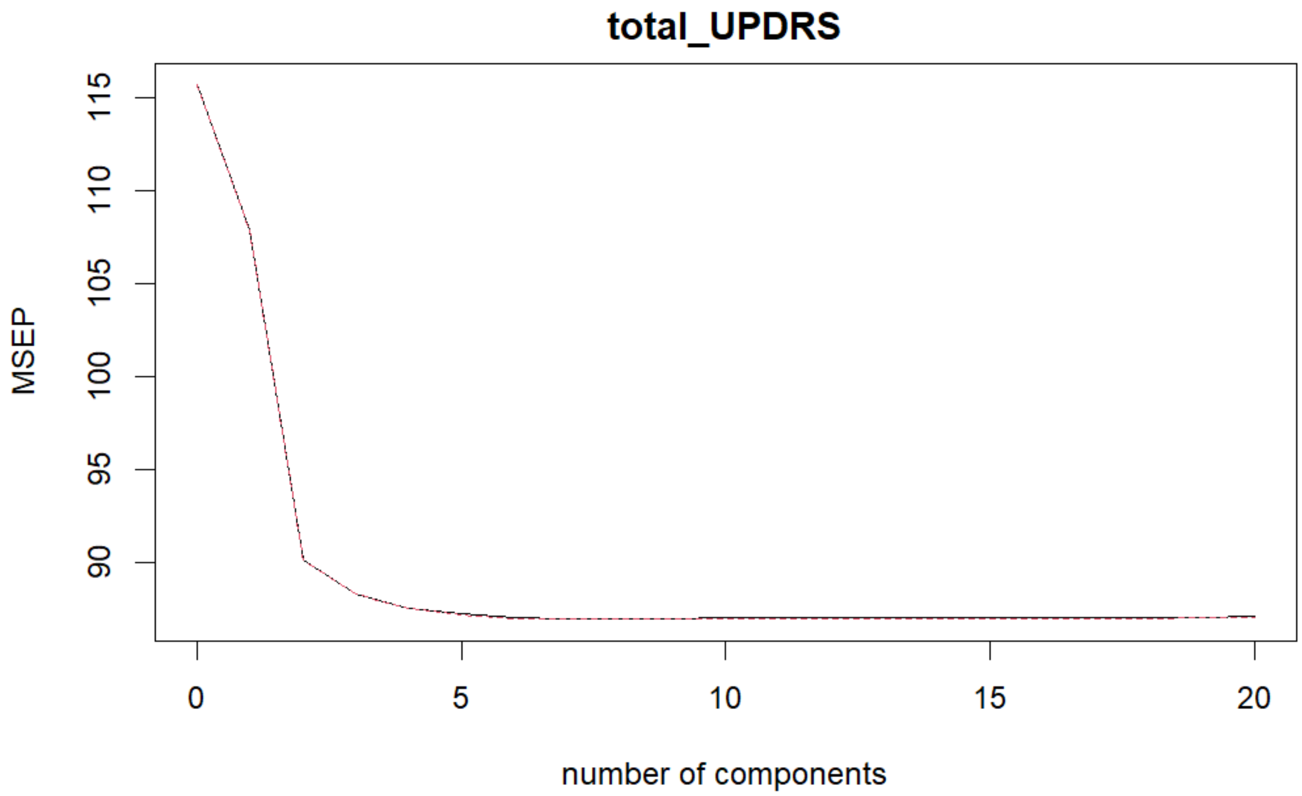
Hide

```
library(pls)

#a) PLS with cross-validation to optimize M
set.seed(123)
pls.cv = plsr(total_UPDRS ~ ., data = training_data, scale = TRUE, validation = "CV")
summary(pls.cv)
```

```
Data:   X dimension: 4700 20
    Y dimension: 4700 1
Fit method: kernelpls
Number of components considered: 20

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
CV           10.76    10.39    9.494    9.398    9.355    9.340    9.328    9.326
adjCV        10.76    10.39    9.493    9.397    9.354    9.338    9.326    9.324
       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
CV       9.326    9.326     9.329     9.329     9.328     9.328     9.328     9.328
adjCV    9.324    9.325     9.327     9.327     9.326     9.326     9.326     9.326
       16 comps  17 comps  18 comps  19 comps  20 comps
CV        9.328     9.328     9.329     9.330     9.332
adjCV     9.326     9.326     9.327     9.328     9.329

TRAINING: % variance explained
             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X             49.964    61.58    68.07    72.51    77.42    81.40    85.51    89.06
total_UPDRS    6.851    22.50    24.04    24.73    25.03    25.22    25.26    25.28
             9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X              92.13     94.07     97.47     98.20     98.63     99.28     99.81
total_UPDRS    25.30     25.33     25.35     25.36     25.38     25.38     25.38
             16 comps  17 comps  18 comps  19 comps  20 comps
X               99.90     99.97    100.00    100.00    100.00
total_UPDRS     25.38     25.38     25.38     25.39     25.39
```

Hide

```
# Plot the validation curve and extract the optimal number of components (M)
validationplot(pls.cv, val.type = "MSEP")
```



Hide

```
optM <- 6
```

Hide

```
#b) Calculate the mean squared error (MSE) using the optimal M=6
pls.pred = predict(pls.cv, testing_data, ncomp = optM)
mse = mean((pls.pred - Y.test)^2)
cat("MSE using optimal M =", mse, "\n")
```

```
MSE using optimal M = 82.59528
```

Hide

```
# Refit the model using all the data and the optimal M
pls.fit = plsr(total_UPDRS ~ ., data = training_data, scale = TRUE, ncomp = optM)
summary(pls.fit)
```

```
Data:   X dimension: 4700 20
    Y dimension: 4700 1
Fit method: kernelpls
Number of components considered: 6
TRAINING: % variance explained
             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X             49.964    61.58    68.07    72.51    77.42    81.40
total_UPDRS    6.851    22.50    24.04    24.73    25.03    25.22
```

Hide

```
#4) a. Are you able to get better predictions by applying feature selection and/or dimension
reduction as compared to previous assignments?

# from the generated models LASSO and BSS-Forward (feature selection) provided better predict
ions when comparing to PLS (dimension reduction), although looking at the MSE, the difference
is almost not that significant. MSE(LASSO)= 82.53 vs MSE(PLS)=82.595, in the feature selectio
n method and the dimension reduction were good at this application producing similar results.

#comparing Forward BSS with PLS, the MSE wasnt to far off for the desired application. MSE(Fo
rward-BSS)= 82.63 vs MSE(PLS)=82.595
```