FIT5209 Assignment 1 - Linear Regression

Bethany Hooper

20/07/2020

Student Number: 31025102

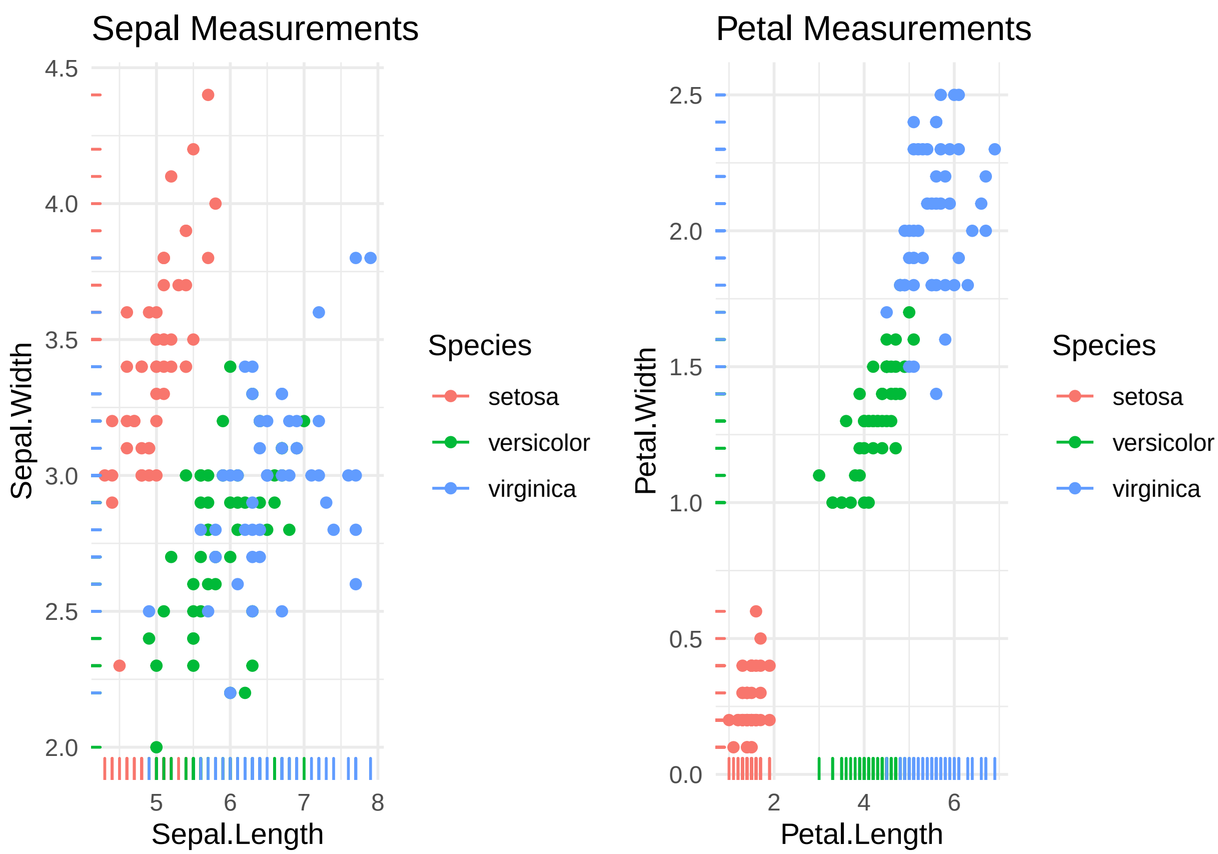
In this assessment, you need to answer all the questions about KNN, Linear Regression, Regularization, Logistic Regression, K-fold cross-validation, and other concepts covered in Module 1-3. R studio is recommended to use to complete your assessment. All codes need comments to help markers to understand your idea. If no comment is given, you may have a 10% redundancy on your mark. Please refer to weekly activities as examples for how to write comments. After you have answered all the questions, please knit your R notebook file to HTML or PDF format. Submit both .rmd file and .html or .pdf file to assessment 1 dropbox via the link on the Assessment page. You can compress your files into a zip file for submission. The total mark of this assessment is 100, which worths 30% of your final result.

hint: Please review all reading materials in Module 1-3 carefully, especially the activities.

Question 1 - KNN (20 marks)

In this question, it is expected that the iris dataset will be split into training and test data sets using a ratio of 7:3 and then run through a KNN classifier in order to predict the class of iris plants. Firstly, it is useful to visualise the data in order to get a general understanding of the iris dataset. Below is a scatterplot comparing the variables *Sepal.Width* and *Sepal.Length* using *Species* to catagorise them. It can be noted that there appears to be relatively high correlation between sepal length and sepal width in the Iris-Setosa, while less of a correlation can be observed in the Iris-Virginica and the Iris-Versicolor flowers. This is evident in the spread of data points for these two flowers, they are more spread out and do not cluster like seen in the Iris-Setosa flowers. A similar pattern is shown when comparing *Petal.Width* and *Petal.Length*. After visualising the data it can then be split into the training and test sets in preperation for the KNN classifier.

```r
#load in iris data from datasets package
library(datasets)
data(iris)

#create scatterplot to illistrate petal measurement and visualise the data
sepal_plot <- ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
    geom_point() + geom_rug()+ theme_minimal() + ggtitle("Sepal Measurements")
petal_plot <- ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
    geom_point() + geom_rug()+ theme_minimal() + ggtitle("Petal Measurements")
grid.arrange(sepal_plot, petal_plot, ncol = 2)
```



a) Split the data set into a training and a test set with the ratio of 7:3. (1 mark)

```
set.seed(223)
iris <- iris[sample(1:nrow(iris),nrow(iris)),]
# create  training and testing subsets:
train.index = 1:105
train.data <- iris[train.index, -5] # grab the first 105 records, leave out the species (last column)
train.label <- iris[train.index, 5]
test.data <- iris[-train.index, -5] # grab the last 45 records, leave out the species (last column)
test.label <- iris[-train.index, 5]
```

b) Implement a KNN classifier. (5 marks)

```
# define a function that calculates the majority votes (or mode!)
majority <- function(x) {
   uniqx <- unique(x)
   uniqx[which.max(tabulate(match(x, uniqx)))]
}

#create Knn function using Euclidean distance
knn1 <- function(train.data, train.label, test.data, K=k, distance = 'euclidean'){
    train.len <- nrow(train.data)
    test.len <- nrow(test.data)
    dist <- as.matrix(dist(rbind(test.data, train.data), method= distance))[1:test.len, (test.len+1):(t
    for (i in 1:test.len){
        nn <- as.data.frame(sort(dist[i,], index.return = TRUE))[1:K,2]
```

```
        test.label[i]<- (majority(train.label[nn]))
    }
    return (test.label)
}
```

```
#create Knn function using Manhattan distance
knn2 <- function(train.data, train.label, test.data, K=k, distance = 'manhattan'){
    train.len <- nrow(train.data)
    test.len <- nrow(test.data)
    dist <- as.matrix(dist(rbind(test.data, train.data), method= distance))[1:test.len, (test.len+1):(te
    for (i in 1:test.len){
        nn <- as.data.frame(sort(dist[i,], index.return = TRUE))[1:K,2]
        test.label[i]<- (majority(train.label[nn]))
    }
    return (test.label)
}
```

```
#create Knn function using Canberra distance
knn3 <- function(train.data, train.label, test.data, K=k, distance = 'canberra'){
    train.len <- nrow(train.data)
    test.len <- nrow(test.data)
    dist <- as.matrix(dist(rbind(test.data, train.data), method= distance))[1:test.len, (test.len+1):(te
    for (i in 1:test.len){
        nn <- as.data.frame(sort(dist[i,], index.return = TRUE))[1:K,2]
        test.label[i]<- (majority(train.label[nn]))
    }
    return (test.label)
}
```

```
#create Knn function using Minkowski distance
knn4 <- function(train.data, train.label, test.data, K=k, distance = 'minkowski'){
    train.len <- nrow(train.data)
    test.len <- nrow(test.data)
    dist <- as.matrix(dist(rbind(test.data, train.data), method= distance))[1:test.len, (test.len+1):(te
    for (i in 1:test.len){
        nn <- as.data.frame(sort(dist[i,], index.return = TRUE))[1:K,2]
        test.label[i]<- (majority(train.label[nn]))
    }
    return (test.label)
}
```

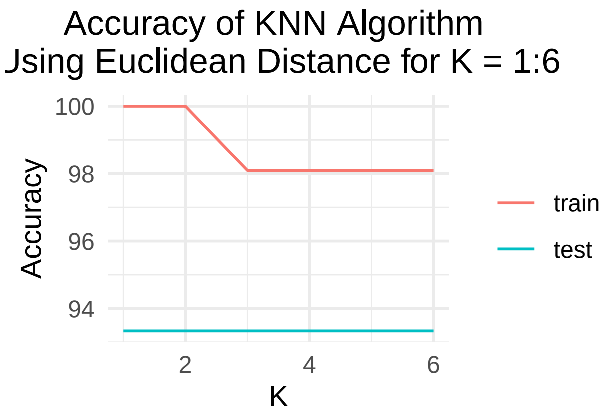
c) Investigate the impact of different K (from 1 to 6) values on the model performance (ACC) and the impact of different distance measurements (euclidean, manhattan, canberra, and minkowski) on the model performance (ACC). Visualize and discuss your findings. (14 marks)

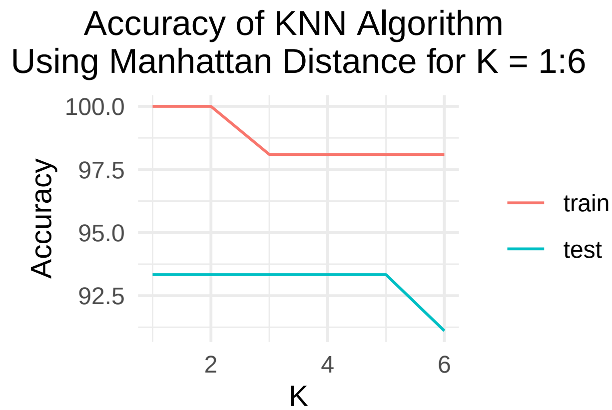
```
# calculate the train and test missclassification rates for K in 1:6 for each distance
acc.Eu <- data.frame('K'=1:6, 'train'=rep(0,6), 'test'=rep(0,6))
for (k in 1:6){
    acc.Eu[k,'train'] <- sum(knn1(train.data, train.label, train.data, K=k) == train.label)/nrow(train.d
    acc.Eu[k,'test'] <-  sum(knn1(train.data, train.label, test.data, K=k)  == test.label)/nrow(test.dat
}
```

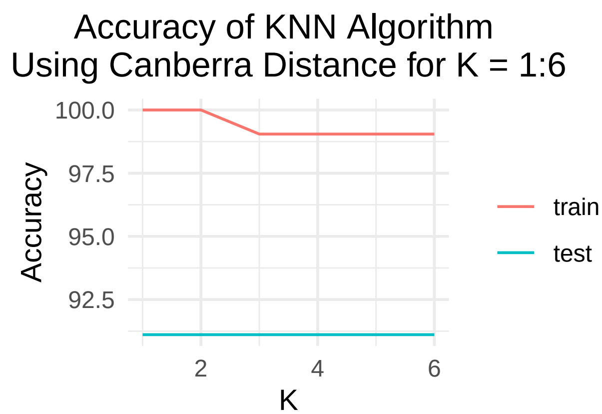
```
acc.Man <- data.frame('K'=1:6, 'train'=rep(0,6), 'test'=rep(0,6))
for (k in 1:6){
    acc.Man[k,'train'] <- sum(knn2(train.data, train.label, train.data, K=k) == train.label)/nrow(train
    acc.Man[k,'test'] <-  sum(knn2(train.data, train.label, test.data, K=k)  == test.label)/nrow(test.da
}

acc.Can <- data.frame('K'=1:6, 'train'=rep(0,6), 'test'=rep(0,6))
for (k in 1:6){
    acc.Can[k,'train'] <- sum(knn3(train.data, train.label, train.data, K=k) == train.label)/nrow(train
    acc.Can[k,'test'] <-  sum(knn3(train.data, train.label, test.data, K=k)  == test.label)/nrow(test.da
}

acc.Min <- data.frame('K'=1:6, 'train'=rep(0,6), 'test'=rep(0,6))
for (k in 1:6){
    acc.Min[k,'train'] <- sum(knn4(train.data, train.label, train.data, K=k) == train.label)/nrow(train
    acc.Min[k,'test'] <-  sum(knn4(train.data, train.label, test.data, K=k)  == test.label)/nrow(test.da
}
```

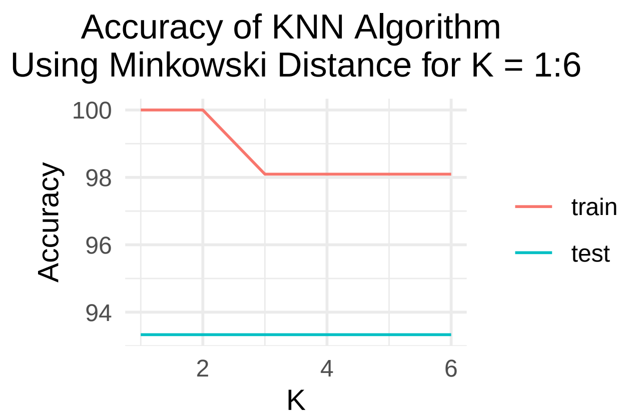
```
# melt each dataframe for plotting
acc.m1 <- melt(acc.Eu, id='K') # reshape for visualization
names(acc.m1) <- c('K', 'Type', 'Accuracy')

acc.m2 <- melt(acc.Man, id='K') # reshape for visualization
names(acc.m2) <- c('K', 'Type', 'Accuracy')

acc.m3 <- melt(acc.Can, id='K') # reshape for visualization
names(acc.m3) <- c('K', 'Type', 'Accuracy')

acc.m4 <- melt(acc.Min, id='K') # reshape for visualization
names(acc.m4) <- c('K', 'Type', 'Accuracy')
```

```
#plot each dataframe for training and test sets
eu.plot <- ggplot(data=acc.m1, aes(x= K, y=Accuracy, color=Type)) + geom_line() +
       scale_color_discrete(guide = guide_legend(title = NULL)) + theme_minimal() +
       ggtitle("Accuracy of KNN Algorithm \nUsing Euclidean Distance for K = 1:6") + theme(plot.title =


man.plot <- ggplot(data=acc.m2, aes(x=K, y=Accuracy, color=Type)) + geom_line() +
       scale_color_discrete(guide = guide_legend(title = NULL)) + theme_minimal() +
       ggtitle("Accuracy of KNN Algorithm \nUsing Manhattan Distance for K = 1:6") + theme(plot.title =


can.plot <- ggplot(data=acc.m3, aes(x=K, y=Accuracy, color=Type)) + geom_line() +
       scale_color_discrete(guide = guide_legend(title = NULL)) + theme_minimal() +
       ggtitle("Accuracy of KNN Algorithm \nUsing Canberra Distance for K = 1:6") + theme(plot.title = e


min.plot <- ggplot(data=acc.m4, aes(x=K, y=Accuracy, color=Type)) + geom_line() +
       scale_color_discrete(guide = guide_legend(title = NULL)) + theme_minimal() +
       ggtitle("Accuracy of KNN Algorithm \nUsing Minkowski Distance for K = 1:6") + theme(plot.title =
grid.arrange(eu.plot, man.plot, can.plot, min.plot, nrow = 2, ncol = 2)
```









Given the results of the KNN classifier it appears that in terms of training data across the four different distance measures; k = 1 and k = 2 have the highest accuracy rate with an accuracy of 100% demonstrated in the graphs provided above. At k =3 Euclidean, Manhattan and Minkowski distance are the same at 98.095% however, the accuracy of the Canberra distance only drops to 99.048%. When using Euclidean, Manhattan and Minkowski distance as measurements on the training data the accuracy results appear to be the same. This could be due to the fact that Minkowski distance is a generalisation of Euclidean and Manhattan distance.

The line indicating the test data's accuracy only changes when using Manhattan distance. Euclidean and Minkowski graphs are exactly the same (93.33% across all k values), this could be a result of the inbuilt *dist()* function that measures the distances has Minkowski distance default to p =2 (Euclidean distance). The Canberra distance graph has the accuracy at a flat 91.11% at any given k value. The Manhattan distance graph shows a drop in accuracy from k = 5 (93.33%) to k = 6 (91.11%). Further investigation could be made in the model performance by bootstrapping or performing cross-validation on the dataset.

Question 2 - Linear Regression (35 marks)

In this question you need to implement a linear regression model to predict health care cost. The data set used in this question can be found in 'insurance.csv'. The data set has 7 features, which are summarized as below.

- Age: insurance contractor age, years
- Sex: insurance contractor gender, [female, male]
- BMI: Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ˆ 2) using the ratio of height to weight, ideally 18.5 to 24.9
- Children: number of children covered by health insurance / Number of dependents
- Smoker: smoking, [yes, no]

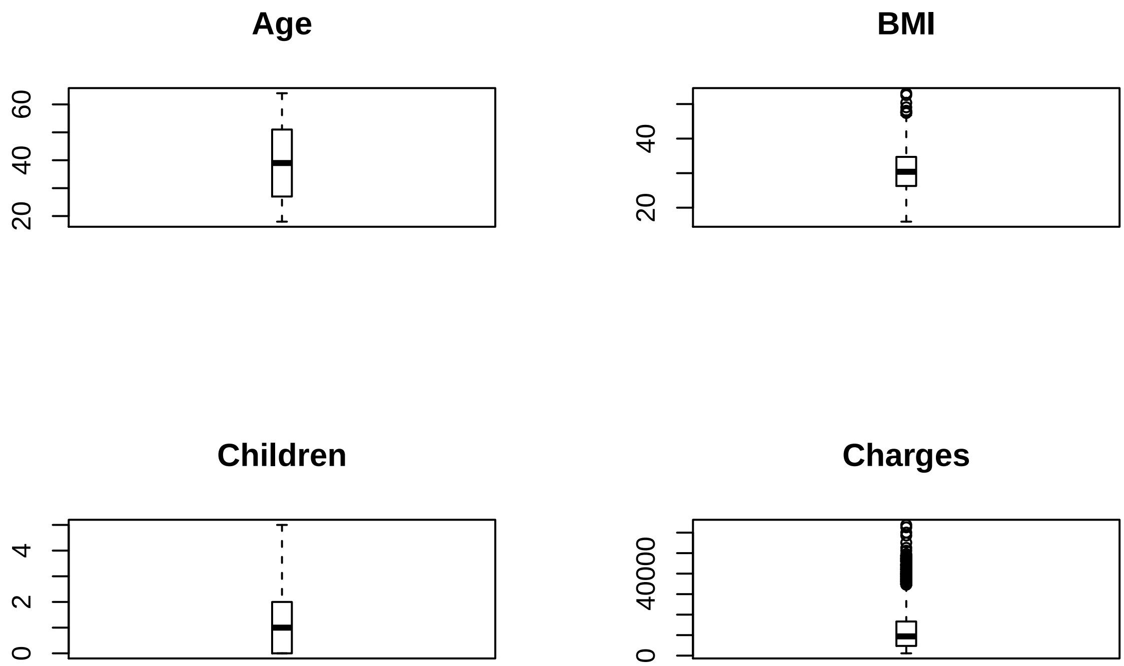
- Region: the beneficiary's residential area in the US, [northeast, southeast, southwest, northwest]
- Charges: Individual medical costs billed by health insurance, $ #predicted value

```
library(dplyr)
data = read.csv('insurance.csv')
```

a) Perform data pre-processing, including removing invalid data and transfromatting the categorical features to numerical features. (4 marks)

Before the data can be used in any sort of linear regression, it first must be checked and cleaned of any missing, invalid or outlier data. Initially all missing values were removed from the dataset. Then each attribute was checked via a box plot to check to see if there were any outliers present in the dataset, 9 outliers were found in the *bmi* attribute and were removed from the data set as they were outside the known range for bmis (12-42). 136 outliers were found in the charges attribute and were also removed from the dataset. The data was also normalised for the convenience of the analysis.

```
##check for missing values
data <- na.omit(data)

par(mfrow = c(2, 2))
##check for outliers
outliers.age <- boxplot.stats(data$age)$out
boxplot(data$age, main = "Age", boxwex = 0.1)


outliers.bmi <- boxplot(data$bmi, plot = FALSE)$out #### appears to be outliers that are unrealistic bm
boxplot(data$bmi, main = "BMI", boxwex = 0.1)


outliers.children <- boxplot.stats(data$children)$out
boxplot(data$children, main = "Children", boxwex = 0.1)


outliers.charges <- boxplot.stats(data$charges)$out
boxplot(data$charges, main = "Charges", boxwex = 0.1)
```




```r
#find which rows contain bmi outliers and remove rows
data[which(data$bmi %in% outliers.bmi), ]
```

```
##      age    sex   bmi children smoker    region   charges
## 117   58   male 49.06        0     no southeast 11381.325
## 287   46 female 48.07        2     no northeast  9432.925
## 402   47   male 47.52        1     no southeast  8083.920
## 544   54 female 47.41        0    yes southeast 63770.428
## 848   23   male 50.38        1     no southeast  2438.055
## 861   37 female 47.60        2    yes southwest 46113.511
## 1048  22   male 52.58        1    yes southeast 44501.398
## 1089  52   male 47.74        1     no southeast  9748.911
## 1318  18   male 53.13        0     no southeast  1163.463
```

```r
data <- data[-which(data$bmi %in% outliers.bmi), ]

data[which(data$charges %in% outliers.charges), ]
```

```
##      age    sex    bmi children smoker    region  charges
## 15    27   male 42.130        0    yes southeast 39611.76
## 20    30   male 35.300        0    yes southwest 36837.47
## 24    34 female 31.920        1    yes northeast 37701.88
## 30    31   male 36.300        2    yes southwest 38711.00
## 31    22   male 35.600        0    yes southwest 35585.58
## 35    28   male 36.400        1    yes southwest 51194.56
```

```
## 39    35   male 36.670        1    yes northeast 39774.28
## 40    60   male 39.900        0    yes southwest 48173.36
## 50    36   male 35.200        1    yes southeast 38709.18
## 54    36   male 34.430        0    yes southeast 37742.58
## 56    58   male 36.955        2    yes northwest 47496.49
## 83    22   male 37.620        1    yes southeast 37165.16
## 85    37 female 34.800        2    yes southwest 39836.52
## 87    57 female 31.160        0    yes northwest 43578.94
## 95    64 female 31.300        2    yes southwest 47291.06
## 110   63   male 35.090        0    yes southeast 47055.53
## 124   44   male 31.350        1    yes northeast 39556.49
## 147   46   male 30.495        3    yes northwest 40720.55
## 159   30   male 35.530        0    yes southeast 36950.26
## 162   18 female 36.850        0    yes southeast 36149.48
## 176   63 female 37.700        0    yes southwest 48824.45
## 186   36   male 41.895        3    yes northeast 43753.34
## 204   27 female 36.080        0    yes southeast 37133.90
## 224   19   male 34.800        0    yes southwest 34779.61
## 241   23 female 36.670        2    yes northeast 38511.63
## 243   55 female 26.800        1     no southwest 35160.13
## 252   63 female 32.200        2    yes southwest 47305.31
## 253   54   male 34.210        2    yes southeast 44260.75
## 255   50   male 31.825        0    yes northeast 41097.16
## 257   56   male 33.630        0    yes northwest 43921.18
## 264   19   male 36.955        0    yes northwest 36219.41
## 266   46   male 42.350        3    yes southeast 46151.12
## 272   50   male 34.200        2    yes southwest 42856.84
## 282   54   male 40.565        3    yes northeast 48549.18
## 289   59 female 36.765        1    yes northeast 47896.79
## 293   25   male 45.540        2    yes southeast 42112.24
## 299   31   male 34.390        3    yes northwest 38746.36
## 313   43   male 35.970        3    yes southeast 42124.52
## 315   27 female 31.400        0    yes southwest 34838.87
## 323   34   male 30.800        0    yes southwest 35491.64
## 328   45   male 36.480        2    yes northwest 42760.50
## 329   64 female 33.800        1    yes southwest 47928.03
## 331   61 female 36.385        1    yes northeast 48517.56
## 339   50   male 32.300        1    yes northeast 41919.10
## 374   26   male 32.900        2    yes southwest 36085.22
## 378   24   male 40.150        0    yes southeast 38126.25
## 382   55   male 30.685        0    yes northeast 42303.69
## 421   64   male 33.880        0    yes southeast 46889.26
## 422   61   male 35.860        0    yes southeast 46599.11
## 423   40   male 32.775        1    yes northeast 39125.33
## 442   33 female 33.500        0    yes southwest 37079.37
## 477   24   male 28.500        0    yes northeast 35147.53
## 489   44 female 38.060        0    yes southeast 48885.14
## 501   29   male 34.400        0    yes southwest 36197.70
## 525   42   male 26.070        1    yes southeast 38245.59
## 531   57   male 42.130        1    yes southeast 48675.52
## 550   43 female 46.200        0    yes southeast 45863.21
## 559   35 female 34.105        3    yes northwest 39983.43
## 570   48   male 40.565        2    yes northwest 45702.02
## 578   31 female 38.095        1    yes northeast 58571.07
```

```
## 588   34 female 30.210        1    yes northwest 43943.88
## 610   30   male 37.800        2    yes southwest 39241.44
## 616   47 female 36.630        1    yes southeast 42969.85
## 622   37   male 34.100        4    yes southwest 40182.25
## 624   18   male 33.535        0    yes northeast 34617.84
## 630   44 female 38.950        0    yes northwest 42983.46
## 666   43   male 38.060        2    yes southeast 42560.43
## 668   40 female 32.775        2    yes northwest 40003.33
## 669   62   male 32.015        0    yes northeast 45710.21
## 675   44 female 43.890        2    yes southeast 46200.99
## 678   60   male 31.350        3    yes northwest 46130.53
## 683   39   male 35.300        2    yes southwest 40103.89
## 690   27   male 31.130        1    yes southeast 34806.47
## 698   41   male 35.750        1    yes southeast 40273.65
## 707   51 female 38.060        0    yes southeast 44400.41
## 726   30 female 39.050        3    yes southeast 40932.43
## 737   37 female 38.390        0    yes southeast 40419.02
## 739   23   male 31.730        3    yes northeast 36189.10
## 740   29   male 35.500        2    yes southwest 44585.46
## 743   53   male 34.105        0    yes northeast 43254.42
## 760   18   male 38.170        0    yes southeast 36307.80
## 804   18 female 42.240        0    yes southeast 38792.69
## 820   33 female 35.530        0    yes northwest 55135.40
## 827   56   male 31.790        2    yes southeast 43813.87
## 829   41   male 30.780        3    yes northeast 39597.41
## 843   23 female 32.780        2    yes southeast 36021.01
## 846   60 female 32.450        0    yes southeast 45008.96
## 851   37 female 30.780        0    yes northeast 37270.15
## 853   46 female 35.530        0    yes northeast 42111.66
## 857   48 female 33.110        0    yes southeast 40974.16
## 884   51 female 37.050        3    yes northeast 46255.11
## 894   47   male 38.940        2    yes southeast 44202.65
## 902   60   male 40.920        0    yes southeast 48673.56
## 918   45   male 22.895        0    yes northeast 35069.37
## 948   37   male 34.200        1    yes northeast 39047.29
## 952   51   male 42.900        2    yes southeast 47462.89
## 954   44   male 30.200        2    yes southwest 38998.55
## 957   54   male 30.800        1    yes southeast 41999.52
## 959   43   male 34.960        1    yes northeast 41034.22
## 1013  61 female 33.330        4     no southeast 36580.28
## 1022  22 female 31.020        3    yes southeast 35595.59
## 1023  47   male 36.080        1    yes southeast 42211.14
## 1032  55 female 35.200        0    yes southeast 44423.80
## 1037  22   male 37.070        2    yes southeast 37484.45
## 1038  45 female 30.495        1    yes northwest 39725.52
## 1050  49   male 30.900        0    yes southwest 39727.61
## 1063  59   male 41.140        1    yes southeast 48970.25
## 1071  37   male 37.070        1    yes southeast 39871.70
## 1079  28   male 31.680        0    yes southeast 34672.15
## 1091  47   male 36.190        0    yes southeast 41676.08
## 1097  51 female 34.960        2    yes northeast 44641.20
## 1112  38   male 38.390        3    yes southeast 41949.24
## 1118  25   male 33.330        2    yes southeast 36124.57
## 1119  33   male 35.750        1    yes southeast 38282.75
```

```
## 1123  53 female 36.860        3    yes northwest 46661.44
## 1125  23 female 42.750        1    yes northeast 40904.20
## 1140  19 female 32.490        0    yes northwest 36898.73
## 1147  60   male 32.800        0    yes southwest 52590.83
## 1153  43 female 32.560        3    yes southeast 40941.29
## 1157  19   male 44.880        0    yes southeast 39722.75
## 1187  20   male 35.625        3    yes northwest 37465.34
## 1207  59 female 34.800        2     no southwest 36910.61
## 1208  36   male 33.400        2    yes southwest 38415.47
## 1219  46 female 34.600        1    yes southwest 41661.60
## 1231  52   male 34.485        3    yes northwest 60021.40
## 1241  52   male 41.800        2    yes southeast 47269.85
## 1242  64   male 36.960        2    yes southeast 49577.66
## 1250  32   male 33.630        1    yes northeast 37607.53
## 1285  61   male 36.300        1    yes southwest 47403.88
## 1289  20   male 39.400        2    yes southwest 38344.57
## 1292  19   male 34.900        0    yes southwest 34828.65
## 1301  45   male 30.360        0    yes southeast 62592.87
## 1302  62   male 30.875        3    yes northwest 46718.16
## 1304  43   male 27.800        0    yes southwest 37829.72
## 1314  19 female 34.700        2    yes southwest 36397.58
## 1324  42 female 40.370        2    yes southeast 43896.38
```

```
data <- data[-which(data$charges %in% outliers.charges), ]

#check to make sure outliers were removed
data[which(data$bmi %in% outliers.bmi), ]
```

```
## [1] age      sex      bmi      children smoker   region   charges
## <0 rows> (or 0-length row.names)
```

```
data[which(data$charges %in% outliers.charges), ]
```

```
## [1] age      sex      bmi      children smoker   region   charges
## <0 rows> (or 0-length row.names)
```

```
## transformatting categorical data into numerical
data$sex <- as.numeric(data$sex)
data$smoker <- as.numeric(data$smoker)
data$region <- as.numeric(data$smoker)
```

b) Split the data set into a training set and a test set, with ratio of 7:3. (2 mark)

The data was normalised and split into into training and test datasets with a ratio of 7:3

```
# set seed for replication of results
set.seed(334)

#create function to normalise the data to aid in creating the linear model
normalize <- function(x){
  num <- x - min(x)
  denom <- max(x) - min(x)
```

```r
  return(num/denom)
}

#normalise the data
normdata <- as.data.frame(lapply(data, normalize))

# divide data into training and testing sets
D <- 7
N <- 1193
train.len <- 835
train.index <- sample(1:N,train.len)
train.data <- normdata[train.index,  1:D]
test.data <- normdata[-train.index, 1:D]
```

c) Implement a linear regression model and train the model with your training data. Visualize the parameter updating process, test error (RMSE) in each iteration, and cost convergence process. Please be advised that built-in models in any realeased R package, like glm, is not allowed to use in this question. You can choose your preferred learning rate and determine the best iteration number. (8 marks)

```r
###create function to calculate coefficients for linear model using charges as the response variable
#data: the whole data frame
#target: column name that serves as the output
#learning rate: learning rate for the gradient descent algoritm
#iteration: stop criterion: maximum iterations allowed for training the gradient descent algorith
#epsilon: stop criterion: if the trained parameter's difference between the two interations is less tha
GradD <- function( data, target, learning_rate, iteration,
                               epsilon = .001, method  )
{
  # separate the input and output variables
  input  <- data %>% select( -one_of(target) ) %>% as.matrix()
    output <- data %>% select( one_of(target) ) %>% as.matrix()

  # implementation trick, after the normalizing the original input column
  # add a new column of all 1's to the first column, this serves as X0
  input <- cbind( theta0 = 1, input )

  # theta_new : initialize the theta value as all 1s
  # theta_old : a random number whose absolute difference between new one is
  #             larger than than epsilon
  theta_new <- matrix( 1, ncol = ncol(input) )
  theta_old <- matrix( 2, ncol = ncol(input) )

  # cost function
  costs <- function( input, output, theta )
  {
    sum( ( input %*% t(theta) - output )^2 ) / ( 2) #* nrow(output) )
  }

  # records the theta and cost value for visualization ; add the inital guess
  theta_trace <- vector( mode = "list", length = iteration )
  theta_trace[[1]] <- theta_new
  costs_trace <- numeric( length = iteration )
```

```
  costs_trace[1] <- costs( input, output, theta_old )

  # first derivative of the cost function
    derivative <- function( input, output, theta, step )
    {
      error <- ( input %*% t(theta) ) - output
      descent <- ( t(input) %*% error ) / nrow(output)
      return( t(descent) )
    }
  # keep updating as long as any of the theta difference is still larger than epsilon
  # or exceeds the maximum iteration allowed
  step <- 1
  while( any( abs(theta_new - theta_old) > epsilon ) & step <= iteration )
  {
    step <- step + 1

    # gradient descent
    theta_old <- theta_new
    theta_new <- theta_old - learning_rate * derivative( input, output, theta_old, step )

    # record keeping
    theta_trace[[step]] <- theta_new
    costs_trace[step]   <- costs( input, output, theta_new )
  }


  # returns the cost and theta record
  costs <- data.frame( iteration = iteration, costs = costs_trace )
  theta <- data.frame( do.call( rbind, theta_trace ), row.names = NULL )

  return( list( costs = costs, theta = theta) )
}
```

```
#create x and y variables using insurance data
set.seed(223)
#run gradient descent function
y <- as.matrix(train.data$charges)
gdmodel <- GradD(data = train.data, target = 'charges',  learning_rate = 0.6, iteration  = 1500, epsilo

#use lm function to check parameters
lm1 <- lm(charges ~., data = train.data)

parameters_gd <- gdmodel$theta[ nrow(gdmodel$theta), ]
```

The gradient descent function was created (above). *train.data* was put through the algorithm using a learning rate of 0.6, with 2000 iterations and epsilon = 10^-10. This combination produced values close to the paramaters produced by using the lm() function as shown in the table below.

model	Intercept/theta0	age	sex	bmi	children	smoker	region
gdmodel	0.0378451598772652	`'r parameters_gd['age']`	`'r parameters_gd['sex']`	`'r parameters_gd['bmi']`	`'r parameters_gd['children']`	`'r parameters_gd['smoker']`	`'r parameters_gd['region']`
lm1	0.0378452	0.3197011	-0.0060906	0.0483259	0.0561027	0.4479719	NA

The model given by the training data is as follows

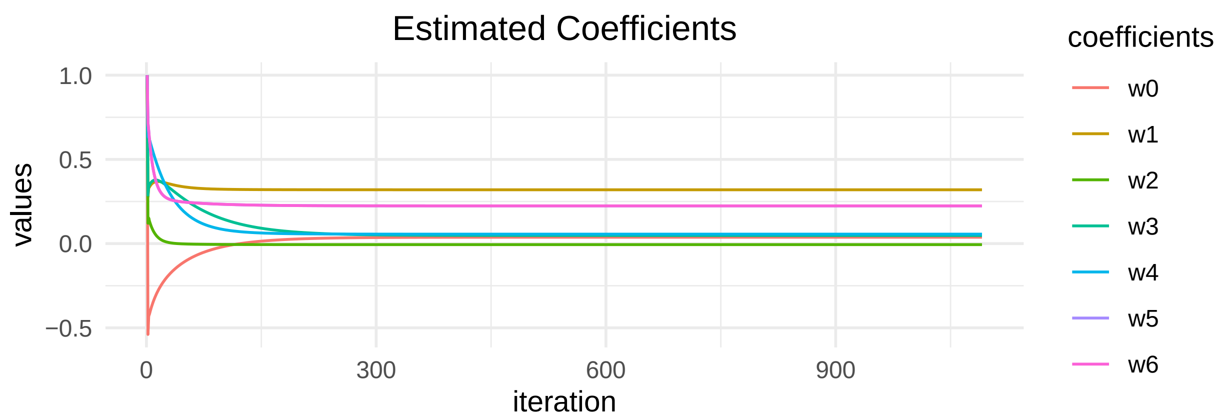
$$charges = 0.0378452 + 0.3197011$$

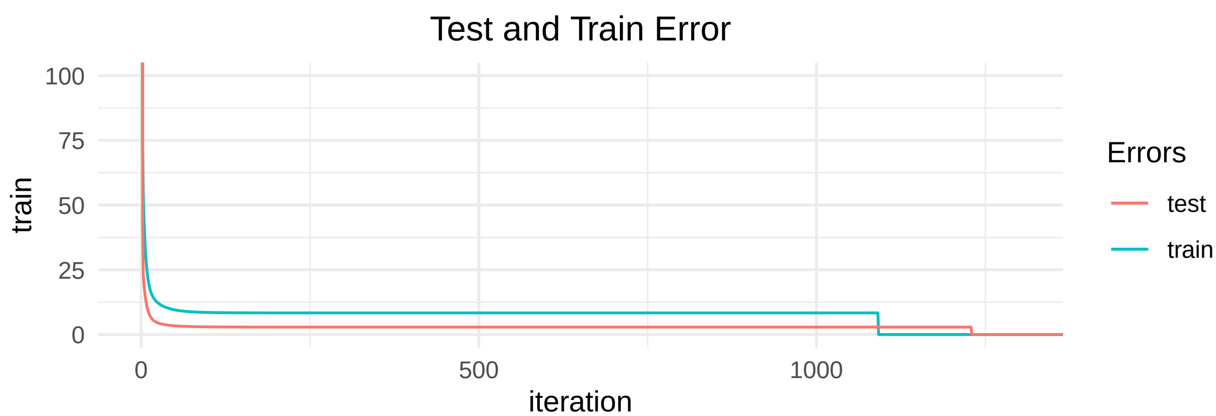
```
W.m <- as.data.frame(gdmodel$theta); names(W.m)<-c('w0','w1','w2','w3','w4', 'w5', 'w6')
W.m$iteration <-1:nrow(gdmodel$theta)
W.m <-melt(W.m, id='iteration'); names(W.m) <- c('iteration', 'coefficients', 'values')
gd.coeff <- ggplot(data=W.m, aes(x=iteration, y=values, color=coefficients)) + geom_line() + ggtitle('E
```

```
gdmodel.test <- GradD(data = test.data, target = 'charges', learning_rate = 0.6, iteration = 1500, epsi

parameters_gdt <- gdmodel.test$theta[ nrow(gdmodel.test$theta), ]
parameters_gdt
```

```
##            theta0       age         sex       bmi   children   smoker   region
## 1229 0.0002987254 0.3776689 -0.01360385 0.0605143 0.07307292 0.205776 0.205776
```

```
error <- data.frame('iteration' = 1:1500)
error['train'] <- as.matrix(cbind(gdmodel$costs[,2]))
error['test'] <- as.matrix(cbind(gdmodel.test$costs[,2]))

error <- as.data.frame(error); names(error)<-c('iteration', 'train', 'test')


gd.error <- ggplot(data=error, aes(x = iteration)) +
    geom_line(aes(y = train, colour = "train")) + geom_line(aes(y = test, colour = "test")) + ggtitle('
grid.arrange(gd.coeff, gd.error, nrow = 2)
```





4. Evaluate your model by calculating the RMSE, and visualizing the residuals of test data. Please note that explanation of your residual plot is needed. (5 marks)

5. Does your model overfit? Which features do you think are not significant? Please justify your answers. For example, you can analyze the significance of a feature from correlation, variance, etc. (8 marks)

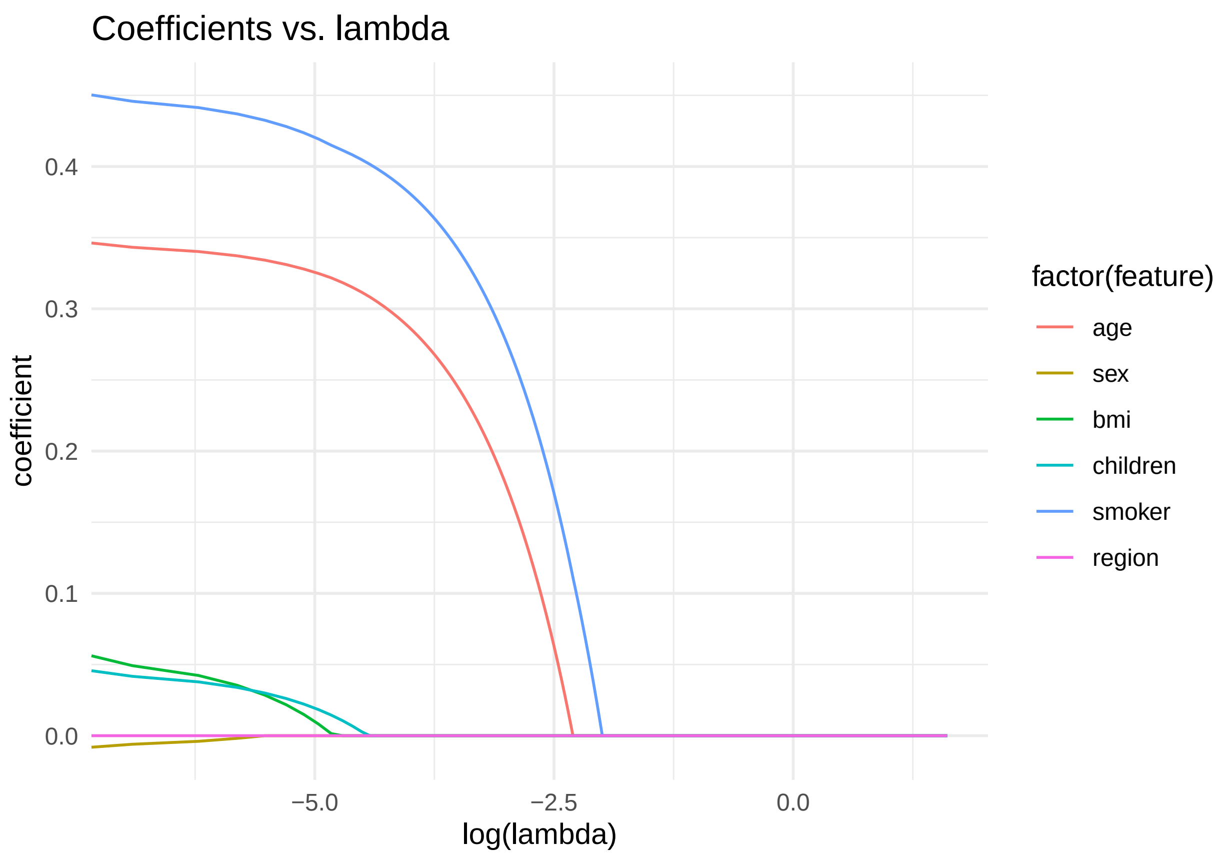
6. Use the glmnet library to biult two linear regression models with Lasso and Ridge regularization, respectively. In comparison to your model, how well do these two models perform? Do the regularized models automatically filter out the less significant features? What are the differences of these two models? Please justify your answers. (8 marks)

```
train.len <- 835
train.index <- sample(1:N,train.len)
train.data1 <- normdata[train.index,  1:6]
train.label <- normdata[train.index, 'charges']
test.data1 <- normdata[-train.index, 1:6]
test.label <- normdata[-train.index, 'charges']
```

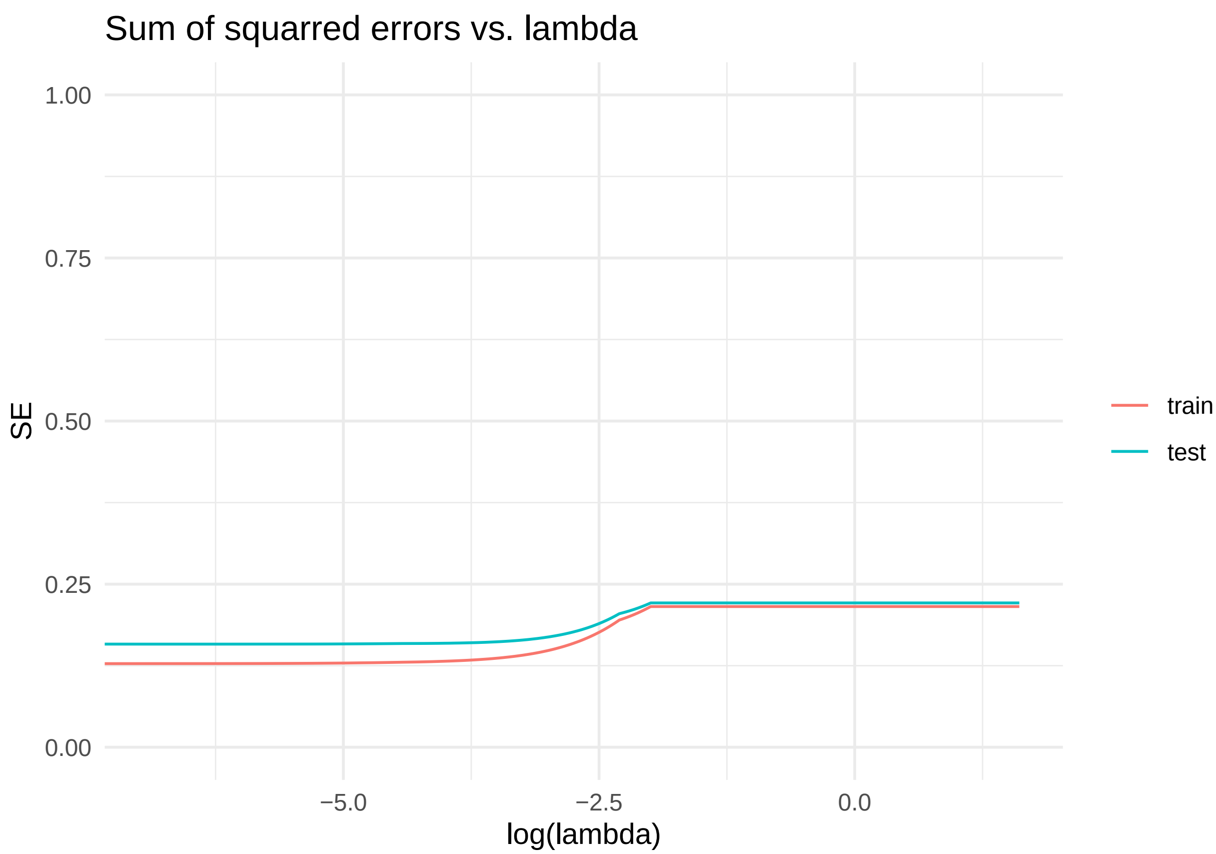
```
#run an linear regression using r package to check coefficients
library(glmnet)
fitAndPlot <- function(train.data1, train.label, alpha=0, lambda = c(0:5000)/1000){
    # fit the model
    fit <- glmnet(x = as.matrix(train.data1), y=train.label, alpha = alpha, lambda = lambda)

    # aggrigate the outputs
    out <- as.data.frame(as.matrix(t(fit$beta)))
```

```
    out[,c('nonzero', 'lambda')] <- c(fit$df, fit$lambda)

    # reshape the outputs (for plotting)
    out.m<-melt(out, id=c('lambda', 'nonzero'))
    names(out.m) <- c('lambda', 'nonzero', 'feature', 'coefficient')

    # plot coefficients vs lambda
    g <- ggplot(data = out.m, aes(x=log(lambda), y=coefficient, color=factor(feature))) + geom_line() +
        ggtitle('Coefficients vs. lambda') + theme_minimal()
    print(g)

    # plot number of nonzero coefficients (as ameasure of model complexity) vs lambda
    #g <- ggplot(data = out.m, aes(x=log(lambda), y=nonzero)) + geom_line() +
    #    scale_color_discrete(guide = guide_legend(title = NULL)) +
    #    ggtitle('Nonzero Coefficients vs. lambda') + theme_minimal()
    #print(g)

    # run the predictions
    train.predict <- predict(fit, newx=as.matrix(train.data1))
    test.predict <- predict(fit, newx=as.matrix(test.data1))

    # calculate the standard errors
    error <- data.frame('lambda' = out$lambda,
                    'train' = sqrt(colSums((train.predict - train.label)^2)/nrow(train.predict)),
                    'test' = sqrt(colSums((test.predict - test.label)^2)/nrow(test.predict)))
    error.m <- melt(error, id='lambda')
    names(error.m) <- c('lambda', 'set', 'SE')

    # plot sum of squarred error for train and test sets vs lambda
    g <- ggplot(data = error.m, aes(x= log(lambda), y = SE, color = factor(set))) + geom_line() +  ylim
        scale_color_discrete(guide = guide_legend(title = NULL)) +
        ggtitle('Sum of squarred errors vs. lambda') + theme_minimal()
    print(g)
}
```

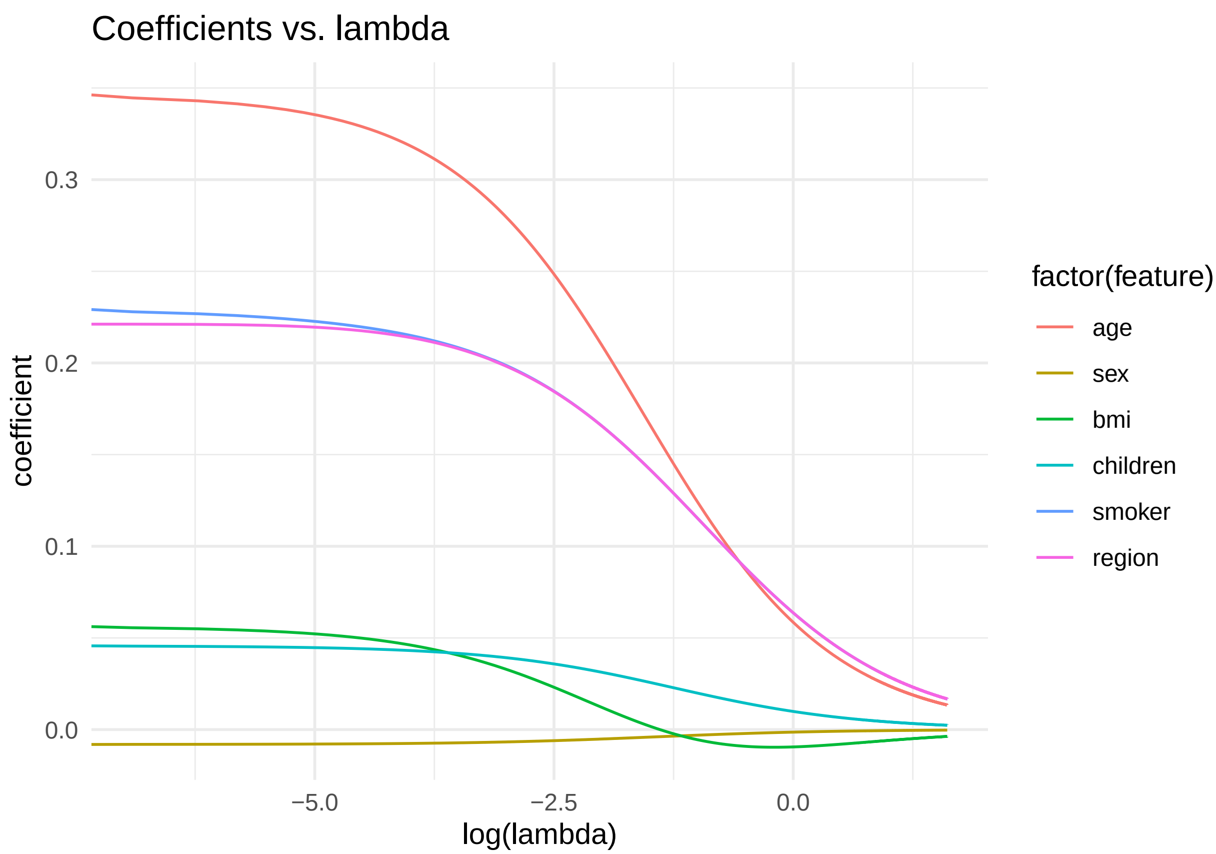
```
##LASSO
#x = as.matrix(train.data[1:6]), y = as.matrix(train.data$charges)
fitAndPlot (train.data1, train.label, alpha=1, lambda = c(0:5000)/1000)
```

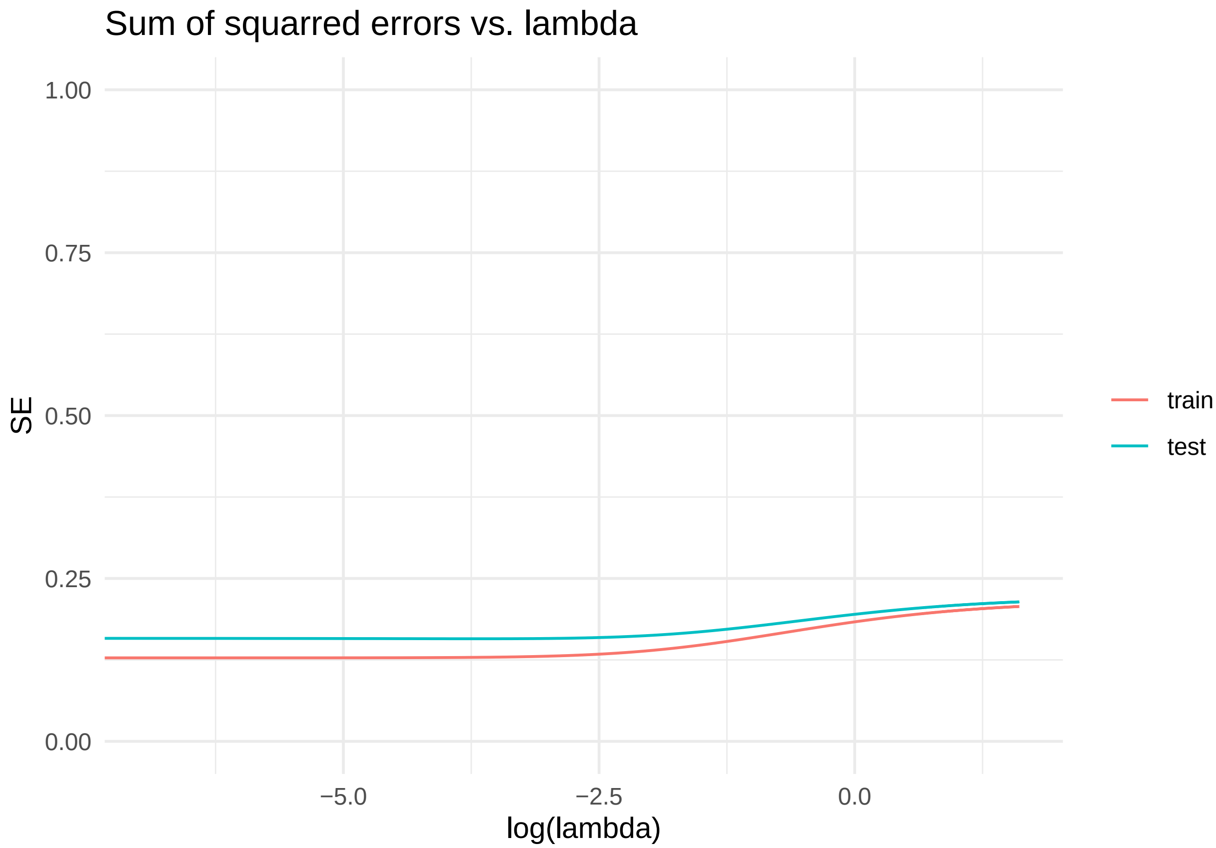


Coefficients vs. lambda
coefficient
0.0
0.1
0.2
0.3
0.4
−5.0
−2.5
0.0
log(lambda)
factor(feature)
age
sex
bmi
children
smoker
region




```r
##Ridge Regularisation
fitAndPlot (train.data1, train.label, alpha=0, lambda = c(0:5000)/1000)
```


Coefficients vs. lambda
coefficient
0.3
0.2
0.1
0.0
−5.0
−2.5
0.0
log(lambda)
factor(feature)
age
sex
bmi
children
smoker
region



Question 3 - Logistic Regression (45 marks) 1

In this question, you are required to implement a Logistic Regression model to classify whether a person donated blood at a Blood Transfusion Service Center in March 2007. Please read the sub-questions below carefully for the deteailed instructions.

1. Check out the Blood Transfusion Service Center Data Set at https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center.
2. Perform data preprocessing to determine and remove invalid samples. Split the data into a training set and a test set with a ratio of 7:3. (2 marks)

```
#read in transfusion data file
transfusion <- read.csv('transfusion.data')

#perform data pre-processing
sum(is.na(transfusion)) #appears to be no missing values
```

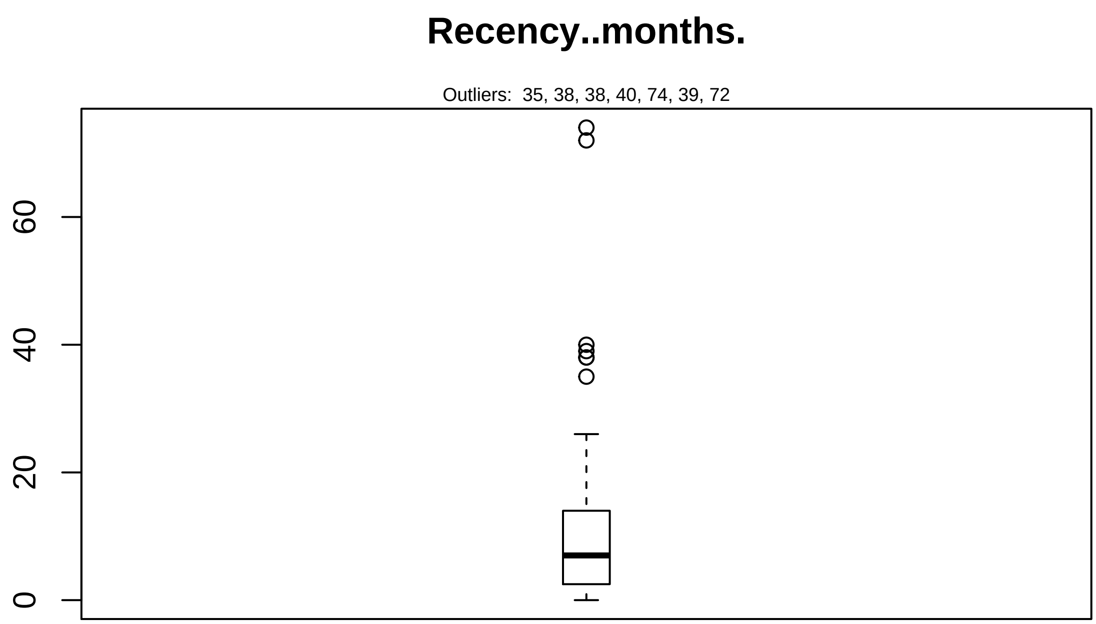
```
## [1] 0
```

```
summary(transfusion)
```

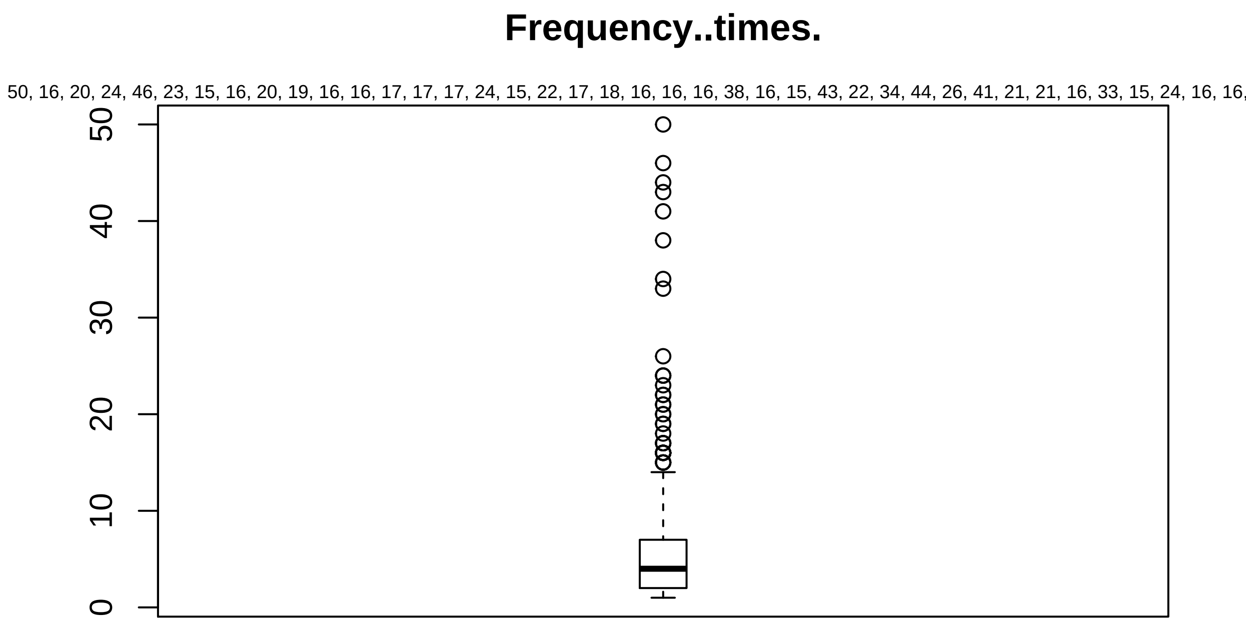
```
##  Recency..months. Frequency..times. Monetary..c.c..blood. Time..months.
##  Min.   : 0.000   Min.   : 1.000    Min.   :  250         Min.   : 2.00
##  1st Qu.: 2.750   1st Qu.: 2.000    1st Qu.:  500         1st Qu.:16.00
##  Median : 7.000   Median : 4.000    Median : 1000         Median :28.00
##  Mean   : 9.507   Mean   : 5.515    Mean   : 1379         Mean   :34.28
```

```
##   3rd Qu.:14.000   3rd Qu.: 7.000    3rd Qu.: 1750         3rd Qu.:50.00
##   Max.   :74.000   Max.   :50.000    Max.   :12500         Max.   :98.00
##   whether.he.she.donated.blood.in.March.2007
##   Min.   :0.000
##   1st Qu.:0.000
##   Median :0.000
##   Mean   :0.238
##   3rd Qu.:0.000
##   Max.   :1.000
```

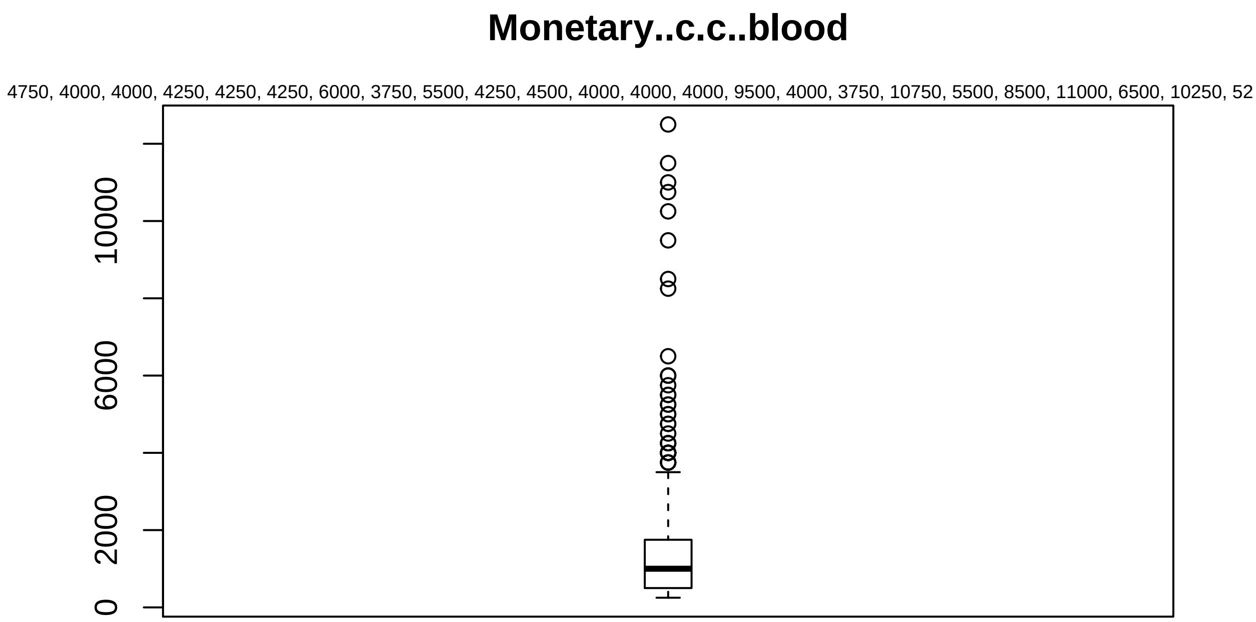
```r
##check for outliers
outliers.recency <- boxplot.stats(transfusion$Recency..months.)$out
boxplot(transfusion$Recency..months., main = "Recency..months.", boxwex = 0.1)
mtext(paste("Outliers: ", paste(outliers.recency, collapse = ", ")), cex = 0.6)
```



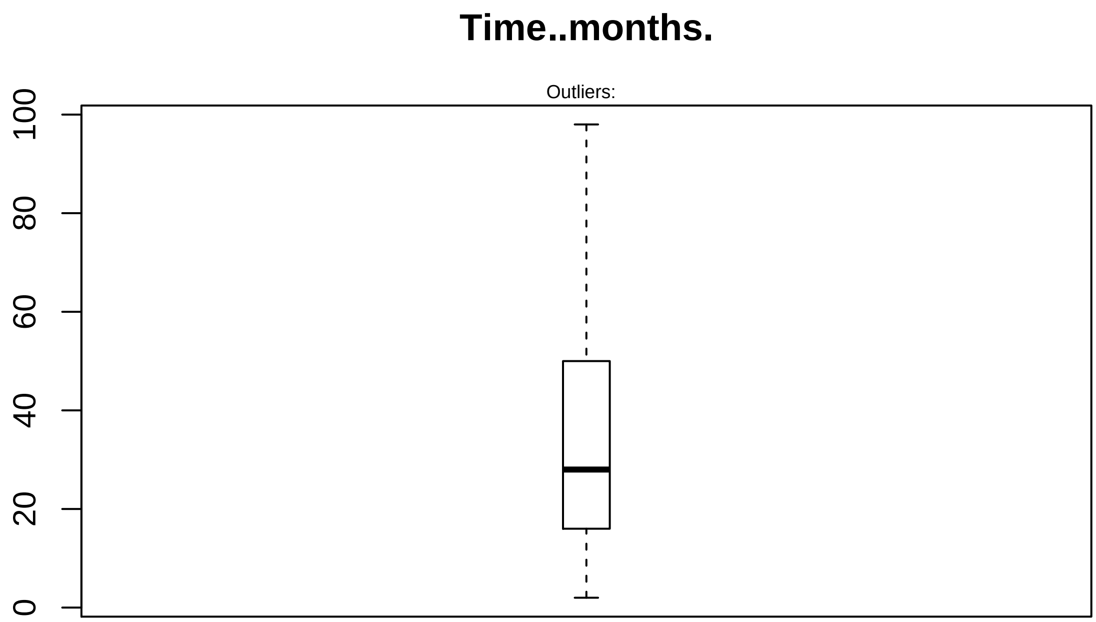

```r
outliers.freq <- boxplot.stats(transfusion$Frequency..times.)$out
boxplot(transfusion$Frequency..times., main = "Frequency..times.", boxwex = 0.1)
mtext(paste("Outliers: ", paste(outliers.freq, collapse = ", ")), cex = 0.6)
```




```
outliers.monetary <- boxplot.stats(transfusion$Monetary..c.c..blood.)$out
boxplot(transfusion$Monetary..c.c..blood., main = "Monetary..c.c..blood", boxwex = 0.1)
mtext(paste("Outliers: ", paste(outliers.monetary, collapse = ", ")), cex = 0.6)
```




```r
outliers.time <- boxplot.stats(transfusion$Time..months.)$out
boxplot(transfusion$Time..months., main = "Time..months.", boxwex = 0.1)
mtext(paste("Outliers: ", paste(outliers.time, collapse = ", ")), cex = 0.6)
```




```r
#find which rows contain Recency outliers and remove rows
transfusion[which(transfusion$Recency..months. %in% outliers.recency), ]
```

```
##     Recency..months. Frequency..times. Monetary..c.c..blood. Time..months.
## 496               35                 3                   750            64
## 497               38                 1                   250            38
## 498               38                 1                   250            38
## 499               40                 1                   250            40
## 500               74                 1                   250            74
## 747               39                 1                   250            39
## 748               72                 1                   250            72
##     whether.he.she.donated.blood.in.March.2007
## 496                                          0
## 497                                          0
## 498                                          0
## 499                                          0
## 500                                          0
## 747                                          0
## 748                                          0
```

```r
transfusion <- transfusion[-which(transfusion$Recency..months. %in% outliers.recency), ]

#find which rows contain Frequency outliers and remove rows
transfusion[which(transfusion$Frequency..times. %in% outliers.freq), ]
```

```
##     Recency..months. Frequency..times. Monetary..c.c..blood. Time..months.
## 1                  2                50                 12500            98
## 3                  1                16                  4000            35
## 4                  2                20                  5000            45
## 5                  1                24                  6000            77
## 10                 5                46                 11500            98
## 11                 4                23                  5750            58
## 18                 2                15                  3750            49
## 35                 2                16                  4000            64
## 45                 4                20                  5000            69
## 56                 4                19                  4750            69
## 59                 2                16                  4000            81
## 66                 3                16                  4000            74
## 73                 4                17                  4250            71
## 97                 3                17                  4250            86
## 106                6                17                  4250            70
## 116               11                24                  6000            64
## 189                8                15                  3750            77
## 242               11                22                  5500            98
## 244               11                17                  4250            79
## 279               14                18                  4500            78
## 281               14                16                  4000            70
## 321               15                16                  4000            82
## 328               14                16                  4000            98
## 342               23                38                  9500            98
## 360               21                16                  4000            64
## 367               23                15                  3750            57
## 501                2                43                 10750            86
## 502                6                22                  5500            28
## 503                2                34                  8500            77
## 504                2                44                 11000            98
## 505                0                26                  6500            76
## 506                2                41                 10250            98
## 507                3                21                  5250            42
## 509                2                21                  5250            52
## 515                4                16                  4000            38
## 518                4                33                  8250            98
## 528                2                15                  3750            64
## 529                5                24                  6000            79
## 543                4                16                  4000            70
## 563                4                16                  4000            98
## 634               12                15                  3750            71
## 636               11                16                  4000            89
## 656               16                16                  4000            77
## 673               16                15                  3750            87
## 678               23                19                  4750            62
##     whether.he.she.donated.blood.in.March.2007
## 1                                            1
## 3                                            1
## 4                                            1
## 5                                            0
## 10                                           1
## 11                                           0
## 18                                           1
```

```
## 35                                          0
## 45                                          1
## 56                                          1
## 59                                          0
## 66                                          0
## 73                                          1
## 97                                          0
## 106                                         0
## 116                                         0
## 189                                         0
## 242                                         0
## 244                                         1
## 279                                         0
## 281                                         0
## 321                                         0
## 328                                         0
## 342                                         0
## 360                                         0
## 367                                         0
## 501                                         1
## 502                                         1
## 503                                         1
## 504                                         0
## 505                                         1
## 506                                         1
## 507                                         1
## 509                                         1
## 515                                         1
## 518                                         1
## 528                                         0
## 529                                         0
## 543                                         1
## 563                                         1
## 634                                         0
## 636                                         0
## 656                                         0
## 673                                         0
## 678                                         0
```

```
transfusion <- transfusion[-which(transfusion$Frequency..times. %in% outliers.freq), ]
```

```
 nor <-function(x) { (x -min(x))/(max(x)-min(x))   }

 ##Run nomalization on first 4 coulumns of dataset because they are the predictors
 transfusion_norm <- as.data.frame(lapply(transfusion, nor))
ran <- sample(1:nrow(transfusion), 0.7 * nrow(transfusion))
transfusion.x <- transfusion[ran, -5]
transfusion.y <- transfusion[ran, 5]
```

3. Develop a Logistic Regression model that use batch gradient descent for optimization. Visualize the parameter updating process, test error (ACC) in each iteration, and the cost convergence process. Please note that you need to develop your model step-by-step, built-in models in any realeased R package, like glm, is not allowed to use in this question. (10 marks)

```r
# auxiliary function that predicts class labels
predict <- function(w, X, c0, c1){
    sig <- sigmoid(w, X)
    return(ifelse(sig>0.5, c1,c0))
}

# auxiliary function that calculate a cost function
cost <- function (w, X, T, c0){
    sig <- sigmoid(w, X)
    return(sum(ifelse(T==c0, 1-sig, sig)))
}
```

```r
# Sigmoid function (=p(C1|X))
sigmoid <- function(w, x){
    return(1.0/(1.0+exp(-w%*%t(cbind(1,x)))))
}
```

```r
# Initializations
# tau.max <- 1000 # maximum number of iterations
# eta <- 0.01 # learning rate
# epsilon <- 0.01 # a threshold on the cost (to terminate the process)
# tau <- 1 # iteration counter
# terminate <- FALSE

## Just a few name/type conversion to make the rest of the code easy to follow
# X <- as.matrix(train.data) # rename just for conviniance
# T <- ifelse(train.label==c0,0,1) # rename just for conviniance

# W <- matrix(,nrow=tau.max, ncol=(ncol(X)+1)) # to be used to store the estimated coefficients
# W[1,] <- runif(ncol(W)) # initial weight (any better idea?)

# project data using the sigmoid function (just for convenient)
# Y <- sigmoid(W[1,],X)

# costs <- data.frame('tau'=1:tau.max)  # to be used to trace the cost in each iteration
# costs[1, 'cost'] <- cost(W[1,],X,T, c0)
```

```r
# while(!terminate){
    # check termination criteria:
    # terminate <- tau >= tau.max | cost(W[tau,],X,T, c0)<=epsilon

    # shuffle data:
    # X <- X[train.index,]
    # T <- T[train.index]

    # for each datapoint:
    # for (i in 1:train.len){
        # check termination criteria:
        # if (tau >= tau.max | cost(W[tau,],X,T, c0) <=epsilon) {terminate<-TRUE;break}

        # Y <- sigmoid(W[tau,],X)

        # Update the weights
```

```r
        # W[(tau+1),] <- W[tau,] - eta * (Y[i]-T[i]) * cbind(1, t(X[i,]))

        # record the cost:
        # costs[(tau+1), 'cost'] <- cost(W[tau,],X,T, c0)

        # update the counter:
        # tau <- tau + 1

        # decrease learning rate:
        # eta = eta * 0.999
    # }
# }
# Done!
# costs <- costs[1:tau, ] # remove the NaN tail of the vector (in case of early stopping)

# the  final result is:
# w <- W[tau,]
# cat('\nThe  final coefficents are:',w)
```

4. Invesitigate the influence of different learning rate to the training process and answer what happend if you apply a too small or a too large learning rate. (5 marks)
5. Expermently compare batch gradient descent and stochastic gradient descent and discuss your findings (e.g., convergence speed). Visualize the comparison in terms of updating process and the cost convergence process. (6 marks)
6. Develop a K-fold (K = 5) cross validation to evaluate your model in step 3. Please note that you need to write R codes to explicitly show how you perform the K-fold cross validation. Built-in validation methods are not allowed to use. Different metrics, e.g. ACC, Recall, precision, etc, should be used to evaluate your model. (8 marks)
7. Use different values of K (from 5 to N, where N denotes the sample number) and summarize the corresponding changes of your model performances. Visualize and explain the changes. (6 marks)
8. How can you modify the cost function to prevent overfitting? Discuss the possibility of adding regularization term(s) and summarize the possible changes in the gradient descent process. (8 marks)