APANPS5205 Homework 2

For this assignment, the following libraries are required, please install and load them by running the following:

```
required_packages = c("rpart", "rpart.plot", "psych", "randomForest")
packages_to_install = setdiff(required_packages, installed.packages()[,"Package"])

if (length(packages_to_install)!=0) {
  install.packages(packages_to_install)
}

library(rpart)
library(rpart.plot)
library(psych)
library(randomForest)
library(bbmle)
library(Deriv)

set.seed(0)
```

Question 1. Maximum Likelihood

1.1 Explain the maximum likelihood estimation process. Intuitively, what is the goal of the likelihood function? Why do we take derivative of the log likelihood? (3 pts)

The MLE process involves estimating the likelihood that a given, randomized, set of data falls into a certain probability. The goal is to represent a randomized sample as best as possible - the derivative of log likelihood is used, as it eliminates the deviation of the population. MLE can be categorized as inferertial statistics.

1.2 Estimate coin toss bias (4 pts)

```
# read in coin toss data and store data in a vector

coin_toss = read.csv("coin_toss.csv")
vec = unlist(coin_toss[,1])
vec
```

```
##    [1] 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1
##   [38] 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0
##   [75] 1 0 1 0 1 1 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1
##  [112] 1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 1 0 1
##  [149] 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0
##  [186] 0 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1
##  [223] 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1
##  [260] 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1
```

```
##  [297] 1 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0
##  [334] 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1
##  [371] 1 0 0 0 0 1 0 1 1 1 0 1 1 1 1 0 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 1 0
##  [408] 0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0
##  [445] 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1
##  [482] 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0
##  [519] 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1
##  [556] 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0
##  [593] 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1
##  [630] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0
##  [667] 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0
##  [704] 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
##  [741] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1
##  [778] 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1
##  [815] 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0
##  [852] 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1
##  [889] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 0 1
##  [926] 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0
##  [963] 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1
## [1000] 1
```

```
# write negative log-likelihood function for bernoulli trial

log_likelihood = function(pi) {-log((pi^(sum(vec[vec==1])))*((1-pi)^(sum(vec==0))))}
log_likelihood2 = function(pi) {-log((pi^(sum(vec2[vec2==1])))*((1-pi)^(sum(vec2==0))))}
log_likelihood3 = function(pi) {-log((pi^(sum(vec3[vec3==1])))*((1-pi)^(sum(vec3==0))))}


# use "optimize" to find the optimal parameter

opt = optimize(log_likelihood, interval = c(0,1))
```

```
## Warning in optimize(log_likelihood, interval = c(0, 1)): NA/Inf replaced by
## maximum positive value
```

```
# report the optimal parameter

opt
```

```
## $minimum
## [1] 0.7349974
##
## $objective
## [1] 578.2221
```

```
# using only the first 50 coin tosses, repeat the process

vec2 = unlist(coin_toss[1:50,])
opt_50 = optimize(log_likelihood2, interval = c(0,1))

opt_50
```

```
## $minimum
## [1] 0.7999832
##
## $objective
## [1] 25.02012
```

```r
# using only the first 100 coin tosses, repeat the process

vec3 = unlist(coin_toss[1:100,])
opt_100 = optimize(log_likelihood3, interval = c(0,1))
opt_100
```

```
## $minimum
## [1] 0.7000058
##
## $objective
## [1] 61.08643
```

1.3 We can also derive the maximum likelihood estimator for a series of coin flip. Using this estimator, what is the maximum likelihood estimate? (2 pts)

```r
# mle estimate of all the data

MLE = mle2(log_likelihood, start=list(pi=.5))
```

```
## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced

## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced

## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced

## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced

## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced

## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced

## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced

## Warning in log((pi^(sum(vec[vec == 1]))) * ((1 - pi)^(sum(vec == 0)))): NaNs
## produced
```

```r
summary(MLE)
```

```
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = log_likelihood, start = list(pi = 0.5))
##
## Coefficients:
##    Estimate Std. Error z value     Pr(z)
## pi 0.735001   0.013956  52.665 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 1156.444
```

```r
MLE
```

```
##
## Call:
## mle2(minuslogl = log_likelihood, start = list(pi = 0.5))
##
## Coefficients:
##       pi
## 0.735001
##
## Log-likelihood: -578.22
```

```r
# mle estimate of random 50 points

MLE2 = mle2(log_likelihood2, start=list(pi=.6))
summary(MLE2)
```

```
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = log_likelihood2, start = list(pi = 0.6))
##
## Coefficients:
##    Estimate Std. Error z value     Pr(z)
## pi  20850.8     2948.7  7.0711 1.537e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: -994.5136
```

```r
# mle estimate of random 100 points

MLE3 = mle2(log_likelihood3, start=list(pi=.7))
summary(MLE3)
```

```
## Maximum likelihood estimation
```

```
##
## Call:
## mle2(minuslogl = log_likelihood3, start = list(pi = 0.7))
##
## Coefficients:
##    Estimate Std. Error z value     Pr(z)
## pi 0.700000   0.045826  15.275 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 122.1729
```

1.4 Comment on how the estimates at 50 or 100 differ from the estimates using all of the data. (1 pt)

With 50 flips of the coin, the MLE is very high (> 1). This is because the probability of predicting the outcome is high, considering the first few flips are heads (or 1). As more flips occur, the chances of guessing the side approaches 50%.

The estimates of 50 flips compared to 100 flips is noted as large, since the sample size is increase, and probability is evening out. The probability of successfully predicting the outcome asymptotically approaches 50%, as seen in the plot graph of pi(y) and number of outcomes(x).

Overall, it appears that the threshold of accuracy on the dataset is about 578 flips, even though the MLE is closer to .5 with 100 flips of the coin.

Question 2. Decision Tree

For this question, we will need to download the Cleveland Clinic heart disease data set from UCI Machine Learning Repository. We'll use decision tree to predict the probability of heart disease.

2.1 Data Preparation & Exploration (3 pts)

```
# read in the data from UCI Machine Learning Repository

heart_disease_url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cl

heart = read.csv(
  url(heart_disease_url),
  header = F,
  col.names = c("age", "sex", "cp", "trestbps", "chol",
                "fbs", "restecg", "thalach", "exang", "oldpeak", "slope",
                "ca", "thal", "num"))

# let's look at the summary of the data

summary(heart)
```

```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0
```

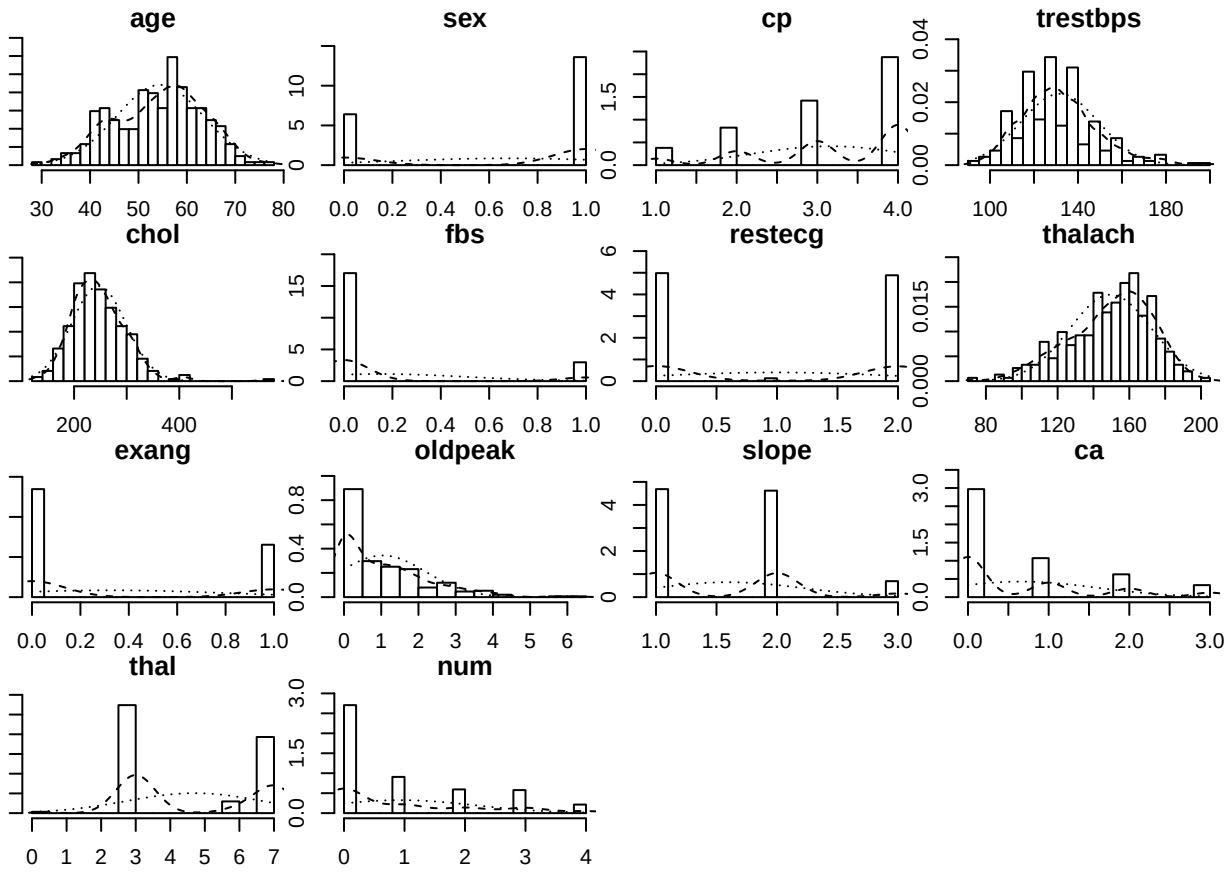
```
##  Median :56.00   Median :1.0000   Median :3.000   Median :130.0
##  Mean   :54.44   Mean   :0.6799   Mean   :3.158   Mean   :131.7
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
##       chol            fbs            restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
##  Median :241.0   Median :0.0000   Median :1.0000   Median :153.0
##  Mean   :246.7   Mean   :0.1485   Mean   :0.9901   Mean   :149.6
##  3rd Qu.:275.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##      exang           oldpeak         slope            ca
##  Min.   :0.0000   Min.   :0.00   Min.   :1.000   Length:303
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   Class :character
##  Median :0.0000   Median :0.80   Median :2.000   Mode  :character
##  Mean   :0.3267   Mean   :1.04   Mean   :1.601
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000
##  Max.   :1.0000   Max.   :6.20   Max.   :3.000
##      thal                num
##  Length:303          Min.   :0.0000
##  Class :character    1st Qu.:0.0000
##  Mode  :character    Median :0.0000
##                      Mean   :0.9373
##                      3rd Qu.:2.0000
##                      Max.   :4.0000
```

```r
# Notice that the variables "ca" and "thal" have "?" in the data.
# Let's remove those from our dataset.

heart[heart=="?"] = 0
heart = sapply(heart, as.numeric)
heart = data.frame(heart)

# use multi.hist() to plot histograms of all of the variables.
# Notice that the data is required to have numeric values.
# To convert your data to numeric, you can use sapply() and is.numeric().

multi.hist(heart)
```



Let's create our target feature. Our target feature should be an indicator that tells us if an individual has a heart disease (1) or not (0). To do this, we will need to transform the feature "num" to 1 if it's greater than zero and 0 if it's zero. Remove the "num" feature after the transformation. Let's label our target feature "disease".

```
# create target variable

heart$num[heart$num>0] = 1
heart$num[heart$num==0] = 0
heart$disease = heart$num
heart$disease = as.factor(heart$disease)

# remove "num"

heart$num = NULL
```

2.2 Data Splitting (2 pts)

Let's split our data into three sets: training (40%), validation (20%), and testing (40%). In other words, we're using 60% of our data to train and tune our model.

```
# split data into training, validation, and testing sets

split = sample(1:3, size=nrow(heart), prob=c(0.4,0.2,0.4), replace = TRUE)
train = heart[split==1,]
```

```
valid = heart[split==2,]
test = heart[split==3,]
```

2.3 Decision Tree Model Fitting & Parameter Tuning (5 pts)

We want to train a decision tree model that generalizes well. Let's vary two parameters in our decision tree model, maxdepth and minsplit, and see how they perform on the validation set. Let's vary maxdepth from 1 to 5 and minsplit from 10 to 30.

```
# loop through the parameter set and store the parameters, models,
# and accuracies on the validation set

library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
fitControl <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 10)
tuneGrid = expand.grid(maxdepth=seq(1,5))

#find best maxdepth

cvModel = train(disease~.,data=train,trControl = fitControl,method="rpart2",tuneGrid = tuneGrid)

cvModel$bestTune
```

```
##   maxdepth
## 5        5
```

```
#find best minsplit

x = 1
results2 = data.frame()
while (x<22){
```

```
cvModel2 = train(disease~.,data=train,method="rpart2", trControl = fitControl,   control=rpart.control(
results = data.frame(max(cvModel2$result$Accuracy))
results2 = rbind(results, results2)

x=1+x
}
```

```
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
```

```
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
##
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
```

```
results2$minsplit = 10:30
results3 <-results2[order(results2$max.cvModel2.result.Accuracy., results2$minsplit, decreasing = TRUE)

# report the best parameters (there can be more than one set, select one of them)

minsplit_best = results3[1, 2]
maxdepth_best = as.numeric(cvModel$bestTune)

# report the best score from the validation set

cvModel = train(disease~.,data=train,method="rpart2", trControl = fitControl,  control=rpart.control(cp
```

```
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
```

```
pred = predict(cvModel, valid)
pred = data.frame(pred)
confmat = cbind(pred, valid$disease)
confmat
```

```
##    pred valid$disease
## 1     1              0
## 2     0              0
## 3     1              1
## 4     1              0
## 5     1              0
## 6     0              0
## 7     1              1
## 8     1              0
## 9     0              0
## 10    1              1
## 11    0              0
## 12    1              0
## 13    1              1
## 14    1              1
## 15    1              0
```

```
## 16    1             0
## 17    0             0
## 18    1             0
## 19    1             1
## 20    1             1
## 21    1             1
## 22    1             1
## 23    0             0
## 24    0             0
## 25    0             0
## 26    0             0
## 27    1             0
## 28    1             1
## 29    1             0
## 30    1             0
## 31    1             1
## 32    1             0
## 33    0             0
## 34    1             1
## 35    1             0
## 36    0             0
## 37    0             1
## 38    1             1
## 39    0             0
## 40    1             0
## 41    1             0
## 42    0             0
## 43    0             0
## 44    1             1
## 45    1             0
## 46    1             0
## 47    1             0
## 48    0             0
## 49    1             1
## 50    1             1
```

```
mean(confmat$pred==confmat$`valid$disease`)
```

```
## [1] 0.62
```

Using the parameters above, let's compare how our model performs with different split metrics, information and ginni.

```
# train using information metric

cvModel_info = train(disease~.,data=train,method="rpart2", trControl = fitControl,  control=rpart.contr
```

```
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
```

```
# report accuracy on validation set
```

```r
pred = predict(cvModel_info, valid)
pred = data.frame(pred)
confmat = cbind(pred, valid$disease)
confmat
```

```
##    pred valid$disease
## 1     1              0
## 2     0              0
## 3     1              1
## 4     1              0
## 5     1              0
## 6     0              0
## 7     1              1
## 8     1              0
## 9     0              0
## 10    1              1
## 11    0              0
## 12    1              0
## 13    1              1
## 14    1              1
## 15    1              0
## 16    1              0
## 17    0              0
## 18    1              0
## 19    1              1
## 20    1              1
## 21    1              1
## 22    1              1
## 23    0              0
## 24    0              0
## 25    0              0
## 26    0              0
## 27    1              0
## 28    1              1
## 29    1              0
## 30    1              0
## 31    1              1
## 32    1              0
## 33    0              0
## 34    1              1
## 35    1              0
## 36    0              0
## 37    0              1
## 38    1              1
## 39    0              0
## 40    1              0
## 41    1              0
## 42    0              0
## 43    0              0
## 44    1              1
## 45    1              0
## 46    1              0
## 47    1              0
```

```
## 48    0             0
## 49    1             1
## 50    1             1
```

```
mean(confmat$pred==confmat$`valid$disease`)
```

```
## [1] 0.62
```

```
# train using ginni metric

cvModel_gini = train(disease~.,data=train,method="rpart2",trControl = fitControl,   control=rpart.contr
```

```
## note: only 2 possible values of the max tree depth from the initial fit.
##  Truncating the grid to 2 .
```

```
# report accuracy on validation set

pred = predict(cvModel_gini, valid)
pred = data.frame(pred)
confmat = cbind(pred, valid$disease)
confmat
```

```
##    pred valid$disease
## 1     1             0
## 2     0             0
## 3     1             1
## 4     1             0
## 5     1             0
## 6     0             0
## 7     1             1
## 8     1             0
## 9     0             0
## 10    1             1
## 11    0             0
## 12    1             0
## 13    1             1
## 14    1             1
## 15    1             0
## 16    1             0
## 17    0             0
## 18    1             0
## 19    1             1
## 20    1             1
## 21    1             1
## 22    1             1
## 23    0             0
## 24    0             0
## 25    0             0
## 26    0             0
## 27    1             0
## 28    1             1
## 29    1             0
```

```
## 30    1             0
## 31    1             1
## 32    1             0
## 33    0             0
## 34    1             1
## 35    1             0
## 36    0             0
## 37    0             1
## 38    1             1
## 39    0             0
## 40    1             0
## 41    1             0
## 42    0             0
## 43    0             0
## 44    1             1
## 45    1             0
## 46    1             0
## 47    1             0
## 48    0             0
## 49    1             1
## 50    1             1
```

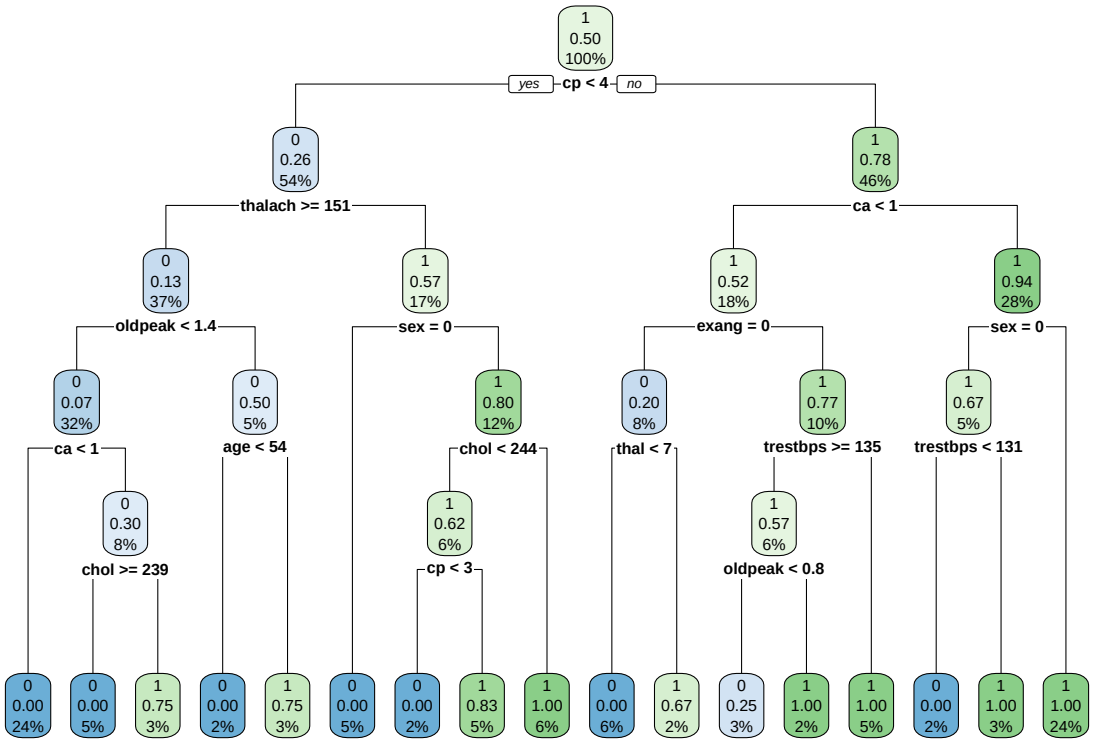
```
mean(confmat$pred==confmat$`valid$disease`)
```

```
## [1] 0.62
```

```
# visualize our best model using rpart.plot()

treeCV = rpart(disease~.,data=train,
               control=rpart.control(maxdepth = minsplit_best, minsplit=maxdepth_best, cp=.01))

rpart.plot(treeCV)
```




```
# visualize the confusion matrix of our actual vs. predicted for our test data

pred = predict(treeCV, test, type = 'class')
pred = data.frame(pred)
confmat = cbind(pred, test$disease)
confmat
```

```
##     pred test$disease
## 4      0            0
## 9      1            1
## 10     1            1
## 14     0            0
## 16     1            0
## 17     0            1
## 18     0            0
## 21     0            0
## 24     1            1
## 32     1            1
## 33     0            1
## 34     1            0
## 37     1            1
## 38     1            1
## 40     1            0
## 41     1            1
## 43     0            0
## 44     1            0
```

```
## 45     0            1
## 46     1            1
## 47     1            0
## 49     0            0
## 50     0            0
## 51     1            0
## 52     1            0
## 54     0            0
## 59     0            0
## 60     0            0
## 61     1            1
## 64     0            0
## 66     1            1
## 68     1            0
## 69     1            1
## 76     0            0
## 80     1            1
## 82     0            0
## 83     0            0
## 86     0            0
## 88     0            0
## 94     0            0
## 96     1            1
## 97     1            1
## 98     1            1
## 99     1            0
## 101    0            0
## 102    0            0
## 106    0            0
## 109    1            1
## 111    1            1
## 113    0            0
## 115    0            1
## 118    0            0
## 120    1            1
## 121    1            1
## 123    1            0
## 124    1            1
## 126    0            0
## 127    1            1
## 128    1            1
## 131    1            0
## 132    1            0
## 135    0            0
## 137    1            1
## 138    1            1
## 139    1            1
## 141    0            0
## 142    0            1
## 143    0            0
## 146    0            1
## 147    1            1
## 149    0            0
## 152    0            0
```

```
## 153    1            0
## 155    1            1
## 156    1            1
## 158    1            1
## 164    0            0
## 165    0            0
## 167    0            0
## 170    0            0
## 172    0            0
## 178    1            1
## 179    0            0
## 180    0            0
## 185    0            1
## 188    1            1
## 189    0            1
## 191    0            0
## 192    1            1
## 197    0            0
## 201    0            0
## 204    0            0
## 207    1            1
## 208    1            1
## 211    0            0
## 214    1            1
## 218    0            0
## 225    0            1
## 227    0            0
## 230    1            1
## 235    1            0
## 237    1            1
## 238    1            1
## 239    0            0
## 240    0            0
## 241    0            0
## 243    0            0
## 246    0            1
## 248    1            1
## 250    0            0
## 252    1            1
## 254    0            0
## 255    0            0
## 256    0            0
## 259    1            0
## 262    0            1
## 265    1            1
## 266    1            1
## 269    1            1
## 270    1            0
## 280    0            0
## 281    1            1
## 283    0            1
## 296    0            0
## 297    1            1
## 302    1            1
```

```r
# compute test accuracy

mean(confmat$pred==confmat$`test$disease`)
```

```
## [1] 0.7777778
```

Question 3. Random Forest

3.1 Read in the iris data set.Subset your data to only include "Sepal Length", "Sepal Width", "Petal Length", and "Petal Width". Estimate the species label for iris flowers using random forest. To do so, first prepare your dataset by spliting to training and testing set(70% for training the model) (2 pts)

```r
# read in "iris" using data()

data(iris)
iris = iris

# split your data in to a features ("Sepal Length", "Sepal Width", "Petal Length",
# and "Petal Width") and target ("Species)

split2 = createDataPartition(y=iris$Species,p = 0.7,list = F,groups = 100)
train2 = iris[split2,]
test2 = iris[-split2,]


# preparing data for training the model
```

3.2 Generate the Random Forest Learning Tree (2 pts)

```r
# generating the random forest learning model. Start with 50 tress

forest = randomForest(Species~.,data=train2,ntree = 50)
```

3.3 Evaluate the model on the test data and check the accuracy (3 pts)

```r
# evalute the model on the test data

predForest = predict(forest,newdata=test2)

# check the accuracy

(1- sum(forest$confusion[, 'class.error']))
```

```
## [1] 0.8285714
```

```
(sum(forest$predicted == train2$Species))/nrow(train2)
```

```
## [1] 0.9428571
```

3.4 Number of trees can be tuned to improve the predictive power of the model.Higher number of trees give you better performance but makes your code slower. Change the number of trees and repeat 3.2 and 3.3. Report any changes. (3 pts)

```
# generating the random forest learning model. Start with 500 tress

forest2 = randomForest(Species~.,data=train2,ntree = 500)


# check the accuracy

(1-sum(forest2$confusion[, 'class.error']))
```

```
## [1] 0.8
```

```
(sum(forest2$predicted == train2$Species))/nrow(train2)
```

```
## [1] 0.9333333
```

Question 4 : Numerically find the minumum solution of the function below. Do a few iterations to improve your initial guess.

$$F(x) = \frac{1}{2}\left(4x_1^2 + 4x_1x_2 + 2x_1x_3 + 5x_2^2 + 6x_2x_3 + 7x_3^2 + x_1x_4 + x_2x_5\right) + 2x_1 - 8x_2 + 9x_3$$

initial guess = 2 for $x_1 - x_5$ and the learning rate =0.01 and epsilon=0.4. (4 pts)

```
# original formula

func = function(x_1, x_2, x_3, x_4, x_5) {(1/2)*(4*(x_1)^2+4*(x_1*x_2)+2*(x_1*x_3)+5*((x_2)^2)+6*(x_2*x


# define the initial value

x=data.frame(c(2,2,2,2,2))

# define the alpha value (learning rate)

learning_rate = .01

# define the epsilon value, maximum iteration allowed

epsilon = .4
```

```
  # gradient descent

Der = Deriv::Deriv(func)

grad =  data.frame(x = 2, value = func(2,2,2,2,2))
grad = 2 - learning_rate * Der(2,2,2,2,2)
grad =  data.frame( x = 2, value = func(2,2,2,2,2))
# record keeping

while (abs(x[1,]) > epsilon)
{x = mean(x[1,], x[2,], x[3,], x[4,], x[5,]) - learning_rate *Der(x[1,], x[2,], x[3,], x[4,], x[5,])
x = data.frame(x)
grad = rbind(grad, func(x[1,], x[2,], x[3,], x[4,], x[5,]))}

grad = grad[,2]


# create the data points' dataframe

x
```

```
##             x
## x_1 0.3709086
## x_2 0.4567467
## x_3 0.3146460
## x_4 0.4220144
## x_5 0.4215938
```

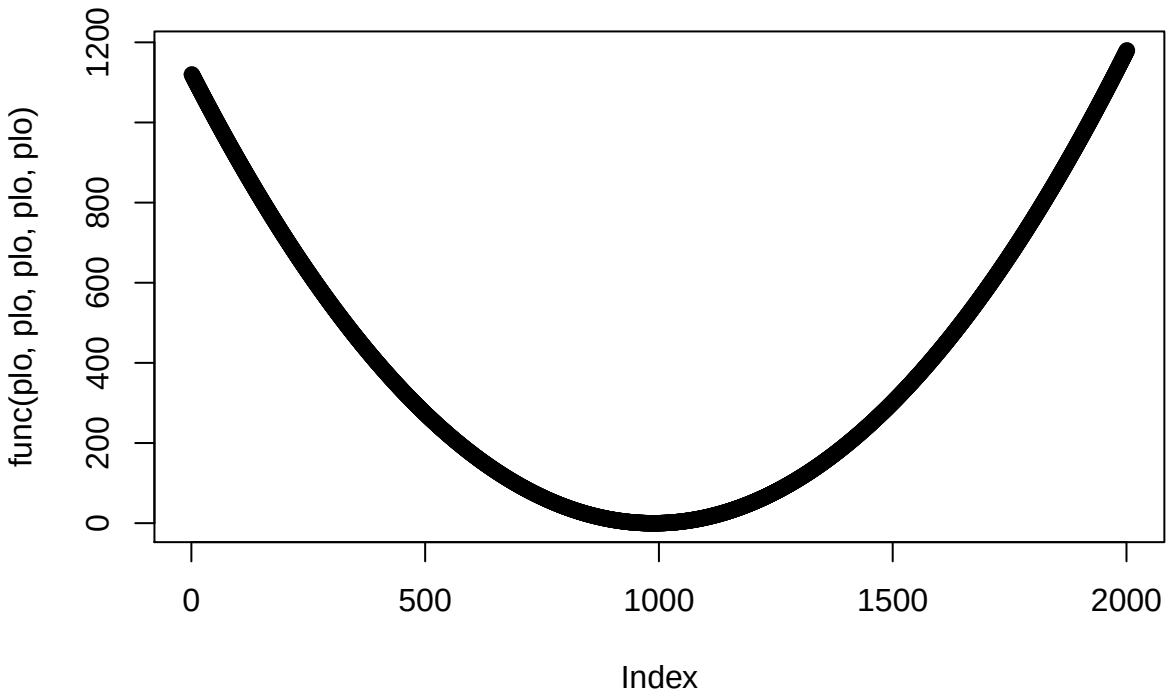
```
dd <- expand.grid(x_1=0:10, x_2=0:10, x_3 = 0:10, x_4 = 0:10, x_5=0:10)
results = do.call(mapply, c(func, unname(dd)))
df = cbind(dd, data.frame(results))
```

4.1 Plot the function value vs. number of iterations, and draw the conclusion. Did the function converged? If not, how can we improve and make it converage? (2 pts)

We can increase the number of iterations or/and decrease the step length to improve the algorithm

```
# draw the function value vs. #iterations

plo = seq(-10,10, by=.01)
plot(func(plo,plo,plo,plo,plo))
```



4.2 Try a few different initial points, repeat the process, and observe the convergence. (2 pts)

```
# try different initial points, repeat, and observe convergence

x=data.frame(c(1,1,1,1,1))

# define the alpha value (learning rate)

learning_rate = .0001

# define the epsilon value, maximum iteration allowed

epsilon = .000001

  # gradient descent

Der = Deriv::Deriv(func)

grad =  data.frame(x = 1, value = func(1,1,1,1,1))
grad = 1 - learning_rate * Der(1,1,1,1,1)
grad =  data.frame( x = 2, value = func(1,1,1,1,1))
# record keeping

while (abs(x[1,]) > epsilon)
```

```
{x = mean(x[1,], x[2,], x[3,], x[4,], x[5,]) - learning_rate *Der(x[1,], x[2,], x[3,], x[4,], x[5,])
x = data.frame(x)
grad = rbind(grad, func(x[1,], x[2,], x[3,], x[4,], x[5,]))}

grad = grad[,2]

grad
```

```
##    [1] 14.5000000000 14.4734046763 14.4487386373 14.4241128006 14.3995261098
##    [6] 14.3749785050 14.3504699259 14.3260003122 14.3015696039 14.2771777410
##   [11] 14.2528246636 14.2285103118 14.2042346260 14.1799975464 14.1557990137
##   [16] 14.1316389682 14.1075173507 14.0834341019 14.0593891625 14.0353824736
##   [21] 14.0114139761 13.9874836110 13.9635913196 13.9397370432 13.9159207230
##   [26] 13.8921423006 13.8684017175 13.8446989152 13.8210338355 13.7974064203
##   [31] 13.7738166113 13.7502643506 13.7267495803 13.7032722424 13.6798322793
##   [36] 13.6564296332 13.6330642466 13.6097360620 13.5864450220 13.5631910692
##   [41] 13.5399741464 13.5167941965 13.4936511625 13.4705449873 13.4474756140
##   [46] 13.4244429859 13.4014470462 13.3784877384 13.3555650058 13.3326787921
##   [51] 13.3098290408 13.2870156957 13.2642387005 13.2414979993 13.2187935358
##   [56] 13.1961252543 13.1734930987 13.1508970135 13.1283369428 13.1058128311
##   [61] 13.0833246229 13.0608722627 13.0384556952 13.0160748651 12.9937297172
##   [66] 12.9714201965 12.9491462479 12.9269078164 12.9047048474 12.8825372859
##   [71] 12.8604050773 12.8383081671 12.8162465006 12.7942200236 12.7722286815
##   [76] 12.7502724202 12.7283511855 12.7064649233 12.6846135795 12.6627971003
##   [81] 12.6410154318 12.6192685201 12.5975563117 12.5758787529 12.5542357902
##   [86] 12.5326273701 12.5110534393 12.4895139445 12.4680088325 12.4465380502
##   [91] 12.4251015445 12.4036992625 12.3823311514 12.3609971582 12.3396972303
##   [96] 12.3184313150 12.2971993599 12.2760013124 12.2548371201 12.2337067307
##  [101] 12.2126100920 12.1915471518 12.1705178581 12.1495221588 12.1285600020
##  [106] 12.1076313358 12.0867361086 12.0658742686 12.0450457643 12.0242505440
##  [111] 12.0034885564 11.9827597501 11.9620640738 11.9414014762 11.9207719063
##  [116] 11.9001753130 11.8796116453 11.8590808523 11.8385828832 11.8181176872
##  [121] 11.7976852137 11.7772854121 11.7569182318 11.7365836224 11.7162815336
##  [126] 11.6960119150 11.6757747165 11.6555698880 11.6353973793 11.6152571405
##  [131] 11.5951491217 11.5750732730 11.5550295448 11.5350178872 11.5150382509
##  [136] 11.4950905861 11.4751748435 11.4552909736 11.4354389272 11.4156186551
##  [141] 11.3958301081 11.3760732372 11.3563479932 11.3366543274 11.3169921909
##  [146] 11.2973615348 11.2777623105 11.2581944693 11.2386579627 11.2191527423
##  [151] 11.1996787596 11.1802359662 11.1608243140 11.1414437547 11.1220942402
##  [156] 11.1027757225 11.0834881536 11.0642314857 11.0450056708 11.0258106613
##  [161] 11.0066464095 10.9875128678 10.9684099886 10.9493377246 10.9302960282
##  [166] 10.9112848523 10.8923041495 10.8733538726 10.8544339747 10.8355444087
##  [171] 10.8166851276 10.7978560845 10.7790572326 10.7602885252 10.7415499157
##  [176] 10.7228413574 10.7041628037 10.6855142083 10.6668955248 10.6483067068
##  [181] 10.6297477081 10.6112184825 10.5927189840 10.5742491665 10.5558089840
##  [186] 10.5373983906 10.5190173406 10.5006657881 10.4823436875 10.4640509932
##  [191] 10.4457876596 10.4275536413 10.4093488928 10.3911733688 10.3730270241
##  [196] 10.3549098134 10.3368216917 10.3187626138 10.3007325348 10.2827314098
##  [201] 10.2647591938 10.2468158422 10.2289013101 10.2110155530 10.1931585263
##  [206] 10.1753301854 10.1575304859 10.1397593834 10.1220168336 10.1043027923
##  [211] 10.0866172153 10.0689600585 10.0513312778 10.0337308293 10.0161586691
##  [216]  9.9986147533  9.9810990382  9.9636114800  9.9461520351  9.9287206599
##  [221]  9.9113173109  9.8939419448  9.8765945180  9.8592749874  9.8419833096
```

```
##  [226] 9.8247194415 9.8074833400 9.7902749620 9.7730942645 9.7559412047
##  [231] 9.7388157397 9.7217178267 9.7046474229 9.6876044858 9.6705889728
##  [236] 9.6536008412 9.6366400488 9.6197065530 9.6028003115 9.5859212821
##  [241] 9.5690694226 9.5522446908 9.5354470447 9.5186764423 9.5019328416
##  [246] 9.4852162008 9.4685264779 9.4518636313 9.4352276194 9.4186184004
##  [251] 9.4020359328 9.3854801750 9.3689510858 9.3524486237 9.3359727473
##  [256] 9.3195234155 9.3031005871 9.2867042209 9.2703342759 9.2539907111
##  [261] 9.2376734855 9.2213825584 9.2051178888 9.1888794361 9.1726671595
##  [266] 9.1564810185 9.1403209724 9.1241869808 9.1080790033 9.0919969995
##  [271] 9.0759409290 9.0599107516 9.0439064272 9.0279279155 9.0119751766
##  [276] 8.9960481704 8.9801468569 8.9642711964 8.9484211490 8.9325966748
##  [281] 8.9167977343 8.9010242877 8.8852762956 8.8695537183 8.8538565164
##  [286] 8.8381846505 8.8225380813 8.8069167695 8.7913206758 8.7757497612
##  [291] 8.7602039865 8.7446833126 8.7291877006 8.7137171117 8.6982715068
##  [296] 8.6828508472 8.6674550942 8.6520842091 8.6367381532 8.6214168880
##  [301] 8.6061203750 8.5908485757 8.5756014518 8.5603789649 8.5451810767
##  [306] 8.5300077490 8.5148589437 8.4997346227 8.4846347478 8.4695592813
##  [311] 8.4545081850 8.4394814212 8.4244789521 8.4095007398 8.3945467468
##  [316] 8.3796169353 8.3647112678 8.3498297067 8.3349722147 8.3201387543
##  [321] 8.3053292881 8.2905437788 8.2757821893 8.2610444823 8.2463306208
##  [326] 8.2316405676 8.2169742857 8.2023317382 8.1877128883 8.1731176990
##  [331] 8.1585461335 8.1439981552 8.1294737274 8.1149728134 8.1004953768
##  [336] 8.0860413809 8.0716107894 8.0572035658 8.0428196739 8.0284590772
##  [341] 8.0141217397 7.9998076252 7.9855166975 7.9712489205 7.9570042583
##  [346] 7.9427826749 7.9285841344 7.9144086010 7.9002560389 7.8861264124
##  [351] 7.8720196857 7.8579358233 7.8438747895 7.8298365490 7.8158210662
##  [356] 7.8018283057 7.7878582321 7.7739108102 7.7599860047 7.7460837805
##  [361] 7.7322041023 7.7183469352 7.7045122441 7.6906999939 7.6769101499
##  [366] 7.6631426771 7.6493975406 7.6356747058 7.6219741380 7.6082958024
##  [371] 7.5946396645 7.5810056898 7.5673938436 7.5538040916 7.5402363995
##  [376] 7.5266907327 7.5131670572 7.4996653385 7.4861855427 7.4727276354
##  [381] 7.4592915827 7.4458773504 7.4324849047 7.4191142117 7.4057652373
##  [386] 7.3924379479 7.3791323097 7.3658482889 7.3525858519 7.3393449650
##  [391] 7.3261255947 7.3129277076 7.2997512700 7.2865962487 7.2734626103
##  [396] 7.2603503215 7.2472593489 7.2341896594 7.2211412199 7.2081139973
##  [401] 7.1951079584 7.1821230704 7.1691593002 7.1562166149 7.1432949817
##  [406] 7.1303943678 7.1175147405 7.1046560670 7.0918183147 7.0790014510
##  [411] 7.0662054433 7.0534302591 7.0406758661 7.0279422317 7.0152293237
##  [416] 7.0025371096 6.9898655574 6.9772146347 6.9645843095 6.9519745496
##  [421] 6.9393853229 6.9268165974 6.9142683413 6.9017405225 6.8892331093
##  [426] 6.8767460697 6.8642793721 6.8518329847 6.8394068759 6.8270010140
##  [431] 6.8146153675 6.8022499048 6.7899045945 6.7775794051 6.7652743053
##  [436] 6.7529892637 6.7407242491 6.7284792303 6.7162541759 6.7040490550
##  [441] 6.6918638363 6.6796984890 6.6675529819 6.6554272841 6.6433213647
##  [446] 6.6312351929 6.6191687378 6.6071219687 6.5950948550 6.5830873658
##  [451] 6.5710994706 6.5591311389 6.5471823400 6.5352530436 6.5233432192
##  [456] 6.5114528363 6.4995818647 6.4877302741 6.4758980342 6.4640851149
##  [461] 6.4522914859 6.4405171171 6.4287619786 6.4170260402 6.4053092721
##  [466] 6.3936116442 6.3819331268 6.3702736899 6.3586333038 6.3470119388
##  [471] 6.3354095651 6.3238261531 6.3122616732 6.3007160958 6.2891893914
##  [476] 6.2776815306 6.2661924839 6.2547222220 6.2432707155 6.2318379351
##  [481] 6.2204238515 6.2090284356 6.1976516583 6.1862934903 6.1749539027
##  [486] 6.1636328664 6.1523303525 6.1410463319 6.1297807758 6.1185336555
##  [491] 6.1073049420 6.0960946066 6.0849026206 6.0737289553 6.0625735821
```

```
##  [496] 6.0514364725 6.0403175978 6.0292169296 6.0181344395 6.0070700990
##  [501] 5.9960238797 5.9849957535 5.9739856918 5.9629936667 5.9520196497
##  [506] 5.9410636129 5.9301255281 5.9192053672 5.9083031022 5.8974187052
##  [511] 5.8865521481 5.8757034032 5.8648724426 5.8540592384 5.8432637629
##  [516] 5.8324859885 5.8217258873 5.8109834318 5.8002585944 5.7895513475
##  [521] 5.7788616636 5.7681895153 5.7575348751 5.7468977157 5.7362780097
##  [526] 5.7256757298 5.7150908488 5.7045233394 5.6939731744 5.6834403268
##  [531] 5.6729247695 5.6624264753 5.6519454174 5.6414815686 5.6310349022
##  [536] 5.6206053912 5.6101930088 5.5997977281 5.5894195224 5.5790583651
##  [541] 5.5687142293 5.5583870886 5.5480769162 5.5377836857 5.5275073704
##  [546] 5.5172479440 5.5070053800 5.4967796521 5.4865707337 5.4763785988
##  [551] 5.4662032208 5.4560445738 5.4459026313 5.4357773674 5.4256687559
##  [556] 5.4155767706 5.4055013857 5.3954425750 5.3854003127 5.3753745728
##  [561] 5.3653653294 5.3553725567 5.3453962290 5.3354363204 5.3254928053
##  [566] 5.3155656580 5.3056548528 5.2957603641 5.2858821665 5.2760202343
##  [571] 5.2661745421 5.2563450644 5.2465317759 5.2367346511 5.2269536648
##  [576] 5.2171887917 5.2074400065 5.1977072841 5.1879905992 5.1782899267
##  [581] 5.1686052415 5.1589365186 5.1492837330 5.1396468597 5.1300258737
##  [586] 5.1204207501 5.1108314641 5.1012579909 5.0917003057 5.0821583836
##  [591] 5.0726322001 5.0631217304 5.0536269500 5.0441478341 5.0346843583
##  [596] 5.0252364980 5.0158042288 5.0063875262 4.9969863657 4.9876007231
##  [601] 4.9782305740 4.9688758940 4.9595366590 4.9502128447 4.9409044268
##  [606] 4.9316113814 4.9223336842 4.9130713112 4.9038242384 4.8945924417
##  [611] 4.8853758972 4.8761745809 4.8669884691 4.8578175377 4.8486617630
##  [616] 4.8395211212 4.8303955886 4.8212851415 4.8121897561 4.8031094089
##  [621] 4.7940440763 4.7849937346 4.7759583605 4.7669379302 4.7579324206
##  [626] 4.7489418080 4.7399660691 4.7310051806 4.7220591192 4.7131278615
##  [631] 4.7042113844 4.6953096646 4.6864226791 4.6775504045 4.6686928179
##  [636] 4.6598498961 4.6510216163 4.6422079552 4.6334088901 4.6246243980
##  [641] 4.6158544560 4.6070990412 4.5983581308 4.5896317021 4.5809197323
##  [646] 4.5722221987 4.5635390786 4.5548703494 4.5462159884 4.5375759731
##  [651] 4.5289502810 4.5203388896 4.5117417764 4.5031589189 4.4945902947
##  [656] 4.4860358816 4.4774956571 4.4689695989 4.4604576849 4.4519598927
##  [661] 4.4434762002 4.4350065853 4.4265510257 4.4181094994 4.4096819844
##  [666] 4.4012684586 4.3928689000 4.3844832866 4.3761115967 4.3677538081
##  [671] 4.3594098992 4.3510798481 4.3427636329 4.3344612320 4.3261726236
##  [676] 4.3178977860 4.3096366976 4.3013893367 4.2931556817 4.2849357112
##  [681] 4.2767294036 4.2685367373 4.2603576910 4.2521922431 4.2440403724
##  [686] 4.2359020574 4.2277772769 4.2196660095 4.2115682340 4.2034839291
##  [691] 4.1954130737 4.1873556466 4.1793116266 4.1712809927 4.1632637238
##  [696] 4.1552597989 4.1472691969 4.1392918970 4.1313278780 4.1233771193
##  [701] 4.1154395998 4.1075152988 4.0996041954 4.0917062689 4.0838214985
##  [706] 4.0759498635 4.0680913432 4.0602459171 4.0524135643 4.0445942645
##  [711] 4.0367879969 4.0289947412 4.0212144767 4.0134471832 4.0056928400
##  [716] 3.9979514269 3.9902229234 3.9825073092 3.9748045641 3.9671146677
##  [721] 3.9594375999 3.9517733403 3.9441218689 3.9364831654 3.9288572098
##  [726] 3.9212439820 3.9136434619 3.9060556295 3.8984804647 3.8909179478
##  [731] 3.8833680586 3.8758307774 3.8683060842 3.8607939592 3.8532943826
##  [736] 3.8458073346 3.8383327954 3.8308707454 3.8234211649 3.8159840341
##  [741] 3.8085593336 3.8011470436 3.7937471446 3.7863596170 3.7789844415
##  [746] 3.7716215984 3.7642710683 3.7569328319 3.7496068697 3.7422931624
##  [751] 3.7349916906 3.7277024351 3.7204253766 3.7131604957 3.7059077735
##  [756] 3.6986671905 3.6914387278 3.6842223661 3.6770180865 3.6698258697
##  [761] 3.6626456969 3.6554775489 3.6483214069 3.6411772518 3.6340450647
```

```
##  [766] 3.6269248269 3.6198165193 3.6127201232 3.6056356198 3.5985629902
##  [771] 3.5915022159 3.5844532780 3.5774161578 3.5703908368 3.5633772962
##  [776] 3.5563755175 3.5493854821 3.5424071715 3.5354405671 3.5284856505
##  [781] 3.5215424032 3.5146108068 3.5076908429 3.5007824931 3.4938857390
##  [786] 3.4870005624 3.4801269450 3.4732648685 3.4664143147 3.4595752653
##  [791] 3.4527477022 3.4459316073 3.4391269624 3.4323337495 3.4255519504
##  [796] 3.4187815472 3.4120225218 3.4052748562 3.3985385326 3.3918135330
##  [801] 3.3850998394 3.3783974340 3.3717062990 3.3650264166 3.3583577689
##  [806] 3.3517003383 3.3450541069 3.3384190571 3.3317951712 3.3251824315
##  [811] 3.3185808204 3.3119903203 3.3054109137 3.2988425830 3.2922853107
##  [816] 3.2857390793 3.2792038714 3.2726796694 3.2661664561 3.2596642139
##  [821] 3.2531729257 3.2466925739 3.2402231414 3.2337646109 3.2273169650
##  [826] 3.2208801867 3.2144542586 3.2080391637 3.2016348847 3.1952414047
##  [831] 3.1888587063 3.1824867728 3.1761255868 3.1697751316 3.1634353900
##  [836] 3.1571063452 3.1507879802 3.1444802780 3.1381832219 3.1318967949
##  [841] 3.1256209802 3.1193557610 3.1131011206 3.1068570422 3.1006235091
##  [846] 3.0944005045 3.0881880118 3.0819860144 3.0757944956 3.0696134388
##  [851] 3.0634428275 3.0572826451 3.0511328751 3.0449935010 3.0388645064
##  [856] 3.0327458747 3.0266375896 3.0205396346 3.0144519935 3.0083746498
##  [861] 3.0023075873 2.9962507896 2.9902042406 2.9841679238 2.9781418232
##  [866] 2.9721259226 2.9661202057 2.9601246564 2.9541392586 2.9481639962
##  [871] 2.9421988532 2.9362438135 2.9302988611 2.9243639800 2.9184391542
##  [876] 2.9125243677 2.9066196048 2.9007248493 2.8948400856 2.8889652977
##  [881] 2.8831004699 2.8772455862 2.8714006310 2.8655655885 2.8597404430
##  [886] 2.8539251788 2.8481197801 2.8423242314 2.8365385171 2.8307626214
##  [891] 2.8249965288 2.8192402239 2.8134936910 2.8077569146 2.8020298793
##  [896] 2.7963125695 2.7906049699 2.7849070651 2.7792188396 2.7735402781
##  [901] 2.7678713652 2.7622120857 2.7565624242 2.7509223654 2.7452918942
##  [906] 2.7396709953 2.7340596534 2.7284578535 2.7228655803 2.7172828187
##  [911] 2.7117095537 2.7061457701 2.7005914529 2.6950465870 2.6895111575
##  [916] 2.6839851492 2.6784685474 2.6729613369 2.6674635030 2.6619750307
##  [921] 2.6564959051 2.6510261114 2.6455656347 2.6401144602 2.6346725732
##  [926] 2.6292399589 2.6238166026 2.6184024894 2.6129976049 2.6076019342
##  [931] 2.6022154627 2.5968381758 2.5914700589 2.5861110975 2.5807612768
##  [936] 2.5754205826 2.5700890001 2.5647665149 2.5594531126 2.5541487787
##  [941] 2.5488534988 2.5435672585 2.5382900433 2.5330218390 2.5277626313
##  [946] 2.5225124057 2.5172711480 2.5120388440 2.5068154793 2.5016010398
##  [951] 2.4963955113 2.4911988796 2.4860111305 2.4808322499 2.4756622236
##  [956] 2.4705010376 2.4653486779 2.4602051303 2.4550703808 2.4499444154
##  [961] 2.4448272202 2.4397187812 2.4346190844 2.4295281159 2.4244458618
##  [966] 2.4193723082 2.4143074413 2.4092512473 2.4042037123 2.3991648226
##  [971] 2.3941345643 2.3891129237 2.3840998871 2.3790954408 2.3740995711
##  [976] 2.3691122644 2.3641335069 2.3591632851 2.3542015854 2.3492483941
##  [981] 2.3443036978 2.3393674828 2.3344397358 2.3295204431 2.3246095913
##  [986] 2.3197071669 2.3148131565 2.3099275468 2.3050503242 2.3001814755
##  [991] 2.2953209872 2.2904688461 2.2856250388 2.2807895521 2.2759623726
##  [996] 2.2711434872 2.2663328825 2.2615305455 2.2567364628 2.2519506214
## [1001] 2.2471730081 2.2424036097 2.2376424132 2.2328894055 2.2281445734
## [1006] 2.2234079040 2.2186793843 2.2139590011 2.2092467417 2.2045425928
## [1011] 2.1998465418 2.1951585755 2.1904786811 2.1858068457 2.1811430565
## [1016] 2.1764873006 2.1718395652 2.1671998374 2.1625681045 2.1579443538
## [1021] 2.1533285724 2.1487207477 2.1441208669 2.1395289173 2.1349448863
## [1026] 2.1303687613 2.1258005296 2.1212401785 2.1166876956 2.1121430682
## [1031] 2.1076062838 2.1030773299 2.0985561938 2.0940428633 2.0895373257
```

```
## [1036]  2.0850395686  2.0805495796  2.0760673463  2.0715928562  2.0671260971
## [1041]  2.0626670565  2.0582157221  2.0537720815  2.0493361225  2.0449078329
## [1046]  2.0404872002  2.0360742124  2.0316688571  2.0272711221  2.0228809954
## [1051]  2.0184984646  2.0141235177  2.0097561425  2.0053963269  2.0010440588
## [1056]  1.9966993261  1.9923621169  1.9880324189  1.9837102203  1.9793955091
## [1061]  1.9750882731  1.9707885006  1.9664961794  1.9622112978  1.9579338438
## [1066]  1.9536638055  1.9494011711  1.9451459286  1.9408980663  1.9366575723
## [1071]  1.9324244349  1.9281986422  1.9239801826  1.9197690442  1.9155652153
## [1076]  1.9113686842  1.9071794393  1.9029974688  1.8988227612  1.8946553047
## [1081]  1.8904950878  1.8863420989  1.8821963264  1.8780577586  1.8739263842
## [1086]  1.8698021915  1.8656851691  1.8615753053  1.8574725889  1.8533770083
## [1091]  1.8492885521  1.8452072088  1.8411329671  1.8370658156  1.8330057429
## [1096]  1.8289527377  1.8249067885  1.8208678842  1.8168360134  1.8128111649
## [1101]  1.8087933273  1.8047824894  1.8007786401  1.7967817680  1.7927918620
## [1106]  1.7888089110  1.7848329037  1.7808638291  1.7769016760  1.7729464332
## [1111]  1.7689980898  1.7650566346  1.7611220566  1.7571943447  1.7532734880
## [1116]  1.7493594754  1.7454522959  1.7415519386  1.7376583925  1.7337716467
## [1121]  1.7298916902  1.7260185122  1.7221521018  1.7182924480  1.7144395402
## [1126]  1.7105933673  1.7067539187  1.7029211835  1.6990951509  1.6952758101
## [1131]  1.6914631504  1.6876571612  1.6838578315  1.6800651509  1.6762791084
## [1136]  1.6724996936  1.6687268958  1.6649607042  1.6612011084  1.6574480977
## [1141]  1.6537016614  1.6499617891  1.6462284702  1.6425016942  1.6387814505
## [1146]  1.6350677286  1.6313605181  1.6276598084  1.6239655892  1.6202778500
## [1151]  1.6165965803  1.6129217698  1.6092534080  1.6055914846  1.6019359893
## [1156]  1.5982869117  1.5946442414  1.5910079682  1.5873780818  1.5837545718
## [1161]  1.5801374281  1.5765266403  1.5729221983  1.5693240919  1.5657323107
## [1166]  1.5621468448  1.5585676838  1.5549948177  1.5514282362  1.5478679294
## [1171]  1.5443138870  1.5407660991  1.5372245554  1.5336892461  1.5301601609
## [1176]  1.5266372900  1.5231206232  1.5196101506  1.5161058623  1.5126077482
## [1181]  1.5091157984  1.5056300030  1.5021503520  1.4986768356  1.4952094439
## [1186]  1.4917481670  1.4882929950  1.4848439181  1.4814009265  1.4779640103
## [1191]  1.4745331598  1.4711083653  1.4676896168  1.4642769047  1.4608702193
## [1196]  1.4574695508  1.4540748895  1.4506862258  1.4473035499  1.4439268522
## [1201]  1.4405561231  1.4371913529  1.4338325321  1.4304796509  1.4271326999
## [1206]  1.4237916694  1.4204565500  1.4171273320  1.4138040060  1.4104865624
## [1211]  1.4071749918  1.4038692846  1.4005694314  1.3972754228  1.3939872493
## [1216]  1.3907049015  1.3874283699  1.3841576453  1.3808927181  1.3776335791
## [1221]  1.3743802189  1.3711326282  1.3678907976  1.3646547179  1.3614243797
## [1226]  1.3581997738  1.3549808909  1.3517677217  1.3485602571  1.3453584879
## [1231]  1.3421624047  1.3389719984  1.3357872599  1.3326081800  1.3294347495
## [1236]  1.3262669594  1.3231048004  1.3199482634  1.3167973395  1.3136520195
## [1241]  1.3105122943  1.3073781550  1.3042495924  1.3011265976  1.2980091615
## [1246]  1.2948972751  1.2917909295  1.2886901158  1.2855948249  1.2825050479
## [1251]  1.2794207759  1.2763420000  1.2732687113  1.2702009009  1.2671385600
## [1256]  1.2640816797  1.2610302511  1.2579842654  1.2549437139  1.2519085877
## [1261]  1.2488788780  1.2458545761  1.2428356731  1.2398221605  1.2368140294
## [1266]  1.2338112711  1.2308138769  1.2278218381  1.2248351461  1.2218537922
## [1271]  1.2188777677  1.2159070640  1.2129416725  1.2099815845  1.2070267916
## [1276]  1.2040772850  1.2011330562  1.1981940967  1.1952603980  1.1923319514
## [1281]  1.1894087485  1.1864907808  1.1835780398  1.1806705170  1.1777682040
## [1286]  1.1748710923  1.1719791734  1.1690924390  1.1662108806  1.1633344899
## [1291]  1.1604632584  1.1575971778  1.1547362397  1.1518804358  1.1490297577
## [1296]  1.1461841972  1.1433437459  1.1405083955  1.1376781378  1.1348529645
## [1301]  1.1320328673  1.1292178380  1.1264078683  1.1236029501  1.1208030752
```

```
## [1306]  1.1180082352  1.1152184222  1.1124336279  1.1096538441  1.1068790628
## [1311]  1.1041092757  1.1013444748  1.0985846520  1.0958297992  1.0930799083
## [1316]  1.0903349712  1.0875949800  1.0848599265  1.0821298027  1.0794046006
## [1321]  1.0766843123  1.0739689297  1.0712584449  1.0685528498  1.0658521366
## [1326]  1.0631562973  1.0604653239  1.0577792086  1.0550979435  1.0524215206
## [1331]  1.0497499322  1.0470831702  1.0444212270  1.0417640945  1.0391117651
## [1336]  1.0364642309  1.0338214840  1.0311835168  1.0285503213  1.0259218899
## [1341]  1.0232982148  1.0206792883  1.0180651025  1.0154556499  1.0128509226
## [1346]  1.0102509131  1.0076556135  1.0050650163  1.0024791138  0.9998978983
## [1351]  0.9973213623  0.9947494980  0.9921822978  0.9896197543  0.9870618597
## [1356]  0.9845086065  0.9819599872  0.9794159941  0.9768766198  0.9743418567
## [1361]  0.9718116973  0.9692861341  0.9667651596  0.9642487664  0.9617369468
## [1366]  0.9592296936  0.9567269992  0.9542288562  0.9517352572  0.9492461948
## [1371]  0.9467616616  0.9442816501  0.9418061531  0.9393351630  0.9368686727
## [1376]  0.9344066747  0.9319491617  0.9294961265  0.9270475616  0.9246034597
## [1381]  0.9221638137  0.9197286162  0.9172978600  0.9148715378  0.9124496423
## [1386]  0.9100321664  0.9076191028  0.9052104443  0.9028061838  0.9004063140
## [1391]  0.8980108277  0.8956197178  0.8932329772  0.8908505987  0.8884725752
## [1396]  0.8860988996  0.8837295647  0.8813645635  0.8790038888  0.8766475337
## [1401]  0.8742954910  0.8719477537  0.8696043148  0.8672651672  0.8649303039
## [1406]  0.8625997179  0.8602734022  0.8579513498  0.8556335538  0.8533200072
## [1411]  0.8510107030  0.8487056343  0.8464047941  0.8441081756  0.8418157718
## [1416]  0.8395275759  0.8372435809  0.8349637800  0.8326881662  0.8304167329
## [1421]  0.8281494730  0.8258863798  0.8236274465  0.8213726661  0.8191220321
## [1426]  0.8168755374  0.8146331754  0.8123949394  0.8101608224  0.8079308178
## [1431]  0.8057049189  0.8034831189  0.8012654111  0.7990517888  0.7968422453
## [1436]  0.7946367739  0.7924353680  0.7902380208  0.7880447257  0.7858554761
## [1441]  0.7836702654  0.7814890868  0.7793119339  0.7771388000  0.7749696785
## [1446]  0.7728045628  0.7706434464  0.7684863227  0.7663331851  0.7641840271
## [1451]  0.7620388422  0.7598976238  0.7577603655  0.7556270607  0.7534977030
## [1456]  0.7513722859  0.7492508028  0.7471332475  0.7450196133  0.7429098938
## [1461]  0.7408040827  0.7387021735  0.7366041598  0.7345100352  0.7324197933
## [1466]  0.7303334277  0.7282509321  0.7261723001  0.7240975254  0.7220266016
## [1471]  0.7199595223  0.7178962814  0.7158368723  0.7137812890  0.7117295250
## [1476]  0.7096815741  0.7076374300  0.7055970865  0.7035605373  0.7015277762
## [1481]  0.6994987969  0.6974735932  0.6954521589  0.6934344878  0.6914205737
## [1486]  0.6894104105  0.6874039919  0.6854013118  0.6834023641  0.6814071425
## [1491]  0.6794156410  0.6774278535  0.6754437738  0.6734633957  0.6714867134
## [1496]  0.6695137205  0.6675444111  0.6655787791  0.6636168185  0.6616585231
## [1501]  0.6597038870  0.6577529041  0.6558055684  0.6538618739  0.6519218146
## [1506]  0.6499853845  0.6480525776  0.6461233879  0.6441978095  0.6422758365
## [1511]  0.6403574628  0.6384426826  0.6365314899  0.6346238788  0.6327198434
## [1516]  0.6308193779  0.6289224762  0.6270291325  0.6251393410  0.6232530958
## [1521]  0.6213703911  0.6194912210  0.6176155796  0.6157434612  0.6138748599
## [1526]  0.6120097699  0.6101481855  0.6082901008  0.6064355100  0.6045844075
## [1531]  0.6027367874  0.6008926439  0.5990519714  0.5972147641  0.5953810163
## [1536]  0.5935507222  0.5917238762  0.5899004726  0.5880805056  0.5862639696
## [1541]  0.5844508589  0.5826411678  0.5808348908  0.5790320221  0.5772325562
## [1546]  0.5754364873  0.5736438099  0.5718545184  0.5700686071  0.5682860705
## [1551]  0.5665069030  0.5647310991  0.5629586530  0.5611895594  0.5594238126
## [1556]  0.5576614071  0.5559023374  0.5541465980  0.5523941833  0.5506450878
## [1561]  0.5488993061  0.5471568327  0.5454176620  0.5436817886  0.5419492071
## [1566]  0.5402199120  0.5384938978  0.5367711591  0.5350516905  0.5333354866
## [1571]  0.5316225420  0.5299128512  0.5282064088  0.5265032096  0.5248032480
```

```
## [1576]  0.5231065188  0.5214130166  0.5197227359  0.5180356716  0.5163518182
## [1581]  0.5146711705  0.5129937230  0.5113194706  0.5096484078  0.5079805294
## [1586]  0.5063158302  0.5046543048  0.5029959480  0.5013407545  0.4996887191
## [1591]  0.4980398365  0.4963941015  0.4947515088  0.4931120533  0.4914757297
## [1596]  0.4898425329  0.4882124576  0.4865854987  0.4849616510  0.4833409092
## [1601]  0.4817232683  0.4801087231  0.4784972685  0.4768888993  0.4752836104
## [1606]  0.4736813966  0.4720822529  0.4704861742  0.4688931553  0.4673031912
## [1611]  0.4657162767  0.4641324069  0.4625515766  0.4609737809  0.4593990145
## [1616]  0.4578272726  0.4562585501  0.4546928419  0.4531301431  0.4515704485
## [1621]  0.4500137533  0.4484600525  0.4469093409  0.4453616138  0.4438168660
## [1626]  0.4422750927  0.4407362889  0.4392004496  0.4376675699  0.4361376449
## [1631]  0.4346106697  0.4330866392  0.4315655487  0.4300473933  0.4285321679
## [1636]  0.4270198678  0.4255104881  0.4240040239  0.4225004704  0.4209998226
## [1641]  0.4195020758  0.4180072251  0.4165152657  0.4150261927  0.4135400014
## [1646]  0.4120566869  0.4105762444  0.4090986692  0.4076239564  0.4061521014
## [1651]  0.4046830992  0.4032169452  0.4017536345  0.4002931626  0.3988355245
## [1656]  0.3973807156  0.3959287311  0.3944795664  0.3930332167  0.3915896773
## [1661]  0.3901489436  0.3887110107  0.3872758742  0.3858435292  0.3844139711
## [1666]  0.3829871953  0.3815631971  0.3801419718  0.3787235149  0.3773078217
## [1671]  0.3758948876  0.3744847079  0.3730772781  0.3716725935  0.3702706496
## [1676]  0.3688714418  0.3674749655  0.3660812161  0.3646901891  0.3633018799
## [1681]  0.3619162839  0.3605333967  0.3591532136  0.3577757302  0.3564009419
## [1686]  0.3550288443  0.3536594328  0.3522927028  0.3509286501  0.3495672699
## [1691]  0.3482085579  0.3468525096  0.3454991205  0.3441483862  0.3428003022
## [1696]  0.3414548641  0.3401120674  0.3387719078  0.3374343807  0.3360994818
## [1701]  0.3347672066  0.3334375509  0.3321105101  0.3307860799  0.3294642559
## [1706]  0.3281450337  0.3268284090  0.3255143774  0.3242029345  0.3228940761
## [1711]  0.3215877977  0.3202840951  0.3189829639  0.3176843998  0.3163883985
## [1716]  0.3150949556  0.3138040669  0.3125157281  0.3112299350  0.3099466831
## [1721]  0.3086659683  0.3073877863  0.3061121329  0.3048390037  0.3035683946
## [1726]  0.3023003013  0.3010347195  0.2997716451  0.2985110739  0.2972530015
## [1731]  0.2959974239  0.2947443369  0.2934937361  0.2922456175  0.2909999769
## [1736]  0.2897568101  0.2885161129  0.2872778812  0.2860421108  0.2848087976
## [1741]  0.2835779375  0.2823495263  0.2811235599  0.2799000341  0.2786789449
## [1746]  0.2774602882  0.2762440599  0.2750302558  0.2738188719  0.2726099041
## [1751]  0.2714033483  0.2701992005  0.2689974566  0.2677981125  0.2666011643
## [1756]  0.2654066078  0.2642144391  0.2630246540  0.2618372487  0.2606522190
## [1761]  0.2594695610  0.2582892706  0.2571113439  0.2559357768  0.2547625655
## [1766]  0.2535917058  0.2524231939  0.2512570258  0.2500931974  0.2489317050
## [1771]  0.2477725445  0.2466157119  0.2454612034  0.2443090150  0.2431591428
## [1776]  0.2420115829  0.2408663314  0.2397233844  0.2385827379  0.2374443881
## [1781]  0.2363083311  0.2351745631  0.2340430801  0.2329138783  0.2317869538
## [1786]  0.2306623028  0.2295399215  0.2284198059  0.2273019523  0.2261863569
## [1791]  0.2250730157  0.2239619250  0.2228530810  0.2217464799  0.2206421179
## [1796]  0.2195399911  0.2184400958  0.2173424283  0.2162469847  0.2151537613
## [1801]  0.2140627543  0.2129739600  0.2118873745  0.2108029943  0.2097208154
## [1806]  0.2086408343  0.2075630471  0.2064874501  0.2054140397  0.2043428121
## [1811]  0.2032737636  0.2022068906  0.2011421893  0.2000796560  0.1990192871
## [1816]  0.1979610788  0.1969050276  0.1958511298  0.1947993817  0.1937497796
## [1821]  0.1927023199  0.1916569990  0.1906138133  0.1895727590  0.1885338326
## [1826]  0.1874970306  0.1864623491  0.1854297848  0.1843993339  0.1833709928
## [1831]  0.1823447581  0.1813206260  0.1802985930  0.1792786557  0.1782608103
## [1836]  0.1772450533  0.1762313812  0.1752197904  0.1742102774  0.1732028387
## [1841]  0.1721974707  0.1711941699  0.1701929327  0.1691937557  0.1681966354
```

```
## [1846]  0.1672015682  0.1662085507  0.1652175793  0.1642286506  0.1632417611
## [1851]  0.1622569074  0.1612740858  0.1602932931  0.1593145257  0.1583377802
## [1856]  0.1573630530  0.1563903409  0.1554196403  0.1544509479  0.1534842601
## [1861]  0.1525195736  0.1515568850  0.1505961908  0.1496374877  0.1486807722
## [1866]  0.1477260410  0.1467732906  0.1458225177  0.1448737189  0.1439268909
## [1871]  0.1429820302  0.1420391335  0.1410981975  0.1401592188  0.1392221941
## [1876]  0.1382871199  0.1373539931  0.1364228102  0.1354935680  0.1345662630
## [1881]  0.1336408921  0.1327174518  0.1317959390  0.1308763502  0.1299586823
## [1886]  0.1290429318  0.1281290956  0.1272171704  0.1263071528  0.1253990397
## [1891]  0.1244928277  0.1235885137  0.1226860943  0.1217855663  0.1208869264
## [1896]  0.1199901716  0.1190952984  0.1182023037  0.1173111843  0.1164219369
## [1901]  0.1155345583  0.1146490454  0.1137653950  0.1128836037  0.1120036686
## [1906]  0.1111255863  0.1102493537  0.1093749676  0.1085024249  0.1076317223
## [1911]  0.1067628568  0.1058958252  0.1050306243  0.1041672510  0.1033057021
## [1916]  0.1024459746  0.1015880652  0.1007319709  0.0998776886  0.0990252151
## [1921]  0.0981745473  0.0973256821  0.0964786165  0.0956333473  0.0947898714
## [1926]  0.0939481858  0.0931082873  0.0922701730  0.0914338397  0.0905992844
## [1931]  0.0897665040  0.0889354955  0.0881062557  0.0872787818  0.0864530705
## [1936]  0.0856291190  0.0848069241  0.0839864829  0.0831677923  0.0823508492
## [1941]  0.0815356508  0.0807221940  0.0799104757  0.0791004931  0.0782922430
## [1946]  0.0774857225  0.0766809287  0.0758778586  0.0750765091  0.0742768773
## [1951]  0.0734789603  0.0726827550  0.0718882586  0.0710954681  0.0703043806
## [1956]  0.0695149931  0.0687273026  0.0679413063  0.0671570011  0.0663743843
## [1961]  0.0655934529  0.0648142040  0.0640366346  0.0632607418  0.0624865229
## [1966]  0.0617139748  0.0609430947  0.0601738797  0.0594063269  0.0586404335
## [1971]  0.0578761965  0.0571136132  0.0563526806  0.0555933960  0.0548357563
## [1976]  0.0540797589  0.0533254009  0.0525726793  0.0518215915  0.0510721345
## [1981]  0.0503243055  0.0495781018  0.0488335205  0.0480905587  0.0473492138
## [1986]  0.0466094828  0.0458713630  0.0451348516  0.0443999458  0.0436666429
## [1991]  0.0429349400  0.0422048343  0.0414763232  0.0407494038  0.0400240734
## [1996]  0.0393003292  0.0385781684  0.0378575884  0.0371385863  0.0364211595
## [2001]  0.0357053052  0.0349910207  0.0342783031  0.0335671499  0.0328575584
## [2006]  0.0321495257  0.0314430491  0.0307381261  0.0300347538  0.0293329297
## [2011]  0.0286326509  0.0279339148  0.0272367187  0.0265410600  0.0258469359
## [2016]  0.0251543439  0.0244632812  0.0237737451  0.0230857331  0.0223992424
## [2021]  0.0217142704  0.0210308145  0.0203488720  0.0196684404  0.0189895169
## [2026]  0.0183120989  0.0176361838  0.0169617690  0.0162888519  0.0156174299
## [2031]  0.0149475003  0.0142790606  0.0136121081  0.0129466404  0.0122826546
## [2036]  0.0116201484  0.0109591191  0.0102995641  0.0096414809  0.0089848669
## [2041]  0.0083297195  0.0076760362  0.0070238143  0.0063730515  0.0057237450
## [2046]  0.0050758924  0.0044294911  0.0037845386  0.0031410324  0.0024989699
## [2051]  0.0018583486  0.0012191660  0.0005814195 -0.0000548933 -0.0006897749
## [2056] -0.0013232280 -0.0019552548 -0.0025858580 -0.0032150401 -0.0038428035
## [2061] -0.0044691507 -0.0050940842 -0.0057176064 -0.0063397199 -0.0069604271
## [2066] -0.0075797305 -0.0081976325 -0.0088141357 -0.0094292423 -0.0100429549
## [2071] -0.0106552760 -0.0112662079 -0.0118757532 -0.0124839142 -0.0130906933
## [2076] -0.0136960931 -0.0143001159
```

```
x
```

```
##                x
## x_1 -5.197889e-07
## x_2  9.992604e-04
## x_3 -7.006098e-04
```

```
## x_4  1.997600e-04
## x_5  1.997100e-04
```

```
scale(x)
```

```
##               x
## x_1 -0.23086200
## x_2  1.41731946
## x_3 -1.38499114
## x_4  0.09930804
## x_5  0.09922564
## attr(,"scaled:center")
##            x
## 0.0001395202
## attr(,"scaled:scale")
##            x
## 0.0006065959
```

```
#lowest value
func(-0.23086200,1.41731946,-1.38499114, 0.09930804,0.09922564)
```

```
## [1] -25.30136
```

4.3 Based on your obeservations above, what are the important factors to improve convergence rate? (2 pts)

As displayed by above, adjusting the parameters for learning process (learning rate, epsilon) are the most important factors to improve convergence rate. Starting, guessed values are less important (even though guessing closer to the best value limits CPU time)