Lab 11

Welcome to the Lab 11! In the first part, we will apply variable selection techniques to find the best subset of covariates to predict the red wine quality using physicochemical tests scores such as citric acid, pH, etc.

The dataset we will be using is related to red variants of the Portuguese *vinho verde* wine. There are 1599 samples available in the dataset. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

The explanatory variables are all continuous variables and based on physicochemical tests:

- **fixed acidity**
- **volatile acidity**
- **citric acid**
- **residual sugar**
- **chlorides**
- **free sulfur dioxide**
- **total sulfur dioxide**
- **density**
- **pH**
- **sulphates**
- **alcohol**

The response variable is the **quality** score between 0 and 10 (based on sensory data).

We randomly split the data into two parts-the `wine` dataset with 1199 samples and the `wine.test` dataset with 400 samples. Splitting the dataset is a common technique when we want to evaluate the model performance. There are training set, validation set, and test set. The validation set is used for model selection. That is, to estimate the performance of the different model in order to choose the best one. The test set is used for estimating the performance of our final model.

```
set.seed(20170413)
wine.dataset <- read.csv("winequality-red.csv", sep = ";")
test.samples <- sample(1:nrow(wine.dataset), 400)
wine <- wine.dataset[-test.samples, ]
wine.test <- wine.dataset[test.samples, ]
```

We now fit a linear regression using all of the explanatory variables:

```
wine.fit <- lm(quality ~. ,data = na.omit(wine))
summary(wine.fit)
```

```
##
## Call:
## lm(formula = quality ~ ., data = na.omit(wine))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.69755 -0.35429 -0.03872  0.42375  1.99847
##
## Coefficients:
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.093e+01  2.416e+01   0.867   0.3864
## fixed.acidity         3.130e-02  2.980e-02   1.050   0.2938
## volatile.acidity     -1.163e+00  1.373e-01  -8.465  < 2e-16 ***
## citric.acid          -3.944e-01  1.649e-01  -2.392   0.0169 *
## residual.sugar        2.289e-02  1.693e-02   1.351   0.1768
## chlorides            -2.191e+00  4.821e-01  -4.544 6.09e-06 ***
## free.sulfur.dioxide   6.202e-03  2.407e-03   2.576   0.0101 *
## total.sulfur.dioxide -3.471e-03  8.193e-04  -4.237 2.44e-05 ***
## density              -1.699e+01  2.468e+01  -0.688   0.4913
## pH                   -4.129e-01  2.176e-01  -1.898   0.0580 .
## sulphates             1.022e+00  1.311e-01   7.802 1.33e-14 ***
## alcohol               2.860e-01  2.991e-02   9.562  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6314 on 1187 degrees of freedom
## Multiple R-squared:  0.3891, Adjusted R-squared:  0.3834
## F-statistic: 68.72 on 11 and 1187 DF,  p-value: < 2.2e-16
```

We start with our full model `wine.fit`.

We can remove the term corresponding to the coefficient estimate with the highest p-value in the full model and print the summary of the new model

```
# Removing coefficient and saving updated model as `wine.backward`
remove.index <- which.max(summary(wine.fit)$coefficients[,4])
wine.backward <- lm(quality ~ . - density, data = wine)
summary(wine.backward)
```

```
##
## Call:
## lm(formula = quality ~ . - density, data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.68059 -0.35615 -0.03571  0.42659  2.01281
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.3100882  0.6873903   6.270 5.04e-10 ***
## fixed.acidity         0.0149461  0.0179866   0.831  0.40617
## volatile.acidity     -1.1749858  0.1361037  -8.633  < 2e-16 ***
## citric.acid          -0.3944298  0.1648857  -2.392  0.01691 *
## residual.sugar        0.0156751  0.0133024   1.178  0.23889
## chlorides            -2.2216268  0.4799194  -4.629 4.07e-06 ***
## free.sulfur.dioxide   0.0063582  0.0023963   2.653  0.00808 **
## total.sulfur.dioxide -0.0035260  0.0008152  -4.325 1.65e-05 ***
## pH                   -0.5014344  0.1754039  -2.859  0.00433 **
## sulphates             0.9986542  0.1263766   7.902 6.21e-15 ***
## alcohol               0.3015574  0.0196246  15.366  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6313 on 1188 degrees of freedom
## Multiple R-squared:  0.3888, Adjusted R-squared:  0.3837
```

```
## F-statistic: 75.58 on 10 and 1188 DF,  p-value: < 2.2e-16
```

Which covariate was dropped?

Your answer here

In R, there are functions which automatically perform variable selection. The `step()` function uses AIC, which is very similar to RSS but also takes the number of explanatory variables into account. For example, to do backward elimination starting with our full model:

```
step(wine.fit, direction = "backward")
```

```
## Start:  AIC=-1090.65
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS     AIC
## - density               1     0.189 473.43 -1092.2
## - fixed.acidity         1     0.440 473.68 -1091.5
## - residual.sugar        1     0.728 473.97 -1090.8
## <none>                              473.24 -1090.7
## - pH                    1     1.436 474.67 -1089.0
## - citric.acid           1     2.281 475.52 -1086.9
## - free.sulfur.dioxide   1     2.646 475.88 -1086.0
## - total.sulfur.dioxide  1     7.156 480.39 -1074.7
## - chlorides             1     8.231 481.47 -1072.0
## - sulphates             1    24.266 497.50 -1032.7
## - volatile.acidity      1    28.567 501.80 -1022.4
## - alcohol               1    36.456 509.69 -1003.7
##
## Step:  AIC=-1092.17
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS      AIC
## - fixed.acidity         1     0.275 473.70 -1093.47
## - residual.sugar        1     0.553 473.98 -1092.77
## <none>                              473.43 -1092.17
## - citric.acid           1     2.280 475.71 -1088.41
## - free.sulfur.dioxide   1     2.806 476.23 -1087.08
## - pH                    1     3.257 476.68 -1085.95
## - total.sulfur.dioxide  1     7.456 480.88 -1075.43
## - chlorides             1     8.540 481.97 -1072.73
## - sulphates             1    24.885 498.31 -1032.74
## - volatile.acidity      1    29.700 503.13 -1021.21
## - alcohol               1    94.097 567.52  -876.81
##
## Step:  AIC=-1093.47
## quality ~ volatile.acidity + citric.acid + residual.sugar + chlorides +
##     free.sulfur.dioxide + total.sulfur.dioxide + pH + sulphates +
##     alcohol
##
##                        Df Sum of Sq    RSS      AIC
## - residual.sugar        1     0.590 474.29 -1093.98
```

```
## <none>                              473.70 -1093.47
## - citric.acid           1     2.140 475.84 -1090.07
## - free.sulfur.dioxide   1     2.940 476.64 -1088.05
## - pH                    1     5.910 479.61 -1080.60
## - total.sulfur.dioxide  1     8.930 482.63 -1073.08
## - chlorides             1     9.930 483.63 -1070.60
## - sulphates             1    25.248 498.95 -1033.21
## - volatile.acidity      1    30.044 503.75 -1021.74
## - alcohol               1    94.495 568.20  -877.39
##
## Step:  AIC=-1093.98
## quality ~ volatile.acidity + citric.acid + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + pH + sulphates + alcohol
##
##                         Df Sum of Sq    RSS      AIC
## <none>                              474.29 -1093.98
## - citric.acid           1     1.867 476.16 -1091.27
## - free.sulfur.dioxide   1     3.331 477.62 -1087.59
## - pH                    1     5.975 480.27 -1080.97
## - total.sulfur.dioxide  1     8.638 482.93 -1074.34
## - chlorides             1     9.691 483.98 -1071.73
## - sulphates             1    24.867 499.16 -1034.71
## - volatile.acidity      1    29.500 503.79 -1023.63
## - alcohol               1    96.007 570.30  -874.96
##
## Call:
## lm(formula = quality ~ volatile.acidity + citric.acid + chlorides +
##     free.sulfur.dioxide + total.sulfur.dioxide + pH + sulphates +
##     alcohol, data = na.omit(wine))
##
## Coefficients:
##          (Intercept)      volatile.acidity           citric.acid
##             4.697601             -1.131930              -0.294894
##            chlorides   free.sulfur.dioxide  total.sulfur.dioxide
##            -2.288882              0.006858              -0.003640
##                   pH             sulphates               alcohol
##            -0.580238              0.994885               0.300961
```

Exercise 1a

Now try to understand the output of `step()` function. Which variables were omitted from the final model? Provide a list of those variables in order of their elimination, and write the final model.

Variables eliminated (in order):

Final model:

We can use `step()` function with `direction = "forward"` to perform forward selection.

We start from the model with only intercept term. Use the `step()` function to perform forward selection.

```
wine.forward <- step(lm(quality~1, data = na.omit(wine)),
                     scope = formula(lm(quality ~., na.omit(wine)))
                     , direction = "forward")
```

```
## Start:  AIC=-521.79
```

```
## quality ~ 1
##
##                         Df Sum of Sq    RSS     AIC
## + alcohol                1   192.427 582.20 -862.19
## + volatile.acidity       1   119.348 655.28 -720.41
## + sulphates              1    56.804 717.82 -611.11
## + citric.acid            1    30.890 743.74 -568.59
## + density                1    29.454 745.17 -566.27
## + total.sulfur.dioxide   1    28.192 746.44 -564.24
## + chlorides              1    15.129 759.50 -543.44
## + fixed.acidity          1     8.531 766.10 -533.07
## + pH                     1     1.596 773.03 -522.27
## <none>                               774.63 -521.79
## + free.sulfur.dioxide    1     0.595 774.03 -520.72
## + residual.sugar         1     0.026 774.60 -519.83
##
## Step:  AIC=-862.19
## quality ~ alcohol
##
##                         Df Sum of Sq    RSS      AIC
## + volatile.acidity       1    67.673 514.53 -1008.34
## + sulphates              1    35.602 546.60  -935.85
## + pH                     1    16.389 565.81  -894.42
## + citric.acid            1    15.880 566.32  -893.35
## + fixed.acidity          1    14.456 567.75  -890.33
## + total.sulfur.dioxide   1     5.078 577.12  -870.69
## + density                1     2.941 579.26  -866.26
## + chlorides              1     0.983 581.22  -862.21
## <none>                               582.20  -862.19
## + free.sulfur.dioxide    1     0.120 582.08  -860.44
## + residual.sugar         1     0.000 582.20  -860.19
##
## Step:  AIC=-1008.34
## quality ~ alcohol + volatile.acidity
##
##                         Df Sum of Sq    RSS     AIC
## + sulphates              1   16.2128 498.32 -1044.7
## + total.sulfur.dioxide   1    4.0390 510.49 -1015.8
## + pH                     1    3.0633 511.47 -1013.5
## + fixed.acidity          1    2.6034 511.93 -1012.4
## + density                1    1.0256 513.50 -1008.7
## + chlorides              1    0.9571 513.57 -1008.6
## <none>                               514.53 -1008.3
## + citric.acid            1    0.3324 514.20 -1007.1
## + residual.sugar         1    0.0264 514.50 -1006.4
## + free.sulfur.dioxide    1    0.0097 514.52 -1006.4
##
## Step:  AIC=-1044.73
## quality ~ alcohol + volatile.acidity + sulphates
##
##                         Df Sum of Sq    RSS     AIC
## + chlorides              1    9.0441 489.27 -1064.7
## + total.sulfur.dioxide   1    5.5025 492.81 -1056.0
## + citric.acid            1    2.1128 496.20 -1047.8
```

```
## + fixed.acidity         1    1.2818 497.03 -1045.8
## + pH                    1    1.2187 497.10 -1045.7
## <none>                               498.32 -1044.7
## + free.sulfur.dioxide   1    0.0388 498.28 -1042.8
## + density               1    0.0282 498.29 -1042.8
## + residual.sugar        1    0.0076 498.31 -1042.8
##
## Step:  AIC=-1064.69
## quality ~ alcohol + volatile.acidity + sulphates + chlorides
##
##                         Df Sum of Sq    RSS     AIC
## + total.sulfur.dioxide  1    6.0221 483.25 -1077.5
## + pH                    1    2.9442 486.33 -1069.9
## + fixed.acidity         1    1.6005 487.67 -1066.6
## <none>                               489.27 -1064.7
## + citric.acid           1    0.5952 488.68 -1064.2
## + residual.sugar        1    0.1247 489.15 -1063.0
## + free.sulfur.dioxide   1    0.1005 489.17 -1062.9
## + density               1    0.0728 489.20 -1062.9
##
## Step:  AIC=-1077.54
## quality ~ alcohol + volatile.acidity + sulphates + chlorides +
##     total.sulfur.dioxide
##
##                        Df Sum of Sq    RSS     AIC
## + pH                    1   2.98298 480.27 -1083.0
## + free.sulfur.dioxide   1   2.92150 480.33 -1082.8
## + residual.sugar        1   0.82248 482.43 -1077.6
## + fixed.acidity         1   0.80760 482.44 -1077.5
## <none>                              483.25 -1077.5
## + citric.acid           1   0.33782 482.91 -1076.4
## + density               1   0.02353 483.23 -1075.6
##
## Step:  AIC=-1082.97
## quality ~ alcohol + volatile.acidity + sulphates + chlorides +
##     total.sulfur.dioxide + pH
##
##                        Df Sum of Sq    RSS     AIC
## + free.sulfur.dioxide   1    4.1082 476.16 -1091.3
## + citric.acid           1    2.6444 477.62 -1087.6
## <none>                              480.27 -1083.0
## + residual.sugar        1    0.5999 479.67 -1082.5
## + fixed.acidity         1    0.1059 480.16 -1081.2
## + density               1    0.0623 480.20 -1081.1
##
## Step:  AIC=-1091.27
## quality ~ alcohol + volatile.acidity + sulphates + chlorides +
##     total.sulfur.dioxide + pH + free.sulfur.dioxide
##
##                   Df Sum of Sq    RSS     AIC
## + citric.acid      1   1.86729 474.29 -1094.0
## <none>                         476.16 -1091.3
## + residual.sugar   1   0.31731 475.84 -1090.1
## + fixed.acidity    1   0.08918 476.07 -1089.5
```

```
## + density         1   0.04765 476.11 -1089.4
##
## Step:  AIC=-1093.98
## quality ~ alcohol + volatile.acidity + sulphates + chlorides +
##     total.sulfur.dioxide + pH + free.sulfur.dioxide + citric.acid
##
##                   Df Sum of Sq    RSS     AIC
## <none>                          474.29 -1094.0
## + residual.sugar  1   0.59005 473.70 -1093.5
## + fixed.acidity   1   0.31187 473.98 -1092.8
## + density         1   0.23861 474.05 -1092.6
```

Exercise 1b

Write the variables added in order of their addition and the final model.

Variables added (in order):

Final model:

Exercise 2

We will use forward and backward selection on diamonds dataset

```
diamonds <- read.csv("diamonds.csv")[1:100,]
```

2a) Find the best model using forward selection.

```
## Your code here
```

2b) Find the best model using backward selection.

```
## Your code here
```

Regression on all subsets of variables

To find the optimal subset of a certain number of variables for a regression, and to compare between different numbers of variables, use the `regsubsets()` function in the `leaps` package.

```
require(leaps)
```

```
## Loading required package: leaps
```

```
regsub_out <- regsubsets(x = wine[, -12], y = wine[, 12])

summary(regsub_out)
```

```
## Subset selection object
## 11 Variables  (and intercept)
##                       Forced in Forced out
## fixed.acidity             FALSE      FALSE
## volatile.acidity          FALSE      FALSE
## citric.acid               FALSE      FALSE
## residual.sugar            FALSE      FALSE
## chlorides                 FALSE      FALSE
## free.sulfur.dioxide       FALSE      FALSE
## total.sulfur.dioxide      FALSE      FALSE
## density                   FALSE      FALSE
## pH                        FALSE      FALSE
```

```
## sulphates                FALSE      FALSE
## alcohol                  FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1  ( 1 ) " "           " "              " "         " "            " "
## 2  ( 1 ) " "           "*"              " "         " "            " "
## 3  ( 1 ) " "           "*"              " "         " "            " "
## 4  ( 1 ) " "           "*"              " "         " "            "*"
## 5  ( 1 ) " "           "*"              " "         " "            "*"
## 6  ( 1 ) " "           "*"              " "         " "            "*"
## 7  ( 1 ) " "           "*"              " "         " "            "*"
## 8  ( 1 ) " "           "*"              "*"         " "            "*"
##          free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates alcohol
## 1  ( 1 ) " "                 " "                  " "     " " " " "       "*"
## 2  ( 1 ) " "                 " "                  " "     " " " " "       "*"
## 3  ( 1 ) " "                 " "                  " "     " " "*" "       "*"
## 4  ( 1 ) " "                 " "                  " "     " " "*" "       "*"
## 5  ( 1 ) " "                 "*"                  " "     " " "*" "       "*"
## 6  ( 1 ) " "                 "*"                  " "     "*" "*" "       "*"
## 7  ( 1 ) "*"                 "*"                  " "     "*" "*" "       "*"
## 8  ( 1 ) "*"                 "*"                  " "     "*" "*" "       "*"
```

Exercise 3

Use the output of `summary(regsub_out)` to answer the following questions

3a) Write the formula for the optimal model that uses 3 variables

quality ~ .... + ... + ...

3b) Write the formula for the optimal model that uses 7 variables

quality ~ ....

We can use coef to find the optimal model

```
#optimal model that uses 7 variables
coef(regsub_out, 7)
```

```
##          (Intercept)     volatile.acidity            chlorides
##          4.152458457         -0.992176453         -2.456920286
##  free.sulfur.dioxide total.sulfur.dioxide                   pH
##          0.007546816         -0.003900354         -0.428126506
##            sulphates              alcohol
##          0.975879324          0.292841170
```

Is the optimal model with 7 covariates the same as the model found in exercise 1?

Cross Validation

Exercise 4

We will use bodyfat data for this exercise

```
body = read.csv("bodyfat_short.csv")
```

To perform cross validation on the BodyFat dataset we can use the following function. Note: the model in the function includes all the variables in the dataset.

```r
# Define a function to perform k-fold cross-validation on  BODYFAT dataset using a linear regression mo
# k - number of folds

cv_lm <- function(k = 10) {

  data = body
  # Set the seed for reproducibility
  set.seed(123)

  # Determine the number of observations and observations per fold
  n <- nrow(data)
  n_per_fold <- floor(n / k)

  # Shuffle the data
  shuffled_data <- data[sample(n), ]

  # Initialize the vector to store the CV performance
  cv_performance <- numeric(k)

  # Loop over each fold
  for (i in 1:k) {

    # Determine the indices for the current fold
    start_index <- (i - 1) * n_per_fold + 1
    end_index <- min(start_index + n_per_fold - 1, n)
    current_indices <- start_index:end_index

    # Split the data into training and validation sets
    train_data <- shuffled_data[-current_indices, ]
    validation_data <- shuffled_data[current_indices, ]

    # Fit the model on the training set
    lm_model <- lm(BODYFAT ~ ., data = train_data)

    # Predict on the validation set
    y_pred <- predict(lm_model, newdata = validation_data)

    # Calculate the mean squared error
    mse <- mean((validation_data$BODYFAT - y_pred)^2)

    # Store the performance measure for the current fold
    cv_performance[i] <- mse
  }

  # Calculate the average performance measure across folds
  avg_cv_performance <- mean(cv_performance)

  # Return the average performance measure
  return(cv_performance)
}
```

```
cv_lm(10)
```

```
##  [1] 23.52821 20.78288 25.43287 23.56875 27.52718 19.05190 18.24675 15.91035
##  [9] 22.41577 14.45312
```

The function provides mean of mse for k folds for a model that includes all covariates. What if we want to perform cross validation for different models?

We can find the optimal subsets using the `regsubsets()` for bodyfat data and get different models.

```
library(leaps)
bFat = regsubsets(BODYFAT ~ ., body)

summary(bFat)
```

```
## Subset selection object
## Call: regsubsets.formula(BODYFAT ~ ., body)
## 7 Variables  (and intercept)
##         Forced in Forced out
## AGE         FALSE      FALSE
## WEIGHT      FALSE      FALSE
## HEIGHT      FALSE      FALSE
## CHEST       FALSE      FALSE
## ABDOMEN     FALSE      FALSE
## HIP         FALSE      FALSE
## THIGH       FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: exhaustive
##          AGE WEIGHT HEIGHT CHEST ABDOMEN HIP THIGH
## 1  ( 1 ) " " " "    " "    " "   "*"     " " " "
## 2  ( 1 ) " " "*"    " "    " "   "*"     " " " "
## 3  ( 1 ) " " "*"    " "    " "   "*"     " " "*"
## 4  ( 1 ) " " "*"    " "    " "   "*"     "*" "*"
## 5  ( 1 ) " " "*"    "*"    " "   "*"     "*" "*"
## 6  ( 1 ) "*" "*"    "*"    " "   "*"     "*" "*"
## 7  ( 1 ) "*" "*"    "*"    "*"   "*"     "*" "*"
```

CV

```
set.seed(78912)
permutation<-sample(1:nrow(body))

folds <- cut(1:nrow(body),breaks=10,labels=FALSE) ## Line 1

#Perform 10 fold cross validation
predErrorMat<-matrix(nrow=10,ncol=nrow(summary(bFat)$which))
for(i in 1:10){
    #Segement your data by fold using the which() function
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- body[permutation,][testIndexes, ] ## Line 2
    trainData <- body[permutation,][-testIndexes, ]
    #Use the test and train data partitions however you desire...


    predError<-apply(summary(bFat)$which[,-1],1,function(x){  ## Line 3
```

```
        lmObj<-lm(trainData$BODYFAT~.,data=trainData[,-1][,x,drop=FALSE])  ## Line 4
        testPred<-predict(lmObj,newdata=testData[,-1])
        mean((testData$BODYFAT-testPred)^2)  ## Line 5
    })
    predErrorMat[i,]<-predError ##Line 6
}
predErrorMat
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
##  [1,] 18.72568 10.95537 11.68551 12.16354 11.83839 11.78985 11.93013
##  [2,] 21.41687 21.08760 21.53709 21.06757 21.10223 21.20400 21.62519
##  [3,] 32.47863 21.97477 22.48690 22.50871 22.97452 22.92450 24.05130
##  [4,] 21.05072 20.22509 19.16631 18.82538 18.90923 18.89133 18.94164
##  [5,] 26.47937 22.92690 23.76934 26.13180 26.17794 26.12684 26.28473
##  [6,] 26.60945 23.35274 22.06232 22.06825 22.15430 23.10201 25.29325
##  [7,] 25.65426 20.48995 19.95947 19.82442 19.53618 19.97744 20.29104
##  [8,] 17.54916 18.79081 18.14251 17.67780 17.74409 17.67456 17.71624
##  [9,] 33.52443 27.26399 25.83256 26.87850 27.80847 28.32894 28.41455
## [10,] 18.64271 14.11973 14.05815 14.53730 14.42609 14.36767 14.57028
```

Describe in a few sentences what Lines 1-6 do

Line 1:

Line 2:

Line 3:

Line 4:

Line 5:

Line 6: